- RSX - BOX -

Adresse	n° Ligne	Code	Label	Code
8CA0	10 (30000)			ORG 40000
9C40	20 =(40000)		BUFF:	DEFS &04
9C44 = 4000H	30	014E9C	réserve 4 octets pour A/B/C/X - 1 2 3 4	LD BC, EXCOMT:
9C47	40	21409C		LD HL, BUFF:
9C4A	50	CDD1BC		CALL &BCD1 -
9C4D	60	C9		RET
9C4E	70	539C	EXCOMT:	DEFW NENAM:
9C50	80	C3579C		JP BOX:
9C53	90	424F	NENAM:	DEFM "Bo"
9C55	100	D8		DEFB "X" + &80
9C56	110	00		DEFB &0
9C57	120		DRAW:	EQU &BBFG
9C57	130		PLOT:	EQU &BBEA
9C57	140		INK:	EQU &BBDE
9C57	150	DD7E00	BOX:	LD A, (IX + 0)
9C5A	160	CDDEBB		CALL INK:
9C5D	170	DD5E08		LD E, (IX + 8)
9C60	180	DD5609		LD D, (IX + 9)
9C63	190	D5		PUSH DE
9C64	200	DD5E06		LD E, (IX + 6)

Adresse	n° ligne	Code	Label	Code
9CG7	210	DD5607		LD D,(IX+7)
9CGA	220	D5		PUSH DE
9CGB	230	DD5E04		LD E,(IX+4)
9CGE	240	DD5605		LD D,(IX+5)
9C71	250	D5		PUSH DE
9C72	260	DD5E02		LD E,(IX+2)
9C75	270	DD5603		LD D,(IX+3)
9C78	280	D5		PUSH DE
9C79	290	DD5E08		LD E,(IX+8)
9C7C	300	DD5609		LD D,(IX+9)
9C7F	310	DD6E06		LD L,(IX+6)
9C82	320	DD6607		LD H,(IX+7)
9C85	330	D5		PUSH DE
9C86	340	CDEABB		CALL PLOT:
9C89	350	D1		POP DE
9C8A	360	E1.		POP HL
9C8B	370	E5		PUSH HL
9C8C	380	CDFGBB		CALL DRAW:
9C8F	390	E1-		POP HL
9C90	400	D1		POP DE
9C91	410	D5		PUSH DE

Adresse	n°Ligne	Code	Label	Code
9C92	420	CDFGBB		CALL DRAW:
9C95	430	D1		POP DE
9C96	440	E1		POP HL
9C97	450	E5		PUSH HL
9C98	460	CDFGBB		CALL DRAW:
9C9B	470	E1		POP HL
9C9C	480	D1		POP DE
9C9D	490	CDFGBB		CALL DRAW:
9CA0	500	C9		RET

1) La pseudo Instruction ORG met le compteur Adresse à 40000 - BUFF : DEF reserve 4 Octets à partir de 40000 / La Routine à Go indique au SED que l'instruction existe - (System Exploitation Disque)

BC = adresse mémoire de départ de la Table d'instructions Externe

HL = " " de départ du BUFFER de 4 Octets.

& BCD1 = Adresse d'appel -

2) L'adresse de la Table d'instruction est donnée par DEFW NENAM une adresse de saut à la n^ele instruction : JP BOX :

3) Nom de la RSX entrée comme chaine Ascii à NENAM, par DEFM "BO", le dernier caractère "X", est entré avec le bit 7 mis (=1) en soustrait & 80 (DEFB) La Fin de la Table est indiquée par DEFB &0 -

DEFS & 04 reserve 4 octets (pour ü/B/O/X)
ce que fait le Z80 - avec Call & BCDH - Mais
ne connait le nom (BOX) encore/-
DEFW. Définie un mot Binaire sur 2 octet soit
NEWGN : qui sera "BO"
DEFM définie une Chaine "ASCII" (NENAM) soit
"BO" - (B = &42 O = &4F).
DEFB définie 1 octet soit "X" = &58 (Code Asc11)
+ & 80 (=128 DEC/2^{7} Binaire) = &D8 / pour que le
bit 7 soit mis, pour le SED.
DEFB Ø fin de table d'instruction Externe

Guide utilisateur Ch. 7 p46

- MÉMOIRES - ROM -

128 Ko de RAM + 48 Ko de ROM -

- Les 1er 64 Ko de RAM divisés en 4 blocs de 16 Ko n° 0 à n° 3 -

+ Bloc 3 = Mémoire Écran de &C000 à &FFFF soit de 49152 à 65535 -

Bloc 2 = Zone de données des µ programmes + Zone de Branchements + Zone de données Basic + Zone de données des ROMS d'expension, dans la partie haute de ce bloc 2 / Juste au dessus de la limite HIMEM se trouve les caractères définis par l'utilisateur + Donc sans modification de HIMEM + 1 que l'ordinateur donne avec PRINT HIMEM 42620 = ~~42620~~ 42619 ; à 49152 soit de &~~A67C~~ A678 à &BFFF -

La limite HIMEM est modifiable par MEMORY

Au départ les caractères définis par l'utilisateur se trouvent juste au dessus de la limite HIMEM (42619), celle ci est modifiable par MEMORY, elle est automatiquement de 4 Ko lors de l'ouverture de fichiers AMSDOS pour créer une mémoire tampon + Le nombre de caractères définis par l'utilisateur qui si aucun changement du jeu de caractères n'est intervenu depuis sa détermination précédente (à moins qu'une commande SYMBOL AFTER 256 n'ait fixé "les de caractères définis par l'utilisateur") + Lors du lancement les caractères définis par l'utilisateur sont fixés comme si SYMBOL AFTER 240 avait été lancé + Il est donc prudent avant de modifier HIMEM définitivement, de libérer la zone de caractères, puis de replacer cette zone (de caractères) dans sa nouvelle position + Les programmes suivants pourront

ainsi modifier l'affectation faite par la commande SYMBOL AFTER _

Guide utilisateur Ch. 8 p 2.

Bloc 1. de 16384 à ~~32767~~ 32767

soit & 4000 à & 7FFF.

Bloc central du programme Basic.

- Les adresses des ports d'Entrée et Sortie, sont reservées par l'ordinateur, - les adresses inférieures à & 7FFF (donc le Bloc 1) sont absolument prohibées (ou RSX ?)

- & 7F00 couleurs (PEN) (Bible p 37) etc -

Bloc 0 (& 0000 à & 3FFF ou 16384) contient le bloc JUMP qui est copié de la ROM dans la RAM, Bloc 1 - Bank 0 (& 4000 à 7FFF = RAM écran / Bloc 2 = BIOS + BDOS + bloc jump (CPM 2) / Bank 2 = 0 et 2 = TPA / Bank 2 & 4000 à 7FFF = CCP tables Resh CP/M - (Bible p 55).

Bloc 0 - Basic

~

Entrées/Sorties supplementaires. Les adresses des ports d'Entrée/Sorties sont, pour la plupart réservées par l'ordinateur.

Les adresses inférieures à & 7FFF (32767) notamment sont absolument prohibées, les lignes A0 à A7 (160 à 167) seront affectées à la désignation du type de périphériques. Les lignes A8 à A9 (168 à 169) à la sélection de l'un des registres du périphérique sollicité. Parmi les lignes restantes seul A10 (170) doit être décodé (état bas) alors que les lignes A11 à A15 (2577 à 2581) doivent être à l'état haut.

Au registre de chaque périphérique seront ainsi affectées des adresses du type & F8 ?? / & F9 ?? / & FA ?? / et & FB ?? / ou les 2 ? désigneront soit une interface de communication (de DC à DF) soit & F8DC à & FBDF (~~63708~~ 63708 à 64479) ou tout autre type de périphériques de F8E0 à FBFE (63712 à 64510)

Langage Machine (MA).

RAM de 0 à 368 soit &0000 à &0170
utilisé par le système
de 369 à 43903 soit &0171 à &~~AB7F~~ A67B –
pour les programmes BASIC
de 42619 à 49151 soit &A67B à &BFFF
~~de 49152~~ de 42620 à 42740 (&A67C / &A6F4) – les caractères (SYMBOL AFTER 240)
CHR$ (240) peut être aussi de 49151 à 50436
(soit &C504) ? utilisé par le système –
de 49152 à 65535 soit &C0000 à &FFFF
Mémoire écran –

ROM de 0 à 16383 (soit &0000 à &3FFF)
Système d'exploitation –
de 49152 à 65535 (soit &C000 à &FFFF)
Basic –
de 14336 à 16383. (soit & 3800 à &3FFF
La mémoire des caractères –
en 58248 soit &E388 –
La table des mots instructions BASIC

Livre du Basic (M4) -

RAM de 368 à 43903 soit &170 à &AB7F
- stockage des programmes Basic et des variables

ROM de 0 à 16383 soit &0000 à &3FFF
système d'exploitation -

ROM de 49152 à 65535 soit &C000 à &FFFF
Bloc - l'interpréteur Basic et la ROM disquette

~ Les ROM ~

- La ROM inférieure - 16 Ko -
de 0000 à &3FFF -
soit 0 à 16383 -

Contient :

- Le système d'Exploitation, dont
Les routines Systèmes groupées par
par Pack de gestion -
- Le Pack KERNAL - KL -
de 0 à 0057 soit.
Points d'Entrée : B900 à BACC (ROM) + BD5B
de 005C à 0586 soit
Points d'Entrée : B82D à B8D9 (RAM - Syst)
- Le Pack MACHINE ou MC - ou
Interfaçage avec le matériel -
de 0591 à 08BA - soit 1425 à 2234 -
Vecteurs Points d'Entrée : BD13 à BD34 + BD58 (ROM)
+ BDF1a
Le Pack JUMPRESTORE - ou JRE
ou Bloc de Saut -
de 08BD à 08DD - 2237 à 2269 -
Vecteurs. Points d'Entrée BD37 (ROM)
- Indirections - (Bible CPC - p 129 et 140).

MAIN - JUMP ADRESS -

de 08DE à 0910 soit 2270 à 2320 BG28 à BG91 (RAM)
dw de KM. BB00 à BB4B (ROM) + BD3A BD3B/BDF4
de 0912 à 0958 soit 2322 2392
dw de TXT BB4E à BBB7 + BD40 (ROM) / BGB5 à B763 (RAM).
de 095A à 0986 soit 2394 a 2438 - BG93 à BGA6 (RAM)
dw de GRA BBBA à BBFC. / BD43 à BD52 / BDDC à BDE2 (ROM).
de 09B8 à 09CA soit 2440 à 2506
dw de SCR - BBFF à BCG2 + BD55 (ROM) / B7C3 à B7C9 (RAM).
de 09CC à 09F6 soit 2508 à 2550
dw de CAS BCG5 à BCA4 (ROM) / B118 à B1E9 (RAM)
de 09F8 à 0A0C soit 2552 à 2572
dw de SOUND BCA7 à BCC5 (ROM) / B1ED à B396 (RAM -
de 0A0E à 0A3E soit 2574 à 2622
dw de KL / BCC8 à BD10 (ROM) B82D à B8D9 (RAM).
de 0A40 à 0A5G soit 2624 à 2646
dw de MC - BD13 à BD34 + BD58 (ROM)
0A58 JUMP RESTORE soit 2648 -
dw de JUMP - BD37 (ROM)
de 0A5A à 0A5C soit 2650 à 2652
dw de KM - BB00 à BB4B + BD3A + BD3B (ROM) / BG28 à BG91 (RAM).
de 0A5E 2654 - BB4E à BBB7 (ROM)
dw de TXT BGB5 à B763 (RAM -

de 0A60 à 0A6A soit 2656 à 2665
DW de GRA - idem - ROM 095A / BG93 à BGA7 (RAM -
de 0A6C soit 2668
DW de SCR idem D988 ROM / idem - (RAM -
de 0A6E soit 2670
DW de MC idem
de 0A70 soit 2672
DW de KL - idem

BASIC - JUMP ADRESS -

de 0A72 à 0AB2 soit 2674 à 2740
DW de FLO. il s'agit des adresses REELLES -

- Pack SCREEN - SCR - ÉCRAN

de 0ABF à 1072 soit 2750 à 4210
Points d'Entrée : BBFF à BCG2

- Pack TEXT SCREEN - TXT - TEXTE

de 1074 à 15A5 4212 à 5541.
Points d'Entrée - BB4E à BBB7

Pack GRAPHIQUE - GRA

de 15A8 à 1B5G 5448 à G998
Points d'Entrée BBBA à BBFC

Pack CLAVIER - KM -

de 1B5C à 1FE7 soit 7004 à 8167
Points d'Entrée BB00 à BB4B -

Pack SOUND - SONORE

de 1FE9 à 2AE2 soit 8169 à 9390

Points d'Entrée -

Pack CASSETTE - CAS -

de 24BC à 2BE9 soit 9404 à 11241

Points d'Entrée:

EDITEUR de SAUT - EDIT

de 2C02 à 2F70 soit 11266 à 12144

Edit. Table de Saut 1 et 2 -

de 2C02 à 2CCB. 11266 à 11467

Points d'Entrée:

ESC - CRSR + CTRL + TAB + SHIFT + COPY + Caractères

Clavier et divers -

de 2CD0 à 2F70 soit 11472 à 12144

Points d'Entrée.

- Arithmétiques - ROUTINES MATH -

de 2F73 à 37FF - à 12147 à 14335

Points d'Entrée

- Vecteurs Basic -

GENERATEUR de CARACTERES

de 3800 à 3FFF soit. 14336 à 16383.

Points d'Entrée.

PROCESSEUR Z 80

UNITÉ CENTRALE

1. CU. Unité de contrôle : contrôle et commande toutes les opérations se déroulant dans l'ordinateur
2. BUS de contrôle : "bras long de la CU" pilote et surveille les composants extérieurs à l'unité centrale.
3. SP. Pointeur de pile (Stack Pointer) Registre 16 bits. Stockage provisoire dans la RAM de données et d'adresse de retour des sous programme
4. PC. Pointeur de programme. Registre 16 bits. Indique l'adresse de la mémoire où figure l'instruction à traiter à un moment donné
5. Registres B, C, D, E, H, L. 8 bits. Stockent des données.
6. FLAGS. Registre F 8 bits. Indicateurs de calcul.
7. BUS D'ADRESSE (hors unité centrale) Lien avec autres processeurs. 16 bits. Indique la case mémoire de ROM ou RAM qui doit être lu ou modifié
8. BUS de DONNEES (en dehors unité centrale) 8 bits. Donnent les données lues ou à écrire.
9. ACCU. Registre A. 8 bits. Registre de calcul (le plus important).
10. ALU. Unité arithmétique et logique. Unité de calcul et logique travaille en fonction du registre FLAGS.
11. DÉCALEUR. Unité décalage. Opérations de rotation et décalage.

REGISTRES de l'unité centrale.

- A (ACCU) 8 bits. Instructions arithmétiques et logiques et comparaison
- F. (FLAGS) 8 bits. Ses bits sont utilisés comme indicateurs pour

UNITÉ CENTRALE

Hors Z80

BUS de DONNÉES de 8 bits.

L'ALU (unité calcul) : S Z H P/V N C flag

7 6 5 4 3 2 1 0 n° du bit

C = retenue / N = soustraction / P/V = Parité - Dépassement

H = Demi-retenue / Z = 0 / S = Signe

FLAG C - Il est mis si une retenue se produit lors d'une addition ou d'une soustraction.

FLAGS N et H interne au Z80

FLAGS P/V : Le V est mis lorsqu'il y a Overflow (dépassement)

P indique la parité de l'octet. (parité = égalité ? ou bit ?)

FLAG Z : Il est mis si le résultat d'une soustraction est 0. Également pour une comparaison lorsqu'il y a identité

FLAG S - Il est mis si le résultat d'une addition ou d'une soustraction est supérieur à 127. (Les octets > 127 sont des nombres négatifs).

Registres B/C = 16 bits - octet fort dans B - Compteur de boucle par ex.

Registres H/L = 16 bits - octet fort dans H - Stockage d'adresse

Registres D/E = 16 bits - octet fort dans D - Stockage intermédiaire d'adresses ou données

Registre SP - 16 bits - Il indique en permanence dans quelle adresse de la mémoire figurent les adresses de retour ou les données stockées provisoirement. Cette adresse désigne une case mémoire qui se trouve dans une zone de la RAM appelée Pile. Le stockage des données s'effectuent ainsi sur la PILE:

A la mise sous tension le SP est fixé à la plus haute adresse de la pile (&C000)

Si un octet doit être placé sur la pile, le SP est diminué de 1 et l'octet est stocké dans l'adresse indiquée par le SP. Il indique toujours la dernière entrée sur la pile. Si l'on retire des données de la Pile, l'octet figurant à l'adresse indiquée par le SP est lu puis le SP est augmenté de 1.

Registre PC 16 bits - non modifiable

Registres IX et IY - Stockage d'adresses ou d'adresses relatives (16 bits) index

Registre I - d'interruption - 8 bits. Contient la partie supérieure de l'adresse à laquelle le programme doit sauter. La partie inférieure est fournie par le composant de l'ordinateur qui a provoqué l'interruption.

Registre R (refresh = rafraîchir) - 8 bits. Utilisé par l'électronique comme compteur pour rafraîchir régulièrement le contenu des mémoires dynamiques Evitant que des infos stockées ne soient perdues. Par le changement renouvelé en très peu de temps du même contenu de mémoire, on évite des pertes de données

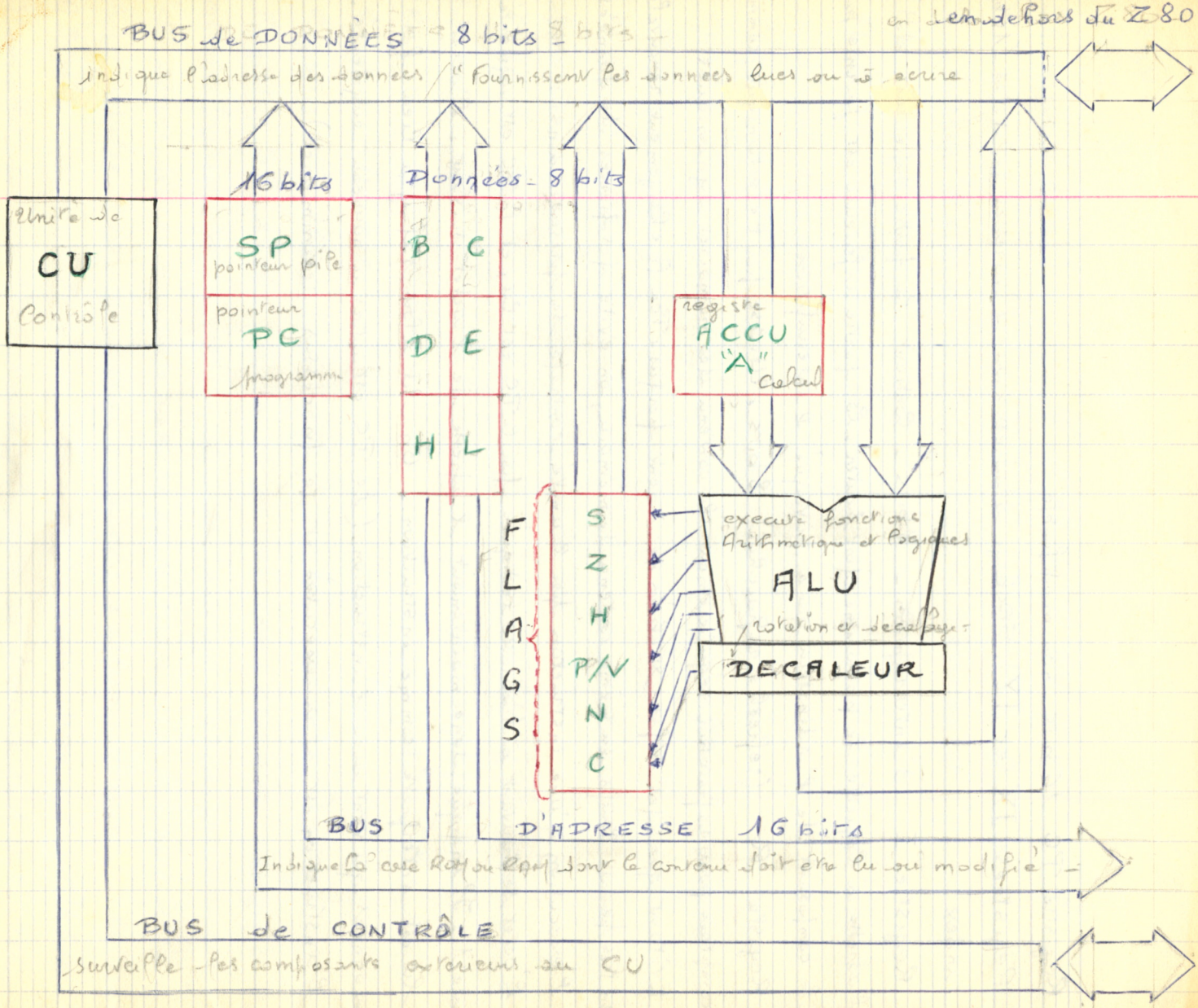
L'exécution d'une instruction par l'unité centrale se présente ainsi :

L'octet se trouvant à l'adresse indiquée par le PC est lu, et le PC est augmenté de 1. Il indique donc maintenant l'adresse de l'octet suivant. L'octet lu est interprété comme instruction. On sait alors s'il y a lieu les données qui vont avec l'instruction (Le PC est alors à nouveau augmenté. L'instruction est alors exécutée et le processus recommence).

(registre F) (8 bits)

D'ADRESSE 16 bits

Hors Z80


BUS de DONNÉES 8 bits
en dehors du Z80
indique l'adresse des données / Fournissent les données lues ou à écrire
16 bits
Données - 8 bits
CU
Contrôle
SP
pointeur pile
PC
pointeur programme
B C
D E
H L
ACCU
"A"
calcul
F
L
A
G
S
S
Z
H
P/V
N
C
exécute fonctions Arithmétiques et Logiques
ALU
DECALEUR
BUS D'ADRESSE 16 bits
Indique la case RAM ou ROM dont le contenu doit être lu ou modifié
BUS de CONTRÔLE
surveille les composants extérieurs au CU

Bible du CPC (MA).

MEMOIRES (suite)

ROM

Parallelement à la RAM se trouve une moitié de la ROM dans les 16 Ko inferieurs de 0 à 16383 soit de &0000 à &3FFF

ROM

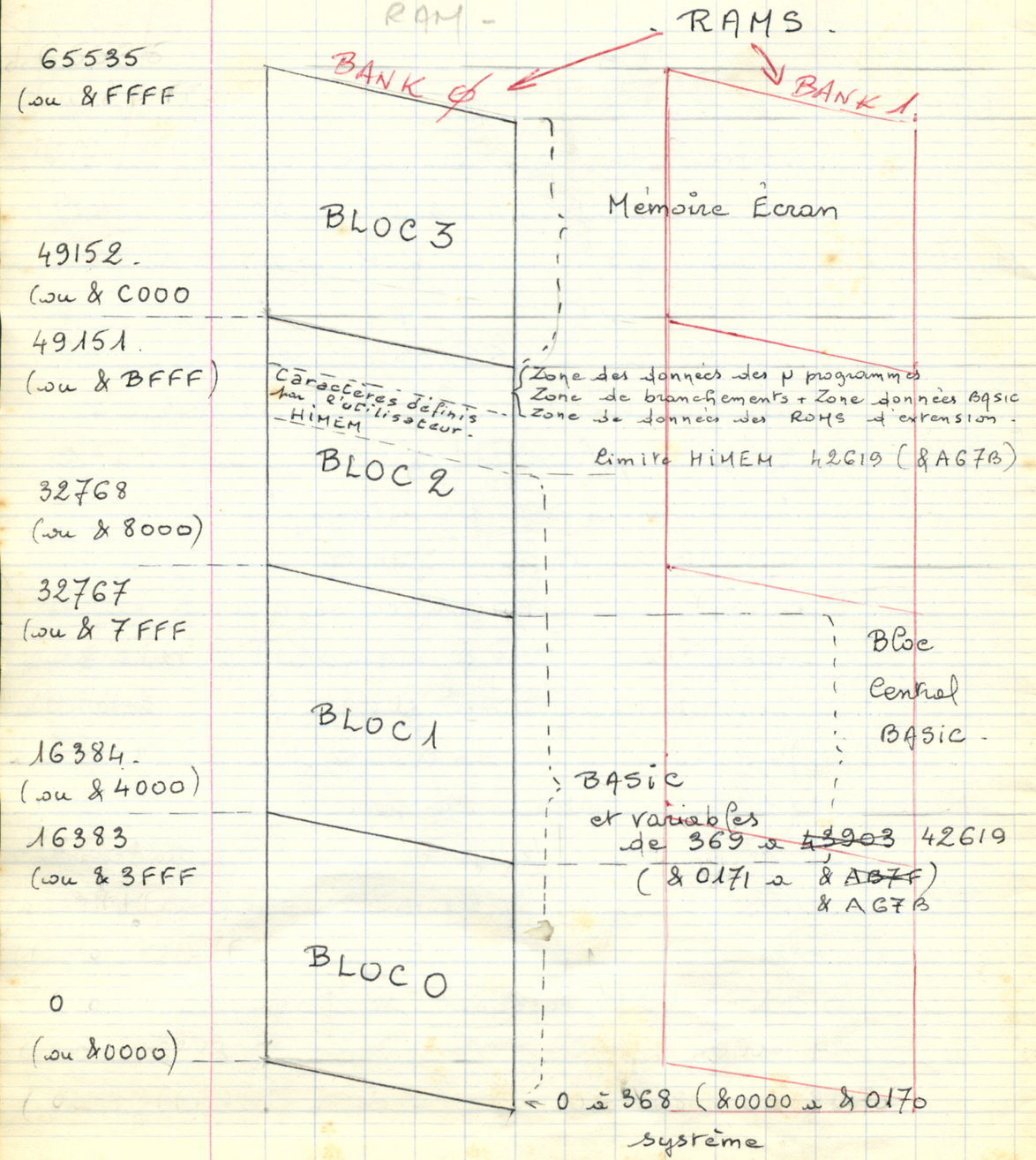
L'autre moitié de la ROM dans les 16 Ko superieurs soit:
de 49152 à 65535 soit &C000 à &FFFF
ainsi que sur toutes la zone d'adresses la banque supplementaire de 64 Ko de RAM (cette derniere n'est pas adressable directement)

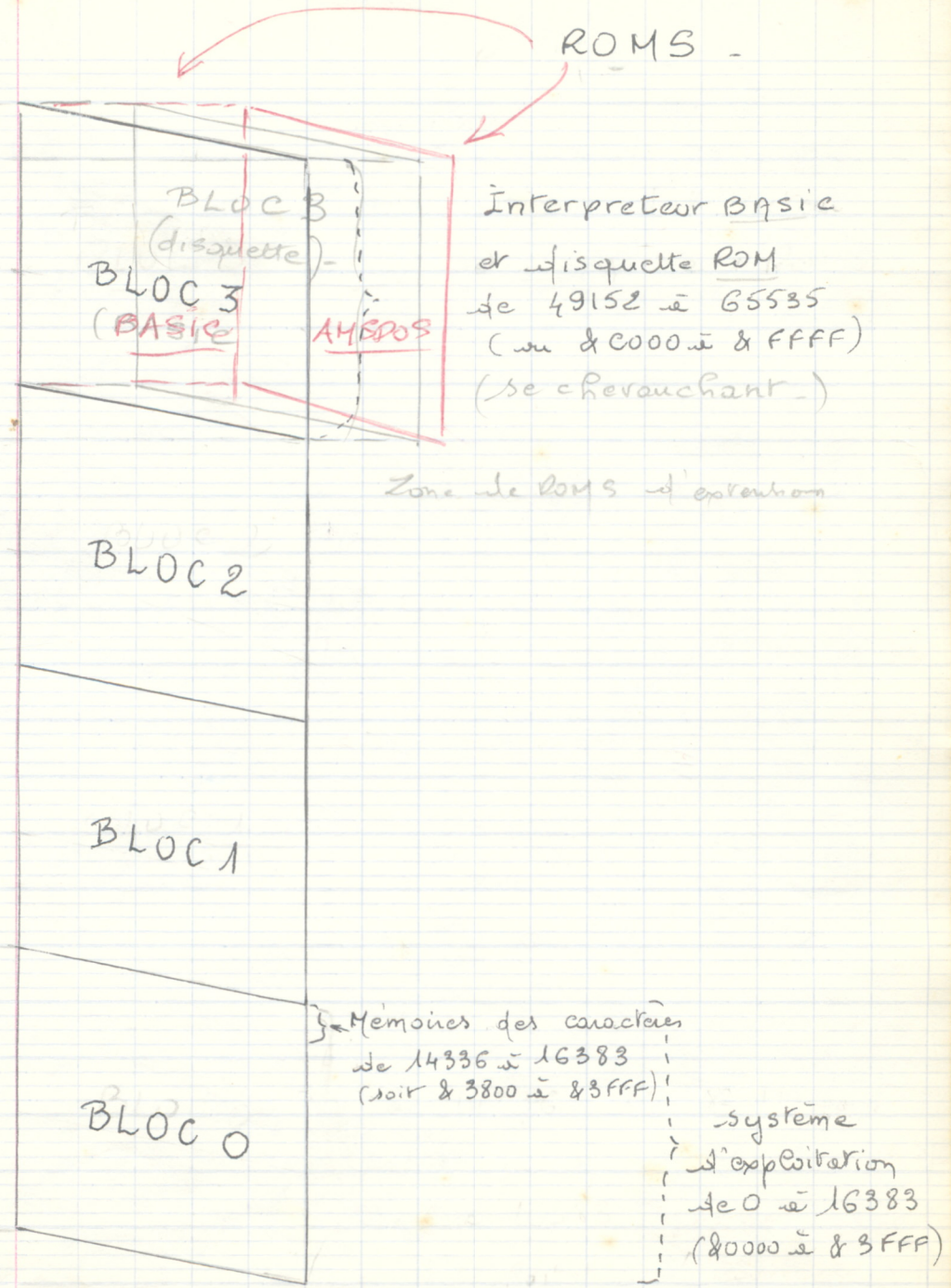
Les 16 Ko inferieurs de la ROM contiennent le système d'exploitation et un base de routines arithmétiques. Dans le système d'exploitation se trouvent toutes les routines dont le CPC a besoin pour lire par ex. un caractère tapé au clavier placer un caractère ou un point à l'écran, mais c'est également le système d'exploitation qui commande lecteur de cassette, et l'interface imprimante et le son.

Dans les 16 Ko superieurs se trouve l'interpréteur BASIC,

Caractère à 3800 à 3FFF soit -

RAM -


RAM -
RAMS -
65535
(ou &FFFF
BANK 0
BANK 1
BLOC 3
Mémoire Écran
49152
(ou &C000
49151
(ou &BFFF)
Caractères définis par l'utilisateur - HIMEM
Zone des données des programmes
Zone de branchements + Zone données BASIC
Zone de données des ROMS d'extension -
Limite HIMEM 42619 (&A67B)
BLOC 2
32768
(ou &8000)
32767
(ou &7FFF
BLOC 1
Bloc
Central
BASIC -
16384
(ou &4000)
BASIC
et variables
de 369 à ~~43903~~ 42619
(&0171 à &~~ABFF~~
&A67B)
16383
(ou &3FFF
BLOC 0
0
(ou &0000)
← 0 à 368 (&0000 à &0170
système


ROMS -
BLOC 3
(disquette) -
BLOC 3
(BASIC
AMSDOS
Interpreteur BASIC
et disquette ROM
de 49152 à 65535
(ou &C000 à &FFFF)
(se chevauchant -)
Zone de ROMS d'extension
BLOC 2
BLOC 1
BLOC 0
Mémoires des caractères
de 14336 à 16383
(soit &3800 à &3FFF)
système
d'exploitation
de 0 à 16383
(&0000 à &3FFF)

Bible 6128. / p285

ROM - Interpréteur Basic &C000 à &FFFF en //
Bloc 3. avec - Pa RAM écran - (49152 à 65535)
ROM Arithmétique entière ou virgule flottante dans
Bloc 0 - Pa ROM, système de &2F73 à &37FF
- soit 12147 à 14335 -

Le Système d'Exploitation ← Ses ROUTINES sont appelées par des (VECTEURS situés en RAM).
(renvoient à travers l'adresse de la ROM) (p 117)
situés de &B900 à BD5B (p 130).

Les VECTEURS utilisés par BASIC - pour ~~utiliser la ROM~~ MATH
de BD5E à BDBE soit - (en RAM)
(Il s'agit de l'appel de la ROM Mathématiques
Arithmétique entière et flottante, située en ROM Interpréteur
aux adresses 2F73 à 3731 - soit. (p 260)

Suivent les INDIRECTIONS / p 140 - (Explication p 129 -
il s'agit de sauts dans le Système qui ne sont pas traités
globalement par chaque PACK (PACK TXT / FLOPPY ...) -
mais individuellement pour chaque PACK lorsque son
RESET ou INITIALISE est rencontré.

La ROM Système d'Exploitation :
de 10000 à 3731 soit - ← GENERATEUR de
ou (37FF)

- LOGICIEL - INTERNE -

LA BIBLE du 6128

Les Vecteurs du Système d'Exploitation p 130 à 138 sont classés par ordre de grandeur -
de B900 à BD37.
p 131 et 132 (+2 p 138) KM →
de BB00 à BB4B -
p 132 à 134 - (+1 p 138) TXT →
BB4E à BBB7
de p. 134 - (+4 p 138) GRA →
BBBA à BBFC -
p 134 à 136 (+1 p 138) SCR →
BBFF à BC62
p 136 - CAS - →

BC65 à BCA4 -

Clefs pour AMSTRAD

→ = Points d'Entrée des routines Système p 81 à 104 - sont classés par groupe de GESTION du code

= CLAVIER n° 00 à 25 - p 81 à 84 -

TEXTE n° 26 à 61 p 84 à 88 -

GRAPHIQUE n° 62 à 84 - p 88 à 91 -

ECRAN - n° 85 à 118 p 91 à 95

CASSETTE n° 119 à 140 p 95 à 98 -

BIBLE du 6128 — Clefs pour AMSTRAD

GESTION de ce site.

p 136 à 137 — SOUND → SONORE n° 141 à 151
BC A7 à BC C5 — p 98 à 99 —

p 137 et 138 (+ 1 p 138) — KL → KERNEL n° 152 à 163
BC C8 à BD 10 — p 100 à 103 —

p 138 — MC → Interfaçage à MATERIEL.
(+ 1 p 138 — n° 176 à 188
BD 13 à BD 34 — p 103 à 104 —

p 138 — JUMP → Bloc de SAUT
BD 37 — n° 189
p 104 —

p 140 — INDIRECTIONS → VECTEURS d'INDIRECTIONS
+ 1 p 140 — n° 1 à 13
BD CD à BP F1. — p 106

p 130 — Vecteurs du Système d'Exploitation (KL-O, KL ROM etc..) — KL → LES VECTEURS NOYAU. et les RESTART
n° 1 à 12
p 107 et 108 —

p 166 à 168 KERNAL (KL) → Les VECTEURS en BAS de MEMOIRE.
n°1 à 16
p 109 et 110 -

000 à 0036 -

p 196 - VECTEURS BASIC → Les VECTEURS d'APPEL
3 sauts différents - des ROUTINES MATHÉMAT.
p 139. adresses VECTEURS + 3 - L'additif p 172 - (664) -
p 196 " REELLES inchangé - Il est probable que
p 260 Vecteurs et REELLES = 664 les adresses VECTEURS
p 412 - adresse réelle inchangé. sont décalés de + 3 -
et les adresses réelles
inchangé / prendre
additif du 664 et + 3
aux VECTEURS / rien aux
adresses réelles -

BIBLE C128

ROM. Inferieure -

p. 402. plus detaillée -

p 165 - de 0000 à 0057 - Les RESTART -

de 005C à 0586 KL

" 0592 à 087F MC -

08BD à 0A70 JP

0ABP à 1072 SCR

1074 à 15A5 TXT

15A8 à 1B56 GRA -

1B5F à 1FE7 = KM / 1FE9 à 2AE2 SOUND

24BC à 25B9 CAS - Edit de

2C02 à 3731 / Générateur CARACTERES 3800 à 3FFF -

p 303 - La ROM Superieure

Codes from AMSTRAD.

Adresses Principales de la ROM Inferieure

- p 196 de l'additif -

et p 182 à 184 (6C4) -

- Les détails dans l'ouvrage principal -

Voir si calcul des Adresses Possible ?

Adresses Principale de la ROM Superieure -

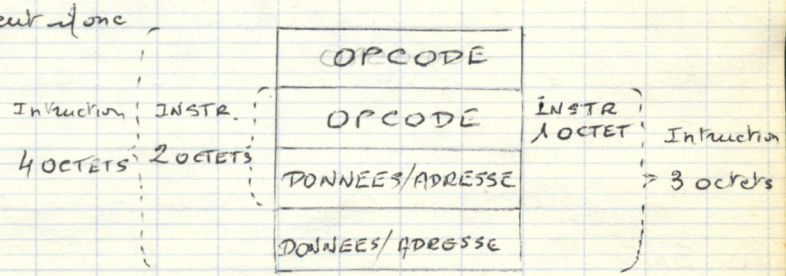
p 199 de l'additif -

Adresse : JP y a 65536 octets dans la RAM, numerotés de &0000 à &FFFF (en Hexadecimal). Le n° correspondant à 1 octet est son adresse.

Valeur - En Basic PEEK lit le contenu de la valeur, POKE stocke la valeur, dans l'adresse indiquée. Comme chaque adresse est affectée à 1 octet, et qu'un octet = 8 bits, la valeur à lire ou stocker, doit être comprise entre 0 et 255 (&00 à &FF)

Attention aux POKES inconsidérés qui peuvent planter l'ordinateur.

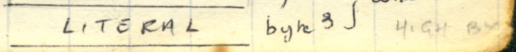
OPCODE Code d'opération. JP contient parfois des bits supplémentaires (qui ne font pas partie du code d'opération) mais admis comme tel pour simplifier. Ces bits utilisés comme pointeur sur un registre. L'opcode peut être suivi d'un opérande ou d'une adresse. JP y a de plus des instructions dont l'opcode est long de 2 octets. Une instruction peut donc avoir une longueur de de 1 à 4 octets.



Opérande : Élément sur lequel se porte une opération

Constante nombre de 8 bits dans une instruction d'adressage immédiat. Également appelée **Littéral**. L'OPCODE sur 8 bits est suivi d'un littéral 16 bits (la constante). Ex : LD C, &7F.

- Adressage — immédiat



BASIC ↓

Exemple p. 54, 53, 52 -

```
10 POKE &C000, &FF
20 POKE &C800, &FF
30 POKE &D000, &FF
40 POKE &D800, &FF
50 POKE &E000, &FF
60 POKE &E800, &FF
70 POKE &F000, &FF
80 POKE &F800, &FF
MODE 2
RUN
```

Langage - Machine ↓

```
LD A, &FF      ← (adressage immédiat :
LD (&C000),A     (Code : 3E, constante 8 bits)
LD (&C800),A     (Tableau p 256) 265
LD (&D000),A   Adressage absolue
LD (&D800),A   ↓ Code 32, octet
LD (&E000),A     faible, octet Fort -
LD (&E800),A     (Tableau p 266 -
LD (&F000),A     (nn = adresse) -
LD (&F800),A     (al = octet faible ah = Fort)
RET ←          Code: C9 p 279 -
```

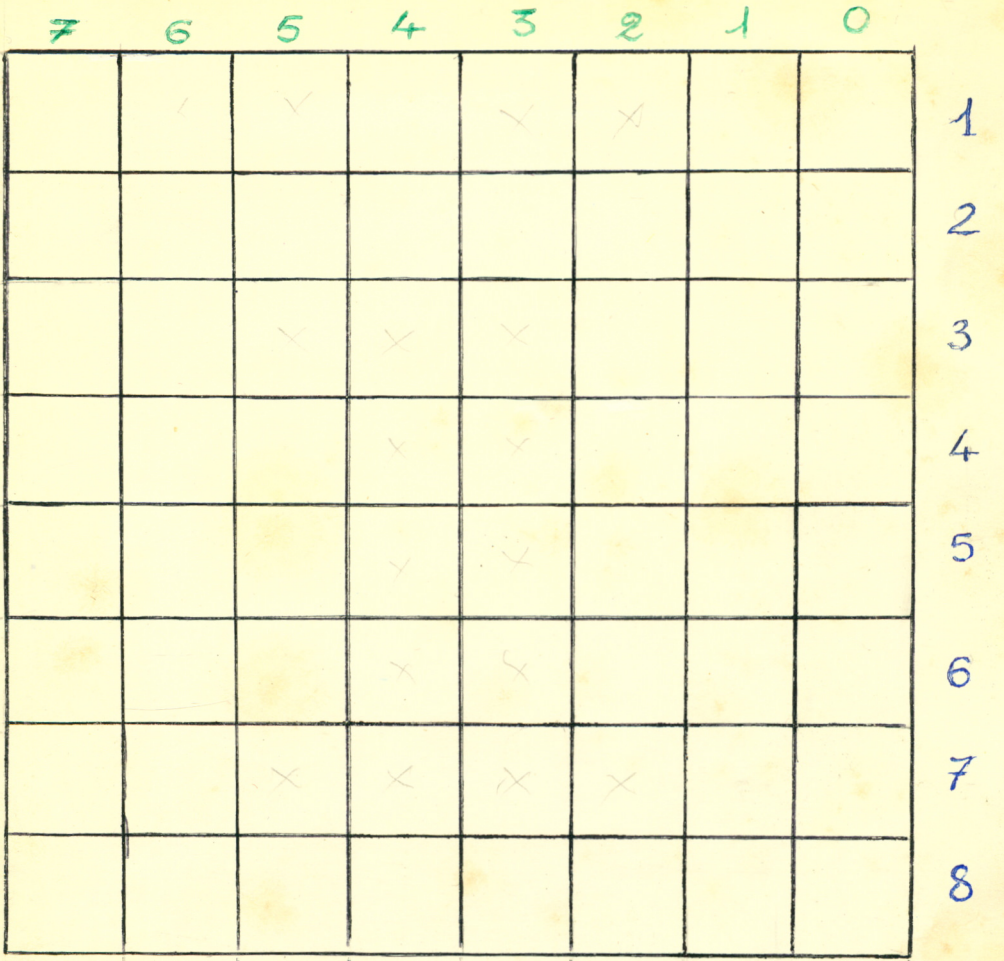
Ce qui donne le prog en Langage Machine :

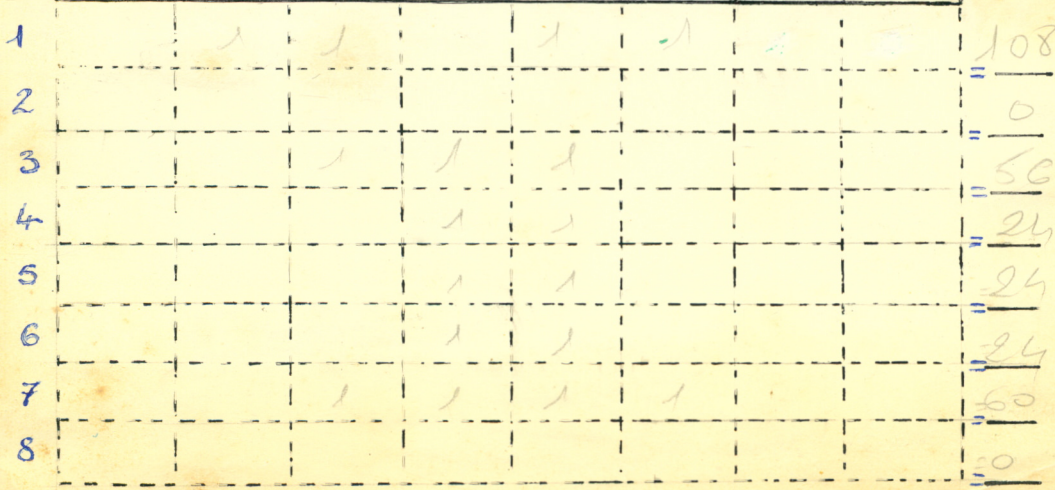
Commentaires -

```
10 MEMORY &9FFF   (choix) -
20 FOR i = &A000 TO &A01A   (&A000 choix) - la longueur d'un prog
30 READ a           = nbre de DATAS - 1 : soit 27. L'adresse d'arrivée est
40 POKE i, a        obtenue par &A000 (adresse départ) + 27 - 1 soit
50 NEXT i           en hexa &A000 + &1A = &A01A -
60 END.
70 DATA. &3E, &FF, &32, &00, &C0, &32, &00, &C8 -
        Code. / Constante / Code / OFA, OFo de &C000 / code / OFA, OFo de &C800
80 DATA &32, &00, &D0, &32, &00, &D8, &32, &00, &E0
        Code / OFA, OFo de &D000 / code / OFA / OFo de D800 / code / OFA, OFo de &E000
90 DATA. &32, &00, &E8, &32, &00, &F0, &32, &00, &F8 -
        Code / OFA, OFo de E800 / code / OFA, OFo de &F000 / code / OFA OFo de &F800
100 DATA &C9 (Code retour au Basic -
```

Adresse	n° ligne	Code	Label	Code
9CG7	210	DD5607		LD D,(IX+7)
9CGA	220	D5		PUSH DE
9CGB	230	DD5E04		LD E,(IX+4)
9CGE	240	DD5605		LD D,(IX+5)
9C71	250	D5		PUSH DE
9C72	260	DD5E02		LD E,(IX+2)
9C75	270	DD5603		LD D,(IX+3)
9C78	280	D5		PUSH DE
9C79	290	DD5E08		LD E,(IX+8)
9C7C	300	DD5609		LD D,(IX+9)
9C7F	310	DD6E06		LD L,(IX+6)
9C82	320	DD6607		LD H,(IX+7)
9C85	330	D5		PUSH DE
9C86	340	CDEABB		CALL PLOT:
9C89	350	D1		POP DE
9C8A	360	E1.		POP HL
9C8B	370	E5		PUSH HL
9C8C	380	CDFGBB		CALL DRAW:
9C8F	390	E1-		POP HL
9C90	400	D1		POP DE
9C91	410	D5		PUSH DE

PUISSANCE : 2

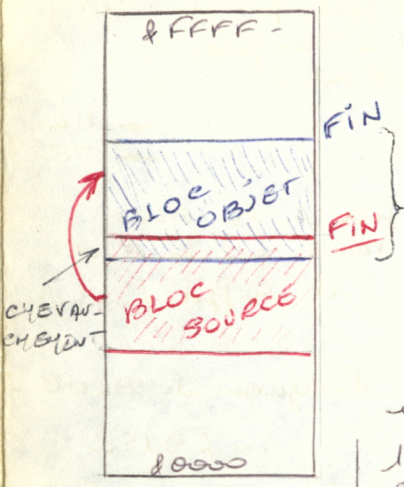




§ = = 166 : CTRL / +]

o = 162 CTR / / E

TRANSFERT DE BLOC



(LDD (R))

LDD = Charge et Décrémente -

LDDR = idem + R = Répète -

Pour utiliser l'Assembleur "AUTOFORMATION" connu en 64018 avec un REM Mettre 64022 MODE 2 -

Assembler -

Taper X pour passer en BASIC -

MEMORY &9FFF

MODE 2 -

ou modifier 640 -

en MODE 2 : MEMORY &9FFF

Décalage d'1 Octet dans la Zone Écran - SOURCE et OBJET se chevauchent. Donc LDD(R)

Adresse FIN SOURCE dans HL

" FIN OBJET dans DE

Longueur de BLOC dans BC

HL = &FFFF - 1 Octet = &FFFE

DE = &FFFF - et BC = Longueur Écran - 1 soit FFFF - C000 = 4000 - 1 = 3FFF -

avec LDD - une boucle est nécessaire.

```
10 ORG &A000 -
20 LD HL, &FFFE;
30 LD DE, &FFFF;
40 LOOP: LDD ← 35 LD BC, 3FFF
50 JP PO, FINISH:
60 JP LOOP:
70 FINISH: RET -
```

Avec LDDR - Pas boucle et inutile -

```
10 ORG &A000 -
20 LD HL, &FFFE
30 LD DE, &FFFF
40 LDDR ← 35 LD BC, 3FFF -
50 RET -
```

Pour décaler 40 Octets (Milieu Ecran) -

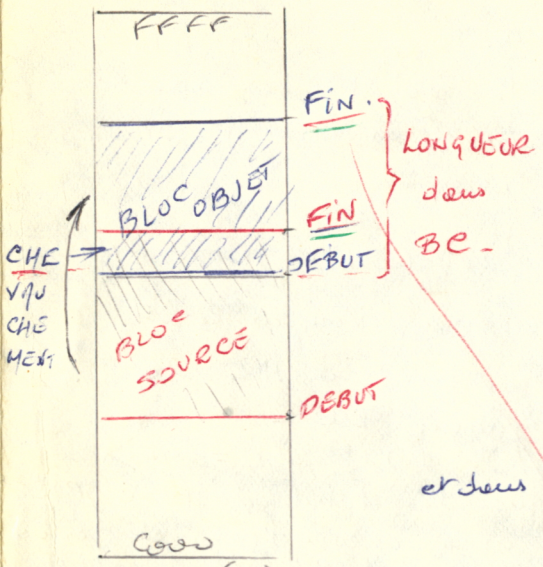
Adresse FIN SOURCE = &FFFF - 40 = &FFD7 -

Adresse FIN OBJET idem &FFFF -

Longueur BLOC = &4000 - 40 = &3FD8 -

```
10 ORG &A000 -
20 LD HL, &FFD7 -
30 LD DE, &FFFF -
40 LD BC, &3FD8 -
50 LDDR -
60 RET -
```

- TRANSFERT de BLOC - LDDR



Décalage d'1 octet dans la Zone Ecran - SOURCE et OBJET se chevauchent pendu LDDR.
Zone Ecran = C000 à FFFF.
Adr de FIN SOURCE = FFFE - (dans HL) - soit FFFF - 1 (octet).
Adr de FIN - Objet = FFFF. (dans DE)
Zone Ecran = 1 4000
et dans BC - Longueur 4000 - 1 = 3FFF

```
LD HL, FFFE
LD DE, FFFF
LD BC, 3FFF
LDDR -
RET.
```

⊛ La case C000 étant allumée - la mettre à 0 - avec

```
LD A, 0
LD (C000), A -
```

Décalage de 40 octets - (Milieu ECRAN) -

Adr FIN SOURCE = FFFF - 40 = FFD7.
Adr FIN OBJET = FFFF.
LONGUEUR = 1 4000 - 40 = 3FD8 -

```
    LD HL, & FFD7.
    LD DE, & FFFF
LDDR LD BC, & 3FD8
    LD A, 0 -
    LD (C000), A.
    RET -
```

LDD(R) - Transfert et Décrémente (à (Répétition) - à cause de fait de fournir des ADRESSES FIN

COMPARAISONS de BLOCS (ou Gotets)

CP reg - Simule une soustraction de l'A - Reg, mais l'Accu reste inchangé, Mais les flags sont modifiés.

- en conséquence : Si -

	C	Z	S	P/V
A > Reg	C = Ø	Z = Ø	S = Ø	P/V = Ø
A = Reg.	C = Ø	Z = 1	S = 0	P/V = 0
A < Reg.	C = 1	Z = Ø	S = 1	P/V = 1.

Reg etout B ou C, D, E, H, L, ou (HL) ou (IX + d) -

Mettre Ø à 255 dans -
A et B -

```
10 ENT
20 LD A,0 à 255
30 LD B,0 à 255 -
40 CP B
50 JP C, GRAND: ;Si C=1, B>A
60 JP Z, EGAL: ;Si Z=1, B=A.
70 LD A, GG ;  Ascii de B
80 CALL BBSA :  Afficher
90 LD A, &3C ;  Ascii de '<'
100 CALL BBSA  Afficher
110 LD A, 65 ;  Ascii de 'A'
120 CALL BBSA :  Afficher
130 RET.
140 GRAND: LD A, GG ; Ascii de B -
150 CALL BBSA - Affiche
160 LD A, &3E ; Ascii de >
170 CALL BBSA -) Affiche
180 LD A, 65 : Ascii de A -
190 CALL BBSA :
200 RET -
210 EGAL: LD A, GG ; Ascii de B -
220 CALL BBSA -
230 LD A, &3D : Ascii de "="
240 CALL BBSA : Afficher
250 LD A, 65 : Ascii de "A"
260 CALL BBSA : Afficher
270 RET -
```

Si vous entrez
20 en A et
10 en B, le résultat reste positif.
Pas de saut à GRAND:
ni à EGAL.
Les FLAGS sont inchangés
L'Ecran affiche : B < A

Si vous entrez
10 en A et
20 en B le résultat est négatif.
Le flag C = 1
Saut à GRAND:
L'Ecran affiche : B > A

Si vous entrez
20 en A et
20 en B, le résultat est Ø
Le flag Z = 1 -
Saut à EGAL:
L'Ecran affiche .
B = A -

COMPARAISON DE BLOCS - (ou Octet).

CP -

CP reg : Compare le contenu de l'Accu, mais ne modifie pas son contenu (Contrairement à SUB) - mais modifie les Flags, si nécessaire -

A > Reg : C = 0 - Z = 0 - S/V = 0 -
A = Reg : C = 0 - Z = 1 - S/V = 0 -
A < Reg : C = 1 - Z = 0 - S/V = 1 -

Reg pourrait être A, B, C, D, E, H, L, ou (HL) ou (adr), ou (XY+-)

Ex : CP H - A = 33 - H = 64 -

```
A =   00100001   ) Comparaison !
H = - 01000000
      1.11100001  (= -31 en CPL A2)
```

C = 1 - Z = 1 - P/V = 1 -

et RECHERCHE DE BLOCS -

CPD : Décremente donc HL = FIN - BC = Longueur

```
LD A, 65     (Charge "A" dans l'Accu)
LD HL, &C0FF (adresse de FIN)
LD BC, &0A   (Longueur &0A = 10).
LOOP : CPD   (Comparer C0FF - 0A avec l'Accu)
CALL &BB5A -  (écrire à l'écran "A" en l'occurrence)
JR NZ, LOOP  (saute si NZ à LOOP sinon RET)
RET -
```

- Un paquet de "A" s'affichait à l'écran (31)

pour CPDR

CPDR et DJNZ, LOOP

remplacer JR NZ, LOOP :

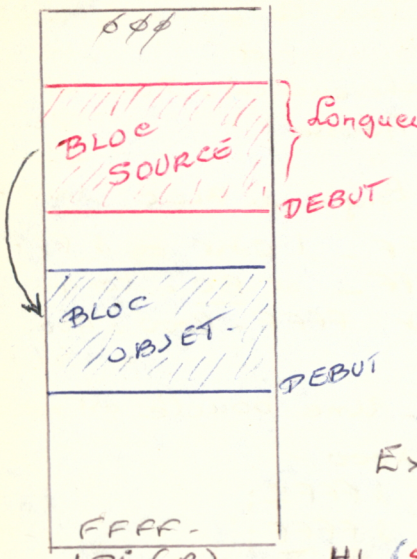
par DJNZ LOOP.

Voir AUTO FORMATION - p - 113 - 114 -

1 - LDI - / LDIR

Charge et Incrémente

Il ne faut pas que les BLOCS à Transférer se touchent -



Adresse DEBUT SOURCE → HL

Adresse DEBUT OBJET → DE

Pour vérifier si le BLOC OBJET ne touche pas le BLOC SOURCE -, il suffit d'ajouter à HL (début BLOC SOURCE) à BC = Longueur de BLOC -

Si HL + BC > DE = LDDR

Ex: DE (début OBJET) = 49152 soit &C000 début Mémoire ECRAN

HL (SOURCE) si l'on veut transférer les 16 Ko de la Mémoire dans l'écran, soit 16384 - (16 x 1024) il faut que HL (SOURCE) soit inférieur de 16384 à DE (OBJET) soit - HL = 49152 - 16384 = 32768 -

LDIR - Charge, Incrémente et Répète - ce qui évite la boucle avec LDI -

```
10 ENT -
20 LD HL, 32768 -
30 LD DE, 49152
40 LD BC, 16384
        (ou &4000)
50 LDIR ( Incremente
          et Repete
          de 49152
          à 65535.
60 RET
```

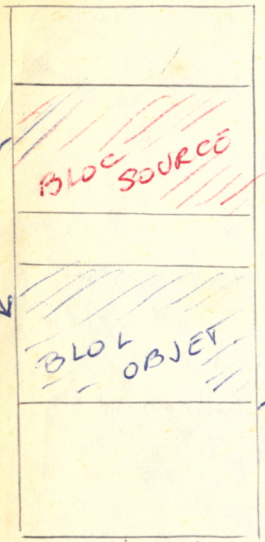
```
10 ENT -
20 LD HL, 32768 - (=DE-BC)
30 LD DE, 49152 (début ECRAN)
40 LD BC, 16384 ; (=DE-HL
                   =16 Ko
50 LOOP: LDI    ; ( de RAM
60 JP PO, FINISH: ( ECRAN
70 JP LOOP:
80 FINISH: RET -
```

HL + BC ne doit pas être = ou > à DE -

Tolérance + 8

et l'écran est remplie -

Ne pas SORTIR de la ZONE ÉCRAN - [49152 à 65535 ou &C000 à &FFFF -



LDI(R)

LDIR -

Transfère les données d'un endroit dans HL/DE - Le début des adresses BLOC SOURCE et OBJET Mais il ne faut pas que le BLOC SOURCE chevauche le BLOC OBJET -

- Facilement vérifiable si l'on ajoute à HL la longueur BC - Si le Total est = ou > que le début du BLOC OBJET il y a chevauchement et pertes de Données -

```
LD   HL, C000  (49152)
LD   DE, DFFE  (57342)
LD   BC, 1FFE  ( 8190)
LDIR
RET -
```

HL	49152
+ BC	+ 8190
= DE	= 57342.

Il y a tolérance jusqu'à 8202

8202
- 8189
0013. soit 0000/0110/1 soit = 2C -

Pour utiliser l'Assembleur "Autoformation" -
Annuler G4018 par un REM -
Mettre G4022 en MODE 2 -
Taper 10 ENT &A000
20 ORG &A000 puis

par X passer en BASIC
Mettre MEMORY : &9FFF
en MODE 2 lancer CALL A

LDIR Transfère et (Incrémente) d'un en fait de partir des adresses de DEBUT -

PS Ne pas sortir de la ZONE ECRAN -
C000 à FFFF soit 49152 à 65535.

CPI / CPD

CPI

Compare et Incrémente, il faut donc entrer le DÉBUT du BLOC à comparé + et dans BC la longueur +1 pour PO = Parité Impaire –

```
10 LD A,65 ; Mettre A dans l'Ac
20 LD HL,0 ; Début Mémoire
30 LD BC,0 ; Longueur 64 Ko
             soit FFFF+1 = Ø
40 LOOP: CPI
50 JP PO,FINISH: Fin si PO=1
60 CALL Z,FOUND ; FOUND si Z=1
70 JP LOOP : Boucle –
80 FINISH: CALL Z,FOUND;
       Vérifier si flag Z = 1 –
90 RET –
100 FOUND : CALL BB5A –
            Affiche
110 – RET –
```

CPD –

Compare et décrémente il faut donc entrer la fin du BLOC à comparé et dans BC

```
SENT
10 LD A,65
20 LD HL,&FFFF (FIN)
30 LD BC, (pas d'ajouter 1
   → &FFFF !! (FFFF est impair)
40 LOOP: CPD
50 JP PO,FINISH –
60 CALL Z,FOUND !
70 JP LOOP
80 FINISH : CALL Z,FOUND
90 RET –
100 FOUND: CALL BB5A
110 RET –
```

La boucle FINISH : FOUND, car l'on teste PO, puis Z, puis retour à PO avec LOOP, évitant de passer, avant le dernier CALL Z, directement à CALL BB5A qui altère les flags –

CPIR –

Compare Incrémente et Répète –

Il suffit de remplacer en 40 – CPI par CPIR

La prog est + rapide car les conditions n'ont pas à être testées après chaque comparaison –

CPDR –

idem –

40 – CPDR –

Le nombre de À à l'ECRAN passant par option 2 –

CPI = 222)
CPIR = 223 } variable
CPD = 226 (
CPDR = 227

CPI Incrémente alors HL = DEBUT BC = Longueur
LD A, 65 (Code Ascii de "A" dans l'accu).
LD HL, &FFF0 (Adresse de DEBUT de Bloc).
LD BC, &ØA (Longueur à ØA soit 10).
LOOP CPI (Comparer &FFF0 + &ØA avec l'Accu)
CALL &BB5A
JR NZ, LOOP (si non Zero LOOP, si Zero RET)
RET
Un paquet de "A" s'affichent à l'écran (31)

---

pour CPIR remplacer CPI par
JR NZ, LOOP: CPIR
par DJNZ LOOP NC, par DJNZ

```
10 ENT
20 LD A,65
30 LD HL, &C00A-
40 LD BC, 4
50 LOOP: CPD
60 CALL &BB5A
70 JP PE, LOOP:
80 RET
```

= 4 "A" affichés-

CALL ne semble pas affecté PE ?

Les Flags V et N (DCB) sont modifiés -
mais P/V est le V de Overflow de PE/PO ou de PO
P/V bit de PARITÉ -
V = Débordement sous report sur le Carry ?

FLAGS ou registre F -

Registre 8 bits, comme A, B, C, D, E, H et L, mais ses fonctions sont totalement différentes -

7	6	5	4	3	2	1	0
S	Z		H		P/V	N	C

Flag -

C = CARRY - Ce bit est mis (C=1) si une opération génère un report du bit 7 sur bit 8 - Ce report ou retenue Externe est sauvegardé dans le Flag C

Ex : sur des Entiers non signés -

(128) 10000000
(129) 10000001
(257) = 1 00000001

Flag N ou Flag de Soustraction : N = 1 si l'opération précédente est une soustraction ou INC DEC

FLAG P/V de PARITÉ ou de DÉBORDEMENT -

DEBORDEMENT V : S'il y a un report du bit 6 sur le bit 7, sans qu'il y ai report du bit 7 sur bit 8, donc s'il n'y a pas de retenue Externe (Carry) - V = 1 -

S'il y a un report du bit 7 sur le bit 8, et que ce report ne vienne pas d'un report interne du bit 6 sur bit 7 alors V = 1 -

PARITÉ - Indique la PARITÉ d'un résultat : Cette Parité est obtenue en additionnant les "UNS" (bits allumés) ou "1" de la représentation Binaire, Si ce total des "1" est pair la PARITÉ est mise

Ex : V = 1 bit 6 et bit 7 . Report interne -

```
  01000000   (64)
+ 01000001   (65)
----------
  10000001 = (129) [En fait -127 en CPL à 2]
```

FLAG H Flag de 1/2 retenue ; utilisé par DAA en DCB si l'opération génère un report du bit 3 à 4 d'Accu .

FLAG Z : Zéro, est mis si le résultat de l'opération = 0 alors Z = 1 -

FLAG S Flag Signe + En représentation en Binaire Signé, le bit 7 indique le Signe -. Si bit 7 = 1 - le nombre est négatif + si bit 7 = 0 le nombre est positif -

SAUTS, BRANCHEMENTS Conditionnels.
JPc, JRc, CALL c, RET c.

Z	saute si	ZÉRO	(Z = 1)
NZ	" si	Non Zéro	(Z = 0)
C	" si	Retenue	(C = 1)
NC	" si	Non Retenue	(C = 0)
PO	" saute si	Impas PARITÉ	(P/V = 0)
PE	" " si	PARITÉ	(P/V = 1)
P	" si	PLUS (+)	(S = 0)
M	" si	MOINS (−)	(S = 1)

- LA PILE - (et le SP).

- C'est : Un bloc mémoire fixée au départ à partir de &C000 soit : 49152 -
- Son utilité : Lors des sous programmes, des interruptions et le stockage temporaire des données -
- Son fonctionnement : Un registre spécial lui est affectée le SP - (Stack Point) ou "Pointeur de Pile" - Elle fonctionne automatiquement, si l'on a pas besoin de ses services +
Mais dans le cas par exemple d'un sous programme avec "CALL adresse", et "RET" si l'on veut protéger des données, ou noter l'adresse de "RETour" + puis récupérer ses données ou adresses, il existe 2 instructions
PUSH : "Empiler" POP : "Dépiler"
- La Pile ressemble à une pile d'assiettes, si l'on en prend une, l'on est obligé de prendre la dernière posée + Dernier entré, Premier Sortie +
- Mais ce qui est moins évident, c'est que lorsque l'on Empile (PUSH) les Pointeur de Pile SP est décrementé de 1 Octet + 1 Octet + Soit -2 + Un peu comme si la 1ère assiette était collée au Plafond, (≈ C000) la 2ème sous la 1ère (≈ C000 - 2) etc....
Et lorsque l'on dépile (POP), c'est à dire lorsque l'on enlève une assiette le Pointeur de Pile augmente de 1 Octet + 1 Octet +
Admettons, sans nous poser trop de questions, que un ordinateur, Empile ses assiettes, (ses adresses plutôt) en partant du plafond, les unes sous les autres + Retirant ainsi celle du dessous, sans faire tomber la pile +
Exemple numérique - PUSH BC, Empilons BC sur la pile +
Valeur de BC : &E0F1 : B = 224 - C = 241 -
Valeur de SP : &BBF4 : soit en Déci - 49140 -
Nous partirons de l'Octet en dessous soit 49139
L'Empilage se fera dans le sens : 0F0 (soit B) puis 0FA (C)
SP = 49139 ⟶ = 224 (Décimal) - (Nous Empilons en descendant)
SP = 49138 ⟶ = 241 (")
2ème Empilage PUSH DE
Valeur de DE : &8401 : D = 129 E = 1

SP = 49137 ——→ = 129 Nous empilons des données ou adresses, et le SP décroit
SP = 49136 ——→ = 1.
- L'ordinateur qui empile fait en dessous, le tout est de le savoir -

~

Maintenant l'instruction POP qui effectue le travail inverse : Elle retire les données/adresses, en commençant par la dernière entrée, c'est-à-dire celle du dessous, Elle dépile la Pile d'assiettes en prenant celle qui se trouve dessous (sans faire tomber la Pile, n'oublions pas, que nous avons "empilé" en partant du plafond et en descendant, vers le bas)

POP DE.

Valeur de DE = ??. - après POP... !?. nous verrons dessous

SP = 49136 contient 1.
SP = 49137 contient 129.

Si nous dépilons la Pile sur/ou vers le registre DE, celui-ci recevra dans l'ordre Octet Faible (OFA) soit E, la dernière valeur pointée par SP
soit SP=49136 = E = 1. (Décimal).
puis en Octet Fort (OFO) D = 129 (Décimal).
La dernière valeur pointée par SP - (nous dépilons, nous enlevons des données mais le SP augmente!)
soit SP 49137 = D = 129.
Soit la valeur du registre DE, maintenant :

OFO = &81 + OFA = &01 = soit DE = &8101 -

- Sa valeur est inchangée, elle est la même que lorsque nous empilons avec PUSH + -

Mais Si nous dépilons encore avec POP et encore le registre DE ? - Dans (Re) POP DE
SP = 49138 contient 241
SP = 49139 contient 224.

Celui-ci recevra dans l'ordre OFA (E) et OFO (D) les valeurs suivantes.
SP 49138 = 241 → E = 241 - (= &F1)
SP = 49139 = 222 → D = 224 - (= &E0)

Soit la nouvelle valeur de DE : &E0F1 +
- DE est désormais égal à BC -

Attention à toute instruction PUSH doit correspondre une instruction POP, sinon Plantage -

SP = 49140 [Empilage] PUSH.

PUSH BC : Valeur $EOF1. (Empilons en décroissant SP)

SP 49139 = $EO (OFO)
SP 49138 = $F1 (OFA) } OFO, puis OFA

PUSH DE : Valeur $8101. (Empilons toujours en décroissant SP)

SP 49137 = $81 (OFO)
SP 49136 = $01 (OFA) } OFO puis OFA.

Dépilage POP.

POP DE (Dépilons en incrémentant SP, cela se fait automatiquement)

SP = 49136 = $01 = E (OFA)
SP = 49137 = $81 = D (OFO)
DE = $8101 (inchangé).

POP DE

SP = 49138 = $F1 = E (OFA) } OFA puis OFO
SP = 49139 = $EO = D (OFO) } (Contrairement à PUSH)

DE = $EOF1 = Valeur de BC

DE = BC (qui lui conserve sa valeur initiale).

Résumé : L'on Empile des données sur la Pile, et le SP va en diminuant. (PUSH)
L'on dépile des données sur la pile pour les charger dans des registres doubles et le SP va en augmentant.

PUSH BC (= $EO) l'OFO / C, l'OFA = $F1.

SP = 49139 contient l'OFO ($EO)
" 49138 " l'OFA ($F1).
" 49137 PUSH DE = Valeur = $8101 = OFO = $81
= OFA = $01
SP = 49137 contient l'OFO ($81).
" 49136 " l'OFA ($01).

POP DE.

49136 E (l'OFA) charge $01 dans le registre DE (E)
49137 F (l'OFO " $81 " DE (D)

POP DE.

49138 = OFA de DE = $F1 = Valeur de E.
49139 = OFO de DE = $EO = Valeur de D.

DE = OFO + OFA = $EOF1.

Le SP pointe la dernière position dans la Pile, c'est à dire le dernier octet de la Pile.

- Autre utilisation + Des valeurs chargées dans des registres doubles, ou l'Accu, peuvent être chargée sur la Pile PUSH HL (s'il s'agit de lui) ou A, l'Accu, si un appel de sous programme "Call adresse" altère - Pour conserver - (Flags)

Ex: LD HL, &0C04 + H colonne 4 + L ligne 6 -
LD A, &4D + valeur Ascii de "M"
PUSH HL - + Charge sur la Pile et protège -
PUSH AF - + Accu " " " " + le registre Flag + L'on ne peut charger que des registres doubles.
CALL &BB75 - Fixe le curseur aux indications HL (1ère ligne)
POP AF Rappelle l'Accu. (donc "M" Ascii)
CALL &BB5A Ecrit "M" avant décalage (ligne suivante)
LD BC, &0101 - Décalage dans BC
POP HL rappelle HL (non altéré)
ADD HL, BC Ajoute le décalé à HL (+1 colonne et ligne)
PUSH AF Sauvegarde Accu + Flag sur la pile ("M" en Ascii)
CALL &BB75 - remet le curseur à HL + décalage
POP AF Rappelle l'Accu et Flag - ("M" en Ascii)
CALL &BB5A - Ecrit "M" à sa nouvelle position.

Les routines (positions HL) - Le 1er CALL &BB75 fixe le curseur, mais altère les Flags d'où protection (PUSH) le 2ème &BB5A Lit le contenu de l'Accu - et RET -

En gros: L'on place sur la Pile une adresse de l'un des registres doubles, à l'aide de PUSH (rr), l'adresse est sauvegardée, et non altérée par les appels de sous programmes: "CALL adresse" - Pour chaque PUSH le registre SP est décrémenté (automatiquement) bien qu'il s'agisse d'un Empilage + Pour retrouver l'adresse de "RETour il faut suivant de POP que de PUSH -

La pile est utilisée pour les sous programmes, le stockage temporaire de données et les interruptions.

INSTRUCTIONS de COMMANDE

NOP Non Operation - Cette Instruction codée &00 permet de relancer un prog ; du fait que le Z80 est des "00" il se fait rien c'espace d'un cycle (10^-6 sec) / Elle est utilisée pour des boucle de temporisation ou pour neutraliser des octets d'un message --

HALT Stoppe le prog, jusqu'à réception d'une interruption - Elle "rafraîchie" la Mémoire en attendant.

DE COMMANDES et INTERRUPTIONS

Une interruption (un sous prog en fait) demande à être exécuté quand elle est prête, et non quand le Z80 est prêt - Dès Lors Le Z-80 doit d'abord sauvegarder ses Flag Regs (de plus souvent sur la Pile), Faire ce que doit faire l'interruption, et il doit restaurer ensuite ses registres et ses flags, et poursuivre son programme originel - Il y a 2 sortes d'interruptions : Les interruptions Masquables qui peuvent être bloqué par l'instruction DI Empêche les interruptions Masquables - Et les interruptions NON MASQUABLES ou NMI qui sont autorisées par l'instruction EI

- L'on retourne au programme principale après une interruption NON MASQUABLE avec RTN - pour les interruptions MASQUABLES avec RETI

MODES d'INTERRUPTION pour MASQUABLES

IM0 Avant de recevoir et d'accepter une interruption en MODE 0 le Z-80 se attend que le périphérique Externe qui donne une Instruction, un CALL ou RST - RST est une instruction de redémarrage

RST n / n = adresses : &0, &8, &10, &18, &20, &28 &30, &38 +

CALL et RST sauvegardent le PC sur la pile par PUSH AF/ PUSH BC / DE - / HL / IX - et IY)

et devrait revenir par POP IY, POP IX, POP HL, DE, BC, AF

IM1 Met le Mode d'interruption à 1 - idem que IM0 mais le Z80 fait un RST &38 avant d'accepter une Interrup MASQUABLE

IM2 Met le Mode d'interruption à 2 + Dans ce Mode après Instruction a chercher PC sauvé, le Z-80 sait à m'importe quelle case mémoire FAIRE - Donc le Bit le moins significatif est 0 Les 8 bits de plus significatif seront mis à l'avance dans le registre "I" + [illegible] Les 8 MSB de l'adresse sont envoyés

L'INTERRUPTION la plus prioritaire
est la "demande de BUS"
ou BXR BUSRQ

053742

sur le paramètre Interrupten. Cela permet en Z-80
de savoir à n'importe quelle des 128 adresses figurent
dans la Table en Mémoire dans une Table en
Mémoire commençant à l'adresse Octal fort
dans le registre "I" -

Le P.C

Ou Compteur Ordinal, ou Compteur de programme. C'est un registre 16 bits, qui contient l'adresse de la prochaine instruction à exécuter. Au début du programme le PC est fixé sur la 1ère case mémoire du programme (par ORG par exemple). Après la 1ère instruction le PC est mis à jour, pour pointer sur l'instruction suivante. Si le programme se fait de façon séquentielle, ligne après ligne, il indiquera l'adresse où il se trouve. Plus le nombre d'octets que réclame l'instruction. Ex: LD A, 83 = 2 Octets, si le programme est à l'adresse 36000, le PC indiquera 36000 + 2 = 36002. Mais s'il y a des branchements, des sauts par exemple (JP/JR etc..) le programme de démonstration indiquera après assemblage.

Ex:

Code Source	n° ligne	Case Mémoire	Code Objet Décimal
ORG 30000	10	7530	
LD A, 83	20	7530	3E/53
JP 30006	30	7532	C3/36/75
RET	40	7535	C9.
ADD A, 1	50	7536	C6/01.
CALL 47962.	60	7538	CD/5A/BB.
JP 30005	70	753B	C3/35/75

L'adresse de la case Mémoire est écrite en Hexa ; soit $(16^3 \times 7) + (16^2 \times 5) + (16^1 \times 3) + (16^0 \times 0) = 30000$ pour la 1ère ligne + (ligne 10). L'adresse augmente ligne par ligne avec le nombre d'Octets. Ex: Ligne 20 = 7530 + 2 (Octets) = 7532 - etc... jusqu'à 753B en Hexa = (voir 30011).

Lors de l'instruction JP 30006, le programme sautera à la case 7536 (en Hexa = 30006) adresse qui est indiquée dans la colonne Code Objet, mais dans la version OFA/OFO, il suffit d'inverser les Octets 36, 75 pour retrouver 75, 36 = 30006 en Décimal. C3 donne l'Opcode. Idem pour la JP 30005 ligne 70.

Valeur du PC au cours du Programme -

	Avant Instruction	Après Instruction
ORG 30000	?	30000
LD A, 83	30000	30002
JP 30006	30002	30006
ADD A, 1.	30006	30008
CALL 47962	30008	30011
JP 30005	30011	Retour à l'Assembleur
RET		

053746

Ligne 70 le PC indique l'adresse de saut 7535 (dans le cas OFA/OFO)

Le programme a été effectué avec des sauts Inconditionnels + (JP) + Pour le cas de sauts Conditionnels (JPc, JR etc...) il faut calculer l'OFFSET, Lorsque le saut réclame un retour en arrière +

Ex: pour dessiner une case dans coin haut gauche

Code Source Instructions	Adresse Case Mémoire	Code Objet Valeur
1o ORG 40960	A000	
2o LD HL, &C000	A000	21/00/C0
3o LD DE, &800	A003	11/00/08
4o LD (HL) &FF	A006	36/FF
5o ADD HL, DE	A008	19
6o JR NC, &A006	A009	30/FB ↑ offset.
7o RET	A00B	C9

Le PC après avoir lu JR NC, &A006, sur 40960 (ORG) plus les 11 octets du Code Objet 40960 + 11 = 40971 - à l'adresse &A009 - Pour revenir en arrière à l'adresse &A006, il lui faudra être décrémenté de 3 octets, celui de ADD HL DE soit (19), plus ceux 2 octets de LD (HL) &FF soit (36) et (FF), ce qui fait 3 octets en moins (−3) ainsi que les 2 octets de JRNC, A006 (30) (FB) + Ce qui fait −3 − 2 = −5.

−5 en Complément à 2 = 5. (déci) =

soit 5 = 00000101

11111010

+ 1

11111011 = 251 (Décimal)

et &FB en Hexadécimal -

Le PC pointera sur 40971 − 5 = 40966 = &A006 - (adresse de saut)

- Le Registre FLAG - ou F

7	6	5	4	3	2	1	0
S	Z		H		P/V	N	C

C'est un registre 8 bits, comme A, B, C, D, E, H, L, mais ses fonctions sont totalement différentes de celles d'un registre +
Chaque bit est utilisé comme indicateur, après une ou plusieurs instructions, opérations, dans l'ALU (unité de calcul) -
Bit 0 ou CARRY: Ce bit est mis (C=1) si une opération génère un report du bit 7 sur le bit 8 + Ce report (ou retenue) est placé (sauvegardé) dans ce Flag C (Carry). Ex: ADD: 10000001 + 10000010 = 100000011 (report du bit 7 sur bit 8 - C=1 -)
ADD A, B ~~A = 5E1 B = 8A2~~ -

Bit 1 - ou Flag de Soustraction N + N=1 si l'opération précédente est une soustraction ou INC -

Bit 2 ou Flag de PARITÉ ou de débordement - (overflow) -
PARITÉ d'un octet (après une opération) P=1 lorsque le Total de ses bits 1 est pair s'il le Total est impair P=0 -

DEPASSEMENT Lorsqu'il y a une retenue du bit 6 sur le bit 7, sans qu'il y ait retenue du bit 7 sur le bit 8 - (Carry).
Lorsqu'il y a ~~une telle retenue~~, ~~Bit 7 sur bit 8~~ - ~~une retenue~~ une retenue du bit 7 sur bit 8 (Carry) sans retenue bit 6 sur bit 7 - (Changement de Signe) -
- En bref: il ~~EX~~ sera positionné s'il y a report bit 6 sur bit 7 sans le Carry - V=1 -
il sera positionné s'il y a report vers l'extérieur (Carry) sans report 6 sur 7

Bit 4 ou Flag de 1/2 retenue H - utilisé par DAA en Décimale codé Binaire (DCB) si l'opération, Addition, ou Soustraction génère un report du bit 3 à l'Accumulateur

si ce report externe (Carry) ne vient pas d'un report interne du bit 6 au bit 7 -

Bit 6 ou Flag Zéro Z est mis si le résultat de l'opération = 0 -
donc si le résultat de l'opération = 0, Z=1 -
Ex: 00110000
- 00110000
00000000 Z=1 -
~~1111 0~~

Bit 7 ou Flag Signe - est mis S=1 si le nombre est NEGATIF
S=0 si nombre est positif.
~~EX ADD: 01100001 - (97)~~
~~01100101 = (101)~~
~~Bit Signe = 11001010 (202)~~
Ex: 01000000 (64)
SUB - 10000000 (-128)
11000000 = 192 FA0x
Bit Signe = 1 ou bien bit 8 = - et bit C = 64

PARITÉ ? pour avoir 4 significations - ?

1 - le total des bits 1 est pair : Ex: 01001000 = 2 / ou 00000011 = 2 (mais valeur 72 mais valeur 3)

2 - le total de Poids valeur est pair Ex = 01110000 = 112 / 00010010 = 18 -

3 - le total soit des bits 1, soit de Poids valeur est identique, égal à l'Octet Comparé, cherché etc... ?

Il semble que ce soit [1] Langage Machine p 150 ?

- idem pour Z-80 : Le bit de PARITÉ vérifie si l'Octet (& contraire) n'a pas été accidentellement changé - Le bit de PARITÉ est mis si le nombre de bits 1 (total des 1) est impair / si total des bits 1 d'un octet est impair le bit de parité, est mis en 8ème bit (?) pour que le total soit pair + Celle

4ème Signification : Comparaison entre 2 nombres égaux - P = 1 - (ainsi que Z = 1 ?! même coup).

Débordement V

Report Interne Lorsqu'il y report du bit 6 sur le bit 7 et donc changement accidentel de Signe Ex :

```
  0 1 0 0 0 0 0 0 0   (64)
+ 0 1 0 0 0 0 0 0 1   (65)
-------------------
  1 0 0 0 0 0 0 1 1 = (-127)
```

Externe

Bit Signe

V = 1 si report bit 6 sur 7 sans ~~report~~ Externe

V = 0 ~~si~~ le

V = 1 si report bit 7 sur bit 8, ~~avec~~ Carry (~~report~~ externe) si ce report ne vient pas d'un report bit 6 sur bit 7 -

SAUTS et BRANCHEMENTS - Conditionnels

JP condition / JR condition / CALL condition / RET cond /

Z =	Saute ou branche si	Zéro : Z = 1 -
NZ =	" "	Non Zéro : Z = 0
C =	" "	si Retenue : C = 1
NC	" "	Non Retenue : C = 0
PO	" "	si pas PARITÉ : P/V = 0
PE	" "	si PARITÉ : P/V = 1.
P	" "	si PLUS (+) : S = 0
M	" "	si MOINS (-) : S = 1 -

BIT de PARITÉ : (Z80 p22) et 23 et CHAPitre 8 !

Signifie PARITÉ PAIRE Ex: Calcul du bit de parité Paire de 00100011 = 3 "1" / Le bit de Parité Paire doit être egal à 1 pour que le total = 4 donc Pair / Resultat 10010011 - Le 1er "1" représente le bit de Parité, et 00100111, le identifient le Caractère -