ΦΩΤΗΣ ΓΕΩΡΓΙΑΔΗΣ

AMSTRAD

ΧΙΛΙΕΣ ΚΑΙ ΜΙΑ ΔΥΝΑΤΟΤΗΤΕΣ

Β΄ ΕΚΔΟΣΗ

- ΕΤΟΙΜΕΣ ΡΟΥΤΙΝΕΣ ΚΩΔΙΚΑ ΜΗΧΑΝΗΣ
- ΡΟΥΤΙΝΑ ΑΝΤΙΓΡΑΦΗΣ
- RANDOM ACCESS
- ΟΙ ΚΛΗΣΕΙΣ ΤΟΥ ΛΕΙΤΟΥΡΓΙΚΟΥ

για CPC 464/664/6128

ΕΚΔΟΣΕΙΣ COMPUPRESS

1800 APX.

AMSTRAD:

ΧΙΛΙΕΣ ΚΑΙ ΜΙΑ ΔΥΝΑΤΟΤΗΤΕΣ

ΦΩΤΗΣ ΓΕΩΡΓΙΑΔΗΣ

ΕΚΔΟΤΙΚΟΣ ΟΡΓΑΝΙΣΜΟΣ COMPUPRESS

Λ. ΣΥΓΓΡΟΥ 44, 117 42 ΑΘΗΝΑ, ΤΗΛ: 9224845 - 9225520 - 9223768
ΧΑΛΚΕΩΝ 29, 546 31 ΘΕΣΣΑΛΟΝΙΚΗ, ΤΗΛ: 282663

Όλα τα αντίτυπα έχουν τιμή.

Α΄ ΕΚΔΟΣΗ 1986
Β΄ ΕΚΔΟΣΗ 1987
Λ. ΣΥΓΓΡΟΥ 44, 117 42 ΑΘΗΝΑ, ΤΗΛ: 9224845 - 9225520 - 9223768
ΧΑΛΚΕΩΝ 29, 546 31 ΘΕΣΣΑΛΟΝΙΚΗ, ΤΗΛ: 282663

AMSTRAD:
ΧΙΛΙΕΣ ΚΑΙ ΜΙΑ ΔΥΝΑΤΟΤΗΤΕΣ

Συγγραφέας: ΦΩΤΗΣ ΓΕΩΡΓΙΑΔΗΣ
Καλλιτεχνική επιμέλεια: ΙΩΑΝΝΑ ΜΑΛΕΣΗ
Επιμέλεια εξωφύλλου: ΕΚΤΩΡ ΧΑΡΑΛΑΜΠΟΥΣ
Φωτοστοιχειοθεσία: INTERFOT
Αναπαραγωγή ασπρόμαυρων/Μοντάζ: ΑΦΟΙ ΤΖΙΦΑ Ο.Ε.
Εκτύπωση: ΧΑΪΔΕΜΕΝΟΣ ΑΕΒΕ
Βιβλιοδεσία: Σ. ΓΚΟΥΝΤΑΡΕΛΛΗΣ

ΑΦΙΕΡΩΣΗ

Το βιβλίο αυτό αφιερώνω στην αγαπημένη μου Ντίνα, που χωρίς τη κατανόηση και συμπαράστασή της, δεν θα μπορούσε να γίνει πραγματικότητα.

ΠΕΡΙΕΧΟΜΕΝΑ

ΕΙΣΑΓΩΓΗ ΣΤΗ ΔΕΥΤΕΡΗ ΕΚΔΟΣΗ

Το Μάρτιο του 1986 που κυκλοφόρησε η πρώτη έκδοση του βιβλίου «AMSTRAD ΧΙΛΙΕΣ ΚΑΙ ΜΙΑ ΔΥΝΑΤΟΤΗΤΕΣ» δεν μπορούσαμε να φανταστούμε ότι θα οδηγούμαστε σε μια δεύτερη έκδοση τόσο γρήγορα. Σε αυτό βέβαια συνέβαλε τόσο η μεγάλη επιτυχία των μοντέλων CPC της Amstrad, όσο και το γεγονός ότι το βιβλίο απευθυνόμενο από τον αρχάριο μέχρι και τον πιο προχωρημένο user, προσφέρει πολλές χρήσιμες εφαρμογές. Ας δούμε όμως τι είναι οι «ΧΙΛΙΕΣ ΚΑΙ ΜΙΑ ΔΥΝΑΤΟΤΗΤΕΣ».

Όποιος ενδιαφέρεται να προγραμματίσει σε κώδικα μηχανής χρειάζεται να έχει ένα ρεπερτόριο από ρουτίνες που θα χρησιμοποιεί στα διάφορα προγράμματά του. Οι ρουτίνες αυτές θα πρέπει να κάνουν διάφορες στερεότυπες λειτουργίες όπως π.χ. να τυπώνουν χαρακτήρες στην οθόνη, να κάνουν input ένα string κλπ. Το να φτιαχτούν όμως αρχικά αυτές οι ρουτίνες απαιτεί αρκετό κόπο και γνώση του μηχανήματος. Από τον κόπο αυτό έχει σκοπό να σας απαλλάξει αυτό το βιβλίο. Στις επόμενες σελίδες μπορείτε να βρείτε ρουτίνες που αντιγράφουν προγράμματα από κασέτα ήδισκέτα, ρουτίνες που αναλύουν τον Header ενός προγράμματος, καθώς και μια ρουτίνα που προσθέτει εντολές για αρχεία τυχαίας προσπέλασης στο disc-drive του Amstrad.

Βέβαια δε χρειάζεται να ξέρετε κώδικα μηχανής για να χρησιμοποιήσετε τις ρουτίνες του βιβλίου. Μπορείτε να έχετε όλες τις ευκολίες που σας παρέχουν και από την Basic.

Επίσης όσοι θέλουν να προγραμματίσουν σε κώδικα μηχανής χρειάζονται και μια πλήρη ανάλυση του λειτουργικού συστήματος. Γι 'αυτόν το λόγο στο κεφάλαιο 8 αναλύονται οι κλήσεις του λειτουργικού της Amstrad, που ισχύουν και στα τρία μοντέλα (CPC 464, 664, 6128).

Δε μένει λοιπόν παρά να σας ευχηθώ καλό ταξίδι στον κόσμο του Amstrad και να αξιοποιήσετε με τη βοήθεια αυτού του βιβλίου, τις δυνατότητες του Amstrad στο έπακρο.

Φώτης Γεωργιάδης
Αθήνα 1987

1ο ΚΕΦΑΛΑΙΟ

Η ΠΡΩΤΗ ΕΠΑΦΗ

Οι ρουτίνες που παρουσιάζονται σ' αυτό το βιβλίο είναι έτοιμες για χρήση από οποιονδήποτε κάτοχο των μοντέλων CPC 464, 664 ή 6128 της Amstrad, είτε αυτός έχει γνώσεις κώδικα μηχανής είτε όχι. Εξάλλου δίνονται παράλληλα και μικρά προγράμματα σε BASIC που φορτώνουν τα αντίστοιχα bytes ώστε να μην χρειάζεται απαραίτητα assembler για να γραφτούν οι διάφορες ρουτίνες. Για τους πιο πεπειραμένους στη γλώσσα του Z80 - του μικροεπεξεργαστή που βρίσκεται στην καρδιά των Amstrad - έχουμε συμπεριλάβει στο τελευταίο κεφάλαιο και τις κλήσεις του λειτουργικού συστήματος με τις οποίες γίνονται οι διάφορες λειτουργίες στον υπολογιστή.

ΠΩΣ ΠΑΡΟΥΣΙΑΖΟΝΤΑΙ ΤΑ ΠΡΟΓΡΑΜΜΑΤΑ

Όλα τα προγράμματα που θα παρουσιάσουμε έχουν γραφτεί μέσω του Assembler - Dissassembler της HISOFT (HISOFT DEVPAC) και τα LISTINGS έχουν την ακόλουθη μορφή.

```
Hisoft GENA3.1 Assembler. Page      1.

Pass 1 errors: 00

9088                10           ORG   37000
9088  219290        20           LD    HL,BUFFER          ;4 bytes buffer needed
908B  019690        30           LD    BC,CTAB            ;command table
908E  CDD1BC        40           CALL  48337              ;put new commands
9091  C9            50           RET
9092                60 BUFFER:   DEFS  4                  ;4 bytes buffer
9096  A190          70 CTAB:     DEFW  CNAM               ;commands' names
9098  C3AC90        80           JP    OPEN               ;first new command "open"
909B  C3CA90        90           JP    PUT                ;command "put"
909E  C3DC90       100           JP    GET                ;command "get"
90A1  4F5045       110 CNAM:     DEFM  "OPE"              ;command names
90A4  CE           120           DEFB  206                ;asc("N")+128
90A5  5055         130           DEFM  "PU"
90A7  D4           140           DEFB  212                ;asc("T")+128
90A8  4745         150           DEFM  "GE"
90AA  D4           160           DEFB  212                ;asc("T")+128
90AB  00           170           DEFB  0                  ;terminate commands
90AC  DD5E00       180 OPEN:     LD    E,(IX+0)           ;command "open" routine
90AF  DD5601       190           LD    D,(IX+1)           ;record length in DE
90B2  0600         200           LD    B,0
90B4  D5           210           PUSH  DE
90B5  210002       220           LD    HL,512             ;512Kbytes in sector
90B8  ED52         230 LOOP1:    SBC   HL,DE              ;calculate number
90BA  3805         240           JR    C,OUT1             ;of records per sector
90BC  2803         250           JR    Z,OUT1
90BE  04           260           INC   B
90BF  18F7         270           JR    LOOP1
90C1  21ACA3       280 OUT1:     LD    HL,41900           ;save records per sectors
90C4  70           290           LD    (HL),B             ;in 41900
```

```
90C5  E1        300          POP  HL
90C6  22AEA3    310          LD   (41902),HL        ;save record length
90C9  C9        320          RET
90CA  CD4A91    330 PUT:     CALL COMMON            ;get string/record/sector
90CD  C5        340          PUSH BC
90CE  E5        350          PUSH HL
90CF  2AAEA3    360          LD   HL,(41902)       ;transfer string
90D2  E5        370          PUSH HL               ;in sector buffer
90D3  C1        380          POP  BC
90D4  E1        390          POP  HL
90D5  EDB0      400          LDIR
90D7  C1        410          POP  BC
90D8  CD2991    420          CALL WSECTOR          ;write sector to disc
90DB  C9        430          RET
90DC  CD4A91    440 GET:     CALL COMMON           ;get string/record/sector
90DF  E5        450          PUSH HL
90E0  2AAEA3    460          LD   HL,(41902)
90E3  E5        470          PUSH HL
90E4  C1        480          POP  BC
90E5  E1        490          POP  HL
90E6  2ADEA3    500          LD   HL,(41950)       ;put record from
90E9  EB        510          EX   DE,HL            ;sector buffer
90EA  EDB0      520          LDIR                  ;to string
90EC  C9        530          RET
90ED  2AACA3    540 FSECTO:  LD   HL,(41900)       ;calculate sector
90F0  EB        550          EX   DE,HL            ;and track number
90F1  010000    560          LD   BC,0             ;for a given record numbe
r
```

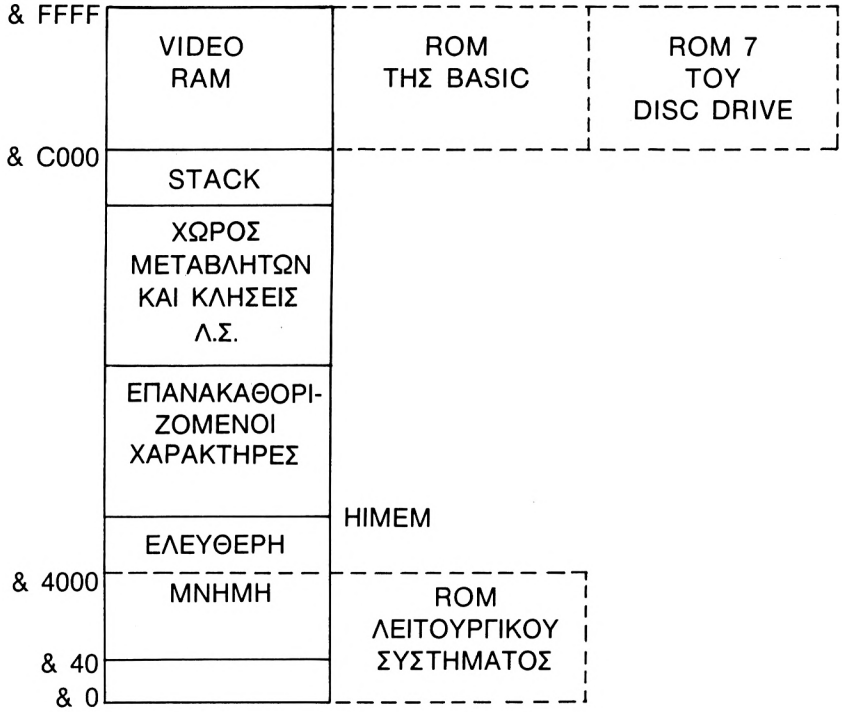
Για όσους διαθέτουν κάποιον διαφορετικό assembler δεν πιστεύουμε να υπάρξει κανένα πρόβλημα, αρκεί να λάβουν υπ' όψιν τους τις ιδιομορφίες που πιθανόν να έχει ο δικός τους assembler. Κάτω από κάθε LISTING σε assebly, θα σας δίνεται όπως είπαμε και ένα πρόγραμμα BASIC, που θα φορτώνει τα bytes με διαδοχικά POKES (για όσους δεν έχουν ASSEMBLER και θέλουν να χρησιμοποιήσουν τις ρουτίνες που υπάρχουν σ' αυτό το βιβλίο). Αυτά τα προγράμματα θα έχουν τη μορφή του παρακάτω LISTING.

```
5 MEMORY 36999
10 J=37000
20 RESTORE
30 READ A$
40 IF A$="TELOS" THEN END
50 FOR I=1 TO LEN(A$) STEP 2
60 POKE J,VAL("&"+MID$(A$,I,2))
70 J=J+1
80 NEXT
90 GOTO 30
100 DATA 21929001969ØCDD1BCC9ØØØØØØØØA19ØC3AC9ØC3CA9ØC3DC904F5045CE50
110 DATA 55D44745D4ØØDD5E ØØDD5601Ø6ØØD521ØØØ2ED523B052B03Ø418F721ACA3
120 DATA 7ØE122AEA3C9CD4A91C5E52AAEA3E5C1E1EDB0C1CD2991C9CD4A91E52AAE
130 DATA A3E5C1E12ADEA3EBEDB0C92AACA3EB01ØØØ3EC1ED5238ØC3CFECACCØ291
140 DATA 18F43EC104C94FC9C50E07CD0FB9CD00B93EC1CD39C02110A4C1C5501E00
150 DATA CD3CC0CD18B9CD03B9C1C9C50E07CD0FB9CD00B93EC1CD39C02110A41E00
160 DATA C1C550CD3FC0CD18B9CD03B9C1C9CD7B91DD6E02DD6603D5E5E5D1CDED90
170 DATA CD0891D1E1C5E52AACA3EBED5238Ø218FA19452AAEA3EB2110A41910FDEB
180 DATA E1C1C9DDE5DD5E00DD5601D5DDE1DD23DD5E00DD560121DEA3732372D5DD
190 DATA E12AAEA3E5C1212BA0EBEDB01128A0DDE1C9
200 DATA TELOS
```

Οσο είναι δυνατόν θα προσπαθήσουμε οι ρουτίνες που θα παρουσιαστούν να είναι «ανεξάρτητες θέσης». Με αυτό εννοούμε ότι θα μπορείτε να τις φορτώνετε σε όποια θέση μνήμης θέλετε, ανάλογα με το που χρειάζεται να τις χρησιμοποιήσετε. Αρκεί, μόνο να είστε μέσα στην διαθέσιμη ελεύθερη μνήμη του υπολογιστή.

Για να διευκολύνουμε κυρίως όσους ασχολούνται για πρώτη φορά με κώδικα μηχανής στον Amstrad θα δώσουμε σ' αυτό το σημείο λίγα στοιχεία σχετικά με τη μνήμη του υπολογιστή και τον τρόπο που έχει οργανωθεί.

ΟΡΓΑΝΩΣΗ ΤΗΣ ΜΝΗΜΗΣ



Απλοποιημένος χάρτης μνήμης για όλα τα μοντέλα της Amstrad.

Όπως βλέπετε και στο σχετικό πίνακα, η μνήμη RAM του Amstrad μπορεί να χωριστεί σε 5 γενικές περιοχές. Από τις θέσεις 0 μέχρι &40 (στο δεκαεξαδικό σύστημα) ή 64 στο δεκαδικό σύστημα, έχει αντιγραφεί ένα κομμάτι του λειτουργικού συστήματος που είναι απαραίτητο για τη λειτουργία του υπολογιστή. Αυτή η περιοχή δεν είναι διαθέσιμη στον χρήστη και δεν πρέπει να την χρησιμοποιείτε.

Στη συνέχεια έχουμε την διαθέσιμη μνήμη όπου αποθηκεύονται τα

προγράμματά μας σε BASIC καθώς και οι διάφορες μεταβλητές. Αυτό το κομμάτι της RAM εκτείνεται μέχρι την θέση του HIMEM που είναι για τον 464 43903 (&AB7F) και για τους 664 και 6128 42619 (&A67B). Στη συνέχεια υπάρχει ένας χώρος που καταλαμβάνεται από τους επανακαθοριζόμενους χαρακτήτες τους οποίους μπορεί να «σχεδιάσει» ο χρήστης με την εντολή SYMBOL, ο χώρος εργασίας και το JUMP-BLOCK του λειτουργικού συστήματος καθώς και το Stack. Στις περιοχές αυτές δεν συνιστάται να «πειράζετε» τίποτα, παρά μόνον αν είστε πεπειραμένος στον προγραμματισμό σε κώδικα μηχανής.

Τέλος στις διευθύνσεις &C000 - &FFFF βρίσκονται τα 16K της Video RAM που διαθέτει ο Amstrad. Εδώ δηλαδή αν κάνετε POKE θα εμφανιστεί κάτι αντίστοχο στην οθόνη του υπολογιστή.

Όπως είναι φυσικό τα προγράμματά μας σε κώδικα μηχανής τα τοποθετούμε σε κάποια θέση της διαθέσιμης μνήμης για το χρήστη. Για να εξασφαλίσουμε τον χώρο που θέλουμε, χρησιμοποιούμε την εντολή MEMORY που διαθέτει ο Amstrad, με την οποία περιορίζουμε τον χώρο για BASIC και μεταβλητές μέχρι μια ορισμένη θέση μνήμης. Από αυτή την θέση μπορεί να αρχίζει το πρόγραμμα σε κώδικα μηχανής που θέλουμε να χρησιμοποιήσουμε. Για παράδειγμα δίνοντας την εντολή

MEMORY 19999

η υψηλότερη θέση μνήμης για τα προγράμματα σε BASIC και τις μεταβλητές είναι η θέση 19999. Από την θέση 20000 και πάνω (μέχρι το άνω όριο της διαθέσιμης μνήμης) μπορούμε να έχουμε οτιδήποτε άλλο θέλουμε.

Κάτι που πρέπει να προσέξουμε εδώ είναι η χρήση της εντολής SYMBOL AFTER σε συνδυασμό με την MEMORY. Πρέπει να πληροφορήσετε τον υπολογιστή από ποιόν χαρακτήρα και μετά θα επανακαθορίσετε (δίνοντας SYMBOL AFTER), πριν χρησιμοποιήσετε την εντολή MEMORY. Αυτό γίνεται επειδή το λειτουργικό του υπολογιστή περιμένει να βρει την HIMEM στην συγκεκριμένη θέση, για να καθορίσει το χώρο που θα φυλαχτούν τα στοιχεία των επανακαθοριζομένων χαρακτήρων, οπότε αν την έχετε τοποθετήσει εσείς κάπου αλλού δεν τη βρίσκει, και θα σας επιστρέψει ένα σχετικό μήνυμα λάθους. Το ίδιο ακριβώς, και για τον ίδιο λόγο συμβαίνει και όταν έχετε κάποιο file ανοιχτό στην κασέτα (με τις εντολές OPENIN και OPENOUT).

Η ΕΝΤΟΛΗ CALL

Για να τρέξουμε κάποιο πρόγραμμα σε κώδικα μηχανής από την Basic, πρέπει να χρησιμοποιήσουμε την εντολή CALL. Στο λειτουργι-

κό σύστημα του Amstrad έχει δοθεί ιδιαίτερη προσοχή σ' αυτήν την εντολή, με αποτέλεσμα οι χρήστες να έχουν μεγάλη ευελιξία, κυρίως όσον αφορά μεταφορά μεταβλητών από την BASIC στον κώδικα μηχανής και αντίστροφα. Επειδή όμως στο manual του υπολογιστή δεν εξηγούνται καλά οι μεγάλες δυνατότητες του CALL, θα δώσουμε σ' αυτό το σημείο όλες τις απαραίτητες διευκρινίσεις, γιατί θα μας χρειαστούν αργότερα για να καλούμε τις ρουτίνες κώδικα μηχανής που θα παρουσιάσουμε.

Η πιο απλή χρήση της εντολής CALL είναι να καλέσουμε απ' ευθείας ένα πρόγραμμα σε κώδικα μηχανής να τρέξει, χωρίς να επιδράσουμε καθόλου σ' αυτό. Για παράδειγμα δίνοντας CALL &BD19 αρχίζει να εκτελείται η ρουτίνα στην θέση μνήμης &BD19 (δεκαεξαδικό) στην RAM του Amstrad. Παρόμοιες δυνατότητες υπάρχουν σε όλους τους home micros σήμερα. Όμως στον Amstrad έχουμε τη δυνατότητα να περάσουμε και τις δικές μας μεταβλητές στον κώδικα μηχανής δίνοντας π.χ.

CALL 42000, a%, b%, c%.

Με την παραπάνω εντολή πάλι αρχίζει να εκτελείται η ρουτίνα σε κώδικα μηχανής που υπάρχει στη θέση 42000 της RAM του υπολογιστή, παίρνοντας όμως και τις ακέραιες μεταβλητές a%, b%, c%. Χρησιμοποιήσαμε εδώ ακέραιες μεταβλητές γιατί αυτές κυρίως θέλουμε να περνάμε στον κώδικα μηχανής. Η πείρα έχει δείξει ότι πολύπλοκοι χειρισμοί πραγματικών μεταβλητών γίνονται πολύ πιο εύκολα από την BASIC. Αλλωστε, στη ROM υπάρχουν πολλές ρουτίνες που μας δίνουν μεγάλες ευκολίες για πολύπλοκους αριθμητικούς υπολογισμούς με πραγματικές μεταβλητές, μέσα από την BASIC.

Οι ακέραιες μεταβλητές που φαίνονται και στο παραπάνω παράδειγμα, είναι αριθμοί που έχουν τιμές από 0 ως 65535 (δεκάξι bit αριθμός που χωράει μέσα σε 2 bytes). Αν δεν έχει οριστεί η τιμή κάποιας ακέραιας μεταβλητής ο υπολογιστής θεωρεί ότι έχει την τιμή 0.

Φυσικά, αντί για μεταβλητές μπορούμε να δώσουμε κατ' ευθείαν αριθμούς. Π.χ., η εντολή

A%=500: CALL 42000, A%

είναι ισοδύναμη με την εντολή

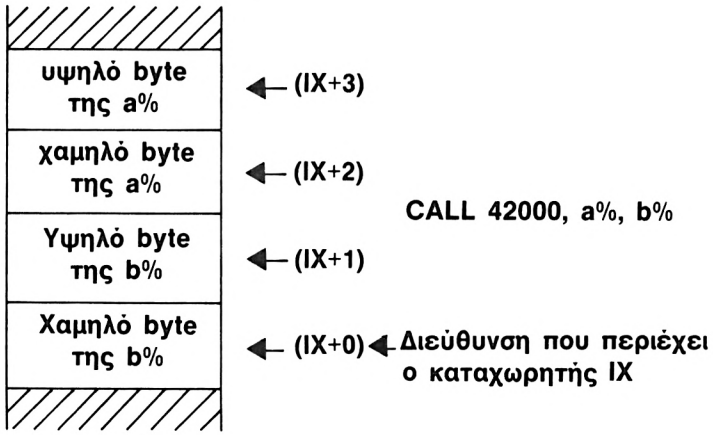
CALL 42000, 500.

Εξίσου εύκολα μπορούμε να περάσουμε ακέραιες μεταβλητές από τον κώδικα μηχανής προς την BASIC. Σ' αυτήν την περίπτωση περνάμε με την βοήθεια της εντολής CALL, την διεύθυνση της RAM όπου κρατείται η τιμή της μεταβλητής από το λειτουργικό σύστημα του υπολογιστή.

Για παράδειγμα δίνοντας
CALL 42000, @ A%
περνάμε στον κώδικα μηχανής, όχι την τιμή της μεταβλητής A% αλλά την διεύθυνση που είναι αποθηκευμένη η τιμή της στη RAM. Στη συνέχεια μέσα από την ρουτίνα κώδικα μηχανής που υπάρχει στη θέση 42000 μπορούμε να φορτώσουμε στην εν λόγω διεύθυνση ό,τι τιμή θέλουμε, οπότε όταν γυρίσουμε ξανά στη BASIC η τιμή αυτή θα είναι και η τιμή της μεταβλητής A%. Για να περάσουμε την θέση που βρίσκεται η τιμή της μεταβλητής A% πρέπει να βάλουμε το σύμβολο @ πριν την A%, στην εντολή CALL. Κάτι που χρειάζεται να προσέξουμε εδώ, είναι ότι η μεταβλητή A% πρέπει να έχει οριστεί πριν την χρησιμοποιήσουμε μέσα από την εντολή π.χ. CALL 42000, @ A% (δίνοντας A%=0) ώστε να έχει κρατηθεί γι' αυτήν μια αντίστοιχη θέση στο χώρο που κρατάει ο υπολογιστής τις μεταβλητές, την οποία θέση θα περάσουμε τελικά στον κώδικα μηχανής.

Επίσης, μπορούμε πολύ εύκολα να περάσουμε strings προς τον κώδικα μηχανής και πίσω στην BASIC δίνοντας εντολές του τύπου CALL 42000,@a$. Αλλά ας δούμε πρώτα πως ανταλλάσσονται οι ακέραιες μεταβλητές μεταξύ της BASIC και του κώδικα μηχανής.

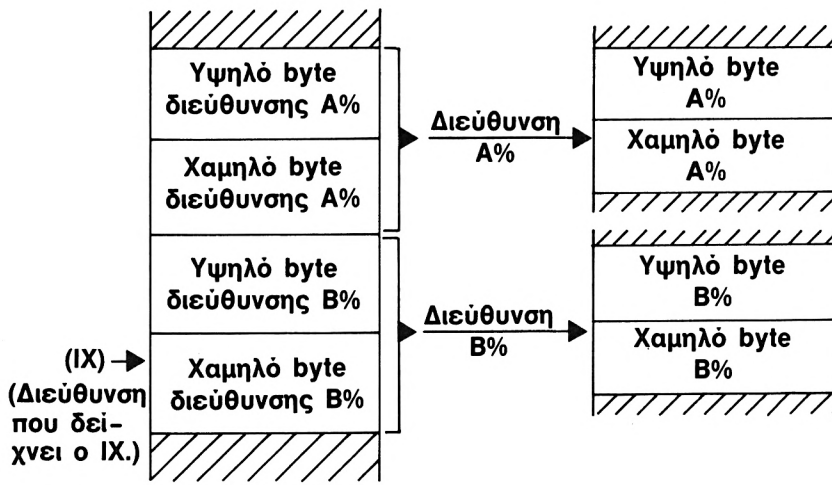
Ας θεωρήσουμε ότι δίνουμε την εντολή CALL 42000, α%, β%. Αρχίζει λοιπόν να εκτελείται ο κώδικας μηχανής ενώ ο καταχωρητής IX «δείχνει» τη διεύθυνση (περιέχει τη διεύθυνση) του block της RAM όπου έχει κρατηθεί η τιμή των δύο αυτών παραμέτρων. Αυτό φαίνεται πιο κατανοητά στο παρακάτω σχήμα:



Αν κοιτάξουμε λίγο πιο αναλυτικά δίνοντας π.χ. την εντολή CALL 42000, 5 θα έχουμε:

ΜΕΤΑΒΛΗΤΕΣ ΜΕ @

Όπως έχουμε ήδη αναφέρει δίνοντας το σύμβολο @ πριν το όνομα της μεταβλητής έχουμε τη δυνατότητα να μεταφέρουμε μεταβλητές από τον κώδικα μηχανής προς την BASIC. Αυτό που συμβαίνει στην περίπτωση των ακέραιων μεταβλητών είναι ότι αντί να μεταφέρεται η τιμή της μεταβλητής, μεταφέρεται η διεύθυνση της RAM όπου κρατείται η τιμή της μεταβλητής από το λειτουργικό του συστήματος. Δίνοντας για παράδειγμα CALL 42000, @ A%, @ B%, ο καταχωρητής IX δείχνει την διεύθυνση ενός block που περιέχει τις διευθύνσεις των δύο μεταβλητών. Μπορείτε να πάρετε μια παραστατικότερη ιδέα του τι γίνεται από το παρακάτω σχήμα:



Βλέπετε εδώ ότι ο IX δείχνει το χαμηλό byte της διεύθυνσης στην οποία κρατείται η τιμή της τελευταίας μεταβλητής, που εισάγεται με

την εντολή CALL. Σε κάθε μεταβλητή αντιστοιχεί μια διεύθυνση με 2 bytes (όπως φαίνεται και στο σχήμα). Στη συνέχεια, μετά το υψηλό byte της διεύθυνσης της τελευταίας μεταβλητής - που βρίσκεται στην αμέσως υψηλότερη θέση μνήμης από εκεί που δείχνει ο IX - ακολουθεί με τον ίδιο τρόπο η διεύθυνση της προ τελευταίας μεταβλητής και ούτω καθεξής.

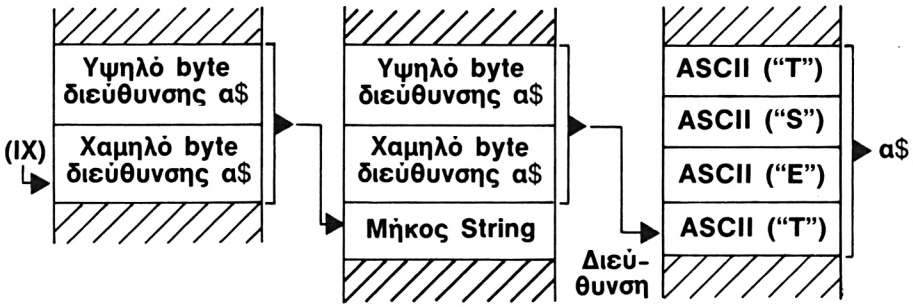
Και μην ξεχνάτε ότι οι μεταβλητές αυτές πρέπει να έχουν οριστεί προηγουμένως (π.χ. a%=0:b%=0) ώστε να αντιστοιχεί σ' αυτές μια διεύθυνση μνήμης για την τιμή τους. Διαφορετικά δεν μπορούμε να τις χρησιμοποιήσουμε με τον τρόπο που αναφέραμε παραπάνω. Οποιεσδήποτε πάντως απορίες σας μένουν, πρέπει να λυθούν μέσα από τις ρουτίνες που θα παρουσιάσουμε στη συνέχεια του βιβλίου αυτού.

Όσον αφορά τα strings η κατάσταση είναι λίγο πιο πολύπλοκη. Βέβαια ο τρόπος που περνάνε με την εντολή CALL στον κώδικα μηχανής, είναι παρόμοιος με αυτόν που περιγράψαμε προηγουμένως για τις ακέραιες μεταβλητές.

Για παράδειγμα μπορούμε να δώσουμε:

a$="TEST": CALL 42000, @a$

Η διαφορά όμως εδώ βρίσκεται στο ότι ο IX δεν δείχνει τη διεύθυνση που φυλάγεται η μεταβλητή από το λειτουργικό σύστημα, αλλά μια διεύθυνση όπου υπάρχουν χρήσιμες πληροφορίες σχετικά με το string. Συγκεκριμένα μπορείτε να πάρετε μια ιδέα του τι γίνεται σ' αυτήν την περίπτωση από το παρακάτω σχήμα.



Βλέπουμε εδώ ότι ο IX «δείχνει» μια διεύθυνση στη RAM όπου υπάρχει η διεύθυνση (σε δύο bytes) που υπάρχουν πληροφορίες σχετικά με το a$. Στην πρώτη θέση αυτής της διεύθυνσης υπάρχει (σε 1 byte) το μήκος του string που μπορεί να είναι από 0 μέχρι 255 χαρακτήρες. Στις αμέσως επόμενες δύο θέσεις της RAM υπάρχει (σε δύο bytes) η διεύθυνση μνήμης που βρίσκεται ο πρώτος χαρακτήρας που περιέχει το string (όπως φαίνεται και στον παραπάνω πίνακα). Όσοι έχουν απορίες που δεν τους λύνονται με τις εξηγήσεις που έχουμε

δώσει μέχρι τώρα, ελπίζουμε να τις λύσουν βλέποντας και αναλύοντας τα προγράμματα σε κώδικα μηχανής που δίνουμε στη συνέχεια του βιβλίου, όπου εφαρμόζονται στην πράξη όλα όσα είπαμε παραπάνω.

Μπορεί - πολύ σωστά - να σκεφτήκατε ότι έχουμε τη δυνατότητα να περάσουμε μεταβλητές προς τον κώδικα μηχανής κάνοντας POKE, και να πάρουμε τα αποτελέσματα στην BASIC με αντίστοιχα PEEK. Όμως η χρήση μεταβλητών μέσα από την εντολή CALL είναι πολύ πιο ευέλικτη, πολύ πιο γρήγορη, και φυσικά κάνει και το πρόγραμμά μας πολύ πιο κομψό.

ΚΩΔΙΚΑΣ ΜΗΧΑΝΗΣ ΚΑΙ ΛΕΙΤΟΥΡΓΙΚΟ ΣΥΣΤΗΜΑ

Όπως είδατε και στον χάρτη της μνήμης που παρουσιάσαμε στην αρχή του κεφαλαίου, το λειτουργικό σύστημα του Amstrad βρίσκεται «παράλληλα» με τα 16 πρώτα Kbytes της RAM, ενώ η ROM της LOCOMOTIVE BASIC και το λειτουργικό σύστημα του DISK DRIVE βρίσκονται παράλληλα στα 16K της Video-Ram (2 ROM των 16K).

Αυτό όμως που είναι πολύ χρήσιμο στον Amstrad και πραγματικά «λύνει τα χέρια» σε όποιον θέλει να προγραμματίσει σε κώδικα μηχανής είναι ένα JUMPBLOCK που υπάρχει στην RAM, δηλαδή ένας πίνακας για κλήσεις του λειτουργικού συστήματος. Μέσω αυτού (κάνοντας τα αντίστοιχα CALL) μπορούμε να κάνουμε όλες τις λειτουργίες που γίνονται στην BASIC μέσα από κώδικα μηχανής. Έτσι για παράδειγμα όταν θέλουμε να τυπώσουμε κάποιο χαρακτήρα στην οθόνη μέσα από κώδικα μηχανής, αντί να φτιάξουμε ολόκληρη ρουτίνα που να κάνει αυτή τη δουλειά, κάνουμε CALL στη διεύθυνση #BB5D (που βρίσκεται μέσα στο JUMPBLOCK) έχοντας απλά στον καταχωρητή A τον κωδικό του χαρακτήρα που θέλουμε να τυπώσουμε. Όλες όμως οι πληροφορίες σχετικά με τα CALLS που μπορούμε να κάνουμε, τις συνθήκες που πρέπει να πληρούνται, και τα αποτελέσματα που θα έχουμε περιγράφονται στο τελευταίο κεφάλαιο αυτού του βιβλίου (Κεφ. 8), το οποίο συνιστούμε να χρησιμοποιείτε για αναφορά σας όταν γράφετε κάποιο πρόγραμμα σε κώδικα μηχανής. Αν είστε πολύ βιαστικοί, και θέλετε να δείτε τι δυνατότητες έχετε μπορείτε να ρίξετε μια ματιά από τώρα, πριν περάσετε στο επόμενο κεφάλαιο. Βέβαια πρέπει να σημειώσουμε ότι περισσότερες πληροφορίες γι' αυτά τα θέματα, μπορείτε να βρείτε στο firmware manual του υπολογιστή.

2ο ΚΕΦΑΛΑΙΟ

ΧΕΙΡΙΣΜΟΣ ΠΛΗΚΤΡΟΛΟΓΙΟΥ ΑΠΟ ΚΩΔΙΚΑ ΜΗΧΑΝΗΣ

Όταν γράφουμε προγράμματα σε κώδικα μηχανής θέλουμε φυσικά να χειριζόμαστε το πληκτρολόγιο ώστε να μπορεί να υπάρχει επικοινωνία με τον χρήστη όταν τρέχει κάποιο πρόγραμμα. Μας ενδιαφέρει συνεπώς να ξέρουμε αν κάποιο πλήκτρο πατήθηκε και ποιό, καθώς και άλλες σχετικές πληροφορίες. Αν κοιτάξετε στο 8ο κεφάλαιο (προς την αρχή) θα δείτε ότι υπάρχουν περί τις 25 κλήσεις που μπορούμε να κάνουμε στο λειτουργικό σύστημα για διάφορους χειρισμούς του πληκτρολογίου.

Ας δούμε κάτι που μπορούμε να χρησιμοποιήσουμε κατ' ευθείαν από την BASIC. Δίνοντας CALL &BB06 ο υπολογιστής περιμένει να πατηθεί κάποιο πλήκτρο (το οποίο θα επιστρέψει στον καταχωρητή A) για να συνεχίσει την εκτέλεση του προγράμματός μας. Αντί λοιπόν σε κάποιο σημείο ενός προγράμματος να δώσουμε εντολές της μορφής WHILE INKEY$=" ": WEND ή 100 IF INKEY$=" " THEN 100 περιμένοντας να πατήσει ο χρήστης κάποιο πλήκτρο για να συνεχίσει το πρόγραμμα να τρέχει, μπορούμε να δώσουμε CALL &BB06. Βέβαια αυτό προϋποθέτει ότι δεν μας ενδιαφέρει ποιο πλήκτρο πατήθηκε, απλώς περιμένουμε μέχρι να πατηθεί κάποιο.

Στη συνέχεια θα σας δώσουμε μερικές ρουτίνες σε κώδικα μηχανής που χρησιμοποιούν τις ρουτίνες χειρισμού του πληκτρολογίου.

ΑΧΡΗΣΤΕΥΣΗ ΤΟΥ RESET

Η παρακάτω ρουτίνα «αχρηστεύει» το RESET που γίνεται στον Amstrad πατώντας ταυτόχρονα τα πλήκτρα CTRL-SHIFT-ESC. Όταν πατηθούν αυτά τα πλήκτρα αντί του γνωστού μας RESET, απλώς εμφανίζεται το μήνυμα BREAK. Επίσης το ESC αχρηστεύεται όταν τρέχουμε κάποιο πρόγραμμα. Έτσι λοιπόν σας παρουσιάζεται ένας πιο ασφαλής τρόπος να κλειδώνετε τα προγράμματά σας από τα ξένα μάτια, και παίρνετε ταυτόχρονα μια ιδέα πώς σε πολλά απ' τα παιχνίδια που κυκλοφορούν στο εμπόριο για τον υπολογιστή δεν γίνεται RESET (με αποτέλεσμα να πρέπει κάθε φορά να κλείνουμε τον υπολογιστή για να τα σβήσουμε).

Σημειώνουμε επίσης ότι αυτή η ρουτίνα μπορεί να φορτωθεί σε

οποιαδήποτε θέση μνήμης και να λειτουργήσει κανονικά, αρκεί βέβαια να κάνουμε CALL στην αντίστοιχη αρχική διεύθυνση.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ ΑΠΟ BASIC: CALL διεύθυνση, A% όπου η A% έχει τιμή 0 για αχρήστευση του RESET ή 1 για ενεργοποίησή του και πάλι.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ ΓΙΑ Μ/C: Ο καταχωρητής IX δείχνει σε μια διεύθυνση μνήμης που περιέχει την τιμή 0 ή 1 και ο A περιέχει 1.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ ΓΙΑ Μ/C: Ο AF έχει τυχαίο περιεχόμενο.

```
Hisoft GENA3.1 Assembler. Page      1.

Pass 1 errors: 00

A410                10          org  42000
A410  FE01          20          CP   1                    ;one parameter
A412  C0            30          RET  NZ
A413  DD7E00        40          LD   A,(IX)               ;choice
A416  FE00          50          CP   0                    ;disable reset?
A418  2806          60          JR   Z,DISAB
A41A  3EC3          70          LD   A,195                ;enable reset
A41C  32EEBD        80          LD   (48622),A
A41F  C9            90          RET
A420  3EC9         100 DISAB:   LD   A,201                ;disable reset
A422  32EEBD       110          LD   (48622),A
A425  C9           120          RET

Pass 2 errors: 00

Table used:     25  from    124
```

```
10 J=42000
20 RESTORE
30 READ A$
40 IF A$="TELOS" THEN END
50 FOR I=1 TO LEN(A$) STEP 2
60 POKE J,VAL("&"+MID$(A$,I,2))
70 J=J+1
80 NEXT
90 GOTO 30
100 DATA FE01C0DD7E00FE002806 3EC332EEBDC93EC932EEBDC9
110 DATA TELOS
```

Οπως βλέπετε ο κώδικας μηχανής σώζεται από την BASIC με SAVE "DRESET", b, 42000, 25 (μήκος 25 bytes). Μπορείτε να τη φορτώσετε όποτε θέλετε δίνοντας MEMORY 41999: LOAD "DRESET" και κάνοντας CALL 42000, 0 αχρηστεύετε το RESET όπως είπαμε.

Οπως φαίνεται στο listing σε ASSEMBLY το RESET εξαρτάται απ' το περιεχόμενο της διεύθυνσης &BDEE της RAM. Αν υπάρχει 195 γίνεται κανονικά ενώ αν υπάρχει 201 αχρηστεύεται.

ΑΝΑΓΝΩΣΗ ΧΑΡΑΚΤΗΡΑ

Η ρουτίνα που παρουσιάζουμε εδώ μεταφέρει σε μια μεταβλητή

μας, τον κωδικό ASCII του χαρακτήρα που παράγεται κατά το πάτημα ενός πλήκτρου του πληκτρολογίου. Το πρόγραμμα ταυτόχρονα σταματάει, περιμένοντας να πατηθεί κάποιο πλήκτρο από τον χρήστη.

Αυτή η λειτουργία εκτός του να σας δείξει πως μπορείτε να διαβάζεται το πληκτρολόγιο από κώδικα μηχανής είναι πολύ λειτουργική και για την BASIC αφού πρέπει να κάνουμε ολόκληρη διαδικασία μέχρι να βάλουμε τον ASCII ενός πλήκτρου που πατήθηκε, σε κάποια μεταβλητή.

Η ρουτίνα καλείται με την εντολή CALL 42000,@A% όπου A% είναι μια μεταβλητή που πρέπει προηγουμένως να έχουμε ορίσει. Ο κωδικός ASCII του πλήκτρου που πατήθηκε μπαίνει ακριβώς σαν τιμή σ'αυτή την μεταβλητή.

```
Hisoft GENA3.1 Assembler. Page      1.

Pass 1 errors: 00

A410                10          org  42000
A410  FE01          20          CP   1                    ;one parameter
A412  C0            30          RET  NZ
A413  DD6E00        40          LD   L,(IX+0)
A416  DD6601        50          LD   H,(IX+1)             ;address of A%
A419  CD18BB        60          CALL 47896
A41C  77            70          LD   (HL),A
A41D  AF            80          XOR  A                    ;clear A
A41E  23            90          INC  HL
A41F  3600         100          LD   (HL),0
A421  C9           110          RET

Pass 2 errors: 00

Table used:     13  from    119


10 J=42000
20 RESTORE
30 READ A$
40 IF A$="TELOS" THEN END
50 FOR I=1 TO LEN(A$) STEP 2
60 POKE J,VAL("&"+MID$(A$,I,2))
70 J=J+1
80 NEXT
90 GOTO 30
100 DATA FE01C0DD6E00DD6601CD18BB77AF233600C9
110 DATA TELOS
```

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ ΑΠΟ BASIC: CALL 42000, @ A% όπως περιγράψαμε προηγουμένως.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ ΑΠΟ M/C: Ο IX «δείχνει» τη διεύθυνση που φυλάσσεται η τιμή της μεταβλητής και ο A έχει την τιμή 1.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ (M/C): Οι AF και HL έχουν τυχαίο περιεχόμενο.

Προσέξτε εδώ τη χρήση του συμβόλου @ που περιγράψαμε στο πρώτο κεφάλαιο του παρόντος βιβλίου. Για χρήση της ρουτίνας μέσα από κώδικα μηχανής, περισσότερες πληροφορίες για το πως δουλεύει η &BB18 μπορείτε να βρείτε στο κεφάλαιο 8 που παρουσιάζουμε τις

ρουτίνες του λειτουργικού συστήματος. Οι τιμές ASCII που αντιστοιχούν σε κάθε πλήκτρο, δίνονται μέσα στο manual του υπολογιστή.

BUFFER ΠΛΗΚΤΡΟΛΟΓΙΟΥ

Το λειτουργικό σύστημα του Amstrad διαθέτει έναν BUFFER (κενό χώρο για προσωρινή αποθήκευση) ειδικά για το πληκτρολόγιο. Εκεί αποθηκεύονται οι χαρακτήρες που πληκτρολογούμε (κάθε φορά που γίνεται έλεγχος του πληκτρολογίου, δηλαδή 50 φορές το δευτερόλεπτο) μέχρι να ζητηθούν και να χρησιμοποιηθούν π.χ. από μία εντολή INPUT. Τη στιγμή που εκτελείται αυτή η εντολή αδειάζει ο BUFFER του πληκτρολογίου στο INPUT. Αυτό γίνεται εύκολα και από κώδικα μηχανής με τη ρουτίνα που παρουσιάζουμε παρακάτω. Επειδή δε χρειαζόμαστε κάτι τέτοιο όταν δουλεύουμε BASIC (αφού το λειτουργικό σύστημα φροντίζει από μόνο του) παραθέτουμε μόνο το LISTING σε ASSEMBLY.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ Μ/C: Ο HL περιέχει τη διεύθυνση μνήμης που θα αποθηκευτούν οι χαρακτήρες.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο HL και ο A έχουν τυχαίο περιεχόμενο και τα flags έχουν επηρεαστεί.

```
Hisoft GENA3.1 Assembler. Page      1.

Pass 1 errors: 00

A410                10          org  42000
A410  CD09BB        20 LOOP:    CALL 47881          ;get chr
A413  D0            30          RET  NC             ;ret if no more
A414  77            40          LD   (HL),A         ;save chr
A415  23            50          INC  HL
A416  18F8          60          JR   LOOP           ;repeat

Pass 2 errors: 00

Table used:     24  from    113
```

Όταν χρησιμοποιούμε τη ρουτίνα ανάγνωσης χαρακτήρα που παρουσιάσαμε προτύτερα, πρέπει ο BUFFER του πληκτρολογίου να είναι άδειος. Αλλιώς υπάρχει περίπτωση να πάρουμε κάποιον άλλον χαρακτήρα που έχουμε πληκτρολογήσει πιο πριν (ίσως και κατά λάθος) και ο οποίος έχει αποθηκευτεί στον BUFFER. Ακριβώς αυτή τη δουλειά κάνει η ρουτίνα που παρουσιάζουμε παρακάτω.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ BASIC-Μ/C: CALL 42000 για να καθαρίσει ο Buffer.

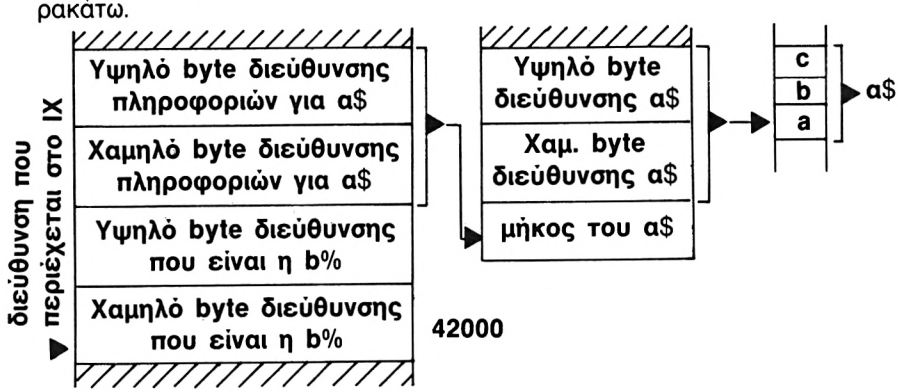
ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ Μ/C: Ο AF έχει τυχαίο περιεχόμενο.

```
Hisoft GENA3.1 Assembler. Page      1.

Pass 1 errors: 00

A410                ~10         org  42000
A410  CD09BB         20 LOOP:   CALL 47881
A413  38FB           30         JR   C,LOOP
A415  C9             40         RET

Pass 2 errors: 00

Table used:     24  from   106
```

```
10 J=42000
20 RESTORE
30 READ A$
40 IF A$="TELOS" THEN END
50 FOR I=1 TO LEN(A$) STEP 2
60 POKE J,VAL("&"+MID$(A$,I,2))
70 J=J+1
75 END
80 NEXT
90 GOTO 30
100 DATA CD09BB38FBC9
110 TELOS
```

ΕΛΕΓΧΟΣ ΠΛΗΚΤΡΩΝ

Η ρουτίνα που παρουσιάζουμε παρακάτω κάνει έλεγχο αν πατήθηκε κάποιο ή κάποια συγκεκριμένα πλήκτρα. Ενεργοποιείται από BASIC με CALL χρησιμοποιώντας δύο μεταβλητές που έχουν μπροστά τους το σύμβολο @. Η πρώτη είναι το string a$ και η δεύτερη είναι η ακέραια μεταβλητή b%. Υπενθυμίζουμε ξανά ότι αν καλούμε κώδικα μηχανής χρησιμοποιώντας το @, πρέπει προηγουμένως να έχουμε ορίσει τις μεταβλητές ώστε να έχει κρατηθεί γι' αυτές ο ανάλογος χώρος στη μνήμη του υπολογιστή. Δίνοντας στην παρακάτω ρουτίνα CALL 42000, @ a$, @b% μεταφέρουμε στον κώδικα μηχανής τις θέσεις μνήμης που είναι αποθηκευμένες οι δύο αυτές μεταβλητές. Συγκεκριμένα ο IX «δείχνει» σε ένα block μνήμης όπως φαίνεται παρακάτω.



Η ρουτίνα μπορεί να φορτωθεί σε οποιαδήποτε θέση της μνήμης, αρκεί βέβαια να κάνουμε CALL στη διεύθυνση αυτή.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ BASIC: CALL 42000, @ a$, @ b%. Το a$ περιέχει τους χαρακτήρες των πλήκτρων που θα ελέγξουμε αν έχουν πατηθεί και το b% πρέπει να οριστεί προηγουμένως (π.χ. b%=0).

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ ΑΠΟ Μ/C: Πρέπει ο IX να περιέχει μια διεύθυνση που να δείχνει ένα block όπως περιγράψαμε προηγουμένως και ο A να περιέχει 2.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ BASIC: Ο b% περιέχει έναν αριθμό ανάλογα με το ποιός απ'τους χαρακτήρες του a$ έχει πατηθεί. (1 για τον πρώτο, 2 για τον δεύτερο κ.ο.κ.).

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ Μ/C: Όλοι οι καταχωρητές έχουν τυχαίο περιεχόμενο. Άμεση έξοδος απ'τη ρουτίνα αν δεν έχουν τηρηθεί οι συνθήκες εισόδου για 2 μεταβλητές, (ή αν ο καταχωρητής A δεν περιέχει τον αρ. 2).

```
Hisoft GENA3.1 Assembler. Page      1.

Pass 1 errors: 00

A410                10          org  42000
A410  FE02          20          CP   2              ;two parameters
A412  C0            30          RET  NZ             ;ret if not two
A413  DD6E00        40          LD   L,(IX+0)
A416  DD6601        50          LD   H,(IX+1)
A419  E5            60          PUSH HL             ;save address of b%
A41A  DD6E02        70          LD   L,(IX+2)
A41D  DD6603        80          LD   H,(IX+3)
A420  7E            90          LD   A,(HL)         ;get length of a$
A421  23           100          INC  HL
A422  4E           110          LD   C,(HL)         ;get address of a$
A423  23           120          INC  HL
A424  46           130          LD   B,(HL)         ;in BC
A425  C5           140          PUSH BC
A426  E1           150          POP  HL             ;a$ address in HL
A427  324CA4       160          LD   (STO1),A       ;save len of a$
A42A  224DA4       170          LD   (STO2),HL      ;save address of HL
A42D  3A4CA4       180 LOOP1:   LD   A,(STO1)       ;get len of a$
A430  47           190          LD   B,A            ;into B
A431  2A4DA4       200          LD   HL,(STO2)      ;recover address of a$
A434  C5           210          PUSH BC
A435  E5           220          PUSH HL
A436  CD18BB       230          CALL 47896          ;wait character
A439  E1           240          POP  HL
A43A  C1           250          POP  BC
A43B  1E01         260          LD   E,1            ;initialize counter
A43D  BE           270 LOOP:    CP   (HL)           ;compare chr
A43E  2806         280          JR   Z,FOUND
A440  1C           290          INC  E              ;increase counter
A441  23           300          INC  HL             ;next character
A442  10F9         310          DJNZ LOOP           ;repeat
```

```
A444  18E7        320          JR   LOOP1
A446  E1          330 FOUND:   POP  HL                ;recover b% address
A447  73          340          LD   (HL),E          ;put value in it
A448  23          350          INC  HL
A449  3600        360          LD   (HL),0
A44B  C9          370          RET
A44C  00          380 STO1:    DEFB 0
A44D              390 STO2:    DEFS 2

Pass 2 errors: 00

Table used:     70. from    194
```

```
10 J=42000
20 RESTORE
30 READ A$
40 IF A$="TELOS" THEN END
50 FOR I=1 TO LEN(A$) STEP 2
60 POKE J,VAL("&"+MID$(A$,I,2))
70 J=J+1
80 NEXT
90 GOTO 30
100 DATA FE02C0DD6E00DD6601E5DD6E02DD66037E234E2346C5E1324CA4224DA43A
110 DATA 4CA4472A4DA4C5E5CD1BBBE1C11E01BE2B061C2310F918E7E173233600C9
120 DATA 000000
130 DATA TELOS
```

Η παραπάνω ρουτίνα μπορεί να σας φανεί χρήσιμη σε πολλές περιπτώσεις. Για παράδειγμα να θέλουμε να πάρουμε απαντήσεις του στυλ Ν/Ο, δηλαδή αν έχουν πατηθεί τα Ν, n ή Ο, ο αντίστοιχα σε κάποιο σημείο του προγράμματός μας μπορούμε να δώσουμε τις εξής εντολές από BASIC

A$="NnOo": b%=0: CALL 42000,@A$,@b%

Η ρουτίνα θα επιστρέψει στη BASIC μόνον εφόσον πατηθεί κάποιο πλήκτρο απ' αυτά που περιμένουμε. Στην παραπάνω περίπτωση ανάλογα αν πατήθηκε Ν, n, Ο ή ο, η b% θα πάρει την τιμή 1, 2, 3 ή 4 αντίστοιχα.

```
Hisoft GENA3.1 Assembler. Page      2.

A47C  3E20        560          LD   A,32
A47E  CD5ABB      570          CALL 47962
A481  3E08        580          LD   A,8
A483  CD5ABB      590          CALL 47962
A486  C1          610          POP  BC
A487  04          620          INC  B
A488  04          630          INC  B
A489  DD2B        640          DEC  IX
A48B  DD360020    650          LD   (IX),32          ;set to space
A48F  18C9        660          JR   LOOP2            ;back
A491  C1          670 NEXE:    POP  BC
A492  04          680          INC  B
A493  04          690          INC  B
A494  18C4        700          JR   LOOP2
A496              710 TEMIX:   DEFS 2
A498  00          720 STO:     DEFB 0

Pass 2 errors: 00

Table used:    132  from    236
```

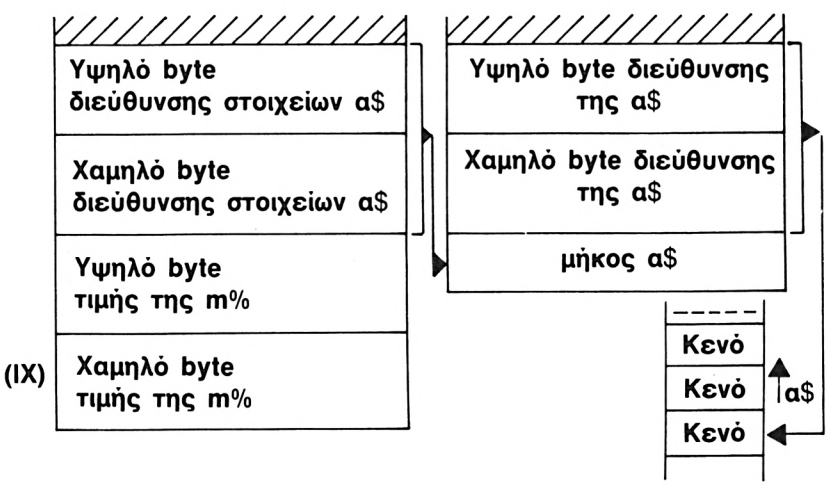
```
10 J=42000
20 RESTORE
30 READ A$
40 IF A$="TELOS" THEN END
```

```
50 FOR I=1 TO LEN(A$) STEP 2
60 POKE J,VAL("&"+MID$(A$,I,2))
70 J=J+1
80 NEXT
90 GOTO 30
100 DATA FE02C0DD7E00DD6E02DD6603BE7E3298A43801C9234E2346C5DD7E0047DD
110 DATA E1DD2296A4DDE5C53A98A447DD360020DD2310F8C1DDE1C5CD18BBFE7F28
120 DATA 18FE0D280DDD7700CD5ABBDD23C110E91801C13E07CD5ABBC92A96A4AFDD
130 DATA E5D1ED522007 3E07CD5ABB181A3E08CD5ABB3E20CD5ABB3E08CD5ABBC104
140 DATA 04DD2BDD36002018C9C1040418C4000000C9000000
150 DATA TELOS
```

Η επόμενη ρουτίνα που σας παρουσιάζουμε υποκαθιστά ακριβώς τη λειτουργία INPUT τη BASIC και είναι ιδιαίτερα χρήσιμη όταν προγραμματίζουμε σε κώδικα μηχανής.

Καλώντας αυτή τη ρουτίνα από BASIC πρέπει να περάσουμε 2 μεταβλητές, την A$ με το σύμβολο @ και την m%. Στην A$ θα μπει το string που θα γίνει INPUT, ενώ με την m% δίνουμε το maximum μήκος της A$. Πρέπει προηγουμένως να έχουμε ορίσει την A$ με μήκος τουλάχιστον όσο το m% ώστε να υπάρχει ο ανάλογος χώρος στη μνήμη. Αυτό μπορεί να γίνει δίνοντας π.χ. A$=SPACE $ (m%) που γεμίζει το A$ με m% κενά.

Όταν αρχίσει να εκτελείται ο κώδικας μηχανής με CALL 42000, @ A$, m% ο IX πρέπει να περιέχει την διεύθυνση μνήμης με την αντιστοιχία που φαίνεται στον παρακάτω πίνακα:



ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ BASIC: CALL 42000, @ A$, m%. Η A$ είναι το string όπου θα μπουν οι χαρακτήρες του INPUT. Η m% δείχνει το μέγιστο αριθμό χαρακτήρων που είναι δεκτοί. Η εισαγωγή χαρακτήρων τελειώνει πατώντας το ENTER.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ Μ/C: Ο IX πρέπει να δείχνει ένα μπλοκ μνήμης όπως φαίνεται παραπάνω και ο A να περιέχει τον αριθμό 2.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ BASIC: Στην A$ βρίσκονται οι χαρακτήρες που έχουμε εισάγει.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ Μ/C: Όλοι οι καταχωρητές έχουν τυχαίο περιεχόμενο. Από τη ρουτίνα βγαίνουμε αμέσως αν δεν έχουμε περάσει ακριβώς 2 παραμέτρους, ή αν ο A δεν έχει τον αριθμό 2 (βλέπε συνθήκες εισόδου).

```
NN
Hisoft GENA3.1 Assembler. Page      1.

Pass 1 errors: 00

A410                10          org  42000
A410  FE02          20          CP   2                ;two parameters
A412  C0            30          RET  NZ               ;ret if not two
A413  DD7E00        40          LD   A,(IX+0)
A416  DD6E02        50          LD   L,(IX+2)
A419  DD6603        60          LD   H,(IX+3)
A41C  BE            70          CP   (HL)
A41D  7E            80          LD   A,(HL)
A41E  3298A4        85          LD   (STO),A
A421  3801          90          JR   C,OK            ;if len(a$)>m% OK
A423  C9           100          RET                  ;else RETURN
A424  23           110 OK:      INC  HL
A425  4E           120          LD   C,(HL)          ;get address of a$
A426  23           130          INC  HL
A427  46           140          LD   B,(HL)          ;into BC
A428  C5           150          PUSH BC
A429  DD7E00       160          LD   A,(IX+0)
A42C  47           170          LD   B,A
A42D  DDE1         180          POP  IX              ;address of a$ in IX
A42F  DD2296A4     190          LD   (TEMIX),IX
A433  DDE5         200          PUSH IX
A435  C5           210          PUSH BC
A436  3A98A4       220          LD   A,(STO)
A439  47           230          LD   B,A
A43A  DD360020     240 CLEAR:   LD   (IX),32
A43E  DD23         250          INC  IX
A440  10F8         260          DJNZ CLEAR           ;clear string
A442  C1           270          POP  BC
A443  DDE1         280          POP  IX
A445  C5           290 LOOP1:   PUSH BC
A446  CD18BB       300          CALL 47896           ;wait for character
A449  FE7F         310          CP   127
A44B  2818         320          JR   Z,DEL
A44D  FE0D         330          CP   13
A44F  280D         340          JR   Z,FIN           ;if ENTER completed
A451  DD7700       350          LD   (IX),A          ;put chr in a$
A454  CD5ABB       360          CALL 47962           ;also print it
A457  DD23         370          INC  IX              ;next location to fill
A459  C1           380          POP  BC              ;is m% exceeded?



A45A  10E9         390 LOOP2:   DJNZ LOOP1
A45C  1801         400          JR   FIN2
A45E  C1           410 FIN:     POP  BC
```

```
A45F  3E07      420 FIN2:  LD   A,7
A461  CD5ABB    430        CALL 47962          ;a humoristic beep
A464  C9        440        RET
A465  2A96A4    450 DEL:   LD   HL,(TEMIX)     ;if DELETE was pressed
A468  AF        460        XOR  A
A469  DDE5      470        PUSH IX
A46B  D1        480        POP  DE
A46C  ED52      490        SBC  HL,DE
A46E  2007      500        JR   NZ,EXE
A470  3E07      510        LD   A,7
A472  CD5ABB    520        CALL 47962
A475  181A      530        JR   NEXE
A477  3E08      540 EXE:   LD   A,8
A479  CD5ABB    550        CALL 47962
```

Σημειώνουμε ότι αν εισάγοντας χαρακτήρες φτάσουμε το μήκος που δίνουμε στο m% βγαίνουμε αυτόματα από τη ρουτίνα. Επίσης δουλεύει το DELETE και ταυτόχρονα η ρουτίνα δεν μας επιτρέπει να κάνουμε DELETE στην αρχή του A$.

Για παράδειγμα μπορείτε να τρέξετε το ακόλουθο πρόγραμμα σε BASIC σε συνεργασία με τη ρουτίνα του κώδικα μηχανής.

```
10 MODE 2
20 m%=20
30 A$=SPACE$ (20)
40 CALL 42000,@A$,m%
50 PRINT: PRINT A$
60 END
```

Περισσότερες ρουτίνες που να χειρίζονται το πληκτρολόγιο μπορείτε να φτειάξετε καλώντας τις σχετικές ρουτίνες του λειτουργικού που παρουσιάζονται στο 8ο κεφάλαιο αυτού του βιβλίου.

3ο ΚΕΦΑΛΑΙΟ

ΧΕΙΡΙΣΜΟΣ ΟΘΟΝΗΣ

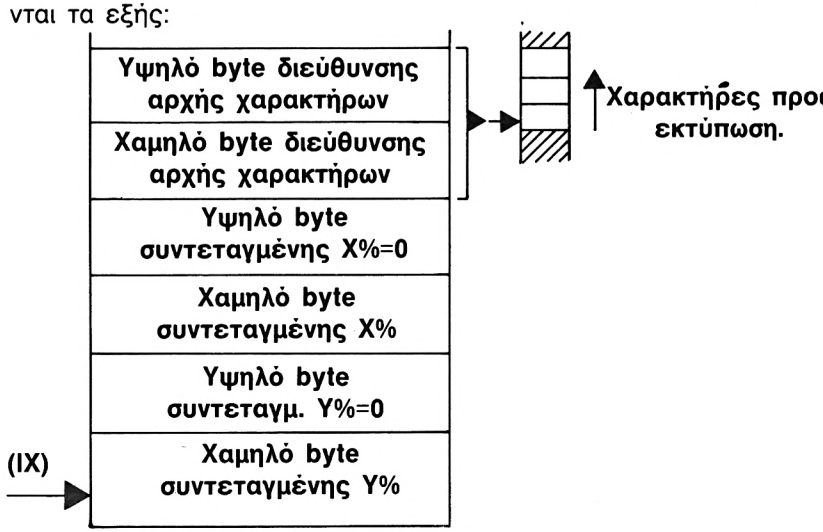
Σε αυτό το κεφάλαιο θα ασχοληθούμε με το χειρισμό της οθόνης μέσω κώδικα μηχανής αρκετά εκτεταμένα, δίνοντας πολλές σχετικές ρουτίνες. Οι ρουτίνες που θα παρουσιάσουμε εδώ είναι απαραίτητες, ιδίως σε προγράμματα κώδικα μηχανής, όπου δε διαθέτουμε τις ευκολίες εντολών όπως η PRINT της BASIC.

ΤΥΠΩΜΑ ΧΑΡΑΚΤΗΡΩΝ

Η πρώτη ρουτίνα που παρουσιάζουμε τυπώνει ένα STRING χαρακτήρων στο stream και με τα χρώματα που έχουν επιλεχθεί. Πρέπει να σημειώσουμε ότι εδώ, οτιδήποτε χαρακτήρες ελέγχου δώσουμε, τυπώνονται, δεν εκτελούνται.

Η ρουτίνα αυτή καλείται από BASIC με CALL 42000, address%, X%, Y% όπου address είναι η διεύθυνση μνήμης που τα περιεχόμενά της θέλουμε να τυπώσουμε σαν χαρακτήρες, X% και Y% είναι οι συντεταγμένες όπου θέλουμε να ξεκινήσουμε το τύπωμα στην οθόνη. Η ρουτίνα αντιλαμβάνεται το τέλος των χαρακτήρων όταν βρει σε μία θέση μνήμης περιεχόμενο 0. Με αυτόν τον τρόπο πρέπει να ορίσουμε λοιπόν τον τερματισμό του string.

Ο καταχωρητής IX «δείχνει» σε μια διεύθυνση μνήμης που περιέχονται τα εξής:



ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ BASIC: CALL 42000, address%, X%, Y% με τις μεταβλητές όπως έχουν οριστεί.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ M/C: Ο IX πρέπει να περιέχει τη διεύθυνση του μπλοκ της μνήμης που φαίνεται στον προηγούμενο πίνακα. Ο A πρέπει να περιέχει 3.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ M/C: Ολοι οι καταχωρητές έχουν τυχαίο περιεχόμενο.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ BASIC: Εχει τυπωθεί το σύνολο των χαρακτήρων στην οθόνη. Αυτόματη εξαγωγή από τη ρουτίνα γίνεται αν δεν έχουμε 3 μεταβλητές στο CALL ή αν ο καταχωρητής A δεν έχει την τιμή 3.

```
Hisoft GENA3.1 Assembler. Page      1.

Pass 1 errors: 00

A410                10          org  42000
A410  FE03          20          CP   3                  ;three parameters
A412  C0            30          RET  NZ
A413  DD4E00        40          LD   C,(IX+0)           ;get y%
A416  DD23          50          INC  IX
A418  DD23          60          INC  IX
A41A  DD4600        70          LD   B,(IX+0)           ;get x%
A41D  DD23          80          INC  IX
A41F  DD23          90          INC  IX
A421  DD6E00       100          LD   L,(IX+0)           ;get address of
A424  DD6601       110          LD   H,(IX+1)           ;string in HL
A427  E5           120          PUSH HL
A428  CD54BB       130          CALL 47956              ;enable screen display
A42B  3E1F         140          LD   A,31
A42D  CD5ABB       150          CALL 47962              ;put text cursor
A430  78           160          LD   A,B                ;using chr$(31)
A431  CD5ABB       170          CALL 47962
A434  79           180          LD   A,C
A435  CD5ABB       190          CALL 47962
A438  DDE1         200          POP  IX                 ;string address in IX
A43A  DD7E00       210 LOOP:    LD   A,(IX+0)           ;get character
A43D  FE00         220          CP   0                  ;is it 0?
A43F  C8           230          RET  Z                  ;if yes,quit
A440  DDE5         240          PUSH IX
A442  CD5DBB       250          CALL 47965              ;print character
A445  DDE1         260          POP  IX                 ;recover IX
A447  DD23         270          INC  IX                 ;point to next chr
A449  18EF         280          JR   LOOP               ;do it again

Pass 2 errors: 00

Table used:     24  from    166
```

```
10 J=42000
20 RESTORE
30 READ A$
40 IF A$="TELOS" THEN END
50 FOR I=1 TO LEN(A$) STEP 2
60 POKE J,VAL("&"+MID$(A$,I,2))
70 J=J+1
80 NEXT
90 GOTO 30
100 DATA FE03C0DD4E00DD23DD23DD4600DD23DD23DD6E00DD6601E5CD54BB3E1FCD
110 DATA 5ABB78CD5ABB79CD5ABBDDE1DD7E00FE00C8DDE5CD5DBBDDE1DD2318EF00
120 DATA TELOS
```

Ένα παράδειγμα για το πώς δουλεύει η παραπάνω ρουτίνα μπορείτε να έχετε με το εξής πρόγραμμα σε BASIC:

```
10 MEMORY 39999
20 CLS
30 INPUT "X%=";X%
40 INPUT "Y%=";Y%
50 INPUT "A$=";A$
60 A$=A$+CHR$(0)
70 FOR I=1 TO LEN(A$):POKE 39999+I,ASC(MID$(A$,I,1)):NEXT
80 CALL 42000,40000,X%,Y%
90 PRINT:GOTO 30
```

Μπορείτε με αυτό να τυπώσετε τους χαρακτήρες που γίνονται INPUT στο a$ (γραμμή 50) στις συντεταγμένες X% και Y% που γίνονται INPUT στις γραμμές 30 - 40 αντίστοιχα.

Αν το σύνολο των προς εκτύπωση χαρακτήρων δε χωράει σε μία σειρά της οθόνης, οι υπόλοιποι θα τυπωθούν στην επόμενη σειρά. Η ρουτίνα φορτώνεται σε οποιαδήποτε θέση μνήμης.

ΤΥΠΩΜΑ ΧΑΡΑΚΤΗΡΩΝ ΚΑΙ ΚΩΔΙΚΩΝ ΕΛΕΓΧΟΥ.

Η διαφορά της παρακάτω ρουτίνας απ' αυτήν που παρουσιάσαμε προηγουμένως, είναι ότι τους χαρακτήρες ελέγχου δεν τους τυπώνει αλλά τους εκτελεί. Έτσι για παράδειγμα ο CHR$(7) δίνει ένα BEEP αντί για τον ανάλογο γραφικό χαρακτήρα που τυπώνεται με την προηγούμενη ρουτίνα. Η ρουτίνα καλείται με CALL 42000,address%,m% όπου address% είναι η διεύθυνση μνήμης που βρίσκονται οι χαρακτήρες - κωδικοί που θα τυπωθούν ή θα εκτελεστούν, και η m% δίνει το μήκος (μέχρι 255 χαρακτήρες). Η ρουτίνα αυτή μπορεί να φορτωθεί σε οποιαδήποτε θέση μνήμης.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ BASIC: CALL 42000, address%, m% με τις μεταβλητές όπως έχουμε πει παραπάνω.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ M/C: Από κώδικα μηχανής μπορούμε να ξεκινήσουμε από την PRINT (παραλείποντας τις προηγούμενες εντολές), έχοντας στον B το μήκος του string και στον IX τη θέση μνήμης που ξεκινάει.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Η ρουτίνα επιστρέφει αφού τελειώσει η «εκτέλεση» των χαρακτήρων (είτε εκτύπωση είτε εκτέλεση των κωδικών). Αυτόματη επιστροφή γίνεται αν δεν έχουν δοθεί 2 μεταβλητές από BASIC. Οι AF, BC και IX έχουν τυχαίο περιεχόμενο.

```
Hisoft GENA3.1 Assembler. Page      1.

Pass 1 errors: 00

A410                10          org  42000
A410  FE02          20          CP   2
A412  C0            30          RET  NZ
A413  DD4600        40          LD   B,(IX+0)          ;get m% in B
A416  DD23          50          INC  IX
A418  DD23          60          INC  IX
A41A  DD6E00        70          LD   L,(IX+0)          ;get address
A41D  DD6601        80          LD   H,(IX+1)          ;of codes in HL
A420  E5            90          PUSH HL                ;and then
A421  DDE1          100         POP  IX                ;in IX
A423  CD54BB        110 PRINT:  CALL 47956             ;enable text VDU
A426  DD7E00        120 LOOP:   LD   A,(IX+0)          ;get code
A429  CD5ABB        130         CALL 47962             ;send code to VDU
A42C  DD23          140         INC  IX                ;point to next code
A42E  10F6          150         DJNZ LOOP              ;repeat m% times
A430  C9            160         RET

Pass 2 errors: 00

Table used:     36  from    140
```

```
10 J=42000
20 RESTORE
30 READ A$
40 IF A$="TELOS" THEN END
50 FOR I=1 TO LEN(A$) STEP 2
60 POKE J,VAL("&"+MID$(A$,I,2))
70 J=J+1
80 NEXT
90 GOTO 30
100 DATA FE02C0DD4600DD23DD23DD6E00DD6601E5DDE1CD54BBDD7E00CD5ABBDD23
110 DATA 10F6C9
120 DATA TELOS
```

Για παράδειγμα από κώδικα μηχανής η παραπάνω ρουτίνα μπορεί να χρησιμοποιηθεί ως εξής:

```
LD   IX, 41000
LD   B, 10
CALL   PRINT
RET
```

Με αυτό τον τρόπο εκτελούμε τους κωδικούς χαρακτήρες ενός string που βρίσκεται στη θέση 41000 και έχει μήκος 10. Από BASIC οι αντίστοιχοι χαρακτήρες εισάγονται στη μνήμη με POKE και μετά δίνουμε CALL 42000, 41000, 10. Οι διάφοροι κωδικοί χαρακτήρες (0-31) και τι κάνει ο καθένας περιγράφονται στο manual του υπολογιστή και μπορείτε να τους βρείτε στον αντίστοιχο πίνακα στο τέλος αυτού του βιβλίου.

ΤΥΠΩΜΑ ΧΑΡΑΚΤΗΡΩΝ ΜΕ ΕΛΕΓΧΟΜΕΝΗ ΑΠΟΣΤΑΣΗ

Στη συνέχεια θα παρουσιάσουμε μια ρουτίνα που τυπώνει τους χαρακτήρες στην οθόνη σε συγκεκριμένες συντεταγμένες X, Y αλλά και

με ορισμένη (από εμάς) απόσταση μεταξύ τους. Για μεγαλύτερη ευκολία θα δώσουμε πρώτα τη μορφή της ρουτίνας για χρήση μέσα από κώδικα μηχανής και στη συνέχεια μια άλλη μορφή που μπορεί να χρησιμοποιηθεί και από BASIC με την εντολή CALL.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ Μ/C: Ο BC έχει την απόσταση μεταξύ δύο χαρακτήρων (σε PIXELS). Ο HL έχει τη συντεταγμένη Y. Ο DE έχει τη συντεταγμένη X. Ο IX περιέχει την διεύθυνση που αρχίζει το string. Σημειώνουμε ότι το string πρέπει να τελειώνει με CHR$(0).

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Όλοι οι REGISTERS έχουν τυχαίο περιεχόμενο.

```
Hisoft GENA3.1 Assembler. Page      1.

Pass 1 errors: 00

A410                10          org  42000
A410  C5            20 PRINT:   PUSH BC              ;preserve registers
A411  D5            30          PUSH DE
A412  E5            40          PUSH HL
A413  CDC0BB        50          CALL 48064           ;put text cursor
A416  E1            60          POP  HL
A417  D1            70          POP  DE
A418  C1            80          POP  BC
A419  DD7E00        90          LD   A,(IX+0)        ;get character
A41C  FE00         100          CP   0               ;is it 0?
A41E  C8           110          RET  Z               ;if so quit
A41F  C5           120          PUSH BC
A420  D5           130          PUSH DE
A421  E5           140          PUSH HL
A422  CDFCBB       150          CALL 48124           ;print character
A425  E1           160          POP  HL
A426  D1           170          POP  DE
A427  C1           180          POP  BC
A428  E5           190          PUSH HL              ;preserve y pos
A429  D5           200          PUSH DE              ;get x position
A42A  E1           210          POP  HL              ;into HL
A42B  AF           220          XOR  A               ;clear carry flag
A42C  ED4A         230          ADC  HL,BC           ;new x position
A42E  E5           240          PUSH HL              ;return it
A42F  D1           250          POP  DE              ;to DE
A430  E1           260          POP  HL              ;recover y
A431  DD23         270          INC  IX              ;point to next chr
A433  18DB         280          JR   PRINT           ;do it again

Pass 2 errors: 00

Table used:     25  from    162
```

Σημειώνουμε ότι η εκτύπωση γίνεται στο GRAPHICS χρώμα που έχουμε, ενώ η απόσταση σε PIXELS εξαρτάται από το MODE που βρισκόμαστε.

Τώρα θα παρουσιάσουμε την ίδια ρουτίνα που μπορεί να χρησιμοποιηθεί από BASIC. Μπορούμε να την χρησιμοποιήσουμε με την εντολή CALL 42000, x%, y%, b%, @a$ όπουοι μεταβλητές x% και y% είναι οι αντίστοιχες συντεταγμένες, η b% δίνει την απόσταση μεταξύ

δύο διαδοχικών χαρακτήρων σε PIXELS και η a$ είναι το string που θέλουμε να τυπώσουμε. Ο τελευταίος χαρακτήρας του a$ πρέπει να είναι 0 πράγμα που το πετυχαίνουμε δίνοντας (αφού βάλουμε πρώτα τους χαρακτήρες που θέλουμε) a$=a$+CHR$(0). Η ρουτίνα αυτή μπορεί να φορτωθεί σε οποιαδήποτε θέση της μνήμης.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ BASIC: CALL 42000, x%, y%, b%, @a$ με μεταβλητές όπως ορίστηκαν προηγουμένως.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Από την ρουτίνα βγαίνουμε όταν τυπωθούν όλοι οι χαρακτήρες. Αυτόματη έξοδος γίνεται αν δώσουμε μεταβλητές που να μην είναι 4.

```
Hisoft GENA3.1 Assembler. Page      1.

Pass 1 errors: 00

A410                10          org   42000
A410  FE04          20          CP    4                ;four parameters
A412  C0            30          RET   NZ
A413  DD6E00        40          LD    L,(IX+0)
A416  DD6601        50          LD    H,(IX+1)
A419  23            60          INC   HL               ;HL points to a$ address
A41A  4E            70          LD    C,(HL)           ;get a$ address
A41B  23            80          INC   HL
A41C  46            90          LD    B,(HL)           ;into BC
A41D  C5           100          PUSH  BC
A41E  DD4E02       110          LD    C,(IX+2)         ;char spacing
A421  DD4603       120          LD    B,(IX+3)         ;into BC
A424  DD6E04       130          LD    L,(IX+4)         ;y coordinate
A427  DD6605       140          LD    H,(IX+5)         ;into HL
A42A  DD5E06       150          LD    E,(IX+6)         ;x coordinate
A42D  DD5607       160          LD    D,(IX+7)         ;into DE
A430  DDE1         170          POP   IX               ;a$ address in IX
A432  C5           180 PRINT:   PUSH  BC               ;preserve registers
A433  D5           190          PUSH  DE
A434  E5           200          PUSH  HL
A435  CDC0BB       210          CALL  48064            ;put text cursor
A438  E1           220          POP   HL
A439  D1           230          POP   DE
A43A  C1           240          POP   BC
A43B  DD7E00       250          LD    A,(IX+0)         ;get character
A43E  FE00         260          CP    0                ;is it 0?
A440  C8           270          RET   Z                ;if so quit
A441  C5           280          PUSH  BC
A442  D5           290          PUSH  DE
A443  E5           300          PUSH  HL
A444  CDFCBB       310          CALL  48124            ;print character
A447  E1           320          POP   HL
A448  D1           330          POP   DE
A449  C1           340          POP   BC
A44A  E5           350          PUSH  HL               ;preserve y pos
A44B  D5           360          PUSH  DE               ;get x position
A44C  E1           370          POP   HL               ;into HL
A44D  AF           380          XOR   A                ;clear carry flag
A44E  ED4A         390          ADC   HL,BC            ;new x position
A450  E5           400          PUSH  HL               ;return it
A451  D1           410          POP   DE               ;to DE
A452  E1           420          POP   HL               ;recover y
A453  DD23         430          INC   IX               ;point to next chr
A455  18DB         440          JR    PRINT            ;do it again

Pass 2 errors: 00
Table used:     25  from    203
```

```
10 J=42000
20 RESTORE
30 READ A$
40 IF A$="TELOS" THEN END
50 FOR I=1 TO LEN(A$) STEP 2
60 POKE J,VAL("&"+MID$(A$,I,2))
70 J=J+1
80 NEXT
90 GOTO 30
100 DATA FE04C0DD6E00DD6601234E2346C5DD4E02DD4603DD6E04DD6605DD5E06DD
110 DATA 5607DDE1C5D5E5CDC0BBE1D1C1DD7E00FE00C8C5D5E5CDFCBBE1D1C1E5D5
120 DATA E1AFED4AE5D1E1DD2318DB
130 DATA TELOS
```

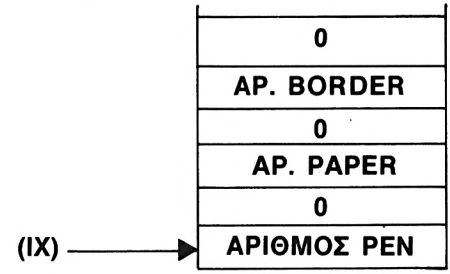
Τώρα για παράδειγμα δείτε το παρακάτω προγραμματάκι σε BASIC, που δείχνει πως χρησιμοποιείται η προηγούμενη ρουτίνα.

```
10 MODE 2
20 MEMORY 41999
30 LOCATE 1,22
40 INPUT "a$=";a$
50 a$=a$+CHR$(0)
60 INPUT"x%=";x%
70 INPUT"y%=";y%
80 INPUT"space between characters=";b%
90 CALL 42000,x%,y%,b%,@a$
100 GOTO 30
```

PEN/PAPER/BORDER

Με την ρουτίνα που παρουσιάζουμε στη συνέχεια μπορείτε να βάλετε μέσω κώδικα μηχανής το PEN το PAPER και το BORDER στα χρώματα που θέλετε, για GRAPHICS.

Ο καταχωρητής IX περιέχει την διεύθυνση ενός block μνήμης που φαίνεται παρακάτω.



Από BASIC η ρουτίνα καλείται με CALL 42000, b%, pa%, pe% όπου b%=αριθμός border pa%=αρ. paper για γραφικά pe%=αριθμός pen για γραφικά. Η ρουτίνα μπορεί να φορτωθεί σε οποιαδήποτε θέση μνήμης.

```
Hisoft GENA3.1 Assembler. Page      1.

Pass 1 errors: 00

A410                10          org  42000
A410  FE03          20          CP   3                    ;three parameters
A412  C0            30          RET  NZ
```

```
A413  DD7E00      40          LD   A,(IX+0)
A416  CDDEBB      50          CALL 48094              ;set graphics pen
A419  DD7E02      60          LD   A,(IX+2)
A41C  CDE4BB      70          CALL 48100              ;set graphics paper
A41F  DD4E04      80          LD   C,(IX+4)
A422  41          90          LD   B,C
A423  CD38BC     100          CALL 48184              ;set border
A426  C9         110          RET

Pass 2 errors: 00

Table used:     13  from    123
```

```
10 J=42000
20 RESTORE
30 READ A$
40 IF A$="TELOS" THEN END
50 FOR I=1 TO LEN(A$) STEP 2
60 POKE J,VAL("&"+MID$(A$,I,2))
70 J=J+1
80 NEXT
90 GOTO 30
100 DATA FE03C0DD7E00CDDEBBDD7E02CDE4BBDD4E0441CD38BCC9
110 DATA TELOS
```

ΤΥΠΩΜΑ ΜΕΓΕΝΘΥΜΕΝΩΝ ΧΑΡΑΚΤΗΡΩΝ

Η ρουτίνα που παρουσιάζουμε στη συνέχεια τυπώνει κάποιον χαρακτήρα που δίνουμε, σε μέγεθος 8×8 χαρακτήρων. Σημειώνουμε ότι μπορούμε να την χρησιμοποιήσουμε σε οποιοδήποτε MODE και ο χαρακτήρας τυπώνεται στη θέση που βρίσκεται ο δρομέας με το χρώμα που υπάρχει εκείνη τη στιγμή στο PEN.

Η ρουτίνα καλείται με CALL 42000, n% όπου n% είναι ο ASCII του χαρακτήρα που θέλουμε να μεγενθύνουμε.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ BASIC: CALL 42000, n%

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ Μ/C: Πρέπει ο καταχωρητής A να έχει την τιμή 1 και ο IX να περιέχει τη διεύθυνση που υπάρχει ο ASCII του χαρακτήρα που θέλουμε να μεγενθύνουμε.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ BASIC: Από τη ρουτίνα βγαίνουμε όταν ολοκληρωθεί η εκτέλεσή της ή αν δεν έχουμε δώσει στο CALL ακριβώς 1 παράμετρο.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ Μ/C: Όλοι οι καταχωρητές έχουν τυχαίο περιεχόμενο.

```
Hisoft GENA3.1 Assembler. Page      1.

Pass 1 errors: 00

A410              10          ORG  42000
A410  FE01        20          CP   1                  ;one parameter
A412  C0          30          RET  NZ
```

```
A413  CD06B9      40          CALL 47366          ;enable lower ROM
A416  F5          50          PUSH AF             ;save previews state
A417  DD7E00      60          LD   A,(IX+0)       ;get character
A41A  CDA5BB      70          CALL 48037          ;get chr matrix
A41D  DD21409C    80          LD   IX,40000       ;dest. of matrix
A421  0608        90          LD   B,8            ;8 bytes of matrix
A423  7E         100 LOOP:    LD   A,(HL)
A424  DD7700     110          LD   (IX+0),A
A427  23         120          INC  HL
A428  DD23       130          INC  IX
A42A  10F7       140          DJNZ LOOP
A42C  F1         150          POP  AF             ;previews ROM state
A42D  CD0CB9     160          CALL 47372          ;recover
A430  DD21409C   170          LD   IX,40000
A434  1608       180          LD   D,8
A436  DD7E00     190 LOOP2:   LD   A,(IX+0)       ;get byte
A439  4F         200          LD   C,A
A43A  0608       210          LD   B,8
A43C  CB21       220 LOOP3:   SLA  C              ;each byte's bit
A43E  3807       230          JR   C,ZERO
A440  3E20       240          LD   A,32           ;space if bit=0
A442  CD5ABB     250          CALL 47962
A445  1805       260          JR   NEXT
A447  3EFF       270 ZERO:    LD   A,255          ;chr 255 if bit=1
A449  CD5ABB     280          CALL 47962
A44C  10EE       290 NEXT:    DJNZ LOOP3
A44E  DD23       300          INC  IX
A450  3E0A       310          LD   A,10           ;one line down
A452  CD5ABB     320          CALL 47962
A455  0608       330          LD   B,8
A457  3E08       340 LOOP4:   LD   A,8            ;send chr$(8)
A459  CD5ABB     350          CALL 47962          ;to be printed
A45C  10F9       360          DJNZ LOOP4          ;eight times
A45E  15         370          DEC  D
A45F  20D5       380          JR   NZ,            LOOP2;next byte
A461  C9         390          RET

Pass 2 errors: 00

Table used:     82  from   190
```

```
10 J=42000
20 RESTORE
30 READ A$
40 IF A$="TELOS" THEN END
50 FOR I=1 TO LEN(A$) STEP 2
60 POKE J,VAL("&"+MID$(A$,I,2))
70 J=J+1
80 NEXT
90 GOTO 30
100 DATA FE01C0CD06B9F5DD7E00CDA5BBDD21409C06087EDD770023DD2310F7F1CD
110 DATA 0CB9DD21409C1608DD7E004F0608CB2138073E20CD5ABB18053EFFCD5ABB
120 DATA 10EEDD233E0ACD5ABB06083E08CD5ABB10F91520D5C9
130 DATA TELOS
```

Για παράδειγμα η ρουτίνα μπορεί να χρησιμοποιηθεί από BASIC με τον ακόλουθο τρόπο:

y=10: a$="AMSTRAD": FOR x=1 TO LEN (a$): LOCATE x*8+1, y: CALL 42000, ASC (MID$ (a$, x, 1)): NEXT.

Ο χαρακτήρας τυπώνεται με τον χαρακτήρα 255 (κάθε "pixel" του μεγάλου χαρακτήρα είναι ένας CHR$ (255)). Αλλάζοντάς τον συνεπώς έχετε ότι αποτέλεσμα θέλετε.

ΤΥΠΩΜΑ ΔΙΠΛΩΝ ΧΑΡΑΚΤΗΡΩΝ

Η επόμενη ρουτίνα τυπώνει τους χαρακτήρες που θέλουμε σε διπλό

ύψος. Και αυτή δουλεύει σε οποιοδήποτε MODE και τυπώνει στη θέση που βρίσκεται ο δρομέας με το χρώμα που υπάρχει αυτή τη στιγμή.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ BASIC: CALL 42000, n%. Όπου n% είναι ο ASCII του χαρακτήρα που θέλουμε να τυπώσουμε σε διπλό ύψος.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ M/C: Ο A πρέπει να έχει τιμή 1 και ο IX να περιέχει την διεύθυνση που υπάρχει ο ASCII του χαρακτήρα που θέλουμε να τυπώσει.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Όλοι οι καταχωρητές έχουν τυχαίο περιεχόμενο. Από τη ρουτίνα βγαίνουμε όταν ολοκληρωθεί το τύπωμα ή αν δεν έχουμε περάσει με το CALL ακριβώς 1 μεταβλητή.

```
riHisoft GENA3.1 Assembler. Page     1.

Pass 1 errors: 00

A410                10          org  42000
A410  FE01          20          CP   1                    ;one parąmeter
A412  C0            30          RET  NZ
A413  CD06B9        40          CALL 47366                ;enable lower ROM
A416  F5            50          PUSH AF                   ;save previews state
A417  DD7E00        60          LD   A,(IX+0)             ;get character
A41A  CDA5BB        70          CALL 48037                ;get chr matrix
A41D  DD21409C      80          LD   IX,40000             ;dest. of matrix
A421  0608          90          LD   B,8                  ;8 bytes to move
A423  7E          100 LOOP1:    LD   A,(HL)
A424  DD7700        110         LD   (IX+0),A
A427  DD23          120         INC  IX                   ;save matsrrix
A429  DD7700        130         LD   (IX+0),A             ;2 times
A42C  DD23          140         INC  IX
A42E  23            150         INC  HL
A42F  10F2          160         DJNZ LOOP1
A431  F1            170         POP  AF
A432  CD0CB9        180         CALL 47372                ;restore pr. ROM
A435  3EFE          190         LD   A,254                ;upper half
A437  21409C        200         LD   HL,40000             ;in chr$(254)
A43A  CDA8BB        210         CALL 48040
A43D  3EFF          220         LD   A,255                ;lower half
A43F  21479C        230         LD   HL,40007             ;in chr$(255)
A442  CDA8BB        240         CALL 48040
A445  3EFE          250         LD   A,254                ;print top half
A447  CD5ABB        260         CALL 47962
A44A  3E0A          270         LD   A,10                 ;one line down
A44C  CD5ABB        280         CALL 47962
A44F  3E08          290         LD   A,8                  ;one space back
A451  CD5ABB        300         CALL 47962
A454  3EFF          310         LD   A,255                ;bottom half
A456  CD5ABB        320         CALL 47962
A459  C9            330         RET

Pass 2 errors: 00

Table used:     25  from    180
```

```
10 J=42000
20 RESTORE
30 READ A$
```

```
40 IF A$="TELOS" THEN END
50 FOR I=1 TO LEN(A$) STEP 2
60 POKE J,VAL("&"+MID$(A$,I,2))
70 J=J+1
80 NEXT
90 GOTO 30
100 DATA FE01C0CD06B9F5DD7E00CDA5BBDD21409C06087EDD7700DD23DD7700DD23
110 DATA 2310F2F1CD0CB93EFE21409CCDA8BB3EFF21479CCDA8BB3EFECD5ABB3E0A
120 DATA CD5ABB3E08CD5ABB3EFFCD5ABBC9
130 DATA TELOS
```

Εδώ δίνουμε για παράδειγμα έναν τρόπο που μπορεί να χρησιμοποιηθεί αυτή η ρουτίνα από BASIC.
x=1: y=10: MODE 1: α$="AMSTRAD": FOR i=1 TO LEN (α$): LOCATE x+i-1, y: CALL 42000, ASC (MID$ (α$, i, 1)): NEXT.

ΤΥΠΩΜΑ ΧΑΡΑΚΤΗΡΩΝ ΜΕΤΑΒΛΗΤΟΥ ΥΨΟΥΣ

Στη συνέχεια σας παρουσιάζουμε μια πολύ χρήσιμη ρουτίνα που μπορεί να εκτυπώσει χαρακτήρες σε διάφορα ύψη. Μπορεί να δουλέψει σε όλα τα MODES, στη θέση που βρίσκεται ο δρομέας γραφικών με το τρέχον χρώμα γραφικών.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ BASIC: CALL 420000, n%, m% όπου n% είναι ο ASCII του χαρακτήρα που θέλουμε να τυπώσουμε και m% η μεταβλητή που χαρακτηρίζει το ύψος.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ Μ/C: Ο Α πρέπει να περιέχει την τιμή 2 και ο IX να δείχνει τη διεύθυνση ενός μπλοκ μνήμης όπου στην διεύθυνση (IX+0) βρίσκεται το ύψος και στην (IX+2) βρίσκεται ο ASCII του χαρακτήρα που θέλουμε να τυπώσουμε.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ολοι οι καταχωρητές έχουν τυχαίο περιεχόμενο. Βγαίνουμε απ'τη ρουτίνα αφού ολοκληρωθεί η εκτέλεσή της ή αν ο αριθμός των μεταβλητών που περνάμε στο CALL δεν είναι ακριβώς 2.

```
Hisoft GENA3.1 Assembler. Page     1.

Pass 1 errors: 00

A410                10          ORG  42000
A410  FE02          20          CP   2
A412  C0            30          RET  NZ
A413  DD7E00        40          LD   A,(IX+0)
A416  32FDA4        50          LD   (HGHT),A          ;store height
A419  CD06B9        60          CALL 47366             ;enable lower ROM
A41C  F5            70          PUSH AF                ;save previews state
```

```
A41D  DD7E02      80          LD   A,(IX+2)          ;get character
A420  CDA5BB      90          CALL 48037             ;get chr matrix
A423  DD21409C   100          LD   IX,40000
A427  0608       110          LD   B,8               ;8 bytes to move
A429  7E         120 LOOP1:   LD   A,(HL)
A42A  DD7700     130          LD   (IX+0),A
A42D  23         140          INC  HL
A42E  DD23       150          INC  IX
A430  10F7       160          DJNZ LOOP1
A432  F1         170          POP  AF
A433  CD0CB9     180          CALL 47372             ;restore previews ROM
A436  DD21409C   190          LD   IX,40000
A43A  1608       200          LD   D,8               ;8 bytes of chr
A43C  DD7E00     210 LOOP2:   LD   A,(IX+0)          ;get chr byte
A43F  4F         220          LD   C,A
A440  0608       230          LD   B,8
A442  CB21       240 LOOP3:   SLA  C                 ;each bit
A444  CD94A4     250          CALL GCURS             ;gr curs position
A447  3805       260          JR   C,ZERO            ;if bit=1
A449  CDE4A4     270          CALL VERTS             ;else print space
A44C  1803       280          JR   NEXT
A44E  CDCFA4     290 ZERO:    CALL VERT              ;draw the bit
A451  10EF       300 NEXT:    DJNZ LOOP3             ;repeat 8 times
A453  DD23       310          INC  IX                ;next byte
A455  CD7CA4     320          CALL OUTH              ;restore cur. position
A458  15         330          DEC  D                 ;repeat for
A459  20E1       340          JR   NZ,LOOP2          ;8 bytes
A45B  C9         350          RET                    ;then return
A45C  E5         360 PBLOCK:  PUSH HL                ;save registers
A45D  D5         370          PUSH DE
A45E  C5         380          PUSH BC
A45F  110600     390          LD   DE,6
A462  210000     400          LD   HL,0
A465  CDF9BB     410          CALL 48121             ;drawr horiz
A468  C1         420          POP  BC                ;restore registers
A469  D1         430          POP  DE
A46A  E1         440          POP  HL
A46B  C9         450          RET
A46C  E5         460 PSPACE:  PUSH HL                ;save registers
A46D  D5         470          PUSH DE
A46E  C5         480          PUSH BC
A46F  110600     490          LD   DE,6
A472  210000     500          LD   HL,0
A475  CDC3BB     510          CALL 48067             ;mover (print space)
A478  C1         520          POP  BC                ;restore registers
A479  D1         530          POP  DE
A47A  E1         540          POP  HL
A47B  C9         550          RET
A47C  C5         560 OUTH:    PUSH BC
A47D  E5         570          PUSH HL
A47E  D5         580          PUSH DE
A47F  210000     590          LD   HL,0
A482  3AFDA4     600          LD   A,(HGHT)
A485  47         610          LD   B,A
A486  2B         620 DLOOP:   DEC  HL
A487  2B         630          DEC  HL
A488  10FC       640          DJNZ DLOOP             ;restore vert. position
A48A  11D0FF     650          LD   DE,-48            ;restore hor. position
A48D  CDC3BB     660          CALL 48067             ;move relative
A490  D1         670          POP  DE
A491  E1         680          POP  HL
A492  C1         690          POP  BC
A493  C9         700          RET
A494  C5         710 GCURS:   PUSH BC
A495  E5         720          PUSH HL
A496  D5         730          PUSH DE
A497  F5         740          PUSH AF
A498  CDC6BB     750          CALL 48070             ;get curs position
A49B  0606       760          LD   B,6
A49D  13         770 CLOOP:   INC  DE                ;increase x pos.
A49E  10FD       780          DJNZ CLOOP             ;by 6
A4A0  ED53F9A4   790          LD   (STORX),DE        ;store x pos.
A4A4  22FBA4     800          LD   (STORY),HL        ;store y pos.
A4A7  F1         810          POP  AF
```

```
A4A8  D1          820          POP  DE
A4A9  E1          830          POP  HL
A4AA  C1          840          POP  BC
A4AB  C9          850          RET
A4AC  C5          860 RESTO:   PUSH BC
A4AD  D5          870          PUSH DE
A4AE  E5          880          PUSH HL
A4AF  11FAFF      890          LD   DE,-6              ;mover x-6
A4B2  21FEFF      900          LD   HL,-2              ;mover y-2 (down)
A4B5  CDC3BB      910          CALL 48067
A4B8  E1          920          POP  HL
A4B9  D1          930          POP  DE
A4BA  C1          940          POP  BC
A4BB  C9          950          RET
A4BC  F5          960 NEXT2:   PUSH AF
A4BD  C5          970          PUSH BC
A4BE  D5          980          PUSH DE
A4BF  E5          990          PUSH HL
A4C0  ED5BF9A4    1000         LD   DE,(STORX)         ;x in DE
A4C4  2AFBA4      1010         LD   HL,(STORY)         ;y in HL
A4C7  CDC0BB      1020         CALL 48064              ;abs. move to x,y
A4CA  E1          1030         POP  HL
A4CB  D1          1040         POP  DE
A4CC  C1          1050         POP  BC
A4CD  F1          1060         POP  AF
A4CE  C9          1070         RET
A4CF  C5          1080 VERT:   PUSH BC
A4D0  CD94A4      1090         CALL GCURS
A4D3  3AFDA4      1100         LD   A,(HGHT)           ;height in A
A4D6  47          1110         LD   B,A                ;and then in B
A4D7  CD5CA4      1120 VLOOP:  CALL PBLOCK             ;draw "pixel"
A4DA  CDACA4      1130         CALL RESTO              ;then restore
A4DD  10F8        1140         DJNZ VLOOP              ;repeate B times
A4DF  CDBCA4      1150         CALL NEXT2              ;then next line
A4E2  C1          1160         POP  BC
A4E3  C9          1170         RET
A4E4  C5          1180 VERTS:  PUSH BC
A4E5  CD94A4      1190         CALL GCURS
A4E8  3AFDA4      1200         LD   A,(HGHT)
A4EB  47          1210         LD   B,A
A4EC  CD6CA4      1220 FLOOP:  CALL PSPACE             ;draw "space"
A4EF  CDACA4      1230         CALL RESTO              ;then restore
A4F2  10F8        1240         DJNZ FLOOP              ;repeat B times
A4F4  CDBCA4      1250         CALL NEXT2              ;then next line
A4F7  C1          1260         POP  BC
A4F8  C9          1270         RET
A4F9              1280 STORX:  DEFS 2                  ;store space for x
A4FB              1290 STORY:  DEFS 2                  ;for y
A4FD  00          1300 HGHT:   DEFB 0                  ;height byte
A4FE  C9          1310         RET

Pass 2 errors: 00

Table used:   250  from   380
```

```
10 J=42000
20 RESTORE
30 READ A$
40 IF A$="TELOS" THEN END
50 FOR I=1 TO LEN(A$) STEP 2
60 POKE J,VAL("&"+MID$(A$,I,2))
70 J=J+1
80 NEXT
90 GOTO 30
100 DATA FE02C0DD7E0032FDA4CD06B9F5DD7E02CDA5BBDD21409C06087EDD770023
110 DATA DD2310F7F1CD0CB9DD21409C1608DD7E004F0608CB21CD94A43805CDE4A4
120 DATA 1803CDCFA410EFDD23CD7CA41520E1C9E5D5C5110600210000CDF9BBC1D1
130 DATA E1C9E5D5C5110600210000CDC3BBC1D1E1C9C5E5D5210000 3AFDA4472B2B
140 DATA 10FC11D0FFCDC3BBD1E1C1C9C5E5D5F5CDC6BB0606131 0FDED53F9A422FB
150 DATA A4F1D1E1C1C9C5D5E511FAFF21FEFFCDC3BBE1D1C1C9F5C5D5E5ED5BF9A4
160 DATA 2AFBA4CDC0BBE1D1C1F1C9C5CD94A43AFDA447CD5CA4CDACA410F8CDBCA4
170 DATA C1C9C5CD94A43AFDA447CD6CA4CDACA410F8CDBCA4C1C900000000000C9
180 DATA TELOS
```

Ενα παράδειγμα του πως χρησιμοποιείται η παραπάνω ρουτίνα από BASIC είναι το παρακάτω προγραμμάτακι.

```
10 MODE 2
20 INPUT "Height=";b%
30 INPUT "String=";a$
40 MOVE 100,100
50 FOR i=1 TO LEN (a$)
60 MOVE I*50+10,100
70 CALL 42000,ASC(MID$(a$,i,1)),b%
80 NEXT
```

Επειδή σ'αυτήν την περίπτωση χρησιμοποιούμε τη θέση του δρομέα ΓΡΑΦΙΚΩΝ πρέπει να καθοριστεί η θέση που θέλουμε να τυπώσουμε τους χαρακτήρες με την εντολή MOVE και όχι με LOCATE.

Οι ρουτίνες του λειτουργικού που χρησιμοποιήσαμε αναλύονται στο κεφάλαιο 8 και σας συνιστούμε να δείτε τι κάνει η καθεμία ώστε να γίνει πιο κατανοητό το LISTING σε ASSEMBLY.

ΧΑΡΑΚΤΗΡΕΣ ΤΥΠΩΜΕΝΟΙ ΑΝΤΙΣΤΡΟΦΑ

Εδώ σας παρουσιάζουμε μια ρουτίνα που τυπώνει τους χαρακτήρες που θέλουμε είτε ανάποδα, είτε αντικατοπτρικά. Αυτό είναι πολύ χρήσιμο όταν τυπώνουμε γραφικούς χαρακτήρες (π.χ. ανθρωπάκια) γιατί μπορούμε να τα κάνουμε να «γυρίζουν» προς όποια κατεύθυνση θέλουμε. Η ρουτίνα καλείται με CALL 42000, n%, b% όπου n% είναι ο ASCII του χαρακτήρα που θέλουμε να τυπώσουμε και n% έχει την τιμή 1 αν θέλουμε να τυπώσουμε χαρακτήρα ανάποδα ή 2 αν θέλουμε να τον εκτυπώσουμε αντικατοπτρικά. Μπορεί να φορτωθεί σε οποιαδήποτε θέση της μνήμης.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ BASIC: CALL 42000, n%, b% όπως έχει οριστεί παραπάνω.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ Μ/C: Ο Α πρέπει να έχει την τιμή 2 και ο IX να περιέχει την θέση ενός μπλοκ όπου η διεύθυνση (IX+O) είναι 1 ή 2 δηλ. η τιμή του b% που περιγράψαμε παραπάνω και η (IX+2) έχει τον ASCII του χαρακτήρα που θέλουμε να τυπώσουμε.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ολοι οι κταχωρητές έχουν τυχαίο περιεχόμενο. Αυτόματη έξοδος γίνεται αν οι μεταβλητές που δίνουμε στο CALL δεν είναι 2 ή αν ο b% δεν έχει τιμή 1 ή 2.

```
Hisoft GENA3.1 Assembler. Page      1.

Pass 1 errors: 00

A410                10          org  42000
A410  FE02          20          CP   2
A412  C0            30          RET  NZ
A413  DD7E00        40          LD   A,(IX+0)          ;get choice
A416  FE02          50          CP   2                 ;if 2
A418  2836          60          JR   Z,MIRH            ;mirror image
A41A  FE01          70          CP   1                 ;if not 1
A41C  C0            80          RET  NZ                ;quit
A41D  DD23          90          INC  IX
A41F  DD23         100          INC  IX
A421  CD06B9       110          CALL 47366             ;enable lower ROM
A424  F5           120          PUSH AF                ;save previews state
A425  DD7E00       130          LD   A,(IX+0)          ;get character
A428  CDA5BB       140          CALL 48037             ;get its matrix
A42B  DD21409C     150          LD   IX,40000          ;place to store
A42F  110700       160          LD   DE,7              ;the matrix
A432  19           170          ADD  HL,DE
A433  0608         180          LD   B,8
A435  7E           190 LOOP:    LD   A,(HL)            ;last byte first
A436  DD7700       200          LD   (IX+0),A
A439  2B           210          DEC  HL
A43A  DD23         220          INC  IX
A43C  10F7         230          DJNZ LOOP              ;for all bytes
A43E  F1           240          POP  AF
A43F  CD0CB9       250          CALL 47372             ;restore ROM '
A442  21409C       260          LD   HL,40000          ;redefine chr 255
A445  3EFF         270          LD   A,255
A447  CDA8BB       280          CALL 48040
A44A  3EFF         290          LD   A,255
A44C  CD5ABB       300          CALL 47962             ;now print it
A44F  C9           310          RET                    ;and exit
A450  DD23         320 MIRH:    INC  IX
A452  DD23         330          INC  IX
A454  CD06B9       340          CALL 47366             ;enable lower ROM
A457  F5           350          PUSH AF                ;save previews state
A458  DD7E00       360          LD   A,(IX+0)          ;get character
A45B  CDA5BB       370          CALL 48037             ;get its matrix
A45E  DD21409C     380          LD   IX,40000          ;new place for matrix
A462  0608         390          LD   B,8
A464  7E           400 LOOP2:   LD   A,(HL)
A465  C5           410          PUSH BC
A466  0608         420          LD   B,8               ;8 bits in a byte
A468  4F           430          LD   C,A               ;original byte
A469  CB11         440 LOOP3:   RL   C                 ;move bit left
A46B  1F           450          RRA                    ;move carry to A
A46C  10FB         460          DJNZ LOOP3             ;repeat 8 times
A46E  C1           470          POP  BC
A46F  DD7700       480          LD   (IX+0),A          ;transfer modif. byte
A472  DD23         490          INC  IX
A474  23           500          INC  HL
A475  10ED         510          DJNZ LOOP2             ;next byte
A477  F1           520          POP  AF                ;restore previews
A478  CD0CB9       530          CALL 47372             ;ROM state
A47B  21409C       540          LD   HL,40000          ;redefine chr 255
A47E  3EFF         550          LD   A,255
A480  CDA8BB       560          CALL 48040
A483  3EFF         570          LD   A,255
A485  CD5ABB       580          CALL 47962             ;then print it
A488  C9           590          RET                    ;and quit

Pass 2 errors: 00

Table used:     59  from    242
```

```
10 J=42000
20 RESTORE
30 READ A$
40 IF A$="TELOS" THEN END
50 FOR I=1 TO LEN(A$) STEP 2
60 POKE J,VAL("&"+MID$(A$,I,2))
```

```
70 J=J+1
80 NEXT
90 GOTO 30
100 DATA FE02C0DD7E00FE022836FE01C0DD23DD23CD06B9F5DD7E00CDA5BBDD2140
110 DATA 9C1107001906087EDD77002BDD2310F7F1CD0CB921409C3EFFCDA8BB3EFF
120 DATA CD5ABBC9DD23DD23CD06B9F5DD7E00CDA5BBDD21409C06087EC506084FCB
130 DATA 111F10FBC1DD7700DD232310EDF1CD0CB921409C3EFFCDA8BB3EFFCD5ABB
140 DATA C9
150 DATA TELOS
```

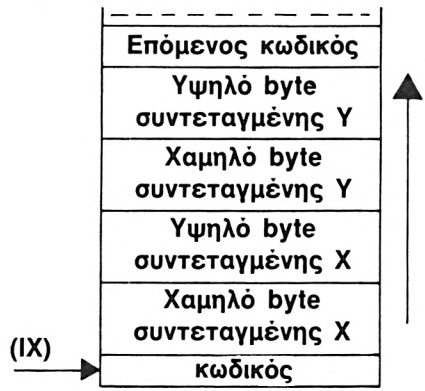
Ενα παράδειγμα χρήσης της παραπάνω ρουτίνας από BASIC είναι το εξής πρόγραμμα:

```
10 MODE 1
20 INPUT "a$=";a$
30 INPUT "ANAPODA H ANTISTROFA (1/2)";b%
40 IF b%<1 OR b%>2 THEN 30
50 LOCATE 10,10
60 FOR i=1 TO LEN(a$)
70 CALL 42000,ASC(MID$(a$,i,1)),b%
80 NEXT
90 LOCATE 1,1:GOTO 20
```

ΧΕΙΡΙΣΜΟΣ PLOT ΚΑΙ DRAW ΓΙΑ ΓΡΑΦΙΚΑ

Στη συνέχεια σας παρουσιάζουμε μια πολύ ενδιαφέρουσα ρουτίνα που σχεδιάζει διάφορα σχήματα στην οθόνη του υπολογιστή, σε οποιοδήποτε mode. Είναι πολύ ενδιαφέρον να αναλύσουμε τον τρόπο που δημιουργείται ο πίνακας δεδομένων της ρουτίνας, γιατί είναι μία απ' τις πιο διαδεδομένες μεθόδους που χρησιμοποιούνται για σχεδίαση γραφικών.

Ο πίνακας δεδομένων αποτελείται από σειρές των 5 bytes. Το πρώτο byte περιέχει έναν κωδικό που δίνει πληροφορίες σχετικά με το ποιά λειτουργία πρόκειται να εκτελεστεί ενώ τα επόμενα 4 είναι τα δεδομένα για την κάθε λειτουργία. Στη δική μας ρουτίνα ο κωδικός 1 κάνει MOVE σε νέες απόλυτες γραφικές συντεταγμένες, ο 2 κάνει απόλυτο DRAW ,ο 3 βάζει ένα χρώμα της επιλογής μας, ο 4 κάνει απόλυτο PLOT ενώ ο 0 σημαίνει τέλος και έξοδο από την ρουτίνα. Η μορφή που έχει ο πίνακας δεδομένων π.χ. στο απόλυτο DRAW είναι η εξής:



Μπορείτε να χρησιμοποιήσετε την ρουτίνα είτε όπως ακριβώς είναι, είτε με διάφορες μετατροπές μέσα στα προγράμματά σας. Μπορείτε βέβαια να δώσετε και τους αντίστοιχους κωδικούς που θέλετε εσείς. Μόνο σας συνιστούμε για μεγαλύτερη ευκολία να κρατήσετε το 0 για έξοδο από τη ρουτίνα. Επίσης κάθε φορά που αρχίζει να εκτελείται μια λειτουργία που καθορίζει κάποιος κωδικός, πρέπει να σώζεται ο καταχωρητής A (με την εντολή PUSH AF). Αυτό είναι απαραίτητο για να μπορέσουμε βγαίνοντας από τη ρουτίνα να αποκαταστήσουμε την τιμή του (με POP AF). Αλλιώς μπορεί ο A να πάρει κάποια τιμή που να ταιριάζει σε κάποια άλλη απ' τις επόμενες λειτουργίες οπότε βγαίνοντας απ' τη συγκεκριμένη λειτουργία που εκτελείται και συνεχίζοντας τους ελέγχους, να κληθεί μια άλλη υπορουτίνα καταστρέφοντας ουσιαστικά τη ροή του προγράμματος.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ Μ/C: ΚΑΜΜΙΑ. Στην πρώτη εντολή του LISTING σε ASSEMBLY δίνεται στον IX η διεύθυνση που αρχίζει ο πίνακας των δεδομένων.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ Μ/C: Όλοι οι καταχωρητές έχουν τυχαίο περιεχόμενο.

```
Hisoft GENA3.1 Assembler. Page      1.

Pass 1 errors: 00

A410                10          org   42000
A410  DD7E00        20 LOOP:    LD    A,(IX+0)          ;get first byte
A413  FE00          30          CP    0                 ;is it 0?
A415  C8            40          RET   Z                 ;if yes then quit
A416  FE01          50          CP    1                 ;is choice "move" ?
A418  CC2EA4        60          CALL  Z,MOVE            ;call rel routine
A41B  FE02          70          CP    2                 ;is choice draw ?
A41D  CC37A4        80          CALL  Z,DRAW            ;call rel routine
A420  FE03          90          CP    3                 ;is choice color ?
A422  CC40A4       100          CALL  Z,COLOR           ;call rel routine
A425  FE04         110          CP    4                 ;is choice plot ?
A427  CC51A4       120          CALL  Z,PLOT            ;call rel routine
A42A  DD23         130          INC   IX                ;point to next code
A42C  18E2         140          JR    LOOP              ;start again
A42E  F5           150 MOVE:    PUSH  AF                ;save choice
A42F  CD5AA4       160          CALL  COORD             ;get coordinates
A432  CDC0BB       170          CALL  48064             ;move absolute
A435  F1           180          POP   AF                ;restore choice
A436  C9           190          RET
A437  F5           200 DRAW:    PUSH  AF                ;save choice
A438  CD5AA4       210          CALL  COORD             ;get coordinates
A43B  CDF6BB       220          CALL  48118             ;draw absolute
A43E  F1           230          POP   AF                ;restore choice
A43F  C9           240          RET
A440  F5           250 COLOR:   PUSH  AF                ;save choice
A441  DD23         260          INC   IX
A443  DD7E00       270          LD    A,(IX+0)          ;get colour
A446  CDDEBB       280          CALL  48094             ;set graphics pen
A449  DD23         290          INC   IX                ;set pointer
A44B  DD23         300          INC   IX                ;to next
A44D  DD23         310          INC   IX                ;block
A44F  F1           320          POP   AF                ;restore choice
A450  C9           330          RET
A451  F5           340 PLOT:    PUSH  AF                ;save choice
A452  CD5AA4       350          CALL  COORD             ;get coordinates
A455  CDEABB       360          CALL  48106             ;absolute plot
```

```
A458  F1          370          POP  AF                ;restore choice
A459  C9          380          RET
A45A  DD23        390 COORD:   INC  IX                ;set pointer
A45C  DD5E00      400          LD   E,(IX+0)
A45F  DD23        410          INC  IX
A461  DD5600      420          LD   D,(IX+0)          ;x coord. in DE
A464  DD23        430          INC  IX
A466  DD6E00      440          LD   L,(IX+0)
A469  DD23        450          INC  IX
A46B  DD6600      460          LD   H,(IX+0)          ;y coord.in HL
A46E  C9          470          RET

Pass 2 errors: 00

Table used:     81  from   228
```

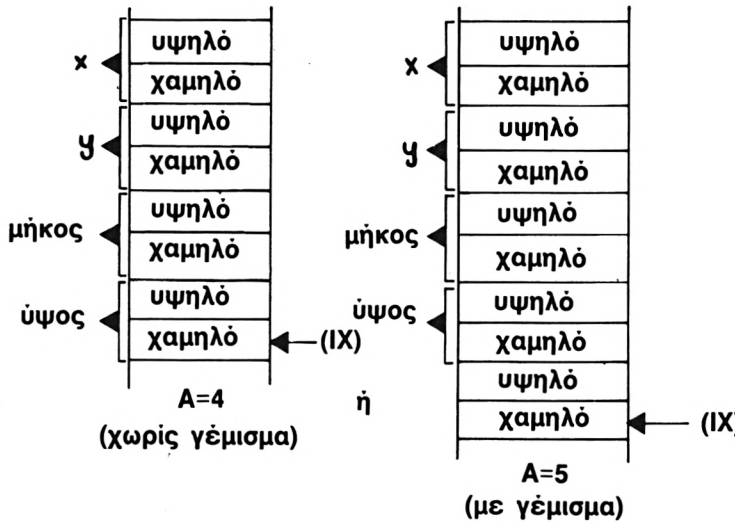
```
10 J=42000
20 RESTORE
30 READ A$
40 IF A$="TELOS" THEN END
50 FOR I=1 TO LEN(A$) STEP 2
60 POKE J,VAL("&"+MID$(A$,I,2))
70 J=J+1
80 NEXT
90 GOTO 30
100 DATA DD7E00FE00C8FE01CC2EA4FE02CC37A4FE03CC40A4FE04CC51A4DD2318E2
110 DATA F5CD5AA4CDC0BBF1C9F5CD5AA4CDF6BBF1C9F5DD23DD7E00CDDEBBDD23DD
120 DATA 23DD23F1C9F5CD5AA4CDEABBF1C9DD23DD5E00DD23DD5600DD23DD6E00DD
130 DATA 23DD6600C9
140 DATA TELOS
```

ΣΧΕΔΙΑΣΗ ΚΑΙ ΓΕΜΙΣΜΑ ΤΕΤΡΑΓΩΝΩΝ

Η παρακάτω ρουτίνα σχεδιάζει τετράγωνα ή παραλληλόγραμμα στην οθόνη, τα οποία - αν θέλουμε - τα γεμίζει με το αντίστοιχο μελάνι γραφικών που χρησιμοποιούμε. Μπορεί να φορτωθεί σε οποιαδήποτε θέση της μνήμης του Amstrad, αρκεί να ληφθεί κατάλληλη μέριμνα για τις υπορουτίνες.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: CALL διεύθυνση, x, y, μήκος, ύψος, [n]. Τα x και y είναι οι συντεταγμένες της κάτω αριστερά κορυφής ενώ το n μπορείτε να μην το δώσετε. Αν δώσετε και n, ανεξάρτητα του τι τιμή μπορεί να έχει θα γεμίσει το τετράγωνο ή παραλληλόγραμμο με χρώμα. Ο IX δείχνει ένα μπλοκ παραμέτρων που φαίνονται στο παρακάτω σχήμα. Για κλήση της ρουτίνας από κώδικα μηχανής πρέπει ο IX να δείχνει ένα τέτοιο μπλοκ και ανάλογα ο A να είναι 4 ή 5 (γέμισμα τετραγ.).

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Όλοι οι καταχωρητές έχουν τυχαίο περιεχόμενο.




```
Hisoft GENA3.1 Assembler. Page      1.

Pass 1 errors: 00

A410                10          org  42000
A410  FE04          20          CP   4              ;four parameters is open
A412  2805          30          JR   Z,OPEN
A414  FE05          40          CP   5              ;if five then fill box
A416  2848          50          JR   Z,CLOSE
A418  C9            60          RET                 ;otherwise quit
A419  CD50A4        70 OPEN:    CALL MOVE
A41C  DD5E02        80          LD   E,(IX+2)
A41F  DD5603        90          LD   D,(IX+3)
A422  210000       100          LD   HL,0
A425  CDF9BB       110          CALL 48121          ;draw bottom line
A428  110000       120          LD   DE,0
A42B  DD6E00       130          LD   L,(IX+0)
A42E  DD6601       140          LD   H,(IX+1)
A431  CDF9BB       150          CALL 48121          ;draw right edge line
A434  CD50A4       160          CALL MOVE
A437  110000       170          LD   DE,0
A43A  DD6E00       180          LD   L,(IX+0)
A43D  DD6601       190          LD   H,(IX+1)
A440  CDF9BB       200          CALL 48121          ;draw left edge line
A443  210000       210          LD   HL,0
A446  DD5E02       220          LD   E,(IX+2)
A449  DD5603       230          LD   D,(IX+3)
A44C  CDF9BB       240          CALL 48121          ;draw top line
A44F  C9           250          RET
A450  DD6E04       260 MOVE:    LD   L,(IX+4)
A453  DD6605       270          LD   H,(IX+5)
A456  DD5E06       280          LD   E,(IX+6)
A459  DD5607       290          LD   D,(IX+7)
A45C  CDC0BB       300          CALL 48064          ;move start position
A45F  C9           310          RET
A460  CDCCBB       320 CLOSE:   CALL 48076          ;get current origin
A463  D5           330          PUSH DE             ;and save it
A464  E5           340          PUSH HL
A465  DD6E06       350          LD   L,(IX+6)
A468  DD6607       360          LD   H,(IX+7)
A46B  DD5E08       370          LD   E,(IX+8)
A46E  DD5609       380          LD   D,(IX+9)
```

```
Pass 2 errors: 00

Table used:    24  from   130
```

```
10 J=42000
20 RESTORE
30 READ A$
40 IF A$="TELOS" THEN END
50 FOR I=1 TO LEN(A$) STEP 2
60 POKE J,VAL("&"+MID$(A$,I,2))
70 J=J+1
80 NEXT
90 GOTO 30
100 DATA CD99BBCD2CBCDD4602F5C5DD4600CD4DBCC1F110F4C9
110 DATA TELOS
```

Σημειώνουμε ότι το scrolling γίνεται απλά με χρήση της ρουτίνας &BC4D (48205) του λειτουργικού συστήματος (βλέπε κεφ. 8). Έχουμε έτσι άλλη μια ευκαιρία να θαυμάσουμε το λειτουργικό του Amstrad που μας δίνει τόσο μεγάλη ευκολία να πετύχουμε αυτό που θέλουμε με την κλήση απλώς μιας ρουτίνας του firmware.

ΠΛΑΓΙΟ SCROLLING

Για να κατανοήσει κανείς το πως μπορεί να γίνει πλάγιο SCROLLING, θα πρέπει να καταλάβει πως αποθηκεύονται τα bytes που σχηματίζουν τους χαρακτήρες στην VIDEO RAM.

Κατ'αρχήν όπως ίσως να ξέρετε η μνήμη που διατίθεται απ'τον υπολογιστή σαν VIDEO RAM βρίσκεται στις θέσεις &C000 μέχρι &FFFF (16K). Χωρίζεται σε 8 blocks των 2K που το καθένα περιέχει πληροφορίες για μία απ'τις 8 γραμμές PIXEL οι οποίες σχηματίζουν κάθε χαρακτήρα στην οθόνη. Τα 8 αυτά blocks φαίνονται στον παρακάτω πίνακα.

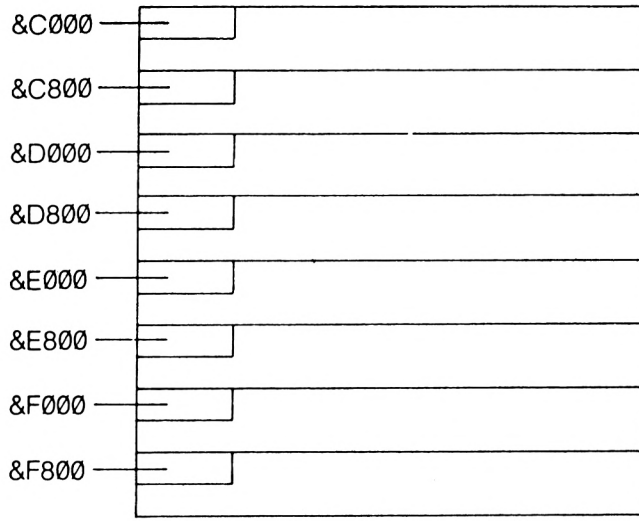
ΒΛΕΠΕ ΣΕΛ. **56**

Δηλαδή ξεκινώντας από &C000 όταν τελειώσει η πρώτη σειρά PIXELS δεν πηγαίνουμε στην αμέσως κατώτερη, αλλά στην πρώτη γραμμή από PIXEL του χαρακτήρα της επόμενης σειράς. Αυτό ακριβώς που βλέπουμε όταν φορτώνεται μια οθόνη στον Amstrad από το κασετόφωνο.

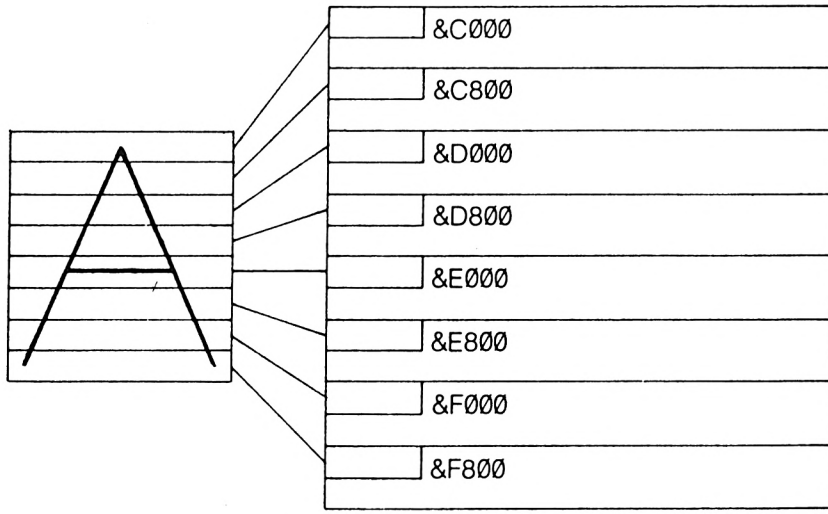
MODE 2

Αλλά ας δούμε πρώτα τι γίνεται στην απλούστερη απεικόνιση του MODE 2. Το πρώτο byte (&C000) απεικονίζει με τα bit του, τα πρώτα 8 pixels του χαρακτήρα που βρίσκεται στην πάνω αριστερά θέση της οθόνης. Το πρώτο byte του επόμενου block (&C800) απεικονίζει τη δεύτερη σειρά από 8 pixels του ίδιου χαρακτήρα κ.ο.κ.

Δηλαδή ο πρώτος χαρακτήρας ορίζεται από 8 bytes της Video RAM



Screen RAM

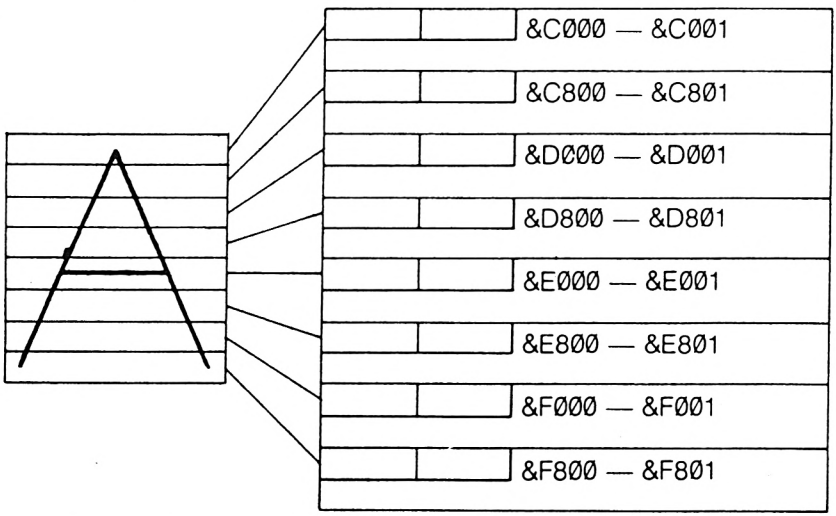


Mode 2

τα οποία φαίνονται στον παραπάνω πίνακα και είναι τα πρώτα bytes των 8 blocks που χωρίζεται η VIDEO RAM όπως αναφέραμε παραπάνω.

MODE 1

Εδώ τα πράγματα γίνονται λιγάκι πιο δύσκολα, λόγω των τεσσάρων χρωμάτων που έχουμε διαθέσιμα σ'αυτό το mode. Τα 8 pixels κάθε οριζόντιας γραμμής pixels ενός χαρακτήρα αποθηκεύονται τώρα σε δύο bytes αντί για ένα όπως γίνεται στο MODE 2. Δηλαδή κάθε PIXEL απεικονίζεται από 2 bit που μπορούν να πάρουν 4 διαφορετικές τιμές (00, 01, 10 ή 11) οι οποίες αντιστοιχούν στα 4 διαθέσιμα χρώματα. Στη συνέχεια βλέπουμε την απεικόνιση του χαρακτήρα στη θέση 1,1 της οθόνης για το MODE 1.



Mode 1

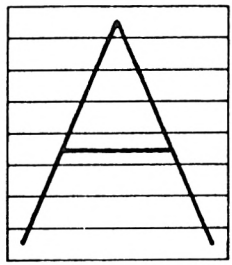
Βλέπουμε λοιπόν εδώ ότι κάθε χαρακτήρας απεικονίζεται από 16 bytes της VIDEO RAM.

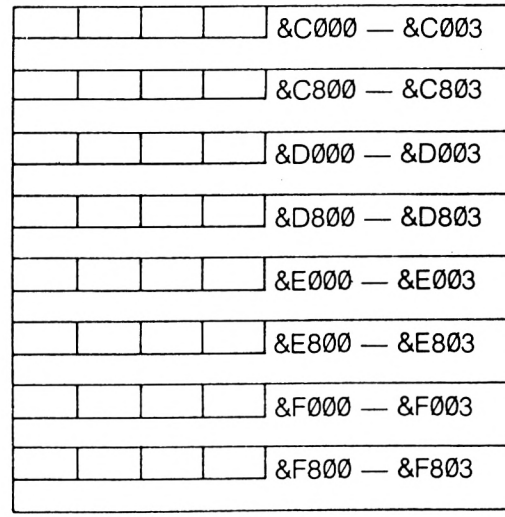
MODE 0

Ανάλογα με τα προηγούμενα εδώ για να απεικονιστούν τα 16 χρώματα πρέπει να χρησιμοποιήσουμε για κάθε PIXEL τέσσερα bit. Οποτέ μια σειρά 8 pixels (μία γραμμή pixels ενός χαρακτήρα) αποθηκεύεται στη μνήμη σε τέσσερα bytes. Παρακάτω βλέπουμε την απεικόνιση του χαρακτήρα στη θέση 1.1 της οθόνης, για το Mode 0.

ΒΛΕΠΕ ΣΕΛ. **58**

Συνεπώς για την απεικόνιση ενός χαρακτήρα στο mode 0 χρειαζό-





Mode 0

μαστε 32 bytes της VIDEO RAM του Amstrad.

Ξέρουμε λοιπόν ότι μία γραμμή χαρακτήρων απεικονίζεται σε 2048 bytes της Video Ram (μπορείτε εύκολα να το βρείτε αν κάνετε το σχετικό πολλαπλασιασμό). Αυτό που χρειαζόμαστε τώρα για να χειριστούμε την οθόνη είναι να ξέρουμε που ξεκινάει η πρώτη γραμμή της οθόνης στη Video Ram, και αυτό γιατί όταν γίνεται κατακόρυφο scrolling η διεύθυνση αυτή αλλάζει. Πάλι εδώ το λειτουργικό σύστημα του Amstrad μας διευκολύνει με τη ρουτίνα στη θέση &BC14 (48154) με την οποία βρίσκουμε τη θέση της Video Ram που απεικονίζεται κάποιος χαρακτήρας (βλέπε κεφ. 8).

Ένας τρόπος να εναλλάσσουμε ή να μετατρέπουμε μια οθόνη, είναι η μεταφορά block μνήμης με την ισχυρή εντολή του Z80 LDIR (ή LDDR). Με αυτό τον τρόπο μπορούμε ακόμα, να δημιουργήσουμε ταχύτατες εναλλαγές οθόνης δηλαδή πραγματικό animattion.

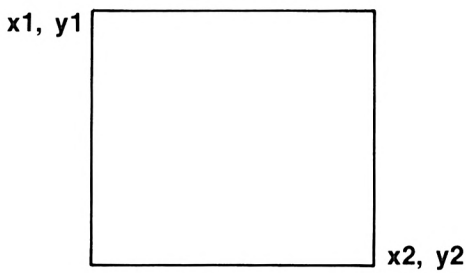
ΑΡΙΣΤΕΡΟ SCROLL 1

Τώρα θα σας παρουσιάσουμε μια ρουτίνα που κάνει αριστερό SCROLLING μιας περιοχής της οθόνης (κατά έναν χαρακτήρα αριστερά). Ότι βγει όμως έξω απ' το αριστερό όριο της οθόνης χάνεται για πάντα.

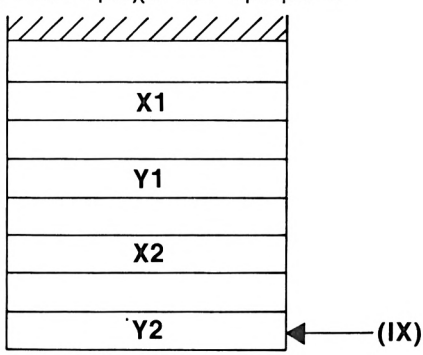
ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: CALL 42000, x1, y1, x2, y2 όπου x1, y1 οι συντεταγμένες του αρι-

στερού άνω άκρου της περιοχής της οθόνης που θέλουμε να κάνουμε scrolling, και x2, y2 οι συντεταγμένες του κάτω δεξιού άκρου αυτής της περιοχής, όπως φαίνεται παρακάτω.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ολοι οι καταχωρητές έχουν τυχαίο περιεχόμενο.



Σε περίπτωση χρήσης της ρουτίνας από κώδικα μηχανής ο IX πρέπει να «δείχνει» σε ένα μπλοκ παραμέτρων όπως ακριβώς φαίνεται στο παρακάτω σχήμα, και ο A να περιέχει τον αριθμό 4.




```
Hisoft GENA3.1 Assembler. Page      1.

Pass 1 errors: 00

A410                10          org  42000
A410  FE04          20          CP   4                    ;four parameters
A412  C0            30          RET  NZ
A413  CD11BC        40          CALL 48145                ;get mode
A416  32B8A4        50          LD   (MODE),A             ;store it
A419  CD9DA4        60          CALL WIDTH                ;characters to move
A41C  32B6A4        70          LD   (CHAR),A
A41F  CD67A4        80          CALL MODEF                ;adjust their value
A422  32B6A4        90          LD   (CHAR),A             ;for the mode
A425  AF           100          XOR  A
A426  32B7A4       110          LD   (CHAR+1),A           ;set upper byte to 0
A429  DD6606       120          LD   H,(IX+6)             ;get top left
```

```
A42C  DD6E04     130          LD   L,(IX+4)        ;coordinates
A42F  CD91A4     140          CALL HGHT            ;no of lines
A432  E5         150 LOOP1:   PUSH HL
A433  C5         160          PUSH BC
A434  CD40A4     170          CALL MOVER           ;scroll1
A437  C1         180          POP  BC
A438  E1         190          POP  HL
A439  CDA8A4     200          CALL PUT             ;fill right column
A43C  2C         210          INC  L
A43D  10F3       220          DJNZ LOOP1
A43F  C9         230          RET
A440  2D         240 MOVER:   DEC  L
A441  25         250          DEC  H
A442  CD1ABC     260          CALL 48154           ;calc screen address
A445  0608       270          LD   B,8             ;8 bytes to move
A447  E5         280          PUSH HL
A448  D1         290          POP  DE
A449  23         295          INC  HL
A44A  CDB1A4     310          CALL FUD
A44D  C5         320 LOOP2:   PUSH BC
A44E  D5         330          PUSH DE
A44F  E5         340          PUSH HL
A450  ED4BB6A4   350          LD   BC,(CHAR)       ;horizontial bytes
A454  EDB0       360          LDIR                 ;do block move
A456  010008     370          LD   BC,2048         ;next char line
A459  E1         380          POP  HL
A45A  09         390          ADD  HL,BC           ;point there
A45B  D1         400          POP  DE
A45C  E5         410          PUSH HL
A45D  D5         420          PUSH DE
A45E  E1         430          POP  HL
A45F  09         440          ADD  HL,BC
A460  E5         450          PUSH HL
A461  D1         460          POP  DE              ;next dest. add
A462  E1         470          POP  HL
A463  C1         480          POP  BC
A464  10E7       490          DJNZ LOOP2           ;repeat for 8 bytes
A466  C9         500          RET
A467  3AB8A4     510 MODEF:   LD   A,(MODE)        ;adjust hor bytes
A46A  FE02       520          CP   2               ;2 means mode 2
A46C  2004       530          JR   NZ,MOD1
A46E  3AB6A4     540          LD   A,(CHAR)
A471  C9         550          RET
A472  FE00       560 MOD1:    CP   0               ;0 means mode 0
A474  2805       570          JR   Z,MOD0
A476  3AB6A4     580          LD   A,(CHAR)
A479  87         590          ADD  A,A             ;two bytes width
A47A  C9         600          RET
A47B  3AB6A4     610 MOD0:    LD   A,(CHAR)
A47E  87         620          ADD  A,A
A47F  87         630          ADD  A,A             ;four bytes width
A480  C9         640          RET
A481  3AB8A4     650 FUD:     LD   A,(MODE)        ;adjust add. to mode
A484  FE02       660          CP   2               ;is it mode 2 ?
A486  C8         670          RET  Z               ;ok as it is
A487  FE01       680          CP   1               ;is it mode 1 ?
A489  2002       690          JR   NZ,MOV0         ;else adjust mode 0
A48B  23         700          INC  HL
A48C  C9         710          RET
A48D  23         720 MOV0:    INC  HL
A48E  23         730          INC  HL
A48F  23         740          INC  HL
A490  C9         750          RET
A491  E5         760 HGHT:    PUSH HL              ;calculate lines
A492  DD7E00     770          LD   A,(IX+0)        ;to be moved
A495  DD6604     780          LD   H,(IX+4)
A498  94         790          SUB  H
A499  3C         800          INC  A
A49A  47         810          LD   B,A
A49B  E1         820          POP  HL
A49C  C9         830          RET
A49D  DD7E02     840 WIDTH:   LD   A,(IX+2)        ;characters to move
A4A0  E5         850          PUSH HL              ;horiz.
A4A1  DD6606     860          LD   H,(IX+6)
```

```
A4A4  94          870          SUB  H
A4A5  3C          880          INC  A
A4A6  E1          890          POP  HL
A4A7  C9          900          RET
A4A8  E5          910 PUT:     PUSH HL                ;fill in
A4A9  DD6602      920          LD   H,(IX+2)          ;right columns
A4AC  CD75BB      930          CALL 47989             ;set cursor
A4AF  3E20        940          LD   A,32
A4B1  CD5ABB      950          CALL 47962             ;print character
A4B4  E1          960          POP  HL
A4B5  C9          970          RET
A4B6              980 CHAR:    DEFS 2
A4B8  00          990 MODE:    DEFB 0

Pass 2 errors: 00

Table used:   159  from   305
```

```
10 J=42000
20 RESTORE
30 READ A$
40 IF A$="TELOS" THEN END
50 FOR I=1 TO LEN(A$) STEP 2
60 POKE J,VAL("&"+MID$(A$,I,2))
70 J=J+1
80 NEXT
90 GOTO 30
100 DATA FE04C0CD11BC32B8A4CD9DA432B6A4CD67A432B6A4AF32B7A4DD6606DD6E
110 DATA 04CD91A4E5C5CD40A4C1E1CDA8A42C10F3C92D25CD1ABC0608E5D123CD81
120 DATA A4C5D5E5ED4BB6A4EDB0010008E109D1E5D5E109E5D1E1C110E7C93AB8A4
130 DATA FE0220043AB6A4C9FE002803AB6A487C93AB6A48787C93AB8A4FE02C8FE
140 DATA 01200223C9232323C9E5DD7E00DD6604943C47E1C9DD7E02E5DD6606943C
150 DATA E1C9E5DD6602CD75BB3E20CD5ABBE1C9000000
160 DATA TELOS
```

Προσέξτε να μην δώσετε άσχετες τιμές στις παραμέτρους X1, Y1, X2, Y2, γιατί μπορεί να έχετε απρόβλεπτα ή και άσχημα αποτελέσματα, και το πρόγραμμά σας να κάνει crash.

Για περισσότερες πληροφορίες σχετικά με τις ρουτίνες του λειτουργικού που χρησιμοποιούνται μπορείτε να ανατρέξετε στο κεφάλαιο 8.

ΑΡΙΣΤΕΡΟ SCROLL 2

Η ρουτίνα αυτή κάνει αριστερό scrolling μιας καθορισμένης περιοχής της οθόνης. Ότι βγαίνει έξω απ' το αριστερό όριο της οθόνης όμως εδώ δε χάνεται, αλλά ξαναμπαίνει από το δεξιό μέρος.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Όμοιες με την προηγούμενη περίπτωση.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Όλοι οι καταχωρητές έχουν τυχαίο περιεχόμενο.

```
  15Hisoft GENA3.1 Assembler. Page      1.

Pass 1 errors: 00

A410                10          ORG  42000
A410  FE04          20          CP   4                 ;four parameters
A412  C0            30          RET  NZ
A413  CD11BC        40          CALL 48145
```

```
A416  3268A5      50          LD   (MODE),A
A419  CDA0A4      60          CALL WIDTH
A41C  3265A5      70          LD   (CHAR),A
A41F  CD6AA4      80          CALL MODEF              ;adjust value
A422  3265A5      90          LD   (CHAR),A
A425  AF          100         XOR  A
A426  3266A5      110         LD   (CHAR+1),A         ;upper byte to 0
A429  DD6606      120         LD   H,(IX+6)           ;top left corner
A42C  DD6E04      130         LD   L,(IX+4)           ;coordinates
A42F  CD94A4      140         CALL HGHT
A432  E5          150 LOOP1:  PUSH HL
A433  C5          160         PUSH BC
A434  CDFAA4      170         CALL CHGET              ;get character
A437  CD43A4      180         CALL MOVE               ;scroll
A43A  C1          190         POP  BC
A43B  E1          200         POP  HL
A43C  CDABA4      210         CALL CHPUT              ;bring char right
A43F  2C          220         INC  L                  ;next line
A440  10F0        230         DJNZ LOOP1              ;repeat
A442  C9          240         RET
A443  2D          250 MOVE:   DEC  L
A444  25          260         DEC  H
A445  CD1ABC      270         CALL 48154              ;screen start pos.
A448  0608        280         LD   B,8                ;8 bytes to be moved
A44A  E5          290         PUSH HL
A44B  D1          300         POP  DE
A44C  23          310         INC  HL
A44D  CD84A4      320         CALL FUD
A450  C5          330 LOOP2:  PUSH BC
A451  D5          340         PUSH DE
A452  E5          350         PUSH HL
A453  ED4B65A5    360         LD   BC,(CHAR)
A457  EDB0        370         LDIR
A459  010008      380         LD   BC,2048
A45C  E1          390         POP  HL
A45D  09          400         ADD  HL,BC              ;next address
A45E  D1          410         POP  DE
A45F  E5          420         PUSH HL
A460  D5          430         PUSH DE
A461  E1          440         POP  HL
A462  09          450         ADD  HL,BC
A463  E5          460         PUSH HL
A464  D1          470         POP  DE                 ;next destination
A465  E1          480         POP  HL
A466  C1          490         POP  BC
A467  10E7        500         DJNZ LOOP2              ;repeat 8 times
A469  C9          510         RET
A46A  3A68A5      520 MODEF:  LD   A,(MODE)
A46D  FE02        530         CP   2                  ;is mode 2?
A46F  2004        540         JR   NZ,MOD1            ;if not so goto mod1
A471  3A65A5      550         LD   A,(CHAR)
A474  C9          560         RET
A475  FE00        570 MOD1:   CP   0                  ;is it mode 0?
A477  2805        580         JR   Z,MOD0             ;if so goto mod0
A479  3A65A5      590         LD   A,(CHAR)
A47C  87          600         ADD  A,A
A47D  C9          610         RET
A47E  3A65A5      620 MOD0:   LD   A,(CHAR)
A481  87          630         ADD  A,A
A482  87          640         ADD  A,A
A483  C9          650         RET
A484  3A68A5      660 FUD:    LD   A,(MODE)
A487  FE02        670         CP   2
A489  C8          680         RET  Z
A48A  FE01        690         CP   1
A48C  2002        700         JR   NZ,MOV0
A48E  23          710         INC  HL
A48F  C9          720         RET
A490  23          730 MOV0:   INC  HL
A491  23          740         INC  HL
A492  23          750         INC  HL
A493  C9          760         RET
A494  E5          770 HGHT:   PUSH HL                 ;no of lines
A495  DD7E00      780         LD   A,(IX+0)           ;to move
```

```
A498  DD6604      790          LD   H,(IX+4)
A49B  94          800          SUB  H
A49C  3C          810          INC  A
A49D  47          820          LD   B,A
A49E  E1          830          POP  HL
A49F  C9          840          RET
A4A0  DD7E02      850 WIDTH:   LD   A,(IX+2)          ;char to be moved
A4A3  E5          860          PUSH HL               ;horiz.
A4A4  DD6606      870          LD   H,(IX+6)
A4A7  94          880          SUB  H
A4A8  3C          890          INC  A
A4A9  E1          900          POP  HL
A4AA  C9          910          RET
A4AB  E5          920 CHPUT:   PUSH HL
A4AC  D5          930          PUSH DE
A4AD  C5          940          PUSH BC
A4AE  DDE5        950          PUSH IX
A4B0  DD6602      960          LD   H,(IX+2)          ;get column
A4B3  25          970          DEC  H
A4B4  2D          980          DEC  L
A4B5  3A68A5      990          LD   A,(MODE)          ;select suitable
A4B8  FE01       1000          CP   1                 ;routine
A4BA  2816       1010          JR   Z,PMOD1           ;mode
A4BC  FE00       1020          CP   0
A4BE  2822       1030          JR   Z,PMOD0
A4C0  CD2EA5     1040          CALL INIT              ;routine for mode 2
A4C3  DD7E00     1050 CHPL:    LD   A,(IX+0)          ;get from buffer
A4C6  77         1060          LD   (HL),A            ;into video RAM
A4C7  DD23       1070          INC  IX                ;next buffer position
A4C9  19         1080          ADD  HL,DE             ;byte of video RAM
A4CA  10F7       1090          DJNZ CHPL              ;repeat for 8 bytes
A4CC  DDE1       1100 POK:     POP  IX                ;restore registers
A4CE  C1         1110          POP  BC
A4CF  D1         1120          POP  DE
A4D0  E1         1130          POP  HL
A4D1  C9         1140          RET
A4D2  CD2EA5     1150 PMOD1:   CALL INIT              ;mode 1
A4D5  CD5DA5     1160 PL1:     CALL HLPUT             ;each char 2 wide
A4D8  CD5DA5     1170          CALL HLPUT
A4DB  2B         1180          DEC  HL
A4DC  2B         1190          DEC  HL
A4DD  19         1200          ADD  HL,DE             ;add next byte (RAM)
A4DE  10F5       1210          DJNZ PL1
A4E0  18EA       1220          JR   POK
A4E2  CD2EA5     1230 PMOD0:   CALL INIT              ;mode 0 is 4 butes
A4E5  CD5DA5     1240 PL0:     CALL HLPUT             ;get  4 bytes
A4E8  CD5DA5     1250          CALL HLPUT
A4EB  CD5DA5     1260          CALL HLPUT
A4EE  CD5DA5     1270          CALL HLPUT
A4F1  2B         1280          DEC  HL
A4F2  2B         1290          DEC  HL
A4F3  2B         1300          DEC  HL
A4F4  2B         1310          DEC  HL
A4F5  19         1320          ADD  HL,DE
A4F6  10ED       1330          DJNZ PL0
A4F8  18D2       1340          JR   POK
A4FA  E5         1350 CHGET:   PUSH HL
A4FB  C5         1360          PUSH BC
A4FC  D5         1370          PUSH DE
A4FD  DDE5       1380          PUSH IX
A4FF  25         1390          DEC  H
A500  2D         1400          DEC  L
A501  3A68A5     1410          LD   A,(MODE)          ;get mode
A504  FE01       1420          CP   1                 ;is it mode 1 etc.
A506  2816       1430          JR   Z,GMOD1
A508  FE00       1440          CP   0
A50A  2831       1450          JR   Z,GMOD0
A50C  CD2EA5     1460          CALL INIT
A50F  7E         1470 CHGL:    LD   A,(HL)
A510  DD7700     1480          LD   (IX),A
A513  DD23       1490          INC  IX
A515  19         1500          ADD  HL,DE
A516  10F7       1510          DJNZ CHGL
A518  DDE1       1520 CHOK:    POP  IX
```

```
A51A  D1          1530          POP  DE
A51B  C1          1540          POP  BC
A51C  E1          1550          POP  HL
A51D  C9          1560          RET
A51E  CD2EA5      1570 GMOD1:   CALL INIT                ;routine for mode 1
A521  CD55A5      1580 ML1:     CALL HLGET
A524  CD55A5      1590          CALL HLGET
A527  2B          1600          DEC  HL
A528  2B          1610          DEC  HL
A529  19          1620          ADD  HL,DE              ;next RAM
A52A  10F5        1630          DJNZ ML1
A52C  18EA        1640          JR   CHOK
A52E  CD1ABC      1650 INIT:    CALL 48154              ;get next char matrix
A531  DD2169A5    1660          LD   IX,TEMP            ;start char square
A535  0608        1670          LD   B,8
A537  110008      1680          LD   DE,2048
A53A  C9          1690          RET
A53B  10E4        1700          DJNZ ML1
A53D  CD2EA5      1710 GMOD0:   CALL INIT               ;routine for mode 0
A540  CD55A5      1720 ML0:     CALL HLGET
A543  CD55A5      1730          CALL HLGET
A546  CD55A5      1740          CALL HLGET
A549  CD55A5      1750          CALL HLGET
A54C  2B          1760          DEC  HL
A54D  2B          1770          DEC  HL
A54E  2B          1780          DEC  HL
A54F  2B          1790          DEC  HL
A550  19          1800          ADD  HL,DE
A551  10ED        1810          DJNZ ML0
A553  18C3        1820          JR   CHOK
A555  7E          1830 HLGET:   LD   A,(HL)              ;transfer bytes from
A556  DD7700      1840          LD   (IX+0),A           ;video RAM
A559  23          1850          INC  HL
A55A  DD23        1860          INC  IX
A55C  C9          1870          RET
A55D  DD7E00      1880 HLPUT:   LD   A,(IX+0)           ;bytes from buffer
A560  77          1890          LD   (HL),A             ;to video RAM
A561  23          1900          INC  HL
A562  DD23        1910          INC  IX
A564  C9          1920          RET
A565              1930 CHAR:    DEFS 2
A567  00          1940 CHAR2:   DEFB 0
A568  00          1950 MODE:    DEFB 0
A569              1960 TEMP:    DEFS 40

Pass 2 errors: 00

Table used:    361  from   466
```

```
10 J=42000
20 RESTORE
30 READ A$
40 IF A$="TELOS" THEN END
50 FOR I=1 TO LEN(A$) STEP 2
60 POKE J,VAL("&"+MID$(A$,I,2))
70 J=J+1
80 NEXT
90 GOTO 30
100 DATA FE04C0CD11BC3268A5CDA0A43265A5CD6AA43265A5AF3266A5DD6606DD6E
110 DATA 04CD94A4E5C5CDFAA4CD43A4C1E1CDABA42C10F0C92D25CD1ABC0608E5D1
120 DATA 23CD84A4C5D5E5ED4B65A5EDB0010008E109D1E5D5E109E5D1E1C110E7C9
130 DATA 3A68A5FE0220043A65A5C9FE002803A65A587C93A65A58787C93A68A5FE
140 DATA 02C8FE01200223C9232323C9E5DD7E00DD6604943C47E1C9DD7E02E5DD66
150 DATA 06943CE1C9E5D5C5DDE5DD6602252D3A68A5FE012816FE002822CD2EA5DD
160 DATA 7E0077DD231910F7DDE1C1D1E1C9CD2EA5CD5DA5CD5DA52B2B1910F518EA
170 DATA CD2EA5CD5DA5CD5DA5CD5DA5CD5DA52B2B2B2B1910ED18D2E5C5D5DDE525
180 DATA 2D3A68A5FE012816FE002831CD2EA57EDD7700DD231910F7DDE1D1C1E1C9
190 DATA CD2EA5CD55A5CD55A52B2B1910F518EACD1ABCDD2169A5060811000BC910
200 DATA E4CD2EA5CD55A5CD55A5CD55A5CD55A52B2B2B2B1910ED18C37EDD770023
210 DATA DD23C9DD7E007723DD23C9
220 DATA TELOS
```

ΔΕΞΙΟ SCROLL 1

Αυτή η ρουτίνα κάνει scrolling μια προκαθορισμένη περιοχή της οθόνης, κατά 1 χαρακτήρα προς τα δεξιά. Όπως και στην πρώτη ρουτίνα του αριστερού SCROLLING, ότι βγαίνει έξω απ'το δεξιό άκρο της οθόνης χάνεται.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Όπως στο αριστερό SCROLL.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Όλοι οι καταχωρητές έχουν τυχαίο περιεχόμενο.

```
Hisoft GENA3.1 Assembler. Page      1.

Pass 1 errors: 00

A410                10          org  42000
A410  FE04          20          CP   4                    ;four parameters
A412  C0            30          RET  NZ
A413  CD11BC        40          CALL 48145                ;get mode
A416  32C6A4        50          LD   (MODE),A             ;save it
A419  CDABA4        60          CALL WIDTH                ;characters to move
A41C  32C4A4        70          LD   (CHAR),A
A41F  CD75A4        80          CALL MODEF                ;adjust value to mode
A422  32C4A4        90          LD   (CHAR),A
A425  AF           100          XOR  A
A426  32C5A4       110          LD   (CHAR+1),A           ;upper byte to 0
A429  DD6602       120          LD   H,(IX+2)             ;get top right
A42C  DD6E04       130          LD   L,(IX+4)             ;coordinates
A42F  CD9FA4       140          CALL HGHT                 ;no of lines
A432  E5           150 LOOP1:   PUSH HL
A433  C5           160          PUSH BC
A434  CD40A4       170          CALL MOVE                 ;scroll
A437  C1           180          POP  BC
A438  E1           190          POP  HL
A439  CDB6A4       200          CALL PUT                  ;fill spaces in right
A43C  2C           210          INC  L                    ;next line
A43D  10F3         220          DJNZ LOOP1
A43F  C9           230          RET
A440  2D           240 MOVE:    DEC  L
A441  25           250          DEC  H
A442  CD1ABC       260          CALL 48154                ;char pos.in video RAM
A445  3AC6A4       270          LD   A,(MODE)             ;adjust with mode
A448  FE02         280          CP   2
A44A  2807         290          JR   Z,OUT
A44C  FE01         300          CP   1
A44E  2802         310          JR   Z,OUT1
A450  23           320          INC  HL
A451  23           330          INC  HL
A452  23           340 OUT1:    INC  HL
A453  0608         350 OUT:     LD   B,8                  ;8 bytes to move
A455  E5           360          PUSH HL
A456  D1           370          POP  DE
A457  2B           380          DEC  HL
A458  CD8FA4       390          CALL FUD
A45B  C5           400 LOOP2:   PUSH BC
A45C  D5           410          PUSH DE
A45D  E5           420          PUSH HL
A45E  ED4BC4A4     430          LD   BC,(CHAR)            ;horiz. bytes
A462  EDB8         440          LDDR                      ;move block
A464  010008       450          LD   BC,2048              ;next char address
A467  E1           460          POP  HL
A468  09           470          ADD  HL,BC                ;next source address
A469  D1           480          POP  DE
A46A  E5           490          PUSH HL
A46B  D5           500          PUSH DE
A46C  E1           510          POP  HL
A46D  09           520          ADD  HL,BC
A46E  E5           530          PUSH HL
A46F  D1           540          POP  DE
```

```
A470  E1          550          POP  HL
A471  C1          560          POP  BC
A472  10E7        570          DJNZ LOOP2              ;repeat for 8 bytes
A474  C9          580          RET
A475  3AC6A4      590 MODEF:   LD   A,(MODE)           ;cal. horizont. bytes
A478  FE02        600          CP   2                  ;to be moved
A47A  2004        610          JR   NZ,MOD1
A47C  3AC4A4      620          LD   A,(CHAR)
A47F  C9          630          RET
A480  FE00        640 MOD1:    CP   0
A482  2805        650          JR   Z,MOD0
A484  3AC4A4      660          LD   A,(CHAR)
A487  87          670          ADD  A,A                ;mode 1 needs 2 bytes
A488  C9          680          RET
A489  3AC4A4      690 MOD0:    LD   A,(CHAR)
A48C  87          700          ADD  A,A
A48D  87          710          ADD  A,A                ;mode 0 needs 4 bytes
A48E  C9          720          RET
A48F  3AC6A4      730 FUD:     LD   A,(MODE)           ;calc. source address
A492  FE02        740          CP   2
A494  C8          750          RET  Z
A495  FE01        760          CP   1
A497  2002        770          JR   NZ,MOV0
A499  2B          780          DEC  HL
A49A  C9          790          RET
A49B  2B          800 MOV0:    DEC  HL
A49C  2B          810          DEC  HL
A49D  2B          820          DEC  HL
A49E  C9          830          RET
A49F  E5          840 HGHT:    PUSH HL                 ;calc. no of lines
A4A0  DD7E00      850          LD   A,(IX+0)
A4A3  DD6604      860          LD   H,(IX+4)
A4A6  94          870          SUB  H
A4A7  3C          880          INC  A
A4A8  47          890          LD   B,A
A4A9  E1          900          POP  HL
A4AA  C9          910          RET
A4AB  DD7E02      920 WIDTH:   LD   A,(IX+2)           ;no of characters
A4AE  E5          930          PUSH HL                 ;to move hor.
A4AF  DD6606      940          LD   H,(IX+6)
A4B2  94          950          SUB  H
A4B3  3C          960          INC  A
A4B4  E1          970          POP  HL
A4B5  C9          980          RET
A4B6  E5          990 PUT:     PUSH HL                 ;put space in
A4B7  DD6606     1000          LD   H,(IX+6)           ;left column
A4BA  CD75BB     1010          CALL 47989
A4BD  3E20       1020          LD   A,32
A4BF  CD5ABB     1030          CALL 47962
A4C2  E1         1040          POP  HL
A4C3  C9         1050          RET
A4C4             1060 CHAR:    DEFS 2
A4C6  00         1070 MODE:    DEFB 0
Pass 2 errors: 00

Table used:    179  from    304
```

```
10 J=42000
20 RESTORE
30 READ A$
40 IF A$="TELOS" THEN END
50 FOR I=1 TO LEN(A$) STEP 2
60 POKE J,VAL("&"+MID$(A$,I,2))
70 J=J+1
80 NEXT
90 GOTO 30
100 DATA FE04C0CD11BC32C6A4CDABA432C4A4CD75A432C4A4AF32C5A4DD6602DD6E
110 DATA 04CD9FA4E5C5CD40A4C1E1CDB6A42C10F3C92D25CD1ABC3AC6A4FE022807
120 DATA FE012802232323060BE5D12BCD8FA4C5D5E5ED4BC4A4EDB801000BE109D1
130 DATA E5D5E109E5D1E1C110E7C93AC6A4FE0220043AC4A4C9FE0028053AC4A487
140 DATA C93AC4A48787C93AC6A4FE02C8FE0120022BC92B2B2BC9E5DD7E00DD6604
150 DATA 943C47E1C9DD7E02E5DD6606943CE1C9E5DD6606CD75BB3E20CD5ABBE1C9
160 DATA TELOS
```

ΔΕΞΙΟ SCROLL 2

Αυτή η ρουτίνα κάνει ότι και η προηγούμενη με την μόνη διαφορά ότι βγαίνει έξω από το δεξιό όριο της οθόνης μπαίνει αντίστοιχα από τα αριστερά.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Όπως και προηγουμένως.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Όλοι οι καταχωρητές έχουν τυχαίο περιεχόμενο.

```
  220
  230
  240 Hisoft GENA3.1 Assembler. Page      1.

Pass 1 errors: 00

A410                10          org   42000
A410  FE04          20          CP    4                 ;four parameters
A412  C0            30          RET   NZ
A413  CD11BC        40          CALL  48145             ;get mode
A416  3276A5        50          LD    (MODE),A
A419  CDAEA4        60          CALL  WIDTH             ;no of char. to move
A41C  3273A5        70          LD    (CHAR),A
A41F  CD78A4        80          CALL  MODEF             ;adjust value
A422  3273A5        90          LD    (CHAR),A          ;with mode
A425  AF           100          XOR   A
A426  3274A5       110          LD    (CHAR+1),A        ;zero upper byte
A429  DD6602       120          LD    H,(IX+2)          ;get to right
A42C  DD6E04       130          LD    L,(IX+4)          ;coordinates
A42F  CDA2A4       140          CALL  HGHT              ;no of lines
A432  E5           150 LOOP1:   PUSH  HL
A433  C5           160          PUSH  BC
A434  CD08A5       170          CALL  CHGET
A437  CD43A4       180          CALL  MOVE              ;scroll
A43A  C1           190          POP   BC
A43B  E1           200          POP   HL
A43C  CDB9A4       210          CALL  CHPUT             ;char to left column
A43F  2C           220          INC   L                 ;next line
A440  10F0         230          DJNZ  LOOP1             ;repeat
A442  C9           240          RET
A443  2D           250 MOVE:    DEC   L
A444  25           260          DEC   H
A445  CD1ABC       270          CALL  48154             ;calc screen for chr pos
A448  3A76A5       280          LD    A,(MODE)          ;adjust to mode
A44B  FE02         290          CP    2                 ;is mode 2 ?
A44D  2807         300          JR    Z,OUT
A44F  FE01         310          CP    1
A451  2802         320          JR    Z,OUT1
A453  23           330          INC   HL
A454  23           340          INC   HL
A455  23           350 OUT1:    INC   HL
A456  0608         360 OUT:     LD    B,8               ;8 bytes to move
A458  E5           370          PUSH  HL
A459  D1           380          POP   DE
A45A  2B           390          DEC   HL
A45B  CD92A4       400          CALL  FUD
A45E  C5           410 LOOP2:   PUSH  BC
A45F  D5           420          PUSH  DE
A460  E5           430          PUSH  HL
A461  ED4B73A5     440          LD    BC,(CHAR)         ;hor. bytes
A465  EDB8         450          LDDR                    ;move the block
A467  010008       460          LD    BC,2048           ;next line of chr
A46A  E1           470          POP   HL
A46B  09           480          ADD   HL,BC             ;next source address
A46C  D1           490          POP   DE
A46D  E5           500          PUSH  HL
A46E  D5           510          PUSH  DE
A46F  E1           520          POP   HL
A470  09           530          ADD   HL,BC
```

```
A471  E5        540          PUSH HL
A472  D1        550          POP  DE
A473  E1        560          POP  HL
A474  C1        570          POP  BC
A475  10E7      580          DJNZ LOOP2          ;repeat for 8 bytes
A477  C9        590          RET
A478  3A76A5    600 MODEF:   LD   A,(MODE)       ;adjust hor bytes
A47B  FE02      610          CP   2              ;to current mode
A47D  2004      620          JR   NZ,MOD1
A47F  3A73A5    630          LD   A,(CHAR)
A482  C9        640          RET
A483  FE00      650 MOD1:    CP   0
A485  2805      660          JR   Z,MOD0
A487  3A73A5    670          LD   A,(CHAR)
A48A  87        680          ADD  A,A            ;mode 1 - 2 bytes
A48B  C9        690          RET
A48C  3A73A5    700 MOD0:    LD   A,(CHAR)
A48F  87        710          ADD  A,A
A490  87        720          ADD  A,A            ;mode 0 - 4 bytes
A491  C9        730          RET
A492  3A76A5    740 FUD:     LD   A,(MODE)       ;adjust screen address
A495  FE02      750          CP   2
A497  C8        760          RET  Z
A498  FE01      770          CP   1
A49A  2002      780          JR   NZ,MOV0
A49C  2B        790          DEC  HL
A49D  C9        800          RET
A49E  2B        830 MOV0:    DEC  HL
A49F  2B        840          DEC  HL
A4A0  2B        850          DEC  HL
A4A1  C9        860          RET
A4A2  E5        870 HGHT:    PUSH HL             ;no of lines to move
A4A3  DD7E00    880          LD   A,(IX+0)
A4A6  DD6604    890          LD   H,(IX+4)
A4A9  94        900          SUB  H
A4AA  3C        910          INC  A
A4AB  47        920          LD   B,A
A4AC  E1        930          POP  HL
A4AD  C9        940          RET
A4AE  DD7E02    950 WIDTH:   LD   A,(IX+2)       ;characters to move
A4B1  E5        960          PUSH HL             ;horizont.
A4B2  DD6606    970          LD   H,(IX+6)
A4B5  94        980          SUB  H
A4B6  3C        990          INC  A
A4B7  E1        1000         POP  HL
A4B8  C9        1010         RET
A4B9  E5        1020 CHPUT:  PUSH HL
A4BA  D5        1030         PUSH DE
A4BB  C5        1040         PUSH BC
A4BC  DDE5      1050         PUSH IX
A4BE  DD6606    1060         LD   H,(IX+6)       ;colour to fill
A4C1  25        1070         DEC  H
A4C2  2D        1080         DEC  L
A4C3  3A76A5    1090         LD   A,(MODE)
A4C6  FE01      1100         CP   1              ;check current mode
A4C8  2816      1110         JR   Z,PMOD1
A4CA  FE00      1120         CP   0
A4CC  2822      1130         JR   Z,PMOD0
A4CE  CD3CA5    1140         CALL INIT
A4D1  DD7E00    1150 CHPL:   LD   A,(IX+0)       ;get from buffer
A4D4  77        1160         LD   (HL),A         ;put in video RAM
A4D5  DD23      1170         INC  IX             ;next
A4D7  19        1180         ADD  HL,DE          ;next byte in video RAM
A4D8  10F7      1190         DJNZ CHPL           ;repeat for 8 bytes
A4DA  DDE1      1200 POK:    POP  IX             ;restore registers
A4DC  C1        1210         POP  BC
A4DD  D1        1220         POP  DE
A4DE  E1        1230         POP  HL
A4DF  C9        1240         RET
A4E0  CD3CA5    1250 PMOD1:  CALL INIT           ;mode 1 routine
A4E3  CD6BA5    1260 PL1:    CALL HLPUT          ;char of two bytes
A4E6  CD6BA5    1270         CALL HLPUT
A4E9  2B        1280         DEC  HL
A4EA  2B        1290         DEC  HL
```

```
A4EB  19        1300          ADD  HL,DE          ;next byte in video RAM
A4EC  10F5      1310          DJNZ PL1
A4EE  18EA      1320          JR   POK
A4F0  CD3CA5    1330 PMOD0:   CALL INIT           ;mode 0 routine
A4F3  CD6BA5    1340 PL0:     CALL HLPUT          ;4 bytes for each
A4F6  CD6BA5    1350          CALL HLPUT
A4F9  CD6BA5    1360          CALL HLPUT
A4FC  CD6BA5    1370          CALL HLPUT
A4FF  2B        1380          DEC  HL
A500  2B        1390          DEC  HL
A501  2B        1400          DEC  HL
A502  2B        1410          DEC  HL
A503  19        1420          ADD  HL,DE          ;next char row
A504  10ED      1430          DJNZ PL0
A506  18D2      1440          JR   POK
A508  E5        1450 CHGET:   PUSH HL
A509  C5        1460          PUSH BC
A50A  D5        1470          PUSH DE
A50B  DDE5      1480          PUSH IX
A50D  25        1490          DEC  H
A50E  2D        1500          DEC  L
A50F  3A76A5    1510          LD   A,(MODE)       ;get mode
A512  FE01      1520          CP   1              ;is it mode 1
A514  2816      1530          JR   Z,GMOD1        ;call rel. routine
A516  FE00      1540          CP   0              ;is it mode 0 ?
A518  2831      1550          JR   Z,GMOD0
A51A  CD3CA5    1560          CALL INIT           ;mode 2, 1 byte
A51D  7E        1570 CHGL:    LD   A,(HL)
A51E  DD7700    1580          LD   (IX+0),A
A521  DD23      1590          INC  IX
A523  19        1600          ADD  HL,DE
A524  10F7      1610          DJNZ CHGL
A526  DDE1      1620 CHOK:    POP  IX             ;restore registers
A528  D1        1630          POP  DE
A529  C1        1640          POP  BC
A52A  E1        1650          POP  HL
A52B  C9        1660          RET
A52C  CD3CA5    1670 GMOD1:   CALL INIT           ;routine for mode 1
A52F  CD63A5    1680 ML1:     CALL HLGET          ;2 bytes video RAM
A532  CD63A5    1690          CALL HLGET          ;for each line
A535  2B        1700          DEC  HL
A536  2B        1710          DEC  HL
A537  19        1720          ADD  HL,DE          ;next video RAM address
A538  10F5      1730          DJNZ ML1
A53A  18EA      1740          JR   CHOK

A53C  CD1ABC    1750 INIT:    CALL 48154
A53F  DD2177A5  1760          LD   IX,TEMP
A543  0608      1770          LD   B,8
A545  110008    1780          LD   DE,2048
A548  C9        1790          RET
A549  10E4      1800          DJNZ ML1
A54B  CD3CA5    1810 GMOD0:   CALL INIT           ;routine for mode 0
A54E  CD63A5    1820 ML0:     CALL HLGET          ;each char 4 bytes
A551  CD63A5    1830          CALL HLGET
A554  CD63A5    1840          CALL HLGET
A557  CD63A5    1850          CALL HLGET
A55A  2B        1860          DEC  HL
A55B  2B        1870          DEC  HL
A55C  2B        1880          DEC  HL
A55D  2B        1890          DEC  HL
A55E  19        1900          ADD  HL,DE
A55F  10ED      1910          DJNZ ML0
A561  18C3      1920          JR   CHOK
A563  7E        1930 HLGET:   LD   A,(HL)          ;transfer byte
A564  DD7700    1940          LD   (IX+0),A       ;video RAM to buffer
A567  23        1950          INC  HL
A568  DD23      1960          INC  IX
A56A  C9        1970          RET
A56B  DD7E00    1980 HLPUT:   LD   A,(IX+0)       ;byte from buffer
A56E  77        1990          LD   (HL),A         ;to video RAM
A56F  23        2000          INC  HL
A570  DD23      2010          INC  IX
A572  C9        2020          RET
A573            2030 CHAR:    DEFS 2
```

```
A575  00          2040 CHAR2:   DEFB 0
A576  00          2050 MODE:    DEFB 0
A577              2060 TEMP:    DEFS 40

Pass 2 errors: 00

Table used:   382  from   505
```

```
10 J=42000
20 RESTORE
30 READ A$
40 IF A$="TELOS" THEN END
50 FOR I=1 TO LEN(A$) STEP 2
60 POKE J,VAL("&"+MID$(A$,I,2))
70 J=J+1
80 NEXT
90 GOTO 30
100 DATA FE04C0CD11BC3276A5CDAEA43273A5CD78A43273A5AF3274A5DD6602DD6E
110 DATA 04CDA2A4E5C5CD08A5CD43A4C1E1CDB9A42C10F0C92D25CD1ABC3A76A5FE
120 DATA 022807FE012802232323060BE5D12BCD92A4C5D5E5ED4B73A5EDB8010008
130 DATA E109D1E5D5E109E5D1E1C110E7C93A76A5FE0220043A73A5C9FE0028053A
140 DATA 73A587C93A73A587B7C93A76A5FE02C8FE0120022BC92B2B2BC9E5DD7E00
150 DATA DD6604943C47E1C9DD7E02E5DD6606943CE1C9E5D5C5DDE5DD6606252D3A
160 DATA 76A5FE012816FE002822CD3CA5DD7E0077DD231910F7DDE1C1D1E1C9CD3C
170 DATA A5CD6BA5CD6BA52B2B1910F518EACD3CA5CD6BA5CD6BA5CD6BA5CD6BA52B
180 DATA 2B2B2B1910ED18D2E5C5D5DDE5252D3A76A5FE012816FE002831CD3CA57E
190 DATA DD7700DD231910F7DDE1D1C1E1C9CD3CA5CD63A5CD63A52B2B1910F518EA
200 DATA CD1ABCDD2177A50608110008C910E4CD3CA5CD63A5CD63A5CD63A5CD63A5
210 DATA 2B2B2B2B1910ED18C37EDD770023DD23C9DD7E00723DD23C9
220 DATA TELOS
```

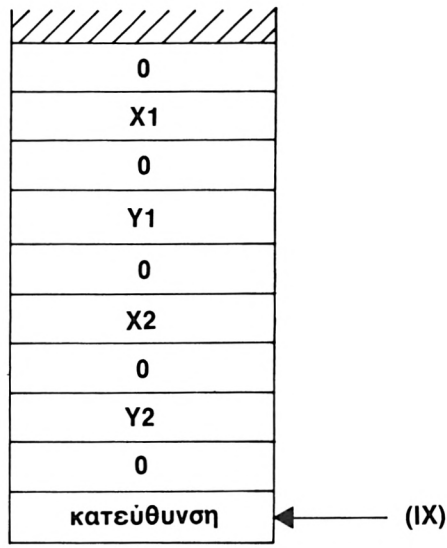
Προσέξτε: Όλες οι παραπάνω ρουτίνες για οριζόντιο scrolling δεν μπορούν να μεταφερθούν σε άλλη θέση μνήμης έτσι όπως είναι. Γι' αυτό το λόγο και είναι καλύτερα να τις χρησιμοποιήσετε εκεί ακριβώς που βρίσκονται στη θέση 42000, παρά να προσπαθήσετε να τις μετατρέψετε.

ΚΑΤΑΚΟΡΥΦΟ SCROLLING ΜΕ ΕΠΑΝΑΦΟΡΑ

Κλείνοντας τη σειρά από ρουτίνες που κάνουν SCROLLING, σας παρουσιάζουμε μία ρουτίνα που κάνει κατακόρυφο scrolling μιας περιοχής της οθόνης ενώ ταυτόχρονα μεταφέρει ότι βγαίνει έξω από το άνω ή το κάτω άκρο της οθόνης, από την αντίθετη πλευρά. Μπορείτε να τη χρησιμοποιήσετε σε οποιοδήποτε MODE.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: CALL 41000, X1, Y1, X2, Y2, κατεύθυνση όπου X1 είναι συντεταγμένη του αριστερού άκρου της περιοχής, το Y1 του άνω άκρου, το X2 του δεξιού άκρου και το Y2 του κάτω άκρου της περιοχής που θέλουμε να κάνουμε SCROLLING. Η μεταβλητή της «κατεύθυνσης», παίρνει τιμή 0 για scroll προς τα κάτω ή 1 για scroll προς τα άνω

αντίστοιχα. Για χρήση της ρουτίνας από κώδικα μηχανής πρέπει ο καταχωρητής A να έχει τιμή 5 και ο IX να «δείχνει» στο παρακάτω block παραμέτρων.



ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Όλοι οι καταχωρητές έχουν τυχαίο περιεχόμενο.

```
Hisoft GENA3.1 Assembler. Page     1.

Pass 1 errors: 00

A028                10          ORG  41000
A028  FE05          20          CP   5                    ;five parameters
A02A  C0            30          RET  NZ
A02B  CD99BB        40          CALL 48025                ;get paper
A02E  CD2CBC        50          CALL 48172                ;encode the ink
A031  32B9A3        60          LD   (PAPER),A            ;then save it
A034  DD7E00        70          LD   A,(IX+0)             ;get choice up/down
A037  FE00          80          CP   0                    ;is it down ?
A039  2826          90          JR   Z,DOWN               ;call relative routine
A03B  CD87A0        100 UP:     CALL WID                  ;width of window
A03E  DD6608        110         LD   H,(IX+8)
A041  DD6E06        120         LD   L,(IX+6)
A044  3E01          130         LD   A,1
A046  32FBA0        140         LD   (DIR),A
A049  CD97A0        150         CALL GET                  ;save the top line
A04C  CDDFA0        160         CALL SCROL                ;do upwards scroll
A04F  CD87A0        170         CALL WID
A052  DD6608        180         LD   H,(IX+8)
A055  DD6E02        190         LD   L,(IX+2)
A058  3E00          200         LD   A,0
A05A  32FBA0        210         LD   (DIR),A
A05D  CD97A0        220         CALL GET                  ;print at bottom
A060  C9            230         RET
A061  CD87A0        240 DOWN:   CALL WID
A064  DD6608        250         LD   H,(IX+8)
A067  DD6E02        260         LD   L,(IX+2)
```

```
A06A  3E01      270          LD   A,1              ;bottom line
A06C  32FBA0    280          LD   (DIR),A
A06F  CD97A0    290          CALL GET              ;get it
A072  CDDFA0    300          CALL SCROL            ;do scroll down 1 line
A075  CD87A0    310          CALL WID
A078  DD6608    320          LD   H,(IX+8)
A07B  DD6E06    330          LD   L,(IX+6)
A07E  3E00      340          LD   A,0
A080  32FBA0    350          LD   (DIR),A
A083  CD97A0    360          CALL GET              ;print at top line
A086  C9        370          RET
A087  E5        380 WID:     PUSH HL               ;save register
A088  DD7E04    390          LD   A,(IX+4)
A08B  DD6608    400          LD   H,(IX+8)
A08E  94        410          SUB  H
A08F  47        420          LD   B,A
A090  3C        430          INC  A
A091  32FCA0    440          LD   (WIDTH),A
A094  04        450          INC  B                ;width in B register
A095  E1        460          POP  HL               ;recover HL
A096  C9        470          RET
A097  C5        480 GET:     PUSH BC               ;reserve registers
A098  D5        490          PUSH DE
A099  E5        500          PUSH HL
A09A  DDE5      510          PUSH IX
A09C  DD21FDA0  520          LD   IX,BUF
A0A0  0608      530          LD   B,8
A0A2  C5        540          PUSH BC
A0A3  25        550          DEC  H
A0A4  2D        560          DEC  L
A0A5  CD1ABC    570          CALL 48154            ;char pos in video RAM
A0A8  ED43F9A0  580          LD   (NWID),BC        ;char width in NWID
A0AC  110008    590          LD   DE,2048          ;next line
A0AF  C1        600          POP  BC
A0B0  C5        610 LOOP1:   PUSH BC
A0B1  3AFCA0    620          LD   A,(WIDTH)        ;get no of char
A0B4  47        630          LD   B,A
A0B5  E5        640          PUSH HL               ;save start ad. line
A0B6  C5        650 LOOP2:   PUSH BC               ;save no of char
A0B7  ED4BF9A0  660          LD   BC,(NWID)        ;width of char in bytes
A0BB  3AFBA0    670 LOOP3:   LD   A,(DIR)          ;byte from video RAM
A0BE  FE00      680          CP   0                ;to the buffer
A0C0  2817      690          JR   Z,PUT            ;according to value
A0C2  7E        700          LD   A,(HL)           ;of (DIR)
A0C3  DD7700    710          LD   (IX+0),A
A0C6  DD23      720 CHOK:    INC  IX               ;next byte
A0C8  23        730          INC  HL
A0C9  10F0      740          DJNZ LOOP3            ;do a screen line
A0CB  C1        750          POP  BC
A0CC  10E8      760          DJNZ LOOP2            ;each char wind
A0CE  E1        770          POP  HL
A0CF  19        780          ADD  HL,DE            ;next line
A0D0  C1        790          POP  BC
A0D1  10DD      800          DJNZ LOOP1            ;repeat
A0D3  DDE1      810          POP  IX
A0D5  E1        820          POP  HL
A0D6  D1        830          POP  DE
A0D7  C1        840          POP  BC
A0D8  C9        850          RET
A0D9  DD7E00    860 PUT:     LD   A,(IX)
A0DC  77        870          LD   (HL),A
A0DD  18E7      880          JR   CHOK
A0DF  DD6608    890 SCROL:   LD   H,(IX+8)         ;get area
A0E2  DD6E06    900          LD   L,(IX+6)         ;to scroll
A0E5  DD5604    910          LD   D,(IX+4)
A0E8  DD5E02    920          LD   E,(IX+2)
A0EB  DD4600    930          LD   B,(IX+0)
A0EE  3AB9A3    940          LD   A,(PAPER)
A0F1  25        950          DEC  H
A0F2  2D        960          DEC  L
A0F3  15        970          DEC  D
A0F4  1D        980          DEC  E
A0F5  CD50BC    990          CALL 48208            ;do the scroll
A0F8  C9       1000          RET
```

```
A0F9              1010 NWID:    DEFS 2
A0FB  00          1020 DIR:     DEFB 0
A0FC  00          1030 WIDTH:   DEFB 0
A0FD              1040 BUF:     DEFS 700
A3B9  00          1050 PAPER:   DEFB 0

Pass 2 errors: 00

Table used:   177  from   325
```

```
10 J=41000
20 RESTORE
30 READ A$
40 IF A$="TELOS" THEN END
50 FOR I=1 TO LEN(A$) STEP 2
60 POKE J,VAL("&"+MID$(A$,I,2))
70 J=J+1
80 NEXT
90 GOTO 30
100 DATA FE05C0CD99BBCD2CBC32B9A3DD7E00FE002826CD87A0DD6608DD6E063E01
110 DATA 32FBA0CD97A0CDDFA0CD87A0DD6608DD6E023E0032FBA0CD97A0C9CD87A0
120 DATA DD6608DD6E023E0132FBA0CD97A0CDDFA0CD87A0DD6608DD6E063E0032FB
130 DATA A0CD97A0C9E5DD7E04DD660894473C32FCA004E1C9C5D5E5DDE5DD21FDA0
140 DATA 0608C5252DCD1ABCED43F9A011000BC1C53AFCA047E5C5ED4BF9A03AFBA2
150 DATA FE0028177EDD7700DD232310F0C110E8E119C110D0DDE1E1D1C1C9DD7E00
160 DATA 7718E7DD6608DD6E06DD5604DD5E02DD46003AB9A3252D151DCD50BCC9
170 DATA TELOS
```

Αυτή η ρουτίνα δεν μπορεί να μεταφερθεί διότι χρησιμοποιεί πολλές υπορουτίνες καθώς και ένα buffer για να κάνει την μεταφορά της οθόνης.

ΚΑΘΑΡΙΣΜΑ ΤΗΣ ΟΘΟΝΗΣ

Μπορείτε πολύ εύκολα τόσο από BASIC όσο και από κώδικα μηχανής να καθαρίσετε την οθόνη του Amstrad. Από Basic δίνοντας την εντολή CLS, ενώ από κώδικα μηχανής με την υπορουτίνα &BB5A (47962) του λειτουργικού. Δίνοντας δηλαδή:

```
LD A, 12
CALL &BB5A
```

καθαρίζει η οθόνη (ή το παράθυρο που βρισκόσαστε) τοποθετώντας τον δρομέα στο άνω αριστερά άκρο.

Ωστόσο, επειδή αυτή η μέθοδος καθαρίζει την οθόνη πολύ απότομα θα σας δώσουμε εδώ μια ρουτίνα που «εξασθενίζει» την οθόνη σταδιακά μέχρι να τη σβήσει τελείως. Η ρουτίνα λοιπόν αυτή «εξασθενεί» την οθόνη μέχρι το ink 0, και μπορεί να μεταφερθεί σε οποιοδήποτε άλλο μέρος της μνήμης.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: CALL 42000

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF, HL και DE έχουν τυχαίο περιεχόμενο.

```
Hisoft GENA3.1 Assembler. Page     1.

Pass 1 errors: 00

A410              10          org  42000
A410              15          ENT  $
```

```
A410  1EFE        20          LD   E,254          ;first mask
A412  2100C0      30 LOOP1:   LD   HL,49152      ;start of video RAM
A415  7B          40 LOOP2:   LD   A,E           ;mask into A
A416  A6          50          AND  (HL)          ;mask current screen
A417  77          60          LD   (HL),A        ;put byte in video RAM
A418  23          70          INC  HL            ;next byte
A419  7D          80          LD   A,L           ;is address
A41A  B4          90          OR   H             ;now 0000 ?
A41B  20F8       100          JR   NZ,LOOP2      ;do all video RAM
A41D  CB13       110          RL   E             ;rotate mask
A41F  38F1       120          JR   C,LOOP1       ;again if C set
A421  C9         130          RET

Pass 2 errors: 00

Table used:    37  from   139
Executes: 42000
```

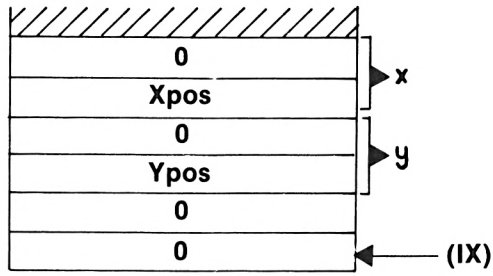
```
10 J=42000
20 RESTORE
30 READ A$
40 IF A$="TELOS" THEN END
50 FOR I=1 TO LEN(A$) STEP 2
60 POKE J,VAL("&"+MID$(A$,I,2))
70 J=J+1
80 NEXT
90 GOTO 30
100 DATA 1EFE2100C07BA677237DB420F8CB1338F1C9
110 DATA TELOS
```

Είναι πολύ ενδιαφέρον να αναλύσετε τον τρόπο που δουλεύει αυτή η ρουτίνα. Σταδιακά κάθε bit της Video RAM γίνεται 0 μέσω ενός byte που αρχικά περιέχει όλο 1. Με αυτό τον τρόπο η οθόνη φαίνεται να «εξασθενίζει» μέχρι να καταλήξουνε όλα τα bits να έχουν 0 οπότε η οθόνη έχει σβήσει.

ΑΝΑΓΝΩΣΗ ΧΑΡΑΚΤΗΡΑ

Η ρουτίνα που παρουσιάζουμε στη συνέχεια επιστρέφει τον κωδικό ASCII οποιουδήποτε χαρακτήρα που μπορεί να αναγνωρίσει ο Amstrad, σε οποιαδήποτε θέση της οθόνης.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: CALL 42000, x, y, @ CHR%, όπου x και y είναι οι συντεταγμένες του χαρακτήρα που θέλουμε να διαβάσουμε και CHR% η μεταβλητή μέσα στην οποία θα επιστραφεί ο ASCII του χαρακτήρα. Πρέπει πριν κληθεί αυτή η ρουτίνα να έχει δοθεί τιμή στη μεταβλητή με π.χ. CHR%=0 ώστε να έχει κρατηθεί θέση γι'αυτήν στη μνήμη. Σε περίπτωση που καλέσουμε τη ρουτίνα από κώδικα μηχανής, πρέπει ο καταχωρητής A να έχει την τιμή 3 και ο IX να δείχνει το παρακάτω block μεταβλητών.



ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ:

Σε Basic επιστρέφοντας, αν ο χαρακτήρας αναγνωρίστηκε από τον υπολογιστή η μεταβλητή CHR% έχει την τιμή του σε ASCII. Αλλιώς περιέχει την τιμή 256. Επιστρέφοντας από κώδικα μηχανής, ο IX θα δείχνει σε μία διεύθυνση όπου θα υπάρχει ο κωδικός ASCII του χαρακτήρα και η IX+1 θα έχει 0, αλλιώς (αν ο χαρακτήρας δεν αναγνωρίστηκε, η διεύθυνση (IX+0) θα περιέχει 0 και η (IX+1) θα περιέχει 1.

```
Hisoft GENA3.1 Assembler. Page      1.

Pass 1 errors: 00

A410                10          org   42000
A410  FE03          20          CP    3                 ;three parameters
A412  C0            30          RET   NZ
A413  DD6604        40          LD    H,(IX+4)
A416  DD6E02        50          LD    L,(IX+2)
A419  CD75BB        60          CALL  47989             ;put the cursor
A41C  CD60BB        70          CALL  47968             ;read character
A41F  DD6E00        80          LD    L,(IX+0)
A422  DD6601        90          LD    H,(IX+1)
A425  3005         100          JR    NC,NCHAR          ;is not known char
A427  77           110          LD    (HL),A            ;put ASCII of char
A428  23           120          INC   HL                ;into variable
A429  3600         130          LD    (HL),0
A42B  C9           140          RET
A42C  3600         150 NCHAR:   LD    (HL),0            ;if not known char
A42E  23           160          INC   HL                ;return 256
A42F  3601         170          LD    (HL),1
A431  C9           180          RET

Pass 2 errors: 00

Table used:     25  from   142


10 J=42000
20 RESTORE
30 READ A$
40 IF A$="TELOS" THEN END
50 FOR I=1 TO LEN(A$) STEP 2
60 POKE J,VAL("&"+MID$(A$,I,2))
70 J=J+1
```

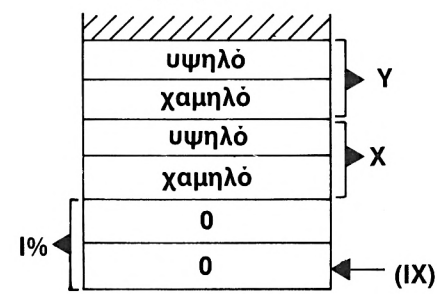
```
80 NEXT
90 GOTO 30
100 DATA FE03C0DD6604DD6E02CD75BBCD60BBDD6E00DD660130057723360 0C93600
110 DATA 233601C9
120 DATA TELOS
```

Η ρουτίνα δεν εκτελείται αν ο αριθμός των παραμέτρων που μπαίνουν στην εντολή CALL δεν είναι ακριβώς 3. Επίσης μην ξεχάσετε να βάλετε το σύμβολο @ πριν από τη μεταβλητή CHR%. Η πιο συνηθισμένη περίπτωση που δεν αναγνωρίζεται ο χαρακτήρας (και επιστρέφει στην CHR% η τιμή 256) είναι να έχει γίνει κάποιο PLOT ή DRAW μέσα στο τετράγωνό του (τα 8×8 pixels που καταλαμβάνει στην οθόνη). Μια δεύτερη επίσης περίπτωση που δεν αναγνωρίζεται ο χαρακτήρας και πρέπει να προσέξετε, είναι να έχει αλλαχτεί από τη στιγμή που τον τυπώσαμε στην οθόνη, το PEN και το PAPER.

ΑΝΑΓΝΩΣΗ PIXEL

Η ρουτίνα που σας παρουσιάζουμε στη συνέχεια δίνει την τιμή του INK που έχει ένα συγκεκριμένο pixel στην οθόνη του Amstrad. Η ρουτίνα μπορεί να μεταφερθεί σε οποιαδήποτε θέση της user RAM

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: CALL 42000, x, y, @ I% όπου x, y οι συντεταγμένες του PIXEL στην οθόνη, και I% η μεταβλητή στην οποία θα επιστραφεί το ink. Μην ξεχάσετε να ορίσετε πρώτα την I% (δίνοντας I%=0) ώστε να έχει κρατηθεί γι' αυτήν θέση στη μνήμη.
Από κώδικα μηχανής πρέπει ο A να περιέχει την τιμή 3 και ο IX να δείχνει στο παρακάτω block παραμέτρων.



ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ολοι οι καταχωρητές έχουν τυχαίο περιεχόμενο.

```
Hisoft GENA3.1 Assembler. Page     1.

Pass 1 errors: 00

A410                10          org  42000
A410  FE03          20          CP   3                    ;three parameters
A412  C0            30          RET  NZ
A413  DD6E02        40          LD   L,(IX+2)
A416  DD6603        50          LD   H,(IX+3)
A419  DD5E04        60          LD   E,(IX+4)
A41C  DD5605        70          LD   D,(IX+5)
A41F  CDF0BB        80          CALL 48112                ;absolute test pixel
A422  DD6E00        90          LD   L,(IX+0)
A425  DD6601       100          LD   H,(IX+1)
A428  77           110          LD   (HL),A               ;store ink in var.
A429  23           120          INC  HL
A42A  3600         130          LD   (HL),0
A42C  C9           140          RET

Pass 2 errors: 00

Table used:     13  from   127
```

```
10 J=42000
20 RESTORE
30 READ A$
40 IF A$="TELOS" THEN END
50 FOR I=1 TO LEN(A$) STEP 2
60 POKE J,VAL("&"+MID$(A$,I,2))
70 J=J+1
80 NEXT
90 GOTO 30
100 DATA FE03C0DD6E02DD6603DD5E04DD5605CDF0BBDD6E00DD660177233600C9
110 DATA TELOS
```

Μπορείτε να έχετε περισσότερες πληροφορίες για τις ρουτίνες του λειτουργικού που χρησιμοποιήθηκαν στο τελευταίο κεφάλαιο του βιβλίου (Κεφ. 8).

ΑΝΤΙΣΤΡΟΦΗ ΟΘΟΝΗΣ

Με την παρακάτω απλή ρουτίνα, αλλάζουμε όλα τα bytes της Video RAM με το συμπλήρωμά τους αντιστρέφοντας έτσι το χρώμα σε ότι φαίνεται στην οθόνη. Μπορείτε να την φορτώσετε σε οποιαδήποτε θέση της USER RAM.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: CALL 42000

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι καταχωρητές AF και HL έχουν τυχαίο περιεχόμενο.

```
Pass 1 errors: 00

A410                10          org  42000
A410  2100C0        20          LD   HL,49152             ;start screen address
A413  7E            30 LOOP:    LD   A,(HL)              ;get screen byte
A414  2F            40          CPL                       ;complement it
A415  77            50          LD   (HL),A               ;put it back
A416  23            60          INC  HL                   ;next byte
A417  7D            70          LD   A,L                  ;did HL reach
A418  B4            80          OR   H                    ;reach 0000 ?
A419  20F8          90          JR   NZ,LOOP              ;if not repeat
A41B  C9           100          RET

Pass 2 errors: 00
Table used:     24  from   127
```

```
10 J=42000
20 RESTORE
30 READ A$
40 IF A$="TELOS" THEN END
50 FOR I=1 TO LEN(A$) STEP 2
60 POKE J,VAL("&"+MID$(A$,I,2))
70 J=J+1
80 NEXT
90 GOTO 30
100 DATA 2100C07E2F77237DB420F8C9
110 DATA TELOS
```

Είπαμε ότι κάθε byte της Video RAM το αλλάζουμε με το συμπλήρωμά του. Αυτό σημαίνει ότι κάθε bit που ήταν 0 γίνεται 1 και κάθε bit που ήταν 1 γίνεται 0. Όπως καταλαβαίνετε με αυτή την ενέργεια γίνονται κάποιες αλλαγές χρωμάτων στην οθόνη πράγμα που είναι πολύ ενδιαφέρον, ιδίως αν δουλεύετε με πολλά χρώματα (όπως π.χ. στο MODE 0).

ΜΕΤΑΦΟΡΑ ΟΘΟΝΗΣ

Η επόμενη ρουτίνα που παρουσιάζουμε είναι πολύ απλή, αλλά καταλαμβάνει πολλή μνήμη για να δουλέψει. Συγκεκριμένα αντιγράφει και τα 16K της Video RAM στην user RAM και συνεπώς την οθόνη ώστε να τη χρησιμοποιήσουμε αργότερα αν θέλουμε να κάνουμε γρήγορη εναλλαγή. Όταν θέλουμε να ξαναπάρουμε τα περιεχόμενα της προηγούμενης οθόνης που αποθηκεύσαμε, μεταφέρουμε τα bytes από το μέρος που τα έχουμε φυλάξει στη Video RAM. Η οθόνη αντιγράφεται στις θέσεις μνήμης 26000 - 42384 ενώ η ρουτίνα που κάνει τη δουλειά βρίσκεται στη θέση 25500.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: CALL 25500, D όπου D είναι η κατεύθυνση μεταφοράς. Για D=0 μεταφέρεται η οθόνη στη USER RAM και για D=1 ξαναγράφεται η οθόνη στη Video RAM.
Για χρήση της ρουτίνας από κώδικα μηχανής πρέπει ο A να έχει την τιμή 1 και ο IX να δείχνει τη διεύθυνση μνήμης όπου περιέχεται η τιμή του D.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Όλοι οι καταχωρητές έχουν τυχαίο περιεχόμενο.

```
Hisoft GENA3.1 Assembler. Page      1.

Pass 1 errors: 00

61AB                10          ORG  25000
61AB  FE01          20          CP   1                    ;one parameter
61AA  C0            30          RET  NZ
61AB  DD7E00        40          LD   A,(IX+0)
```

```
61AE  FE00        50          CP   0                ;0 means screen to RAM
61B0  280C        60          JR   Z,DOWN           ;so do it
61B2  1100C0      70          LD   DE,49152         ;move RAM to screen
61B5  219065      80          LD   HL,26000
61B8  010040      90          LD   BC,16384
61BB  EDB0       100          LDIR                  ;do the move
61BD  C9         110          RET
61BE  119065     120 DOWN:    LD   DE,26000         ;screen to RAM
61C1  2100C0     130          LD   HL,49152
61C4  010040     140          LD   BC,16384
61C7  EDB0       150          LDIR                  ;do the move
61C9  C9         160          RET

Pass 2 errors: 00

Table used:     24  from   135
```

```
10 J=25000
20 RESTORE
30 READ A$
40 IF A$="TELOS" THEN END
50 FOR I=1 TO LEN(A$) STEP 2
60 POKE J,VAL("&"+MID$(A$,I,2))
70 J=J+1
80 NEXT
90 GOTO 30
100 DATA FE01C0DD7E00FE00280C1100C02190650100400EDB0C911906521000C00100
110 DATA 40EDB0C9
120 DATA TELOS
```

Κάτι που πρέπει να προσέχετε όταν χρησιμοποιείτε αυτή τη ρουτίνα είναι να μην έχει συμβεί SCROLLING όταν σώζετε ή αποκαθιστάτε μια οθόνη. Και αυτό γιατί αλλάζει η οργάνωση της VIDEO RAM. Επίσης όταν επαναφέρετε μια οθόνη, προσέχετε να βρισκόσαστε στο σωστό MODE για να μην εμφανιστούν τίποτα περίεργα σχέδια.

ΑΝΤΑΛΛΑΓΗ ΟΘΟΝΩΝ

Το μειονέκτημα που παρουσιάζει η προηγούμενη ρουτίνα είναι ότι όταν μεταφέρουμε μια οθόνη που έχουμε αποθηκεύσει στη θέση 26000 προς τη VIDEO RAM, η εικόνα που έχουμε στην οθόνη σβήνεται.

Η ρουτίνα που παρουσιάζουμε εδώ κάνει ανταλλαγή οθονών, δηλαδή ανταλλάσσει την οθόνη που βλέπουμε με αυτή που υπάρχει στη θέση 26000 της μνήμης, ώστε να μην χάνεται η τρέχουσα οθόνη.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: CALL 25500

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Όλοι οι καταχωρητές έχουν τυχαίο περιεχόμενο.

```
Pass 1 errors: 00

61A8              10          org  25000
61A8  119065      20          LD   DE,26000         ;initialize ragisters
61AB  2100C0      30          LD   HL,49152
61AE  4E          40 LOOP:    LD   C,(HL)           ;get byte from screen
61AF  1A          50          LD   A,(DE)           ;get byte from store
61B0  77          60          LD   (HL),A           ;swap bytes
61B1  79          70          LD   A,C
61B2  12          80          LD   (DE),A
61B3  23          90          INC  HL               ;next byte
61B4  13         100          INC  DE
```

```
61B5  7D        110       LD   A,L
61B6  B4        120       OR   H                    ;if DE<>0000
61B7  20F5      130       JR   NZ,LOOP              ;do it again
61B9  C9        140       RET

Pass 2 errors: 00

Table used:     24  from   133
```

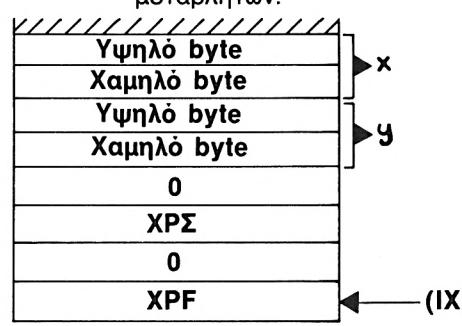
```
10 J=25000
20 RESTORE
30 READ A$
40 IF A$="TELOS" THEN END
50 FOR I=1 TO LEN(A$) STEP 2
60 POKE J,VAL("&"+MID$(A$,I,2))
70 J=J+1
80 NEXT
90 GOTO 30
100 DATA 1190652100C04E1A77791223137DB420F5C9
110 DATA TELOS
```

ΡΟΥΤΙΝΑ ΓΙΑ FILL

Στη συνέχεια σας παρουσιάζουμε μια ρουτίνα που γεμίζει μία συγκεκριμένη γραμμή (ενός pixel) της οθόνης, μεταξύ δύο «συνοριακών» pixels με ένα χρώμα που έχουμε καθορίσει εμείς. Το χρώμα που έχουν τα δύο συνοριακά pixels καθώς και το χρώμα που γίνεται το fill πρέπει να καθοριστεί από μας.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: CALL 42000, x, y, ΧΡΣ, ΧΡF όπου x, y οι συντεταγμένες όπου ξεκινάει το FILL, ΧΡΣ το χρώμα που έχουν τα οριακά δεξιά και αριστερά pixels (που μαρκάρουν το τέλος του FILL) και ΧΡF το χρώμα με το οποίο θα γίνει το FILL.

Από κώδικα μηχανής ο A πρέπει να περιέχει την τιμή 4 και ο IX να δείχνει το παρακάτω block μεταβλητών.



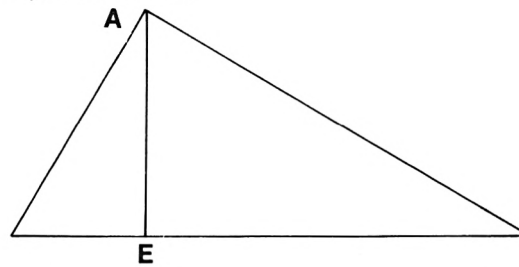
ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Όλοι οι καταχωρητές έχουν τυχαίο περιεχόμενο.

```
Hisoft GENA3.1 Assembler. Page      1.

Pass 1 errors: 00

A410                10          org  42000
A410  FE04          20          CP   4                    ;four parameters
A412  C0            30          RET  NZ
A413  DD6E04        40          LD   L,(IX+4)
A416  DD6605        50          LD   H,(IX+5)
A419  DD5E06        60          LD   E,(IX+6)
A41C  DD5607        70          LD   D,(IX+7)
A41F  E5            80          PUSH HL
A420  D5            90          PUSH DE
A421  E5           100 LOOP1:   PUSH HL                 ;start scan right
A422  D5           110          PUSH DE
A423  CDF0BB       120          CALL 48112              ;test abs of pixel
A426  D1           130          POP  DE
A427  E1           140          POP  HL
A428  DDBE02       150          CP   (IX+2)             ;is it border colour?
A42B  2808         160          JR   Z,OUTR             ;if yes, out
A42D  3E03         170          LD   A,3
A42F  BA           180          CP   D                  ;is out of screen?
A430  2803         190          JR   Z,OUTR             ;if yes, out
A432  13           200          INC  DE                 ;next pixel
A433  18EC         210          JR   LOOP1
A435  1B           220 OUTR:    DEC  DE                 ;back one pixel
A436  ED5369A4     230          LD   (RIGHT),DE         ;store right edge
A43A  D1           240          POP  DE                 ;restore reg.
A43B  E1           250          POP  HL
A43C  E5           260 LOOP2:   PUSH HL                 ;scan to left
A43D  D5           270          PUSH DE
A43E  CDF0BB       280          CALL 48112              ;abs. test pixel
A441  D1           290          POP  DE
A442  E1           300          POP  HL
A443  DDBE02       310          CP   (IX+2)             ;is it border colour?
A446  2807         320          JR,  Z,OUTL             ;if yes, out
A448  1B           330          DEC  DE                 ;next pixel
A449  7B           340          LD   A,E
A44A  B2           350          OR   D                  ;is out of screen?
A44B  2802         360          JR   Z,OUTL             ;if yes,out
A44D  18ED         370          JR   LOOP2              ;repeat
A44F  13           380 OUTL:    INC  DE                 ;next pixel
A450  DD6E04       390          LD   L,(IX+4)           ;y coord.in HL
A453  DD6605       400          LD   H,(IX+5)
A456  E5           410          PUSH HL                 ;saveit
A457  CDC0BB       420          CALL 48064              ;move absolute
A45A  DD7E00       430          LD   A,(IX+0)           ;get ink to fill
A45D  CDDEBB       440          CALL 48094              ;set graphics pe
A460  E1           450          POP  HL
A461  ED5B69A4     460          LD   DE,(RIGHT)         ;get right edge
A465  CDF6BB       470          CALL 48118              ;draw line absolute
A468  C9           480          RET                     ;finished
A469               490 RIGHT:   DEFS 2
Pass 2 errors: 00

Table used:     71  from    220
```

```
10 J=42000
20 RESTORE
30 READ A$
40 IF A$="TELOS" THEN END
50 FOR I=1 TO LEN(A$) STEP 2
60 POKE J,VAL("&"+MID$(A$,I,2))
70 J=J+1
80 NEXT
90 GOTO 30
100 DATA FE04C0DD6E04DD6605DD5E06DD5607E5D5E5D5CDF0BBD1E1DDBE0228083E
110 DATA 03BA2803131 8EC1BED5369A4D1E1E5D5CDF0BBD1E1DDBE0228071B7BB228
120 DATA 0218ED13DD6E04DD6605E5CDC0BBDD7E00CDDEBBE1ED5B69A4CDF6BBC9
130 DATA TELOS
```

Ο τρόπος που δουλεύει αυτή η ρουτίνα είναι ο εξής. Ξεκινάμε από τη θέση x, y και ψάχνουμε το δεξιό όριο, δηλαδή ή το pixel που θα έχει το χρώμα που δώσαμε για το όριο αυτό, ή το τέλος της οθόνης (συντεταγμένη μεγαλύτερη από 750). Αποθηκεύει στη συνέχεια αυτήν τη θέση σε δύο bytes μνήμης, και ξεκινάει από τις αρχικές συντεταγμένες x, y κάνοντας την ίδια εργασία προς τα αριστερά (μέχρι να βρει οριακό pixel ή να φτάσει στο αριστερό άκρο της οθόνης - συντεταγμένη 0). Στη συνέχεια γεμίζει το τμήμα της γραμμής από PIXEL, που έχει βρει, με το χρώμα XPF που του έχουμε δώσει.

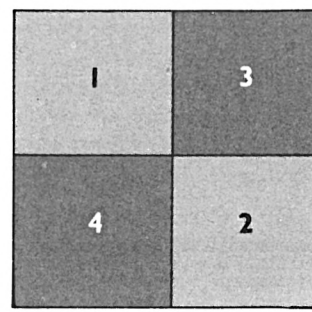
Με αυτό τον τρόπο κάνουμε διαδοχικά call στη ρουτίνα και πετυχαίνουμε να γεμίσουμε ολόκληρα γεωμετρικά σχήματα. Για να το πετύχουμε αυτό διαλέγουμε μια κατακόρυφη που έχει το μεγαλύτερο μήκος και χρησιμοποιούμε το Fill. Για παράδειγμα στο παρακάτω τρίγωνο χρησιμοποιούμε την ευθεία AE.



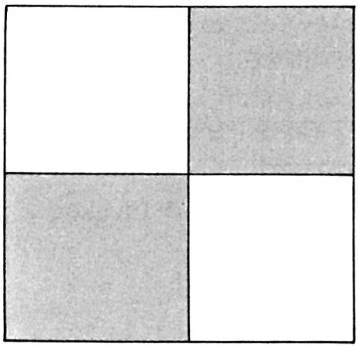
Οπότε το συνολικό FILL του σχήματος ολοκληρώνεται μέσα σε ένα FOR/NEXT loop.

ΠΟΛΥΧΡΩΜΟΥ ΧΑΡΑΚΤΗΡΕΣ

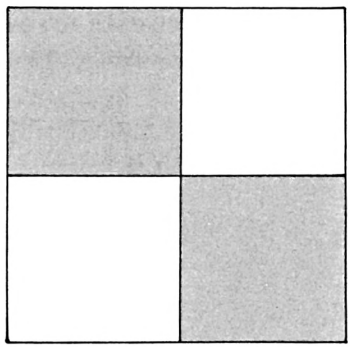
Εδώ σας παρουσιάζουμε μια ρουτίνα που σας δίνει τη δυνατότητα να έχετε πολύχρωμους χαρακτήρες στην οθόνη του Amstrad. Αυτό επιτυγχάνεται με τη χρήση του XOR (exclusive OR). Ας υποθέσουμε ότι έχουμε τον παρακάτω χαρακτήρα ο οποίος αποτελείται από 2 χρώ-



ματα. Τα τετράγωνα 1 και 2 έχουν ένα συγκεκριμένο χρώμα και τα τετράγωνα 2 και 4 κάποιο άλλο. Αν χωρίζαμε λοιπόν αυτόν το χαρακτήρα σε δύο διαφορετικούς ανάλογα με το χρώμα των τετραγώνων θα είχαμε τους



και



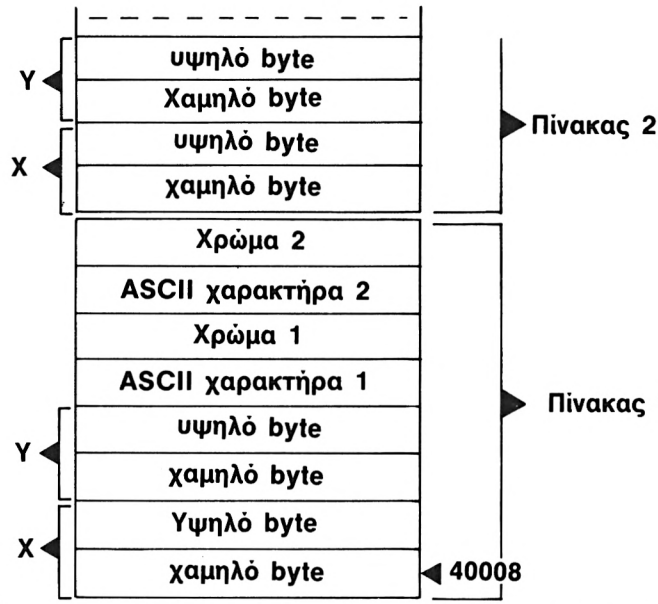
οι οποίοι υπάρχουν στο σετ χαρακτήρων του Amstrad και είναι αντίστοιχα οι CHR$ (134) και CHR$ (137). Αυτό λοιπόν που χρειάζεται να κάνουμε είναι να χρησιμοποιήσουμε την εντολή TAG του Amstrad, ώστε να τυπωθούν οι χαρακτήρες ο ένας πάνω στον άλλο (με XOR). Το αποτέλεσμα θα είναι ο πολύχρωμος χαρακτήρας που είχαμε στην αρχή. Για παράδειγμα θα μπορούσαμε να το καταφέρουμε αυτό με το παρακάτω προγραμμάτακι σε BASIC.

```
10 MODE 1
20 GOSUB 1000
30 FOR i=0 TO 300:NEXT
50 END
1000 TAG
1010 PLOT 1000,1000,1:REM colour of first character
1020 MOVE 100,100
1030 PRINT CHR$(134);:REM print the character
1040 TAGOFF:PRINT CHR$(23);CHR$(1):REM XOR mode
1050 TAG
1060 PLOT 1000,1000,2:REM colour for second character
1070 MOVE 100,100
1080 PRINT CHR$(137)
1090 RETURN
```

Με τον ίδιο τρόπο θα μπορούσατε να σχεδιάσετε τους εξωγήϊνους σας σε δύο ή περισσότερους χαρακτήρες ανάλογα με το πόσα χρώματα θέλετε να τους δώσετε.

Στη συνέχεια σας δίνουμε μια ρουτίνα που μετακινεί δύο χαρακτήρες που τυπώσατε τον ένα στον άλλο, με τον τρόπο που περιγράψαμε προηγουμένως. Θα σας φανεί πολύ χρήσιμη στο να μετακινείτε στους πολύχρωμους χαρακτήρες σας στην οθόνη του υπολογιστή. Αρκεί λοιπόν να τυπώσετε μια φορά τους χαρακτήρες σας, για να τους μετακινείτε μετά όπου θέλετε πάνω στην οθόνη του Amstrad.

Κάτι άλλο απαραίτητο εδώ, είναι ένας πίνακας που πρέπει να κάνουμε POKE στη μνήμη του υπολογιστή απ' τη θέση 40008 και πάνω, ο οποίος θα περιέχει τις παλιές συντεταγμένες X, Y, τον ASCII του χαρακτήρα 1, το χρώμα του, τον ASCII του χαρακτήρα 2 και το αντίστοιχο χρώμα του (8 bytes). Μπορούμε να έχουμε - θεωρητικά - μέχρι 255 τέτοιους πίνακες τον ένα μετά τον άλλον ώστε να κινούμε ταυτόχρονα πολλά ζευγάρια πολύχρωμων χαρακτήρων.

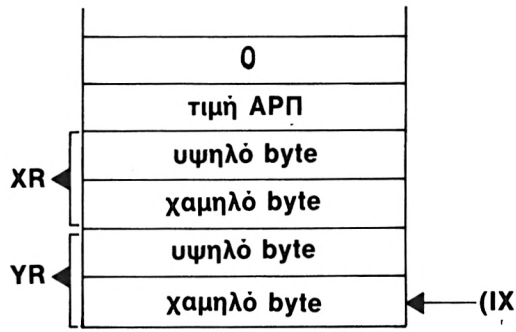


Αφού τυπώσουμε λοιπόν για πρώτη φορά τους 2 χαρακτήρες με τη ρουτίνα σε BASIC που αναφέραμε ήδη, και κάνουμε POKE τον πίνακα (ή για περισσότερα ζευγάρια τους πίνακες) με τα στοιχεία που περιγράψαμε προηγουμένως, μπορούμε να χρησιμοποιήσουμε την παρακάτω ρουτίνα για να τους κινήσουμε στην οθόνη του Amstrad:

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: CALL 42000, ΑΡΠ, XR, YR, όπου ΑΡΠ είναι ο αριθμός πίνακα που αντιστοιχεί το ζευγάρι των χαρακτήρων που θέλουμε να μετακινήσουμε, και XR, YR οι σχετικές νέες συντεταγμένες που θέλουμε να τους πάμε.

Από κώδικα μηχανής πρέπει ο A να περιέχει 3, ο IX να δείχνει το παρακάτω μπλοκ μεταβλητών, ενώ

πρέπει παράλληλα να υπάρχει και ο πίνακας που περιγράψαμε παραπάνω στη θέση 40008.



ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ολοι οι καταχωρητές έχουν τυχαίο περιεχόμενο. Το pen Γραφικών θα έχει το χρώμα του τελευταίου χαρακτήρα που τυπώσαμε. Ο δρομέας γραφικών θα βρίσκεται στο σημείο όπου έχει μετακινηθεί το ζεύγος χαρακτήρων.

```
Hisoft GENA3.1 Assembler. Page      1.

Pass 1 errors: 00

A410                10          ORG  42000
A410  FE03          20          CP   3                    ;three parameters
A412  C0            30          RET  NZ
A413  DD4604        40          LD   B,(IX+4)
A416  DDE5          50          PUSH IX
A418  DD21409C      60          LD   IX,40000             ;address of shapetable
A41C  DD23          70 LOOP:    INC  IX                   ;get entry
A41E  DD23          80          INC  IX
A420  DD23          90          INC  IX
A422  DD23         100          INC  IX
A424  DD23         110          INC  IX
A426  DD23         120          INC  IX
A428  DD23         130          INC  IX
A42A  DD23         140          INC  IX
A42C  10EE         150          DJNZ LOOP
A42E  DD22BFA4     160          LD   (STORE),IX           ;store start entry
A432  DD5E00       170          LD   E,(IX+0)             ;get x coordinate
A435  DD5601       180          LD   D,(IX+1)
A438  DD6E02       190          LD   L,(IX+2)             ;get y coordinate
A43B  DD6603       200          LD   H,(IX+3)
A43E  E5           210          PUSH HL
A43F  D5           220          PUSH DE
A440  CDC0BB       230          CALL 48064                ;move absolute
A443  DD7E05       240          LD   A,(IX+5)             ;set first colour
A446  CDDEBB       250          CALL 48094                ;set graphics pen
A449  3E01         260          LD   A,1                  ;set the XOR mode
A44B  CD59BC       270          CALL 48217
A44E  DD7E04       280          LD   A,(IX+4)
A451  CDFCBB       290          CALL 48124                ;print first char
A454  DD7E07       300          LD   A,(IX+7)
A457  CDDEBB       310          CALL 48094                ;set second colour
A45A  D1           320          POP  DE
```

```
A45B  E1          330          POP  HL
A45C  E5          340          PUSH HL
A45D  D5          350          PUSH DE
A45E  CDC0BB      360          CALL 48064          ;absolute move
A461  DD7E06      370          LD   A,(IX+6)
A464  CDFCBB      380          CALL 48124          ;print second char
A467  D1          390          POP  DE
A468  E1          400          POP  HL
A469  DDE1        410          POP  IX             ;recover IX
A46B  E5          420          PUSH HL
A46C  D5          430          PUSH DE
A46D  E1          440          POP  HL
A46E  DD4E02      450          LD   C,(IX+2)       ;add increment
A471  DD4603      460          LD   B,(IX+3)       ;to x coordinate
A474  09          470          ADD  HL,BC
A475  E5          480          PUSH HL
A476  D1          490          POP  DE             ;get it in DE
A477  DDE5        500          PUSH IX
A479  DD2ABFA4    510          LD   IX,(STORE)
A47D  DD7300      520          LD   (IX+0),E       ;store in table
A480  DD7201      530          LD   (IX+1),D
A483  DDE1        540          POP  IX
A485  DD4E00      550          LD   C,(IX+0)       ;add increment to
A488  DD4601      560          LD   B,(IX+1)       ;the y coordinate
A48B  E1          570          POP  HL
A48C  09          580          ADD  HL,BC
A48D  DD2ABFA4    590          LD   IX,(STORE)
A491  DD7502      600          LD   (IX+2),L       ;store in table
A494  DD7403      610          LD   (IX+3),H
A497  E5          620          PUSH HL
A498  D5          630          PUSH DE
A499  CDC0BB      640          CALL 48064          ;move to new locat.
A49C  DD7E05      650          LD   A,(IX+5)       ;get first colour
A49F  CDDEBB      660          CALL 48094          ;set graphics pen
A4A2  DD7E04      670          LD   A,(IX+4)       ;print first char
A4A5  CDFCBB      680          CALL 48124
A4A8  D1          690          POP  DE
A4A9  E1          700          POP  HL
A4AA  CDC0BB      710          CALL 48064          ;move back
A4AD  DD7E07      720          LD   A,(IX+7)       ;get second colour
A4B0  CDDEBB      730          CALL 48094          ;set graphics pen
A4B3  DD7E06      740          LD   A,(IX+6)       ;get second char
A4B6  CDFCBB      750          CALL 48124          ;and print it
A4B9  3E00        760          LD   A,0            ;"force" mode
A4BB  CD59BC      770          CALL 48217
A4BE  C9          780          RET
A4BF              790 STORE:   DEFS 2
9C40              800          ORG  40000
9C40              810 TABLE:   DEFS 8
9C48  64          820          DEFB 100
9C49  00          830          DEFB 0
9C4A  64          840          DEFB 100
9C4B  00          850          DEFB 0
9C4C  86          860          DEFB 134
9C4D  01          870          DEFB 1
9C4E  89          880          DEFB 137
9C4F  02          890          DEFB 2
Pass 2 errors: 00
Table used:    48  from   293
```

```
10 J=42000
20 RESTORE
30 READ A$
40 IF A$="TELOS" THEN END
50 FOR I=1 TO LEN(A$) STEP 2
60 POKE J,VAL("&"+MID$(A$,I,2))
70 J=J+1
80 NEXT
90 GOTO 30
100 DATA FE03C0DD4604DDE5DD21409CDD23DD23DD23DD23DD23DD23DD23DD2310EE
110 DATA DD22BFA4DD5E00DD5601DD6E02DD6603E5D5CDC0BBDD7E05CDDEBB3E01CD
120 DATA 59BCDD7E04CDFCBBDD7E07CDDEBBD1E1E5D5CDC0BBDD7E06CDFCBBD1E1DD
130 DATA E1E5D5E1DD4E02DD460309E5D1DDE5DD2ABFA4DD7300DD7201DDE1DD4E00
140 DATA DD4601E109DD2ABFA4DD7502DD7403E5D5CDC0BBDD7E05CDDEBBDD7E04CD
150 DATA FCBBD1E1CDC0BBDD7E07CDDEBBDD7E06CDFCBB3E00CD59BCC9
160 DATA TELOS
```

Η ρουτίνα αυτή μπορεί να φορτωθεί και σε άλλη θέση της μνήμης αρκεί να φροντίσετε για τη θέση του πίνακα και της μεταβλητής STORE.

Παρακάτω σας δίνουμε ένα προγραμματάκι σε BASIC που επιδεικνύει τη χρήση της παραπάνω ρουτίνας.

```
10 MODE 1
20 XR=1
30 YR=1
40 PLOT 1000,1000,1:REM FIRST COLOUR
50 PRINT CHR$(23)+CHR$(1);:REM XOR MODE
60 MOVE 100,100
65 CALL &BD19
70 TAG:PRINT CHR$(134)
80 PLOT 1000,1000,2:REM SECOND COLOUR
90 MOVE 100,100
95 CALL &BD19
100 PRINT CHR$(137);
110 TAGOFF
120 FOR I=0 TO 300
130 CALL 42000,1,XR,YR
140 FOR J=1 TO 20
150 NEXT
160 NEXT
170 END
```

Εδώ τελειώνει το κεφάλαιο για το χειρισμό της οθόνης των μοντέλων της Amstrad. Στη συνέχεια θα ασχοληθούμε λίγο με τον ήχο...

4ο ΚΕΦΑΛΑΙΟ

ΧΕΙΡΙΣΜΟΣ ΤΟΥ ΗΧΟΥ

Όπως σίγουρα θα ξέρετε ο Amstrad έχει μερικές πολύ εντυπωσιακές δυνατότητες όσον αφορά την παραγωγή ήχου. Θα έχετε παρατηρήσει άλλωστε τα πολύ καλά ηχητικά εφφέ που έχουν διάφορα παιχνίδια που κυκλοφορούν στην αγορά.

Στο λειτουργικό σύστημα του Amstrad υπάρχουν πολλές ρουτίνες τις οποίες μπορούμε να καλέσουμε για να δημιουργήσουμε ήχο με κάποιο Tone και Envelope ή και να συνδυάσουμε περισσότερους από έναν ήχους. Μπορείτε λοιπόν καλώντας αυτές τις ρουτίνες και δίνοντας αντίστοιχες τιμές στους καταχωρητές, να δημιουργήσετε τους ήχους που θέλετε - όπως και στη BASIC. Γι' αυτό σ' αυτό το κεφάλαιο δε θα ασχοληθούμε με αυτόν τον τρόπο δημιουργίας ήχου, αλλά με το τσιπάκι του ήχου το ίδιο. Και αυτό γιατί χρησιμοποιώντας τις ρουτίνες του firmware χρειαζόμαστε όλους αυτούς τους πολύπλοκους πίνακες δεδομένων που σίγουρα θα σας έχουν μπερδέψει όταν φτιάχνατε ήχο από την BASIC. Χρησιμοποιώντας όμως το τσιπάκι του ήχου κάνουμε τη ζωή μας πιο εύκολη, χωρίς βέβαια να έχουμε όλες τις δυνατότητες που μας δίνει το Firmware.

Για παράδειγμα με την απλή ρουτίνα

```
LD A, 7
CALL #BB5A
RET
```

μπορούμε να έχουμε το απλούστερο ηχητικό αποτέλεσμα, δηλαδή ένα beep.

ΤΟ ΟΛΟΚΛΗΡΩΜΕΝΟ ΠΑΡΑΓΩΓΗΣ ΗΧΟΥ

Το ολοκληρωμένο παραγωγής ήχου που διαθέτει ο Amstrad είναι μια πολύ ευέλικτη προγραμματιζόμενη συσκευή. Υπάρχουν 3 διαφορετικά κανάλια ήχου, που σημαίνει ότι μπορούμε να ακούμε 3 διαφορετικούς ήχους ταυτόχρονα. Επίσης μας δίνει κι από μόνο του τις δυνατότητες να προσδιορίσουμε διάφορα χαρακτηριστικά του ήχου (VOLUME και ENVELOPE). Ακόμα το τσιπάκι αυτό έχει τη δυνατότητα να παράγει «λευκό θόρυβο» από ένα ειδικό μανάλι που διαθέτει. Όλα τα παραπάνω γίνονται μέσα από τους 15 καταχωρητές που διαθέτει αυτό το προγραμματιζόμενο ολοκληρωμένο ήχου, που μπορούν να γραφτούν και να διαβαστούν όπως και οι καταχωρητές του Z80. Αν και το λειτουργικό του Amstrad χειρίζεται το ολοκληρωμένο του ήχου μέσω ενός PPI τσιπ (Parallel Peripheral Interface), δεν είναι συνετό

να γράψουμε στους καταχωρητες του PPI. Έχουμε άλλωστε μια ρουτίνα στη θέση &BD34 (48436) που μας προσφέρει τη δυνατότητα να φορτώνουμε τιμές απ' ευθείας στους καταχωρητές του ολοκληρωμένου για τον ήχο, χωρίς κίνδυνο να κάνουμε κανένα μοιραίο σφάλμα, που θα είχε άσχημες συνέπειες στον υπολογιστή μας.

ΦΟΡΤΩΜΑ ΚΑΤΑΧΩΡΗΤΗ

Τη ρουτίνα του λειτουργικού που αναφέραμε, μπορείτε να την βρείτε στο κεφάλαιο 8 όπου γίνεται πλήρης περιγραφή της. Εδώ υπενθυμίζουμε ότι, όταν την καλούμε, ο Α περιέχει τον αριθμό του καταχωρητή του τσιπ ήχου ενώ ο C την τιμή που θέλουμε να του δώσουμε (0-255). Η ρουτίνα που σας δίνουμε παρακάτω δίνει την δυνατότητα να φορτώνουμε τους καταχωρητές του τσιπ ήχου από BASIC.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: CALL 42000, κατ, τιμή όπου «κατ» είναι ο αριθμός του καταχωρητή (όπως θα εξηγήσουμε παρακάτω) και «τιμή» η τιμή που θέλουμε να του δώσουμε.

```
Hisoft GENA3.1 Assembler. Page      1.

Pass 1 errors: 00

A410                10          org   42000
A410  DD7E02        20          LD    A,(IX+2)
A413  DD4E00        30          LD    C,(IX+0)
A416  CD34BD        40          CALL  48436
A419  C9            50          RET

Pass 2 errors: 00

Table used:     13  from    107


10 J=42000
20 RESTORE
30 READ A$
40 IF A$="TELOS" THEN END
50 FOR I=1 TO LEN(A$) STEP 2
60 POKE J,VAL("&"+MID$(A$,I,2))
70 J=J+1
80 NEXT
90 GOTO 30
100 DATA DD7E02DD4E00CD34BDC9
110 DATA TELOS
```

Σημειώνουμε εδώ ότι τα περισσότερα ηχητικά εφφέ που ακούτε στα περισσότερα παιχνίδια χρησιμοποιούν αυτόν ακριβώς τον τρόπο. Πάντως πριν την χρησιμοποιήσετε πρέπει να ξέρετε μερικά πράγματα για τους καταχωρητές του προγραμματιζόμενου ολοκληρωμένου για τον ήχο.

ΟΙ ΚΑΤΑΧΩΡΗΤΕΣ

Το τσιπάκι που χρησιμοποιείται για τον ήχο είναι το AY-3-8912, για

το οποίο περισσότερες πληροφορίες απ'όσες σας παρέχουμε εδώ μπορείτε να βρείτε στα σχετικά DATA SHEETS γι'αυτό το τσιπ, τα οποία μπορείτε να βρείτε σε καταστήματα που πουλάνε ολοκληρωμένα κυκλώματα.

Οι καταχωρητές του AY-3-8912 αριθμούνται από 0-15 (σύνολο 16). Οι καταχωρητές 14 και 15 δε χρησιμοποιούνται όπως οι υπόλοιποι αλλά έχουν να κάνουν με τα I/0 (είσοδος/έξοδος) του τσιπ. Γι'αυτό και απαιτούν προσεκτική αντιμετώπιση: αν δεν ξέρετε πως να τα χρησιμοποιείτε ΜΗΝ ΤΑ ΠΕΙΡΑΖΕΤΕ!

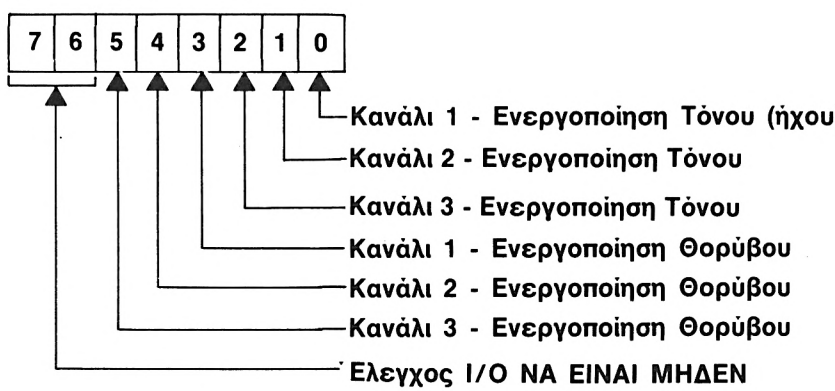
Οι **καταχωρητές 0-5** ελέγχουν την οξύτητα μιας νότας για τα κανάλια 1-3. Δουλεύουν ανά ζευγάρια όπου οι 0 και 1 ελέγχουν το κανάλι 1, οι 2 και 3 το κανάλι 2, και οι 4 και 5 το κανάλι 3. Κάθε ζευγάρι είναι των 12-bit. Τα 8 χαμηλά bit αποθηκεύονται στους 0, 2 και 4 ενώ τα υπόλοιπα 4 στους 1, 3 και 5. Τα 8 χαμηλά bit ενός ζεύγους κάνουν τις «λεπτές» ρυθμίσεις του ήχου ενώ τα 4 υψηλότερα τις πιο «χοντρές». Ο χαρακτηρισμός «λεπτές» και «χοντρές» είναι λογικός γιατί μια μικρή μετατροπή στα 4 υψηλά bit φέρνει μεγάλη μετατροπή στον ήχο, ενώ στα 8 χαμηλότερα bit η διαφορά μόλις που ακούγεται.

Όσο μεγαλύτερη τιμή δίνουμε σ'αυτούς τους καταχωρητές, τόσο χαμηλότερη είναι η οξύτητα του ήχου στο συγκεκριμένο κανάλι.

Ο **καταχωρητής 6** είναι 5-bit και περιέχει τη περίοδο του λευκού θορύβου. Εφ'όσον είναι 6 bit παίρνει τιμές από 0 μέχρι 31. Υπάρχει μόνο ένας καταχωρητής που ελέγχει την περίοδο του λευκού θορύβου πράγμα που σημαίνει ότι δεν μπορούμε να έχουμε ξεχωριστό έλεγχο λευκού θορύβου για κάθε κανάλι.

Η τιμή 0 δίνει την υψηλότερη οξύτητα ενώ η τιμή 31 τη χαμηλότερη.

Ο **καταχωρητής 7** είναι 8 bit και ελέγχει τον ήχο και το θόρυβο στα 3 κανάλια ενώ έχει δύο bit για έλεγχο εισόδου-εξόδου του τσιπ. Σχηματικά έχουμε εξής:



Μπορείτε να φορτώσετε τον καταχωρητή σύμφωνα με το παραπάνω διάγραμμα. Αυτό όμως που προτείνω είναι να έχετε τα bit 6 και 7 πάντοτε 0 γιατί μπορεί να υπάρξουν απρόβλεπτα αποτελέσματα.

Τα bit 0-2 ελέγχουν τη δημιουργία κάποιου ήχου για τα 3 κανάλια αντίστοιχα. Όταν βρίσκονται σε κατάσταση 0 επιτρέπουν τον ήχο ενώ σε κατάσταση 1 τον «κλείνουν». Αν για παράδειγμα θέλουμε να ακουστεί κάποιος ήχος απ'το κανάλι 1, αφού καθορίσουμε την οξύτητα (καταχωρητές 0 και 1) και το πλάτος (καταχωρητής 8 - που θα δούμε παρακάτω) δίνουμε στον καταχωρητή 7 την τιμή &X00111110.

Επίσης στην περίπτωση που θέλουμε να ακούσουμε τη σύνθεση των ήχων και των τριών καναλιών, δεν έχουμε παρά να δώσουμε στον καταχωρητή 7 την τιμή &X00111000.

Τα bit 3-5 ελέγχουν αν θα παραχθεί λευκός θόρυβος σε κάποιο (ή και σε όλα) από τα 3 κανάλια. Όταν βρίσκονται λοιπόν σε κατάσταση 0 επιτρέπουν το λευκό θόρυβο να ακουστεί από κάποιο κανάλι, ενώ όταν βρίσκονται σε 1 τον απαγορεύουν. Για να ακουστεί ο θόρυβος που θέλουμε πρέπει να έχουμε δώσει κατάλληλη τιμή στον καταχωρητή 6 και να έχουμε το πλάτος ήχου που θέλουμε, στους καταχωρητές 8, 9 και 10 (όπως θα εξηγήσουμε παρακάτω).

Επίσης έχουμε τη δυνατότητα να ακούμε από κάποιο κανάλι (ή κανάλια) ταυτόχρονα ήχο και θόρυβο. Αυτό γίνεται όταν θέσουμε τα αντίστοιχα bit του θορύβου και του ήχου στον καταχωρητή 7 σε κατάσταση 0.

Οι **καταχωρητές 8-10** ελέγχουν το πλάτος του ήχου των τριών καναλιών του προγραμματιζόμενου ολοκληρωμένου για τον ήχο. Είναι όλοι των 5-bit - που σημαίνει ότι μπορούν να έχουν μια τιμή από 0 μέχρι 31. Η ένταση του ήχου σε κάθε κανάλι καθορίζεται από τα bit 0-3. Αν η τιμή τους είναι 0, δε θα έχουμε καθόλου ήχο, ενώ αν είναι 15 έχουμε την υψηλότερη δυνατή ένταση. Το bit 4 έχει μια ειδική σημασία. Αν είναι 1 τότε ο έλεγχος του πλάτους δίνεται σε ένα από τα ενσωματωμένα Envelopes του τσιπ τα οποία θα εξετάσουμε παρακάτω, αντί για τα bit 0-3 του καταχωρητή. Ο καταχωρητής 8 ελέγχει το πλάτος του καναλιού 1, ο καταχωρητής 9 του καταλιού 2, και ο καταχωρητής 10 του καναλιού 3.

Υπάρχουν δύο παράμετροι που καθορίζουν τα Envelopes που παρέχει το ίδιο το τσιπ. Η πρώτη είναι η περίοδος και η δεύτερη το «σχήμα» του Envelope.

Οι **καταχωρητές 11 και 12** δίνουν αυτήν ακριβώς την περίοδο. Σχηματίζουν μαζί ένα ζεύγος των 16 bit, όπου ο καταχωρητής 11 είναι το χαμηλό byte και ο καταχωρητής 12 το υψηλό byte. Η τιμή που παίρνουν αυτοί οι δύο καταχωρητές είναι ανάλογη με την περίοδο. Δηλαδή,

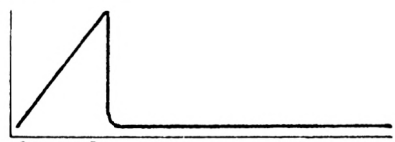
όσο μεγαλύτερη τιμή έχουν, τόσο πιο «αργός» ακούγεται ο ήχος.

Επίσης ο **καταχωρητής 13** με το περιεχόμενό του καθορίζει το σχήμα του Envelope όπως φαίνεται στον παρακάτω πίνακα.

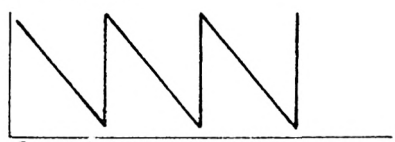
ΠΕΡΙΕΧΟΜΕΝΑ ΚΑΤΑΧΩΡΗΤΗ 13 ΣΧΗΜΑ ENVELOPE

1

4

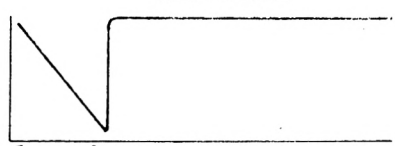


8

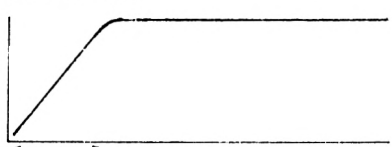


10

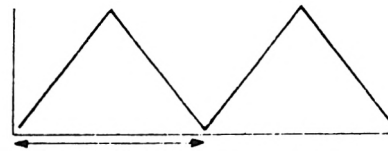
11



13



14



Το αν θα ισχύσει ή όχι το Envelope για έναν ήχο που θα ακουστεί από δεδομένο κανάλι, εξαρτάται απ'το bit-4 των καταχωρητών 8-10 που περιγράψαμε προηγουμένως. Όλα τα κανάλια που έχουν αυτό το bit υψηλό (1) θα «παίξουν» τους ήχους τους με το ίδιο Envelope. Αυτό συμβαίνει γιατί κατ'ουσίαν υπάρχει μόνο ένας καταχωρητής για το σχήμα του Envelope, και μόνο ένα ζευγάρι καταχωρητών που καθορίζουν την περίοδο.

ΤΕΧΝΙΚΕΣ ΗΧΟΥ

Σύμφωνα μ'αυτά που είπαμε, φαίνεται καθαρά ότι είναι σχετικά εύκολο να προγραμματίσουμε απ'ευθείας το τσιπάκι του ήχου. Ωστόσο σίγουρα θα χρειαστήτε αρκετές προσπάθειες πριν ακούσετε τον ήχο που θέλετε απ'το μεγαφωνάκι του Amstrad.

Η καλύτερη μέθοδος είναι να φορτώσετε τους διάφορους καταχωρητές, ενώ έχετε «κλείσει» όλα τα κανάλια (δίνοντας στον καταχωρητή 7 τιμή &X00111111), και αφού τοποθετήσετε παντού τις τιμές που θέλετε, αλλάζετε την τιμή του καταχωρητή 7 ώστε να ακουστεί ο πολυπόθητος ήχος.

Μόλις αρχίσει να ακούγεται ο ήχος που θέλετε, μπορείτε να αλλάξετε τα περιεχόμενα των καταχωρητών. Έτσι έχετε τη δυνατότητα να πετύχετε ό,τι αλλαγές ή εφφέ θέλετε (π.χ. εξασθένιση κάποιου ήχου). Ακριβώς το ίδιο σύστημα ακολουθεί και το λειτουργικό σύστημα του Amstrad, και το κάνει ελέγχοντας συνεχώς τους καταχωρητές της γεννήτριας ήχου, μέσω interrupts. Αυτό όμως σημαίνει αντίστοιχα ότι η CPU καθυστερεί τις εργασίες της (το τρέξιμο κάποιου προγράμματος) για τον έλεγχο του τσιπ ήχου.

Εδώ βοηθάει πάρα πολύ η ύπαρξη έτοιμων Envelopes μέσα στο ίδιο το τσιπ (που περιγράψαμε για τους καταχωρητές 11-13). Με τη χρήση αυτών μπορούμε να «ανοίξουμε» έναν ήχο (μέσω του καταχωρητή 7) και μετά να αφήσουμε τη CPU να εκτελέσει ανενόχλητη τις εργασίες της.

Κλείνοντας αυτό το κεφάλαιο, παραθέτουμε μερικά παραδειγματικά εφφέ απλώς για να πάρετε μια γεύση του τι μπορείτε να πετύχετε. Τις τιμές αυτές μπορείτε να τις βάλετε στους αντίστοιχους καταχωρητές

με τη βοήθεια της ρουτίνας που σας παρουσιάσαμε στην αρχή του κεφαλαίου.

Σημειώνουμε ότι καταχωρητές που δεν αναφέρονται υποτίθεται ότι περιέχουν 0.

ΕΚΡΗΞΗ

ΚΑΤΑΧΩΡΗΤΗΣ	ΤΙΜΗ
6	30
7	&X00110111
8	16
11	255
12	30
13	1

ΤΡΑΙΝΟ

6	30
7	&X00110111
8	16
11	255
12	0
13	14

ΛΕΗΖΕΡ

0	255
1	0
7	&X00111110
8	16
11	100
12	0
13	10

ΜΠΟΙΝΓΚ

0	0
1	9
7	&X00111110
8	16
11	191
12	10
13	1

ΠΥΡΟΒΟΛΙΣΜΟΣ

6	30
7	&X00110111
8	16
11	255
12	2
13	1

5ο ΚΕΦΑΛΑΙΟ

ΧΕΙΡΙΣΜΟΣ ΚΑΣΕΤΑΣ ΑΠΟ ΚΩΔΙΚΑ ΜΗΧΑΝΗΣ

Υπάρχει ένα πλήθος από ρουτίνες, που μας παρέχει το λειτουργικό σύστημα του Amstrad, ειδικά για το χειρισμό της κασέτας. Βέβαια, και η BASIC δίνει επαρκείς δυνατότητες, όμως κυρίως στις περιπτώσεις που θέλουμε να γράψουμε utilities η χρήση των ρουτίνων αυτών είναι απαραίτητη. Επίσης σας δίνουμε τον τρόπο που θα μπορέσετε - εφόσον θέλετε - να σώσετε τα προγράμματά σας σε μορφή headerless για ταχύτερο φόρτωμα και... καλύτερο κλείδωμα.

Οι κάτοχοι των μοντέλων CPC 664 και 6128 θα πρέπει στις περισσότερες ρουτίνες που παρουσιάζουμε σ'αυτό το κεφάλαιο να δώσουν πριν τις χρησιμοποιήσουν την εντολή ITAPE.

Ο ΕΛΕΓΧΟΣ ΤΟΥ ΜΟΤΕΡ

Ένα χρήσιμο χαρακτηριστικό του Amstrad είναι ότι το μοτέρ του κασετοφώνου ελέγχεται από τον υπολογιστή. Έτσι ακόμα κι αν έχουμε πατημένο το PLAY, το κασετόφωνο δε δουλεύει αν δεν του δώσουμε κάποια σχετική εντολή (π.χ. CAT, LOAD ή SAVE).

Για τον έλεγχο του μοτέρ υπάρχουν δύο ρουτίνες του λειτουργικού, η &BC6E που το «ανοίγει» και η &BC71 που το «κλείνει». Δώστε λοιπόν CALL &BC6E και το κασετόφωνο θα αρχίσει να λειτουργεί. Εδώ πρέπει να σημειώσουμε ότι θα περάσει λίγος χρόνος μέχρι να ξαναεμφανιστεί το γνωστό μας "Ready", αλλά μην ανησυχήσετε. Ο υπολογιστής απλά περιμένει να σιγουρευτεί ότι η ταχύτητα του μοτέρ έγινε ομαλή πριν αφήσει τον έλεγχό του. Τις ρουτίνες αυτές μπορείτε να τις βρείτε στο 8ο κεφάλαιο του βιβλίου μαζί με τις υπόλοιπες ρουτίνες που αφορούν το κασετόφωνο.

HEADER READER

Η ρουτίνα που παρουσιάζουμε εδώ διαβάζει τον header κάποιου προγράμματος σε κασέτα και μας δίνει χρήσιμες πληροφορίες για το αντίστοιχο πρόγραμμα. Σας δίνουμε πρώτα τη ρουτίνα σε κώδικα μηχανής που φορτώνει τον header, και στη συνέχεια μια ρουτίνα σε BASIC που τυπώνει στην οθόνη τις πληροφορίες για το πρόγραμμα που μας ενδιαφέρουν. Για χρήση της ρουτίνας από κώδικα μηχανής απλώς την καλούμε οπότε φορτώνεται ο header στη διεύθυνση που έχουμε καθορίσει σαν buffer. Η ρουτίνα του λειτουργικού που χρησι-

μοποιείται είναι η &BCA1 η οποία απευθύνεται πάντα στην κασέτα, ανεξάρτητα ποιό μοντέλο έχουμε και αν έχουμε δώσει προηγουμένως την εντολή TAPE. Η ακριβής χρήση καθώς και το πως καλείται η ρουτίνα αυτή περιγράφονται στο 8ο Κεφάλαιο του βιβλίου.

Για να ερευνήσετε τον Header κάποιου προγράμματος από κασέτα δεν έχετε παρά να πληκτρολογήσετε το LISTING της BASIC που από μόνο του κάνει POKE τον κώδικα μηχανής οποίος θα φορτώσει τον Header.

```
Hisoft GENA3.1 Assembler. Page      1.

Pass 1 errors: 00

A410                10          org   42000
A410  3E2C          20          LD    A,44                ;expected sync. code
A412  21409C        30          LD    HL,40000            ;load address
A415  114000        40          LD    DE,64               ;length of bytes
A418  CDA1BC        50          CALL  48289               ;load header
A41B  C9            60          RET

Pass 2 errors: 00

Table used:     13  from    116

10 J=42000
20 RESTORE
30 READ A$
40 IF A$="TELOS" THEN GOTO 1000
50 FOR I=1 TO LEN(A$) STEP 2
60 POKE J,VAL("&"+MID$(A$,I,2))
70 J=J+1
80 NEXT
90 GOTO 30
100 DATA 3E2C21409C114000CDA1BCC9
110 DATA TELOS
1000 CALL 42000:NAME$=""
1010 FOR I=40000 TO 40015:NAME$=NAME$+CHR$(PEEK(I))
1020 NEXT I
1030 START=PEEK(40021)+256*PEEK(40022)
1040 LENGTH=PEEK(40024)+256*PEEK(40025)
1045 EXE=PEEK(40026)+256*PEEK(40027)
1050 MODE 1:PRINT NAME$
1060 PRINT "Length:__________";LENGTH
1070 PRINT "Start address:___";START
1080 PRINT "Exec. address:___";EXE
1090 END
```

Στο επόμενο κεφάλαιο που θα ασχοληθούμε με τη δισκέτα θα δώσουμε ένα γενικό HEADER READER που θα μπορεί να διαβάσει Header είτε από δισκέτα είτε από κασέτα. Στη συνέχεια παραθέτουμε έναν πίνακα, στον οποίο περιγράφεται το τί περιέχουν τα 64 bytes του Header.

BYTES 0-15: Όνομα προγράμματος
BYTE 16: Αριθμός block.
BYTE 17: Μη μηδενικό (&FF) αν είναι το τελευταίο block.
BYTE 18: Τύπος προγράμματος:
Bit 0: 1για προστασία
Bits 1-3: 0 = Basic
1 = Binary

2 = Screen Image
3 = ASCII
4-7 = Αχρησιμοποίητα
Bits 4-7: Version (1 για ASCII αλλιώς 0)
BYTES 19/20: Μήκος δεδομένων block.
BYTES 21/22: Διεύθυνση δεδομένων (που φορτώνονται).
BYTE 23: Μη μηδενικό (&FF) αν είναι το πρώτο block.
BYTES 24/25: Αριθμός bytes στο πρόγραμμα.
BYTES 26/27: Διεύθυνση που αρχίζει η εκτέλεση (για κώδικα μηχανής).
BYTES 28/63: Αχρησιμοποίητα.

ΡΟΥΤΙΝΑ SAVE

Στη συνέχεια σας παρουσιάζουμε μια πολύ απλή ρουτίνα που κάνει SAVE ένα πρόγραμμα, από κώδικα μηχανής. Εφόσον είναι ακριβώς αντίστοιχη της εντολή SAVE της BASIC χρησιμεύει μόνο για προγράμματα που γράφουμε σε κώδικα μηχανής και έτσι δε δίνουμε αντίστοιχο LISTING για να φορτωθεί από BASIC. Η ρουτίνα αυτή δουλεύει αποκλειστικά σε κασέτα ανεξάρτητα το ποιό μοντέλο διαθέτουμε.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Οπως φαίνεται στην εντολή 20 φορτώνουμε στον IX τη διεύθυνση του HEADER όπως τον περιγράψαμε παραπάνω.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ολοι οι καταχωρητές έχουν τυχαίο περιεχόμενο.

```
Hisoft GENA3.1 Assembler. Page      1.

Pass 1 errors: 00

A410              10          org  42000
A410              15          ENT  $
A410  DD2135A4    20          LD   IX,HEAD
A414  DDE5        30          PUSH IX
A416  2135A4      40          LD   HL,HEAD          ;header address
A419  114000      50          LD   DE,64            ;header length
A41C  3E2C        60          LD   A,44             ;sync. code for header
A41E  CD9EBC      70          CALL 48286            ;save header
A421  DDE1        80          POP  IX               ;restore IX
A423  DD6E15      90          LD   L,(IX+21)        ;address of data
A426  DD6616      100         LD   H,(IX+22)        ;in HL
A429  DD5E13      110         LD   E,(IX+19)        ;length of data
A42C  DD5614      120         LD   D,(IX+20)        ;in DE
A42F  3E16        130         LD   A,22             ;sync. code for data
A431  CD9EBC      140         CALL 48286            ;save data
A434  C9          150         RET
A435  54455854    160 HEAD:   DEFM "TEXT"
A439              170         DEFS 12
A445  01          180         DEFB 1
A446  00          190         DEFB 0
A447  00          200         DEFB 0
A448  E8          210         DEFB 232
A449  03          220         DEFB 3
A44A  69          230         DEFB 105
A44B  01          240         DEFB 1
```

```
A44C  00          250          DEFB 0
A44D  00          260          DEFB 0

Pass 2 errors: 00

Table used:     24  from    156
Executes: 42000
```

Προσοχή γιατί όπως είπαμε στην περιγραφή του Header, πρώτο βρίσκεται το όνομα (bytes 0 - 15), οπότε αν δεν υπάρχουν 16 χαρακτήρες για να γεμίσει ο χώρος αυτός, τα υπόλοιπα bytes πρέπει να είναι 0.

Επίσης μελετήστε τους κωδικούς συγχρονισμού που μπαίνουν στον A (&2C (44) για Header ή &16 (22) για δεδομένα) οι οποίοι περιγράφονται και στο κεφάλαιο 8, στις αντίστοιχες ρουτίνες του λειτουργικού.

ΡΟΥΤΙΝΑ LOAD ΑΠΟ ΚΩΔΙΚΑ ΜΗΧΑΝΗΣ

Η ρουτίνα που παρουσιάζουμε εδώ κάνει με κώδικα μηχανής το αντίστοιχο LOAD της BASIC και μπορείτε να τη συμπεριλάβετε στο πρόγραμμά σας. Δε χρησιμεύει στην BASIC γι' αυτό και δε δίνουμε τον τρόπο χρήσης της παρά μόνο για προγράμματα σε κώδικα μηχανής.

Είναι απαραίτητο να δώσουμε το όνομα του προγράμματος που θέλουμε να φορτώσουμε (στη μεταβλητή NAME) καθώς και έναν buffer για τον Header (δηλ. την περιοχή της μνήμης όπου θα αποθηκευτούν τα αντίστοιχα 64 bytes).

```
Hisoft GENA3.1 Assembler. Page     1.

Pass 1 errors: 00

A410              10          org  42000
A410              15          ENT  $
A410  DD21C5A4    20          LD   IX,SEAR
A414  CD69A4      30          CALL SPRINT              ;print message
A417  DD21ADA4    40 LOAD:    LD   IX,OK
A41B  CD69A4      50          CALL SPRINT
A41E  CD16BB      60          CALL 47894               ;wait for key
A421  FE79        70          CP   121                 ;is it "Y" ?
A423  2803        80          JR   Z,YES
A425  FE59        90          CP   89                  ;is it "y" ?
A427  C0         100          RET  NZ
A428  114000     110 YES:     LD   DE,64               ;length of header
A42B  21FCA4     120          LD   HL,HEAD             ;load address of header
A42E  3E2C       130          LD   A,44                ;sync. code for header
A430  CDA1BC     140          CALL 48289               ;load header
A433  3056       150          JR   NC,FAULT            ;test error
A435  21FCA4     160          LD   HL,HEAD             ;point to filename
A438  11EAA4     170          LD   DE,NAME             ;and desired name
A43B  1A         180 LOOP:    LD   A,(DE)
A43C  BE         190          CP   (HL)                ;check each character
A43D  20D8       200          JR   NZ,LOAD             ;if check fails,again
A43F  23         210          INC  HL
A440  13         220          INC  DE                  ;next character
A441  1A         230          LD   A,(DE)
A442  FE00       240          CP   0                   ;end of name ?
A444  20F5       250          JR   NZ,LOOP             ;if not end,again
A446  DD21D8A4   260          LD   IX,LDING
A44A  CD69A4     270          CALL SPRINT
A44D  DD21EAA4   280          LD   IX,NAME             ;load address by us
A451  DD6E10     290          LD   L,(IX+16)
```

```
A454  DD6611      300          LD   H,(IX+17)          ;load address in HL
A457  DD21FCA4    310          LD   IX,HEAD           ;pick header
A45B  DD5E18      320          LD   E,(IX+24)
A45E  DD5619      330          LD   D,(IX+25)         ;length in HL
A461  3E16        340          LD   A,22              ;sync code for data
A463  CDA1BC      350          CALL 48289             ;load data
A466  3023        360          JR   NC,FAULT
A468  C9          370          RET
A469  DDE5        380 SPRINT:  PUSH IX
A46B  CD54BB      390          CALL 47956             ;VDU enable
A46E  3E0D        400          LD   A,13
A470  CD5ABB      410          CALL 47962             ;do line feed
A473  3E0A        420          LD   A,10
A475  CD5ABB      430          CALL 47962             ;down one line
A478  DDE1        440          POP  IX
A47A  DD7E00      450 LOOP2:   LD   A,(IX)            ;get character
A47D  FE00        460          CP   0                 ;is it terminator ?
A47F  C8          470          RET  Z                 ;if yes,finished sprint
A480  DDE5        480          PUSH IX
A482  CD5DBB      490          CALL 47965             ;print character
A485  DDE1        500          POP  IX
A487  DD23        510          INC  IX                ;next character
A489  18EF        520          JR   LOOP2             ;do it again
A48B  DD2198A4    530 FAULT:   LD   IX,ERROR
A48F  CD69A4      540          CALL SPRINT            ;print error message
A492  3E07        550          LD   A,7
A494  CD5ABB      560          CALL 47962             ;beep
A497  C9          570          RET
A498  416E2065    580 ERROR:   DEFM "An error has occured"
A4AC  00          585          DEFB 0
A4AD  436F6E74    590 OK:      DEFM "Continue search ? (Y/N)"
A4C4  00          595          DEFB 0
A4C5  53656172    600 SEAR:    DEFM "Searching........."
A4D7  00          605          DEFB 0
A4D8  4C6F6164    610 LDING:   DEFM "Loading........."
A4E9  00          615          DEFB 0
A4EA  54455854    620 NAME:    DEFM "TEXT"
A4EE  00          630          DEFB 0
A4EF              640          DEFS 11
A4FA  409C        650          DEFW 40000             ;load address
A4FC              660 HEAD:    DEFS 64

Pass 2 errors: 00

Table used:   148  from   284
Executes: 42000
```

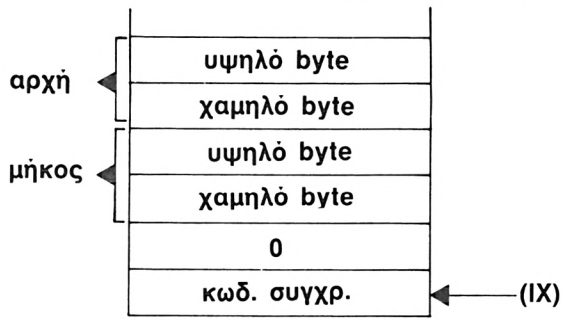
Πολύ ενδιαφέροντα θα βρείτε τα μηνύματα "Searching" και "Continue Search?" τα οποία έχουν περιληφθεί στη ρουτίνα. Απαντώντας στο δεύτερο με "Y" ή "y" θα συνεχιστεί το ψάξιμο ενώ σε οποιαδήποτε άλλη περίπτωση θα βγείτε από τη ρουτίνα. Βέβαια αυτή η ερώτηση θα εμφανίζεται κάθε φορά που συναντάμε ένα πρόγραμμα με διαφορετικό όνομα από αυτό που του έχουμε δώσει να φορτώσει.

HEADERLESS ΣΩΣΙΜΟ ΚΑΙ ΦΟΡΤΩΜΑ

Στη συνέχεια θα σας δώσουμε δύο ρουτίνες που θα διαβάζουν και θα σώσουν προγράμματα χωρίς Header (σε ένα μονοκόμματο block). Μπορείτε ένα πρόγραμμά σας (κυρίως binary) να το σώσετε σε μορφή Headerless με την πρώτη ρουτίνα, αλλά για να το ξαναφορτώσετε θα πρέπει να χρησιμοποιήσετε τη δεύτερη ρουτίνα γιατί δεν μπορεί να το αναγνωρίσει η Basic του Amstrad. Και οι δύο ρουτίνες μπορούν να φορτωθούν σε οποιαδήποτε θέση της μνήμης.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ ΡΟΥΤΙΝΑΣ 1: CALL 42000, αρχή, μήκος, κωδ. συγχρ.: όπου «αρχή» η αρχή και «μήκος» το μήκος αντίστοιχα του προγράμματος που θέλουμε να σώσουμε και «κωδ. συγχρ.» είναι ο κωδικός συγχρονισμού που μπορεί να είναι ένας αριθμός μέχρι 255.

Για χρήση της ρουτίνας από κώδικα μηχανής πρέπει ο A να περιέχει 3 και ο IX να δείχνει το παρακάτω block μεταβλητών.



ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Όλοι οι καταχωρητές έχουν τυχαίο περιεχόμενο.

```
Hisoft GENA3.1 Assembler. Page      1.

Pass 1 errors: 00

A410                10          ORG  42000
A410  FE03          20          CP   3                    ;three parameters
A412  C0            30          RET  NZ
A413  DD7E00        40          LD   A,(IX+0)             ;sync code in A
A416  DD5E02        50          LD   E,(IX+2)             ;data length
A419  DD5603        60          LD   D,(IX+3)             ;in DE
A41C  DD6E04        70          LD   L,(IX+4)             ;address to start saving
A41F  DD6605        80          LD   H,(IX+5)             ;in HL
A422  CD9EBC        90          CALL 48286                ;save data
A425  C9           100          RET

Pass 2 errors: 00
Table used:     13  from   126
```

```
10 J=42000
20 RESTORE
30 READ A$
40 IF A$="TELOS" THEN END
50 FOR I=1 TO LEN(A$) STEP 2
60 POKE J,VAL("&"+MID$(A$,I,2))
70 J=J+1
80 NEXT
90 GOTO 30
100 DATA FE03C0DD7E00DD5E02DD5603DD6E04DD6605CD9EBCC9
110 DATA TELOS
```

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ ΡΟΥΤΙΝΑΣ 2: Ομοιες ακριβώς με της προηγούμενης.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ολοι οι καταχωρητές έχουν τυχαίο περιεχόμενο.

```
Hisoft GENA3.1 Assembler. Page      1.

Pass 1 errors: 00

A410                10          ORG  42000
A410  FE03          20          CP   3                    ;three parameters
A412  C0            30          RET  NZ
A413  DD7E00        40          LD   A,(IX+0)             ;sync code in A
A416  DD5E02        50          LD   E,(IX+2)             ;data length
A419  DD5603        60          LD   D,(IX+3)             ;in DE
A41C  DD6E04        70          LD   L,(IX+4)             ;loading address
A41F  DD6605        80          LD   H,(IX+5)             ;in HL
A422  CDA1BC        90          CALL 48289                ;load data
A425  C9           100          RET

Pass 2 errors: 00

Table used:     13  from    125
10 J=42000
20 RESTORE
30 READ A$
40 IF A$="TELOS" THEN END
50 FOR I=1 TO LEN(A$) STEP 2
60 POKE J,VAL("&"+MID$(A$,I,2))
70 J=J+1
80 NEXT
90 GOTO 30
100 DATA FE03C0DD7E00DD5E02DD5603DD6E04DD6605CDA1BCC9
110 DATA TELOS
```

Οπως είπαμε και πριν, ένα πρόγραμμα που έχετε σώσει σαν Headerless με την πρώτη ρουτίνα, μπορείτε να το ξαναφορτώσετε μόνο με τη χρήση της δεύτερης (ή κάποιας ανάλογης).

VERIFY

Αυτή η ρουτίνα ελέγχει κάποιο πρόγραμμα που έχουμε σώσει σε κασέτα, αν έχει ακριβώς τα ίδια bytes με το πρόγραμμα που βρίσκεται στη μνήμη του υπολογιστή. Είναι χρήσιμο να ξέρετε κάθε φορά αν το πρόγραμμα που έχετε σώσει έχει γραφτεί σωστά στην κασέτα, πριν το σβήσετε από τη μνήμη του υπολογιστή.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: CALL 42000, @α$ όπου στη μεταβλητή α$ έχουμε βάλει το όνομα του προγράμματος (με κεφαλαία).

Από κώδικα μηχανής μπορεί να γίνει η κλήση της ρουτίνας με τον A να περιέχει 1 και τον IX να δείχνει σε μια θέση μνήμης όπου υπάρχει η διεύθυνση που φυλάσσονται οι πληροφορίες για το α$ (όπως περιγράφτηκε στο κεφάλαιο 1).

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ολοι οι καταχωρητές έχουν τυχαίο περιεχόμενο.

```
Hisoft GENA3.1 Assembler. Page     1.

Pass 1 errors: 00

A410                10          org   42000
A410  FE01          20          CP    1             ;one parameter
A412  C0            30          RET   NZ
A413  DD6E00        40          LD    L,(IX+0)      ;get address of a$
A416  DD6601        50          LD    H,(IX+1)      ;inf. into HL
A419  E5            60          PUSH  HL            ;and then
A41A  DDE1          70          POP   IX            ;into IX
A41C  DD7E00        80          LD    A,(IX+0)      ;get len(a$)
A41F  DD23          90          INC   IX
A421  DD6E00        100         LD    L,(IX+0)      ;a$ address
A424  DD6601        110         LD    H,(IX+1)      ;into HL
A427  E5            120         PUSH  HL            ;and then
A428  DDE1          130         POP   IX            ;in IX
A42A  32FEA4        140         LD    (LEN),A
A42D  DD22FCA4      150         LD    (NAME),IX
A431  DD21DAA4      160         LD    IX,SEAR       ;print search
A435  CD88A4        170         CALL  SPRINT        ;message
A438  DD21C8A4      180 LOAD:   LD    IX,CONT
A43C  CD88A4        190         CALL  SPRINT
A43F  CD18BB        200         CALL  47896         ;wait key
A442  FE79          210         CP    121           ;is it "y" ?
A444  2803          220         JR    Z,YES
A446  FE59          230         CP    89            ;is it "Y" ?
A448  C0            240         RET   NZ
A449  114000        250 YES:    LD    DE,64         ;64 bytes in header
A44C  21FFA4        260         LD    HL,HEAD       ;address to load it
A44F  3E2C          270         LD    A,44          ;sync code of header
A451  CDA1BC        280         CALL  48289         ;load header
A454  3054          290         JR    NC,FAULT      ;is all OK ?
A456  21FFA4        300         LD    HL,HEAD       ;header address in HL
A459  ED5BFCA4      310         LD    DE,(NAME)     ;address of a$ in DE
A45D  3AFEA4        320         LD    A,(LEN)       ;len(a$) in A
A460  47            330         LD    B,A           ;and then in B
A461  1A            340 LOOP1:  LD    A,(DE)        ;compare desired
A462  BE            350         CP    (HL)          ;name of file
A463  20D3          360         JR    NZ,LOAD
A465  23            370         INC   HL
A466  13            380         INC   DE
A467  10F8          390         DJNZ  LOOP1
A469  DD21EBA4      400         LD    IX,LDING
A46D  CD88A4        410         CALL  SPRINT
A470  DD21FFA4      420         LD    IX,HEAD       ;pick load address
A474  DD6E15        430         LD    L,(IX+21)     ;in HL
A477  DD6616        440         LD    H,(IX+22)
A47A  DD5E18        450         LD    E,(IX+24)     ;data length in DE
A47D  DD5619        460         LD    D,(IX+25)
A480  3E16          470         LD    A,22          ;sync code
A482  CDA4BC        480         CALL  48292         ;check record
A485  3023          490         JR    NC,FAULT
A487  C9            500         RET
A488  DDE5          510 SPRINT: PUSH  IX
A48A  CD54BB        520         CALL  47956         ;enable text VDU
A48D  3E0D          530         LD    A,13
A48F  CD5ABB        540         CALL  47962         ;do carriage return
A492  3E0A          550         LD    A,10
A494  CD5ABB        560         CALL  47962         ;do line feed
A497  DDE1          570         POP   IX
A499  DD7E00        580 LOOP2:  LD    A,(IX)
A49C  FE00          590         CP    0             ;did it finish ?
A49E  C8            600         RET   Z             ;if so return
A49F  DDE5          610         PUSH  IX
A4A1  CD5DBB        620         CALL  47965         ;print character
A4A4  DDE1          630         POP   IX
A4A6  DD23          640         INC   IX
A4A8  18EF          650         JR    LOOP2         ;do it again
A4AA  DD21B7A4      660 FAULT:  LD    IX,ERROR
A4AE  CD88A4        670         CALL  SPRINT
A4B1  3E07          680         LD    A,7
A4B3  CD5ABB        690         CALL  47962         ;beep
A4B6  C9            700         RET
```

```
A4B7  56657269    710 ERROR:  DEFM "Verify error !!!"
A4C7  00          720         DEFB 0
A4C8  436F6E74    730 CONT:   DEFM "Continue search ?"
A4D9  00          740         DEFB 0
A4DA  53656172    750 SEAR:   DEFM "Searching......."
A4EA  00          760         DEFB 0
A4EB  56657269    770 LDING:  DEFM "Verifying......."
A4FB  00          780         DEFB 0
A4FC              790 NAME:   DEFS 2
A4FE              800 LEN:    DEFS 1
A4FF              810 HEAD:   DEFS 64

Pass 2 errors: 00

Table used:   161  from   296
```

```
10 J=42000
20 RESTORE
30 READ A$
40 IF A$="TELOS" THEN END
50 FOR I=1 TO LEN(A$) STEP 2
60 POKE J,VAL("&"+MID$(A$,I,2))
70 J=J+1
80 NEXT
90 GOTO 30
100 DATA FE01C0DD6E00DD6601E5DDE1DD7E00DD23DD6E00DD6601E5DDE132FEA4DD
110 DATA 22FCA4DD21DAA4CD88A4DD21CBA4CD88A4CD1EBdFE792803FE59C0114000
120 DATA 21FFA43E2CCDA1BC305421FFA4ED5BFCA43AFEA4471ABE20D3231310F8DD
130 DATA 21EBA4CD88A4DD21FFA4DD6E15DD6616DD5E18DD56193E16CDA4BC3023C9
140 DATA DDE5CD54BB3E0DCD5ABB3E0ACD5ABBDDE1DD7E00FE00C8DDE5CD5DBBDDE1
150 DATA DD2318EFDD21B7A4CD88A43E07CD5ABBC9566572696679206572726F7220
160 DATA 2121210043 6F6E74696E756520736561726368203F00536561726368696E
170 DATA 672E2E2E2E2E2E2E00566572696679696E672E2E2E2E2E2E0000000000
180 DATA TELOS
```

Εδώ κλείνουμε το κεφάλαιο χειρισμού της κασέτας. Περισσότερες πληροφορίες για τις ρουτίνες του λειτουργικού που χρησιμοποιήσαμε μπορείτε να βρείτε στο 8ο κεφάλαιο του βιβλίου.

Επίσης στο επόμενο κεφάλαιο θα δώσουμε ένα αντιγραφικό πρόγραμμα που μπορεί να λειτουργεί είτε σε κασέτα είτε σε δισκέτα το οποίο πιστεύουμε να βρείτε πολύ ικανοποιητικό.

6ο ΚΕΦΑΛΑΙΟ

ΧΕΙΡΙΣΜΟΣ ΔΙΣΚΕΤΑΣ ΚΑΙ RANDOM ACCESS

Σ'αυτό το κεφάλαιο θα δούμε πώς μπορούμε να χειριστούμε τη δισκέτα μέσα από κώδικα μηχανής, για όσους έχουν τον 664, τον 6128 ή έχουν το DDI Disc Drive της Amstrad για τον 464 (μερικές ρουτίνες μπορούν να χρησιμοποιηθούν και με κασέτα).

Όταν υπάρχει disc drive και είναι «ενεργοποιημένο» (δεν έχει δοθεί εντολή TAPE) μερικές ρουτίνες του λειτουργικού που απευθύνονταν στην κασέτα έχουν «γυρίσει» στο δίσκο. Με την εντολή TAPE μπορούμε να τις «γυρίσουμε» πάλι στην κασέτα όποτε θελήσουμε. Έτσι οι περισσότερες από τις ρουτίνες που παρουσιάζουμε εδώ, μπορούν να χρησιμοποιηθούν και για την κασέτα.

ΑΝΑΓΝΩΣΗ HEADER.

Το παρακάτω προγραμμάτων αποτελείται από ένα κομμάτι BASIC και ένα κομμάτι κώδικα μηχανής. Το LISTING του κώδικα μηχανής σε ASSEMBLY δίνεται στο LISTING 2 ώστε να μπορέσετε να το χρησιμοποιήσετε αν θέλετε σε διάφορα προγράμματά σας. Η ρουτίνα αυτή διαβάζει τον Header είτε από κασέτα είτε από δισκέτα και μας δίνει χρήσιμες πληροφορίες για όποιο πρόγραμμα θέλουμε. Για να δείτε πώς είναι ταξινομημένα και τι περιέχουν τα 64 bytes του Header μπορείτε να κοιτάξετε στο προηγούμενο κεφάλαιο, στο σχετικό πίνακα.

Τρέξτε το πρόγραμμα και απαντήστε στην ερώτηση αν θέλετε να φορτώσετε από δίσκο ή κασέτα, πατώντας D ή κάποιο άλλο γράμμα αντίστοιχα. Αν έχετε επιλέξει δισκέτα, πρέπει οπωσδήποτε να δώσετε το όνομα του προγράμματος του οποίου θέλετε να διαβάσετε τον Header.

```
10 ON ERROR GOTO 1000
20 MEMORY &8E00
30 INPUT"Press D to read from disc or other key for tape";dd$
40 IF UPPER$(dd$)="D" THEN 60 ELSE |TAPE.IN
50 '
60 length=&9000:type=&9002:location=&9004
70 execute=&9006:header=&9008:buffread=&8F00:inlength=&9010:inname=&9020
80 DIM t$(4)
90 DEF FNget(address)=PEEK(address)+256*PEEK(address+1)
110 FOR x=0 TO 41:READ x$:x$="&"+x$:POKE buffread+x,VAL(x$):NEXT x
130 INPUT "filename";name$:POKE(inlength),LEN(name$)
140 FOR x=1 TO LEN(name$):POKE(inname+x-1),ASC(MID$(name$,x,1)):NEXT x
160 CALL buffread
170 type=(FNget(type)/2) AND 7
180 t$(0)="Internal Basic":t$(1)="Binary file":t$(2)="screen image":t$(3)="ASCII
190 PRINT"Type: ";t$(type)
```

```
200 PRINT"Location: ";HEX$(FNget(location))
210 PRINT"Execution address: ";HEX$(FNget(execute))
220 PRINT"Length: ";HEX$(FNget(length))
230 DATA "3A","10","90","47","21","20","90","11"
240 DATA "40","90","CD","77","BC","ED","43","00"
250 DATA "90","32","02","90","ED","53","04","90"
260 DATA "22","08","90","DD","2A","08","90","DD"
270 DATA "5E","1A","DD","56","1B","ED","53","06"
280 DATA "90","C9"
1000 RESUME NEXT


Hisoft GENA3.1 Assembler. Page      1.

Pass 1 errors: 00

8F00                10          ORG  36608
8F00  3A1090        20          LD   A,(36880)
8F03  47            30          LD   B,A                ;filename length
8F04  212090        40          LD   HL,36896           ;address of filename
8F07  114090        50          LD   DE,36928           ;2K buffer
8F0A  CD77BC        60          CALL 48247              ;openin file
8F0D  ED430090      70          LD   (36864),BC         ;save filelength
8F11  320290        80          LD   (36866),A          ;save filetype
8F14  ED530490      90          LD   (36868),DE         ;save data location
8F18  220890       100          LD   (36872),Hl         ;save address of header
8F1B  DD2A0890     110          LD   IX,(36872)
8F1F  DD5E1A       120          LD   E,(IX+26)          ;execution address
8F22  DD561B       130          LD   D,(IX+27)          ;in DE
8F25  ED530690     140          LD   (36870),DE         ;save it
8F29  C9           150          RET

Pass 2 errors: 00

Table used:     13  from    146
```

ΣΩΣΙΜΟ ΠΡΟΓΡΑΜΜΑΤΟΣ ΣΤΗ ΔΙΣΚΕΤΑ

Η παρακάτω ρουτίνα κάνει SAVE ένα πρόγραμμά σας σε δισκέτα (ή κασέτα αν έχετε δώσει TAPE) από κώδικα μηχανής. Επειδή λειτουργεί όπως ακριβώς η εντολή SAVE της BASIC δίνουμε μόνο τον τρόπο χρήσης της από κώδικα μηχανής.

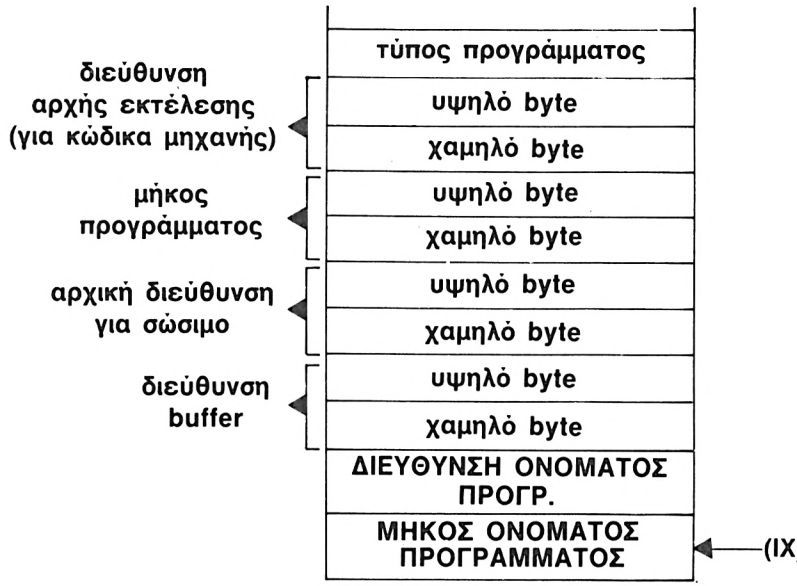
ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: CALL 42000. Η κλήση της ρουτίνας γίνεται με τον IX να δείχνει το ακόλουθο block μεταβλητών.

ΒΛΕΠΕ ΣΕΛ. **112**

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Όλοι οι καταχωρητές έχουν τυχαίο περιεχόμενο.

Οι πρώτες δύο παράμετροι που δείχνει ο IX αφορούν το μήκος του ονόματος που θέλουμε να δώσουμε στο πρόγραμμα και τη διεύθυνση που το έχουμε γράψει (σε κωδικούς ASCII) στη RAM. Στη συνέχεια χρειάζεται ένας buffer των 2K για να ξεκινήσει το σώσιμο.

Δίνουμε λοιπόν την αρχική διεύθυνση των 2K που θα χρησιμοποιηθούν σαν BUFFER. Σε περίπτωση που δεν έχετε 2K διαθέσιμα στην USER RAM μπορείτε να δώσετε τη διεύθυνση #C000 ώστε να χρησι-




```
Hisoft GENA3.1 Assembler. Page      1.

Pass 1 errors: 00

A410                10          org   42000
A410  DD4600        20 WRITE:   LD    B,(IX+0)          ;length of filename
A413  DD6E01        30          LD    L,(IX+1)          ;filename address
A416  DD6602        40          LD    H,(IX+2)          ;in HL
A419  DD5E03        50          LD    E,(IX+3)          ;buffer area
A41C  DD5604        60          LD    D,(IX+4)          ;in DE
A41F  DDE5          70          PUSH  IX
A421  CD8CBC        80          CALL  48268             ;openout file
A424  DDE1          90          POP   IX
A426  3040         100          JR    NC,OPEN
A428  DD6E05       110          LD    L,(IX+5)          ;data address
A42B  DD6606       120          LD    H,(IX+6)          ;in HL
A42E  DD5E07       130          LD    E,(IX+7)          ;data length
A431  DD5608       140          LD    D,(IX+8)          ;in DE
A434  DD4E09       150          LD    C,(IX+9)
A437  DD460A       160          LD    B,(IX+10)
A43A  DD7E0B       170          LD    A,(IX+11)         ;filetype in A
A43D  DDE5         180          PUSH  IX
A43F  CD98BC       190          CALL  48280             ;save data
A442  DDE1         200          POP   IX
A444  3014         210          JR    NC,EP             ;is there an error ?
A446  DDE5         220          PUSH  IX
A448  CD8FBC       230          CALL  48271             ;closeout
A44B  DDE1         240          POP   IX
A44D  C9           250          RET
A44E  DDE5         260 FN:      PUSH  IX                ;File not opened
A450  DD21A6A4     270          LD    IX,FNO
A454  CD84A4       280          CALL  SPRINT
A457  DDE1         290          POP   IX
A459  C9           300          RET
A45A  20F2         310 EP:      JR    NZ,FN             ;is not file opened?
A45C  DDE5         320          PUSH  IX
A45E  DD21BAA4     330          LD    IX,ESC            ;else ESC was pressed
A462  CD84A4       340          CALL  SPRINT
A465  DDE1         350          POP   IX
A467  C9           360          RET
```

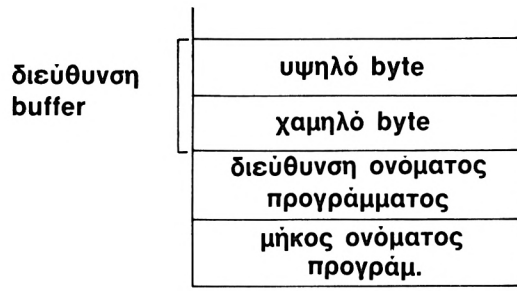
```
A468  DDE5        370 OPEN:   PUSH IX                    ;file allready opened
A46A  DD21DFA4    380         LD   IX,FAO
A46E  CD84A4      390         CALL SPRINT
A471  DD21F6A4    400         LD   IX,RETRY
A475  CD84A4      410         CALL SPRINT
A478  CD18BB      420         CALL 47896                 ;get answer
A47B  FE79        430         CP   121                   ;is it "y" ?
A47D  2891        440         JR   Z,WRITE
A47F  FE59        450         CP   89                    ;is it "Y" ?
A481  288D        460         JR   Z,WRITE
A483  C9          470         RET
A484  DDE5        480 SPRINT: PUSH IX
A486  CD54BB      490         CALL 47956                 ;VDU enable
A489  3E0D        500         LD   A,13                  ;
A48B  CD5ABB      510         CALL 47962                 ;line feed
A48E  3E0A        520         LD   A,10
A490  CD5ABB      530         CALL 47962                 ;one line down
A493  DDE1        540         POP  IX
A495  DD7E00      550 LOOP:   LD   A,(IX+0)              ;get first char
A498  FE00        560         CP   0                     ;is it end marker?
A49A  C8          570         RET  Z
A49B  DDE5        580         PUSH IX
A49D  CD5DBB      590         CALL 47965                 ;print character
A4A0  DDE1        600         POP  IX
A4A2  DD23        610         INC  IX                    ;next character
A4A4  18EF        620         JR   LOOP                  ;repeat
A4A6  46696C65    630 FNO:    DEFM "File not opened...."
A4B9  00          640         DEFB 0
A4BA  45736361    650 ESC:    DEFM "Escaped pressed. Write abandoned...."
A4DE  00          660         DEFB 0
A4DF  46696C65    670 FAO:    DEFM "File already opened...."
A4F6  446F2079    680 RETRY:  DEFM "Do you want to try again ?"
A510  00          690         DEFB 0

Pass 2 errors: 00

Table used:   120  from   263
```

μοποιηθεί σαν BUFFER η VIDEO RAM. Η «διεύθυνση αρχής εκτέλεσης» που υπάρχει στον προηγούμενο πίνακα είναι η διεύθυνση απ' όπου θα αρχίσει να τρέχει ένα πρόγραμμα κώδικα μηχανής όταν το φορτώσουμε με RUN"... Ο τύπος τέλος του προγράμματος είναι ακριβώς το byte 18 του Header που περιγράψαμε στο προηγούμενο κεφάλαιο.

ΦΟΡΤΩΜΑ ΠΡΟΓΡΑΜΜΑΤΟΣ ΑΠΟ ΔΙΣΚΕΤΑ

Στη συνέχεια παρουσιάζουμε μια ρουτίνα σε κώδικα μηχανής που φορτώνει ένα πρόγραμμα από δισκέτα (ή κασέτα αν έχουμε δώσει εντολή TAPE). Επειχή λειτουργεί όπως ακριβώς και η εντολή LOAD της BASIC δίνουμε μόνο τον τρόπο χρήσης της από κώδικα μηχανής. Η ρουτίνα μπορεί να μεταφερθεί σε οποιαδήποτε θέση μνήμης.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: CALL 42000. Η ρουτίνα καλείται έχοντας στον IX τη διεύθυνση του block μεταβλητών που φαίνεται παρακάτω.




```
Hisoft GENA3.1 Assembler. Page      1.

Pass 1 errors: 00

A410                10          org  42000
A410  DD4600        20          LD   B,(IX+0)          ;filename length
A413  DD6E01        30          LD   L,(IX+1)          ;address of filename
A416  DD6602        40          LD   H,(IX+2)          ;in HL
A419  DD5E03        50          LD   E,(IX+3)          ;buffer address
A41C  DD5604        60          LD   D,(IX+4)          ;in DE
A41F  DD7E05        70          LD   A,(IX+5)          ;filetype in A
A422  F5            80          PUSH AF
A423  CD77BC        90          CALL 48247             ;openin file
A426  D0           100          RET  NC
A427  EB           110          EX   DE,HL             ;data address in HL
A428  F1           120          POP  AF
A429  CD83BC       130          CALL 48259             ;read complete file
A42C  CD7ABC       140          CALL 48250             ;close input file
A42F  C9           150          RET

Pass 2 errors: 00
Table used:     13  from    141
```

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Όλοι οι καταχωρητές έχουν τυχαίο περιεχόμενο.

Οι μεταβλητές που χρησιμοποιούνται είναι ίδιες με τις αντίστοιχες μεταβλητές της προηγούμενης ρουτίνας.

ΡΟΥΤΙΝΑ ΑΝΤΙΓΡΑΦΗΣ

Η ρουτίνα που παρουσιάζουμε παρακάτω αντιγράφει ένα πρόγραμμα από κασέτα ή δισκέτα σε κασέτα ή δισκέτα. Αποτελείται από κώδικα μηχανής που κάνει όλη την αντιγραφή (LISTING 1), και το πρόγραμμα BASIC (LISTING 2) που κάνει τον κώδικα μηχανής POKE στην RAM, και περνάει σ'αυτόν το όνομα και τον τύπο του προγράμματος. Η επιλογή κασέτας ή δισκέτας για φόρτωμα ή σώσιμο αντίστοιχα γίνεται δίνοντας προηγουμένως π.χ. την εντολή TAPE.IN για φόρτωμα από κασέτα και σώσιμο σε δισκέτα. Αν έχετε τον 464 χωρίς disc-drive θα πρέπει να παραλείψετε την γραμμή 15, και το πρόγραμα θα αντιγραφεί από κασέτα σε κασέτα κανονικά. Αν διαθέτετε disc-drive, ή κάποιον απ'τα άλλα μοντέλα της Amstrad, η αυτόματη επιλογή (όταν ανοίξετε

τον υπολογιστή) είναι για αντιγραφή από δισκέτα σε δισκέτα. Αν θέλετε λοιπόν να φορτώσετε ή να σώσετε σε κασέτα πρέπει να το καθορίσετε δίνοντας μία από τις εντολές TAPE, TAPE.IN ή TAPE.OUT για αντιγραφή από κασέτα σε κασέτα, από κασέτα σε δισκέτα ή από δισκέτα σε κασέτα αντίστοιχα.

Όταν τρέξετε το πρόγραμμα, θα σας ζητηθεί να δώσετε πρώτα το όνομα του προγράμματος που θέλετε να αντιγράψετε, και τον τύπο του για το σώσιμο (byte 18 του header). Έτσι για παράδειγμα, μπορείτε να φορτώσετε ένα πρόγραμμα σε προστατευμένη BASIC και να το σώσετε απροστάτευτο.

```
Hisoft GENA3.1 Assembler. Page      1.

Pass 1 errors: 00

A68C                10          org  42636
A68C  CD57BB        20          CALL 47959              ;screen disable
A68F  3A7CA6        30          LD   A,(42620)          ;get filename length
A692  47            40          LD   B,A                ;into B
A693  217DA6        50          LD   HL,42621           ;filename address
A696  1100C0        60          LD   DE,49152
A699  CD77BC        70          CALL 48247              ;open input file
A69C  D5            80          PUSH DE
A69D  C5            90          PUSH BC
A69E  EB           100          EX   DE,HL
A69F  CD83BC       110          CALL 48259              ;load input file
A6A2  E5           120          PUSH HL
A6A3  CD7ABC       130          CALL 48250
A6A6  21A700       140          LD   HL,167             ;setup speed write 1
A6A9  3E32         150          LD   A,50
A6AB  CD68BC       160          CALL 48232              ;set cas. speed
A6AE  3A7CA6       170          LD   A,(42620)          ;filename length
A6B1  47           180          LD   B,A                ;into B
A6B2  217DA6       190          LD   HL,42621           ;filename address
A6B5  1100C0       200          LD   DE,49152           ;buffer
A6B8  CD8CBC       210          CALL 48268              ;open output file
A6BB  C1           220          POP  BC                 ;restore save data
A6BC  D1           230          POP  DE
A6BD  E1           240          POP  HL
A6BE  3A8BA6       250          LD   A,(42635)          ;filetype in A
A6C1  CD98BC       260          CALL 48280              ;write output file
A6C4  CD8FBC       270          CALL 48271              ;close output file
A6C7  CD54BB       280          CALL 47956              ;enable screen display
A6CA  C9           290          RET

Pass 2 errors: 00
Table used:     13  from    178
```

```
10 MODE 2
20 GOSUB 130
30 |TAPE.OUT
40 INPUT "ONOMA";a$
50 FOR i=1 TO LEN(A$)
60 POKE 42620+i,ASC(MID$(a$,i,1))
70 NEXT
80 POKE 42620,LEN(a$)
90 INPUT"TYPOS ";B
100 POKE 42635,B
110 CALL 42636
120 GOTO 30
130 J=42636
140 RESTORE
150 READ A$
160 IF A$="TELOS" THEN RETURN
170 FOR I=1 TO LEN(A$) STEP 2
180 POKE J,VAL("&"+MID$(A$,I,2))
190 J=J+1
```

```
200 NEXT
210 GOTO 150
220 DATA CD57BB3A7CA647217DA61100C0CD77BCD5C5EBCD83BCE5CD7ABC21A7003E
230 DATA 32CD68BC3A7CA647217DA61100C0CD8CBCC1D1E13A8BA6CD98BCCD8FBCCD
240 DATA 54BBC9
250 DATA TELOS
```

Ο μόνος περιορισμός του προγράμματος, είναι ότι πρέπει το πρόγραμμα που θέλετε να αντιγράψετε να μην καταλαμβάνει τις θέσεις μνήμης από 42620 και πάνω, γιατί είναι η περιοχή που βρίσκεται ο κώδικας μηχανής. Επίσης μόλις το φορτώσει, περιμένει να πατήσετε κάποιο πλήκτρο πριν ξεκινήσει το σώσιμο, χωρίς να εμφανίζει μήνυμα στην οθόνη (για να μην χαλάνε οι οθόνες που θέλετε να αντιγράψετε). Προσέξτε τα ονόματα που δίνετε να είναι σωστά, ιδίως όταν χρησιμοποιείτε δισκέτα, γιατί δεν εμφανίζονται μηνύματα και μπορεί να νομίζετε ότι το πρόγραμμα αντιγράφτηκε, ενώ δεν έχει γίνει τίποτα τέτοιο.

EDITOR ΔΙΣΚΕΤΑΣ

Το πρόγραμμα που σας παρουσιάζουμε παρακάτω, σας δίνει τη δυνατότητα να αλλάξετε τα περιεχόμενα ενός SECTOR που θέλετε σε κάποια δισκέτα. Γι' αυτό και πρέπει να το χρησιμοποιείτε πολύ προσεκτικά. Ούτε εγώ ούτε οι εκδότες φέρουμε καμία ευθύνη για οποιαδήποτε απώλεια δεδομένων μπορεί να προκύψει από τη χρήση του.

Στο LISTING 1 φαίνεται το πρόγραμμα σε BASIC, ολοκληρωμένο με τον κώδικα μηχανής που γίνεται POKE στη μνήμη. Το LISTING 2 δίνει το κομμάτι του κώδικα μηχανής που χρησιμοποιείται, σε LISTING ASSEMBLY.

Όταν τρέξετε το πρόγραμμα του LISTING 1 θα σας ζητηθεί να δώσετε τον αριθμό του TRACK και του SECTOR που θέλετε να διαβάσετε και να μετατρέψετε. Στη συνέχεια θα εμφανιστεί στην οθόνη σας το πρώτο byte του SECTOR που έχετε επιλέξει. Για να πάτε στο επόμενο byte, απλώς πατάτε ENTER. Αν θέλετε να το αλλάξετε, πριν πατήσετε ENTER δώστε σε δεκαεξαδική μορφή τη νέα τιμή που θέλετε να έχει. Αφού κάνετε ό,τι αλλαγές θέλετε, πατήστε CTRL\ και ENTER για να γράφετε τον SECTOR που έχετε κάνει EDITING ξανά στο δίσκο.

```
10 MODE 1:GOSUB 280:GOSUB 390
20 PRINT#1,TAB(13)"track & sector.":PRINT#1,TAB(13);STRING$(14,"_")
30 PRINT#0:PRINT#0,"track no.-(0 to 39)":INPUT t%
40 IF t%<0 OR t%>39 THEN k%=39:GOSUB 430:GOTO 30
50 PRINT#0:PRINT#0,"sector no. (0 to 8)":INPUT s%
60 IF s%<0 OR s%>8 THEN k%=8:GOSUB 430:GOTO 50
70 s%=s%+1+&40:d%=t%*256+s%
80 CALL &A000,d%
90 CLS#0:CLS#1
100 PRINT#1:PRINT#1,TAB(8)"byte no.";TAB(20)"hex";TAB(27)"char"
110 PRINT#2,"type hex. number to change,enter to"
115 PRINT#2,"ignore, ctrl \ to put back to dick."
120 FOR n%=0 TO 511:k%=PEEK(b%+n%)
130 PRINT#0,TAB(7);n%;TAB(16);HEX$(k%,2);
```

```
140 IF k%<32 THEN k%=127
150 PRINT#0,TAB(24);CHR$(k%)
160 INPUT#3, k$:IF k$=""THEN 220
170 IF k$=CHR$(28) THEN 260
180 IF LEN(k$)>2 THEN PRINT"faulty number":GOTO 160
190 k%=VAL("&"+k$)
200 POKE b%+n%,k%
210 GOTO 130
220 NEXT
230 PRINT#0:PRINT#0,"another one- y or n?"
240 INPUT a$:IF a$="Y" OR a$"y" THEN 20
250 END
260 CALL &A023:REM record
270 END
280 MEMORY &9FFF:b%=&A000
290 INK 0,0:INK 2,26: INK 3,1
300 CLS:WINDOW #1,1,40,1,3
310 WINDOW#2,1,40,22,24
320 WINDOW#3,10,30,25,25
330 WINDOW#0,5,35,4,21
340 BORDER 4
350 PAPER#1,3:PAPER#2,3
360 CLS#1:CLS#2:CLS#3
370 PEN#0,1:PEN#1,2:PEN#2,2
380 RETURN
390 cs%=0:FOR n%=0 TO 61:READ d$:d%=VAL("&"+d$)
400 POKE b%+n%,d%:cs%=cs%+d%:NEXT
410 IF cs%<>7825 THEN PRINT"faulty data,please re-type":END
420 RETURN
430 PRINT#0:PRINT#0,"range 0 to";k%;"only, try again":RETURN
440 DATA dd,7e,00,dd,56,01,32,3e,a0
450 DATA 1e,00,ed,53,3f,a0,21,41,a0
460 DATA f5,0e,07,cd,0f,b9,f1,c5,4f
470 DATA cd,66,c6,c1,cd,18,b9,c9,3a
480 DATA 3e,a0,ed,5b,3f,a0,21,41,a0
490 DATA f5,0e,07,cd,0f,b9,f1,c5,4f
500 DATA cd,4e,c6,c1,cd,18,b9,c9

Hisoft GENA3.1 Assembler. Page     1.

Pass 1 errors: 00

A000                10          org   #A000
A000  DD7E00        20          LD    A,(IX+0)
A003  DD5601        30          LD    D,(IX+1)
A006  323EA0        40          LD    (STORE),A
A009  1E00          50          LD    E,0
A00B  ED533FA0      60          LD    (STORE+1),DE
A00F  2141A0        70          LD    HL,BUFFER
A012  F5            80          PUSH AF
A013  0E07          90          LD    C,#07
A015  CD0FB9       100          CALL #B90F                  ;select ROM
A018  F1           110          POP   AF
A019  C5           120          PUSH BC
A01A  4F           130          LD    C,A
A01B  CD66C6       140          CALL #C666                  ;read sector
A01E  C1           150          POP   BC
A01F  CD18B9       160          CALL #B918                  ;diselect ROM
A022  C9           170          RET
A023  3A3EA0       180          LD    A,(STORE)
A026  ED5B3FA0     190          LD    DE,(STORE+1)
A02A  2141A0       200          LD    HL,BUFFER
A02D  F5           210          PUSH AF
A02E  0E07         220          LD    C,#07
A030  CD0FB9       230          CALL #B90F
A033  F1           240          POP   AF
A034  C5           250          PUSH BC
A035  4F           260          LD    C,A
A036  CD4EC6       270          CALL #C64E                  ;write sector
A039  C1           280          POP   BC
A03A  CD18B9       290          CALL #B918                  ;diselect ROM
A03D  C9           300          RET
A03E               310 STORE:   DEFS 3
A041               320 BUFFER:  DEFS 512

Pass 2 errors: 00

Table used:     38  from    156
```

Μπορείτε να αναλύσετε σχετικά εύκολα το LISTING σε ASSEMBLY και τις ρουτίνες του λειτουργικού που χρησιμοποιεί, ώστε να φορτώνετε και να γράφετε SECTORS μέσα από προγράμματά σας σε κώδικα μηχανής.

RANDOM ACCESS ΑΠΟ ΤΟ AMSDOS

Η ρουτίνα που δίνουμε στη συνέχεια σας προσφέρει τη δυνατότητα να κάνετε Random Access μέσα από τη BASIC του Amstrad, μια δυνατότητα που δεν παρέχει η LOCOMOTIVE BASIC και το AMDOS. Γι'αυτό το λόγο προστίθενται τρεις νέες εντολές - το πώς γίνεται αυτό θα το εξηγήσουμε στο επόμενο κεφάλαιο - οι ¦OPEN, α%, ¦GET β%, @α$, PUT, c%, @α$. Για να εργοποιηθούν οι νέες εντολές πρέπει - αφού πληκτρολογηθεί ο κώδικας μηχανής - να δώσετε CALL 37000. Τώρα έχετε στη διάθεσή σας τις extra εντολές τις οποίες περιγράφουμε στη συνέχεια. Πρέπει να διευκρινήσουμε, ότι το RANDOM ACCESS που προσφέρει αυτή η ρουτίνα χρειάζεται έναν ολόκληρο δίσκο που πρέπει να είναι σε DATA FORMAT.

OPEN, α%: Δίνει στο WORKSPACE του κώδικα μηχανής το μήκος του κάθε φακέλου, που θα έχουμε στο αρχείο μας. Το α% πρέπει να είναι μικρότερο του 256.

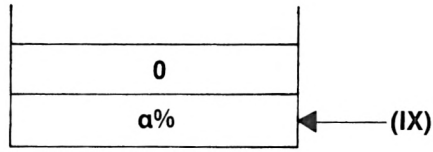
GET, β%, @α$: Διαβάζει από τη δισκέτα τον φάκελο με αριθμό b% και το βάζει στη μεταβλητή α$. Πρέπει πρώτα να έχει καθοριστεί γι'αυτήν θέση στη μνήμη του Amstrad, με το να δώσουμε α$=SPACE$ (α%) όπου α% το μήκος φακέλου που έχουμε.

PUT, c%, @α%: Γράφει στη δισκέτα τα περιεχόμενα της α$ σαν φάκελο με αριθμό c%. Το μήκος του string της μεταβλητής α$ πρέπει να είναι όσο και το μήκος φακέλου που έχουμε δώσει στην ¦OPEN.

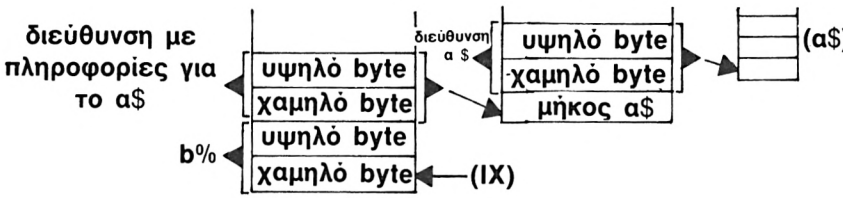
Η ρουτίνα υπολογίζει για όποιον αριθμό φακέλου θέλουμε, τη θέση που βρίσκεται ο φάκελος αυτός στη δισκέτα. Αυτό γίνεται με βάση τη μεταβλητή α% (μήκος φακέλου) που της περνάμε με την εντολή OPEN, α% την οποία πρέπει να δώσουμε πριν χρησιμοποιήσουμε τις υπόλοιπες εντολές.

Για χρήση της ρουτίνας από κώδικα μηχανής μπορούμε να καλέσουμε τις ρουτίνες OPEN, GET και PUT ξεχωριστά.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ OPEN: CALL OPEN. Ο IX πρέπει να δείχνει το παρακάτω block μεταβλητών. Η α% δίνει το μήκος του φακέλου.



ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ GET: GALL GET. Ο IX πρέπει να δείχνει το παρακάτω block μεταβλητών.



ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ PUT: CALL PUT. Ο IX πρέπει να δείχνει το ίδιο block μεταβλητών με την GET.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Όλοι οι καταχωρητές έχουν τυχαίο περιεχόμενο.

```
Pass 1 errors: 00

9088                10          org  37000
9088  219290        20          LD   HL,BUFFER
908B  019690        30          LD   BC,CTAB
908E  CDD1BC        40          CALL #BCD1          ;put new commands
9091  C9            50          RET
9092                60 BUFFER:  DEFS 4              ;4 bytes buffer
9096  A190          70 CTAB:    DEFW CNAM           ;names address
9098  C3AC90        80          JP   OPEN           ;relative routines
909B  C3CC90        90          JP   PUT
909E  C3DE90       100          JP   GET
90A1  4F5045       110 CNAM:    DEFM "OPE"          ;name of first command
90A4  CE           120          DEFB #CE            ;asc("N")+#80
90A5  5055         130          DEFM "PU"
90A7  D4           140          DEFB #D4
90A8  4745         150          DEFM "GE"
90AA  D4           160          DEFB #D4
90AB  00           170          DEFB 0              ;end of commands
90AC  DD5E00       180 OPEN:    LD   E,(IX+0)       ;rec. length
90AF  DD5601       190          LD   D,(IX+1)       ;in DE
90B2  0600         200          LD   B,0
90B4  D5           210          PUSH DE
90B5  210002       220          LD   HL,#200        ;sector length
90B8  B7           230          OR   A              ;clear carry
90B9  ED52         240 LOOP1:   SBC  HL,DE          ;calculate records
90BB  3803         250          JR   C,OUT1         ;per sector
90BD  04           260          INC  B
90BE  18F9         270          JR   LOOP1
90C0  21ACA3       280 OUT1:    LD   HL,41900       ;recs per sector
90C3  70           290          LD   (HL),B         ;in store
90C4  23           300          INC  HL
90C5  3600         310          LD   (HL),0
90C7  E1           320          POP  HL             ;record length
90C8  22AEA3       330          LD   (41902),HL     ;in store
90CB  C9           340          RET                 ;and finished
90CC  CD6F91       350 PUT:     CALL COMMON         ;(common to put and get
90CF  C5           360          PUSH BC
90D0  E5           370          PUSH HL
90D1  2AAEA3       380          LD   HL,(41902)     ;get record length
```

```
90D4  E5          390          PUSH HL
90D5  C1          400          POP  BC
90D6  E1          410          POP  HL                ;put variable
90D7  EDB0        420          LDIR                   ;into sector buffer
90D9  C1          430          POP  BC                ;sector/track no
90DA  CD4491      440          CALL WSECTO            ;put sector to disc
90DD  C9          450          RET
90DE  CD6F91      460 GET:     CALL COMMON
90E1  E5          470          PUSH HL
90E2  2AAEA3      480          LD   HL,(41902)        ;get record length
90E5  E5          490          PUSH HL
90E6  C1          500          POP  BC
90E7  E1          510          POP  HL
90E8  2ADEA3      520          LD   HL,(41950)
90EB  EB          530          EX   DE,HL             ;put string from sector
90EC  EDB0        540          LDIR                   ;into variable
90EE  C9          550          RET
90EF  2AACA3      560 FSECTO:  LD   HL,(41900)        ;get recs per sector
90F2  EB          570          EX   DE,HL
90F3  0600        580          LD   B,0
90F5  3EC1        590          LD   A,#C1
90F7  B7          600 LUP1:    OR   A                 ;clear carry
90F8  ED52        610          SBC  HL,DE             ;calculate sector & track
90FA  380E        620          JR   C,UT1             ;corresponding to
90FC  280C        630          JR   Z,UT1             ;record number
90FE  3C          640          INC  A
90FF  FECA        650          CP   #CA
9101  CC0691      660          CALL Z,LUP2
9104  18F1        670          JR   LUP1
9106  3EC1        680 LUP2:    LD   A,#C1
9108  04          690          INC  B
9109  C9          700          RET
910A  4F          710 UT1:     LD   C,A
910B  C9          720          RET
910C  2AB6A3      730 RSECTO:  LD   HL,(41910)        ;get track and sector
910F  B7          740          OR   A                 ;if already in buffer RET
9110  ED42        750          SBC  HL,BC
9112  C8          760          RET  Z
9113  C5          770          PUSH BC                ;preserve sector/treck
9114  0E07        780          LD   C,7
9116  CD0FB9      790          CALL #B90F             ;select ROM 7
9119  D1          800          POP  DE
911A  C5          810          PUSH BC
911B  F5          820          PUSH AF
911C  D5          830          PUSH DE
911D  CD00B9      840          CALL #B900             ;enable upper ROM
9120  3EC1        850          LD   A,#C1
9122  1E00        860          LD   E,0
9124  CD39C0      870          CALL #C039             ;select data format
9127  2110A4      880          LD   HL,42000
912A  C1          890          POP  BC
912B  C5          900          PUSH BC
912C  50          910          LD   D,B
912D  1E00        920          LD   E,0
912F  CD3CC0      930          CALL #C03C             ;read sector
9132  D1          940          POP  DE
9133  F1          950          POP  AF
9134  C1          960          POP  BC
9135  D5          970          PUSH DE
9136  CD18B9      980          CALL #B918             ;diselect ROM
9139  CD03B9      990          CALL #B903             ;disable upper ROM
913C  C1         1000          POP  BC
913D  21B6A3     1010          LD   HL,41910          ;save sector and track
9140  71         1020          LD   (HL),C            ;we just read
9141  23         1030          INC  HL
9142  70         1040          LD   (HL),B
9143  C9         1050          RET
9144  C5         1060 WSECTO:  PUSH BC
9145  0E07       1070          LD   C,7
9147  CD0FB9     1080          CALL #B90F             ;select ROM 7
914A  D1         1090          POP  DE
914B  C5         1100          PUSH BC
914C  F5         1110          PUSH AF
914D  D5         1120          PUSH DE
914E  CD00B9     1130          CALL #B900             ;enable upper ROM
9151  3EC1       1140          LD   A,#C1
9153  1E00       1150          LD   E,0
9155  CD39C0     1160          CALL #C039             ;select data format
9158  2110A4     1170          LD   HL,42000
```

```
915B  1E00      1180          LD   E,0
915D  C1        1190          POP  BC
915E  C5        1200          PUSH BC
915F  50        1210          LD   D,B
9160  CD3FC0    1220          CALL #C03F              ;write sector
9163  D1        1230          POP  DE
9164  F1        1240          POP  AF
9165  C1        1250          POP  BC
9166  D5        1260          PUSH DE
9167  CD18B9    1270          CALL #B918              ;diselect ROM
916A  CD03B9    1280          CALL #B903              ;disable upper ROM
916D  C1        1290          POP  BC
916E  C9        1300          RET
916F  CDB391    1310 COMMON:  CALL STRING
9172  DD6E02    1320          LD   L,(IX+2)
9175  DD6603    1330          LD   H,(IX+3)
9178  D5        1340 COM1:    PUSH DE
9179  E5        1350          PUSH HL
917A  E5        1360          PUSH HL
917B  D1        1370          POP  DE
917C  CDEF90    1380          CALL FSECTO
917F  CD0C91    1390          CALL RSECTO
9182  D1        1400          POP  DE
9183  E1        1410          POP  HL
9184  C5        1420          PUSH BC
9185  E5        1430          PUSH HL
9186  2AACA3    1440          LD   HL,(41900)
9189  EB        1450          EX   DE,HL
918A  B7        1460          OR   A
918B  ED52      1470 LOPC1:   SBC  HL,DE              ;calculate place in secto
                                                      r
918D  3802      1480          JR   C,OUTC1            ;that record is
918F  18FA      1490          JR   LOPC1
9191  19        1500 OUTC1:   ADD  HL,DE
9192  45        1510          LD   B,L
9193  2AAEA3    1520          LD   HL,(41902)
9196  EB        1530          EX   DE,HL
9197  2110A4    1540          LD   HL,42000
919A  78        1550          LD   A,B
919B  FE00      1560          CP   0
919D  280C      1570          JR   Z,LOPC4
919F  05        1580          DEC  B
91A0  97        1590          SUB  A
91A1  B8        1600          CP   B
91A2  2803      1610          JR   Z,LOPC3
91A4  19        1620 LOPC2:   ADD  HL,DE
91A5  10FD      1630          DJNZ LOPC2
91A7  EB        1640 LOPC3:   EX   DE,HL
91A8  E1        1650          POP  HL

91A9  C1        1660          POP  BC
91AA  C9        1670          RET
91AB  3AACA3    1680 LOPC4:   LD   A,(41900)
91AE  47        1690          LD   B,A
91AF  05        1700          DEC  B
91B0  C3A491    1710          JP   LOPC2
91B3  DDE5      1720 STRING:  PUSH IX

91B5  DD5E00    1730          LD   E,(IX+0)           ;calculate record's addr
ss
91B8  DD5601    1740          LD   D,(IX+1)
91BB  D5        1750          PUSH DE
91BC  DDE1      1760          POP  IX
91BE  DD23      1770          INC  IX
91C0  DD5E00    1780          LD   E,(IX+0)           ;and put it into DE
91C3  DD5601    1790          LD   D,(IX+1)
91C6  21DEA3    1800          LD   HL,41950           ;then store it
91C9  73        1810          LD   (HL),E
91CA  23        1820          INC  HL
91CB  72        1830          LD   (HL),D
91CC  2ADEA3    1840          LD   HL,(41950)
91CF  E5        1850          PUSH HL
91D0  D1        1860          POP  DE
91D1  DDE1      1870          POP  IX
91D3  C9        1880          RET

Pass 2 errors: 00

Table used:   270  from   478
```

```
10 J=37000
20 RESTORE
30 READ A$
40 IF A$="TELOS" THEN END
50 FOR I=1 TO LEN(A$) STEP 2
60 POKE J,VAL("&"+MID$(A$,I,2))
70 J=J+1
80 NEXT
90 GOTO 30
100 DATA 219290019690CDD1BCC900000000A190C3AC90C3CC90C3DE904F5045CE50
110 DATA 55D44745D400DD5E00DD56010600D5210002B7ED5238030418F921ACA370
120 DATA 233600E122AEA3C9CD6F91C5E52AAEA3E5C1E1EDB0C1CD4491C9CD6F91E5
130 DATA 2AAEA3E5C1E12ADEA3EBEDB0C92AACA3EB06003EC1B7ED52380E280C3CFE
140 DATA CACC069118F13EC104C94FC92AB6A3B7ED42C8C50E07CD0FB9D1C5F5D5CD
150 DATA 00B93EC11E00CD39C02110A4C1C5501E00CD3CC0D1F1C1D5CD18B9CD03B9
160 DATA C121B6A3712370C9C50E07CD0FB9D1C5F5D5CD00B93EC11E00CD39C02110
170 DATA A41E00C1C550CD3FC0D1F1C1D5CD18B9CD03B9C1C9CDB391DD6E02DD6603
180 DATA D5E5E5D1CDEF90CD0C91D1E1C5E52AACA3EBB7ED52380218FA19452AAEA3
190 DATA EB2110A478FE00280C0597B828031910FDEBE1C1C93AACA34705C3A491DD
200 DATA E5DD5E00DD5601D5DDE1DD23DD5E00DD560121DEA37323722ADEA3E5D1DD
210 DATA E1C9
220 DATA TELOS
```

Κάθε φορά που δίνουμε GET ή PUT, διαβάζεται ή γράφεται ένας ολόκληρος SECTOR στη δισκέτα. ΠΡΟΣΕΞΤΕ δεν υπάρχει έλεγχος για το μεγαλύτερο δυνατό αριθμό φακέλου που μπορεί να χωρέσει η δισκέτα. Πρέπει αυτό - ανάλογα με το μήκος του φακέλου που δίνετε - να το προσδιορίζετε εσείς. Αλλιώς το πρόγραμμα μπορεί να αρχίσει να ψάχνει για κάποιο ανύπαρκτο track με δυσάρεστα ίσως αποτελέσματα για το disk-drive.

Εδώ τελειώνει το κεφάλαιο για χειρισμό της δισκέτας. Μερικές ρουτίνες που χρησιμοποιούνται είναι καθ'εαυτό ρουτίνες της ROM 7 (ROM του Disk Drive) και όχι του JUMPBLOCK του λειτουργικού συστήματος και συνεπώς δεν περιέχονται στο κεφάλαιο 8. Μπορείτε να δείτε τι κάνουν αναλύοντας τον τρόπο που τις χρησιμοποιούμε στον EDITOR δισκέτας και το πρόγραμμα για RANDOM ACCESS.

7ο ΚΕΦΑΛΑΙΟ

ΧΡΗΣΙΜΕΣ ΡΟΥΤΙΝΕΣ ΚΩΔΙΚΑ ΜΗΧΑΝΗΣ ΓΙΑ ΠΡΟΓΡΑΜΜΑΤΑ ΣΕ BASIC

Σ'αυτό το κεφάλαιο θα παρουσιάσουμε μερικές ρουτίνες που έχουν σκοπό να βοηθήσουν κυρίως αυτούς που προγραμματίζουν σε BASIC. Το πρώτο πράγμα με το οποίο θα ασχοληθούμε είναι το πως γίνεται η επέκταση της LOCOMOTIVE BASIC με τη δημιουργία extra εντολών.

ΟΙ ΕΞΤΡΑ ΕΝΤΟΛΕΣ

Το λειτουργικό σύστημα του Amstrad μας παρέχει μεγάλες δυνατότητες για προσθήκη έξτρα εντολών. Συγκεκριμένα με τη ρουτίνα &BCD1 (48337) του λειτουργικού, μπορούμε να προσθέσουμε όσες νέες εντολές θέλουμε. Καλώντας τη ρουτίνα αυτή πρέπει ο BC να περιέχει τη διεύθυνση ενός πίνακα για τις extra εντολές - που θα περιγράψουμε παρακάτω και ο HL τη διεύθυνση ενός BUFFER με 4 bytes. Η ρουτίνα αυτή περιγράφεται και στο επόμενο κεφάλαιο μαζί με τις άλλες ρουτίνες του λειτουργικού.

Ας δούμε τι περιέχει ο πίνακας για τις νέες εντολές. Τα πρώτα δύο bytes είναι μια διεύθυνση της μνήμης όπου έχουμε βάλει τα ονόματα που θέλουμε να δώσουμε στις νέες εντολές. Στη συνέχεια ακολουθούν μια σειρά από JUMPS, όσα και οι νέες εντολές που βάζουμε, στις αντίστοιχες ρουτίνες κάθε εντολής, με την ίδια σειρά που δίνουμε τα ονόματά τους.

Τα ονόματα των νέων εντολών τα βάζουμε με τους αντίστοιχους κωδικούς ASCII των γραμμάτων τους. Στον ASCII του τελευταίου χαρακτήρα κάθε εντολής προσθέτουμε τον αριθμό &80, και μ'αυτόν τον τρόπο ειδοποιούμε τον υπολογιστή ότι τελείωσε η εντολή, οπότε ο επόμενος χαρακτήρας θεωρείται από το λειτουργικό σαν ο πρώτος της επόμενης. Μετά τον τελευταίο χαρακτήρα της τελευταίας εντολής αφήνουμε ένα μηδενικό byte, το οποίο ειδοποιεί το λειτουργικό ότι οι νέες εντολές τελείωσαν.

Αφού λοιπόν προσθέσουμε τις νέες εντολές όπως αναφέραμε προηγουμένως μπορούμε να τις χρησιμοποιήσουμε από BASIC βάζοντας μπροστά το σύμβολο ¦ που βρίσκεται πάνω από το @ στο πληκτρολόγιο. Με αυτόν τον τρόπο δίνουμε στον υπολογιστή να «καταλάβει» ότι

πρόκειται για μια πρόσθετη εντολή που δεν περιέχεται στην LOCOMOTIVE BASIC. Η χρήση κάθε εντολής ισοδυναμεί απόλυτα με ένα CALL στη διεύθυνση που βρίσκεται η ρουτίνα κώδικα μηχανής της εντολής. Μπορούμε συνεπώς να περάσουμε τις μεταβλητές μας με τον ίδιο ακριβώς τρόπο που το κάνουμε και με την εντολή CALL - που έχουμε εξηγήσει στο πρώτο κεφάλαιο αυτού του βιβλίου.

Ας υποθέσουμε δηλαδή ότι έχουμε στη θέση 42000 της μνήμης τη ρουτίνα μιας εντολής που έχουμε ονομάσει TICK και θέλουμε να περάσουμε σ' αυτήν την τιμή της μεταβλητής α%. Μπορούμε να δώσουμε είτε |TICK, α% είτε CALL 42000, α% με το ίδιο και στις δύο περιπτώσεις αποτέλεσμα.

Κάτι που πρέπει να προσέξετε όταν προσθέτετε νέες εντολές είναι να μην καλέσετε για τις ίδιες εντολές πάνω από μία φορά τη ρουτίνα στο &BCD1 γιατί υπάρχει μεγάλη πιθανότητα να κάνει CRASH το σύστημα. Ας δούμε όμως δύο ρουτίνες που δίνουν νέες εντολές ώστε να αποκτήσετε μια πιο πρακτική άποψη για το θέμα.

CLS ΚΑΙ FRAME

Η παρακάτω ρουτίνα προσθέτει δύο νέες εντολές τις ICLS και IFRAME. Καλώντας μια φορά τη διεύθυνση 40000 (CALL 40000) οι δύο νέες εντολές έχουν ενεργοποιηθεί.

Η CLS καθαρίζει την οθόνη και επανέρχεται στα αρχικά χρώματα που έχουμε όταν ανοίξουμε τον υπολογιστή.

Η FRAME σταματάει το processing της οθόνης μέχρι να ολοκληρωθεί το επόμενο FRAME. Αυτό είναι ιδιαίτερα χρήσιμο για ρουτίνες γραφικών και γίνεται με τη χρήση της ρουτίνας &BD19 του λειτουργικού.

```
Hisoft GENA3.1 Assembler. Page      1.

Pass 1 errors: 00

9C40                10          org   40000
9C40  01579C        20          LD    BC,COMTAB
9C43  21689C        30          LD    HL,BUFFER
9C46  CDD1BC        40          CALL  48337              ;set new commands
9C49  C9            50          RET
9C4A  CD19BD        60 FRAME:   CALL  48409              ;wait frame
9C4D  C9            70          RET
9C4E  CD02BC        80 CLS:     CALL  48130              ;reset screen pack
9C51  3E0C          90          LD    A,12
9C53  CD5ABB       100          CALL  47962              ;output code to VDU
9C56  C9           110          RET
9C57  5F9C         120 COMTAB: DEFW  NAMES              ;addr. of com. names
9C59  C34A9C       130          JP    FRAME             ;frame routine
9C5C  C34E9C       140          JP    CLS               ;cls routine
9C5F  4652414D     150 NAMES:  DEFM  "FRAM"             ;frame com. name
9C63  C5           160          DEFB  197                ;asc("E")+128
9C64  434C         170          DEFM  "CL"               ;cls com. name
9C66  D3           180          DEFB  211                ;asc("S")+128
9C67  00           190          DEFB  0                  ;terminate commands
9C68               200 BUFFER: DEFS  4                  ;buffer space (4bytes)
```

```
Pass 2 errors: 00

Table used:     73  from    157

10 J=40000
20 RESTORE
30 READ A$
40 IF A$="TELOS" THEN END
50 FOR I=1 TO LEN(A$) STEP 2
60 POKE J,VAL("&"+MID$(A$,I,2))
70 J=J+1
80 NEXT
90 GOTO 30
100 DATA 01579C21689CCDD1BCC9CD19BDC9CD02BC3E0CCD5ABBC95F9CC34A9CC34E
110 DATA 9C4652414DC5434CD300
120 DATA TELOS
```

PAUSE

Η εντολή που προσθέτει η παρακάτω ρουτίνα είναι η PAUSE, η οποία δημιουργεί μια καθυστέρηση για n/50 δευτερόλεπτα. Αυτό επιτυγχάνεται με το να περιμένει η φορές να «σχεδιαστεί» το επόμενο FRAME πράγμα που γίνεται κάθε πεντηκοστό του δευτερολέπτου. Η καθυστέρηση αυτή μπορεί να σταματήσει άμα πατήσουμε κάποιο πλήκτρο.

Η νέα αυτή εντολή ενεργοποιείται δίνοντας αρχικά CALL 40000 (μόνο μία φορά). Στη συνέχεια μπορούμε όποτε θελήσουμε να την εκτελέσουμε δίνοντας PAUSE, η όπου το η είναι όπως ορίστηκε προηγουμένως.

```
Hisoft GENA3.1 Assembler. Page      1.

Pass 1 errors: 00

9C40                10          org   40000
9C40  014A9C        20          LD    BC,COMTAB
9C43  21559C        30          LD    HL,BUFFER
9C46  CDD1BC        40          CALL  48337             ;set new command
9C49  C9            50          RET
9C4A  4F9C          60 COMTAB:  DEFW  NAMES             ;name table
9C4C  C3599C        70          JP    PAUSE
9C4F  50415553      80 NAMES:   DEFM  "PAUS"            ;command name
9C53  C5            90          DEFB  197
9C54  00           100          DEFB  0
9C55               110 BUFFER:  DEFS  4
9C59  FE01         120 PAUSE:   CP    1
9C5B  2018         130          JR    NZ,ERROR         ;one parameter
9C5D  DD4E00       140          LD    C,(IX)           ;pause duration
9C60  DD4601       150          LD    B,(IX+1)         ;in BC
9C63  C5           160 LOOP:    PUSH  BC               ;save register
9C64  CD19BD       170          CALL  48409            ;wait flyback
9C67  CD1BBB       180          CALL  47899            ;is key pressed ?
9C6A  3807         190          JR    C,FINISH         ;if so exit
9C6C  C1           200          POP   BC               ;decrease delay counter
9C6D  0B           210          DEC   BC               ;decrease delay counter
9C6E  78           220          LD    A,B              ;is BC 0 ?
9C6F  B1           230          OR    C                ;is BC 0 ?
9C70  20F1         240          JR    NZ,LOOP          ;if not return
9C72  C9           250          RET
9C73  C1           260 FINISH:  POP   BC               ;if key pressed
9C74  C9           270          RET                    ;then return
9C75  DD21979C     280 ERROR:   LD    IX,ERR           ;print error message
9C79  3E07         290          LD    A,7
9C7B  CD5ABB       300          CALL  47962            ;beep
9C7E  DD7E00       310 LOOP2:   LD    A,(IX+0)
```

```
9C81  FE00        320          CP   0
9C83  2807        330          JR   Z,END
9C85  CD5ABB      340          CALL 47962                ;write character
9C88  DD23        350          INC  IX
9C8A  18F2        360          JR   LOOP2                ;do it again
9C8C  3E0D        370 END:     LD   A,13
9C8E  CD5ABB      380          CALL 47962                ;line feed
9C91  3E0A        390          LD   A,10
9C93  CD5ABB      400          CALL 47962                ;one line down
9C96  C9          410          RET
9C97  4572726F    420 ERR:     DEFM "Error in parameters...."
9CAE  00          430          DEFB 0

Pass 2 errors: 00

Table used:   131  from   207
```

```
10 J=40000
20 RESTORE
30 READ A$
40 IF A$="TELOS" THEN END
50 FOR I=1 TO LEN(A$) STEP 2
60 POKE J,VAL("&"+MID$(A$,I,2))
70 J=J+1
80 NEXT
90 GOTO 30
100 DATA 014A9C21559CCDD1BCC94F9CC3599C50415553C50000000000FE012018DD
110 DATA 4E00DD4601C5CD19BDCD1BBB3807C10B78B120F1C9C1C9DD21979C3E07CD
120 DATA 5ABBDD7E00FE002807CD5ABBDD2318F23E0DCD5ABB3E0ACD5ABBC9457272
130 DATA 6F7220696E20706172616D65746572732E2E2E2E
140 DATA TELOS
```

Βλέπετε λοιπόν ότι πολύ εύκολα μπορείτε να επεκτείνετε την LOCOMOTIVE BASIC με οποιεσδήποτε εντολές θέλετε. Όμοια όλες τις ρουτίνες τις έχουμε παρουσιάσει μέχρι τώρα μπορείτε να τις δώσετε σαν πρόσθετες εντολές. Αρκεί βέβαια να προσέξετε να μην δώσετε σε δύο ρουτίνες το ίδιο όνομα εντολής.

Στη συνέχεια θα δώσουμε μερικές άλλες γενικής χρήσης ρουτίνες σε κώδικα μηχανής, γι' αυτούς που προγραμματίζουν σε BASIC.

ΚΑΘΟΡΙΣΜΟΣ ΩΡΑΣ

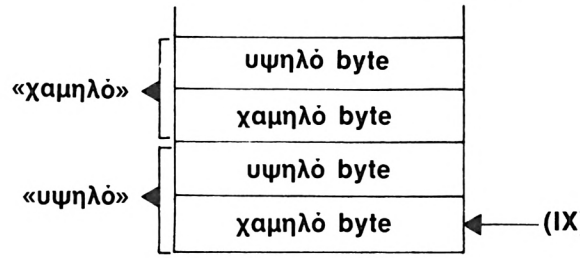
Υπάρχει μια μεταβλητή του λειτουργικού συστήματος του Amstrad που λέγεται TIME, και επιστρέφει ανά πάσα στιγμή το πόσος χρόνος έχει περάσει από τότε που πρωτοανοίξαμε τον υπολογιστή ή κάναμε RESET. Δεν περιέχει βέβαια μέσα τους χρόνους κατά τους οποίους έχουν καταργηθεί τα interrupts μέσα από τον κώδικα μηχανής (με την εντολή DI). Η τιμή αυτής της μεταβλητής δίνεται από 4 bytes (32 bit) και αυξάνεται κατά 1 κάθε τριακοσιοστό (1/300) του δευτερολέπτου. Αυτό είναι κάτι πολύ χρήσιμο για ρουτίνες που κάνουν χρονομέτρηση. Όμως δε μας δίνει τη δυνατότητα η BASIC του Amstrad να μηδενίσουμε την TIME ή να της δώσουμε μια συγκεκριμένη αρχική τιμή.

Η ρουτίνα που παρουσιάζουμε παρακάτω δίνει την αρχική τιμή που θέλουμε στη μεταβλητή αυτή.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: CALL 42000, χαμηλό, υψηλό όπου «χαμηλό» είναι τα δύο χαμηλά bytes (16 bit) της μετα-

βλητής και «υψηλό» τα δύο υψηλά bytes αντίστοιχα (σύνολο 4 bytes).

Για χρήση της ρουτίνας από κώδικα μηχανής ο A πρέπει να έχει την τιμή 2 και ο IX να δείχνει στο παρακάτω block μεταβλητών.



ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Όλοι οι καταχωρητές έχουν τυχαίο περιεχόμενο.

```
Hisoft GENA3.1 Assembler. Page      1.

Pass 1 errors: 00

A410                10          org  42000
A410  FE02          20          CP   2                    ;two parameters
A412  2032          30          JR   NZ,ERROR
A414  DD5E00        40          LD   E,(IX+0)             ;high bytes of TIME
A417  DD5601        50          LD   D,(IX+1)             ;in DE
A41A  DD6E02        60          LD   L,(IX+2)             ;low bytes of TIME
A41D  DD6603        70          LD   H,(IX+3)             ;in HL
A420  CD10BD        80          CALL 48400                ;set TIME
A423  C9            90          RET
A424  DDE5         100 SPRINT:  PUSH IX
A426  CD54BB       110          CALL 47956                ;enable text VDU
A429  3E0D         120          LD   A,13
A42B  CD5ABB       130          CALL 47962                ;line feed
A42E  3E0A         140          LD   A,10
A430  CD5ABB       150          CALL 47962                ;one line down
A433  DDE1         160          POP  IX
A435  DD7E00       170 LOOP:    LD   A,(IX+0)             ;get character
A438  FE00         180          CP   '0                   ;did string end
A43A  C8           190          RET  Z
A43B  DDE5         200          PUSH IX
A43D  CD5DBB       210          CALL 47965                ;print character
A440  DDE1         220          POP  IX
A442  DD23         230          INC  IX                   ;next character
A444  18EF         240          JR   LOOP
A446  DD2153A4     250 ERROR:   LD   IX,ERR               ;error messages
A44A  CD24A4       260          CALL SPRINT
A44D  3E07         270          LD   A,7
A44F  CD5ABB       280          CALL 47962                ;beep
A452  C9           290          RET
A453  4572726F     300 ERR:     DEFM "Error in parameters...."
A46A  00           310          DEFB 0

Pass 2 errors: 00

Table used:     59  from   172
```

```
10 J=42000
20 RESTORE
30 READ A$
40 IF A$="TELOS" THEN END
50 FOR I=1 TO LEN(A$) STEP 2
```

```
60 POKE J,VAL("&"+MID$(A$,I,2))
70 J=J+1
80 NEXT
90 GOTO 30
100 DATA FE022032DD5E00DD5601DD6E02DD6603CD10BDC9DDE5CD54BB3E0DCD5ABB
110 DATA 3E0ACD5ABBDDE1DD7E00FE00C8DDE5CD5DBBDDE1DD2318EFDD2153A4CD24
120 DATA A43E07CD5ABBC94572726F7220696E20706172616D65746572732E2E2E2E
130 DATA 00
140 DATA TELOS
```

Βλέπουμε εδώ ότι επειδή η TIMH είναι 32-bit μεταλητή, πρέπει να περάσουμε την αρχική τιμή της με 2 16-bit δικές μας ακέραιες μεταβλητές με πρώτη τη «χαμηλό». Αν για παράδειγμα θέλαμε να δώσουμε την αρχικήτ ιμή 1000 θα πρέπει να δώσουμε CALL 42000, 1000, 0. Η τελική τιμή της TIME προσδιορίζεται στον
TIME = «χαμηλό» + (65536 * «υψηλό»).

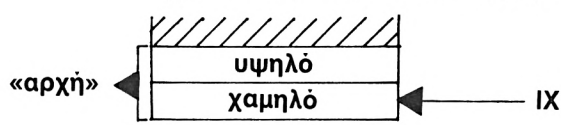
ΔΙΑΒΑΣΜΑ ΤΗΣ ROM

Όταν ανοίξουμε τον υπολογιστή η μνήμη που «βλέπουμε» κάνοντας POKE (ή PEEK) είναι RAM. Όμως όπως έχουμε ήδη πει στο 1ο κεφάλαιο η VIDEO RAM (16K) έχει «παράλληλα» 16K ROM καθώς επίσης άλλα 16K RAM βρίσκονται παράλληλα με τα πρώτα 16K της RAM. Όμως αργά ή γρήγορα θα θελήσετε να κάνετε LISTING στα bytes της ROM που βρίσκονται «κρυμμένα».

Αυτήν ακριβώς τη δουλειά κάνει η παρακάτω ρουτίνα. Δίνει LISTING των bytes της ROM στην οθόνη ανά 10, και είναι γραμμένη ώστε να δουλεύει καλύτερα στο MODE 2. Έχουν χρησιμοποιηθεί μερικές ρουτίνες του λειτουργικού που «ενεργοποιούν» την ROM απ' την οποία μπορούμε στη συνέχεια να διαβάσουμε.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: CALL 42000, αρχή όπου «αρχή» είναι η αρχική διεύθυνση απ' όπου θα πάρουμε λίστα των bytes.

Για χρήση της ρουτίνας από κώδικα μηχανής, πρέπει ο A να περιέχει 1 και ο IX να δείχνει το παρακάτω block παραμέτρων.



ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Όλοι οι καταχωρητές έχουν τυχαίο περιεχόμενο.

```
Hisoft GENA3.1 Assembler. Page      1.

Pass 1 errors: 00

A410                10          org  42000
```

```
A410  FE01        20          CP   1
A412  C0          30          RET  NZ
A413  0614        40          LD   B,20          ;print 20 bytes
A415  DD6E00      50          LD   L,(IX+0)      ;start address
A418  DD6601      60          LD   H,(IX+1)      ;in HL
A41B  C5          70 LOOP1:   PUSH BC
A41C  E5          80          PUSH HL
A41D  CD06B9      90          CALL 47366         ;enable lower ROM
A420  F5         100          PUSH AF            ;save ROM state
A421  CD00B9     110          CALL 47360         ;enable upper ROM
A424  7E         120          LD   A,(HL)        ;get byte
A425  32B2A4     130          LD   (STORE),A     ;then save it
A428  F1         140          POP  AF            ;get previews ROM state
A429  CD0CB9     150          CALL 47372         ;restore ROM state
A42C  E1         160          POP  HL
A42D  E5         170          PUSH HL
A42E  CD9BA4     180          CALL NUM1          ;print address in HEX
A431  DD21A4A4   190          LD   IX,SPACE      ;then spaces
A435  CD5AA4     200          CALL SPRINT
A438  3AB2A4     210          LD   A,(STORE)     ;get byte read
A43B  CD72A4     220          CALL NUM2          ;print byte in HEX
A43E  DD21A4A4   230          LD   IX,SPACE      ;print spaces
A442  CD5AA4     240          CALL SPRINT
A445  3AB2A4     250          LD   A,(STORE)     ;get byte again
A448  CD5DBB     260          CALL 47965         ;print its ASCII
A44B  3E0D       270          LD   A,13          ;line feed
A44D  CD5ABB     280          CALL 47962
A450  3E0A       290          LD   A,10          ;one line down
A452  CD5ABB     300          CALL 47962
A455  E1         310          POP  HL
A456  23         320          INC  HL            ;next byte
A457  C1         330          POP  BC
A458  10C1       340          DJNZ LOOP1         ;repeat 20 times
A45A  DDE5       350 SPRINT: PUSH IX
A45C  CD54BB     360          CALL 47956         ;enable text VDU
A45F  DDE1       370          POP  IX
A461  DD7E00     380 LOOP2:  LD   A,(IX+0)      ;get character
A464  FE00       390          CP   0             ;did it finish ?
A466  C8         400          RET  Z             ;if so exit
A467  DDE5       410          PUSH IX
A469  CD5DBB     420          CALL 47965         ;print character
A46C  DDE1       430          POP  IX
A46E  DD23       440          INC  IX
A470  18EF       450          JR   LOOP2
A472  0600       460 NUM2:   LD   B,0            ;the routines are
A474  4F         470          LD   C,A           ;already documented
A475  CB1F       480          RR   A
A477  CB1F       490          RR   A
A479  CB1F       500          RR   A
A47B  CB1F       510          RR   A
A47D  E60F       520 PRLO:   AND  15
A47F  FE0A       530          CP   10
A481  3008       540          JR   NC,ATOF
A483  C630       550          ADD  A,48
A485  C5         560          PUSH BC
A486  CD5ABB     570          CALL 47962         ;write code
A489  1806       580          JR   OUT
A48B  C637       590 ATOF:   ADD  A,55
A48D  C5         600          PUSH BC
A48E  CD5ABB     610          CALL 47962         ;output code to VDU
A491  C1         620 OUT:    POP  BC
A492  78         630          LD   A,B
A493  FE01       640          CP   1
A495  C8         650          RET  Z
A496  79         660          LD   A,C
A497  0601       670          LD   B,1
A499  18E2       680          JR   PRLO
A49B  7C         690 NUM1:   LD   A,H
A49C  CD72A4     700          CALL NUM2
A49F  7D         710          LD   A,L
A4A0  CD72A4     720          CALL NUM2
A4A3  C9         730          RET
A4A4  20202020   740 SPACE:  DEFM "
A4B1  00         750          DEFB 0
```

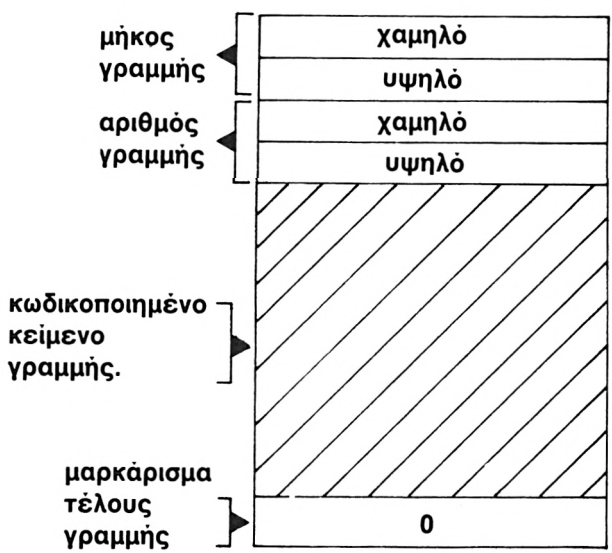
```
A4B2  00          760 STORE:  DEFB 0

Pass 2 errors: 00

Table used:   128  from   264
```

```
10 J=42000
20 RESTORE
30 READ A$
40 IF A$="TELOS" THEN END
50 FOR I=1 TO LEN(A$) STEP 2
60 POKE J,VAL("&"+MID$(A$,I,2))
70 J=J+1
80 NEXT
90 GOTO 30
100 DATA FE01C00614DD6E00DD6601C5E5CD06B9F5CD00B97E32B2A4F1CD0CB9E1E5
110 DATA CD9BA4DD21A4A4CD5AA43AB2A4CD72A4DD21A4A4CD5AA43AB2A4CD5DBB3E
120 DATA 0DCD5ABB3E0ACD5ABBE123C110C1DDE5CD54BBDDE1DD7E00FE00C8DDE5CD
130 DATA 5DBBDDE1DD2318EF06004FCB1FCB1FCB1FCB1FE60FFE0A3008C630C5CD5A
140 DATA BB1806C637C5CD5ABBC178FE01C8790601 18E27CCD72A47DCD72A4C92020
150 DATA 2020202020202020202020200000
160 DATA TELOS
```

Αξίζει να μελετήσετε λίγο αναλυτικώτερα τις ρουτίνες που χειρίζονται τις επιλογές ROM, ανατρέχοντας στο επόμενο κεφάλαιο.

ΔΟΜΗ ΤΩΝ ΠΡΟΓΡΑΜΜΑΤΩΝ ΣΕ BASIC

Κάτι το οποίο πρέπει να ενδιαφέρει πολύ όσους ασχολούνται τόσο με BASIC όσο και με κώδικα μηχανής είναι ο τρόπος που φυλάγονται στη μνήμη του Amstrad τα διάφορα προγράμματα σε BASIC. Μια γραμμή που έχουμε γράψει σε BASIC έχει την εξής δομή:



Το μήκος της γραμμής είναι ο συνολικός αριθμός bytes στη γραμμή, συμπεριλαμβανομένων των 2 bytes του μήκους γραμμής των 2 bytes του αριθμού γραμμής και του ενός byte που κυριολεκτικά μέσα μαρκάρει το τέλος της. Αυτό το τελευταίο ουσιαστικά χρησιμεύει για να ξεχωρίζει τη μία γραμμή από την άλλη. Άμα θέλουμε να εντοπίσουμε το τέλος ενός προγράμματος λοιπόν αρκεί να ψάξουμε για τη γραμμή που έχει μήκος 0 και αριθμό γραμμής επίσης 0.

Μπορείτε να έχετε πολύ ενδιαφέροντα αποτελέσματα κάνοντας POKE στο μήκος γραμμής ή στον αριθμό γραμμής κάποιας γραμμής του προγράμματός σας.

Το κομμάτι αυτό του προγράμματος φαίνεται πολλές φορές να χάνετε - παρόλο που βρίσκεται ακριβώς όπως ήταν στην ίδια θέση - και πρέπει να διορθώσετε τα POKEs που έχετε κάνει για να το ξαναδείτε. Είναι ένας ενδιαφέρων τρόπος να κάνετε το πρόγραμμά σας αόρατο για τα αδιάκριτα βλέμματα, χωρίς να του δημιουργήσετε κανένα πρόβλημα. Πάντως ο σκοπός μας εδώ δεν είναι να αναλύσουμε κάτι τέτοιο.

Μια άλλη χρήσιμη δυνατότητα που μας δίνεται σύμφωνα με αυτά που αναφέραμε παραπάνω είναι να βάλουμε κώδικα μηχανής κυριολεκτικά μέσα σε κάποιο πρόγραμμά μας σε BASIC. Κάτι όμως που πρέπει να προσεχθεί σε τέτοια περίπτωση είναι μήπως το πρόγραμμά μας κάνει χρήση ρουτίνων του λειτουργικού που απαιτούν PAGING της LOWER ROM γιατί τότε ίσως να έχουμε προβλήματα.

Ας εξετάσουμε όμως πως φυλάγονται οι διάφορες μεταβλητές μέσα στις γραμμές ενός προγράμματος σε BASIC. Έστω ότι έχουμε δώσει τη γραμμή 100 που είναι

```
100 a$ = "Amstrad"
```

Άμα δούμε τα bytes στην περιοχή μνήμης όπου φυλάγεται η γραμμή 100, θα δούμε ότι το Amstrad είναι γραμμένο με τους αντίστοιχους κωδικούς ASCII των χαρακτήρων. Ο τελευταίος χαρακτήρας μαρκάρεται προσθέτοντας στον ASCII του τον αριθμό 128 (&80). Είναι δηλαδή η ίδια τεχνική που περιγράψαμε και στην προσθήκη νέων εντολών. Δηλαδή το "Amstrad" θα φαίνεται στη μνήμη με τους εξής κωδικούς:

A 65
m 109
s 115
t 116
r 114
a 97
d 228 (128 + 100)

Οι εντολές πάλι φυλάγονται στη μνήμη με διάφορους κωδικούς αριθμούς, που ονομάζονται TOKENS.

ΕΥΡΕΣΗ ΧΑΡΑΚΤΗΡΩΝ

Η ρουτίνα που παρουσιάζουμε στη συνέχεια, βρίσκει τα ονόματα παραμέτρων ή κειμένου μέσα σ' ένα πρόγραμμα BASIC και τυπώνει τους αριθμούς γραμμής που τα βρήκε. Το μόνο πρόβλημα που ενδεχομένως να υπάρξει είναι όταν τυχαίνει το string που θέλουμε να βρούμε να είναι πολύ μικρό (π.χ. ενός χαρακτήρα) και να συμπίπτει με κάποιο TOKEN εντολής οπότε να μας τυπώσει λάθος αριθμό γραμμής.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: CALL 42000, @α$, κωδ. όπου α$ το string που ψάχνουμε να βρούμε, και «κωδ» ένας κωδικός που είναι 0 για όνομα μεταβλητής (όπου ο τελευταίος χαρακτήρας είναι επαυξημένος κατά &80) ή 1 για καθαρό κείμενο.

```
Hisoft GENA3.1 Assembler. Page      1.

Pass 1 errors: 00

A410                10          org   42000
A410  FE02          20          CP    2                ;two parameters
A412  205A          30          JR    NZ,ERROR
A414  DD6E02        40          LD    L,(IX+2)
A417  DD6603        50          LD    H,(IX+3)
A41A  7E            60          LD    A,(HL)
A41B  325DA5        70          LD    (LENGTH),A       ;length of string
A41E  23            80          INC   HL
A41F  4E            90          LD    C,(HL)
A420  23           100          INC   HL
A421  46           110          LD    B,(HL)
A422  ED435BA5     120          LD    (STRING),BC      ;string address
A426  DD7E00       130          LD    A,(IX+0)
A429  FE01         140          CP    1
A42B  2817         150          JR    Z,OK             ;text or variable
A42D  3A5DA5       160          LD    A,(LENGTH)       ;otherwise modify
A430  5F           170          LD    E,A              ;the last character
A431  1600         180          LD    D,0              ;of variable
A433  1B           190          DEC   DE               ;by adding 128to it
A434  2A5BA5       200          LD    HL,(STRING)
A437  19           210          ADD   HL,DE
A438  7E           220          LD    A,(HL)
A439  C680         230          ADD   A,128
A43B  77           240          LD    (HL),A           ;back in memory
A43C  3A5DA5       250          LD    A,(LENGTH)
A43F  FE01         260          CP    1                ;if var name only 1 char
A441  CCC8A4       270          CALL  Z,PPROB          ;message start prog
A444  DD217001     280 OK:      LD    IX,368           ;in Tape system
A448  DD5E00       290 LOOP1:   LD    E,(IX+0)
A44B  DD5601       300          LD    D,(IX+1)         ;line length in DE
A44E  DD6E02       310          LD    L,(IX+2)         ;line number in HL
A451  DD6603       320          LD    H,(IX+3)
A454  2259A5       330          LD    (LINE),HL
A457  7D           340          LD    A,L
A458  B4           350          OR    H
A459  2875         360          JR    Z,FINISH         ;have we done it?
A45B  CD1BBB       370          CALL  47899            ;is key pressed ?
A45E  3870         380          JR    C,FINISH         ;if yes exit
A460  CD70A4       390          CALL  SLINE
A463  DDE5         400          PUSH  IX
A465  E1           410          POP   HL
A466  1B           420          DEC   DE
A467  19           430          ADD   HL,DE            ;update HL to point
A468  23           440          INC   HL               ;the start of next line
A469  E5           450          PUSH  HL
A46A  DDE1         460          POP   IX               ;tranfer it to IX
A46C  18DA         470          JR    LOOP1            ;repeat
```

```
A46E  1850      480 ERROR:  JR   ERROR2
A470  DDE5      490 SLINE:  PUSH IX
A472  E1        500         POP  HL                ;start of line inHL
A473  DDE5      510         PUSH IX
A475  D5        520         PUSH DE
A476  C1        530         POP  BC                ;line length in BC
A477  0B        540         DEC  BC                ;line number-4
A478  0B        550         DEC  BC                ;to scan text only
A479  0B        560         DEC  BC
A47A  0B        570         DEC  BC
A47B  D5        580         PUSH DE
A47C  110400    590         LD   DE,4
A47F  19        600         ADD  HL,DE             ;point HL to start
A480  ED5B5BA5  610 LOOP2:  LD   DE,(STRING)       ;target in DE
A484  C5        620         PUSH BC                ;save line length
A485  1A        630         LD   A,(DE)
A486  BE        640         CP   (HL)
A487  280B      650         JR   Z,YES             ;two characters match
A489  C1        660 NOTOK:  POP  BC                ;decrement counter
A48A  0B        670         DEC  BC
A48B  23        680         INC  HL                ;point to next character
A48C  78        690         LD   A,B
A48D  B1        700         OR   C                 ;is BC=0
A48E  20F0      710         JR   NZ,LOOP2          ;if not repeat
A490  D1        720         POP  DE                ;restore registers
A491  DDE1      730         POP  IX
A493  C9        740         RET
A494  3A5DA5    750 YES:    LD   A,(LENGTH)
A497  47        760         LD   B,A
A498  1A        770 LOOP3:  LD   A,(DE)            ;rest of target
A499  BE        780         CP   (HL)              ;does it match ?
A49A  20ED      790         JR   NZ,NOTOK          ;if not get back
A49C  23        800         INC  HL
A49D  13        810         INC  DE
A49E  10F8      820         DJNZ LOOP3
A4A0  CDA5A4    830         CALL FOUND             ;found it
A4A3  18E4      840         JR   NOTOK
A4A5  3E0A      850 FOUND:  LD   A,10
A4A7  CD5ABB    860         CALL 47962             ;code to VDU
A4AA  3E0D      870         LD   A,13
A4AC  CD5ABB    880         CALL 47962             ;line feed
A4AF  DDE5      890         PUSH IX
A4B1  E5        900         PUSH HL
A4B2  D5        910         PUSH DE
A4B3  C5        920         PUSH BC
A4B4  2A59A5    930         LD   HL,(LINE)
A4B7  CDF8A4    940         CALL PDECHL
A4BA  C1        950         POP  BC
A4BB  D1        960         POP  DE
A4BC  E1        970         POP  HL
A4BD  DDE1      980         POP  IX
A4BF  C9        990         RET
A4C0  DD2146A5  1000 ERROR2: LD   IX,PARA
A4C4  CDD1A4    1010         CALL SPRINT
A4C7  C9        1020         RET
A4C8  DD2126A5  1030 PPROB:  LD   IX,MESS
A4CC  CDD1A4    1040         CALL SPRINT
A4CF  C9        1050         RET
A4D0  C9        1060 FINISH: RET
A4D1  DDE5      1070 SPRINT: PUSH IX
A4D3  CD54BB    1080         CALL,47956            ;enable text VDU
A4D6  3E0A      1090         LD   A,10             ;one line down
A4D8  CD5ABB    1100         CALL 47962
A4DB  3E0D      1110         LD   A,13             ;line feed
A4DD  CD5ABB    1120         CALL 47962
A4E0  3E07      1130         LD   A,7
A4E2  CD5ABB    1140         CALL 47962            ;beep
A4E5  DDE1      1150         POP  IX
A4E7  DD7E00    1160 LOOP4:  LD   A,(IX+0)         ;get character
A4EA  FE00      1170         CP   0                ;is it end ?
A4EC  C8        1180         RET  Z                ;if so exit
A4ED  DDE5      1190         PUSH IX
A4EF  CD5DBB    1200         CALL 47965            ;print character
A4F2  DDE1      1210         POP  IX
```

```
A4F4  DD23      1220          INC  IX                    ;next character
A4F6  18EF      1230          JR   LOOP4                 ;repeat
A4F8  111027    1240 PDECHL:  LD   DE,10000
A4FB  CD13A5    1250          CALL PDECH
A4FE  11E803    1260          LD   DE,1000
A501  CD13A5    1270          CALL PDECH
A504  116400    1280          LD   DE,100
A507  CD13A5    1290          CALL PDECH
A50A  110A00    1300          LD   DE,10
A50D  CD13A5    1310          CALL PDECH
A510  110100    1320          LD   DE,1
A513  AF        1330 PDECH:   XOR  A
A514  37        1340 LOOP5:   SCF
A515  3F        1350          CCF
A516  ED52      1360          SBC  HL,DE
A518  3803      1370          JR   C,PDOUT
A51A  3C        1380          INC  A
A51B  18F7      1390          JR   LOOP5
A51D  19        1400 PDOUT:   ADD  HL,DE
A51E  C630      1410          ADD  A,48
A520  E5        1420          PUSH HL
A521  CD5ABB    1430          CALL 47962
A524  E1        1440          POP  HL
A525  C9        1450          RET
A526  596F7520  1460 MESS:    DEFM "You may get some odd results !!"
A545  00        1470          DEFB 0
A546  50617261  1480 PARA:    DEFM "Parameter Error !!"
A558  00        1490          DEFB 0
A559  0000      1500 LINE:    DEFW 0000
A55B  0000      1510 STRING:  DEFW 0000
A55D  00        1520 LENGTH:  DEFB 0

Pass 2 errors: 00

Table used:    287  from    428
```

```
10 J=42000
20 RESTORE
30 READ A$
40 IF A$="TELOS" THEN END
50 FOR I=1 TO LEN(A$) STEP 2
60 POKE J,VAL("&"+MID$(A$,I,2))
70 J=J+1
80 NEXT
90 GOTO 30
100 DATA FE02205ADD6E02DD66037E325DA5234E2346ED435BA5DD7E00FE0128173A
110 DATA 5DA55F16001B2A5BA5197EC680773A5DA5FE01CCC8A4DD217001DD5E00DD
120 DATA 5601DD6E02DD66032259A57DB42875CD1BBB3870CD70A4DDE5E11B1923E5
130 DATA DDE119DA1850DDE5E1DDE5D5C10B0B0B0BD5110400019ED5B5BA5C51ABE28
140 DATA 0BC10B2378B120F0D1DDE1C93A5DA5471ABE20ED231310F8CDA5A418E43E
150 DATA 0ACD5ABB3E0DCD5ABBDDE5E5D5C52A59A5CDF8A4C1D1E1DDE1C9DD2146A5
160 DATA CDD1A4C9DD2126A5CDD1A4C9C9DDE5CD54BB3E0ACD5ABB3E0DCD5ABB3E07
170 DATA CD5ABBDDE1DD7E00FE00CBDDE5CD5DBBDDE1DD2318EF111027CD13A511E8
180 DATA 03CD13A5116400CD13A5110A00CD13A5110100AF373FED5238033C18F719
190 DATA C630E5CD5ABBE1C9596F75206D617920676574207367F6D65206F64642072
200 DATA 6573756C747320212100506172616D65746572204572726F72202121000
210 DATA 000000000
220 DATA TELOS
```

Αν για παράδειγμα θέλουμε να βρούμε τη μεταβλητή με όνομα «NTINA» δίνουμε:

α$ = «NTINA»: CALL 42000, @α%, 0

Μη βάζετε IDENTIFIERS (%, $) μέσα στο όνομα μιας μεταβλητής. Να δίνετε το όνομα μόνο ανεξάρτητα πώς την έχετε ορίσει.

Αν πάλι θέλετε να βρείτε κάποιο κείμενο (π.χ. τη φράση «ΚΕΙΜ» που ίσως βρίσκεται σε REM ή PRINT του προγράμματος) δώστε:

α$ = «ΚΕΙΜ»: CALL 42000, @α$, 1

Πρέπει να ξέρετε όμως ότι εδώ μπορεί να βρει η ρουτίνα το «ΚΕΙΜ»

είτε σαν τμήμα του ονόματος μιας μεταβλητής είτε μιας σειράς κειμένου.

ΜΗΚΟΣ ΠΡΟΓΡΑΜΜΑΤΟΣ BASIC

Η επόμενη ρουτίνα τυπώνει το μήκος σε BYTES ενός προγράμματος BASIC.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: CALL 42000

```
Hisoft GENA3.1 Assembler. Page     1.

Pass 1 errors: 00

A410                10          org   42000
A410  DD217001      20 OK:      LD    IX,368          ;start program
A414  010000        30          LD    BC,0
A417  DD5E00        40 LOOP1:   LD    E,(IX+0)
A41A  DD5601        50          LD    D,(IX+1)        ;line length in DE
A41D  DD6E02        60          LD    L,(IX+2)
A420  DD6603        70          LD    H,(IX+3)        ;line number
A423  7D            80          LD    A,L
A424  B4            90          OR    H               ;is line number 0 ?
A425  2812         100          JR    Z,FINISH        ;if so finished
A427  E5           110          PUSH  HL
A428  C5           120          PUSH  BC
A429  E1           130          POP   HL              ;BC into HL
A42A  19           140          ADD   HL,DE           ;add line length to DE
A42B  E5           150          PUSH  HL
A42C  C1           160          POP   BC              ;BC is updated
A42D  E1           170          POP   HL
A42E  DDE5         180          PUSH  IX
A430  E1           190          POP   HL
A431  1B           200          DEC   DE
A432  19           210          ADD   HL,DE           ;HL to point start of
A433  23           220          INC   HL              ;next line
A434  E5           230          PUSH  HL              ;then transfer it
A435  DDE1         240          POP   IX              ;to IX
A437  18DE         250          JR    LOOP1           ;repeat again
A439  C5           260 FINISH: PUSH  BC
A43A  E1           270          POP   HL              ;BC into HL
A43B  CD3FA4       280          CALL  PDECHL          ;print it
A43E  C9           290          RET
A43F  111027       300 PDECHL: LD    DE,10000
A442  CD5AA4       310          CALL  PDECH
A445  11E803       320          LD    DE,1000
A448  CD5AA4       330          CALL  PDECH
A44B  116400       340          LD    DE,100
A44E  CD5AA4       350          CALL  PDECH
A451  110A00       360          LD    DE,10
A454  CD5AA4       370          CALL  PDECH
A457  110100       380          LD    DE,1
A45A  AF           390 PDECH:  XOR   A
A45B  37           400 LOOP2:  SCF
A45C  3F           410          CCF
A45D  ED52         420          SBC   HL,DE
A45F  3803         430          JR    C,OUT
A461  3C           440          INC   A
A462  18F7         450          JR    LOOP2
A464  19           460 OUT:    ADD   HL,DE
A465  C630         470          ADD   A,48
A467  E5           480          PUSH  HL
A468  CD5ABB       490          CALL  47962
A46B  E1           500          POP   HL
A46C  C9           510          RET


Pass 2 errors: 00

Table used:     94  from    199
```

```
10 J=42000
20 RESTORE
30 READ A$
40 IF A$="TELOS" THEN END
50 FOR I=1 TO LEN(A$) STEP 2
60 POKE J,VAL("&"+MID$(A$,I,2))
70 J=J+1
80 NEXT
90 GOTO 30
100 DATA DD21700101000ODD5E00DD5601DD6E02DD66037DB42812E5C5E119E5C1E1
110 DATA DDE5E11B1923E5DDE118DEC5E1CD3FA4C9111027CD5AA411E803CD5AA411
120 DATA 6400CD5AA4110A00CD5AA41101000AF373FED5238033C18F719C630E5CD5A
130 DATA BBE1C9
140 DATA TELOS
```

Εδώ λοιπόν τελειώνουν οι έτοιμες ρουτίνες που παρουσιάσαμε μέσα απ΄ αυτό το βιβλίο. Ελπίζουμε να τις βρήκατε πολύ χρήσιμες, και να σας έδωσαν μια ώθηση να φτιάξετε τα δικά σας προγράμματα σε κώδικα μηχανής. Επειδή όμως χρειαζόσαστε τις ρουτίνες του λειτουργικού, αφιερώνουμε το επόμενο και τελευταίο κεφάλαιο του βιβλίου για να τις παραθέσουμε.

8ο ΚΕΦΑΛΑΙΟ

ΚΛΗΣΕΙΣ ΤΟΥ ΛΕΙΤΟΥΡΓΙΚΟΥ ΣΥΣΤΗΜΑΤΟΣ

Οι κλήσεις του λειτουργικού συστήματος που περιγράφονται εδώ, έχουμε προσπαθήσει να είναι όσο πιο σαφείς γίνεται. Περισσότερες πληροφορίες πάντως για κάθε ρουτίνα μπορείτε να βρείτε στο FIRMWARE MANUAL του Amstrad.

Ο τρόπος που παρουσιάζονται είναι ο εξής: Κάθε CALL έχει ένα όνομα και μία διεύθυνση (τη διεύθυνση που πρέπει να καλέσουμε). Στη συνέχεια εξηγείται το τι κάνει η ρουτίνα αυτή, οι συνθήκες εισόδου και οι συνθήκες εξόδου. Αν υπάρχει κάποια σημαντική λεπτομέρεια, την παραθέτουμε στο τέλος σαν σημείωση.

ΟΙ ΚΛΗΣΕΙΣ

INITIALIZING ΠΛΗΚΤΡΟΛΟΓΙΟΥ #BB00

Ξεκινάει τον χειριστή πληκτρολογίου με τις αρχικές τιμές. Όλες οι μεταβλητές και τα buffers που σχετίζονται με το πληκτρολόγιο χάνονται.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: AF, BC, DE και HL έχουν τυχαίο περιεχόμενο.

RESET ΠΛΗΚΤΡΟΛΟΓΙΟΥ #BB03

Καθαρίζει τα buffers και οτιδήποτε μετατροπές έχουν σχέση με τον χειριστή πληκτρολογίου του υπολογιστή.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: AF, BC, DE και HL έχουν τυχαίο περιεχόμενο.

ΠΕΡΙΜΕΝΕ ΧΑΡΑΚΤΗΡΑ ΑΠΟ ΠΛΗΚΤΡ. #BB06

Περιμένει έναν χαρακτήρα από τον buffer του πληκτρολογίου ή το string επέκτασης.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο κωδικός του χαρακτήρα βίσκεται στον A και το CARRY FLAG είναι αληθές (σε λογικό 1). Τα άλλα flags είναι επηρεασμένα.

ΔΙΑΒΑΣΕ ΧΑΡΑΚΤΗΡΑ ΑΠΟ ΠΛΗΚΤΡ. #BB09

Επιστρέφει έναν χαρακτήρα από τον buffer του πληκτρολογίου ή το string επέκτασης, αν υπάρχει κάποιος διαθέσιμος. Δεν περιμένει.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Αν υπάρχει διαθέσιμος χαρακτήρας, ο κωδικός του βρίσκεται στον A και το CARRY είναι σε λογικό 1 (αληθές). Αν δεν υπάρχει, το CARRY είναι σε λογικά 0 (ψευδές) και ο A έχει τυχαίο περιεχόμενο. Τα άλλα flags είναι επηρεασμένα.

ΕΠΙΣΤΡΟΦΗ ΧΑΡΑΚΤΗΡΑ ΑΠΟ ΠΛΗΚΤΡ. #BB0C

Ένας χαρακτήρας επανέρχεται στον buffer του πληκτρολογίου, ώστε να μπορεί να ξαναδιαβαστεί αργότερα. Αυτό επιτρέπει να ελέγξουμε κάποιο χαρακτήρα χωρίς να τον χάσουμε. Μόνο ένας χαρακτήρας μπορεί να επανέρθει κάθε φορά. Για να βάλουμε κάποιον άλλον πρέπει πρώτα αυτός που έχουμε βάλει να «διαβαστεί».

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο A έχει τον κωδικό του χαρακτήρα που θέλουμε να επανέλθει.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Όλοι οι καταχωρητές έχουν διατηρηθεί αλλά τα flags είναι όλα επηρεασμένα.

ΒΑΛΕ ΕΠΕΚΤΑΣΗ ΠΛΗΚΤΡ. #BB0F

Βάζει ένα string επέκτασης σε σχέση με κάποιον κωδικό TOKEN.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο B περιέχει το TOKEN επέκτασης που θα χρησιμοποιηθεί, ο C περιέχει το μήκος του αντίστοιχου STRING και ο HL «δείχνει» το string.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Αν όλα πήγαν καλά το CARRY είναι σε λογικό 1 (αληθές), αλλιώς είναι ψευδές. Οι A, BC, DE, HL έχουν τυχαίο περιεχόμενο και τα άλλα flags είναι επηρεασμένα.

ΣΗΜΕΙΩΣΗ: Το string πρέπει να βρίσκεται στη RAM.

ΔΙΑΒΑΣΕ ΕΠΕΚΤΑΣΗ ΠΛΗΚΤΡ. #BB12

Διαβάζει έναν χαρακτήρα από κάποιο string επέκτασης. Ο πρώτος χαρακτήρας είναι ο αριθμός 0.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο A περιέχει το TOKEN επέκτασης και ο L αριθμό του χαρακτήρα.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Αν όλα πήγαν καλά το CARRY είναι σε λογικό 1 (αληθές) και ο κωδικός του χαρακτήρα στο A, αλλιώς το CARRY είναι ψευδές και ο A έχει τυχαίο περιεχόμενο. Ο DE και τα άλλα flags (εκτός του CARRY) είναι επηρεασμένα.

BUFFER ΕΠΕΚΤΑΣΗΣ ΠΛΗΚΤΡ. #BB15

Καθορίζει έναν buffer για τα string επέκτασης, και βάζει τις αυτό-

ματες τιμές των string αυτών.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο DE έχει τη διεύθυνση του buffer και ο HL το μήκος του.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Το CARRY σε λογικό 1 (αληθές) εκτός αν ο buffer είναι πολύ μικρός. Οι A, BC, DE και HL έχουν τυχαίο περιεχόμενο και τα άλλα flags (εκτός το CARRY) είναι επηρεασμένα.

ΠΕΡΙΜΕΝΕ ΠΛΗΚΤΡΟ **#BB18**

Περίμενε το πάτημα ενός πλήκτρου.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Αν όλα πήγαν καλά το CARRY είναι σε λογικό 1 (αληθές), και ο A κρατάει το χαρακτήρα ή το TOKEN επέκτασης. Τα TOKENS δεν μπορούν να επεκταθούν).

ΔΙΑΒΑΣΕ ΠΛΗΚΤΡΟ **#BB1B**

Διαβάζει ένα πλήκτρο από τον buffer πληκτρολογίου, αν κάποιο είναι διαθέσιμο, χωρίς να περιμένει.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Αν υπήρχε διαθέσιμο πλήκτρο (πατημένο) τότε το CARRY είναι σε λογικό 1 (αληθές) και ο A έχει τον αντίστοιχο κωδικό. Αλλιώς το CARRY είναι ψευδές και ο A έχει τυχαίο περιεχόμενο. Τα άλλα flags είναι επηρεασμένα.

ΕΛΕΓΞΕ ΠΛΗΚΤΡΟ **#BB1E**

Ελέγχει αν πατάμε κάποιο συγκεκριμένο πλήκτρο.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο A περιέχει τον αριθμό του πλήκτρου.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Αν το πλήκτρο που ελέγξαμε είναι πατημένο το ZERO flag είναι σε λογικό 0 (ψευδές). Ο C περιέχει την κατάσταση του SHIFT και του CONTROL. Ο A, ο HL έχουν τυχαίο περιεχόμενο και τα άλλα flags είναι αλλοιωμένα.

ΕΛΕΓΞΕ ΚΑΤΑΣΤΑΣΗ ΠΛΗΚΤΡΟΛ. **#BB21**

Ελέγχει αν είναι ενεργοποιημένα το CAPS LOCK ή το SHIFT LOCK.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο L περιέχει &FF αν υπάρχει SHIFT LOCK, αλλιώς είναι 0. Όμοια ο H περιέχει &FF αν υπάρχει CAPS LOCK, αλλιώς είναι 0. Ο AF έχει τυχαίο περιεχόμενο.

ΕΛΕΓΞΕ JOYSTICK **#BB24**

Ελέγχει την είσοδο του JOYSTICK.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο A και ο H περιέχουν την κατάσταση του Joystick 0. Ο L περιέχει την κατάσταση του Joystick 1. Τα FLAGS είναι όλα επηρεασμένα.

Τα bits των αντίστοιχων bytes των H και L που επιστρέφουν έχουν την εξής σημασία:
BIT 0 - Άνω
BIT 1 - Κάτω
BIT 2 - Αριστερά
BIT 3 - Δεξιά
BIT 4 - FIRE 2
BIT 5 - FIRE 1
BIT 6 - Έξτρα κουμπί
BIT 7 - 0

ΒΑΛΕ ΑΝΤΙΣΤΟΙΧΙΑ **#BB27**

Βάζει έναν κωδικό ή ένα token να αντιστοιχεί σε κάποιο συγκεκριμένο πλήκτρο.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο A περιέχει τον αριθμό του πλήκτρου, και ο B τον κωδικό ή το token.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF και HL έχουν τυχαίο περιεχόμενο.

Τα ακόλουθα tokens έχουν ειδική χρήση:

80 - 9F Tokens επέκτασης.
E0 - FC Edit του κωδικούς του συστήματος
FD CAPS LOCK
FE SHIFT LOCK
FF Καμία σημασία. (Αγνοείται).

ΠΑΡΕ ΑΝΤΙΣΤΟΙΧΙΑ #BB2A

Επιστρέφει την αντιστοιχία ενός πλήκτρου χωρίς SHIFT και CONTROL.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο A περιέχει τον αριθμό πλήκτρου.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο A περιέχει τον κωδικό ή το token που αντιστοιχεί στο πλήκτρο. Ο HL έχει τυχαίο περιεχόμενο και τα flags έχουν επηρεαστεί.

ΒΑΛΕ SHIFT #BB2D

Βάζει την αντιστοιχία ενός πλήκτρου (δηλ. κάποιο κωδικό ή token) όταν το πατάμε με SHIFT.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο A περιέχει τον αριθμό του πλήκτρου και ο B την αντιστοιχία του.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF και HL έχουν τυχαίο περιεχόμενο.

ΣΗΜΕΙΩΣΗ: Για τα tokens ισχύουν τα ίδια που αναφέραμε στη ρουτίνα «ΒΑΛΕ ΑΝΤΙΣΤΟΙΧΙΑ».

ΠΑΡΕ SHIFT #BB30

Επιστρέφει την αντιστοιχία κάποιου πλήκτρου όταν το πατάμε με SHIFT.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο A περιέχει τον αριθμό πλήκτρου.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο A περιέχει τον αντίστοιχο κωδικό ή token. Ο HL έχει τυχαίο περιεχόμενο και τα flags έχουν επηρεαστεί.

ΒΑΛΕ CONTROL #BB33

Βάζει την αντιστοιχία ενός πλήκτρου (δηλ. κάποιο κωδικό ή token) όταν το πατάμε ταυτόχρονα με CONTROL.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο A περιέχει τον αριθμό πλήκτρου, και ο B τον αντίστοιχο κωδικό ή token.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF και HL έχουν τυχαίο περιεχόμενο.

ΣΗΜΕΙΩΣΗ: Για τα tokens ισχύουν τα ίδια που έχουμε αναφέρει στη ρουτίνα «ΒΑΛΕ ΑΝΤΙΣΤΟΙΧΙΑ».

ΠΑΡΕ CONTROL #BB36

Επιστρέφει την αντιστοιχία ενός πλήκτρου (δηλ. κάποιο κωδικό ή token) όταν το πατάμε ταυτόχρονα με το CONTROL.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο A περιέχει τον αριθμό πλήκτρου.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο A περιέχει τον αντίστοιχο κωδικό ή token. Ο HL έχει τυχαίο περιεχόμενο και τα flags έχουν επηρεαστεί.

ΒΑΛΕ ΕΠΑΝΑΛΗΨΗ #BB39

Καθορίζει αν ένα πλήκτρο θα επαναλαμβάνεται όταν το κρατούμε πατημένο, ή όχι.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο A περιέχει τον αριθμό πλήκτρου, και ο B περιέχει &FF αν επιτρέπουμε την επανάληψη ή 0 αν δεν την επιτρέπουμε.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF, BC και HL έχουν τυχαίο περιεχόμενο.

ΠΑΡΕ ΕΠΑΝΑΛΗΨΗ #BB3C

Ελέγχει αν σε κάποιο πλήκτρο έχει επιτραπεί η επανάληψη ή όχι.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο A περιέχει τον αριθμό πλήκτρου.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Το ZERO flag είναι σε λογικό 0 (ψευδές) αν επιτρέπεται η επανάληψη, αλλιώς είναι σε λογικό 1.

Το CARRY flag είναι σε λογικό 0 και οι A και HL έχουν τυχαίο περιεχόμενο.

ΒΑΛΕ ΚΑΘΥΣΤΕΡΗΣΗ #BB3F

Βάζει την αρχική καθυστέρηση μέχρι να αρχίσει η αυτόματη επανάληψη ενός πλήκτρου καθώς και την περίοδο της αυτόματης επανάληψης.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο H περιέχει την αρχική καθυστέρηση και ο L την περίοδο επανάληψης. Οι τιμές δίνονται σε πεντηκοστά του δευτερολέπτου. Η τιμή 0 αντιστοιχεί στην 256.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο AF έχει τυχαίο περιεχόμενο.

ΠΑΡΕ ΚΑΘΥΣΤΕΡΗΣΗ **#BB42**

Επιστρέφει την τιμή της αρχικής καθυστέρησης και της περιοδου επανάληψης.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο Η περιέχει την αρχική καθυστέρηση και ο L περιέχει την περίοδο επανάληψης. Ο AF έχει τυχαίο περιεχόμενο.

ΕΝΕΡΓΟΠΟΙΗΣΕ BREAK **#BB45**

Ενεργοποιεί τα Breaks.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο DE περιέχει τη διεύθυνση μιας ρουτίνας που καλείται μόλις πατηθεί BREAK, και ο C περιέχει τη διεύθυνση επιλογής ROM για τη ρουτίνα.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF, BC, DE και HL έχουν τυχαίο περιεχόμενο.

ΑΠΟΕΝΕΡΓΟΠΟΙΗΣΕ BREAK **#BB48**

Αποενεργοποιεί τα break. Είναι η κατάσταση που ισχύει όταν ανοίξουμε τον υπολογιστή.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF και HL έχουν τυχαίο περιεχόμενο.

ΡΟΥΤΙΝΑ BREAK **#BB4B**

Δημιουργεί ένα break αν τα breaks είναι ενεργοποιημένα. Στη συνέχεια τα αποενεργοποιεί για να μη συμβούν τυχόν πολλαπλά breaks.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF και HL έχουν τυχαίο περιεχόμενο.

ΕΚΚΙΝΗΣΗ ΟΘΟΝΗΣ ΚΕΙΜΕΝΟΥ **#BB4E**

Γίνεται πλήρης επαναφορά της οθόνης κειμένου στην αρχική κατάσταση, μαζί με όλες τις μεταβλητές και τις έμμεσες αντιστοιχίες.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF, BC, DE και HL έχουν τυχαίο περιεχόμενο.

RESET ΚΕΙΜΕΝΟΥ **#BB51**

Κάνει RESET τις έμμεσες αντιστοιχίες οθόνης κειμένου και του «πίνακα ελέγχου» (control table).

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF, BC, DE και HL έχουν τυχαίο περιεχόμενο.

ΕΝΕΡΓΟΠΟΙΗΣΗ ΟΘΟΝΗΣ ΚΕΙΜΕΝΟΥ **#BB54**

Επιτρέπει χαρακτήρες να τυπωθούν στην οθόνη.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο AF έχει τυχαίο περιεχόμενο.

ΑΠΟΕΝΕΡΓΟΠΟΙΗΣΗ ΟΘΟΝΗΣ ΚΕΙΜΕΝΟΥ **#BB57**

Εμποδίζει την εκτύπωση χαρακτήρων στην οθόνη. Ο δρομέας επίσης αποενεργοποιείται.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο AF έχει τυχαίο περιεχόμενο.

ΕΞΟΔΟΣ ΚΕΙΜΕΝΟΥ **#BB5A**

Εξάγει έναν κωδικό στην οθόνη κειμένου. Αν είναι χαρακτήρας ελέγχου (&00÷&1F) εκτελείται η αντίστοιχη λειτουργία.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο A περιέχει τον κωδικό που θέλουμε να σταλεί.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Όλα τα flags και τα περιεχόμενα των καταχωρητών διατηρούνται.

ΤΥΠΩΜΑ ΧΑΡΑΚΤΗΡΑ **#BB5D**

Τυπώνει έναν χαρακτήρα στη θέση που βρίσκεται ο δρομέας. Οι χαρακτήρες ελέγχου (&00÷&1F) τυπώνονται - δεν εκτελούνται.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο A περιέχει τον κωδικό του χαρακτήρα.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF, BC, DE και HL έχουν τυχαίο περιεχόμενο.

ΔΙΑΒΑΣΜΑ ΧΑΡΑΚΤΗΡΑ **#BB60**

Διαβάζει έναν χαρακτήρα από την οθόνη στη θέση που βρίσκεται ο δρομέας.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Αν βρέθηκε κάποιος αναγνωρίσιμος χαρακτήρας το Carry flag είναι σε λογικό 1 (αληθές) και ο A περιέχει τον κωδικό του χαρακτήρα. Αλλιώς το Carry flag είναι σε λογικό 0 και ο A περιέχει 0.

ΟΘΟΝΗ ΓΡΑΦΙΚΩΝ **#BB63**

Ενεργοποιεί ή απενεργοποιεί την οθόνη γραφικών για το τύπωμα χαρακτήρων.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο A περιέχει 0 για να την ενεργοποιήσει, ή κάποιον άλλο αριθμό για να την αποενεργοποιήσει.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο AF έχει τυχαίο περιεχόμενο.

ΔΗΜΙΟΥΡΓΙΑ ΠΑΡΑΘΥΡΟΥ **#BB66**

Δημιουργεί κάποιο παράθυρο στο τρέχον stream (κανάλι).

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο D και ο H περιέχουν τις ακραίες στήλες του παραθύρου, όπου η μικρότερη θεωρείται σαν το αριστερό άκρο. Ο E και ο L περιέχουν τις ακραίες σειρές και η μικρότερη θεωρείται σαν το άνω άκρο.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF, BC, DE και HL έχουν τυχαίο περιεχόμενο.

ΣΗΜΕΙΩΣΗ: Οι τιμές συντεταγμένων ξεκινούν απ' το 0 και όχι το 1 όπως στη BASIC.

ΕΛΕΓΞΕ ΠΑΡΑΘΥΡΟ **#BB69**

Ελέγχει τα όρια του παράθυρου στο τρέχον stream (κανάλι).

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο H περιέχει την αριστερή ακραία στήλη, ο D την δεξιά, ο L την άνω ακραία σειρά και ο E την κάτω. Το Carry flag είναι σε λογικό 0 (ψευδές) αν το παράθυρο καταλαμβάνει ολόκληρη την οθόνη.

ΣΗΜΕΙΩΣΗ: Οι τιμές συντεταγμένων ξεκινούν απ' το 0 και όχι το 1 όπως στην BASIC.

ΚΑΘΑΡΙΣΕ ΠΑΡΑΘΥΡΟ **#BB6C**

Βάζει το παράθυρο κειμένου του τρέχοντος stream στο μελάνι του paper (γι' αυτό το stream) και τοποθετεί το δρομέα στην άνω αριστερή γωνία.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF, BC, DE και HL έχουν τυχαίο περιεχόμενο.

ΒΑΛΕ ΣΤΗΛΗ **#BB6F**

Μεταφέρει το δρομέα για το stream που βρισκόμαστε σε μία συγκεκριμένη στήλη.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο A περιέχει τον αριθμό στήλης. Σαν στήλη 1 θεωρείται η αριστερή ακραία στήλη του παράθυρου.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF και HL έχουν τυχαίο περιεχόμενο.

ΒΑΛΕ ΣΕΙΡΑ **#BB72**

Μεταφέρει το δρομέα για το stream που βρισκόμαστε σε μια συγκεκριμένη σειρά.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο A περιέχει τον αριθμό σειράς. Σαν σειρά 1 θεωρείται η ανώτατη σειρά του παράθυρου.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF και HL έχουν τυχαίο περιεχόμενο.

ΒΑΛΕ ΔΡΟΜΕΑ **#BB75**

Μεταφέρει το δρομέα για το stream που βρισκόμαστε σε ορισμένη σειρά και στήλη.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο Η περιέχει τη στήλη και ο L τη σειρά (ξεκινώντας με 1 για την ακραία στήλη του παραθύρου που βρισκόμαστε).

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF και HL έχουν τυχαίο περιεχόμενο.

ΔΙΑΒΑΣΕ ΔΡΟΜΕΑ **#BB78**

Ελέγχει τη θέση του δρομέα στο τρέχον παράθυρο καθώς και το πόσες φορές έχει κάνει scrolling το παράθυρο που βρισκόμαστε.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο Η περιέχει τη στήλη και ο L τη σειρά. Ο Α περιέχει τον αριθμό των scrolling ο οποίος μειώνεται όταν το παράθυρο κάνει scroll προς τα πάνω, και αυξάνεται όταν το παράθυρο κάνει scroll προς τα κάτω. Η θέση του δρομέα που μας επιστρέφεται δεν είναι απαραίτητα έγκυρη.

ΕΝΕΡΓΟΠΟΙΗΣΗ ΔΡΟΜΕΑ **#BB7B**

Ενεργοποιεί το δρομέα για το τρέχον stream.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο AF έχει τυχαίο περιεχόμενο.

ΑΠΟΕΝΕΡΓΟΠΟΙΗΣΗ ΔΡΟΜΕΑ **#BB7E**

Αποενεργοποιεί το δρομέα για το τρέχον stream.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο AF έχει τυχαίο περιεχόμενο.

ΕΜΦΑΝΙΣΕ ΔΡΟΜΕΑ **#BB81**

Εμφανίζει το δρομέα στο stream που βρισκόμαστε.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Όλα τα flags και οι καταχωρητές διατηρούνται.

ΣΒΗΣΙΜΟ ΔΡΟΜΕΑ **#BB84**

Δεν επιτρέπει την εμφάνιση δρομέα για το stream που βρισκόμαστε.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Όλα τα flags και τα περιεχόμενα των καταχωρητών διατηρούνται.

ΕΛΕΓΧΟΣ ΔΡΟΜΕΑ **#BB87**

Ελέγχει αν ο δρομέας βρίσκεται μέσα στα όρια του τρέχοντος παραθύρου.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο H περιέχει τη στήλη και ο L τη σειρά της θέσης του δρομέα.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Αν βρίσκεται μέσα στο παράθυρο το carry flag είναι σε λογικό 1 (αληθές) και ο B έχει τυχαίο περιεχόμενο. Αν το παράθυρο έκανε scrolling πάνω, το Carry flag είναι σε λογικό 0 και ο B περιέχει &FF. Αν το παράθυρο έχει κάνει scrolling προς τα κάτω το carry είναι σε λογικό 0 και ο B περιέχει την τιμή 0. Ο A έχει τυχαίο περιεχόμενο και τα άλλα flags έχουν επηρεαστεί.

ΤΟΠΟΘΕΤΗΣΕ ΔΡΟΜΕΑ **#BB8A**

Εμφανίζει το δρομέα στο stream που χρησιμοποιούμε. Αυτό επιτρέπει την ύπαρξη πολλαπλών δρομέων.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο AF έχει τυχαίο περιεχόμενο.

ΑΦΑΙΡΕΣΕ ΔΡΟΜΕΑ **#BB8D**

Αφαιρεί έναν πολλαπλό δρομέα από την οθόνη.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο AF έχει τυχαίο περιεχόμενο.

ΟΡΙΣΕ PEN **#BB90**

Ορίζει ένα μελάνι σαν PEN στο stream που βρισκόμαστε.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο A περιέχει τον αριθμό INK.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF και HL έχουν τυχαίο περιεχόμενο.

ΔΙΑΒΑΣΕ PEN **#BB93**

Ελέγχει το μελάνι του PEN για το stream που βρισκόμαστε.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο A περιέχει τον αριθμό μελανιού. Τα flags είναι επηρεασμένα.

ΟΡΙΣΕ PAPER **#BB96**

Ορίζει το μελάνι του PAPER για το stream (παράθυρο) που βρισκόμαστε.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο A περιέχει τον αριθμό μελανιού.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF και HL έχουν τυχαίο περιεχόμενο.

ΔΙΑΒΑΣΕ PAPER **#BB99**

Ελέγχει το μελάνι του PAPER για το stream (παράθυρο) που βρισκόμαστε.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο A περιέχει τον αντίστοιχο αριθμό INK. Τα flags είναι επηρεασμένα.

ΑΝΤΙΣΤΡΟΦΗ ΧΡΩΜΑΤΩΝ **#BB9C**

Εναλλάσσει το PAPER και το PEN (δηλ. τα αντίστοιχα μελάνια) για το stream (παράθυρο) που βρισκόμαστε.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF και HL έχουν τυχαίο περιεχόμενο.

ΒΑΛΕ OVER - PRINT **#BB9F**

Καθορίζει αν ένας χαρακτήρας τυπώνεται πάνω σ'αυτό που ήδη υπάρχει στην οθόνη χωρίς να το σβήνει, ή αν όταν τυπώνεται σβήνει ό,τι υπάρχει από κάτω.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο A περιέχει 0 για να τα σβήσει ή μη μηδενική τιμή για να γράφεται ο χαρακτήρας πάνω σε ό,τι ήδη υπάρχει.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF και HL έχουν τυχαίο περιεχόμενο.

ΣΗΜΕΙΩΣΗ: Η ρουτίνα αυτή δεν επηρεάζει την οθόνη γραφικών.

ΕΛΕΓΞΕ OVER - PRINT **#BBA2**

Ελέγχει αν ο χαρακτήρας τυπώνεται πάνω σε ό,τι ήδη υπάρχει χωρίς να το σβήνει, ή αν όταν τυπώνεται σβήνει ότι υπάρχει από κάτω.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο A περιέχει 0 αν ότι τυπώνεται σβήνει αυτό που υπάρχει από κάτω, αλλιώς περιέχει μη μηδενική τιμή. Οι DE και HL έχουν τυχαίο περιεχόμενο και τα flags έχουν επηρεαστεί.

ΠΑΡΕ ΠΙΝΑΚΑ ΧΑΡ. **#BBA5**

Επιστρέφει τη διεύθυνση του πίνακα του σχεδίου κάποιου χαρακτήρα.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο A περιέχει τον κωδικό του χαρακτήρα.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Αν αυτό βρίσκεται στη ROM, το carry flag είναι σε λογικό 0. Ο A έχει τυχαίο περιεχόμενο και τα flags έχουν επηρεαστεί.

ΒΑΛΕ ΠΙΝΑΚΑ ΧΑΡ. **#BBA8**

Τοποθετεί έναν πίνακα σχεδίου, κάποιου χαρακτήρα.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο A περιέχει τον κωδικό του χαρακτήρα και ο HL έχει τη διεύθυνση του πίνακα που θέλουμε να τοποθετήσουμε.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Αν ο χαρακτήρας είναι επανακαθοριζόμενος από τον χρήστη το carry flag είναι σε λογικό 1 (αληθές), αλλιώς είναι σε λογικό 0 (ψευδές). Οι A, BC, DE, HL έχουν τυχαίο περιεχόμενο και τα flags έχουν επηρεαστεί.

ΒΑΛΕ ΠΟΛΛΑΠΛΟΥΣ ΠΙΝΑΚΕΣ **#BBAB**

Τοποθετεί πολλαπλούς πίνακες σχεδίων χαρακτήρων (για μέχρι &FF (255) χαρακτήρες).

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο DE περιέχει τον κωδικό του πρώτου χαρακτήρα και ο HL την αρχική διεύθυνση των πινάκων.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Αν δεν υπήρχαν τέτοιοι πολλαπλοί πίνακες με επανακαθοριζόμενους χαρακτήρες το carry flag είναι σε λογικό 0 και οι A και HL έχουν τυχαίο περιεχόμενο. Αλλιώς το carry είναι σε λογικό 1 (αληθές), ο A περιέχει το πρώτο χαρακτήρα με το νέο σχέδιο και ο HL έχει τη διεύθυνση του παλιού table. Οι BC και DE έχουν τυχαίο περιεχόμενο και τα άλλα flags έχουν επηρεαστεί.

ΔΙΑΒΑΣΕ ΠΟΛΛΑΠΛΟΥΣ ΠΙΝΑΚΕΣ #BBAE

Ελέγχει τη διεύθυνση των επανακαθοριζόμενων χαρακτήρων και τον κωδικό του πρώτα χαρακτήρα στους πίνακες.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Αν υπάρχει τέτοιος πολλαπλός πίνακας το carry είναι σε λογικό 1, ο A περιέχει τον κωδικό του πρώτου χαρακτήρα, και ο HL την αρχική διεύθυνση του πίνακα. Αλλιώς το carry είναι σε λογικό 0 και οι A και HL έχουν τυχαίο περιεχόμενο. Τα άλλα flags έχουν επηρεαστεί.

ΠΑΡΕ ΠΙΝΑΚΑ ΧΑΡΑΚΤΗΡ. ΕΛΕΓΧΟΥ #BBB1

Επιστρέφει τη διεύθυνση του πίνακα με τους χαρακτήρες ελέγχου (&00÷&1F). Υπάρχουν τρία bytes για κάθε τέτοιο χαρακτήρα. Το πρώτο δίνει τον αριθμό των παραμέτρων που απαιτούνται για την εκτέλεση της ρουτίνας που αντιστοιχεί στο χαρακτήρα ελέγχου. Τα επόμενα δύο δίνουν τη διεύθυνση αυτής της ρουτίνας.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο HL περιέχει τη διεύθυνση του πίνακα των χαρακτήρων ελέγχου.

ΕΠΙΛΟΓΗ STREAM #BBB4

Κάνει την επιλογή stream (καναλιού) π.χ. στην περίπτωση που θέλουμε να ανοίξουμε κάποιο παράθυρο στην οθόνη.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο A περιέχει τον αριθμό του stream που θέλουμε.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο HL έχει τυχαίο περιεχόμενο και τα flags έχουν επηρεαστεί.

ΕΝΑΛΛΑΓΗ STREAMS #BBB7

Εναλλάσσει δύο streams (κανάλια).

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο B και ο C περιέχουν τους αριθμούς των δύο streams.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF, BC, DE και HL έχουν τυχαίο περιεχόμενο.

ΤΟΠΟΘΕΤΗΣΗ ΟΘΟΝΗΣ ΓΡΑΦΙΚΩΝ #BBBA

Τοποθετεί την οθόνη γραφικών με όλες τις αρχικές τιμές.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF, BC, DE και HL έχουν τυχαίο περιεχόμενο.

RESET ΟΘΟΝΗΣ ΓΡΑΦΙΚΩΝ #BBBD

Δίνει τις αρχικές τιμές σε όλες τις ρουτίνες που σχετίζονται με την οθόνη γραφικών (π.χ. PLOT, TEST κλπ.).

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF, BC, DE και HL έχουν τυχαίο περιεχόμενο.

ΑΠΟΛΥΤΟ MOVE #BBC0

Μετακινεί το δρομέα γραφικών σε μια απόλυτη θέση (δηλ. με απόλυτες συντεταγμένες).

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο DE περιέχει την απαιτούμενη οριζόντια συντεταγμένη (X) και ο HL την κατακόρυφη συντεταγμένη (Y).

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF, BC, DE και HL έχουν τυχαίο περιεχόμενο.

ΣΧΕΤΙΚΟ MOVE #BBC3

Μετακινεί το δρομέα γραφικών σε μια θέση σχετική με την παρούσα θέση του.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο DE περιέχει τη προσημασμένη σχετική οριζόντια συντεταγμένη (X) και ο HL την προσημασμένη σχετική κατακόρυφη συντεγμένη (Y).

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF, BC, DE και HL έχουν τυχαίο περιεχόμενο.

ΕΛΕΓΧΟΣ ΔΡΟΜΕΑ ΓΡΑΦΙΚΩΝ **#BBC6**

Ελέγχει τη θέση του δρομέα γραφικών.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο DE περιέχει την οριζόντια συντεταγμένη X και ο HL την κατακόρυφη συντεταγμένη Y. Ο AF έχει τυχαίο περιεχόμενο.

ΤΟΠΟΘΕΤΗΣΕ ORIGIN **#BBC9**

Τοποθετεί την αρχή των συντεταγμένων του χρήστη, και μετακινεί την τρέχουσα θέση εκεί.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο DE περιέχει την οριζόντια συντεταγμένη X και ο HL την κατακόρυφη συντεταγμένη Y του origin.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF, BC, DE και HL έχουν τυχαίο περιεχόμενο.

ΔΙΑΒΑΣΕ ORIGIN **#BBCC**

Ελέγχει τη θέση του origin.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο DE περιέχει τη σταθερή οριζόντια συντεταγμένη X και ο HL τη σταθερή κατακόρυφη συντεταγμένη Y του origin.

ΤΟΠΟΘΕΤΗΣΕ ΠΛΑΤΟΣ **#BBCF**

Τοποθετεί το αριστερό και το δεξιό άκρο του παράθυρου γραφικών.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο DE και ο HL περιέχουν τις οριζόντιες συντεταγμένες των άκρων. Η μικρότερη τιμή καθορίζει το αριστερό άκρο.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF, BC, DE και HL έχουν τυχαίο περιεχόμενο.

ΤΟΠΟΘΕΤΗΣΕ ΥΨΟΣ **#BBD2**

Τοποθετεί το άνω και το κάτω άκρο του παράθυρου γραφικών.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο DE και ο HL περιέχουν τις κατακόρυφες συντεταγμένες για τα άκρα. Η μικρότερη τιμή καθορίζει το άνω άκρο.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF, BC, DE και HL έχουν τυχαίο περιεχόμενο.

ΔΙΑΒΑΣΕ ΠΛΑΤΟΣ **#BBD5**

Ελέγχει τα δεξιά και αριστερά άκρα του παράθυρου γραφικών.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο DE περιέχει την οριζόντια συντεταγμένη του αριστερού άκρου και ο HL αντίστοιχα του δεξιού. Ο AF έχει τυχαίο περιεχόμενο.

ΔΙΑΒΑΣΕ ΥΨΟΣ **#BBD8**

Ελέγχει το άνω και το κάτω άκρο του παράθυρου γραφικών.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο DE περιέχει την κατακόρυφη συντεταγμένη του άνω άκρου και ο HL αντίστοιχα του κάτω άκρου. Ο AF έχει τυχαίο περιεχόμενο.

ΚΑΘΑΡΙΣΕ ΓΡΑΦΙΚΑ **#BBDB**

Καθαρίζει το παράθυρο γραφικών και το επαναφέρει στο χρώμα του PAPER που έχει.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF, BC, DE και HL έχουν τυχαίο περιεχόμενο.

ΣΗΜΕΙΩΣΗ: Ο δρομέας γραφικών πηγαίνει στη θέση του origin.

ΚΑΘΟΡΙΣΕ PEN ΓΡΑΦΙΚΩΝ **#BBDE**

Καθορίζει το pen για τα γραφικά.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο A περιέχει τον αριθμό του αντίστοιχου INK.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο AF έχει τυχαίο περιεχόμενο.

ΔΙΑΒΑΣΕ PEN ΓΡΑΦΙΚΩΝ **#BBE1**

Ελέγχει το μελάνι που έχει το pen γραφικών.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο A περιέχει τον αριθμό του INK. Τα flags είναι επηρεασμένα.

ΚΑΘΟΡΙΣΕ PAPER ΓΡΑΦΙΚΩΝ **#BBE4**

Καθορίζει το μελάνι για το paper των γραφικών.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο A περιέχει τον αριθμό του INK.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο AF έχει τυχαίο περιεχόμενο.

ΔΙΑΒΑΣΕ PAPER ΓΡΑΦΙΚΩΝ **#BBE7**

Ελέγχει το μελάνι του paper για τα γραφικά.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο A περιέχει τον αριθμό του INK. Τα flags είναι επηρεασμένα.

ΑΠΟΛΥΤΟ PLOT **#BBEA**

Εμφανίζει ένα pixel σε μία θέση που καθορίζεται με απόλυτες συντεταγμένες.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο DE περιέχει την οριζόντια συντεταγμένη X και ο HL την κατακόρυφη συντεταγμένη Y.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF, BC, DE και HL έχουν τυχαίο περιεχόμενο.

ΣΧΕΤΙΚΟ PLOT **#BBED**

Εμφανίζει ένα pixel στην οθόνη, σε μια θέση σχετική με την τρέχουσα θέση του δρομέα γραφικών.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο DE περιέχει την προσημασμένη σχετική οριζόντια συντεταγμένη X και ο HL την προσημασμένη σχετική κατακόρυφη συντεταγμένη Y.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF, BC, DE και HL έχουν τυχαίο περιεχόμενο.

ΑΠΟΛΥΤΟ TEST **#BBF0**

Ελέγχει το μελάνι ενός pixel σε μία απόλυτη θέση.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο DE περιέχει την οριζόντια συντεταγμένη X και ο HL την κατακόρυφη συντεταγμένη Y.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο A περιέχει τον αριθμό του ink του pixel. Οι BC, DE και HL έχουν τυχαίο περιεχόμενο και τα flags έχουν επηρεαστεί.

ΣΧΕΤΙΚΟ TEST **#BBF3**

Ελέγχει το ink ενός pixel που βρίσκεται σε μία θέση σχετική με την τρέχουσα θέση του δρομέα γραφικών.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο DE περιέχει την προσημασμένη σχετική οριζόντια συντεταγμένη X και ο HL την προσημασμένη κατακόρυφη συντεταγμένη Y.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο A περιέχει τον αριθμό του ink. Οι BC, DE και HL έχουν τυχαίο περιεχόμενο και τα flags έχουν επηρεαστεί.

ΑΠΟΛΥΤΟ DRAW **#BBF6**

Σχεδιάζει μια ευθεία γραμμή από την τρέχουσα θέση του δρομέα γραφικών μέχρι την απόλυτη θέση που καθορίζουμε.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο DE περιέχει την οριζόντια συντεταγμένη X και ο HL την κατακόρυφη συντεταγμένη Y.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF, BC, DE και HL έχουν τυχαίο περιεχόμενο.

ΣΧΕΤΙΚΟ DRAW **#BBF9**

Σχεδιάζει μια ευθεία γραμμή από την τρέχουσα θέση του δρομέα γραφικών μέχρι μια σχετική προς αυτήν θέση.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο DE περιέχει την προσημασμένη οριζόντια σχετική συντεταγμένη X και ο HL την προσημασμένη σχετική κατακόρυφη συντεταγμένη Y.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF, BC, DE και HL έχουν τυχαίο περιεχόμενο.

ΤΥΠΩΣΕ ΧΑΡΑΚΤΗΡΑ ΣΕ ΟΘΟΝΗ ΓΡΑΦΙΚΩΝ **#BBFC**

Τυπώνει έναν χαρακτήρα στην οθόνη, στην τρέχουσα θέση γραφικών.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο A περιέχει τον κωδικό του χαρακτήρα.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF, BC, DE και HL έχουν τυχαίο περιεχόμενο.

ΤΟΠΟΘΕΤΗΣΗ ΑΡΧΙΚΩΝ ΣΥΝΘΗΚΩΝ ΟΘΟΝΗΣ **#BBFF**

Οτι έχει σχέση με την οθόνη επανέρχεται στις αρχικές τιμές. Αυτό ισχύει ακόμα και για το mode και τα inks.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF, BC, DE και HL έχουν τυχαίο περιεχόμενο.

RESET ΟΘΟΝΗΣ **#BC02**

Κάνει RESET στα χρώματα και τα έμμεσα δεδομένα της οθόνης καθώς και το ρυθμό εναλλαγής χρωμάτων (FLASH) και το mode εκτύπωσης.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF, BC, DE και HL έχουν τυχαίο περιεχόμενο.

ΒΑΛΕ OFFSET ΟΘΟΝΗΣ **#BC05**

Βάζει το offset της οθόνης.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο HL περιέχει το απαιτούμενο offset.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF και HL έχουν τυχαίο περιεχόμενο.

ΤΟΠΟΘΕΤΗΣΕ ΒΑΣΗ ΟΘΟΝΗΣ **#BC08**

Τοποθετεί τη διεύθυνση της βάσης της οθόνης δηλ. την αρχική διεύθυνση περιοχή της μνήμης που θεωρείται ως VIDEO RAM. Μπορεί να πάρει τιμές #C000 ή #4000.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο A περιέχει το υψηλό byte της διεύθυνσης βάσης.

ΕΛΕΓΞΕ ΔΙΕΥΘΥΝΣΗ ΟΘΟΝΗΣ **#BC0B**

Ελέγχει τη διεύθυνση της βάσης της οθόνης και του offset.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο A περιέχει το υψηλό byte της διεύθυνσης βάσης της οθόνης και ο HL το offset. Τα flags έχουν επηρεαστεί.

ΒΑΛΕ MODE **#BC0E**

Βάζει ένα mode οθόνης (0, 1 ή 2).

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο A περιέχει το mode που θέλουμε.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF, BC, DE και HL έχουν τυχαίο περιεχόμενο.

ΕΛΕΓΞΕ MODE **#BC11**

Ελέγχει το τρέχον mode οθόνης.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Αν υπάρχει mode 0 το carry flag βρίσκεται σε λογικό 1, το zero flag σε λογικό 0, και ο A περιέχει 0. Αν υπάρχει mode 1 το carry flag βρίσκεται σε λογικό 0, το zero flag σε λογικό 1 και ο A περιέχει 1. Αν υπάρχει mode 2 το carry και το zero βρίσκονται σε λογικό 0 και ο A περιέχει 2. Τα άλλα flags έχουν επηρεαστεί.

ΚΑΘΑΡΙΣΕ ΟΘΟΝΗ **#BC14**

Η οθόνη καθαρίζεται και παίρνει το χρώμα του INK 0.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF, BC, DE και HL έχουν τυχαίο περιεχόμενο.

ΜΕΓΕΘΟΣ ΟΘΟΝΗΣ ΧΑΡΑΚΤΗΡΩΝ #BC17

Ελέγχει το μέγεθος της οθόνης σε χαρακτήρες.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο Β περιέχει την τελευταία στήλη της οθόνης και ο C την τελευταία σειρά της οθόνης. Ο AF έχει τυχαίο περιεχόμενο.

ΥΠΟΛΟΓΙΣΕ ΘΕΣΗ ΧΑΡΑΚΤΗΡΑ #BC1A

Υπολογίζει τη θέση της οθόνης για την άνω αριστερά γωνία του χαρακτήρα.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο Η περιέχει τη στήλη και ο L τη σειρά του χαρακτήρα. (Ξεκινώντας απ' την τιμή 0, 1, ...).

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο HL περιέχει τη διεύθυνση της οθόνης και ο Β τα bytes ανά χαρακτήρα. Ο AF έχει τυχαίο περιεχόμενο.

ΥΠΟΛΟΓΙΣΕ ΘΕΣΗ PIXEL #BC1D

Υπολογίζει τη θέση της οθόνης για κάποιο pixel.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο DE περιέχει την οριζόντια συντεταγμένη και ο HL την κατακόρυφη του pixel που θέλουμε.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο HL περιέχει την αντίστοιχη διεύθυνση της οθόνης, ο Β περιέχει τα pixels ανά byte πλην 1 και ο C το mask του pixel. Οι AF και DE έχουν τυχαίο περιεχόμενο.

ΥΠΟΛΟΓΙΣΕ ΕΠΟΜΕΝΟ BYTE #BC20

Υπολογίζει τη διεύθυνση οθόνης για ένα byte δεξιά μιας δεδομένης διεύθυνσης.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο HL περιέχει μια διεύθυνση οθόνης.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο HL περιέχει την ενημερωμένη νέα διεύθυν-

ση οθόνης. Ο AF έχει τυχαίο περιεχόμενο.

ΥΠΟΛΟΓΙΣΕ ΠΡΟΗΓΟΥΜΕΝΟ BYTE #BC23

Υπολογίζει τη διεύθυνση οθόνης ενός byte αριστερά της διεύθυνσης οθόνης που του δίνουμε.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο HL περιέχει μια διεύθυνση οθόνης.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο HL περιέχει τη νέα (ενημερωμένη) διεύθυνση οθόνης. Ο AF έχει τυχαίο περιεχόμενο.

ΥΠΟΛΟΓΙΣΕ ΕΠΟΜΕΝΗ ΓΡΑΜΜΗ #BC26

Υπολογίζει τη διεύθυνση οθόνης ένα byte κάτω από τη διεύθυνση που του δίνουμε (δηλ. μια γραμμή κάτω).

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο HL περιέχει μια διεύθυνση οθόνης.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο HL περιέχει τη νέα (ενημερωμένη) διεύθυνση οθόνης. Ο AF έχει τυχαίο περιεχόμενο.

ΥΠΟΛΟΓΙΣΕ ΠΡΟΗΓΟΥΜΕΝΗ ΓΡΑΜΜΗ #BC29

Υπολογίζει τη διεύθυνση οθόνης ένα byte επάνω από τη διεύθυνση που του δίνουμε (δηλ. μια γραμμή πάνω).

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο HL περιέχει μια διεύθυνση οθόνης.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο HL περιέχει τη νέα (ενημερωμένη) διεύθυνση οθόνης. Ο AF έχει τυχαίο περιεχόμενο.

ΚΩΔΙΚΟΠΟΙΗΣΕ INK #BC2C

Μετατρέπει κάποιο ink που του δίνουμε στην κωδικοποιημένη μορφή που βάζει όλα τα pixels σε ένα byte, σε αυτό το ink.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο A περιέχει τον αριθμό τον ink.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο A περιέχει το κωδικοποιημένο ink. Τα flags είναι επηρεασμένα.

ΑΠΟΚΩΔΙΚΟΠΟΙΗΣΕ INK #BC2F

Μας δίνει τον αριθμό ink όταν του δώσουμε την κωδικοποιημένη μορφή που βάζει όλα τα pixels ενός byte σ'αυτό το μελάνι.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο Α περιέχει το κωδικοποιημένο ink.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο Α περιέχει τον αντίστοιχο αριθμό ink. Τα flags είναι όλα επηρεασμένα.

ΤΟΠΟΘΕΤΗΣΕ INK **#BC32**

Δίνει τα δύο χρώματα που θα αντιστοιχούν σε κάποιον αριθμό ink. Αν τα χρώματα αυτά διαφέρουν, θα εναλλάσσονται και θα έχουμε flashing.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο Α περιέχει τον αριθμό ink, ο Β το πρώτο χρώμα και ο C το δεύτερο.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF, BC, DE και HL έχουν τυχαίο περιεχόμενο.

ΔΙΑΒΑΣΕ INK **#BC35**

Ελέγχει τα χρώματα που αντιστοιχούν σε κάποιο ink.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο Α περιέχει έναν αριθμό ink.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο Β περιέχει το πρώτο χρώμα που αντιστοιχεί σ'αυτό το ink και ο C το δεύτερο. Οι AF, DE και HL έχουν τυχαίο περιεχόμενο.

ΤΟΠΟΘΕΤΗΣΕ BORDER **#BC38**

Ορίζει τα χρώματα του border. Αν τα δύο χρώματα διαφέρουν θα έχουμε flashing.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο Β περιέχει το πρώτο χρώμα και ο C το δεύτερο.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF, BC, DE και HL έχουν τυχαίο περιεχόμενο.

ΔΙΑΒΑΣΕ BORDER **#BC3B**

Ελέγχει τα χρώματα του border.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο Β περιέχει το πρώτο χρώμα και ο C το δεύτερο. Οι AF, DE και HL έχουν τυχαίο περιεχόμενο.

ΤΟΠΟΘΕΤΗΣΕ FLASHING **#BC3E**

Εισάγει τις περιόδους που εναλλάσσονται τα χρώματα (flash).

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο H περιέχει την περίοδο του πρώτου χρώματος και ο L την περίοδο του δεύτερου.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF και HL έχουν τυχαίο περιεχόμενο.

ΔΙΑΒΑΣΕ FLASHING **#BC41**

Ελέγχει τις περιόδους που εναλλάσσονται τα χρώματα (flash).

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο H περιέχει την περίοδο του πρώτου χρώματος και ο L του δεύτερου. Ο AF έχει τυχαίο περιεχόμενο

ΓΕΜΙΣΕ ΚΟΥΤΙ 1 **#BC44**

Γεμίζει μια ορθογώνια περιοχή της οθόνης με ένα δεδομένο μελάνι (ink).

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο A περιέχει το κωδικοποιημένο ink που θα χρησιμοποιηθεί, ο H περιέχει την αριστερή στήλη που θα γεμίσει χρώμα, ο L την άνω σειρά, ο D την δεξιά στήλη και ο E την κάτω σειρά που θα γεμίσει χρώμα. Οι συντεταγμένες ξεκινούν από το 0.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF, BC, DE και HL έχουν τυχαίο περιεχόμενο.

ΣΗΜΕΙΩΣΗ: Το τρέχον MODE τυπώματος αγνοείται.

ΓΕΜΙΣΕ ΚΟΥΤΙ 2 **#BC47**

Γεμίζει μια ορθογώνια περιοχή της οθόνης με ένα δεδομένο ink, με όρια bytes.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο HL περιέχει τη διεύθυνση οθόνης της άνω αριστερής γωνίας της περιοχής που θα γεμίσει, ο D το πλάτος (σε bytes) που θα γεμίσει. Ο E περιέχει το ύψος αυτής της περιοχής (σε γραμμές οθόνης) και ο C το κωδικοποιημένο μελάνι που θα χρησιμοποιηθεί.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF, BC, DE και HL έχουν τυχαίο περιεχόμενο.

ΣΗΜΕΙΩΣΗ: Το mode τυπώματος γραφικών αγνοείται.

ΑΝΤΙΣΤΡΟΦΗ ΧΑΡΑΚΤΗΡΑ #BC4A

Όλα τα pixels ενός χαρακτήρα με ένα ink, εμφανίζονται με κάποιο άλλο (εναλλαγή χρώματος).

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο B και ο C περιέχουν τα δύο κωδικοποιημένα inks. Ο H ορίζει τη στήλη ενός κανονικού χαρακτήρα και ο L τη σειρά ενός κανονικού χαρακτήρα.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF, BC, DE και HL έχουν τυχαίο περιεχόμενο.

SCROLL ΟΘΟΝΗΣ #BC4D

Κάνε scroll ολόκληρη την οθόνη προς τα πάνω ή τα κάτω κατά οκτώ γραμμές pixels (ένα χαρακτήρα). Η νέα γραμμή που εισέρχεται στην οθόνη είναι κενή.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο B περιέχει 0 για scroll προς τα κάτω, ή μη μηδενική τιμή για προς τα πάνω. Ο A περιέχει το κωδικωποιημένο ink (χρώμα) που θα έχει η νέα γραμμή που εισάγεται.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF, BC, DE και HL έχουν τυχαίο περιεχόμενο.

ΜΕΤΑΤΡΕΨΕ ΠΙΝΑΚΑ ΧΑΡΑΚΤΗΡΑ #BC53

Μετατρέπει τον πίνακα ενός χαρακτήρα σε ένα σύνολο από κωδικοποιημένα pixels για το τρέχον mode της οθόνης.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο HL περιέχει τη διεύθυνση του πίνακα του χαρακτήρα και ο DE τη διεύθυνση της περιοχής που θα αποθηκευτεί το αποτέλεσμα.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF, BC, DE και HL έχουν τυχαίο περιεχόμενο.

ΕΠΑΝΕΦΕΡΕ ΠΙΝΑΚΑ ΧΑΡΑΚΤΗΡΑ #BC56

Μετατρέπει τον πίνακα ενός χαρακτήρα που είναι κωδικοποιημένος στην οθόνη, στη standard μορφή του.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο A περιέχει το κωδικοποιημένο μελάνι του

χαρακτήρα, ο Η τη στήλη που βρίσκεται και ο L τη σειρά. Ο DE περιέχει τη διεύθυνση μιας περιοχής όπου θα μεταφερθεί το αποτέλεσμα. (Οι συντεταγμένες ξεκινούν από 0).

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF, BC, DE και HL έχουν τυχαίο περιεχόμενο.

ΟΘΟΝΗ ΓΡΑΦΙΚΩΝ #BC59

Βάζει το mode οθόνης γραφικών.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο A περιέχει το mode που θέλουμε.
Οι τιμές αντιστοιχούν: 0 - Απόλυτο Mode
1 - XOR Mode
2 - AND Mode
3 - OR Mode

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF, BC, DE και HL έχουν τυχαίο περιεχόμενο.

ΓΡΑΨΕ PIXEL #BC5C

Γράφει ένα pixel στην οθόνη, αγνοώντας το mode οθόνης γραφικών.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο B περιέχει το κωδικοποιημένο ink που θα χρησιμοποιηθεί, ο C περιέχει τον κώδικα (mask) του pixel και ο HL τη διεύθυνση οθόνης.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο AF έχει τυχαίο περιεχόμενο.

ΟΡΙΖΟΝΤΙΟ PLOT ΓΡΑΜΜΗΣ #BC5F

Κάνει PLOT μια οριζόντια γραμμή.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο A περιέχει το απαιτούμενο κωδικοποιημένο ink, ο BC την οριζόντια συντεταγμένη X για το τέλος της γραμμής, ο DE την οριζόντια συντεταγμένη X για την αρχή της γραμμής και ο HL περιέχει την κατακόρυφη συντεταγμένη Y. (Ο DE δεν πρέπει να είναι μεγαλύτερος από τον BC).

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF, BC, DE και HL έχουν τυχαίο περιεχόμενο.

ΚΑΤΑΚΟΡΥΦΟ PLOT ΓΡΑΜΜΗΣ #BC62

Κάνει PLOT μια κατακόρυφη γραμμή.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο A περιέχει το απαιτούμενο κωδικοποιημένο ink, ο BC την κατακόρυφη συντεταγμένη Υ για το τέλος της γραμμής, ο HL την κατακόρυφη συντεταγμένη Υ για την αρχή της γραμμής και ο DE την οριζόντια συντεταγμένη Χ της γραμμής. (Ο HL δεν πρέπει να είναι μεγαλύτερος του BC).

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF, BC, DE και HL έχουν τυχαίο περιεχόμενο.

ΑΡΧΙΚΕΣ ΣΥΝΘΗΚΕΣ ΚΑΣΕΤΟΦΩΝΟΥ #BC65

Τοποθετεί τις αρχικές τιμές χειρισμού της κασέτας, κλείνοντας streams και βάζοντας τις αρχικές τιμές στην ταχύτητα μετάδοσης. Ενεργοποιεί τα διάφορα μηνύματα.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF, BC, DE και HL έχουν τυχαίο περιεχόμενο.

ΚΑΘΟΡΙΣΕ ΤΑΧΥΤΗΤΑ #BC68

Καθορίζει την ταχύτητα εγγραφής (μετάδοσης) και την αντιστάθμιση.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο A περιέχει την αντιστάθμιση, ο HL ορίζει την περίοδο εγγραφής. Η αντιστάθμιση μπορεί να είναι μεταξύ 0 και 255 (microseconds), αν και οι μεγάλες τιμές είναι ακατάλληλες. Τα περιεχόμενα του HL δίνουν την ημιπερίοδο ενός μηδενικού bit σε microseconds. Μια μέση ταχύτητα βρίσκεται ως $10^6/3*HL$) HL και πρέπει να είναι μεταξύ των τιμών 130 και 480. Οι αρχικές τιμές είναι: ΓΡΗΓΟΡΗ (2000 BAUD) HL=167, A=50, ΑΡΓΗ (1000 BAUD) HL=333, A=25.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF και HL έχουν τυχαίο περιεχόμενο.

ΣΗΜΕΙΩΣΗ: Η «αντιστάθμιση» χρησιμοποιείται για να μεταβάλει το λόγο διάρκειας των τετραγωνικών παλμών για το κασετόφωνο, με αρχική τιμή 25 microseconds. Η τιμή της προστίθεται στον κύκλο του υψηλού bit και αφαιρείται από τον κύκλο του χαμηλού bit, δίνοντας

έτσι έμφαση στη διαφορά μεταξύ του. Η αντιστάθμιση πρέπει να αυξάνεται όσο η μέση συχνότητα εγγραφής αυξάνεται.

ΕΛΕΓΧΟΣ ΜΗΝΥΜΑΤΟΣ ΚΑΣΕΤΑΣ #BC6B

Ενεργοποιεί ή αποενεργοποιεί τα μηνύματα που σχετίζονται με το χειρισμό του κασετοφώνου.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο A περιέχει 0 για ενεργοποίηση ή μη μηδενική τιμή για αποενεργοποίηση.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο AF έχει τυχαίο περιεχόμενο.

ΞΕΚΙΝΗΜΑ MOTER #BC6E

Ξεκινά το moter του κασετόφωνου και περιμένει την ταχύτητα να σταθεροποιηθεί.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο A περιέχει τον κωδικό της προηγούμενης κατάστασης του moter. Το carry flag είναι σε λογικό 1, εκτός αν η ρουτίνα εγκαταλήφθηκε πατώντας ESCAPE.

ΣΤΑΜΑΤΗΜΑ MOTER #BC71

Σταματάει το moter του κασετόφωνου.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο A περιέχει τον κωδικό της προηγούμενης κατάστασης του κασετόφωνου. Το carry flag είναι σε λογικό 1 εκτός αν πατήθηκε ESCAPE.

ΕΠΑΝΑΦΟΡΑ MOTER #BC74

Επαναφέρει το moter του κασετόφωνου στην προηγούμενη κατάστασή του.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο A περιέχει τον κώδικα της προηγούμενης κατάστασης.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Το carry flag είναι σε λογικό 1 εκτός αν πατήθηκε ESCAPE.

ΑΝΟΙΞΕ INPUT ΑΠΟ ΚΑΣΕΤΑ ΄Η ΔΙΣΚΕΤΑ #BC77

Διαβάζει τον header και τα 2 πρώτα Kbytes ενός προγράμματος από το κασετόφωνο ή το disk-drive.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο B περιέχει το μήκος του ονόματος του προγράμματος, ο HL τη διεύθυνση μνήμης όπου έχουμε αποθηκεύσει το όνομα αυτό και ο DE τη διεύθυνση ενός buffer των 2K.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Αν όλα πήγαν καλά, το carry είναι σε λογικό 1 και το zero flag σε λογικό 0. Ο HL έχει τη διεύθυνση του buffer που περιέχει τον header του προγράμματος, ο DE τη διεύθυνση που πρέπει να φορτωθούν τα bytes, ο BC το μήκος του προγράμματος και ο A τον τύπο του.

Αν υπήρχε ήδη ανοιχτό input, το carry και το zero flag είναι σε λογικό 0 και οι A, BC, DE και HL έχουν τυχαίο περιεχόμενο.
Αν πατήθηκε το ESC το carry είναι σε λογικό 0, το zero σε λογικό 1, οι A, BC, DE και HL έχουν τυχαίο περιεχόμενο.

Σε όλες τις περιπτώσεις ο IX έχει τυχαίο περιεχόμενο και τα άλλα flags έχουν επηρεαστεί.

ΚΛΕΙΣΕ INPUT ΑΠΟ ΚΑΣΕΤΑ ΄Η ΔΙΣΚΕΤΑ #BC7A

Κλείνει το input stream που έχουμε ανοίξει στο κασετόφωνο ή το disk-drive.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Το carry είναι σε λογικό 1 αν όλα πήγαν καλά, ή σε λογικό 0 αν δεν υπήρχε ανοικτό input stream. Οι A, BC, DE και HL έχουν τυχαίο περιεχόμενο.

ΕΓΚΑΤΕΛΕΙΨΕ INPUT ΑΠΟ ΚΑΣΕΤΑ ΄Η ΔΙΣΚΕΤΑ #BC7D

Εγκαταλείπει ένα πρόγραμμα που φορτώνουμε από το κασετόφωνο ή το disk-drive.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF, BC, DE και HL έχουν τυχαίο περιεχόμενο.

ΔΙΑΒΑΣΕ ΧΑΡΑΚΤΗΡΑ ΑΠΟ ΚΑΣΕΤΑ Ή ΔΙΣΚΕΤΑ #BC80

Διαβάζει έναν χαρακτήρα από το αρχείο που έχουμε ανοίξει στην κασέτα ή τη δισκέτα.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Αν όλα πήγαν καλά, ο A περιέχει τον κωδικό του χαρακτήρα, το carry είναι σε λογικό 1 και το zero flag σε λογικό 0.

Αν βρέθηκε τέλος αρχείου (E OF), ο A έχει τυχαίο περιεχόμενο, το carry είναι σε λογικό 0 και το zero σε λογικό 0.

Αν πατήθηκε ESC, ο A έχει τυχαίο περιεχόμενο, το carry είναι σε λογικό 0 και το zero σε λογικό 1.

Σε όλες τις περιπτώσεις ο IX έχει τυχαίο περιεχόμενο και τα άλλα flags έχουν επηρεαστεί.

ΔΙΑΒΑΣΕ ΠΡΟΓΡΑΜΜΑ ΑΠΟ ΚΑΣΕΤΑ Ή ΔΙΣΚΕΤΑ #BC83

Διαβάζει ολόκληρο το πρόγραμμα (που έχουμε ανοίξει σαν input stream) από το κασετόφωνο ή το disk-drive.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο HL περιέχει τη διεύθυνση που θα φορτωθεί το πρόγραμμα.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Αν όλα πήγαν καλά, ο HL περιέχει τη διεύθυνση που ξεκινάει να τρέχει το πρόγραμμα (από το header), το carry είναι σε λογικό 1 και το zero σε λογικό 0.

Αν δεν υπήρχε ανοικτό input stream, ο HL έχει τυχαίο περιεχόμενο, το carry και το zero flag είναι σε λογικό 0.

Αν πατήθηκε ESC, ο HL έχει τυχαίο περιεχόμενο, το carry είναι σε λογικό 0 και το zero σε λογικό 1.

Σε όλες τις περιπτώσεις οι A, BC, DE και IX έχουν τυχαίο περιεχόμενο και τα άλλα flags έχουν επηρεαστεί.

ΕΠΙΣΤΡΕΨΕ ΧΑΡΑΚΤΗΡΑ #BC86

Βάζει τον τελευταίο χαρακτήρα που διαβάστηκε από το κασετόφωνο ή το disk-drive πίσω στον buffer. Ο χαρακτήρας αυτός θα ξαναδιαβαστεί αν καλέσουμε τη ρουτίνα «ΔΙΑΒΑΣΕ ΧΑΡΑΚΤΗΡΑ».

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Δεν έχει γίνει καμία αλλαγή στους καταχωρητές και τα flags.

ΕΛΕΓΞΕ ΤΕΛΟΣ ΑΡΧΕΙΟΥ #BC89

Ελέγχει αν φτάσαμε το τέλος ενός αρχείου που διαβάζουμε από κασέτα ή δισκέτα.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Αν δεν υπάρχει EOF, το carry σε λογικό 1 και το zero flag σε λογικό 0.

Αν συναντήθηκε EOF, το carry και το zero είναι σε λογικό 0.

Αν πατήθηκε ESC, το carry είναι σε λογικό 0 και το zero σε λογικό 1.

Σε όλες τις περιπτώσεις οι A και IX έχουν τυχαίο περιεχόμενο και στα υπόλοιπα flags έχουν επηρεαστεί.

ΑΝΟΙΞΕ OUTPUT ΣΕ ΚΑΣΕΤΑ Η ΔΙΣΚΕΤΑ #BC8C

Ανοίγει ένα output stream (κανάλι εξόδου) στο κασετόφωνο ή το disk-drive.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο B περιέχει το μήκος του ονόματος του προγράμματος που θέλουμε να σώσουμε, ο HL τη διεύθυνση που έχουμε το όνομα στη μνήμη (σε κωδικούς ASCII) και ο DE τη διεύθυνση ενός BUFFER των 2K.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Αν υπήρχε ήδη ανοιχτό output stream, το carry είναι σε λογικό 0, ο HL έχει τυχαίο περιεχόμενο, και ο A περιέχει #OE.

Αν όλα πήγαν καλά, το carry είναι σε λογικό 1 και ο HL περιέχει τη διεύθυνση του buffer του header.

Και στις δύο περιπτώσεις το zero είναι σε λογικό 0 και οι A, BC, DE και IX έχουν τυχαίο περιεχόμενο. Τα άλλα flags έχουν επηρεαστεί.

ΚΛΕΙΣΕ OUTPUT STREAM #BC8F

Κλείνει ένα output stream σε κασέτα ή δισκέτα.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Αν όλα πήγαν καλά, το carry είναι σε λογικό 1 και το zero flag σε λογικό 0.

Αν δεν υπήρχε ανοιχτό output stream, το carry και το zero flag είναι σε λογικό 0.

Αν πατήθηκε ESC, το carry είναι σε λογικό 0 και το zero σε λογικό 1.

Σε όλες τις περιπτώσεις οι A, BC, DE, HL και IX έχουν τυχαίο περιεχόμενο και τα υπόλοιπα flags έχουν επηρεαστεί.

ΕΓΚΑΤΑΛΕΙΨΕ OUTPUT STREAM #BC92

Εγκαταλείπει το output stream που έχουμε ανοίξει σε κασέτα ή δισκέτα.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF, BC, DE και HL έχουν τυχαίο περιεχόμενο.

ΓΡΑΨΕ ΧΑΡΑΚΤΗΡΑ ΣΕ ΚΑΝΑΛΙ ΕΞΟΔΟΥ #BC95

Γράφει έναν χαρακτήρα στο output stream που έχουμε ανοίξει, σε κασέτα ή δισκέτα.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο A περιέχει τον κωδικό του χαρακτήρα.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Αν όλα πήγαν καλά το carry είναι σε λογικό 1 και το zero σε λογικό 0.

Αν δεν υπάρχει ανοικτό output stream, το carry και το zero είναι σε λογικό 0.

Αν πατήθηκε ESC, το carry είναι σε λογικό 0 και το zero σε λογικό 1.

Σε όλες τις περιπτώσεις οι A και IX έχουν τυχαίο περιεχόμενο και τα υπόλοιπα flags έχουν επηρεαστεί.

ΣΩΣΕ ΟΛΟΚΛΗΡΟ ΠΡΟΓΡΑΜΜΑ #BC98

Σώζει ολόκληρο το πρόγραμμα σε κασέτα ή δισκέτα.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο HL περιέχει την αρχική διεύθυνση των bytes που θέλουμε να σώσουμε, ο DE το μήκος του προγράμματος, ο BC τη διεύθυνση που ξεκινάει να τρέχει το πρόγραμμα (η οποία θα γραφτεί στον header) και ο A τον τύπο του προγράμματος.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Αν όλα πήγαν καλά το carry είναι σε λογικό 1 και το zero flag σε λογικό 0.

Αν δεν υπήρχε ανοικτό output stream, το carry και το zero είναι σε λογικό 0.

Αν πατήθηκε ESC το carry είναι σε λογικό 0 και το zero σε λογικό 1.

Σε όλες τις περιπτώσεις οι A, BC, DE, HL και IX έχουν τυχαίο περιεχόμενο και τα άλλα flags έχουν επηρεαστεί.

CATALOG **#BC9B**

Κάνει CAT στην κασέτα ή τη δισκέτα (όπως και η αντίστοιχη εντολή της BASIC).

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο DE περιέχει τη διεύθυνση ενός buffer των 2K.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Αν όλα πήγαν καλά το carry είναι σε λογικό 1, αλλιώς αν υπάρχει ανοιχτό input stream είναι σε λογικό 0. Σε όλες τις περιπτώσεις το zero flag είναι σε λογικό 0, οι A, BC, DE, HL και IX έχουν τυχαίο περιεχόμενο και τα άλλα flags έχουν επηρεαστεί.

ΓΡΑΨΕ ΣΕ ΚΑΣΕΤΑ **#BC9E**

Γράφει ένα σύνολο bytes σε κασέτα (μόνο).

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο HL περιέχει την αρχική διεύθυνση των bytes, ο DE το μήκος και ο A το χαρακτήρα συγχρονισμού (#2C για header ή #16 για data).

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Αν όλα πήγαν καλά το carry είναι σε λογικό 1 και ο A έχει τυχαίο περιεχόμενο, αλλιώς το carry είναι σε λογικό 0 και ο A περιέχει έναν κωδικό σφάλματος. (0 αν πατήθηκε ESCAPE ή 1 για σφάλμα στο σώσιμο των δεδομένων στην κασέτα).

Σε όλες τις περιπτώσεις οι BC, DE, HL και IX έχουν τυχαίο περιεχόμενο.

ΔΙΑΒΑΣΕ ΑΠΟ ΚΑΣΕΤΑ **#BCA1**

Διαβάζει ένα σύνολο bytes από κασέτα (μόνο).

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο HL περιέχει τη διεύθυνση που θα φορτωθούν τα δεδομένα, ο DE το μήκος τους και ο A τον αντίστοιχο χαρακτήρα συγχρονισμού.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Αν όλα πήγαν καλά το carry είναι σε λογικό 1 και ο A έχει τυχαίο περιεχόμενο. Αλλιώς το carry είναι σε λογικό 0 και ο A έχει έναν κωδικό σφάλματος. (0 αν πατήθηκε ESC, 1 αν υπήρξε σφάλμα κατά την ανάγνωση, 2 αν υπήρξε σφάλμα κατά τον κυκλικό

έλεγχο των δεδομένων).

Σε όλες τις περιπτώσεις οι BC, DE, HL και IX έχουν τυχαίο περιεχόμενο και τα άλλα flags έχουν επηρεαστεί.

VERIFY (ΓΙΑ ΚΑΣΕΤΑ) #BCA4

Συγκρίνει τα δεδομένα στην κασέτα (μόνο) με τα περιεχόμενα της μνήμης.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο HL περιέχει τη διεύθυνση των δεδομένων της μνήμης που θα συγκριθούν, ο DE το μήκος τους και ο A τον αναμενόμενο χαρακτήρα συγχρονισμού.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Αν ο έλεγχος ήταν επιτυχής το carry είναι σε λογικό 1 και ο A έχει τυχαίο περιεχόμενο. Αλλιώς το carry είναι σε λογικό 0 και ο A περιέχει έναν κωδικό σφάλματος. (0 αν πατήθηκε ESC, 1 αν υπήρξε σφάλμα κατά την ανάγνωση, 2 αν υπήρξε σφάλμα κατά τον έλεγχο που γίνεται όταν φορτώνονται δεδομένα από την κασέτα ή 3 αν η σύγκριση απέτυχε).

Σε όλες τις περιπτώσεις οι BC, DE, HL και IX έχουν τυχαίο περιεχόμενο και τα άλλα flags έχουν επηρεαστεί.

RESET ΗΧΟΥ #BCA7

Κάνει reset τον ήχο, «κλείνοντας» ότι ήχο ακούγεται τη στιγμή που καλείται η ρουτίνα.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF, BC, DE και HL έχουν τυχαίο περιεχόμενο.

ΗΧΗΤΙΚΟΣ ΦΟΝΤΟΣ #BCAA

Προσθέτει έναν ήχο στον «ηχητικό φόντο», αν αυτό είναι δυνατόν.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο HL δείχνει ένα block από 9 bytes που ορίζουν τον ήχο που θέλουμε, τα οποία είναι:

BYTE 0: Bit 0: Κανάλι A
Bit 1: Κανάλι B
Bit 2: Κανάλι C
Bit 3: Ραντεβού με το A
Bit 4: Ραντεβού με το B
Bit 5: Ραντεβού με το C

Bit 6: Περίμενε μέχρι να ενεργοποιηθεί
Bit 7: Flush
BYTE 1: Επιλογή ENVELOPE πλάτους
BYTE 2: Επιλογή ENVELOPE τόνου
BYTE 3: Περίοδος τόνου
BYTE 4:
BYTE 5: Περίοδος θορύβου
BYTE 6: Αρχικό πλάτος
BYTE 7: Διάρκεια ή επανάληψη
BYTE 8:

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Αν όλα πήγαν καλά το carry είναι σε λογικό 1 και ο HL έχει τυχαίο περιεχόμενο. Αν όχι, το carry είναι σε λογικό 0 και ο HL παραμένει. Οι A, BC, DE και IX έχουν τυχαίο περιεχόμενο και τα άλλα flags έχουν επηρεαστεί.

ΕΛΕΓΞΕ ΗΧΗΤΙΚΟ ΦΟΝΤΟ #BCAD

Ελεγξε αν υπάρχει «χώρος» μέσα σε ένα ηχητικό φόντο.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο A περιέχει το bit που δείχνει ποιο κανάλι θέλουμε να ελέγξουμε (δες προηγούμενη ρουτίνα).

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο A περιέχει την κατάσταση του καναλιού που θέλουμε να ελέγξουμε:
Bits 0-2: Αριθμός ελεύθερων «οπών» στο ηχητικό φόντο.
Bit 3: Περιμένει ραντεβού με το A
Bit 4: Περιμένει ραντεβού με το B
Bit 5: Περιμένει ραντεβού με το C
Bit 6: Περιμένει ενεργοποίηση
Bit 7: Παράγεται ήχος.

Οι BC, DE και HL έχουν τυχαίο περιεχόμενο και τα flags έχουν επηρεαστεί.

ΕΝΕΡΓΟΠΟΙΗΣΗ ΡΟΥΤΙΝΑΣ ΚΕΝΟΥ ΦΟΝΤΟΥ #BCB0

Δίνει μια ρουτίνα (event) που θα τρέχει όταν το ηχητικό φόντο για κάποια κανάλια είναι κενό.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο A περιέχει το bit που υποδεικνύει ένα κανάλι και ο HL τη διεύθυνση της ρουτίνας που θα εκτελείται.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF, BC, DE και HL έχουν τυχαίο περιεχόμενο.

ΕΝΕΡΓΟΠΟΙΗΣΕ ΗΧΟ **#BCB3**

Ενεργοποιεί έναν ήχο που περιμένει.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο A περιέχει τα bit που υποδεικνύουν το/τα κανάλια (όπως προηγουμένως).

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF, BC, DE, HL και IX έχουν τυχαίο περιεχόμενο.

ΣΤΑΜΑΤΗΣΕ ΗΧΟ **#BCB6**

Σταματά όλους τους ήχους αμέσως.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Αν υπήρχε ήχος το carry είναι σε λογικό 1. Οι A, BC και HL έχουν τυχαίο περιεχόμενο και τα flags είναι επηρεασμένα.

ΣΥΝΕΧΙΣΕ ΗΧΟ **#BCB9**

Ξαναξεκινάει τους ήχους που είχαν σταματήσει με την προηγούμενη ρουτίνα.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF, BC, DE και IX έχουν τυχαίο περιεχόμενο.

ENVELOPE ΠΛΑΤΟΥΣ **#BCBC**

Στήνει έναν envelope πλάτους.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο A περιέχει τον αριθμό envelope, ο HL τη διεύθυνση ενός block δεδομένων που δίνονται παρακάτω:

Byte 0: Αριθμός τμημάτων
Bytes 1-3: Τμήμα 1
Bytes 4-6: Τμήμα 2
Bytes 7-9: Τμήμα 3
Bytes 10-12: Τμήμα 4
Bytes 13-15: Τμήμα 5

Μέσα σε κάθε τμήμα το πρώτο byte δίνει το μέτρημα βήματος (step count), το δεύτερο το μέγεθος βήματος (step size) και το τρίτο το χρόνο παύσης (pause time). Αχρησιμοποίητα τμήματα δε χρειάζεται

να περιέχουν δεδομένα. Το block αυτό δεν πρέπεινα βρίσκεται σε εκείνα τα τμήματα της RAM που υπάρχουν παράλληλα με την ανώτερη ή την κατώτερη ROM.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Το carry είναι σε λογικό 1 αν όλα πήγαν καλά. Ο HL περιέχει τη διεύθυνση του block συν #10, ο A και ο BC έχουν τυχαίο περιεχόμενο.

ΣΗΜΕΙΩΣΗ: Επειδή τα δεδομένα αυτής της ρουτίνας καθώς και της επόμενης είναι πολύ πολύπλοκα, μπορείτε να βοηθηθείτε ανατρέχοντας στο Manual του υπολογιστή.

ENVELOPE ΤΟΝΟΥ #BCBF

Στήνει έναν envelope για τον τόνο.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Οι ίδιες με την προηγούμενη ρουτίνα «ENVELOPE ΠΛΑΤΟΥΣ», με τη μόνη διαφορά ότι το μέγεθος βήματος αναφέρεται στην οξύτητα (pitch) αντί για το πλάτος (amplitude).

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ίδιες με την προηγούμενη ρουτίνα.

ΔΙΕΥΘΥΝΣΗ ENVELOPE ΠΛΑΤΟΥΣ #BCC2

Βρίσκει τη διεύθυνση των δεδομένων για κάποιο envelope πλάτους.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο A περιέχει τον αριθμό envelope.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Αν όλα πήγαν καλά το carry είναι σε λογικό 1, ο HL περιέχει τη διεύθυνση του block δεδομένων και ο BC το μήκος του envelope. Αν η ρουτίνα απέτυχε το carry είναι σε λογικό 0, ο HL έχει τυχαίο περιεχόμενο και ο BC διατηρείται. Ο A σε κάθε περίπτωση έχει τυχαίο περιεχόμενο.

ΔΙΕΥΘΥΝΣΗ ENVELOPE ΤΟΝΟΥ #BCC5

Επιστρέφει τη διεύθυνση των δεδομένων για κάποιο envelope τόνου.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ίδιες όπως στην προηγούμενη ρουτίνα.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ίδιες όπως στην προηγούμενη ρουτίνα «ΔΙΕΥΘΥΝΣΗ ENVELOPE ΠΛΑΤΟΥΣ».

ΚΑΘΑΡΙΣΕ ΡΟΥΤΙΝΕΣ ΣΧΕΤΙΚΕΣ ΜΕ TIMER #BCC8

Καθαρίζει όλες τις σειρές ρουτινών που υπάρχουν σαν events, τα blocks που αντιστοιχούν στα frame flybacks καθώς και όλες τις ρουτίνες που υπάρχουν σαν σύγχρονο events εκτός των ρουτινών του ήχου, και του ελέγχου του πληκτρολογίου.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο B περιέχει τη διεύθυνση επιλογής ROM, από τη ROM που βρίσκεται σε «πρώτο πλάνο» (αν υπάρχει), ο C τη διεύθυνση επιλογής ROM για ένα πρόγραμμα της RAM που βρίσκεται σε πρώτο πλάνο και ο DE τη διεύθυνση εισόδου στην τρέχουσα ROM που βρίσκεται σε πρώτο πλάνο. Οι AF και HL έχουν τυχαίο περιεχόμενο.

ΣΗΜΕΙΩΣΗ: FRAME FLYBACK είναι ένα σήμα του CRT controller που χρησιμεύει για να δηλώνει την έναρξη του κατακόρυφου παλμού επιστροφής για την οθόνη.
Κατά τη διάρκεια αυτού του παλμού η οθόνη δε γράφεται, οπότε μπορούν να γίνουν διάφορες λειτουργίες, χωρίς να έχουμε, άσχημο από πλευράς εμφάνισης αποτέλεσμα (π.χ. σχεδιασμός sprites ή scrolling της οθόνης).

Το σήμα FRAME FLYBACK διαρκεί μόνο μερικές εκατοντάδες microsecond αλλά η διάρκεια του παλμού επιστροφής είναι αρκετά πιο μεγάλη. Ωστόσο, θα υπάρξει ένα interrupt στη μέση του παραπάνω παλμού που θα προκαλέσει το πάγωμα της επεξεργασίας οθόνης για ένα σημαντικό χρόνο. Είναι σκόπιμο λοιπόν να εκτελεστεί η λειτουργία που θέλουμε όσο το δυνατόν πιο γρήγορα μετά το σήμα FRAME FLYBACK.

ΕΝΕΡΓΟΠΟΙΗΣΕ ΤΙΣ ROM #BCCB

Αναγνωρίζει και ενεργοποιεί όλες τις ROM (εκτός αυτή που βρίσκεται σε πρώτο πλάνο και ήδη λειτουργεί).

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο DE περιέχει τη διεύθυνση του πρώτου χρησιμοποιήσουμε byte της μνήμης, και ο HL του τελευταίου χρησιμοποιήσιμου byte.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο DE περιέχει τη διεύθυνση του νέου πρώτου χρησιμοποιήσιμου byte, και ο HL του νέου τελευταίου χρησιμοποιήσιμου byte. Ο AF και ο BC έχουν τυχαίο περιεχόμενο.

ΕΝΕΡΓΟΠΟΙΗΣΕ ΜΙΑ ROM #BCCE

Ενεργοποιεί τη ROM πού θέλουμε.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο C περιέχει τον αριθμό ROM που θέλουμε (διεύθυνση επιλογής ROM), ο DE τη διεύθυνση του πρώτου χρησιμοποιήσιμου byte της μνήμης και ο HL του τελευταίου χρησιμοποιήσιμου byte.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο DE περιέχει τη διεύθυνση του νέου πρώτου χρησιμοποιήσιμου byte και ο HL του νέου τελευταίου χρησιμοποιήσιμου byte της μνήμης. Οι AF και B έχουν τυχαίο περιεχόμενο.

ΤΟΠΟΘΕΤΗΣΕ EXTRA ΕΝΤΟΛΕΣ #BCD1

Τοποθετεί μια επέκταση στις εντολές της Basic (RSX).

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο BC περιέχει τη διεύθυνση του block νέων εντολών και ο HL τη διεύθυνση ενός buffer 4 bytes στη RAM.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο DE έχει τυχαίο περιεχόμενο.

ΣΗΜΕΙΩΣΗ: Ούτε το block νέων εντολών ούτε ο buffer των 4 bytes μπορούν να είναι σε μία διεύθυνση που βρίσκεται παράλληλα με κάποια ROM. Το πως χρησιμοποιείται αυτή η ρουτίνα περιγράφεται στην αρχή του κεφαλαίου 7 αυτού του βιβλίου.

ΒΡΕΣ EXTRA ΕΝΤΟΛΗ #BCD4

Ψάχνει για μία extra εντολή (RSX).

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο HL περιέχει τη διεύθυνση στην οποία είναι αποθηκευμένο το όνομα της εντολής.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Αν βρέθηκε η εντολή το carry είναι σε λογικό 1, ο C περιέχει τη διεύθυνση επιλογής ROM (αριθμός ROM) και ο HL τη διεύθυνση της αντίστοιχης ρουτίνας που καλεί η εντολή. Αν η εντολή δε βρέθηκε, το carry είναι σε λογικό 0 και οι HL και C έχουν τυχαίο περιεχόμενο. Σε όλες τις περιπτώσεις οι A, B και DE έχουν τυχαίο περιεχόμενο.

ΣΗΜΕΙΩΣΗ: Κάθε όνομα εντολής έχει maximum 16 χαρακτήρες. Ο τελευταίος χαρακτήρας του ονόματος της εντολής είναι προσαυξημένος κατά #80.

ΝΕΟ FRAME FLY **#BCD7**

Δίνει αρχικές τιμές και προσθέτει ένα block στη λίστα του FRAME FLYBACK. (Για την εξήγηση του FRAME FLYBACK βλέπε τη σημείωση στη ρουτίνα #BCC8).

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο HL περιέχει τη διεύθυνση του block, ο B την κατηγορία του EVENT, ο C περιέχει την απαιτούμενη διεύθυνση επιλογής ROM (αριθμό ROM) για τη ρουτίνα του EVENT, και ο DE τη διεύθυνση αυτής της ρουτίνας.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF, BC και HL έχουν τυχαίο περιεχόμενο.

ΠΡΟΣΘΕΣΕ BLOCK ΣΤΟ FRAME FLY **#BCDA**

Προσθέτει ένα block στη λίστα του FRAME FLYBACK. (Για την εξήγηση του FRAME FLYBACK βλέπε τη σημείωση στη ρουτίνα #BCC8).

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο HL περιέχει τη διεύθυνση του block.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF, DE και HL έχουν τυχαίο περιεχόμενο.

ΑΦΑΙΡΕΣΕ BLOCK ΑΠΟ FLYBACK **#BCDD**

Αφαιρεί ένα block από τη λίστα του FRAME FLYBACK. (Για την εξήγηση του FRAME FLYBACK δες τη σημείωση της ρουτίνας #BCC8).

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο HL περιέχει τη διεύθυνση του block.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF, DE και HL έχουν τυχαίο περιεχόμενο.

ΝΕΟ FAST TICKER **#BCE0**

Δίνει αρχικές τιμές και προσθέτει ένα block στη λίστα του event που συγχρονίζεται με το FAST TICKER (παλμοί που παράγει ο TIMER κάθε τριακοσιοστό του δευτερολέπτου).

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο HL περιέχει τη διεύθυνση του block, ο B την κατηγορία του event, ο C τη διεύθυνση επιλογής ROM (αριθμός ROM) και ο DE τη διεύθυνση της ρουτίνας που αντιστοιχεί στο event.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF, DE και HL έχουν τυχαίο περιεχόμενο.

ΠΡΟΣΘΕΣΕ ΣΤΟ FAST TICKER **#BCE3**

Προσθέτει ένα block στη λίστα του event που συγχρονίζεται με το FAST TICKER.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο HL περιέχει τη διεύθυνση του block.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF, DE και HL έχουν τυχαίο περιεχόμενο.

ΑΦΑΙΡΕΣΕ ΑΠΟ ΤΟ FAST TICKER **#BCE6**

Αφαιρεί ένα block από τη λίστα του event που συγχρονίζεται με το FAST TICKER.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο HL περιέχει τη διεύθυνση του block.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF, DE και HL έχουν τυχαίο περιεχόμενο.

ΠΡΟΣΘΕΣΕ ΣΤΟ TICKER **#BCE9**

Προσθέτει ένα block στη λίστα του event που συγχρονίζεται με το TICKER (παλμοί που παράγει ο TIMER κάθε πεντηκοστό του δευτερολέπτου).

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο HL περιέχει τη διεύθυνση του block, ο DE την αρχική τιμή του μετρητή και ο BC την τιμή που παίρνει ο μετρητής κάθε φορά, μετά την εκτέλεση της ρουτίνας.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF, BC, DE και HL έχουν τυχαίο περιεχόμενο.

ΑΦΑΙΡΕΣΕ ΑΠΟ ΤΟ TICKER **#BCEC**

Αφαιρεί ένα block από τη λίστα του event που συγχρονίζεται με το TICKER.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο HL περιέχει τη διεύθυνση του block.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Αν το block βρέθηκε το carry είναι σε λογικό 1 και ο DE περιέχει την τρέχουσα τιμή του μετρητή. Αλλιώς το carry είναι σε λογικό 0 και ο DE έχει τυχαίο περιεχόμενο. Σε όλες τις περιπτώσεις οι A και HL τυχαίο περιεχόμενο και τα άλλα flags έχουν επηρεαστεί.

ΝΕΟ EVENT BLOCK #BCEF

Δίνει αρχικές τιμές και ενεργοποιεί ένα event block.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο HL περιέχει τη διεύθυνση του event block, ο B την κατηγορία του event, ο C τη διεύθυνση επιλογής ROM (αριθμός ROM) για το event και ο DE τη διεύθυνση της ρουτίνας που αντιστοιχεί στο event.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο HL περιέχει τη διεύθυνση του event block +7 και όλοι οι άλλοι καταχωρητές διατηρούνται.

ΕΝΕΡΓΟΠΟΙΗΣΕ EVENT #BCF2

Ενεργοποιεί ένα event block ώστε να εκτελεστεί η αντίστοιχη ρουτίνα.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο HL περιέχει τη διεύθυνση του event block.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF, BC, DE και HL έχουν τυχαίο περιέχομενο.

ΚΑΝΕ RESET ΤΑ ΣΥΓΧΡΟΝΑ EVENTS #BCF5

Καθαρίζει τη σειρά των σύγχρονων events.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF και HL έχουν τυχαίο περιεχόμενο.

ΑΦΑΙΡΕΣΕ ΣΥΓΧΡΟΝΟ EVENT #BCF8

Αφαιρεί ένα event από τη σειρά των σύγχρονων events.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο HL περιέχει τη διεύθυνση του event block.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF, BC, DE και HL έχουν τυχαίο περιεχόμενο.

ΕΠΟΜΕΝΟ ΣΥΓΧΡΟΝΟ EVENT #BCFB

Παίρνει το επόμενο event από τη σειρά των σύγχρονων events.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Αν υπάρχει επόμενο event το οποίο περιμένει

να εκτελεστεί το carry είναι σε λογικό 1, ο HL περιέχει τη διεύθυνση του event block και ο A την προηγούμενη προτεραιότητα του event (αν υπάρχει). Αλλιώς το carry είναι σε λογικό 0, και οι A και HL έχουν τυχαίο περιεχόμενο. Ο DE έχει τυχαίο περιεχόμενο σε όλες τις περιπτώσεις.

ΕΚΤΕΛΕΣΕ ΣΥΓΧΡΟΝΟ EVENT **#BCFE**

Εκτελεί τη ρουτίνα ενός event.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο HL περιέχει τη διεύθυνση του event block.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF, BC, DE και HL έχουν τυχαίο περιεχόμενο.

ΟΛΟΚΛΗΡΩΣΕ ΣΥΓΧΡΟΝΟ EVENT **#BD01**

Ολοκληρώνει την επεξεργασία ενός σύγχρονου event. Χρησιμεύει για να επαναφέρει την τρέχουσα προτεραιότητα του event, και να δώσει αντίστοιχη τιμή στο μετρητή.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο C περιέχει την προηγούμενη προτεραιότητα του event, και ο HL τη διεύθυνση του event block.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF, BC, DE και HL έχουν τυχαίο περιεχόμενο.

ΑΠΟΕΝΕΡΓΟΠΟΙΗΣΕ EVENTS **#BD04**

Αποενεργοποιεί όλα τα κανονικά σύγχρονα events.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο HL έχει τυχαίο περιεχόμενο.

ΕΝΕΡΓΟΠΟΙΗΣΕ EVENTS **#BD07**

Ενεργοποιεί όλα τα κανονικά σύγχρονα events.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο HL έχει τυχαίο περιεχόμενο.

ΑΦΟΠΛΙΣΕ EVENT **#BD0A**

Εμποδίζει ένα event από το να εκτελεστεί.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο HL περιέχει τη διεύθυνση του event block.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο AF έχει τυχαίο περιεχόμενο.

ΒΡΕΣ ΩΡΑ **#BD0D**

Βρίσκει την ώρα που πέρασε από τότε που ανοίξαμε τον υπολογιστή ή τον κάναμε RESET ή καθορίσαμε την ώρα σε τριακοσιοστά του δευτερολέπτου.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι DE και HL περιέχουν την τιμή του μετρητή (που είναι 4 bytes) και ο DE έχει τα δύο πιο σημαντικά bytes.

ΚΑΘΟΡΙΣΕ ΩΡΑ **#BD10**

Δίνει αρχική τιμή στον μετρητή της ώρας.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Οι DE και HL περιέχουν τα 4 bytes της αρχικής τιμής που θέλουμε (ο DE τα δύο σημαντικά).

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο AF έχει τυχαίο περιεχόμενο.

ΦΟΡΤΩΣΕ ΚΑΙ ΤΡΕΞΕ ΠΡΟΓΡΑΜΜΑ (BOOT) **#BD13**

Φορτώνει και τρέχει κάποιο πρόγραμμα.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο HL περιέχει τη διεύθυνση της ρουτίνας που καλείται για να φορτώσει το πρόγραμμα.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Αν όλα πάνε καλά δεν υπάρχει έξοδος. Αν συμβεί κάποιο λάθος το carry θα επιστραφεί σε λογικό 0.

ΤΡΕΞΕ ΠΡΟΓΡΑΜΜΑ **#BD16**

Τρέχει ένα πρόγραμμα που βρίσκεται σε πρώτο πλάνο.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο HL περιέχει τη διεύθυνση που θα ξεκινήσει να τρέχει και ο C τη διεύθυνση επιλογής ROM (αριθμός ROM).

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Δεν υπάρχει έξοδος.

ΠΕΡΙΜΕΝΕ FRAME FLYBACK #BD19

Περιμένει frame flyback. (Δες σημείωση της ρουτίνας #BCC8).

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι τιμές όλων των καταχωρητών καθώς και τα flags διατηρούνται.

ΒΑΛΕ MODE #BD1C

Βάζει το mode της οθόνης.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο A περιέχει το mode που θέλουμε.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο AF έχει τυχαίο περιεχόμενο.

ΒΑΛΕ OFFSET ΟΘΟΝΗΣ #BD1F

Φορτώνει στο hardware την απόκλιση του πρώτου byte της οθόνης από την αρχική διεύθυνση της VIDEO RAM, μέσα σε ένα από τα 8 blocks των 2K της οθόνης, καθώς και σε ποια θέση βρίσκεται η VIDEO RAM.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο A περιέχει την απαιτούμενη βάση της οθόνης (#C0 ή #40), και ο HL την απαιτούμενη απόκλιση.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο AF έχει τυχαίο περιεχόμενο.

ΚΑΘΑΡΙΣΕ ΤΑ INKS #BD22

Βάζει όλα τα inks σε ένα χρώμα καθώς επίσης και το χρώμα του border.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο DE είναι φορτωμένος έτσι ώστε ο D να περιέχει το κοινό χρώμα των ink και ο E το χρώμα του border.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο AF έχει τυχαίο περιεχόμενο.

ΚΑΘΟΡΙΣΕ ΤΑ INKS #BD25

Καθορίζει τα χρώματα για όλα τα inks.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο DE περιέχει τη διεύθυνση ενός block για τον καθορισμό των inks, όπου το byte 0 δίνει το χρώμα του border και τα bytes 1-16 το χρώμα των inks 0-15 αντίστοιχα.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο AF έχει τυχαίο περιεχόμενο.

KANE RESET TON PRINTER #BD28

Επαναφέρει τις αρχικές μεταβλητές του λειτουργικού για τον εκτυπωτή.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF, BC, DE και HL έχουν τυχαίο περιεχόμενο.

ΤΥΠΩΣΕ ΧΑΡΑΚΤΗΡΑ ΣΤΟΝ PRINTER #BD2B

Στέλνει ένα χαρακτήρα στην έξοδο του PRINTER.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο Α περιέχει το κωδικό του χαρακτήρα. (Το όγδοο bit θα αγνοηθεί).

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Αν όλα πήγαν καλά το carry είναι σε λογικό 1, αλλιώς αν πέρασε ο χρόνος αναμονής (- 0.4 sec) χωρίς να γίνει τίποτα το carry είναι σε λογικό 0. Πάντα ο Α έχει τυχαίο περιεχόμενο και τα άλλα flags έχουν επηρεαστεί.

ΕΛΕΓΞΕ PRINTER #BD2E

Ελέγχει αν ο εκτυπωτής είναι απασχολημένος (busy).

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Το carry είναι σε λογικό 1 αν η έξοδος του printer δεν είναι διαθέσιμη (απασχολημένη), αλλιώς είναι σε λογικό 0. Οι άλλες σημαίες έχουν επηρεαστεί.

ΣΤΕΙΛΕ ΧΑΡΑΚΤΗΡΑ ΣΤΟΝ PRINTER #BD31

Στέλνει ένα χαρακτήρα στην έξοδο του εκτυπωτή (αφού έχουμε ελέγξει ότι δεν είναι απασχολημένη).

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο Α περιέχει το χαρακτήρα που θέλουμε να στείλουμε (7-bit).

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Το carry είναι σε λογικό 1 και ο Α έχει τυχαίο περιεχόμενο.

ΦΟΡΤΩΣΕ ΚΑΤΑΧΩΡΗΤΗ ΟΛΟΚΛΗΡΩΜΕΝΟΥ ΗΧΟΥ #BD34

Στέλνει δεδομένα στο ολοκληρωμένο του ήχου.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο A περιέχει τον αριθμό του REGISTER και ο C το byte που θέλουμε να στείλουμε.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF και BC έχουν τυχαίο περιεχόμενο.

ΕΠΑΝΑΦΟΡΑ JYMPBLOCK #BD37

Επαναφέρει το standard jymblock (πίνακα κλήσεων) του λειτουργικού συστήματος δηλαδή αυτής της περιοχής της RAM που αναφέραμε σ' αυτό το κεφάλαιο και περιέχει τις κλήσεις του λειτουργικού.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF, BC, DE και HL έχουν τυχαίο περιεχόμενο.

Εδώ ολοκληρώθηκαν οι άμεσες κλήσεις του λειτουργικού συστήματος. Στη συνέχεια θα σας παρουσιάσουμε τις έμμεσες κλήσεις του λειτουργικού. Ενδιάμεσα όπως θα δείτε, υπάρχει ένας χώρος μνήμης για τον οποίο όμως δεν υπάρχει διαθέσιμη τεκμηρίωση.

ΟΙ ΕΜΜΕΣΕΣ ΚΛΗΣΕΙΣ

ΒΓΑΛΕ ΤΟ ΔΡΟΜΕΑ #BDD0

Εξαφανίζει το δρομέα από την οθόνη.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο AF έχει τυχαίο περιεχόμενο.

ΓΡΑΨΕ ΧΑΡΑΚΤΗΡΑ #BDD3

Τυπώνει έναν χαρακτήρα στην οθόνη.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο A περιέχει τον κωδικό του χαρακτήρα, ο H τη στήλη και ο L τη σειρά που θα τυπωθεί. (Οι συντεταγμένες ξεκινούν από 0).

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF, BC, DE και HL έχουν τυχαίο περιεχόμενο.

ΔΙΑΒΑΣΕ ΧΑΡΑΚΤΗΡΑ **#BDD6**

Διαβάζει έναν χαρακτήρα από την οθόνη.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο H περιέχει τη στήλη και ο L τη σειρά.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Αν βρέθηκε κάποιο αναγνωρίσιμος χαρακτήρας το carry είναι σε λογικό 1 και ο A περιέχει τον κωδικό του, αλλιώς το carry είναι σε λογικό 0 και ο A περιέχει 0.

Σε όλες τις περιπτώσεις οι BC, DE και HL έχουν τυχαίο περιεχόμενο και τα άλλα flags έχουν επηρεαστεί.

ΕΚΤΕΛΕΣΕ ΧΑΡΑΚΤΗΡΑ **#BDD9**

Στέλνει έναν κωδικό (χαρακτήρα ελέγχου) στην οθόνη.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο A περιέχει τον κωδικό του χαρακτήρα που θέλουμε (0÷255).

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF, BC, DE και HL έχουν τυχαίο περιεχόμενο.

ΚΑΝΕ PLOT **#BDDC**

Κάνει plot ένα pixel στην οθόνη.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο DE περιέχει την οριζόντια συντεταγμένη X και ο HL την κατακόρυφη συντεταγμένη Y.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF, BC, DE και HL έχουν τυχαίο περιεχόμενο.

ΕΛΕΓΞΕ PIXEL **#BDDF**

Ελέγχει τι μελάνι έχει ένα pixel της οθόνης.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο DE περιέχει την οριζόντια συντεταγμένη X και HL την κατακόρυφη συντεταγμένη Y.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο A περιέχει το κωδικοποιημένο ink του pixel. Οι BC, DE και HL έχουν τυχαίο περιεχόμενο και τα flags έχουν επηρεαστεί.

ΣΧΕΔΙΑΣΕ ΓΡΑΜΜΗ **#BDE2**

Σχεδιάζει μια γραμμή στην οθόνη από την τρέχουσα θέση του δρομέα γραφικών.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο DE περιέχει την οριζόντια συντεταγμένη X και ο HL την κατακόρυφη συντεγαχμένη Y του σημείου μέχρι το οποίο θα τραβηχτεί η γραμμή.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF, BC, DE και HL έχουν τυχαίο περιεχόμενο.

ΔΙΑΒΑΣΕ PIXEL **#BDE5**

Διαβάζει ένα pixel στην οθόνη και αποκωδικοποιεί το μελάνι του.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο HL περιέχει τη διεύθυνση της Video Ram για το pixel και ο C έναν κωδικό (mask) για το pixel.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο A περιέχει το αποκωδικοποιημένο μελάνι για το pixel. Τα flags έχουν επηρεαστεί.

ΓΡΑΨΕ PIXELS **#BDE8**

Γράψε ένα pixel ή περισσότερα στην οθόνη.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο HL περιέχει τη διεύθυνση της Video Ram για τα pixels, ο C έναν κωδικό (mask) του pixel και ο B το κωδικοποιημένο ink που θα χρειαστεί.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο AF έχει τυχαίο περιεχόμενο.

ΚΑΘΑΡΙΣΕ ΤΗΝ ΟΘΟΝΗ **#BDE8**

Καθαρίζει την οθόνη βάζοντας όλα τα pixels στο ink 0.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF, BC, DE και HL έχουν τυχαίο περιεχόμενο.

ΕΛΕΓΞΕ BREAK **#BDEE**

Ελεγξε για BREAK και RESET, καλώντας την αντίστοιχη ρουτίνα.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Τα interrupts πρέπει να έχουν αποενεργοποι-

ηθεί (με DI). Ο C περιέχει την κατάσταση του SHIFT και CONTROL.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF και HL έχουν τυχαίο περιεχόμενο.

ΠΕΡΙΜΕΝΕ ΤΟΝ PRINTER #BDF1

Περιμένει μέχρι ο Printer να είναι διαθέσιμος (not busy) για να στείλει έναν χαρακτήρα στην έξοδό του. Αν περάσει όμως ορισμένος χρόνος η ρουτίνα επιστέφει χωρίς να στείλει τον χαρακτήρα.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο A περιέχει τον κωδικό του χαρακτήρα που θέλουμε να στείλουμε στον printer.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Το carry είναι σε λογικό 1 αν όλα πήγαν καλά, αλλιώς είναι σε λογικό 0. Οι A και BC έχουν τυχαίο περιεχόμενο.

Εδώ ολοκληρώνονται και οι έμμεσες κλήσεις του λειτουργικού συστήματος. Στη συνέχεια θα εξετάσουμε το jumpblock (πίνακα κλήσεων) του «Υψηλού Πυρήνα» που καλεί ρουτίνες οι οποίες υπάρχουν στη RAM.

Ο ΥΨΗΛΟΣ ΠΥΡΗΝΑΣ

ΕΝΕΡΓΟΠΟΙΗΣΕ ΤΗΝ ΑΝΩ ROM #B900

Ενεργοποιεί την άνω ROM (#C000 ÷ #FFFF).

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο A περιέχει την προηγούμενη κατάσταση της ROM.

ΑΠΟΕΝΕΡΓΟΠΟΙΗΣΕ ΤΗΝ ΑΝΩ ROM #B903

Αποενεργοποιεί την άνω ROM (#C000 ÷ #FFFF).

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο A περιέχει την προηγούμενη κατάσταση της ROM.

ΕΝΕΡΓΟΠΟΙΗΣΕ ΤΗΝ ΚΑΤΩ ROM #B906

Ενεργοποιεί την κάτω ROM (0 ÷ #3FFF).

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο Α περιέχει την προηγούμενη κατάσταση της ROM).

ΑΠΟΕΝΕΡΓΟΠΟΙΗΣΕ ΤΗΝ ΚΑΤΩ ROM **#B909**

Αποενεργοποίησε την κάτω ROM.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο Α περιέχει την προηγούμενη κατάσταση της ROM.

ΕΠΑΝΑΦΕΡΕ ΤΗΝ ROM **#B90C**

Επανάφερε την προηγούμενη κατάσταση ROM.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο Α περιέχει την προηγούμενη κατάσταση της ROM όπως επιστρέφει από τις ρουτίνες που περιγράψαμε παραπάνω.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο AF έχει τυχαίο περιεχόμενο.

ΕΠΙΛΕΞΕ ROM **#B90F**

Επιλέγει μία άνω ROM.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο C περιέχει τη διεύθυνση επιλογής ROM (αριθμό ROM) που θέλουμε.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο C περιέχει την προηγούμενη διεύθυνση επιλογής ROM (αριθμό προηγούμενης ROM), ο Β περιέχει την προηγούμενη κατάσταση της ROM και ο AF έχει τυχαίο περιεχόμενο.

ΕΛΕΓΞΕ ΤΗΝ ROM **#B912**

Ελέγχει την άνω ROM που έχουμε επιλέξει.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο Α περιέχει τη διεύθυνση επιλογής της τρεχουσας ROM (αριθμό ROM).

ΤΑΞΗ ΚΑΙ ΕΚΔΟΣΗ ROM **#B915**

Ελέγχει την τάξη και την έκδοση μιας άνω ROM.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο C περιέχει τη διεύθυνση επιλογής (αριθμό)

της ROM που θέλουμε να ελέγχουμε.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο A περιέχει την τάξη της ROM, ο L τον χαρακτηριστικό αριθμό και ο H τον αριθμό της έκδοσής της.

ΑΠΟΚΑΤΑΣΤΗΣΕ ΠΡΟΗΓΟΥΜΕΝΗ ΕΠΙΛΟΓΗ ROM #B918

Επαναφέρει την προηγούμενη επιλεγμένη άνω ROM.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο B περιέχει την προηγούμενη κατάσταση της ROM και ο C τη διεύθυνση επιλογής της προηγούμενης ROM.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο C περιέχει τη διεύθυνση επιλογής της τρέχουσας ROM και ο B έχει τυχαίο περιεχόμενο.

LDIR #B91B

Αντιγράφει ένα block μνήμης από μια θέση σε μια άλλη (όμοια με την αντίστοιχη εντολή του Z-80). Οι ROMS κατά τη μεταφορά είναι αποενεργοποιημένες. Η μεταφορά γίνεται με αυξανόμενο POINTER.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο BC περιέχει το μήκος του block που θέλουμε να μεταφέρουμε, ο DE την αρχική διεύθυνση του σημείου όπου θα το μεταφέρουμε και ο HL την αρχική διεύθυνση του block που θέλουμε να μεταφέρουμε.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι F, BC, DE και HL έχουν τιμές όπως διαμορφώνονται από την εντολή LDIR.

LDDR #B91E

Όπως και στην LDIR που περιγράψαμε προηγουμένως με τον pointer όμως να μειώνεται (δίνονται οι τελευταίες τιμές των block).

ΕΛΕΓΞΕ EVENT #B921

Ελέγχει αν εκκρεμεί κάποιο event με υψηλότερη προτεραιότητα.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Το carry είναι σε λογικό 1 αν εκκρεμεί κάποιο event υψηλότερης προτεραιότητας. Ο A έχει τυχαίο περιεχόμενο και τα άλλα flags έχουν επηρεαστεί.

Εδώ ολοκληρώθηκε και η παρουσίαση του jumpblock του Υψηλού Πυρήνα. Στη συνέχεια θα παρουσιάσουμε το jumpblock του Χαμηλού

Πυρήνα δηλ. τις κλήσεις του λειτουργικού που βρίσκονται στις θέσεις #00 ÷ #3F της RAM, όπου είναι και η περιοχή των RESTARTS. Έτσι θα ολοκληρώσουμε και το τελευταίο αυτό κεφάλαιο του βιβλίου.

Ο ΧΑΜΗΛΟΣ ΠΥΡΗΝΑΣ

ΠΛΗΡΕΣ RESET ΣΥΣΤΗΜΑΤΟΣ — RST #00

Εκτελεί πλήρες reset και ο υπολογιστής έρχεται στην κατάσταση που βρίσκεται όταν τον πρωτοανοίξουμε.

ΧΑΜΗΛΟ JUMP 1 — RST #08

Πηδάει σε μια ρουτίνα της κάτω ROM ή της RAM, «παίρνοντας» μαζί τη διεύθυνση που θα πηδήξει.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Όλοι οι καταχωρητές και τα flags μεταφέρονται αυτούσια στη ρουτίνα - στόχο.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Όλοι οι καταχωρητές και τα flags είναι όπως τα διαμορφώνουν η ρουτίνα - στόχος. 4 bytes έχουνε γίνει PUSHED στο STACK. Η ROM επανέρχεται στην προηγούμενη κατάσταση.

ΣΗΜΕΙΩΣΗ: Η διεύθυνση της ρουτίνας που θέλουμε να καλέσουμε ακολουθεί την εντολή RST #08 σε δύο bytes που έχουν την εξής σημασία:

Bit 15 14 | A | K | Διεύθυνση | 0

Η διεύθυνση δηλαδή είναι 14-bit και τα «A» και «K» δίνουν την επιθυμητή κατάσταση της άνω και κάτω ROM αντίστοιχα και είναι σε λογικό 1 για αποενεργοποίηση. Τα INTERRUPTS ενεργοποιούνται.

ΧΑΜΗΛΟ JUMP 2 — #0B

Πηδάει σε μία διεύθυνση της κάτω ROM ή της RAM.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Η διεύθυνση που πηδάει και η απαιτούμενη κατάσταση της ROM περιέχονται στον HL (2 bytes όπως στο ΧΑΜΗΛΟ JUMP 1). Όλοι οι καταχωρητές και τα flags περνάνε όπως είναι στη ρουτίνα - στόχο.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Όλοι οι καταχωρητές και τα flags είναι όπως διαμορφώνονται από τη ρουτίνα - στόχο.

JUMP 1 #OE

Ισοδύναμο με την εντολή JP (HL) του Z-80 μόνο που η διεύθυνση αντί του HL βρίσκεται στον BC (σαν JP (BC)).

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο BC περιέχει τη διεύθυνση όπου θα γίνει το jump. Όλοι οι καταχωρητές και τα flags περνάνε στη ρουτίνα - στόχο.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Όλοι οι καταχωρητές και τα flags είναι όπως διαμορφώθηκαν από τη ρουτίνα - στόχο.

ΚΛΗΣΗ SIDEWAYS ROM 1 RST #10

Καλεί μια sideways ROM (παράλληλη ROM) παίρνοντας τη διεύθυνση που θα κληθεί μαζί. Μετά την εντολή RST #10 ακολουθεί η διεύθυνση της ρουτίνας - στόχου της ROM σε δύο bytes που έχουν ως εξής: 15 14

OFF	Διεύθυνση

όπου η διεύθυνση της ρουτίνας της ROM είναι «Διεύθυνση» +#C000 και η διεύθυνση επιλογής της (ο αριθμός τη ROM) δίνεται από την τιμή του "OFF" (0 ÷ 3) συν τον αριθμό επιλογής της τρέχουσας ROM.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Οι καταχωρητές και τα flags περνάνε όπως είναι στη ρουτίνα - στόχο εκτός τον IY (που δείχνει την περιοχή δεδομένων της άνω ROM).

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο IY διατηρείται. Όλοι οι άλλοι καταχωρητές και τα flags είναι όπως διαμορφώνονται από την ρουτίνα - στόχο.

ΚΛΗΣΗ SIDEWAYS ROM 2 #23

Καλεί μια sideways ROM (παράλληλη ROM) με τη διεύθυνση να περιέχεται στον HL. Ο HL περιέχει δύο bytes ακριβώς όπως περιγράφτηκαν στην προηγούμενη ρουτίνα.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο HL περιέχει τα bytes με τη διεύθυνση και την άνω ROM που επιλέγουμε. Όλοι οι καταχωρητές και τα flags περνούν στη ρουτίνα - στόχο εκτός του IY (που δείχνει τη διεύθυνση δεδομένων της άνω ROM).

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο IY διατηρείται. Οι άλλοι καταχωρητές και τα flags είναι όπως διαμορφώνονται από τη ρουτίνα - στόχο.

JUMP 2 **#16**

Κάνει JUMP σε μία θέση μνήμης που περιέχεται στον DE. Λειτουργεί όπως ακριβώς η JP (HL) μόνο που αντί του HL η διεύθυνση υπάρχει στον DE.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο DE περιέχει τη διεύθυνση για το JUMP. Οι καταχωρητές και τα flags περνάνε στη ρουτίνα - στόχο.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Όλοι οι καταχωρητές και τα flags είναι όπως διαμορφώνονται από τη ρουτίνα - στόχο.

ΜΑΚΡΙΝΟ CALL 1 **RST #18**

Καλεί μια υπορουτίνα της RAM ή κάποιας ROM. Η διεύθυνση της ρουτίνας - στόχου πρέπει να βρίσκεται στα 2 επόμενα bytes από την εντολή RST #18 και η απαιτούμενη κατάσταση ROM στο 3ο byte που έχει ως εξής:

BYTE 3: #00 ÷ #FB: Διεύθυνση επιλογής ROM, ενεργοποίησε την άνω, αποενεργοποίησε την κάτω ROM.
#FC: Καμία αλλαγή στην επιλογή ROM, ενεργοποίησε άνω και κάτω ROM.
#FD: Καμία αλλαγή στην επιλογή ROM, ενεργοποίησε άνω, αποενεργοποίησε κάτω ROM.
#FE: Καμία αλλαγή στην επιλογή ROM, αποενεργοποίησε άνω, ενεργοποίησε κάτω ROM.
#FF: Καμία αλλαγή στην επιλογή ROM, αποενεργοποίησε άνω και κάτω ROM.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Όλοι οι καταχωρητές και τα flags περνάνε όπως είναι στη ρουτίνα - στόχο, εκτός του IY (που δείχνει την περιοχή δεδομένων της άνω ROM).

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο IY διατηρείται. Οι άλλοι καταχωρητές και τα flags είναι όπως διαμορφώνονται από τη ρουτίνα - στόχο.

ΜΑΚΡΙΝΟ CALL 2 **#1B**

Καλεί μια ρουτίνα σε κάποια διεύθυνση της RAM ή οποιασδήποτε ROM.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο HL περιέχει τη διεύθυνση της ρουτίνας - στόχου. Ο C περιέχει το byte της κατάστασης ROM όπως ακριβώς περιγράψαμε στην προηγούμενη ρουτίνα. Όλοι οι καταχωρητές και τα

flags περνάνε στη ρουτίνα - στόχο εκτός του ΙΥ (που δείχνει την περιοχή δεδομένων της άνω ROM).

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο ΙΥ διατηρείται. Οι άλλοι καταχωρητές και τα flags είναι όπως διαμορφώνονται από τη ρουτίνα - στόχο.

JUMP 3 **#1E**

Κάνει JUMP στη διεύθυνση που περιέχει ο HL ή JP (HL).

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο HL περιέχει τη διεύθυνση της ρουτίνας - στόχου. Όλοι οι καταχωρητές και τα flags περνάνε στη ρουτίνα -στόχο.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Όλοι οι καταχωρητές και τα flags είναι όπως διαμορφώνονται από τη ρουτίνα - στόχο.

LD A, (HL) ΑΠΟ RAM **RST #20**

Φορτώνει τον Α με το περιεχόμενο της μνήμης που έχει διεύθυνση το περιεχόμενο του HL αλλά πάντα από τη RAM ανεξάρτητα τι κατάσταση ROM έχουμε (ενεργοποιημένη άνω ή κάτω ROM κλπ.).

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο HL περιέχει τη διεύθυνση της RAM της οποίας θέλουμε να διαβάσουμε το περιεχόμενο.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο Α περιέχει την αντίστοιχη τιμή και όλοι οι άλλοι καταχωρητές και τα flags διατηρούνται.

ΜΑΚΡΙΝΟ CALL 3 **#23**

Καλεί μια ρουτίνα στη RAM ή οποιαδήποτε ROM με τον HL να δείχνει μια διεύθυνση της RAM όπου υπάρχουν τα 3 bytes που δίνουν τη διεύθυνση της ρουτίνας - στόχου και την επιλογή ROM όπως περιγράφτηκαν στη ρουτίνα RST #18 ότι ακολουθούν την εντολή.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Ο HL δείχνει τη διεύθυνση της ρουτίνας - στόχου και της επιλογής ROM. Οι καταχωρητές και τα flags περνάνε στη ρουτίνα - στόχο εκτός του ΙΥ (που δείχνει τη διεύθυνση δεδομένων της άνω ROM).

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Ο ΙΥ διατηρείται. Οι άλλοι καταχωρητές και τα flags είναι όπως διαμορφώνονται από τη ρουτίνα - στόχο. Αποκαθίσταται η προηγούμενη κατάσταση ROM και 4 bytes έχουν γίνει PUSHED στο STACK.

FIRMWARE JUMP **RST #28**

Πηδάει στη χαμηλή ROM, σε μια διεύθυνση που βρίσκεται ακριβώς μετά την εντολή RST #28. Η χαμηλή ROM ενεργοποιείται.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Όλοι οι καταχωρητές και τα flags περνάνε στη ρουτίνα - στόχο.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Όλοι οι καταχωρητές και τα flags είναι όπως διαμορφώνονται από τη ρουτίνα - στόχο.

RESTART ΧΡΗΣΤΗ **RST #30**

Οκτώ bytes από το #30 μέχρι και το #37 μπορούν να γραφούν από το χρήστη ώστε να χρησιμοποιήσει αυτό το RESTART όπως θέλει.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Αγνωστες.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Αγνωστες.

ΕΙΣΑΓΩΓΗ INTERRUPTS **RST #38**

Ο Z-80 τρέχει στο interrupt mode 1 και συνεπώς μεταχειρίζεται τα κανονικά interrupts σαν εντολές RST #38.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Όλοι οι καταχωρητές και τα flags διατηρούνται.

ΕΞΩΤΕΡΙΚΑ INTERRUPTS **#3B**

Τα 5 bytes (#B3 ÷ #3F) πρέπει να φορτωθούν από το χρήστη αφού αντιστοιχούν σε εξωτερικά hardware interrupts. Σε αυτή την περίπτωση η χαμηλή ROM αποενεργοποιείται και καλείται η διεύθυνση #3B.

ΣΥΝΘΗΚΕΣ ΕΙΣΟΔΟΥ: Καμία.

ΣΥΝΘΗΚΕΣ ΕΞΟΔΟΥ: Οι AF, BC, DE και HL έχουν τυχαίο περιεχόμενο.

Όσοι θέλουν να εμβαθύνουν περισσότερο στα interrupts του Amstrad μπορούν να μελετήσουν το Firmware Manual του υπολογιστή.

1ο ΠΑΡΑΡΤΗΜΑ

ΟΙ ΧΑΡΑΚΤΗΡΕΣ ΕΛΕΓΧΟΥ

ASCII ΑΠΟΤΕΛΕΣΜΑ

0 Κανένα αποτέλεσμα.
1 Χρειάζεται 1 παράμετρο από 0 ÷ 255. Το σύμβολο (χαρακτήρας που έχει αυτόν τον κωδικό τυπώνεται. Οι χαρακτήρες ελέγχου (0 ÷ 32) τυπώνονται - δεν εκτελούνται.
2 Σβήνει το δρομέα κειμένου.
3 Ανάβει το δρομέα κειμένου.
4 Χρειάζεται μία παράμετρο που δίνει το mode. Αν δηλαδή δώσουμε PRINT CHR$ (4) + CHR$ (1) θα μπει Mode 1.
5 Μία παράμετρος από 0 μέχρι 255, που είναι ο κωδικός ASCII ενός χαρακτήρα που τυπώνεται στην τρέχουσα θέση του δρομέα γραφικών.
6 Ενεργοποιεί την οθόνη κειμένου.
7 Παράγει BEEP.
8 Μετακινεί τον δρομέα ένα χαρακτήρα πίσω.
9 Μετακινεί το δρομέα ένα χαρακτήρα μπροστά.
10 Μετακινεί το δρομέα μία γραμμή κάτω.
11 Μετακινεί το δρομέα μία γραμμή πάνω.
12 Καθαρίζει το τρέχον παράθυρο κειμένου.
13 Μετακινεί το δρομέα στην αρχή της γραμμής.
14 Μία παράμετρος, που δίνει τον αριθμό INK για το PAPER.
15 Μία παράμετρος, που δίνει τον αριθμό INK για το PEN.
16 Σβήνει το χαρακτήρα κάτω από το δρομέα (σαν το CLR).
17 Σβήνει από το αριστερό άκρο του παράθυρου μέχρι τη θέση του τρέχοντος χαρακτήρα.
18 Σβήνει από τον τρέχοντα χαρακτήρα μέχρι το δεξιό άκρο του παράθυρου.
19 Σβήνει από την αρχή του παράθυρου μέχρι την τρέχουσα θέση του δρομέα.
20 Σβήνει από την τρέχουσα θέση του δρομέα μέχρι το τέλος του παράθυρου.
21 «Σβήνει» την οθόνη κειμένου.
22 Χρειάζεται μία παράμετρο 0 ή 1. Το 0 αποενεργοποιεί το mode για OVERPRINT και το 1 το ενεργοποιεί.
23 Χρειάζεται μία παράμετρο που τοποθετεί το mode γραφικών:
1 Mode XOR
2 Mode AND
3 Mode OR
24 Εναλλάσσει τα inks των PAPER και PEN.

25 Χρειάζεται 9 παραμέτρους και είναι ισοδύναμος με την εντολή SYMBOL της Basic. Η πρώτη παράμετρος είναι ο κωδικός του χαρακτήρα που θα επανακαθορίσουμε και οι υπόλοιποι 8 τα 8 bytes όπως και στην εντολή SYMBOL.
26 Είναι ισοδύναμος με την εντολή WINDOW της BASIC που καθορίζει τα παράθυρα. Έχει 4 παραμέτρους. Οι πρώτες δύο καθορίζουν το αριστερό και δεξιό όριο του παράθυρου με τη μικρότερη τιμή να αντιστοιχεί πάντα στο άνω όριο.
27 Κανένα αποτέλεσμα.
28 Βάζει ένα ζευγάρι χρώματα σε κάποιο ink. Χρειάζεται 3 παραμέτρους. Η πρώτη δίνει τον αριθμό του ink και οι επόμενες δύο τα δύο χρώματα.
29 Είναι ισοδύναμο με την εντολή BORDER της BASIC και χρειάζεται δύο παραμέτρους που αντιστοιχούν στα δύο χρώματα.
30 Επιστρέφει το δρομέα στην άνω αριστερά άκρη του παραθύρου που βρισκόμαστε.
31 Είναι ισοδύναμο με την εντολή LOCATE της BASIC, και χρειάζεται δύο παραμέτρους. Η πρώτη αντιστοιχεί στην οριζόντια συντεταγμένη x και η δεύτερη στην κατακόρυφη y. Ο δρομέας τοποθετείται στην αντίστοιχη θέση.

Οι χαρακτήρες ελέγχου που παρουσιάσαμε παραπάνω μπορούν να εκτελέσουν τις ρουτίνες που αντιστοιχούν μέσω της εντολής PRINT της BASIC ή της ρουτίνας #BB5A από κώδικα μηχανής. Η ρουτίνα #BB5D του λειτουργικού συστήματος δεν εκτελεί τους χαρακτήρες ελέγχου, απλώς τους τυπώνει (τον αντίστοιχο χαρακτήρα).

2ο ΠΑΡΑΡΤΗΜΑ

ΕΝΤΟΛΕΣ ΤΟΥ Z-80

MNEMONIC	HEXADECIMAL
ADC A, (HL)	8E
ADC A, (IX+dis)	DD 8E XX
ADC A,(IY+dis)	FD 8E xx
ADC A,A	8F
ADC A,B	88
ADC A,C	89
ADC A,D	8A
ADC A,n	CE XX
ADC A,E	8B
ADC A,H	8C
ADC A,L	8D
ADC HL,BC	ED 4A
ADC HL,DE	ED 5A
ADC HL,HL	ED 6A
ADC HL,SP	ED 7A
ADD A, (HL)	86
ADD A,(IX+dis)	DD 86XX
ADD A,(IY+dis)	FD 86XX
ADD A,A	87
ADD A,B	80
ADD A,C	81
ADD A,D	82
ADD A,n	C6 XX
ADD A,E	83
ADD A,H	84
ADD A,L	85
ADD HL,BC	09
ADD HL,DE	19
ADD HL,HL	29
ADD HL,SP	39
ADD IX,BC	DD 09
ADD IX,DE	DD 19
ADD IX,IX	DD 29
ADD IX,SP	DD 39
ADD IY,BC	FD 09
ADD IY,DE	FD 19
ADD IY,IY	FD 29
ADD IY,SP	FD 39
AND (HL)	A6
AND (IX+dis)	DD A6 XX
AND (IY+dis)	FD A6 XX
AND A	A7
AND B	A0
AND C	A1
AND D	A2
AND n	E6 XX
AND E	A3
AND H	A4
AND L	A5
BIT 0,(HL)	CB 46
BIT 0,(IX+dis)	DD CB XX 46
BIT 0,(IY+dis)	FD CB XX 46
BIT 0,A	CB 47
BIT o,B	CB 40
BIT 0,C	CB 41
BIT 0,D	CB 42
BIT 0,E	CB 43
BIT 0,H	CB 44
BIT 0,L	CB 45
BIT 1,(HL)	CB 4E
BIT 1,(IX+dis)	DD CB XX 4E
BIT 1,(IY+dis)	FD CB XX 4E
BIT 1,A	CB 4F
BIT 1,B	CB 48
BIT 1,C	CB 49
BIT 1,D	CB 4A
BIT 1,E	CB 4B
BIT 1,H	CB 4C
BIT 1,L	CB 4D
BIT 2,(HL)	CB 56
BIT 2,(IX+dis)	DD CB XX 56
BIT 2,(IY+dis)	FD CB XX 56
BIT 2,A	CB 57
BIT 2,B	CB 50
BIT 2,C	CB 51
BIT 2,D	CB 52
BIT 2,E	CB 53
BIT 2,H	CB 54
BIT 2,L	CB 55
BIT 3,(HL)	CB 5E
BIT 3,(IX+dis)	DD CB XX 5E
BIT 3,(IY+dis)	FD CB XX 5E
BIT 3,A	CB 5F
BIT 3,B	CB 58
BIT 3,C	CB 59
BIT 3,D	CB 5A
BIT 3,E	CB 5B
BIT 3,H	CB 5C
BIT 3,L	CB 5D
BIT 4,(HL)	CB 66
BIT 4,(IX+dis)	DD CB XX 66
BIT 4,(IY+dis)	FD CB XX 66
BIT 4,A	CB 67
BIT 4,B	CB 60
BIT 4,C	CB 61
BIT 4,D	CB 62
BIT 4,E	CB 63
BIT 4,H	CB 64
BIT 4,L	CB 65
BIT 5,(HL)	CB 6E
BIT 5,(IX+dis)	DD CB XX 6E
BIT 5,(IY+dis)	FD CB XX 6E
BIT 5,A	CB 6F
BIT 5,B	CB 68
BIT 5,C	CB 69
BIT 5,D	CB 6A
BIT 5,E	CB 6B
BIT 5,H	CB 6C
BIT 5,L	CB 6D
BIT 6,(HL)	CB 76
BIT 6,(IX+dis)	DD CB XX 76
BIT 6,(IY+dis)	FD CB XX 76
BIT 6,A	CB 77
BIT 6,B	CB 70
BIT 6,C	CB 71
BIT 6,D	CB 72
BIT 6,E	CB 73
BIT 6,H	CB 74
BIT 6,L	CB 75
BIT 7,(HL)	CB 7E
BIT 7,(IX+dis)	DD CB XX 7E
BIT 7,(IY+dis)	FD CB XX 7E
BIT 7,A	CB 7F
BIT 7,B	CB 78
BIT 7,C	CB 79
BIT 7,D	CB 7A
BIT 7,E	CB 7B
BIT 7,H	CB 7C
BIT 7,L	CB 7D
CALL ADDR	CD XX XX
CALL C,ADDR	DC XX XX
CALL M,ADDR	FC XX XX
CALL NC,ADDR	D4 XX XX
CALL NZ,ADDR	C4 XX XX
CALL P,ADDR	F4 XX XX
CALL PE,ADDR	EC XX XX
CALL PO,ADDR	E4 XX XX
CALL Z,ADDR	CC XX XX
CCF	3F
CP (HL)	BE
CP (IX+dis)	DD BE XX
CP (IY+dis)	FD BE XX
CP A	BF
CP B	B8
CP C	B9
CP D	BA
CP n	FE XX
CP E	BB
CP H	BC
CP L	BD
CPD	ED A9
CPDR	ED B9
CPI	ED A1
CPIR	ED B1
CPL	2F
DAA	27
DEC (HL)	35
DEC (IX+dis)	DD 35 XX
DEC (IY+dis)	FD 35 XX
DEC A	3D
DEC B	05
DEC BC	0B
DEC C	0D
DEC D	15
DEC DE	1B
DEC E	1D
DEC H	25
DEC HL	2B
DEC IX	DD 2B
DEC IY	FD 2B
DEC L	2D
DEC SP	3B
DI	F3
DJNZ,dis	10 XX
EI	FB
EX (SP) ,HL	E3
EX (SP) ,IX	DD E3
EX (SP) ,IY	FD E3
EX AF,AF'	08
EX DE,HL	EB
EXX	D9
HALT	76
IM 0	ED 46
IM 1	ED 56
IM 2	ED 5E
IN A, (C)	ED 78
IN A,port	DB XX
IN B, (C)	ED 40
IN C, (C)	ED 48
IN D, (C)	ED 50
IN E, (C)	ED 58
IN H, (C)	ED 60
IN L, (C)	ED 68
INC (HL)	34
INC (IX+dis)	DD 34 XX
INC (IY+dis)	FD 34 XX
INC A	3C
INC B	04
INC BC	03
INC C	0C
INC D	14
INC DE	13
INC E	1C
INC H	24
INC HL	23
INC IX	DD 23
INC IY	FD 23
INC L	2C
INC SP	33
IND	ED AA
INDR	ED BA
INI	ED A2
INIR	ED B2
JP (HL)	E9
JP (IX)	DD E9
JP (IY)	FD E9
JP ADDR	C3 XX XX
JP C,ADDR	DA XX XX
JP M,ADDR	FA XX XX

MNEMONIC	HEXADECIMAL
JP NC,ADDR	D2 XX XX
JP NZ,ADDR	C2 XX XX
JP P,ADDR	F2 XX XX
JP PE,ADDR	EA XX XX
JP PO,ADDR	E2 XX XX
JP Z,ADDR	CA XX XX
JR C,dis	38 XX
JR dis	18 XX
JR NC,dis	30 XX
JR NZ,dis	20 XX
JR Z,dis	28 XX
LD (ADDR) ,A	32 XX XX
LD(ADDR) ,BC	ED 43 XX XX
LD (ADDR) ,DE	ED 53 XX XX
LD(ADDR) ,HL	ED 63 XX XX
LD (ADDR) ,HL	22 XX XX
LD (ADDR) ,IX	DD 22 XX XX
LD (ADDR) , IY	FD 22 XX XX
LD (ADDR) ,SP	ED 73 XX XX
LD (BC) ,A	02
LD (DE) ,A	12
LD (HL) ,A	77
LD (HL) ,B	70
LD (HL), C	71
LD (HL) ,D	72
LD (HL) ,n	36 XX
LD (HL) ,E	73
LD (HL) ,H	74
LD (HL) ,L	75
LD (IX+dis) ,A	DD 77 XX
LD (IX+dis) ,B	DD 70 XX
LD (IX+dis) ,C	DD 71 XX
LD (IX+dis) ,D	DD 72 XX
LD (IX+dis) ,n	DD 36 XX XX
LD (IX+dis) ,E	DD 73 XX
LD (IX+dis) ,H	DD 74 XX
LD (IX+dis) ,L	DD 75 XX
LD (IY+dis) ,A	FD 77 XX
LD (IY+dis) ,B	FD 70 XX
LD (IY+dis) ,C	FD 71 XX
LD (IY+dis) ,D	FD 72 XX
LD (IY+dis) ,n	FD 36 XX XX
LD (IY+dis) ,E	FD 73 XX
LD (IY+dis) ,H	FD 74 XX
LD (IY+dis) ,L	FD 75 XX
LD A, (ADDR)	3A XX XX
LD A, (BC)	0A
LD A, (DE)	1A
LD A, (HL)	7E
LD A, (IX+dis)	DD 7E XX
LD A, (IY+dis)	FD 7E XX
LD A,A	7F
LD A,B	78
LD A,C	79
LD A,D	7A
LD A,n	3E XX
LD A,E	7B
LD A,H	7C
LD A,I	ED 57
LD A,L	7D
LD A,R	ED 5F
LD B, (HL)	46
LD B, (IX+dis)	DD 46 XX
LD B, (IY+dis)	FD 46 XX
LD B,A	47
LD B,B	40
LD B,C	41
LD B,D	42
LD B,n	06 XX
LD B,E	43
LD B,H	44
LD B,L	45
LD BC, (ADDR)	ED 4B XX XX
LD BC,nn	01 XX XX
LD C, (HL)	4E
LD C, (IX+dis)	DD 4E xx
LD C, (IY+dis)	FD 4E XX
LD C,A	4F
LD C,B	48
LD C,C	49
LD C,D	4A
LD C,n	0E XX
LD C,E	4B
LD C,H	4C
LD C,L	4D
LD D, (HL)	56
LD D, (IX+dis)	DD 56 XX
LD D, (IY+dis)	FD 56 XX
LD D,A	57
LD D,B	50
LD D,C	51
LD D,D	52
LD D,n	16 XX
LD D,E	53
LD D,H	54
LD D,L	55
LD DE, (ADDR)	ED 5B XX XX
LD DE,nn	11 XX XX
LD E, (HL)	5E
LD E, (IX+dis)	DD 5E XX
LD E, (IY+dis)	FD 5E XX
LD E,A	5F
LD E,B	58
LD E,C	59
LD E,D	5A
LD E,n	1E XX
LD E,E	5B
LD E,H	5C
LD E,L	5D
LD H, (HL)	66
LD H, (IX+dis)	DD 66 XX
LD H, (IY+dis)	FD 66 XX
LD H,A	67
LD H,B	60
LD H,C	61
LD H,D	62
LD H,n	26 XX
LD H,E	63
LD H,H	64
LD H,L	65
LD HL, (ADDR)	ED 6B XX XX
LD HL,(ADDR)	2A XX XX
LD HL,nn	21 XX XX
LD I,A	ED 47
LD IX, (ADDR)	DD 2A XX XX
LD IX,nn	DD 21 XX XX
LD IY (ADDR)	FD 2A XX XX
LD IY,nn	FD 21 XX XX
LD L,A	6F
LD L,B	68
LD L,C	69
LD L,D	6A
LD L,n	2E XX
LD L,E	6B
LD L, (HL)	6E
LD L,(IX+dis)	DD 6E XX
LD L, (IY+dis)	FD 6E XX
LD L,H	6C
LD L,L	6D
LD R,A	ED 4F
LD SP, (ADDR)	ED 7B XX XX
LD SP,nn	31 XX XX
LD SP,HL	F9
LD SP,IX	DD F9
LD SP,IY	FD F9
LDD	ED A8
LDDR	ED B8
LDI	ED A0
LDIR	ED B0
NEG	ED 44
NOP	00
OR (HL)	B6
OR (IX+dis)	DD B6 XX
OR (IY+dis)	FD B6 xx
OR A	B7
OR B	B0
OR C	B1
OR D	B2
OR n	F6 XX
OR E	B3
OR H	B4
OR L	B5
OTDR	ED BB
OTIR	ED B3
OUT (C) ,A	ED 79
OUT (C) ,B	ED 41
OUT (C) ,C	ED 49
OUT (C) ,D	ED 51
OUT (C) ,E	ED 59
OUT (C) ,H	ED 61
OUT (C) ,L	ED 69
OUT port,A	D3 port
OUTD	ED AB
OUTI	ED A3
POP AF	F1
POP BC	C1
POP DE	D1
POP HL	E1
POP IX	DD E1
POP IY	FD E1
PUSH AF	F5
PUSH BC	C5
PUSH DE	D5
PUSH HL	E5
PUSH IX	DD E5
PUSH IY	FD E5
RES 0, (HL)	CB 86
RES 0, (IX+dis)	DD CB XX 86
RES 0, (IY+dis)	FD CB XX 86
RES 0,A	CB 87
RES 0,B	CB 80
RES 0,C	CB 81
RES 0,D	CB 82
RES 0,E	CB 83
RES 0,H	CB 84
RES 0,L	CB 85
RES 1, (HL)	CB 8E
RES 1, (IX+dis)	DD CB XX 8E
RES 1, (IY+dis)	FD CB XX 8E
RES 1,A	CB 8F
RES 1,B	CB 88
RES 1,C	CB 89
RES 1,D	CB 8A
RES 1,E	CB 8B
RES 1,H	CB 8C
RES 1,L	CB 8D
RES 2, (HL)	CB 96
RES 2, (IX+dis)	DD CB XX 96
RES 2, (IY+dis)	FD CB XX 96
RES 2,A	CB 97
RES 2,B	CB 90
RES 2,C	CB 91
RES 2,D	CB 92
RES 2,E	CB 93
RES 2,H	CB 94
RES 2,L	CB 95
RES 3, (HL)	CB 9E
RES 3, (IX+dis)	DD CB XX 9E
RES 3, (IY+dis)	FD CB XX 9E
RES 3,A	CB 9F

MNEMONIC	HEXADECIMAL
RES 3,B	CB 98
RES 3,C	CB 99
RES 3,D	CB 9A
RES 3,E	CB 9B
RES 3,H	CB 9C
RES 3,L	CB 9D
RES 4, (HL)	CB A6
RES 4, (IX+dis)	DD CB XX A6
RES 4, (IY+dis)	FD CB XX A6
RES 4,A	CB A7
RES 4,B	CB A0
RES 4,C	CB A1
RES 4,D	CB A2
RES 4,E	CB A3
RES 4,H	CB A4
RES 4,L	CB A5
RES 5 (HL)	CB AE
RES 5, (IX+dis)	DD CB XX AE
RES 5, (IY+dis)	FD CB XX AE
RES 5,A	CB AF
RES 5,B	CB A8
RES 5,C	CB A9
RES 5,D	CB AA
RES 5,E	CB AB
RES 5,H	CB AC
RES 5,L	CB AD
RES 6, (HL)	CB B6
RES 6, (IX+dis)	DD CB XX B6
RES 6, (IY+dis)	FD CB XX B6
RES 6,A	CB B7
RES 6,B	CB B0
RES 6,C	CB B1
RES 6,D	CB B2
RES 6,E	CB B3
RES 6,H	CB B4
RES 6,L	CB B5
RES 7, (HL)	CB BE
RES 7, (IX+dis)	DD CB XX BE
RES 7, (IY+dis)	FD CB XX BE
RES 7,A	CB BF
RES 7,B	CB B8
RES 7,C	CB B9
RES 7,D	CB BA
RES 7,E	CB BB
RES 7,H	CB BC
RES 7,L	CB BD
RET	C9
RET C	D8
RET M	F8
RET NC	D0
RET NZ	C0
RET P	F0
RET PE	E8
RET PO	E0
RET Z	C8
RETI	ED 4D
RETN	ED 45
RL (HL)	CB 16
RL (IX+dis)	DD CB XX 16
RL (IY+dis)	FD CB XX 16
RL A	CB 17
RL B	CB 10
RL C	CB 11
RL D	CB 12
RL E	CB 13
RL H	CB 14
RL L	CB 15
RLA	17
RLC (HL)	CB 06
RLC (IX+dis)	DD CB XX 06
RLC (IY+dis)	FD CB XX 06
RLC A	CB 07
RLC B	CB 00

MNEMONIC	HEXADECIMAL
RLC C	CB 01
RLC D	CB 02
RLC E	CB 03
RLC H	CB 04
RLC L	CB 05
RLCA	07
RLD	ED 6F
RR (HL)	CB 1E
RR (IX+dis)	DD CB XX 1E
RR (IY+dis)	FD CB XX 1E
RR A	CB 1F
RR B	CB 18
RR C	CB 19
RR D	CB 1A
RR E	CB 1B
RR H	CB 1C
RR L	CB 1D
RRA	1F
RRC (HL)	CB 0E
RRC (IX+dis)	DD CB XX 0E
RRC (IY+dis)	FD CB XX 0E
RRC A	CB 0F
RRC B	CB 08
RRC C	CB 09
RRC D	CB 0A
RRC E	CH 0B
RRC H	CB 0C
RRC L	CB 0D
RRCA	0F
RRD	ED 67
RST 00	C7
RST 08	CF
RST 10	D7
RST 18	DF
RST 20	E7
RST 28	EF
RST 30	F7
RST 38	FF
SBC A, (HL)	9E
SBC A, (IX+dis)	DD 9E XX
SBC A, (IY+dis)	FD 9E XX
SBC A,A	9F
SBC A,B	98
SBC A,C	99
SBC A,D	9A
SBC A,n	DE XX
SBC A,E	9B
SBC A,H	9C
SBC A,L	9D
SBC HL,BC	ED 42
SBC HL,DE	ED 52
SBC HL,HL	ED 62
SBC HL,SP	ED 72
SCF	37
SET 0, (HL)	CB C6
SET 0, (IX+dis)	DD CB XX C6
SET 0, (IY+dis)	FD CB XX C6
SET 0,A	CB C7
SET 0,B	CB C0
SET 0,C	CB C1
SET 0,D	CB C2
SET 0,E	CB C3
SET 0,H	CB C4
SET 0,L	CB C5
SET 1, (HL)	CB CE
SET 1, (IX+dis)	DD CB XX CE
SET 1, (IY+dis)	FD CB XX CE
SET 1,A	CB CF
SET 1,B	CB C8
SET 1,C	CB C9
SET 1,D	CB CA
SET 1,E	CB CB
SET 1,H	CB CC

MNEMONIC	HEXADECIMAL
SET 1,L	CB CD
SET 2, (HL)	CB D6
SET 2, (IX+dis)	DD CB XX D6
SET 2, (IY+dis)	FD CB XX D6
SET 2,A	CB D7
SET 2,B	CB D0
SET 2,C	CB D1
SET 2,D	CB D2
SET 2,E	CB D3
SET 2,H	CB D4
SET 2,L	CB D5
SET 3, (HL)	CB DE
SET 3, (IX+dis)	DD CB XX DE
SET 3, (IY+dis)	FD CB XX DE
SET 3,A	CB DF
SET 3,B	CB D8
SET 3,C	CB D9
SET 3,D	CB DA
SET 3,E	CB DB
SET 3,H	CB DC
SET 3,L	CB DD
SET 4, (HL)	CBE6
SET 4, (IX+dis)	DD CB XX E6
SET 4, (IY+dis)	FD CB XX E6
SET 4,A	CB E7
SET 4,B	CB E0
SET 4,C	CB E1
SET 4,D	CB E2
SET 4,E	CB E3
SET 4,H	CB E4
SET 4,L	CB E5
SET 5, (HL)	CB EE
SET 5, (IX+dis)	DD CB XX EE
SET 5, (IY+dis)	FD CB XX EE
SET 5,A	CB EF
SET 5,B	CB E8
SET 5,C	CB E9
SET 5,D	CB EA
SET 5,E	CB EB
SET 5,H	CB EC
SET 5,L	CB ED
SET 6, (HL)	CB F6
SET 6, (IX+dis)	DD CB XX F6
SET 6, (IY+dis)	FD CB XX F6
SET 6,A	CB F7
SET 6,B	CB F0
SET 6,C	CB F1
SET 6,D	CB F2
SET 6,E	CB F3
SET 6,H	CB F4
SET 6,L	CB F5
SET 7, (HL)	CB FE
SET 7, (IX+dis)	DD CB XX FE
SET 7,(IY+dis)	FD CB XX FE
SET 7,A	CB FF
SET 7,B	CB F8
SET 7,C	CB F9
SET 7,D	CB FA
SET 7,E	CB FB
SET 7,H	CB FC
SET 7,L	CB FD
SLA (HL)	CB 26
SLA (IX+dis)	DD CB XX 26
SLA (IY+dis)	FD CB XX 26
SLA A	CB 27
SLA B	CB 20
SLA C	CB 21
SLA D	CB 22
SLA E	CB 23
SLA H	CB 24
SLA L	CB 25
SRA (HL)	CB 2E
SRA (IX+dis)	DD CB XX 2E

MNEMONIC	HEXADECIMAL	MNEMONIC	HEXADECIMAL	MNEMONIC	HEXADECIMAL
SRA (IY+dis)	FD CB XX 2E				
SRA A	CB 2F				
SRA B	CB 28				
SRA C	CB 29				
SRA D	CB 2A				
SRA E	CB 2B				
SRA H	CB 2C				
SRA L	CB 2D				
SRL (HL)	CB 3E				
SRL (IX+dis)	DD CB XX 3E				
SRL (IY+dis)	FD CB XX 3E				
SRL A	CB 3F				
SRL B	CB 38				
SRL C	CB 39				
SRL D	CB 3A				
SRL E	CB 3B				
SRL H	CB 3C				
SRL L	CB 3D				
SUB (HL)	96				
SUB (IX+dis)	DD 96 XX				
SUB (IY+dis)	FD 96 XX				
SUB A	97				
SUB B	90				
SUB C	91				
SUB D	92				
SUB E	93				
SUB n	D6 XX				
SUB H	94				
SUB L	95				
XOR (HL)	AE				
XOR (IX+dis)	DD AE XX				
XOR (IY+dis)	FD AE XX				
XOR A	AF				
XOR B	A8				
XOR C	A9				
XOR D	AA				
XOR n	EE XX				
XOR E	AB				
XSOR H	AC				
XOR L	AD				

ΦΩΤΗΣ ΓΕΩΡΓΙΑΔΗΣ

AMSTRAD

ΧΙΛΙΕΣ ΚΑΙ ΜΙΑ ΔΥΝΑΤΟΤΗΤΕΣ

Β΄ ΕΚΔΟΣΗ

- ΕΤΟΙΜΕΣ ΡΟΥΤΙΝΕΣ ΚΩΔΙΚΑ ΜΗΧΑΝΗΣ
- ΡΟΥΤΙΝΑ ΑΝΤΙΓΡΑΦΗΣ
- RANDOM ACCESS
- ΟΙ ΚΛΗΣΕΙΣ ΤΟΥ ΛΕΙΤΟΥΡΓΙΚΟΥ

για CPC 464/664/6128

ΕΚΔΟΣΕΙΣ COMPUPRESS

ΦΩΤΗΣ ΓΕΩΡΓΙΑΔΗΣ
ΝΑΤΟΤΗΤΕΣ
MSTI

AMSTRAD CPC

MÉMOIRE ÉCRITE
MEMORY ENGRAVED
MEMORIA ESCRITA

https://acpc.me/