

# Le Grand Livre du Langage Machine des SHARP PC

1245/51/60/61/62/80

1350/60

1401/02/03/21/50/60/75

2500

## S O M M A I R E

Introduction . . . . .	1
Organisation interne . . . . .	2
La CPU . . . . .	3
La ROM . . . . .	3
La RAM . . . . .	5
Instructions BASIC . . . . .	6
BASIC - Langage Machine . . . . .	9
L'Arithmétique binaire . . . . .	12
Les Opérateurs logiques . . . . .	21
La Numérotation hexadécimale . . . . .	23
Le Décimal codé binaire . . . . .	26
L'Architecture du SC-61860 . . . . .	27
La Programmation . . . . .	32
Les RAM des SHARP PC . . . . .	35
Adresses des Indicateurs . . . . .	36
Instructions Machine du SC-61860 . . . . .	38
Le Jeu d'Instructions . . . . .	42
ADB . . . . .	43
ADCM . . . . .	45
ADIA n . . . . .	46
ADIM n . . . . .	47
ADM . . . . .	48
ADN . . . . .	49
ADW . . . . .	51
ANIA n . . . . .	53
ANID n . . . . .	54
ANIM n . . . . .	55
ANMA . . . . .	56
CAL 1n . . . . .	57
CALL nm . . . . .	58
CPIA n . . . . .	59
CPIM n . . . . .	60
CPMA . . . . .	61
DATA . . . . .	62
DECA . . . . .	64
DECB . . . . .	65
DECI . . . . .	66

DECJ	67
DECK	68
DECL	69
DECM	70
DECN	71
DECP	72
DTJ	73
DX	75
DXL	76
DY	78
DYS	79
EXAB	81
EXAM	82
EXB	83
EXB	84
EXW	85
EXWD	86
FILD	87
FILM	88
INA	89
INB	91
INCA	92
INCB	93
INCI	94
INCJ	95
INCK	96
INCL	97
INCM	98
INCN	99
INCP	100
IX	101
IXL	102
IY	104
IYS	105
JP nm	107
JPC nm	108
JPNC nm	109
JPNZ nm	110
JPZ nm	111

## II.

JRCM n	112
JRCP n	113
JRM n	114
JRNCM n	115
JRNCP n	116
JRNZM n	117
JRNZP n	118
JRP n	119
JRZM n	120
JRZP n	121
LEAVE	122
LDD	123
LDM	124
LDP	125
LDQ	126
LDR	127
LIA n	128
LIB n	129
LIDL n	130
LIDP nm	131
LII n	132
LIJ n	133
LIP n	134
LIQ n	135
LOOP n	136
LP 1	137
MVB	138
MVBD	139
MVDM	140
MVMD	141
MVW	142
MVWD	143
NOPT	144
NOPW	145
ORIA n	146
ORID n	147
ORIM n	148
ORMA	149
OUTA	150

## III.

OUTB . . . . .	151
OUTC . . . . .	152
OUTF . . . . .	154
POP . . . . .	155
PTJ . . . . .	156
PUSH . . . . .	158
RA . . . . .	159
RC . . . . .	160
READ . . . . .	161
READM . . . . .	162
RTN . . . . .	163
SBB . . . . .	164
SBCM . . . . .	166
SBIA n . . . . .	167
SBIM n . . . . .	168
SBM . . . . .	169
SBN . . . . .	170
SBW . . . . .	172
SC . . . . .	174
SL . . . . .	175
SLW . . . . .	176
SR . . . . .	178
SRW . . . . .	179
STD . . . . .	181
STP . . . . .	182
STQ . . . . .	183
STR . . . . .	184
SWP . . . . .	185
TEST n . . . . .	186
TSIA n . . . . .	188
TSID n . . . . .	189
TSIM n . . . . .	190
TSMA (TSIP). . . . .	191
WAIT n . . . . .	192
WAITI (WAITJ) . . . . .	193
Applications pratiques - Programmes en LM . . . . .	194
ANNEXES . . . . .	211

## INTRODUCTION

Cet ouvrage s'adresse tout particulièrement aux utilisateurs désireux d'exploiter au mieux toutes les ressources des ordinateurs de poche Sharp grâce à la programmation en langage machine.

L'approche choisie pour cet ouvrage répondra, nous l'espérons, aux exigences de tous ceux désireux d'acquérir rapidement les mécanismes leur permettant d'élaborer leurs propres programmes et routines en langage d'assemblage (ou assembleur).

La première partie vous familiarisera avec l'arithmétique binaire. Indispensable à la programmation.

La seconde partie décrira la façon d'intégrer des sous-programmes assembleur dans des programmes eux-mêmes écrits en Basic, avant de traiter des adresses mémoires des variables de travail ainsi que des variables système associées.

Le contenu de l'exposé qui suit a été soigneusement vérifié et testé par de nombreux spécialistes Sharp. Cependant, il se peut qu'une erreur se soit glissée dans le texte et, par avance, nous vous prions de bien vouloir nous en excuser.

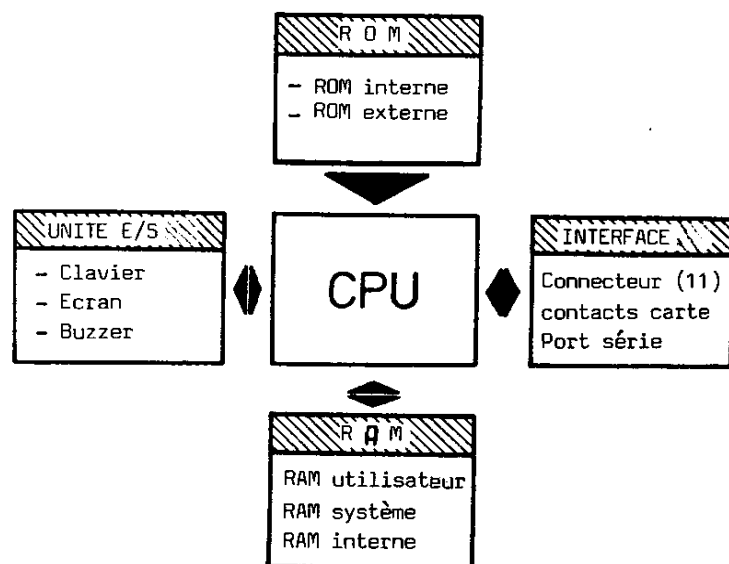
Ce livre n'aurait pu être écrit sans le concours efficace de la firme Sharp, qui nous a fait bénéficier d'une assistance précieuse et nous a autorisés à publier certaines données techniques, dont le lecteur passionné de hardware appréciera la valeur. Néanmoins, l'auteur n'assume aucune responsabilité quant à l'exactitude de ces informations.

Aachen, Avril 1986

Bernard Saretz  
Becker & Partner

## ORGANISATION INTERNE :

Voici un aperçu des différents sous-ensembles d'un ordinateur:



Les ordinateurs de poche Sharp comprennent un ensemble de circuits intégrés C-MOS qui présentent l'avantage d'une très haute intégration allié à une faible consommation d'énergie. La technologie MOS tire son appellation de ce qu'elle utilise des transistors à effet de champ à grille isolée, en anglais "metal oxide semiconductor". Les caractéristiques intrinsèques de la carte mère sont reprises dans le diagramme donné en annexe.

Description des principaux éléments:

## LA CPU

(Central Processing Unit = Unité centrale de traitement)

Les ordinateurs de poche intègrent un microprocesseur 8 bits: le SC 61860. La CPU correspond en quelque sorte au "cerveau" de l'ordinateur. C'est elle qui gère l'ensemble des fonctions offertes par l'ordinateur. Au fil de ce livre, vous apprendrez à "influencer" ou programmer la CPU au moyen du langage d'assemblage (ou assembleur).

Un générateur d'impulsion (horloge) rythme le travail de la CPU. Un quartz stabilise la fréquence des impulsions émises par l'horloge. Une fréquence de 576 KHz (p.ex. PC 1401/02) autorise le traitement de 576.000 valeurs binaires par seconde.

Les registres permettent à la CPU de communiquer, c'est-à-dire d'échanger des signaux avec la mémoire vive (RAM), la mémoire morte (ROM), l'écran ou les ports d'entrée/sortie. Il est intéressant de noter que la CPU fonctionne de façon permanente, l'interrupteur d'alimentation ON/OFF n'ayant qu'une action software. En position OFF l'affichage disparaît et le PC est placé dans une boucle sans fin. L'alimentation électrique de la CPU n'est cependant pas interrompue. Le contenu de la mémoire n'est donc pas perdu (=MEMORY SAFE GUARD). Le programme de gestion de la CPU est inscrit en ROM.

## LA ROM

(Read Only Memory = mémoire morte)

La mémoire ROM est répartie en deux zones distinctes:

La ROM interne: &0000-&1FFF  
00000-8191

La ROM externe: &8000-&FFFF  
32768-65535

#### La ROM interne:

Celle-ci contient le système d'exploitation, c'est-à-dire les routines écrites en langage machine qui assurent la gestion des ressources de l'ordinateur au niveau le plus bas. Sont ainsi inscrites en ROM les routines chargées du démarrage à froid et du démarrage à chaud. Une analyse approfondie de la ROM interne permet de dégager les routines suivantes:

- Routine d'effacement d'une zone déterminée de la RAM interne.
- Routine de transfert de valeurs entre différentes zones mémoire.
- Routine de mise en fonction de l'affichage.
- Routine de mise en fonction/hors fonction.
- Routine d'évaluation des données contenues dans diverses zones mémoire (p.ex. mémoire écran, tampon d'entrée/sortie).
- Routine de traitement des caractères tels que: &,.( )<>+ -\* /
- Routine de préparation des messages d'erreur.
- Routine de gestion du clavier.
- Routine de conversion des codes clavier en codes ASCII.
- Routine de conversion d'un nombre exprimé en virgule flottante (DCB) en un nombre entier, et inversement.
- Diverses routines utilisées par l'interpréteur Basic.

L'examen attentif du diagramme fourni en annexe permet de constater que la ROM interne est partie intégrante de la CPU. Ceci explique pourquoi l'instruction Basic "PEEK" ne peut lire la ROM interne. Toute tentative de lecture du contenu d'une adresse de la ROM interne renvoie la partie entière de la division de l'adresse mémoire spécifiée par 256 (octet de poids fort).

La fonction de l'instruction Basic "PEEK" sera explicitée en détail dans un chapitre ultérieur.

La ROM externe peut être lue à l'aide de l'instruction "PEEK" et renferme, entre autres, les routines et tables suivantes:

- Table des codes ASCII.
- Table de saut des instructions Basic.
- Routine d'instructions Basic (p.ex. PRINT, LPRINT...).
- Routine de sortie à l'écran ou sur imprimante.
- Routine de traitement des opérateurs mathématiques (\*,/ ,+ , -).
- Routine de gestion de l'éditeur ligne.

## **LA RAM**

---

(Random Access Memory = mémoire vive)

La mémoire RAM est répartie en plusieurs circuits intégrés d'une capacité de 2 Ko à 8 Ko (à l'exception des PC dont la capacité mémoire est limitée à 2 Ko: PC1401 (USA), 1245, 1250).

La RAM est divisée en trois zones:

- La zone programme
- La zone système
- La RAM interne

La zone programme renferme, comme son nom l'indique, vos programmes Basic ainsi les tables et tableaux définis.

La zone système regroupe les variables fixes et certaines données vitales gérées par le système, telles que l'adresse de début de Basic. Cette zone renferme également les informations relatives au statut du système d'exploitation. Enfin, nous y trouvons la zone mémoire écran et le tampon entrée/sortie.

Tout comme la ROM interne, la RAM interne est intégrée à la CPU. Cette mémoire vive comprend les registres, la pile ainsi que l'adresse des ports.

Contrairement aux mémoires mortes dont le contenu est inscrit de manière indélébile, l'utilisateur peut changer le contenu des mémoires vives au gré de sa fantaisie. Les instructions Basic "PEEK" et "POKE" permettent respectivement de lire ou d'écrire une valeur dans la case mémoire spécifiée. Il est à noter, qu'à l'image de la ROM interne, la RAM interne n'est pas accessible à ces instructions Basic.

## **INSTRUCTIONS BASIC**

### **1.) CLOAD M "(fichier)"; (adresse de début)**

Cette instruction charge en mémoire un programme en langage machine ou un bloc de données à partir de l'adresse de début spécifiée.

L'attribution d'un nom de fichier n'est pas obligatoire. En l'absence du paramètre de début de programme, cette adresse correspond à celle fournie lors de la sauvegarde par CSAVE M. L'adjonction du caractère "&" permet de saisir une adresse de début exprimée sous forme hexadécimale.

Abréviation: voir CLOAD (?)

Exemple: CLOAD "ML1";&4000

Charge le programme dénommé "ML1" à partir de l'adresse hexadécimale &4000 (=16384).

### **2.) CSAVE M "(fichier)"; (adresse de début), (adresse de fin)**

Cette instruction sauvegarde sur cassette la zone mémoire comprise entre les adresses de début et de fin de programme spécifiées. Cette instruction s'applique plus particulièrement aux programmes écrits en langage machine. Ici aussi, l'attribution d'un nom de fichier est optionnelle.

Abréviation: voir CSAVE

Exemple: CSAVE M "ML1";&4000,&40FF

Sauvegarde sur cassette, sous la dénomination "ML1", le programme résidant en RAM entre les adresses mémoire &4000 et &40FF.

### **3.) PEEK (adresse)**

Cette instruction fournit le contenu de l'adresse mémoire spécifiée (en numération décimale). Le microprocesseur 8 bits fait que la valeur contenue dans l'octet mémoire spécifié oscille toujours entre 0 et 255. L'instruction

PEEK est sans effet pour les adresses comprises entre 0 et 8191 (&0000 - &1FFF = ROM interne).

Abréviation: PE., PEE.

Exemple: PEEK 16384

Fournit le contenu de l'adresse mémoire 16384.

#### 4.) POKE (adresse),(valeur),(valeur),...

Cette instruction range à l'adresse mémoire spécifiée le contenu d'un octet. De façon similaire à PRINT, cette instruction accepte plusieurs arguments séparés par une virgule. L'adressage des mémoires ROM par le biais de l'instruction POKE n'est pas autorisé.

Abréviation: POK.

Exemple: POKE 17985,32,215

Affecte la valeur 32 à l'adresse 17985 et la valeur 215 à l'adresse 17986.

#### 5.) CALL (adresse)

Appelle un sous-programme écrit en langage machine et placé à l'adresse spécifiée. A la fin de la routine, le programme se repositionne au point de sortie Basic.

Abréviation: CA., CAL.

Exemple: CALL &4000

Exécute le programme en langage machine situé à l'adresse hexadécimale &4000.

## BASIC — LM

Ce chapitre se propose d'expliquer les fonctionnalités du langage machine. Pour ce faire, nous commencerons par opposer les avantages et désavantages respectifs du langage machine et du Basic.

### LIDP <-> PRINT

Ausgedruckt mit dem Sharp CE-140P

#### Basic:

- ☐ Indépendant du microprocesseur
- ☐ Interpréteur (traduit les commandes Basic en instructions machine)
- ☐ Variables (fixes et définissables)
- ☐ Messages d'erreur (ERROR...)
- ☐ Arithmétique en virgule flottante
- ☐ Editeur
- ☐ Bonne lisibilité des programmes
- ☐ Temps d'exécution accru
- ☐ La sortie 11 broches ne peut être adressée que par l'instruction LPRINT.
- ☐ Aucune influence sur les autres sorties.

#### Langage machine:

- ☐ Lié au microprocesseur. Chaque type de microprocesseur utilise son propre jeu d'instructions, ses propres registres.

- Aucun message d'erreur prédéfini (=planté)
- Nombre de registres fixe (mémoire)
- Arithmétique sur 8 ou 16 octets
- Immédiatement exploitable (sans interpréteur)
- Vitesse d'exécution accrue
- Gestion directe du clavier et des interfaces
- Aucune fonction d'édition
- Lisibilité réduite des programmes

A première vue, il est clair que le confort d'utilisation penche en faveur du Basic. L'utilisation du langage machine ne se justifie que pour les applications où la vitesse d'exécution joue un rôle primordial. Elle se justifie également pour les programmes proches de la machine et qui ne peuvent être construits en Basic. La plupart des programmeurs intègrent des sous-programmes assembleur dans des programmes écrits eux-mêmes en Basic. Ceci afin de maintenir le temps de développement dans des limites raisonnables.

Comment le microprocesseur différencie-t-il les instructions Basic des instructions assembleur?

Non seulement les caractères et les chiffres sont codés sous forme d'une table Sharp-ASCII mais également les instructions Basic (token)

Exemple:

READ = 219 (décimal) = DE (hexadécimal)

A chaque instruction assembleur est associé bijectivement un code binaire. Il faut noter que le microprocesseur ne reconnaît que cette suite de nombres binaires, qui transposée sous forme de signaux électriques, constitue le fondement de son langage maternel. Le programmeur n'utilise que très rarement le code des instructions en nombres binaires. Il a à sa disposition d'autres codes, plus faciles à manipuler, où les suites binaires sans signification apparente sont remplacées par des symboles plus mnémoniques, les instructions assembleur. Un programme d'assemblage, encore appelé un "assembleur", effectue la conversion du langage mnémonique en langage machine (nombres binaires). Comme dans le cas de la transformation instruction Basic -> token, cette conversion est bijective. L'instruction assembleur ci-dessous possède le même code que l'instruction Basic "READ":

EXAM = 219 (décimal) = DE (hexadécimal)

Ceci signifie que chaque code ASCII a une double signification. Il correspond à une instruction Basic et une instruction assembleur. L'instruction qui lance le programme ou la routine permet au microprocesseur de faire la distinction. Après "RUN", les lignes de code sont interprétées comme des instructions Basic. "CALL" initialise une série d'instructions assembleur.

Tout comme les instructions Basic, les instructions assembleur ne constituent généralement pas une entité mais nécessitent un, deux voire trois arguments. Si nous prenons l'instruction Basic "STR\$(1)" comme exemple, "STR\$" représente une instruction dont "(1)" est l'argument.

Le nombre d'arguments associés aux instructions assembleur est bien défini.

Il existe:

- Les instructions codées sur 1 octet, sans argument
- Les instructions codées sur 2 octets, avec 1 argument
- Les instructions codées sur 3 octets, avec 2 arguments



- Les instructions codées sur 4 octets, avec 3 arguments

L'argument est un nombre dont la valeur est comprise entre 0 et 255.

Nous allons à présent aborder l'arithmétique binaire, dont il est nécessaire de connaître les bases pour programmer en langage assembleur.

## L'ARITHMETIQUE BINAIRE

Si de votre côté vous évoluez dans un univers à 10 chiffres (décimal), l'ordinateur, lui n'en reconnaît que deux (binaire). Un *élément binaire* (on rencontre plus fréquemment le terme "bit") est un élément qui ne peut prendre que deux états, l'état 1 ou l'état 0. Un *mot de microprocesseur* représente la longueur du mot binaire que le microprocesseur peut traiter. Celui d'un ordinateur de poche Sharp fait 8 bits. Un tel mot est un *octet* (byte).

$$01 + 01 = 10$$

En électronique numérique, les signaux électriques ne peuvent prendre que deux valeurs symbolisées par "0" et "1". La valeur binaire "0" est associée au niveau de tension le plus bas (moins de 2 volts). Inversement, la valeur binaire "1" est associée au niveau de tension le plus élevé (plus de 2 volts). Un mot de 8 bits peut représenter des valeurs comprises entre 0 et 255 ( $2^8 = 255$ ; 2 représente le nombre de possibilités et 8 le nombre de positions).

Nous sommes à présent en mesure de définir le tableau suivant:

BINAIRE	CONVERSION	VALEUR DECIMALE
00000000	= 0	= 0
00000001	= $2^0$	= 1
00000010	= $2^1$	= 2
00000011	= $2^0 + 2^1$	= 3
00000100	= $2^2$	= 4
00000101	= $2^2 + 2^0$	= 5
00000110	= $2^2 + 2^1$	= 6
00000111	= $2^2 + 2^1 + 2^0$	= 7
00001000	= $2^3$	= 8
.	.	.
11111111	= $2^7 + 2^6 + 2^5 + \dots + 2^1 + 2^0$	= 255

### Définition:

1<sup>ère</sup> à n<sup>ème</sup> position:  $(base^{(n-1)}) * \text{chiffre} \Rightarrow M+$

### Exemple: 1011

1<sup>ère</sup> position:  $(2^0) * 1 = 1 \Rightarrow M+ \Rightarrow 1$   
 2<sup>ème</sup> position:  $(2^1) * 1 = 2 \Rightarrow M+ \Rightarrow 3$   
 3<sup>ème</sup> position:  $(2^2) * 0 = 0 \Rightarrow M+ \Rightarrow 3$   
 4<sup>ème</sup> position:  $(2^3) * 1 = 8 \Rightarrow M+ \Rightarrow 11$

=> binaire: 1011 = 11 décimal!

Le programme Basic ci-dessous réalise la conversion binaire-décimal et décimal-binaire.

DEF "A" : BIN. -> DEC.

DEF "Z" : DEC. -> BIN.

### Listing:

```

10: "A" CLEAR : INPUT "NOMBRE POSITIONS:"
    ;S: DIM B$(0)*S
20: INPUT "NOMBRE BINAIRE:";B$(0)
30: FOR I=1 TO S: IF VAL (MID$ (B$(0),I
    ,1))=1 LET X=X+2^(S-I)
40: NEXT I
50: PRINT "=>";X: END
100: "Z" CLEAR : INPUT "NOMBRE POSITIONS:"
  
```

```

;S: DIM B$(0)*S
110: INPUT "NOMBRE DECIMAL:";A
115: IF 2^S<=A PRINT "ERREUR": GOTO 100
120: FOR I=S-1 TO 0 STEP -1
130: IF A/2^I>=1 LET B$(0)=B$(0)+"1": A=A
    -2^I: GOTO 150
140: B$(0)=B$(0)+"0"
150: NEXT I
160: PRINT B$(0)

```

Représentation exponentielle d'un octet:

$2^7$	$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$
-------	-------	-------	-------	-------	-------	-------	-------

Nous allons à présent aborder le problème des opérations élémentaires en arithmétique binaire.

#### Addition binaire:

L'exemple suivant additionne deux nombres à 8 bits: "100" et "25"

```

100 : 0 1 1 0 0 1 0 0
+25 : 0 0 0 1 1 0 0 1
125 = 0 1 1 1 1 1 0 1

```

Vous remarquerez que le procédé fonctionne de la même manière que celui employé pour l'addition décimale. Un plus Zéro donne Un. Un plus Un donne Zéro avec Un de retenue.

Pour additionner deux chiffres binaires, procéder comme suit:

```

0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 0 plus 1 de retenue
        = 10

```

L'exemple ci-dessous additionne les nombres quatre et douze

```

  4 = 0 0 0 0 0 1 0 0
+12 = 0 0 0 0 1 1 0 0
retenue: 1 1
16 = 0 0 0 1 0 0 0 0

```

La retenue ("CARRY") est régie par les mêmes règles que celles de l'addition décimale.

#### Soustraction binaire:

Le système binaire autorise deux types de soustraction:

##### a) La soustraction directe:

Pour soustraire deux chiffres binaires positifs, procéder comme suit:

```

0 - 0 = 0
1 - 0 = 1
1 - 1 = 0

```

Exemple: Soustraire le nombre cinq du nombre neuf.

```

  9 = 0 0 0 0 1 0 0 1
-5 = 0 0 0 0 0 1 0 1
4 = 0 0 0 0 0 1 0 0

```

Ceci implique que le microprocesseur maîtrise à la fois les règles de l'addition et celles de la soustraction binaires. Il serait plus aisé de transformer l'opération de soustraction en une simple addition. C'est le sujet que nous allons aborder avec le système du complément à deux.

##### b) Complément à deux

La soustraction est transformée en une simple opération d'addition. Le nombre à soustraire est considéré de signe négatif et complémenté. Ce qui revient à remplacer les 1 par des 0 et vice et versa. Le huitième bit est utilisé comme bit de signe (sign bit). Ce système permet également de représenter 256

nombre différents compris entre -127 et +127.

Notation:

+/-	2 <sup>6</sup>	2 <sup>5</sup>	2 <sup>4</sup>	2 <sup>3</sup>	2 <sup>2</sup>	2 <sup>1</sup>	2 <sup>0</sup>
-----	----------------	----------------	----------------	----------------	----------------	----------------	----------------

Exemple: Il s'agit, comme dans le premier exemple, de soustraire le nombre cinq du nombre neuf. Dans un premier temps, le chiffre binaire est complété:

$$\begin{array}{r} 5 = 0\ 0\ 0\ 0\ 0\ 1\ 0\ 1 \\ -5 = 1\ 1\ 1\ 1\ 1\ 0\ 1\ 0 \end{array}$$

↑
↑  
 signe      complément

Le bit de signe permet de différencier le chiffre négatif -5 du nombre positif 122 (=01111010). L'opération se réduit à présent à une simple addition:

$$\begin{array}{r} 9 = 0\ 0\ 0\ 0\ 1\ 0\ 0\ 1 \\ -5 = 1\ 1\ 1\ 1\ 1\ 0\ 1\ 0 \\ \hline \text{retenue : } 1\ 1\ 1\ 1 \\ (1)0\ 0\ 0\ 0\ 0\ 0\ 1\ 1 \\ \hline \phantom{000000} + 1 \\ \phantom{000000} 1\ 1\ 0 \\ \hline 4 = 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0 \end{array}$$

Il est à noter que le bit de débordement (1) est additionné au premier bit.

Remarque:

Le microprocesseur n'additionne pas les séries de nombres d'un seul trait mais deux à deux.

Exemple: 15	Processeur: 15	
+ 4	+ 4	19
+ 19	= 19	+ 19
+ 3		38
= 41		+ 3
		= 41

La vitesse de traitement du microprocesseur autorise ce genre de manipulation. Le résultat est toujours obtenu dans un délai très court.

Multiplication binaire:

Pour multiplier deux chiffres binaires, procéder comme suit:

$$\begin{array}{l} 0 * 0 = 0 \\ 0 * 1 = 0 \\ 1 * 0 = 0 \\ 1 * 1 = 1 \end{array}$$

L'exemple ci-dessous multiplie le nombre 15 par 13. La multiplication s'effectue de la façon suivante en mode décimal:

$$\begin{array}{r} 15 * 13 \\ \phantom{00} 15 \\ + \phantom{00} 45 \\ \hline 195 \end{array}$$

Il s'agit d'exécuter la même opération en mode binaire. Afin de rendre notre exemple plus simple, nous représenterons les nombres décimaux sur 4 bits et non pas sur un octet. Les quatre bits de poids fort étant à 0.

$$\begin{array}{r} 1111 * 1101 \\ \phantom{0000} 1111 \\ + \phantom{0000} 1111 \\ \hline 101101 \\ + \phantom{0000} 0000 \\ \hline 1011010 \\ + \phantom{0000} 1111 \\ \hline 11000011 = 195 \text{ (décimal)} \end{array}$$

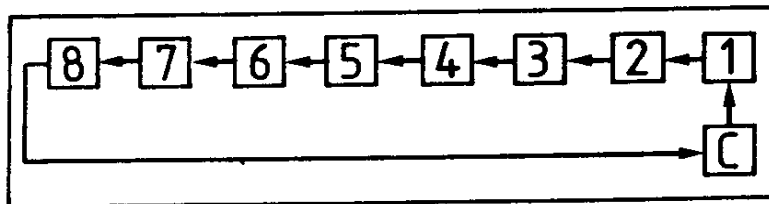
Dans un programme écrit en assembleur, la multiplication est réalisée la plupart du temps par additions successives.

Exemple:  $13 * 3 = 39$   
 $13 + 13 + 13 = 39$

A nouveau, la vitesse de traitement des opérations autorise ce type de manipulations. La multiplication s'effectue en mode binaire comme suit:

0 0		0 0 0 0 0 0 0 0
+ 1 3	1.Addition	0 0 0 0 1 1 0 1
-----		-----
1 3		0 0 0 0 1 1 0 1
1 3		0 0 0 0 1 1 0 1
+ 1 3	2.Addition	0 0 0 0 1 1 0 1
-----		-----
2 6		0 0 0 1 1 0 1 0
2 6		0 0 0 1 1 0 1 0
+ 1 3	3.Addition	0 0 0 0 1 1 0 1
-----		-----
3 9		0 0 1 0 0 1 1 1

Un aspect important de la multiplication est représenté par la *rotation* à gauche. Noter la "retenue" (dernière position  $1 + 1 = 0 + \text{retenue} = 1$ ) lorsqu'il y a débordement sur la dernière position significative de l'octet.



L'unité centrale de traitement dispose d'un bit d'état ("Carry-Flag") particulier pour enregistrer ce type de retenue. Cet indicateur de retenue est également mis à 1 si une addition ou une sous-

traction produit une retenue. L'exemple suivant explicite une multiplication par 2:

Exemple:

13	=	0 0 0 0 1 1 0 1
26	=	0 0 0 1 1 0 1 0
52	=	0 0 1 1 0 1 0 0
104	=	0 1 1 0 1 0 0 0
208	=	1 1 0 1 0 0 0 0
160 + 256	=	416 = 1 0 1 0 0 0 0 0

Dans la dernière addition, il s'agit bien d'additionner "256" car le bit de retenue (Carry flag) est positionné à 1. Cependant, ce nombre ne peut être représenté sur 8 bits. Le microprocesseur afficherait 160, le bit de retenue n'étant pas pris en compte lors d'une addition sur 8 bits. Le microprocesseur Scharp dispose d'un jeu d'instructions qui permet d'effectuer additions et rotations. Cependant, il ne fournit aucune instruction de multiplication. Toute multiplication doit être convertie en une suite d'additions ou de rotations.

### La division binaire

Le microprocesseur ne peut réaliser directement des divisions. Nous pouvons utiliser les instructions de soustraction pour diviser des nombres entiers. Cette limitation s'applique également à la multiplication. Nous parlerons dans un chapitre ultérieur du système DCB (Décimal Codé Binaire) qui permet de traiter les nombres exprimés en virgule flottante.

Exemple de division: 39 par 13

$$39 / 13 = 3$$

$$39 - 13 - 13 - 13 = 0$$

Si nous soustrayons par *trois* fois le nombre 13 de 39, le résultat est nul. La fréquence des soustractions détermine le résultat de la division. Le processus en système binaire est le suivant:

1ère soustraction:

```

  39 =      0 0 1 0 0 1 1 1
- 13 =      1 1 1 1 0 0 1 0
-----
  26 = (1) 0 0 0 1 1 0 0 1  (=retenue)
                1
            -----
            0 0 0 1 1 0 1 0

```

2ème soustraction:

```

  26 =      0 0 0 1 1 0 1 0
- 13 =      1 1 1 1 0 0 1 0
-----
  13 = (1) 0 0 0 0 1 1 0 0
                1
            -----
            0 0 0 0 1 1 0 1

```

3ème soustraction:

```

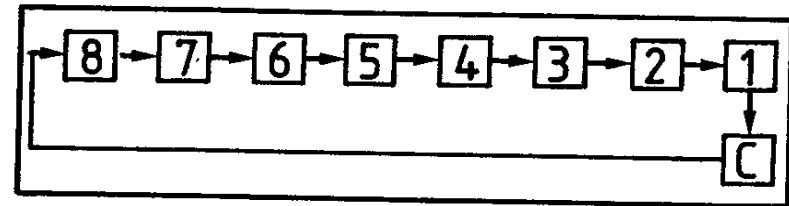
  13 =      0 0 0 0 1 1 0 1
- 13 =      1 1 1 1 0 0 1 0
-----
   0 = (1) 1 1 1 1 1 1 1 1
                1
            -----
            0 0 0 0 0 0 0 0

```

En programmation, on définit une variable-compteur incrémentée d'une unité à chaque nouvelle soustraction. A la fin de l'exécution du programme, cette variable contient le résultat de la division. Dans l'exemple ci-dessus, le reste est égal à zéro. Ce n'est qu'un cas particulier de la division. La variable-compteur ne donne que la partie entière du résultat de la division de deux nombres.

Exemple:  $13/5 = 2$  reste 3

De façon identique à une multiplication, une division binaire peut être effectuée par une succession de rotations. Cette fois, vers la droite.



Exemple:  $208 = 1\ 1\ 0\ 1\ 0\ 0\ 0\ 0$   
 $104 = 0\ 1\ 1\ 0\ 1\ 0\ 0\ 0$

## LES OPERATEURS LOGIQUES

### L'opérateur booléen ET

Chacun des bits de deux nombres sont comparés en fonction de la table de vérité définie ci-dessous et produisent comme résultat numérique soit 1, soit 0 suivant que la relation est vraie ou fausse.

```

0^0 = 0
0^1 = 0
1^0 = 0
1^1 = 1

```

Le ET de deux nombres à un chiffre binaire à 1 pour chaque position où les deux nombres ont un chiffre binaire égal à 1. Ce type de logique existe dans le langage familier.

Exemple: Un ordinateur de marque XY est d'utilisation confortable, son microprocesseur est rapide et il est compatible avec d'autres systèmes.

Expression A: L'ordinateur est lent (faux)  
 Expression B: L'ordinateur est confortable (vrai)

L'assertion "L'ordinateur est lent et confortable" est fausse. Seule la relation "L'ordinateur est rapide et confortable" est vraie. Cette relation transposée à deux nombres produit le résultat suivant:

$$\begin{array}{r} 9 = 0\ 0\ 0\ 0\ 1\ 0\ 0\ 1 \\ \wedge \\ 13 = 0\ 0\ 0\ 0\ 1\ 1\ 0\ 1 \\ \hline 9 = 0\ 0\ 0\ 0\ 1\ 0\ 0\ 1 \end{array}$$

Cette opération peut s'effectuer en Basic. Saisir "9 AND 13" en mode RUN. L'ordinateur affiche également le résultat "9". Ceci prouve que le microprocesseur fonctionne en mode binaire. Le langage machine dispose d'un jeu d'instructions logiques, comme Basic, correspondant au ET et au OU.

#### L'opérateur booléen OU

Ce type de logique existe également dans le langage familier. Dans l'exemple décrit ci-dessus, nous pouvons affirmer que l'expression "L'ordinateur est lent ou confortable" est vraie, car un membre de l'expression est vrai. Ce qui nous permet de définir la table de vérité suivante:

$$\begin{array}{l} 0 \vee 0 = 0 \\ 0 \vee 1 = 1 \\ 1 \vee 0 = 1 \\ 1 \vee 1 = 1 \end{array}$$

Remarque: Les signes " $\wedge$ " et " $\vee$ " correspondent respectivement aux relations ET et OU.

Exemple:

$$\begin{array}{r} 9 = 0\ 0\ 0\ 0\ 1\ 0\ 0\ 1 \\ \vee \\ 13 = 0\ 0\ 0\ 0\ 1\ 1\ 0\ 1 \\ \hline 13 = 0\ 0\ 0\ 0\ 1\ 1\ 0\ 1 \end{array}$$

"9 OR 13" réalise cette opération en Basic. Bien entendu, le résultat obtenu est égal à 13.

L'informatique connaît un troisième opérateur OU-exclusif, pour lequel le microprocesseur ne dispose d'aucune instruction. Le OU-exclusif à un chiffre binaire à 1 là où les deux nombres ont des chiffres binaires de valeurs opposées; toutes les autres positions sont à zéro.

De l'électronique numérique, qui réalise de telles fonctions logiques à l'aide de portes, l'informatique a également hérité les opérateurs ET-NON (NAND) et OU-NON (NOR). Il s'agit des fonctions ET et OU complémentées (inversées).

L'opérateur logique ET est fréquemment utilisé comme masque logique. Il sert principalement à forcer à 0 (masquer) certains bits. Ne restent à 1 que les chiffres binaires où le second nombre a des chiffres binaires positionnés à 1.

## LA NUMEROTATION HEXADÉCIMALE

Nous allons à présent aborder une nouvelle base de numérotation, l'hexadécimal. Ce système autorise une simplification dans la représentation des nombres binaires. Il n'est pas facile de se souvenir d'un nombre exprimé sous forme d'une suite de 1 et de 0. Exprimé en hexadécimal, il sera plus aisé à manipuler, à retenir, sans pour autant présenter de difficulté de conversion. Il existe une réelle relation entre un nombre binaire et son équivalent hexadécimal. Les bits sont groupés par quatre, ce qui est commode lorsque l'ordinateur est un modèle à 8 bits. Chaque groupe a un poids compris entre 0 et 15 ( $2^4$  possibilités). L'hexadécimal comporte donc seize symboles différents. Les valeurs 10 à 15 sont représentées par les symboles A à F. Pour mieux saisir ce système de numérotation, reportez-vous à la table ci-dessous.

<u>DECIMAL</u>	<u>HEXADECIMAL</u>	<u>BINAIRE</u>
00	0	0 0 0 0
01	1	0 0 0 1
02	2	0 0 1 0
03	3	0 0 1 1
04	4	0 1 0 0
05	5	0 1 0 1
06	6	0 1 1 0
07	7	0 1 1 1
08	8	1 0 0 0
09	9	1 0 0 1
10	A	1 0 1 0
11	B	1 0 1 1
12	C	1 1 0 0
13	D	1 1 0 1
14	E	1 1 1 0
15	F	1 1 1 1

Tous les ordinateurs de poche Sharp exécutent la conversion de nombres hexadécimaux (précédés du caractère "&") en nombres décimaux. Les ordinateurs de poches de la série 14xx exécutent certaines opérations en numération hexadécimale. Le programme Basic ci-dessous permet de générer la valeur hexadécimale d'un nombre décimal, et inversement. Il transpose également la valeur hexadécimale dans une variable chaîne de caractères.

#### Programme HEX <-> DEC

DEF"A" : HEX vers DEC  
DEF"B" : DEC vers HEX

#### Listing:

```
10:"A" CLEAR : DIM B$(0) * 80
20:INPUT "NBRE HEX:";B$(0):S=LEN B$(0)
30:FOR I=1 TO S
32:A=ASC (MID$(B$(0),I,1))
35:A=A-48-(A>64)*7
45:X=X+A*16^(S-I)
```

```
50:NEXT I
55:PRINT "=>";X:END
100:"Z" CLEAR :DIM B$(1)*80
110:INPUT "NBRE DEC:";B$(1):S=LEN B$(1)
:A=VAL B$(1)
120:FOR I=S-1 TO 0 STEP -1
125:N= INT (A/16^I)
130:IF N>9 LET B$(0)=B$(0)+CHR$ (55+N):
GOTO 155
140:B$(0)=B$(0)+STR$ N
155:A=A-N*16^I
160:NEXT I
170:PRINT "=>";B$(0)
```

Pour convertir manuellement un nombre hexadécimal, il faut procéder comme suit:

1<sup>è</sup> à la n<sup>ième</sup> pos.:  $16^{(n-1)} \times \text{chiffre} \Rightarrow M+$   
(de droite à gauche)  
Exemple: E A 6 0

1<sup>ère</sup> pos.:  $(16^0) \times 0 = 0 \Rightarrow M+ \Rightarrow 0$   
2<sup>ème</sup> pos.:  $(16^1) \times 6 = 96 \Rightarrow M+ \Rightarrow 96$   
3<sup>ème</sup> pos.:  $(16^2) \times 10 = 2560 \Rightarrow M+ \Rightarrow 2656$   
4<sup>ème</sup> pos.:  $(16^3) \times 14 = 57344 \Rightarrow M+ \Rightarrow 60000$

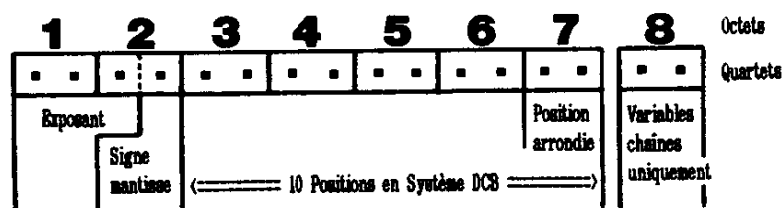
E A 6 0 = 60000 (décimal)

Lorsque nous appréhendons un nouveau système de numération, les nombres hexadécimaux par exemple, une impulsion naturelle nous pousse à convertir les nombres dans une base plus familière, à effectuer les opérations nécessaires et, enfin, à convertir à nouveau le résultat en base 16. Ce procédé est très coûteux en temps et conduit, bien souvent, au rejet du système hexadécimal. C'est pourquoi nous pensons qu'il est préférable de s'appliquer à travailler directement avec les nombres hexadécimaux.

## LE DECIMAL CODE BINAIRE

Le microprocesseur représente les nombres en virgule flottante exprimés en décimal codé binaire (DCB). Comme la numérotation hexadécimale, le DCB travaille par groupe de 4 bits. Cependant le microprocesseur n'exploite que 10 chiffres (0-9) parmi les 16 combinaisons possibles.

**Représentation DCB:**



Ainsi une variable sur 8 octets peut contenir des nombres allant de  $1 \cdot 10^{-99}$  à  $9.99999999 \cdot 10^{99}$ .  
l'exemple ci-dessous mon-tre la représentation interne de nombres exprimés en DCB.

**Example:**

nombre en virgule 1. 2. 3. 4. 5. 6. 7. 8.  
flottante

1.23456789	E 99	09	90	12	34	56	78	90	00
1.23456789	E-99	09	10	12	34	56	78	90	00
-1.23456789		08	88	12	34	56	78	90	00
1234		-00	30	12	34	00	00	00	00
987.65		00	20	98	76	50	00	00	00

Ces valeurs peuvent être lues, exprimées en hexadécimal, dans une variable fixe. Il faut remarquer que puisque le microprocesseur n'exploite que 10 chiffres parmi les 16 possibles, un ajustage est appliqué lorsque le résultat dépasse la valeur 9. L'exemple suivant additionne deux nombres et

ajuste le résultat. L'utilisation de ce type de représentation n'est pas fréquente en langage machine.

**Example:**

$$\begin{array}{r} 5 = 0 \ 1 \ 0 \ 1 \\ + 7 = 0 \ 1 \ 1 \ 1 \\ \hline 12 = 1 \ 1 \ 0 \ 0 \\ (+ 6) = 0 \ 1 \ 1 \ 0 \\ \hline 0 \ 0 \ 0 \ 1 \quad 0 \ 0 \ 1 \ 0 \end{array} \quad \begin{array}{l} \\ \\ \\ \\ \hline > 9! \end{array}$$

Le microprocesseur possède plusieurs instructions d'opérations arithmétiques en décimal codé binaire.

# L'ARCHITECTURE DU SC 61860

Avant d'aborder la programmation en langage machine, nous allons approfondir l'étude des composantes internes du microprocesseur SC 61860. La figure ci-dessous donne une vision synthétique du microprocesseur (page 29).

La ROM interne et d'autres composantes sont détaillées dans le diagramme repris en annexe. Le programmeur en Basic à l'habitude d'utiliser les variables. Les ordinateurs de poche Sharp reconnaissent les variables fixes, les variables définies par l'utilisateur et les variables de tableau. Ces variables renferment aussi bien des valeurs numériques, que des chaînes de caractères. Il est également possible d'effectuer sur ces variables diverses opérations mathématiques ou logiques.

Ce confort d'utilisation est quelque peu réduit en langage machine:

Le microprocesseur ne connaît que les registres, placés dans la RAM interne. Chacun d'eux peut être



adressé séparément (adresse 00 à 0B). Nous traiterons des adresses 0C à 5F de la RAM interne dans un paragraphe ultérieur.

Les registres sont tous de 8 bits (il peuvent contenir des valeurs allant de 0 à 255). D'autres registres sont en dehors de la RAM interne, mais sont utilisés de façon spécifique.

Les registres de la RAM interne:

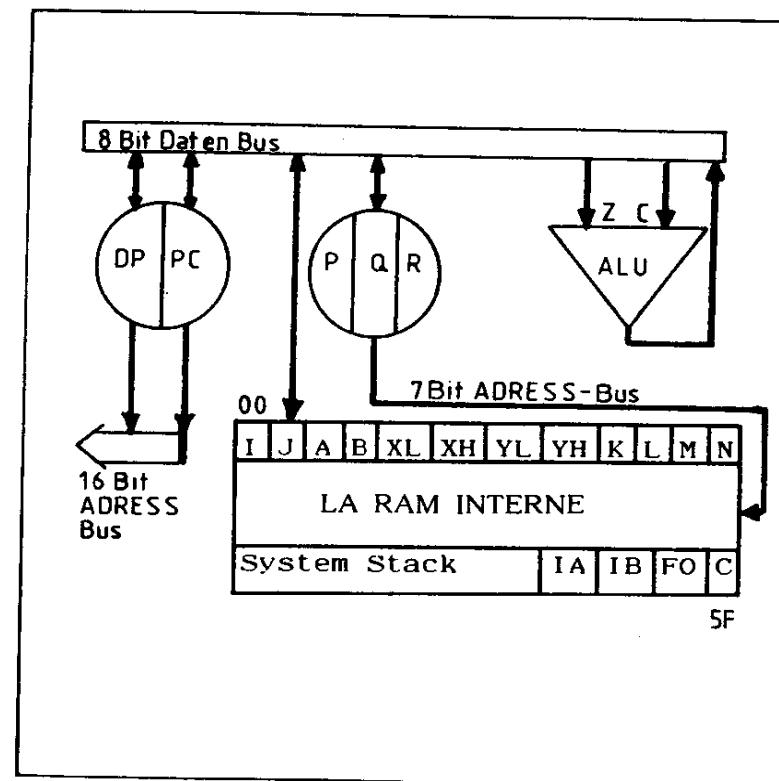
Adresse	Registre
00	I
01	J
02	A
03	B
04	XL
05	XH
06	YL
07	YH
08	K
09	L
0A (10)	M
0B (11)	N

Il y a deux registres d'index. Leur nom respectif est I et J. Ils servent en général de compteur (telle la variable I dans la boucle FOR-NEXT: "FOR I=0 TO 100"). Ces registres sont utilisés dans les transferts de tableaux et sont décrémentés (diminués d'une unité) jusqu'à ce que leur valeur soit égale à &FF (ex: 04 03 02 01 00 FF). Il existe également un registre de boucle D (non repris dans la figure) qui est utilisé lorsqu'on désire conserver la valeur initiale des registres I et J.

L'interpréteur Basic utilise le registre J. Ceci peut provoquer des perturbations dans l'exécution d'un programme lorsqu'un sous-programme en langage machine modifie la valeur contenue dans le registre J. Aussi est-il préférable de travailler exclusivement avec le registre I.

Le registre A, d'adresse 02, est appelé accumulateur. Il intervient dans tous les mouvements

d'information ainsi que dans les manipulations logiques et arithmétiques.



Le registre B, d'adresse 03, est le complément du registre A. Il dispose de fonctionnalités identiques à celles de l'accumulateur (registre A). Ces deux registres sont quelques fois vus comme un seul registre de 16 bits. Ce qui permet de gérer des valeurs allant jusqu'à 65535.

Ce principe est également appliqué aux registres X et Y.

XL, d'adresse 04, correspond à la partie basse (Low) du registre X, XH est la partie haute (High). Cette segmentation est utilisée pour représenter les a-

dresses système (adresse de début de Basic, par exemple).

Les registres X (XL=04, XH=05) et Y (YL=06, YH=07) sont également appelés registres adresse car ils sont utilisés pour stocker l'adresse de l'information manipulée dans l'accumulateur. Le microprocesseur dispose d'instructions spécifiques, faisant appel au registre X, lorsqu'il s'agit de transférer des données vers l'accumulateur, à partir d'une adresse mémoire de 16 bits. Ces instructions sont identiques à l'instruction BASIC "PEEK", à la différence près que la valeur est placée dans l'accumulateur.

Le registre Y effectue l'opération inverse, il copie les informations de l'accumulateur vers l'adresse mémoire.

Les registres K, L, M, et N sont généralement utilisés comme compteurs ou comme registres à usage général.

La RAM interne renferme des registres d'entrée/sortie aux adresses 5C, 5D, 5E, et 5F. Ces registres envoient des données par les ports d'entrée/sortie. Le microprocesseur dispose à cet effet des instructions OUTA, OUTB, OUTF, et OUTC.

La RAM interne contient également la pile système. Elle débute à l'adresse 5B et grandit vers les adresses inférieures. Le rôle de la pile est, par exemple, de stocker lors de l'exécution d'un sous-programme (instruction CALL). Cette instruction incrémente la pile ("empile") deux fois, tandis que l'instruction RTN (retour de sous-programme) décrémente la pile ("dépile") deux fois. Les instructions PUSH et POP empilent et dépilent des valeurs.

Les registres sur 7 bits P, Q, et R sont placés en dehors de la pile.

Le registre R est le pointeur de pile. Il donne en permanence l'adresse du sommet de la pile. La

méthode d'accès à la pile est une méthode LIFO (Last In, First Out), ce qui signifie: "dernier entré, premier sorti", c'est-à-dire que l'on dépile dans l'ordre inverse duquel on a empilé. Les 7 bits du registre R (au lieu des 8 bits généralement utilisés) lui permettent d'adresser l'ensemble de la RAM interne. Ceci vaut également pour les registres P et Q.

Les registres P et Q servent à adresser la RAM interne. Le diagramme donné en annexe montre que les registres P et Q accèdent à la RAM interne par un bus d'adresse à 7 bits.

Le microprocesseur dispose d'un second bus d'adresse à 16 bits qui permet d'adresser la RAM externe et d'un bus de données à 8 bits (les lignes de circulation entre les différents organes d'un ensemble à microprocesseur sont appelées des bus). Le bus d'adresse pointe vers les cases de la mémoire, tandis que le bus de données assure le transport des données vers l'intérieur ou l'extérieur du microprocesseur. Avec ses 8 bits, le bus de données peut véhiculer 256 informations différentes.

Le registre DP (16 bits) adresse l'espace mémoire des ROM et RAM externes. L'abréviation DP correspond au terme anglais "Data Pointer". Ceci explique l'utilisation faite de ce registre.

Le registre PC, de l'anglais "Program Counter" (compteur de programme), donne l'adresse de la prochaine instruction à exécuter. Il s'agit également d'un registre sur 16 bits, ce qui permet d'adresser un espace de 65535 octets. A la suite de l'exécution d'une instruction, le registre PC est incrémenté d'une unité. Les instructions CALL et JUMP modifient la valeur du registre PC.

Enfin, le microprocesseur renferme un organe appelé "unité arithmétique et logique" (UAL, ou ALU de l'anglais "arithmetic and logic unit") et deux

indicateurs (flags). L'UAL peut effectuer 6 opérations arithmétiques ou logiques différentes. Le résultat des opérations est généralement placé dans l'accumulateur. Les indicateurs sont tenus à jour en fonction de la dernière opération effectuée. L'indicateur de retenue (carry flag) est mis à 1 si une opération produit une retenue. L'indicateur de nullité (zéro flag) est mis à 1 si le résultat d'une opération est nul. Certaines instructions permettent de tester ou de modifier l'état des indicateurs.

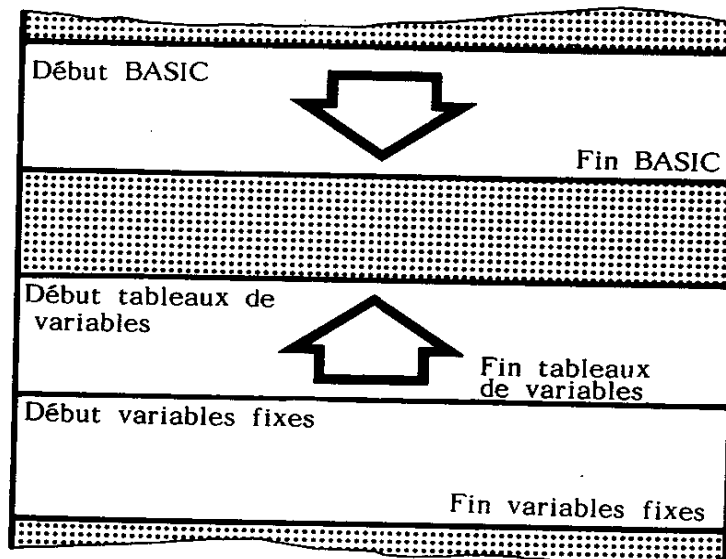
## LA PROGRAMMATION

Nous allons, à présent décrire la façon d'intégrer des sous-programmes assembleur dans des programmes écrits eux-mêmes en Basic. Si l'interpréteur Basic charge les programmes à partir de l'adresse de début de Basic, les programmes écrits en langage machine peuvent être rangés n'importe où en mémoire. Dans le cas de programmes mixtes, le danger existe de voir les différents modules se chevaucher ou s'écraser mutuellement. Ainsi, un programme Basic ou une variable peuvent détruire une routine en langage machine, et vice-versa.

Un programmeur en assembleur doit donc connaître parfaitement la structure de la mémoire vive de son ordinateur de poche. Avant d'aller plus loin dans l'étude du langage machine, nous allons, pour le lecteur qui ne possède pas les ouvrages de la série "Guide de l'utilisateur" ou de la série "Le grand livre des Sharp", brièvement rappeler l'organisation de la mémoire des ordinateurs de poche Sharp:

Les programmes Basic consomment la mémoire utilisateur "de bas en haut", tandis que tables et tableaux utilisent cette mémoire de "haut en bas". L'espace libre entre les deux zones peut être utilisé pour ranger des routines écrites en langage

machine. Cependant il faut alors gérer la croissance du programme Basic et des tables afin d'empêcher ces derniers d'écraser la routine. Trois adresses système pointent respectivement le début de la zone programme Basic, la fin de la zone programme Basic et l'adresse de début des tables et tableaux.



Le programme Basic ci-dessous détermine et édite à l'imprimante les adresses de début et de fin de Basic. Lancez le programme par "RUN".

Listing:(SORTIE: LPRINT = Imprimante / PRINT = Ecran)

```
5:REM @
6:CLEAR :DIM Z$(0)*1: Z$(0)="@"
8:LPRINT "***ANALYSE RAM***"
9:LPRINT "TEMPS EXECUTION
  ELEVE"
10:A=&2000:B=&8000
20:FOR I=A TO B
25:IF PEEK I=255 GOTO 30
26:NEXT I:LPRINT" MEMOIRE NON
  PRESENTE": END
```

```

30:IF PEEK (I+1)=0 AND PEEK (I+2)=5
  AND PEEK (I+4)=215 LET S=I:I=B:
  NEXT I:GOTO 40
35:GOTO 26
40:LPRINT "DEBUT DE LA MEMOIRE=";S
45:Z$="AB" :A$="CD"
50:FOR I=S TO B
60:IF PEEK I=13 AND PEEK (I+1)=255 LET
  X=I+1:LPRINT "FIN DE PROGRAMME=";X:I=B:
  NEXT I:GOTO 75
70:NEXT I:LPRINT "FIN DE PROGRAMME NON
  TROUVEE"
75:FOR I=X TO B:IF PEEK I=90 THEN 80
76:NEXT I:LPRINT "TABLE NON TROUVEE.":END
80:IF PEEK (I+1)=160 AND PEEK (I+7)=64
  LET Y=I:LPRINT "DEBUT DES TABLES
  =" ;Y:I=B: NEXT I:GOTO 85
82:GOTO 76
85:FOR I=Y TO B:IF PEEK I=245 GOTO 87
86:NEXT I:LPRINT "DEBUT DES VAR. NON
  TROUVE":END
87:IF PEEK (I+1)=65 AND PEEK (I+2)=66
  LET V=I:LPRINT "DEBUT DES VAR.
  FIXES=";V:I=B:NEXT I:
  GOTO 90
88:GOTO 86
90:FOR I=V TO B:IF PEEK I=245 GOTO 95
92:NEXT I:LPRINT "FIN VARIABLES NON
  TROUVEE":END
95:IF PEEK (I+1)=67 AND PEEK (I+2)=68
  LET W=I+7:LPRINT "FIN DES VARIABLES
  FIXES=";W:I=B: NEXT I:
  GOTO 100
96:GOTO 92
100:REM SYSTEM
102: WAIT
110:FOR I=W+1 TO B
120:Q=PEEK I+PEEK (I+1)*256:Z=PEEK I
130:IF Q=S LPRINT "DEBUT BASIC - FAIBLE=";I:
  LPRINT "DEBUT BASIC - FORT=";I+1
135:IF Q=X LPRINT "FIN BASIC - FAIBLE=";I:
  LPRINT "FIN BASIC - POIDS FORT=" ;I+1
140:IF Q=Y LPRINT "DEBUT DES TABLES - FAIBLE";I:

```

```

LPRINT "DEBUT DES TABLES - FORT";I+1
190:NEXT I

```

Remarque: Les modèles Sharp qui disposent de l'instruction MERGE possèdent une seconde adresse de début de Basic, sans importance dans le contexte qui nous préoccupe.

### LES RAM DES SHARP PC

MODELE		DEBUT BASIC	(HEX)	FIN BASIC	(HEX)
1245		49200	C030	50639	C5CF
1251		47152	B830	50639	C5CF
1260		22656	5880	25855	65CF
1261/1262		16512	4080	25855	65CF
1280/1360/1475	(8 Ko)	57392	E030	63951	F9CF
1360/1475	(16 Ko)	49200	C030	63951	F9CF
1280/1360/1475	(32 Ko)	32816	8030	63951	F9CF
1350/2500	(5 Ko)	24624	6030	27695	6C2F
1350/2500	(13 Ko)	16432	4030	27695	6C2F
1350/2500	(21 Ko)	8240	2030	27695	6C2F
1401		14336	3800	17871	45CF
1402		8192	2000	17871	45CF
1403/1425/1460	(8 Ko)	57392	E030	64271	F80F
1425/1460	(16 Ko)	49200	C030	64271	F80F
1403+/25/60	(32 Ko)	32816	8030	64271	F80F
1421		14336	3800	17791	457F
1421+	(10 Ko)	8192	2000	17791	457F
1450	(4 Ko)	20528	5030	23599	5C2F
1450	(8 Ko)	16432	4030	23599	5C2F
1450	(16 Ko)	8240	2030	23599	5C2F

} cf remarque  
page 36

## ADRESSES DES INDICATEURS

MODELE	DEB. BASIC/&	FIN BASIC/&	DEB. VARIABLES
1245/51	L 50913/C6E1 H 50914/C6E2	50915/C6E3 50916/C6E4	50940/C6FC 50941/C6FD
1260/61/62	L 26337/66E1 H 26338/66E1	26339/66E3 26340/66E4	28423/6F11 28424/6F12
1280/1360/1475	L 65495/FFD7 H 65496/FFD8	65497/FFD9 65498/FFDA	65501/FFDD 65502/FFDE
1350 <i>si une seule carte (ou ser men "C")</i>	L 28417/6F01 H 24418/6F02	28419/6F03 28420/6F04	28364/66FC 28365/66FD
1401/02/21/22	L 18145/46E1 H 18146/46E2	18147/46E3 18148/46E4	18172/46FC 18173/46FD
1403/25/60	L 65281/FF01 H 65282/FF02	65283/FF03 65284/FF04	65287/FF07 65288/FF08
1450	L 24321/5F01 H 24322/5F02	24323/5F03 24324/5F04	22268/56FC 22269/56FD
2500	L 28049/6D91 H 28050/6D92	28051/6D93 28052/6D94	28055/6D97 28056/6D98

Une fois la zone mémoire libre définie. L'instruction POKE permet de ranger une routine écrite en langage machine dans cette même zone. (cf. les applications pratiques en langage machine). D'autres techniques existent pour stocker en mémoire vive une routine en langage machine sans qu'il n'y ait conflit avec BASIC. On peut, par exemple, après définition des adresses RAM, ranger une routine dans l'espace mémoire occupé par une instruction Basic "REM". Le stockage par POKE s'effectue manuellement ou par programme. Cette dernière solution étant plus

efficace. L'instruction Basic "REM" doit être suivie d'autant de points qu'il y a d'instructions en langage machine à "poker". Cette seconde méthode permet de sauvegarder sur cassette la routine écrite en langage machine, comme s'il s'agissait d'un véritable programme Basic.

Une troisième méthode consiste à tirer parti de l'organisation des tables qui permettent de définir une routine langage machine sous la forme d'une table. La table doit avoir une dimension égale au nombre d'instructions de la routine. Ces instructions sont alors "pokées" dans la table ainsi dimensionnée à la place des valeurs normalement attribuées à la table. Cette méthode a l'avantage de protéger la routine écrite en langage machine contre toute forme d'écrasement. L'instruction PRINT#, et non pas CSAVEM, permet de sauvegarder la routine sur cassette.

Une autre méthode utilise les variables système. En effet certaines régions de la zone mémoire réservée aux variables système ne sont pratiquement jamais utilisées par un programme Basic. Cette méthode est réservée aux programmeurs qui maîtrisent toutes les arcanes des variables système.

Diverses méthodes existent pour stocker en mémoire une routine écrite en langage machine, composée de valeurs numériques comprises entre 0 et 255.

Les petites routines sont généralement directement placées en mémoire au moyen de l'instruction POKE. Il est préférable d'exprimer les valeurs saisies en numération hexadécimale afin de conserver le contrôle du code machine généré. Les valeurs ainsi exprimées sont précédées du caractère "&" (exemple: POKE &6800, &FF, &03, &00,...)

Le programme ci-dessous génère automatiquement des lignes de DATA. Les valeurs sont saisies une à une et la saisie prend fin par la frappe de touche "F". "L:" demande le numéro de ligne et "P" le pas.

L'adresse de fin de basic est saisie à la ligne 2.

#### Générateur de lignes de DATA

```
1: CLEAR : DIM Z$(0)*16
2: Q=24323: REM PC-1450(FIN BASIC)
3: GOTO 15
5: H= INT (Z/256): L=Z-H*256: RETURN
15: P=PEEK Q+PEEK (Q+1)*256
20: INPUT "L:";Z,"P:";S
25: GOSUB 5: POKE P,H,L,79,220: P=P+4: W=1
30: INPUT "VALEUR:";Z$(0): N=LEN Z$(0)
35: IF Z$(0)="F" GOTO 200
40: IF W+N>79 GOTO 100
45: W=W+N+1
50: FOR I=1 TO N: A=ASC (MID$(Z$(0),I,1))
   ): POKE (P+I-1),A
60: NEXT I: POKE (P+I-1),44
70: P=P+N+1
80: GOTO 30
100: POKE P-W-1,W: Z=Z+S: GOSUB 5: POKE P-1,
   13,H,L,79,220: P=P+4: W=W+2: GOTO 50
200: POKE P-W-1,W: POKE P-1,13,255: Z=P:
   GOSUB 5: POKE Q,L,H: END
```

### INSTRUCTIONS MACHINE DE LA CPU SC 61860

L'unité centrale dispose de plus de 100 instructions en langage machine qui peuvent être classées comme suit:

#### 1.) Instructions sur un octet:

Elles consistent, à une exception près, en un code opération à 8 bits qui représente l'instruction destinée à l'unité centrale.

L'instruction LP constitue l'exception. Il s'agit d'une instruction combinée, dont 2 bits correspondent au code opération et 6 bits à l'opérande. LP permet de placer une constante dans le registre P avec un seul octet.

#### 2.) Instructions sur 2 octets:

Ces instructions sont composées d'un code opération à 8 bits suivi d'un opérande à 8 bits. Là aussi il y a une exception: L'instruction CAL.

CAL permet d'appeler dans la ROM interne des sous-programmes avec une instruction à 2 au lieu de 3 bits en utilisant 3 bits pour le code opération et 13 pour l'opérande.

#### 3.) Instructions sur 3 octets:

Elles se composent d'un code opération à 8 bits et d'un opérande à 16 bits. Il en résulte que les deux derniers octets forment un nombre à 16 bits dont le dernier octet est l'octet de poids faible et l'avant dernier l'octet de poids fort.

#### 4.) Instructions sur plus de 3 octets:

Ces instructions sont composées d'un code opération à 8 bits et de plusieurs opérandes à 8 ou 16 bits. Elles sont utilisées, par exemple, pour les sauts de table.

Dans les explications du jeu d'instructions, les fonctions de toutes les instructions sont décrites avec leurs codes hexadécimaux. De plus, on trouve le nombre de cycles ainsi que les modifications apportées aux indicateurs.

#### REGISTRES:

A Accumulateur (8 bits)  
 B Accumulateur (8 bits)  
 DP Pointeur de données, adresses externes (16 bits)  
 I Pointeur index (8 bits)  
 J Pointeur index (8 bits)  
 K Registre à usage général (8 bits)  
 L Registre à usage général (8 bits)  
 M Registre à usage général (8 bits) nouveau  
 N Registre à usage général (8 bits) nouveau  
 P Pointeur d'adresses - RAM interne (7 bits)  
 PC Compteur ordinal (16 bits)  
 Q Pointeur d'adresses - RAM interne (7 bits)  
 R Pointeur de pile (7 bits)  
 X Pointeur index - RAM externe (16 bits)  
 Y Pointeur index - RAM externe (16 bits)

Ces mnémoniques seront utilisés dans les explications concernant les instructions en langage machine et représentent le contenu de ces registres. Si un mnémonique est mentionné entre parenthèses (ex: (A), (B)), il représente le contenu d'une adresse mémorisée dans ce registre.

#### Autres abréviations:

C Indicateur de retenue  
 Z Indicateur de zéro  
 L Valeur 6 bits (00-3F) pour l'instruction LP  
 Valeur 5 bits (00-IF) pour l'instruction CALL  
 N Valeur 8 bits  
 Valeur 7 bits (registres P et Q)  
 NM Valeur 16 bits (N= poids fort, M= poids faible)  
 D Registre de boucle (est décrémenté à la place de I et J)  
 XX Octets de poids fort des adresses de la RAM dans lesquelles on écrit par POKE les programmes LM donnés comme exemples.

Nous avons également indiqué le cycle de chaque instruction. Voici quelques explications sur ce point:

L'unité centrale charge le code opération d'une instruction LM et déchiffre ensuite les bits un par un. Dans le cas d'instructions à plusieurs octets, les arguments sont également déchiffrés.

On comprend ainsi que la durée d'exécution varie d'une instruction à l'autre.

Le temps d'exécution d'une instruction peut être calculé précisément d'après la fréquence utilisée par le quartz de votre ordinateur. La connaissance de cette fréquence et des cycles des instructions peut-être importante pour la mise au point des programmes LM gérant le Beeper ou l'enregistreur à cassettes. Par exemple, un programme peut positionner à 1 un bit une sortie spécifique est ensuite, le remettre à 0.

La donnée "temps" est de plus en plus importante dans le transfert des données vers des unités périphériques, car le transfert bit par bit nécessite des réponses à un rythme déterminé.

## LE JEU D'INSTRUCTIONS

Les instructions sont classées par ordre alphabétique afin de vous permettre d'utiliser cette partie du livre comme manuel de référence. Vous trouverez en annexe la liste des mnémoniques classée par ordre croissant de code hexadécimal.

## ADB

Cette instruction réalise une addition sur deux octets. Le contenu de l'adresse pointée par le registre P (16 bits) est additionné au contenu des registres A et B (le registre B complémente le registre A pour former un registre sur 16 bits). Le résultat est placé dans l'adresse pointée par le registre P. Comme le registre P est un registre sur 8 bits, l'adresse (P+1) est utilisée pour former un registre sur 16 bits.

$(P+1, P) + (BA) \rightarrow (P+1, P) \quad C, Z \quad P+1 \rightarrow P$

Code HEX: 14                      Code DEC: 20  
Code BIN: 0 0 0 1 0 1 0 0      Indic. : CZ  
Cycles : 5                      Octets : 01

Remarques: L'indicateur de nullité Z est mis à 1 si le résultat est égal à 0. L'indicateur de retenue C est mis à 1 si l'opération produit une retenue.

Il faut noter que le registre P est incrémenté de une unité. C'est-à-dire que si le registre P contient la valeur 5, celle-ci passera à 6 après exécution de l'instruction.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LP 08	88
	xx01	LIA 30	02 30
	xx03	EXAM	DB
	xx04	LP 09	89
	xx05	LIA 21	02 21
	xx07	EXAM	DB
	xx08	LIB 10	03 10
	xx0A	LIA 40	02 40
	xx0C	LP 08	88
	xx0D	ADB	14
	xx0E	RTN	37



**Description:** Ce programme additionne les valeurs &2130 et &1040. Pour additionner ces deux nombres, il faut placer les valeurs correspondantes dans les registres de la CPU.

Le registre P est chargé avec la valeur 8 pour pouvoir ranger le nombre &2130 dans les registres K et L. Ensuite, l'accumulateur reçoit l'octet de poids faible &30. L'instruction EXAM effectue le transfert du contenu de l'accumulateur (&30) dans le registre K. La même opération est répétée pour le registre L d'adresse 09 qui cette fois contiendra l'octet de poids fort &21.

La valeur &1040 est ensuite placée dans les 2 accumulateurs. Le résultat de l'opération se trouve dans les registres K et L.

Après exécution de l'instruction ADB, le registre P contient la valeur 9.

Pour pouvoir afficher le résultat &3170 il reste à procéder au transfert du contenu des registres K et L vers la RAM externe et à la lecture au moyen de l'instruction Basic PEEK.

## ADCM

Cette instruction ajoute le contenu de l'adresse référencée par le registre P au contenu de l'accumulateur et de l'indicateur de retenue. Le résultat est placé dans l'octet mémoire pointé par le registre P.

$(P)+A+C \rightarrow (P) \quad C, Z$

Code HEX: C4

Code DEC: 196

Code BIN: 1 1 0 0 0 1 0 0

Indic. : CZ

Cycles : 3

Octets : 01

**Remarques:** Voir l'exemple d'une addition dans la partie de ce manuel consacrée aux applications pratiques.

Exemple:	Adresse	Assembleur	Code Hex
	xx00	LP 08	88
	xx01	LIA 30	02 30
	xx03	EXAM	DB
	xx04	LIA 40	02 40
	xx06	SC	D0
	xx07	ADCM	C4
	xx08	RTN	37

**Description:** Ce programme additionne les nombres &30 et &40 et la retenue. La valeur &30 est chargée dans le registre K (08), la valeur &40 est placée dans l'accumulateur A. L'instruction SC positionne l'indicateur de retenue à 1. Le résultat &71 est placé dans le registre K.

## ADIA n

Cette instruction ajoute la valeur n sur 8 bits au contenu de l'accumulateur.

A+n -> A C,Z

Code HEX: 74                      Code DEC: 116  
Code BIN: 0 1 1 1 0 1 0 0      Indic. : CZ  
Cycles : 4                      Octets : 02

Remarques: L'indicateur de nullité Z est mis à 1 si le résultat est égal à 0. L'indicateur de retenue C est mis à 1 si l'opération produit une retenue.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LIA 2F	02 2F
	xx02	ADIA 12	74 12
	xx04	RTN	37

Description: Ce programme additionne le contenu de l'accumulateur &2F et la valeur &12. Le résultat &41 est placé dans l'accumulateur.

## ADIM n

Cette instruction ajoute la valeur n au contenu de l'adresse référencée par le registre P.

(P)+n -> (P) C,Z

Code HEX: 70                      Code DEC: 112  
Code BIN: 0 1 1 1 0 0 0 0      Indic. : CZ  
Cycles : 4                      Octets : 02

Remarques: L'indicateur de nullité Z est mis à 1 si le résultat est égal à 0. L'indicateur de retenue C est mis à 1 si l'opération produit une retenue.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LIA 70	02 70
	xx02	LP 08	88
	xx03	EXAM	DB
	xx04	ADIM 0F	70 0F
	xx06	RTN	37

Description: Ce programme additionne les nombres &70 et &0F. Le registre utilisé est le registre K. La valeur &70 est chargée dans l'accumulateur et ensuite transférée dans le registre K grâce à l'instruction EXAM. Après exécution de l'opération d'addition, le résultat &7F est placé dans le registre K.

## ADM

Cette instruction ajoute le contenu de l'accumulateur au contenu de l'adresse de la RAM interne référencée par le registre P.

(P)+A -> (P) C,Z

Code HEX: 44                      Code DEC: 68  
Code BIN: 0 1 0 0 1 0 0 0      Indic. : CZ  
Cycles : 3                      Octets : 01

**Remarques:** L'indicateur de nullité Z est mis à 1 si le résultat est égal à 0. L'indicateur de retenue C est mis à 1 si l'opération produit une retenue.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LP 08	88
	xx01	LIA 30	02 30
	xx03	EXAM	DB
	xx04	LIA 40	02 40
	xx06	ADM	44
	xx07	RTN	37

**Description:** Ce programme additionne les nombres &30 et &40. Le registre utilisé est le registre K. La valeur &30 est chargée dans l'accumulateur et ensuite transférée dans le registre K grâce à l'instruction EXAM. Après exécution de l'opération d'addition, le résultat &70 est placé dans le registre K.

## ADN

Cette instruction ajoute d+1 fois le contenu de l'accumulateur aux adresses mémoire référencée par le registre P. Au départ, le registre P pointe l'adresse la plus haute.

I -> d  
Répéter:  
(P)+A -> (P) (DCB),P-1,d-1 C,Z  
Jusqu'à ce que:  
d=FF

Code HEX: 0C                      Code DEC: ~~104~~ 12  
Code BIN: 0 0 0 0 1 1 0 0      Indic. : CZ  
Cycles : 7+3d                      Octets : 01

**Remarques:** L'indicateur de nullité Z est mis à 1 si le résultat est égal à 0. L'indicateur de retenue C est mis à 1 si l'opération produit une retenue.

Le registre I contient la valeur d.  
Les registres D et P sont décrémentés.  
Le nombre de cycles dépend bien évidemment du nombre d'itérations de la boucle.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LP 08	88
	xx01	LIA 10	02 10
	xx03	EXAM	DB
	xx04	LP 09	89
	xx05	LIA 99	02 99
	xx07	EXAM	DB
	xx08	LP 0A	8A
	xx09	LIA 99	02 99
	xx0B	EXAM	DB
	xx0C	LP 0B	8B
	xx0D	LIA 99	02 99
	xx0F	EXAM	DB
	xx10	LII 03	00 03
	xx12	LIA 14	02 14

xx14	ADN	0C
xx15	RTN	37

**Description:** Ce programme additionne les nombres 10999999 et 14 sous forme DCB. Le nombre 10999999 est placé aux adresses mémoire 08-0B (xx00-xx0F). Le nombre 14 est stocké dans l'accumulateur. Le registre I est chargé avec la valeur 3 (d-1=3) car les adresses 08-0B correspondent à des registres sur 4 bits. Au sein de l'instruction ADN, le contenu du registre I est transféré dans le registre d. le résultat 11000013 est placé dans les registres 08-0B. Chaque registre contient la valeur suivante:

K (08) = 11  
 L (09) = 00  
 M (0A) = 00  
 N (0B) = 13  
 P = 08

## ADW

Cette instruction effectue également une addition sous forme DCB. Elle ajoute le contenu de d+1 octets, à partir de l'adresse référencée par le registre Q, aux d+1 octets, à partir de l'adresse pointée par le registre P. Au départ, les registres P et Q contiennent les adresses les plus hautes. Le résultat est placé aux adresses référencées par le registre P.

I -> d  
 Répéter:  
 (P)+(Q) -> (P) (DCB),P-1,Q-1,d-1 C,Z  
 Jusqu'à ce que:  
 d=FF

Code HEX: 0E                      Code DEC: 14  
 Code BIN: 0 0 0 0 1 1 1 0      Indic. : CZ  
 Cycles : 7+3d                    Octets : 01

**Remarques:** L'indicateur de nullité Z est mis à 1 si le résultat est égal à 0. L'indicateur de retenue C est mis à 1 si l'opération produit une retenue.

Le registre I contient la valeur d.

Les registres d, P et Q sont décrémentés.

Le nombre de cycles dépend bien évidemment du nombre d'itérations de la boucle.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LP 08	88
	xx01	LIA 10	02 10
	xx03	EXAM	DB
	xx04	LP 09	89
	xx05	LIA 99	02 99
	xx07	EXAM	DB
	xx08	LP 0A	8A
	xx09	LIA 99	02 99
	xx0B	EXAM	DB
	xx0C	LP 0B	8B

xx0D	LIA	99	02 99
xx0F	EXAM		DB
xx10	LII	01	00 01
xx12	LIQ	09	13 09
xx14	ADW		0E
xx15	RTN		37

**Description:** Ce programme additionne les nombres 1099 et 9999 sous forme DCB. Le nombre 1099 est stocké dans les registres K et L tandis que la valeur 9999 est rangée dans les registres M et N. Pour que l'opération d'addition puisse s'effectuer, il faut que le registre P référence l'adresse 08 et que le registre Q pointe l'adresse 09. L'opération d'addition positionne l'indicateur de retenue à 1 car le résultat (1099+9999=11098) ne peut être représenté sur deux octets. Chaque registre contient la valeur suivante:

K (08) = 10  
L (09) = 99  
M (0A) = 10  
N (0B) = 98

P = 09  
Q = 06  
C = 1

## ANIA n

Cette instruction effectue un ET logique entre la valeur n sur 8 bits et l'accumulateur.

A^n -> A Z

Code HEX: 64                      Code DEC: 100  
Code BIN: 0 1 1 0 0 1 0 0      Indic. : Z  
Cycles : 4                      Octets : 02

**Remarques:** L'indicateur de nullité est positionné à 1 si le résultat est égal à zéro.

**Exemple:**

Adresse	Assembleur	Code HEX
xx00	LIA 2F	02 2F
xx02	ANIA 12	64 12
xx04	RTN	37

**Description:** Ce programme effectue un ET logique entre la valeur &2F contenue dans l'accumulateur et la valeur &12. Le résultat &02 est placé dans l'accumulateur.

## ANID n

Cette instruction effectue un ET logique entre l'octet référencé par le registre DP et la constante n (8 bits). Le résultat est placé à l'adresse référencée par le registre DP.

$(DP)^n \rightarrow (DP) \quad Z$

Code HEX: D4                      Code DEC: 212  
Code BIN: 1 1 0 1 0 1 0 0      Indic. : Z  
Cycles : 6                      Octets : 02

Remarque: L'indicateur de nullité est positionné à 1 si le résultat est égal à zéro.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LIDP xx50	10 xx 50
	xx03	LIA 15	02 15
	xx05	STD	52
	xx06	ANID 63	D4 63
	xx08	RTN	37

Description: Les caractères "xx" correspondent d'une part à l'octet de poids fort de l'adresse mémoire en RAM externe qui doit contenir le programme en langage machine et d'autre part à l'octet de poids fort de l'adresse où sera effectué le ET logique. Les valeurs associées aux caractères "xx" dépendent du modèle d'ordinateur de poche utilisé. L'exemple présenté ci-dessus effectue un ET-logique entre les valeurs &15 et &63. La valeur &15 est placée à l'adresse mémoire xx50. Ensuite, le programme effectue un ET-logique avec la valeur &63. Le résultat &01 est placé à l'adresse xx50 et peut être lu à l'aide de l'instruction Basic "PEEK".

## ANIM n

Cette instruction effectue un ET-logique entre l'octet référencé par le registre P et la constante n (8 bits). Le résultat est placé à l'adresse référencée par le registre P.

$(P)^n \rightarrow (P) \quad Z$

Code HEX: 60                      Code DEC: 96  
Code BIN: 0 1 1 0 0 0 0 0      Indic. : Z  
Cycles : 4                      Octets : 02

Remarque: L'indicateur de nullité est positionné à 1 si le résultat est égal à zéro.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LP 0A	8A
	xx03	LIA 15	02 15
	xx05	EXAM	DB
	xx06	ANIM 63	60 63
	xx08	RTN	37

Description: Le programme présenté ci-dessus effectue un ET-logique entre les valeurs &15 et &63. La valeur &15 est placée à l'adresse mémoire 0A (registre M) de la RAM interne. Ensuite, le programme effectue un ET-logique avec la valeur &63. Le résultat &01 est placé dans le registre M (adresse 0A).

## ANMA

Cette instruction effectue un ET-logique entre l'octet référencé par le registre P et l'accumulateur. Le résultat est placé à l'adresse référencée par le registre P.

$(P)^A \rightarrow (P) Z$

Code HEX: 46                      Code DEC: 70  
Code BIN: 0 1 0 0 0 1 1 0      Indic. : Z  
Cycles : 3                      Octets : 01

Remarque: L'indicateur de nullité est positionné à 1 si le résultat est égal à zéro.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LP 0A	8A
	xx03	LIA 15	02 15
	xx05	EXAM	DB
	xx06	LIA 63	02 63
	xx08	ANMA	46
	xx09	RTN	37

**Description:** Le programme présenté ci-dessus effectue un ET-logique entre les valeurs &15 et &63. La valeur &15 est placée à l'adresse mémoire 0A (registre M) de la RAM interne. Ensuite, le programme effectue un ET-logique avec la valeur &63 de l'accumulateur. Le résultat est placé dans le registre M (adresse 0A).

## CAL In

Cette instruction effectue un appel incondi- tionnel à un sous-programme de la ROM interne d'adresse absolue In. 1 correspond a une valeur sur 5 bits qui complémente le code opération pour former une adresse sur 8 bits. Ceci permet d'utiliser des adresses comprises entre 0000 et 1FFF.

PC+2  $\rightarrow$  (R-1,R-2)  
R-2  $\rightarrow$  R  
0001  $\rightarrow$  PCH  
n  $\rightarrow$  PCL

Code HEX: E0+1                      Code DEC: 224+1  
Code BIN: 1 1 1 < - 1 - >      Indic. :  
Cycles : 7                      Octets : 02

**Remarques:** Cette instruction ne consomme que deux octets au lieu des trois nécessaires à l'instruction CALL. Cependant, l'adresse passée en paramètre doit impérativement référencer une case mémoire de la RAM interne. Tout comme l'instruction LP, cette instruction possède divers codes opérations (E0-FF). PCH représente l'octet de poids fort du comp- teur ordinal. PCL l'octet de poids faible.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	CAL 05A2	E5 A2
	xx02	RTN	37

**Description:** Ce programme appelle une routine de la ROM interne placée à l'adresse &05A2. Il est impératif que l'adresse passée en paramètre corresponde au début d'un sous-programme. Dans le cas contraire, l'appel de la routine par CAL provoquerait inmanquablement un "plantage" de l'ordinateur de poche. L'exemple proposé ci-dessus met le PC-1401/02 en mode graphique.

## CALL nm

Cette instruction effectue un appel incondi-  
tionnel à un sous-programme de la ROM interne,  
de la ROM externe ou de la RAM et d'adresse nm.

PC+3	-> (R-1,R-2)
R-2	-> R
n	-> PCH
m	-> PCL

Code HEX: 78                      Code DEC: 120  
Code BIN: 0 1 1 1 1 0 0 0      Indic. :  
Cycles : 8                      Octets : 03

Remarques: PCH représente l'octet de poids fort  
du compteur ordinal. PCL l'octet de poids  
faible.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	CALL 05A2	78 05 A2
	xx03	RTN	37

Description: Ce programme appelle une routine  
de placée à l'adresse &05A2. Il est impératif  
que l'adresse passée en paramètre corresponde  
au début d'un sous-programme. Dans le cas con-  
traire, l'appel de la routine par CALL provoquerait  
inmanquablement un "plantage" de l'ordina-  
teur de poche. L'exemple proposé ci-dessus met  
le PC-1401/02 en mode graphique.

Cette instruction consomme un octet de plus que  
l'instruction CAL. Cependant, elle n'est pas  
limitée aux adresses inférieures ou égales à  
&1FFF.

## CPIA n

Cette instruction compare à l'accumulateur (re-  
gistre A) la constante n (8 bits) donnée en pa-  
ramètre.

A - n C,Z
-----------

Code HEX: 67                      Code DEC: 103  
Code BIN: 0 1 1 0 0 1 1 1      Indic. : CZ  
Cycles : 4                      Octets : 02

Remarques: Le résultat affecte les indicateurs  
de la façon suivante:

A < n	C = 1	Z = 0
A = n	C = 0	Z = 1
A > n	C = 0	Z = 0

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LIA 02	02 02
	xx02	INCA	42
	xx03	CPIA 30	67 30
	xx05	JRNZM 04	29 04
	xx07	RTN	37

Description: Ce programme incrémente de une  
unité le contenu de l'accumulateur jusqu'à ce  
que sa valeur soit égale à &30. La valeur &02  
chargée dans l'accumulateur. L'instruction INCA  
incrémente le contenu de l'accumulateur d'une  
unité. Après comparaison, le programme effectue  
un saut à l'adresse xx02 tant que la valeur  
contenue dans l'accumulateur est inférieure à  
&30.



## CPIM n

Cette instruction compare l'octet de la RAM interne référencé par le registre P à la constante n (8 bits) donnée en paramètre.

(P) - n C,Z

Code HEX: 63                      Code DEC: 99  
Code BIN: 0 1 1 0 0 0 1 1      Indic. : CZ  
Cycles : 4                      Octets : 02

Remarques: Le résultat affecte les indicateurs de la façon suivante:

(P) < n	C = 1	Z = 0
(P) = n	C = 0	Z = 1
(P) > n	C = 0	Z = 0

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LP 09	89
	xx01	LIA 02	02 02
	xx03	EXAM	DB
	xx04	INCL	C8
	xx05	CPIM 30	63 30
	xx07	JRNZM 04	29 04
	xx09	RTN	37

Description: Ce programme incrémente de une unité le registre L jusqu'à ce que son contenu soit égal à &30. Au départ, la valeur &02 est chargée dans l'accumulateur puis transférée dans le registre L. L'instruction INCL incrémente le contenu du registre L de une unité. Après comparaison, le programme effectue un saut à l'adresse xx04 tant que la valeur contenue dans le registre L est inférieure à &30.

## CPMA

Cette instruction compare à l'accumulateur l'octet de la RAM interne référencé par le registre P.

(P) - A C,Z

Code HEX: C7                      Code DEC: 199  
Code BIN: 1 1 0 0 0 1 1 1      Indic. : CZ  
Cycles : 3                      Octets : 01

Remarques: Le résultat affecte les indicateurs de la façon suivante:

(P) < A	C = 1	Z = 0
(P) = A	C = 0	Z = 1
(P) > A	C = 0	Z = 0

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LP 09	89
	xx01	LIA 02	02 02
	xx03	EXAM	DB
	xx04	LIA 30	02 30
	xx06	INCL	C8
	xx07	CPMA	C7
	xx08	JRNZM 03	29 03
	xx0A	RTN	37

Description: Ce programme incrémente de une unité le registre L (adresse 09) jusqu'à ce que son contenu soit égal à &30. Au départ, la valeur &02 est chargée dans l'accumulateur puis transférée dans le registre L. L'opérande de comparaison est ensuite chargé dans l'accumulateur. L'instruction INCL incrémente le contenu du registre L de une unité. Après comparaison, le programme effectue un saut à l'adresse xx06 tant que la valeur contenue dans le registre L est inférieure à &30.

## DATA

Cette instruction permet d'écrire un bloc de données comprenant d+1 octets pointés par le registre sur 16 bits AB dans une zone mémoire située d+1 octets après l'adresse référencée par le registre P.

I -> d  
Répéter:  
(BA) -> (P) ,P+1,BA+1,d-1  
jusqu'à ce que:  
d = FF

Code HEX: 35                      Code DEC: 53  
Code BIN: 0 0 1 1 0 1 0 1      Indic. : CZ  
Cycles : 11+4d                      Octets : 01

Remarques: L'existence de cette instruction n'avait auparavant jamais été révélée par la firme SHARP.

Contrairement à l'instruction "PEEK", celle-ci permet de lire le contenu de la ROM interne. Le contenu du registre I est placé dans le registre boucle d.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LII 03	00 03
	xx02	LP 08	88
	xx03	LIB 00	03 00
	xx05	LIA 0A	02 0A
	xx07	DATA	35
	xx08	RTN	37

Description: Ce programme écrit 4 octets de la ROM interne, à partir de l'adresse &000A, dans la RAM interne. Ces 4 octets sont copiés dans les adresses comprises entre &08 et &0B (registre K à N). Au départ, le registre I est chargé avec le nombre d'octets -1 et le registre P avec l'adresse de la RAM interne où le

premier octet sera écrit. Le registre A contient l'octet de poids faible, le registre B l'octet de poids fort. L'instruction DATA écrit le contenu des adresses &000A à &000D dans la zone de la RAM interne comprise entre les adresses &08 et &0B. Après transfert dans la RAM externe, ces valeurs peuvent être lues au moyen de l'instruction PEEK. (cf. les instructions EXWD et EXBD).

## DECA

Cette instruction décrémente le registre A de une unité.

A - 1 -> A C,Z

Code HEX: 43                      Code DEC: 67  
Code BIN: 0 1 0 0 0 0 1 1      Indic. : CZ  
Cycles : 4                      Octets : 01

Remarques: Cette instruction modifie le contenu du registre Q.  
Le résultat affecte les indicateurs.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LIA 03	02 03
	xx02	DECA	43
	xx03	RTN	37

Description: Après exécution du programme, le contenu de l'accumulateur égal 2.  
Le contenu du registre Q est aussi égal à 2.

## DECB

Cette instruction décrémente le registre B d'une unité.

B - 1 -> B C,Z

Code HEX: C3                      Code DEC: 195  
Code BIN: 1 1 0 0 0 0 1 1      Indic. : CZ  
Cycles : 4                      Octets : 01

Remarques: Cette instruction modifie le contenu du registre Q.  
Le résultat affecte les indicateurs.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LIB 03	03 03
	xx02	DECB	C3
	xx03	RTN	37

Description: Après exécution du programme, le contenu du registre B égal 2.  
Le contenu du registre Q est égal à 3.

## DECI

Cette instruction décrémente le registre I d'une unité.

I - 1 -> I C,Z

Code HEX: 41                      Code DEC: 65  
Code BIN: 0 1 0 0 0 0 0 1      Indic. : CZ  
Cycles : 4                      Octets : 01

**Remarques:** Cette instruction modifie le contenu du registre Q.  
Le résultat affecte les indicateurs.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LII 03	00 03
	xx02	DECI	41
	xx03	RTN	37

**Description:** Après exécution du programme, le contenu du registre I égal 2.  
Le contenu du registre Q est égal à 0.

## DECJ

Cette instruction décrémente le registre J d'une unité.

J - 1 -> J C,Z

Code HEX: C1                      Code DEC: 193  
Code BIN: 1 1 0 0 0 0 0 1      Indic. : CZ  
Cycles : 4                      Octets : 01

**Remarques:** Cette instruction modifie le contenu du registre Q.  
Le résultat affecte les indicateurs.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LIJ 03	01 03
	xx02	DECJ	C1
	xx03	RTN	37

**Description:** Après exécution du programme, le contenu du registre J égal 2.  
Le contenu du registre Q est égal à 1.

## DECK

Cette instruction décrémente le registre K d'une unité.

K - 1 -> K C,Z

Code HEX: 49                      Code DEC: 73  
Code BIN: 0 1 0 0 1 0 0 1      Indic. : CZ  
Cycles : 4                      Octets : 01

Remarques: Cette instruction modifie le contenu du registre Q.  
Le résultat affecte les indicateurs.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LP 08	88
	xx01	LIA 03	02 03
	xx03	EXAM	DB
	xx04	DECK	49
	xx05	RTN	37

Description: Pour pouvoir charger le registre K avec la valeur &03, il faut auparavant placer cette valeur dans l'accumulateur. L'instruction EXAM effectue le transfert de valeur entre l'accumulateur et le registre K. Après exécution du programme, le contenu du registre K égal 2.

Le contenu du registre Q est égal à 8.

## DECL

Cette instruction décrémente le registre L d'une unité.

L - 1 -> L C,Z

Code HEX: C9                      Code DEC: 201  
Code BIN: 1 1 0 0 1 0 0 1      Indic. : CZ  
Cycles : 4                      Octets : 01

Remarques: Cette instruction modifie le contenu du registre Q.  
Le résultat affecte les indicateurs.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LP 09	89
	xx01	LIA 03	02 03
	xx03	EXAM	DB
	xx04	DECL	C9
	xx05	RTN	37

Description: Pour pouvoir charger le registre L avec la valeur &03, il faut auparavant placer cette valeur dans l'accumulateur. L'instruction EXAM effectue le transfert de valeur entre l'accumulateur et le registre L. Après exécution du programme, le contenu du registre L égal 2.

Le contenu du registre Q est égal à 9.

## DECM

Cette instruction décrémente le registre M d'une unité.

M - 1 -> M C,Z

Code HEX: 4B                      Code DEC: 75  
Code BIN: 0 1 0 0 1 0 1 1      Indic. : CZ  
Cycles : 4                      Octets : 01

**Remarques:** Cette instruction modifie le contenu du registre Q.  
Le résultat affecte les indicateurs.  
Cette instruction ne fait pas partie du jeu d'instructions standard défini par la firme Sharp.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LP 0A	8A
	xx01	LIA 03	02 03
	xx03	EXAM	DB
	xx04	DECM	4B
	xx05	RTN	37

**Description:** Pour pouvoir charger le registre M avec la valeur &03, il faut auparavant placer cette valeur dans l'accumulateur. L'instruction EXAM effectue le transfert de valeur entre l'accumulateur et le registre M. Après exécution du programme, le contenu du registre M égal 2.

Le contenu du registre Q est égal à &0A.

## DECN

Cette instruction décrémente le registre N d'une unité.

N - 1 -> N C,Z

Code HEX: CB                      Code DEC: 203  
Code BIN: 1 1 0 0 1 0 1 1      Indic. : CZ  
Cycles : 4                      Octets : 01

**Remarques:** Cette instruction modifie le contenu du registre Q.  
Le résultat affecte les indicateurs.  
Cette instruction ne fait pas partie du jeu d'instructions standard défini par la firme Sharp.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LP 0B	8B
	xx01	LIA 03	02 03
	xx03	EXAM	DB
	xx04	DECN	CB
	xx05	RTN	37

**Description:** Pour pouvoir charger le registre N avec la valeur &03, il faut auparavant placer cette valeur dans l'accumulateur. L'instruction EXAM effectue le transfert de valeur entre l'accumulateur et le registre N. Après exécution du programme, le contenu du registre N égal 2.

Le contenu du registre Q est égal à &0B.

## DECP

Cette instruction décrémente le registre P d'une unité.

P - 1 -> P C,Z

Code HEX: 51                      Code DEC: 81  
Code BIN: 0 1 0 1 0 0 0 1      Indic. : CZ  
Cycles : 2                      Octets : 01

Remarques: Cette instruction modifie le contenu du registre Q.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LP 09	89
	xx01	DECP	51
	xx02	RTN	37

Description: Ce programme charge le registre P avec la valeur &09 et le décrémente de une unité. Après exécution, le contenu du registre P égal &08.

## DTJ

Cette instruction exécute la table de sauts (Do Table Jump) définie à l'aide de l'instruction PTJ. Cette instruction est suivie d'informations au format suivant:

1<sup>er</sup> opérande de comparaison

adresse de saut - poids fort

1<sup>ère</sup> entrée de la table

adresse de saut - poids faible

2<sup>ème</sup> opérande de comparaison

adresse de saut - poids fort

2<sup>ème</sup> entrée de la table

adresse de saut - poids faible

⋮

N<sup>ème</sup> opérande de comparaison

adresse de saut - poids fort

dernière entrée

adresse de saut - poids faible

adresse de saut - poids fort

si l'opérande n'est  
pas trouvée

adresse de saut - poids faible

Code HEX: 69

Code BIN: 0 1 1 0 1 0 0 1

Cycles : -

Code DEC: 105

Indic. : CZ

Octets :

Remarques: Cette instruction ne fait pas partie  
du jeu d'instructions standard défini par la  
firme Sharp.

Cette instruction doit être utilisée en con-  
jonction avec l'instruction PTJ (Prepare Table  
Jump).

## DX

Cette instruction décrémente de une unité le regis-  
tre X (16 bits) et écrit la valeur décrémentée dans  
le registre DP.

XH,XL -> DP  
DP-1 -> DP,X

Code HEX: 05

Code BIN: 0 0 0 0 0 1 0 1

Cycles : 6

Code DEC: 05

Indic. :

Octets : 01

Remarques: Cette instruction modifie le contenu  
du registre Q.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LP 04	84
	xx01	LIA 20	02 20
	xx03	EXAM	DB
	xx04	LP 05	85
	xx05	LIA 27	02 27
	xx07	EXAM	DB
	xx08	DX	05
	xx09	RTN	37

Description: Ce programme place la valeur &2720  
dans le registre X et la décrémente. Le résul-  
tat &271F figure dans les registres X et DP. Il  
faut noter que l'octet de poids fort et l'octet  
de poids faible de la valeur &2720 sont rangés  
séparément dans le registre X car il n'existe  
aucune instruction de chargement d'une valeur  
sur 16 bits pour le registre X.

Après exécution du programme, le registre Q  
contient la valeur &05.



## DXL

Cette instruction effectue des opérations multiples:

- a) soustrait 1 du registre X.
- b) range la valeur décrétementée dans les registres X et DP.
- c) place le contenu de l'octet référencé par le registre DP dans l'accumulateur.

X	->	DP
DP-1	->	DP,X
(DP)	->	A

Code HEX: 25                      Code DEC: 37  
Code BIN: 0 0 1 0 0 1 0 1      Indic. :  
Cycles : 7                      Octets : 01

Remarques: Cette instruction modifie le contenu du registre Q.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LP 04	84
	xx01	LIA 20	02 20
	xx03	EXAM	DB
	xx04	LP 05	85
	xx05	LIA 27	02 27
	xx07	EXAM	DB
	xx08	DXL	25
	xx09	RTN	37

Description: Ce programme place la valeur &2720 dans le registre X et la décrémente. Le résultat &271F figure dans les registres X et DP. Il faut noter que l'octet de poids fort et l'octet de poids faible de la valeur &2720 sont rangés séparément dans le registre X car il n'existe aucune instruction de chargement d'une valeur

sur 16 bits pour le registre X.  
Le contenu de l'adresse mémoire &271F est placé dans l'accumulateur.  
Après exécution du programme, le registre Q contient la valeur &05.

## DY

Cette instruction décrémente de une unité le registre Y (16 bits) et écrit la valeur décrémentée dans le registre DP.

YH,YL -> DP  
DP-1 -> DP,Y

Code HEX: 07                      Code DEC: 07  
Code BIN: 0 0 0 0 0 1 1 1      Indic. :  
Cycles : 6                      Octets : 01

**Remarques:** Cette instruction modifie le contenu du registre Q.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LP 06	86
	xx01	LIA 20	02 20
	xx03	EXAM	DB
	xx04	LP 07	87
	xx05	LIA 27	02 27
	xx07	EXAM	DB
	xx08	DY	07
	xx09	RTN	37

**Description:** Ce programme place la valeur &2720 dans le registre Y et la décrémente. Le résultat &271F figure dans les registres Y et DP. Il faut noter que l'octet de poids fort et l'octet de poids faible de la valeur &2720 sont rangés séparément dans le registre Y car il n'existe aucune instruction de chargement d'une valeur sur 16 bits pour le registre Y. Après exécution du programme, le registre Q contient la valeur &07.

## DYS

Cette instruction effectue des opérations multiples:

- soustrait 1 du registre Y (16 bits).
- range la valeur décrémentée dans les registres Y et DP.
- place le contenu de l'accumulateur dans l'octet référencé par le registre DP.

Y -> DP  
DP-1 -> DP,Y  
A -> (DP)

Code HEX: 27                      Code DEC: 39  
Code BIN: 0 0 1 0 0 1 1 1      Indic. :  
Cycles : 6                      Octets : 01

**Remarques:** Cette instruction modifie le contenu du registre Q.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LP 06	86
	xx01	LIA 20	02 20
	xx03	EXAM	DB
	xx04	LP 07	87
	xx05	LIA 27	02 27
	xx07	EXAM	DB
	xx08	LIA FF	02 FF
	xx0A	DYS	27
	xx0B	RTN	37

**Description:** Ce programme place la valeur &2720 dans le registre Y et la décrémente. Le résultat &271F figure dans les registres Y et DP. Il faut noter que l'octet de poids fort et l'octet de poids faible de la valeur &2720 sont rangés séparément dans le registre Y car il n'existe

aucune instruction de chargement d'une valeur sur 16 bits pour le registre Y.  
 Le contenu de l'accumulateur (&FF) est placé à l'adresse mémoire &271F.  
 Après exécution du programme, le registre Q contient la valeur &07.

## EXAB

Cette instruction échange le contenu de l'accumulateur avec celui du registre B.

A <-> B

Code HEX: DA                      Code DEC: 218  
 Code BIN: 1 1 0 1 1 0 1 0      Indic. :  
 Cycles : 5                      Octets : 01

**Remarques:** Cette instruction est souvent utilisée pour vérifier le contenu de l'accumulateur.

**Exemple:**

<u>Adresse</u>	<u>Assembleur</u>	<u>Code HEX</u>
xx00	LIA 06	02 06
xx02	LIB F0	03 F0
xx04	EXAB	DA
xx05	RTN	37

**Description:** Ce programme échange l'accumulateur (&06) avec le registre B (&F0).

## EXAM

Cette instruction échange le contenu de l'accumulateur avec le contenu de l'octet de la RAM interne référencé par le registre P.

A <-> (P)

Code HEX: DB                      Code DEC: 219  
Code BIN: 1 1 0 1 1 0 1 1      Indic. :  
Cycles : 3                      Octets : 01

Remarques: Cette instruction est souvent utilisée pour placer une valeur dans un registre de la RAM interne.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LIA 06	02 06
	xx02	LP 08	88
	xx03	EXAM	DB
	xx04	RTN	37

Description: Ce programme échange le contenu de l'accumulateur (&06) avec celui du registre K.

## EXB

Cette instruction échange le contenu de d+1 octets, à partir de l'adresse référencée par le registre Q, avec le contenu des adresses placées d+1 octets après l'adresse pointée par le registre P.

J -> d  
répéter:  
(P) <-> (Q), P+1, Q+1, d-1  
jusqu'à ce que:  
d=FF

Code HEX: 0B                      Code DEC: 11  
Code BIN: 0 0 0 0 1 0 1 1      Indic. :  
Cycles : 6 + 3d                      Octets : 01

Remarques: Le registre d prend la valeur attribuée au registre J.  
Les registres P et Q sont incrémentés.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LIA 06	02 06
	xx02	LIB 20	03 20
	xx04	LP 04	84
	xx05	LIQ 02	13 02
	xx07	LIJ 01	01 01
	xx09	EXB	0B
	xx0A	RTN	37

Description: Ce programme place le contenu de l'accumulateur (&06) et du registre B (&20) dans le registre X (&2006).  
Le registre P prend la valeur &06 et le registre Q &04.

## EXBD

Cette instruction échange le contenu de d+1 octets, à partir de l'adresse référencée par le registre DP, avec le contenu des adresses placées d+1 octets après l'adresse pointée par le registre P.

J -> d  
répéter:  
(P) <-> (DP), P+1, Q+1, d-1  
jusqu'à ce que:  
d=FF

Code HEX: 1B                      Code DEC: 27  
Code BIN: 0 0 0 1 1 0 1 1      Indic. :  
Cycles : 7 + 6d                  Octets : 01

**Remarques:** Le registre d prend la valeur attribuée au registre J.  
Les registres P et DP sont incrémentés.  
Cette instruction permet de copier le contenu de la RAM interne dans la RAM externe.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LIA 06	02 06
	xx02	LIB 20	03 20
	xx04	LP 02	82
	xx05	LIDP xx20	10 xx 20
	xx08	LIJ 01	01 01
	xx0A	EXBD	1B
	xx0B	RTN	37

**Description:** Ce programme place le contenu de l'accumulateur (&06) et du registre B (&20) dans les adresses &xx20 et &xx21.

Le registre P prend la valeur &06 et le registre DP &xx22.

## EXW

Cette instruction échange le contenu de d+1 octets, à partir de l'adresse référencée par le registre Q, avec le contenu des adresses placées d+1 octets après l'adresse pointée par le registre P.

I -> d  
répéter:  
(P) <-> (Q), P+1, Q+1, d-1  
jusqu'à ce que:  
d=FF

Code HEX: 09                      Code DEC: 09  
Code BIN: 0 0 0 0 1 0 0 1      Indic. :  
Cycles : 6 + 3d                  Octets : 01

**Remarques:** Le registre d prend la valeur attribuée au registre I.  
Les registres P et Q sont incrémentés.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LIA 06	02 06
	xx02	LIB 20	03 20
	xx04	LP 04	84
	xx05	LIQ 02	13 02
	xx07	LII 01	00 01
	xx09	EXW	09
	xx0A	RTN	37

**Description:** Ce programme place le contenu de l'accumulateur (&06) et du registre B (&20) dans le registre X (&2006).  
Le registre P prend la valeur &06 et le registre Q &04.

## EXWD

Cette instruction échange le contenu de d+1 octets, à partir de l'adresse référencée par le registre DP, avec le contenu des adresses placées d+1 octets après l'adresse pointée par le registre P.

I -> d  
répéter:  
(P) <-> (DP), P+1, DP+1, d-1  
jusqu'à ce que:  
d=FF

Code HEX: 19                      Code DEC: 25  
Code BIN: 0 0 0 1 1 0 0 1      Indic. :  
Cycles : 7 + 6d                  Octets : 01

**Remarques:** Le registre d prend la valeur attribuée au registre I.  
Les registres P et DP sont incrémentés.  
Cette instruction permet de copier le contenu de la RAM interne dans la RAM externe.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LIA 06	02 06
	xx02	LIB 20	03 20
	xx04	LP 02	82
	xx05	LIDP xx20	10 xx 20
	xx08	LII 01	00 01
	xx0A	EXWD	19
	xx0B	RTN	37

**Description:** Ce programme place le contenu de l'accumulateur (&06) et du registre B (&20) dans les adresses &xx20 et &xx21.

Le registre P prend la valeur &06 et le registre DP &xx22.

## FILD

Cette instruction remplit une zone mémoire de la RAM avec une constante. Le contenu de l'accumulateur est écrit dans les adresses situées d+1 octets après l'octet référencé par le registre DP.

I -> d  
répéter:  
A -> (DP), DP+1, d-1  
jusqu'à ce que:  
d=FF

Code HEX: 1F                      Code DEC: 31  
Code BIN: 0 0 0 1 1 1 1 1      Indic. :  
Cycles : 4 + 3d                  Octets : 01

**Remarques:** Le registre d prend la valeur attribuée au registre I.  
Le registre DP est incrémenté.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LIA FF	02 FF
	xx02	LIDP xx20	10 xx 20
	xx05	LII 01	00 01
	xx07	FILD	1F
	xx08	RTN	37

**Description:** Ce programme copie le contenu de l'accumulateur (&FF) aux adresses de la RAM externe &xx20 et &xx21.  
Le registre DP contient alors la valeur &xx02.

## FILM

Cette instruction remplit une zone mémoire de la RAM avec une constante. Le contenu de l'accumulateur est écrit dans les adresses situées d+1 octets après l'octet référencé par le registre P.

I -> d  
répéter:  
A -> (P), P+1, d-1  
jusqu'à ce que:  
d=FF

Code HEX: 1E                      Code DEC: 30  
Code BIN: 0 0 0 1 1 1 1 0      Indic. :  
Cycles : 5 + d                      Octets : 01

Remarques: Le registre d prend la valeur attribuée au registre I.  
Le registre P est incrémenté.

Exemple:	Adresse	Assembleur	Code HEX
xx00	LIA FF		02 FF
xx02	LP 08		88
xx03	LII 01		00 01
xx05	FILM		1E
xx06	RTN		37

Description: Ce programme copie le contenu de l'accumulateur (&FF) aux adresses de la RAM interne &08 et &09.  
Le registre P contient alors la valeur &0A.

## INA

Cette instruction lit un octet sur le port IA et le place dans l'accumulateur.

Port-IA -> A

Code HEX: 4C                      Code DEC: 76  
Code BIN: 0 1 0 0 1 1 0 0      Indic. : Z  
Cycles : 2                      Octets : 01

Remarques: Avant d'utiliser l'instruction INA, il est nécessaire de rafraichir le contenu du port IA à l'aide de l'instruction OUTA. Pour plus de détails sur les adresses des ports, référez-vous aux instructions OUT..

Exemple:	Adresse	Assembleur	Code HEX
xx00	LIP 5C		12 5C
xx02	LIA 00		02 00
xx04	EXAM		DB
xx05	OUTA		5D
xx06	LIDP 3E00		10 3E 00
xx09	LIA 3F		02 3F
xx0B	STD		52
xx0C	INA		4C
xx0D	CPIA 00		67 00
xx0F	JRZM 04		39 04
xx11	RTN		37

Description: Ce programme a été spécialement conçu pour les Sharp PC-1350 et 1450. Il utilise le port d'adresse &3E00 qui n'existe pas sur les autres modèles. Il est cependant tout à fait possible de développer un programme similaire destiné aux autres ordinateurs de poche Sharp. Ce programme adresse la matrice de clavier des PC 1350 et 1450. Une matrice de clavier se compose d'une série de liaisons qui relient le clavier au microprocesseur (cf. diagramme en annexe). D'un côté de cette matrice nous trouvons les key-port et de l'autre, le

port, IA. Celui-ci permet de transmettre et également de lire des données.

Si l'on compare la matrice de clavier des PC 1350 et 1450 aux ports, on remarque que les key-ports utilisent 6 bits tandis que le port-IA en utilise 7.

Si l'on veut contrôler l'état d'une touche du clavier, il faut d'abord mettre le port à 0. Pour ce faire, il suffit de placer cette valeur dans le registre &5C et d'utiliser l'instruction OUTA. Le port-IA est alors prêt à recevoir des informations.

Pour initialiser les 6 bits des Key-ports on place la valeur &3F à l'adresse &3E00. (pour les autres modèles, il est nécessaire d'utiliser l'instruction OUTB).

Le port est contrôlé jusqu'à ce que le contenu de l'accumulateur soit différent de 0. La valeur contenue dans l'accumulateur correspond à la touche du clavier utilisée.

## INB

Cette instruction lit un octet sur le port-IB et le place dans l'accumulateur.

Port-IB -> A

Code HEX: CC                      Code DEC: 204  
Code BIN: 1 1 0 0 1 1 0 0      Indic. : Z  
Cycles : 2                      Octets : 01

**Remarques:** Avant d'utiliser l'instruction INB, il est nécessaire de rafraîchir le contenu du port IB à l'aide de l'instruction OUTB. Pour plus de détails sur les adresses des ports, référez-vous aux instructions OUT..

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LIP 5D	12 5D
	xx02	LIA 00	02 00
	xx04	EXAM	DB
	xx05	OUTB	DD
	xx06	INB	CC
	xx07	TSIA 40	66 40
	xx09	JRZM 04	39 04
	xx0B	RTN	37

**Description:** Ce programme interroge le bit 6 du port-IB. La ligne de transmission de données de ce bit est reliée au connecteur 11 broches. Il peut ainsi être positionné à 1 à l'aide d'un signal électrique.

**ATTENTION!** L'ordinateur de poche utilise une technologie CMOS. Toute surcharge électrique peut provoquer la destruction du microprocesseur et d'autres composants!



## INCA

Cette instruction incrémente le contenu de l'accumulateur (registre A) d'une unité.

$A + 1 \rightarrow A \quad C, Z$

Code HEX: 42                      Code DEC: 66  
Code BIN: 0 1 0 0 0 0 1 0      Indic. : CZ  
Cycles : 4                      Octets : 01

**Remarques:** Cette instruction modifie le contenu du registre Q.  
Les indicateurs sont positionnés en fonction du résultat.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LIA 03	02 03
	xx02	INCA	42
	xx03	RTN	37

**Description:** Après exécution du programme, le contenu de l'accumulateur est égal à 4. Le registre Q a la même valeur.

## INCB

Cette instruction incrémente de une unité le contenu du registre B.

$B + 1 \rightarrow B \quad C, Z$

Code HEX: C2                      Code DEC: 194  
Code BIN: 1 1 0 0 0 0 1 0      Indic. : CZ  
Cycles : 4                      Octets : 01

**Remarques:** Cette instruction modifie le contenu du registre Q.  
Les indicateurs sont positionnés en fonction du résultat.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LIB 03	03 03
	xx02	INCB	C2
	xx03	RTN	37

**Description:** Après exécution du programme, le contenu du registre B est égal à 4. Le registre Q prend la valeur 3.

## INCI

Cette instruction incrémente de une unité le contenu du registre I.

$I + 1 \rightarrow I$  C,Z

Code HEX: 40                      Code DEC: 64  
Code BIN: 0 1 0 0 0 0 0 0      Indic. : CZ  
Cycles : 4                      Octets : 01

**Remarques:** Cette instruction modifie le contenu du registre Q.  
Les indicateurs sont positionnés en fonction du résultat.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LII 03	00 03
	xx02	INCI	40
	xx03	RTN	37

**Description:** Après exécution du programme, le contenu du registre I est égal à 4. Le registre Q prend la valeur 0.

## INCJ

Cette instruction incrémente de une unité le contenu du registre J.

$J + 1 \rightarrow J$  C,Z

Code HEX: C0                      Code DEC: 192  
Code BIN: 1 1 0 0 0 0 0 0      Indic. : CZ  
Cycles : 4                      Octets : 01

**Remarques:** Cette instruction modifie le contenu du registre Q.  
Les indicateurs sont positionnés en fonction du résultat.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LIJ 03	01 03
	xx02	INCJ	C0
	xx03	RTN	37

**Description:** Après exécution du programme, le contenu du registre J est égal à 4. Le registre Q prend la valeur 1.

## INCK

Cette instruction incrémente de une unité le contenu du registre K.

$K + 1 \rightarrow K \quad C, Z$

Code HEX: 48                      Code DEC: 72  
Code BIN: 0 1 0 0 1 0 0 0      Indic. : CZ  
Cycles : 4                      Octets : 01

**Remarques:** Cette instruction modifie le contenu du registre Q.  
Les indicateurs sont positionnés en fonction du résultat.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LP 08	88
	xx01	LIA 03	02 03
	xx03	EXAM	DB
	xx04	INCK	48
	xx05	RTN	37

**Description:** Pour charger la valeur &03 dans le registre K, il faut d'abord placer cette même valeur dans l'accumulateur et ensuite transférer la valeur dans le registre K à l'aide de l'instruction EXAM. Après exécution du programme, le contenu du registre K est égal à 4. Le registre Q prend la valeur 8.

## INCL

Cette instruction incrémente de une unité le contenu du registre L.

$L + 1 \rightarrow L \quad C, Z$

Code HEX: C8                      Code DEC: 200  
Code BIN: 1 1 0 0 1 0 0 0      Indic. : CZ  
Cycles : 4                      Octets : 01

**Remarques:** Cette instruction modifie le contenu du registre Q.  
Les indicateurs sont positionnés en fonction du résultat.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LP 09	89
	xx01	LIA 03	02 03
	xx03	EXAM	DB
	xx04	INCL	C8
	xx05	RTN	37

**Description:** Pour charger la valeur &03 dans le registre L, il faut d'abord placer cette même valeur dans l'accumulateur et ensuite transférer la valeur dans le registre L à l'aide de l'instruction EXAM. Après exécution du programme, le contenu du registre L est égal à 4. Le registre Q prend la valeur 9.

## INCM

Cette instruction incrémente de une unité le contenu du registre M.

$M + 1 \rightarrow M \quad C, Z$

Code HEX: 4A                      Code DEC: 74  
Code BIN: 0 1 0 0 1 0 1 0      Indic. : CZ  
Cycles : 4                      Octets : 01

**Remarques:** Cette instruction modifie le contenu du registre Q.  
Les indicateurs sont positionnés en fonction du résultat.

Cette instruction ne fait pas partie du jeu d'instructions standard publié par Sharp.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LP 0A	8A
	xx01	LIA 03	02 03
	xx03	EXAM	DB
	xx04	INCM	4A
	xx05	RTN	37

**Description:** Pour charger la valeur &03 dans le registre M, il faut d'abord placer cette même valeur dans l'accumulateur et ensuite transférer la valeur dans le registre M à l'aide de l'instruction EXAM. Après exécution du programme, le contenu du registre M est égal à 4. Le registre Q prend la valeur &0A.

## INCN

Cette instruction incrémente de une unité le contenu du registre N.

$N + 1 \rightarrow N \quad C, Z$

Code HEX: CA                      Code DEC: 202  
Code BIN: 1 1 0 0 1 0 1 0      Indic. : CZ  
Cycles : 4                      Octets : 01

**Remarques:** Cette instruction modifie le contenu du registre Q.  
Les indicateurs sont positionnés en fonction du résultat.

Cette instruction ne fait pas partie du jeu d'instructions standard publié par Sharp.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LP 0B	8B
	xx01	LIA 03	02 03
	xx03	EXAM	DB
	xx04	INCN	CA
	xx05	RTN	37

**Description:** Pour charger la valeur &03 dans le registre N, il faut d'abord placer cette même valeur dans l'accumulateur et ensuite transférer la valeur dans le registre N à l'aide de l'instruction EXAM. Après exécution du programme, le contenu du registre N est égal à 4. Le registre Q prend la valeur &0B.

## INCP

Cette instruction incrémente de une unité le contenu du registre P.

P + 1 -> P

Code HEX: 50                      Code DEC: 80  
Code BIN: 0 1 0 1 0 0 0 0      Indic. : CZ  
Cycles : 2                      Octets : 01

**Remarques:** Cette instruction modifie le contenu du registre Q.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LP 09	89
	xx01	INCP	51
	xx05	RTN	37

**Description:** Ce programme place la constante &09 dans le registre P. L'instruction INCP incrémente le contenu du registre P d'une unité. Après exécution du programme, le contenu du registre P est égal à &0A.

## IX

Cette instruction ajoute une unité au contenu du registre X (16 bits) et place la valeur incrémentée dans le registre DP.

XH,XL -> DP  
DP+1 -> DP,X

Code HEX: 04                      Code DEC: 04  
Code BIN: 0 0 0 0 0 1 0 0      Indic. :  
Cycles : 6                      Octets : 01

**Remarques:** Cette instruction modifie le contenu du registre Q.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LP 04	84
	xx01	LIA 20	02 20
	xx03	EXAM	DB
	xx04	LP 05	85
	xx05	LIA 27	02 27
	xx07	EXAM	DB
	xx08	IX	04
	xx09	RTN	37

**Description:** Ce programme place la valeur &2720 dans le registre X et l'incrémente. Le résultat &2721 se trouve à la fois dans le registre X et le registre DP. Il est à noter que le nombre &2720 est chargé sous forme d'octet de poids fort et de poids faible car il n'existe aucune instruction de chargement d'une valeur sur 16 bits pour le registre X. Après exécution du programme, le registre Q prend la valeur &05.

## IXL

Cette instruction effectue les opérations suivantes:

- Elle ajoute une unité au contenu du registre X (16 bits).
- Elle copie la valeur incrémentée dans le registre DP et dans le registre X.
- Elle place dans l'accumulateur le contenu de l'adresse ainsi obtenue.

X	-> DP
DP+1	-> DP, X
(DP)	-> A

$X \leftarrow X + 1$   
 $DP \leftarrow X$   
 $A \leftarrow (DP)$

Code HEX: 24                      Code DEC: 36  
Code BIN: 0 0 1 0 0 1 0 0      Indic. :  
Cycles : 7                      Octets : 01

Remarques: Cette instruction modifie le contenu du registre Q.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LP 04	84
	xx01	LIA 20	02 20
	xx03	EXAM	DB
	xx04	LP 05	85
	xx05	LIA 27	02 27
	xx07	EXAM	DB
	xx08	IXL	24
	xx09	RTN	37

Description: Ce programme place la valeur &2720 dans le registre X et l'incrmente. Le résultat &2721 se trouve à la fois dans le registre X et le registre DP. Enfin, le contenu de l'adresse &2721 est placé dans l'accumulateur. Il est à noter que le nombre &2720 est chargé sous for-

me d'octet de poids fort et de poids faible car il n'existe aucune instruction de chargement d'une valeur sur 16 bits pour le registre X. Après exécution du programme, le registre Q prend la valeur &05.

## IY

Cette instruction ajoute une unité au contenu du registre Y (16 bits) et place la valeur incrémentée dans le registre DP.

YH,YL -> DP  
DP+1 -> DP,Y

Code HEX: 06                      Code DEC: 06  
Code BIN: 0 0 0 0 0 1 1 0      Indic. :  
Cycles : 6                      Octets : 01

**Remarques:** Cette instruction modifie le contenu du registre Q.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LP 06	86
	xx01	LIA 20	02 20
	xx03	EXAM	DB
	xx04	LP 07	87
	xx05	LIA 27	02 27
	xx07	EXAM	DB
	xx08	IY	06
	xx09	RTN	37

**Description:** Ce programme place la valeur &2720 dans le registre Y et l'incrémente. Le résultat &2721 se trouve à la fois dans le registre Y et le registre DP. Il est à noter que le nombre &2720 est chargé sous forme d'octet de poids fort et de poids faible car il n'existe aucune instruction de chargement d'une valeur sur 16 bits pour le registre Y.

Après exécution du programme, le registre Q prend la valeur &07.

## IYS

Cette instruction effectue les opérations suivantes:

- Elle ajoute une unité au contenu du registre Y (16 bits).
- Elle copie la valeur incrémentée dans les registres DP et Y.
- Elle place le contenu de l'accumulateur à l'adresse ainsi obtenue.

Y -> DP  
DP+1 -> DP,Y  
A -> (DP)

Code HEX: 26                      Code DEC: 38  
Code BIN: 0 0 1 0 0 1 1 0      Indic. :  
Cycles : 6                      Octets : 01

**Remarques:** Cette instruction modifie le contenu du registre Q.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LP 06	86
	xx01	LIA 20	02 20
	xx03	EXAM	DB
	xx04	LP 07	87
	xx05	LIA 27	02 27
	xx07	EXAM	DB
	xx08	LIA FF	02 FF
	xx0A	IYS	26
	xx0B	RTN	37

**Description:** Ce programme place la valeur &2720 dans le registre Y et l'incrémente. Le résultat &2721 se trouve à la fois dans le registre Y et le registre DP. Enfin, le contenu &FF de l'accumulateur est placé à l'adresse &2721. Il

est à noter que le nombre &2720 est chargé sous forme d'octet de poids fort et de poids faible car il n'existe aucune instruction de chargement d'une valeur sur 16 bits pour le registre Y.

Après exécution du programme, le registre Q prend la valeur &07.

## JP nm

Cette instruction provoque un branchement absolu inconditionnel à l'adresse nm (16 bits). Aucune condition n'est testée.

n -> PCH
m -> PCL

Code HEX: 79                      Code DEC: 121  
 Code BIN: 0 1 1 1 1 0 0 1      Indic. :  
 Cycles : 6                      Octets : 03

**Remarques:** PCH correspond à l'octet de poids fort du compteur programme et PCL à l'octet de poids faible.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LIA 01	02 01
	xx02	JP xx07	79 xx 07
	xx05	LIA 03	02 03
	xx07	RTN	37

**Description:** Ce programme permet de sauter l'exécution de l'instruction LIA 03 et d'exécuter immédiatement RTN (adresse xx07). Après exécution du programme, le contenu de l'accumulateur est toujours égal à &01.



## JPC nm

Cette instruction provoque un branchement absolu inconditionnel à l'adresse nm (16 bits) si l'indicateur de retenue est positionné. Dans le cas contraire, l'ordinateur exécute l'instruction qui suit l'instruction de branchement.

Si C=1 alors n -> PCH, m -> PCL  
Si C=0 alors PC+3 -> PC

Code HEX: 7F                      Code DEC: 127  
Code BIN: 0 1 1 1 1 1 1 1      Indic. :  
Cycles : 6                      Octets : 03

Remarques: PCH correspond à l'octet de poids fort du compteur programme et PCL à l'octet de poids faible.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LIA 01	02 01
	xx02	CPIA 10	67 10
	xx04	JPC xx0A	7F xx 0A
	xx07	LIB 05	03 05
	xx09	RTN	37
	xx0A	LIB 50	03 50
	xx0C	RTN	37

**Description:** Le contenu de l'accumulateur (&01) est comparé à la constante &10. Si le contenu de l'accumulateur est inférieur à la constante (ce qui correspond à notre exemple), l'indicateur de retenue est positionné et le branchement à l'adresse xx0A exécuté. Ensuite, le programme place la valeur &50 dans le registre B. Si vous remplacez l'instruction LIA 01 par LIA 20, le branchement n'est pas exécuté et le programme place la valeur &05 dans le registre B.

## JPNC nm

Cette instruction provoque un branchement inconditionnel absolu à l'adresse nm (16 bits) si l'indicateur de retenue n'est pas positionné. Dans le cas contraire, l'ordinateur exécute l'instruction qui suit l'instruction de branchement.

Si C=0 alors n -> PCH, m -> PCL  
Si C=1 alors PC+3 -> PC

Code HEX: 7D                      Code DEC: 125  
Code BIN: 0 1 1 1 1 1 0 1      Indic. :  
Cycles : 6                      Octets : 03

Remarques: PCH correspond à l'octet de poids fort du compteur programme et PCL à l'octet de poids faible.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LIA 01	02 01
	xx02	CPIA 10	67 10
	xx04	JPNC xx0A	7D xx 0A
	xx07	LIB 05	03 05
	xx09	RTN	37
	xx0A	LIB 50	03 50
	xx0C	RTN	37

**Description:** Le contenu de l'accumulateur (&01) est comparé à la constante &10. Si le contenu de l'accumulateur est inférieur à la constante (ce qui correspond à notre exemple), l'indicateur de retenue est positionné et le branchement à l'adresse xx0A n'est pas exécuté. Ensuite, le programme place la valeur &05 dans le registre B. Si vous remplacez l'instruction LIA 01 par LIA 20, le branchement est exécuté et le programme place la valeur &50 dans le registre B.

## JPNZ nm

Cette instruction provoque un branchement inconditionnel absolu à l'adresse nm (16 bits) si l'indicateur de nullité n'est pas positionné. Dans le cas contraire, l'ordinateur exécute l'instruction qui suit l'instruction de branchement.

Si Z=0 alors n -> PCH, m -> PCL  
Si Z=1 alors PC+3 -> PC

Code HEX: 7C                      Code DEC: 124  
Code BIN: 0 1 1 1 1 0 0      Indic. :  
Cycles : 6                      Octets : 03

Remarques: PCH correspond à l'octet de poids fort du compteur programme et PCL à l'octet de poids faible.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LIA 01	02 01
	xx02	CPIA 10	67 10
	xx04	JPNZ xx0A	7C xx 0A
	xx07	LIB 05	03 05
	xx09	RTN	37
	xx0A	LIB 50	03 50
	xx0C	RTN	37

**Description:** Le contenu de l'accumulateur (&01) est comparé à la constante &10. Si le contenu de l'accumulateur est inférieur à la constante (ce qui correspond à notre exemple), l'indicateur de nullité est positionné et le branchement à l'adresse xx0A n'est pas exécuté. Ensuite, le programme place la valeur &05 dans le registre B.  
Si vous remplacez l'instruction LIA 01 par LIA 20, le branchement est exécuté et le programme place la valeur &50 dans le registre B.

## JPZ nm

Cette instruction provoque un branchement inconditionnel absolu à l'adresse nm (16 bits) si l'indicateur de nullité est positionné. Dans le cas contraire, l'ordinateur exécute l'instruction qui suit l'instruction de branchement.

Si Z=1 alors n -> PCH, m -> PCL  
Si Z=0 alors PC+3 -> PC

Code HEX: 7E                      Code DEC: 126  
Code BIN: 0 1 1 1 1 1 0      Indic. :  
Cycles : 6                      Octets : 03

Remarques: PCH correspond à l'octet de poids fort du compteur programme et PCL à l'octet de poids faible.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LIA 01	02 01
	xx02	CPIA 10	67 10
	xx04	JPZ xx0A	7E xx 0A
	xx07	LIB 05	03 05
	xx09	RTN	37
	xx0A	LIB 50	03 50
	xx0C	RTN	37

**Description:** Le contenu de l'accumulateur (&01) est comparé à la constante &10. Si le contenu de l'accumulateur est inférieur à la constante (ce qui correspond à notre exemple), l'indicateur de nullité est positionné et le branchement à l'adresse xx0A exécuté. Ensuite, le programme place la valeur &50 dans le registre B.  
Si vous remplacez l'instruction LIA 01 par LIA 20, le branchement n'est pas exécuté et le programme place la valeur &05 dans le registre B.

## JRCM n

Cette instruction provoque un branchement conditionnel relatif avec déplacement en arrière de n octets en mémoire si l'indicateur de retenue est positionné.

Dans le cas contraire, l'ordinateur exécute l'instruction qui suit l'instruction de branchement.

Si C=1 alors PC+1-n -> PC  
Si C=0 alors PC+2 -> PC

Code HEX: 3B                      Code DEC: 59

Code BIN: 0 0 1 1 1 0 1 1      Indic. :

Cycles : 7 si branch. /4      Octets : 02

**Remarques:** Le nombre de cycles dépend de l'exécution ou non du branchement.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LIB 50	03 50
	xx02	RTN	37
	xx03	LIA 01	02 01
	xx05	CPIA 10	67 10
	xx07	JRCM 08	3B 08
	xx09	LIB 05	03 05
	xx0B	RTN	37

**Description:** Lancez l'exécution du programme par CALL xx03.

Le contenu de l'accumulateur (&01) est comparé à la constante &10. Si le contenu de l'accumulateur est inférieur à la constante (ce qui correspond à notre exemple), l'indicateur de retenue est positionné et le branchement à l'adresse xx00 exécuté. Ensuite, le programme place la valeur &50 dans le registre B.

Si vous remplacez l'instruction LIA 01 par LIA 20, le branchement n'est pas exécuté et le programme place la valeur &05 dans le registre B.

## JRCP n

Cette instruction provoque un branchement conditionnel relatif avec déplacement en avant de n octets en mémoire si l'indicateur de retenue est positionné.

Dans le cas contraire, l'ordinateur exécute l'instruction qui suit l'instruction de branchement.

Si C=1 alors PC+1+n -> PC  
Si C=0 alors PC+2 -> PC

Code HEX: 3A                      Code DEC: 58

Code BIN: 0 0 1 1 1 0 1 0      Indic. :

Cycles : 7 si branch. /4      Octets : 02

**Remarques:** Le nombre de cycles dépend de l'exécution ou non du branchement.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LIA 01	02 01
	xx02	CPIA 10	67 10
	xx04	JRCP 04	3A 04
	xx06	LIB 05	03 05
	xx08	RTN	37
	xx09	LIB 50	03 50
	xx0B	RTN	37

**Description:** Le contenu de l'accumulateur (&01) est comparé à la constante &10. Si le contenu de l'accumulateur est inférieur à la constante (ce qui correspond à notre exemple), l'indicateur de retenue est positionné et le branchement à l'adresse xx09 (xx04 + 1 + 4 = xx09) exécuté. Ensuite, le programme place la valeur &50 dans le registre B.

Si vous remplacez l'instruction LIA 01 par LIA 20, le branchement n'est pas exécuté et le programme place la valeur &05 dans le registre B.

## JRM n

Cette instruction provoque un branchement inconditionnel relatif avec déplacement en arrière de n octets.

PC + 1 - n -> PC

Code HEX: 2D                      Code DEC: 45  
Code BIN: 0 0 1 0 1 1 0 1      Indic. :  
Cycles : 7                      Octets : 02

Remarques: Aucune condition n'est testée.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	RTN	37
	xx01	LIA 01	02 01
	xx03	JRM 04	2D 04
	xx05	LIA 05	02 05
	xx07	NOPT	CE

**Description:** Lancez l'exécution du programme par CALL xx01. L'accumulateur est chargé avec la valeur &01. Ensuite, l'instruction JRM 04 provoque un branchement à l'adresse xx00 et l'exécution de l'instruction RTN. L'instruction LIA &05 n'est donc jamais exécutée.

## JRNCM n

Cette instruction provoque un branchement conditionnel relatif avec déplacement en arrière de n octets si l'indicateur de retenue n'est pas positionné.

Dans le cas contraire, l'ordinateur exécute l'instruction qui suit l'instruction de branchement.

Si C=0 alors PC+1-n -> PC  
Si C=1 alors PC+2 -> PC

Code HEX: 2B                      Code DEC: 43  
Code BIN: 0 0 1 0 1 0 1 1      Indic. :  
Cycles : 7 si branch. /4      Octets : 02

Remarques: Le nombre de cycles dépend de l'exécution ou non du branchement.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LIB 50	03 50
	xx02	RTN	37
	xx03	LIA 01	02 01
	xx05	CPIA 10	67 10
	xx07	JRNCM 08	2B 08
	xx09	LIB 05	03 05
	xx0B	RTN	37

**Description:** Lancez l'exécution du programme par CALL xx03.

Le contenu de l'accumulateur (&01) est comparé à la constante &10. Si le contenu de l'accumulateur est inférieur à la constante (ce qui correspond à notre exemple), l'indicateur de retenue est positionné et le branchement à l'adresse xx00 n'est pas exécuté. Ensuite, le programme place la valeur &05 dans le registre B.

Si vous remplacez l'instruction LIA 01 par LIA 20, le branchement est exécuté et le programme place la valeur &50 dans le registre B.

## JRNCP n

Cette instruction provoque un branchement conditionnel relatif avec déplacement en avant de n octets si l'indicateur de retenue n'est pas positionné.

Dans le cas contraire, l'ordinateur exécute l'instruction qui suit l'instruction de branchement.

Si C=0 alors PC+1+n -> PC  
Si C=1 alors PC+2 -> PC

Code HEX: 2A                      Code DEC: 42  
Code BIN: 0 0 1 0 1 0 1 0      Indic. :  
Cycles : 7 si branch. /4      Octets : 02

**Remarques:** Le nombre de cycles dépend de l'exécution ou non du branchement.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LIA 01	02 01
	xx02	CPIA 10	67 10
	xx04	JRNCP 04	2A 04
	xx06	LIB 05	03 05
	xx08	RTN	37
	xx09	LIB 50	03 50
	xx0B	RTN	37

**Description:** Le contenu de l'accumulateur (&01) est comparé à la constante &10. Si le contenu de l'accumulateur est inférieur à la constante (ce qui correspond à notre exemple), l'indicateur de retenue est positionné et le branchement à l'adresse xx09 (xx04 + 1 + 4 = xx09) n'est pas exécuté. Ensuite, le programme place la valeur &05 dans le registre B.

Si vous remplacez l'instruction LIA 01 par LIA 20, le branchement est exécuté et le programme place la valeur &50 dans le registre B.

## JRNZM n

Cette instruction provoque un branchement conditionnel relatif avec déplacement en arrière de n octets si l'indicateur de nullité n'est pas positionné.

Dans le cas contraire, l'ordinateur exécute l'instruction qui suit l'instruction de branchement.

Si Z=0 alors PC+1-n -> PC  
Si Z=1 alors PC+2 -> PC

Code HEX: 29                      Code DEC: 41  
Code BIN: 0 0 1 0 1 0 0 1      Indic. :  
Cycles : 7 si branch. /4      Octets : 02

**Remarques:** Le nombre de cycles dépend de l'exécution ou non du branchement.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LIB 50	03 50
	xx02	RTN	37
	xx03	LIA 10	02 10
	xx05	CPIA 10	67 10
	xx07	JRNZM 08	29 08
	xx09	LIB 05	03 05
	xx0B	RTN	37

**Description:** Lancez l'exécution du programme par CALL xx03.

Le contenu de l'accumulateur (&10) est comparé à la constante &10. Si le contenu de l'accumulateur est égal à la constante (ce qui correspond à notre exemple), l'indicateur de nullité est positionné et le branchement à l'adresse xx00 n'est pas exécuté. Ensuite, le programme place la valeur &05 dans le registre B.

Si vous remplacez l'instruction LIA 10 par LIA 20, le branchement est exécuté et le programme place la valeur &50 dans le registre B.

## JRNZP n

Cette instruction provoque un branchement conditionnel relatif avec déplacement en avant de n octets si l'indicateur de nullité n'est pas positionné.

Dans le cas contraire, l'ordinateur exécute l'instruction qui suit l'instruction de branchement.

Si Z=0 alors PC+l+n -> PC  
Si Z=1 alors PC+2 -> PC

Code HEX: 28                      Code DEC: 40  
Code BIN: 0 0 1 0 1 0 0 0      Indic. :  
Cycles : 7 si branch. /4      Octets : 02

**Remarques:** Le nombre de cycles dépend de l'exécution ou non du branchement.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LIA 10	02 10
	xx02	CPIA 10	67 10
	xx04	JRNZP 04	28 04
	xx06	LIB 05	03 05
	xx08	RTN	37
	xx09	LIB 50	03 50
	xx0B	RTN	37

**Description:** Le contenu de l'accumulateur (&10) est comparé à la constante &10. Si le contenu de l'accumulateur est égal à la constante (ce qui correspond à notre exemple), l'indicateur de nullité est positionné et le branchement à l'adresse xx09 (xx04 + 1 + 4 = xx09) n'est pas exécuté. Ensuite, le programme place la valeur &05 dans le registre B.

Si vous remplacez l'instruction LIA 10 par LIA 20, le branchement est exécuté et le programme place la valeur &50 dans le registre B.

## JRP n

Cette instruction provoque un branchement inconditionnel relatif avec déplacement en avant de n octets.

PC + 1 + n -> PC

Code HEX: 2C                      Code DEC: 44  
Code BIN: 0 0 1 0 1 1 0 0      Indic. :  
Cycles : 7                      Octets : 02

**Remarques:** Aucune condition n'est testée.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LIA 01	02 01
	xx02	JRP 03	2C 03
	xx04	LIA 03	02 03
	xx06	RTN	37

**Description:** L'accumulateur est chargé avec la valeur &01. Ensuite, l'instruction JRP 03 provoque un branchement à l'adresse xx06 et l'exécution de l'instruction RTN. L'instruction LIA &03 n'est donc jamais exécutée.

## JRZM n

Cette instruction provoque un branchement relatif conditionnel avec déplacement en arrière de n octets si l'indicateur de nullité est positionné.

Dans le cas contraire, l'ordinateur exécute l'instruction qui suit l'instruction de branchement.

Si Z=1 alors PC+1-n -> PC  
Si Z=0 alors PC+2 -> PC

Code HEX: 39                      Code DEC: 57  
Code BIN: 0 0 1 1 1 0 0 1      Indic. :  
Cycles : 7 si branch. /4      Octets : 02

**Remarques:** Le nombre de cycles dépend de l'exécution ou non du branchement.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LIB 50	03 50
	xx02	RTN	37
	xx03	LIA 10	02 10
	xx05	CPIA 10	67 10
	xx07	JRZM 08	39 08
	xx09	LIB 05	03 05
	xx0B	RTN	37

**Description:** Lancez l'exécution du programme par CALL xx03.

Le contenu de l'accumulateur (&10) est comparé à la constante &10. Si le contenu de l'accumulateur est égal à la constante (ce qui correspond à notre exemple), l'indicateur de nullité est positionné et le branchement à l'adresse xx00 exécuté. Ensuite, le programme place la valeur &50 dans le registre B.

Si vous remplacez l'instruction LIA 10 par LIA 20, le branchement n'est pas exécuté et le programme place la valeur &05 dans le registre B.

## JRZP n

Cette instruction provoque un branchement conditionnel relatif avec déplacement en avant de n octets si l'indicateur de nullité est positionné.

Dans le cas contraire, l'ordinateur exécute l'instruction qui suit l'instruction de branchement.

Si Z=1 alors PC+1+n -> PC  
Si Z=0 alors PC+2 -> PC

Code HEX: 38                      Code DEC: 56  
Code BIN: 0 0 1 1 1 0 0 0      Indic. :  
Cycles : 7 si branch. /4      Octets : 02

**Remarques:** Le nombre de cycles dépend de l'exécution ou non du branchement.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LIA 10	02 10
	xx02	CPIA 10	67 10
	xx04	JRZP 04	38 04
	xx06	LIB 05	03 05
	xx08	RTN	37
	xx09	LIB 50	03 50
	xx0B	RTN	37

**Description:** Le contenu de l'accumulateur (&10) est comparé à la constante &10. Si le contenu de l'accumulateur est égal à la constante (ce qui correspond à notre exemple), l'indicateur de nullité est positionné et le branchement à l'adresse xx09 (xx04 + 1 + 4 = xx09) exécuté. Ensuite, le programme place la valeur &50 dans le registre B.

Si vous remplacez l'instruction LIA 10 par LIA 20, le branchement n'est pas exécuté et le programme place la valeur &05 dans le registre B.

## LEAVE

Cette instruction écrit un 0 au sommet de la pile.

0 -> (R)

Code HEX: D8                      Code DEC: 216  
Code BIN: 1 1 0 1 1 0 0 0      Indic. :  
Cycles : 2                      Octets : 01

Remarques: Cette instruction permet d'écraser par un 0 la valeur placée au sommet de la pile, sans pour autant modifier le pointeur de pile (R).

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LIA 03	02 03
	xx02	PUSH	34
	xx03	LIA 05	02 05
	xx05	LEAVE	D8
	xx06	POP	5B
	xx07	RTN	37

Description: La valeur &03 est placée dans l'accumulateur et sauvegardée sur la pile à l'aide de l'instruction PUSH. Ensuite, LEAVE écrase le sommet de la pile en y mettant un "0". C'est pourquoi, après exécution du programme, l'accumulateur contient la valeur 0. Si LEAVE était remplacé par NOPT, l'accumulateur contiendrait la valeur 3.

## LDD

Cette instruction charge dans l'accumulateur le contenu de l'emplacement mémoire référencé par le registre DP.

(DP) -> A

Code HEX: 57                      Code DEC: 87  
Code BIN: 0 1 0 1 0 1 1 1      Indic. :  
Cycles : 3                      Octets : 01

Remarques: Cette instruction permet de charger une valeur de la RAM ou ROM externe.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LIDP xx02	10 xx 02
	xx03	LDD	57
	xx04	RTN	37

Description: Ce programme charge l'accumulateur avec le contenu de l'adresse &02.



## LDM

Cette instruction charge dans l'accumulateur le contenu de l'emplacement de la RAM interne référencé par le registre P.

(P) -> A

Code HEX: 59                      Code DEC: 89  
Code BIN: 0 1 0 1 1 0 0 1      Indic. :  
Cycles : 2                      Octets : 01

**Remarques:** Cette instruction permet de charger facilement une valeur de la RAM interne.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LIA FF	02 FF
	xx02	LP 08	88
	xx03	EXAM	DB
	xx04	INCK	48
	xx05	LDM	59
	xx06	RTN	37

**Description:** Ce programme échange le contenu de l'accumulateur avec celui du registre K (EXAM). Ensuite, l'instruction INCK incrémente le contenu du registre K. Enfin, LDM replace la valeur obtenue dans l'accumulateur sans modifier le contenu du registre K. Après exécution du programme, l'accumulateur contient &0. En effet, &FF + &01 = &00. Les indicateurs de retenue et de nullité sont positionnés.

## LDP

Cette instruction charge dans l'accumulateur le contenu du registre P.

P -> A

Code HEX: 20                      Code DEC: 32  
Code BIN: 0 0 1 0 0 0 0 0      Indic. :  
Cycles : 2                      Octets : 01

**Remarques:** P constitue un registre sur 7 bits.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LP 08	88
	xx01	INCP	50
	xx02	LDP	20
	xx03	RTN	37

**Description:** Ce programme charge le registre P avec la valeur &08 et l'incrémente. Le résultat &09 est placé dans l'accumulateur.

## LDQ

Cette instruction charge dans l'accumulateur le contenu du registre Q.

Q -> A

Code HEX: 21                      Code DEC: 33  
Code BIN: 0 0 1 0 0 0 0 1      Indic. :  
Cycles : 2                      Octets : 01

Remarques: Q constitue un registre sur 7 bits.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LIQ 08	13 08
	xx02	LDQ	21
	xx03	RTN	37

Description: Ce programme charge le registre Q avec la valeur &08. Cette valeur est ensuite placée dans l'accumulateur.

## LDR

Cette instruction charge dans l'accumulateur le contenu du registre R.

R -> A

Code HEX: 22                      Code DEC: 34  
Code BIN: 0 0 1 0 0 0 1 0      Indic. :  
Cycles : 2                      Octets : 01

Remarques: R constitue un registre sur 7 bits.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LIA 08	02 08
	xx02	PUSH	34
	xx03	LDR	22
	xx04	EXAB	DA
	xx05	POP	5B
	xx06	RTN	37

Description: Ce programme sauvegarde la valeur &08 de l'accumulateur à l'aide de l'instruction PUSH. Le pointeur de pile est décrémenté et prend la valeur &4F. Cette valeur est ensuite placée dans le registre B à l'aide des instruction LDR et EXAB. Enfin, l'instruction POP replace la valeur &08 dans l'accumulateur et incrémente le pointeur de pile R.

## LIA n

Cette instruction charge l'accumulateur (registre &02 de la RAM interne) avec la constante n (8 bits).

n -> A

Code HEX: 02                      Code DEC: 02  
Code BIN: 0 0 0 0 0 0 1 0      Indic. :  
Cycles : 4                      Octets : 02

**Remarques:** Cette instruction permet de placer facilement une valeur dans l'accumulateur.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LIA FF	02 FF
	xx02	RTN	37

**Description:** Ce programme charge l'accumulateur avec la valeur &FF.

## LIB n

Cette instruction charge le registre B (registre &03 de la RAM interne) avec la constante n (8 bits).

n -> B

Code HEX: 03                      Code DEC: 03  
Code BIN: 0 0 0 0 0 0 1 1      Indic. :  
Cycles : 4                      Octets : 02

**Remarques:** Cette instruction permet de placer facilement une valeur dans le registre B.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LIB FF	03 FF
	xx02	RTN	37

**Description:** Ce programme charge le registre B avec la valeur &FF.

## LIDL n

Cette instruction charge l'octet de poids faible du registre DP avec la constante n (8 bits).

n -> DPL

Code HEX: 11                      Code DEC: 17  
Code BIN: 0 0 0 1 0 0 0 1      Indic. :  
Cycles : 5                      Octets : 02

**Remarques:** Cette instruction permet d'économiser de l'espace mémoire. En effet, il suffit d'utiliser une seule fois l'instruction LIDP et ensuite de modifier les octets de poids faible au moyen de l'instruction LIDL. Bien évidemment, cette astuce ne fonctionne que tant que l'octet de poids fort de l'adresse mémoire de la RAM ou ROM externe, référencée par le registre DP, n'est pas modifié.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LIDP xxFF	10 xx FF
	xx03	LIA 20	02 20
	xx05	STD	52
	xx06	LIDL 34	11 34
	xx08	STD	52
	xx09	RTN	37

**Description:** La constante &20 contenue dans l'accumulateur est placée aux adresses xxFF et xx34 de la RAM externe.

## LIDP nm

Cette instruction charge le registre DP avec la constante n (16 bits).

n -> DPH  
m -> DPL

Code HEX: 10                      Code DEC: 16  
Code BIN: 0 0 0 1 0 0 0 0      Indic. :  
Cycles : 8                      Octets : 03

**Remarques:** DPH représente l'octet de poids fort et DPL l'octet de poids faible du registre DP (16 bits).

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LIDP xxFF	10 xx FF
	xx03	RTN	37

**Description:** Ce programme place l'adresse xxFF dans le registre DP. Cette instruction ne permet pas de lire le contenu exact d'une adresse de la ROM interne (&000-&1FFF). (cf. l'instruction DATA).

## LII n

Cette instruction charge le registre I (registre &00 de la RAM interne) avec la constante n (8 bits).

n -> I

Code HEX: 00                      Code DEC: 00  
Code BIN: 0 0 0 0 0 0 0 0      Indic. :  
Cycles : 4                      Octets : 02

**Remarques:** Cette instruction permet de placer facilement une valeur dans le registre I.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LII FF	00 FF
	xx02	RTN	37

**Description:** Ce programme charge le registre I avec la valeur &FF.

## LIJ n

Cette instruction charge le registre J (registre &01 de la RAM interne) avec la constante n (8 bits).

n -> J

Code HEX: 01                      Code DEC: 01  
Code BIN: 0 0 0 0 0 0 0 1      Indic. :  
Cycles : 4                      Octets : 02

**Remarques:** Cette instruction permet de placer facilement une valeur dans le registre J.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LIJ FF	01 FF
	xx02	RTN	37

**Description:** Ce programme charge le registre J avec la valeur &FF.

## LIP n

Cette instruction charge le registre P avec la constante n (7 bits).

n -> P

Code HEX: 12                      Code DEC: 18  
Code BIN: 0 0 0 1 0 0 1 0      Indic. :  
Cycles : 4                      Octets : 02

**Remarques:** Cette instruction permet de placer des valeurs supérieures à &3F dans le registre P. Nous avons vu précédemment que les instructions LP.. autorisent le chargement de valeur inférieures à &40 .

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LIP 5E	12 5E
	xx02	RTN	37

**Description:** Ce programme charge le registre P avec la valeur &5E.

## LIQ n

Cette instruction charge le registre Q avec la constante n (7 bits).

n -> Q

Code HEX: 13                      Code DEC: 19  
Code BIN: 0 0 0 1 0 0 1 1      Indic. :  
Cycles : 4                      Octets : 02

**Remarques:** Cette instruction permet de placer facilement une valeur dans le registre Q.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LIQ 5E	13 5E
	xx02	RTN	37

**Description:** Ce programme charge le registre Q avec la valeur &5E.

## LOOP n

Cette instruction décrémente le sommet de la pile. L'instruction suivante est ensuite exécutée si l'indicateur de retenue est positionné. Dans le cas contraire, l'instruction exécute un branchement relatif avec un déplacement en arrière de n octets.

L'indicateur de retenue est positionné à 0 si (R) ≠ FF.

Il est positionné à 1 si (R) = FF.

R est incrémenté (R=R+1) pour simuler l'instruction POP, dans le cas où l'indicateur de retenue est positionné à 1.

(R) - 1 -> (R)
Si C=0 alors PC+1-n -> PC
Si C=1 alors PC+2 -> PC

Code HEX: 2F                      Code DEC: 47  
 Code BIN: 0 0 1 0 1 1 1 1      Indic. : CZ  
 Cycles : 10 C=0 / 7 C=1      Octets : 02

Remarques: La pile est utilisée dans les opérations de calcul.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LIA 07	02 07
	xx02	PUSH	34
	xx03	LIB 00	03 00
	xx05	INCB	C2
	xx06	LOOP 02	2F 02
	xx07	RTN	37

Description: L'instruction PUSH place &07 au sommet de la pile. Le programme exécute l'instruction INCB jusqu'à ce que le sommet de la pile soit égal à &FF. C'est-à-dire 8 fois dans notre exemple. Après exécution du programme, le registre B contient la valeur 8.

## LP 1

Cette instruction charge la constante 1 (6 bits &00 - &3F) dans le registre P.

1 -> P
--------

Code HEX: 80 + 1                      Code DEC: 128+1  
 Code BIN: 1 0 < - 1 - >      Indic. :  
 Cycles : 2                      Octets : 01

Remarques: Cette instruction permet de placer une instruction sur 1 octet au lieu de 2. Avec cependant une limite concernant l'argument qui ne peut dépasser &3F.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LP 05	85
	xx01	RTN	37

Description: Ce programme place la constante &05 dans le registre P.

## MVB

Cette instruction écrit le contenu de d+1 octets, à partir de l'adresse pointée par le registre Q, aux adresses situées d+1 octets après l'adresse référencée par le registre P.

J -> d  
Répéter:  
(Q) -> (P), P+1, Q+1, d-1  
Jusqu'à ce que:  
d=FF

Code HEX: 0A                      Code DEC: 10  
Code BIN: 0 0 0 0 1 0 1 0      Indic. :  
Cycles : 5 + 2d                  Octets : 01

Remarques: La valeur du registre J est placée dans le registre d.  
Les registres P et Q sont incrémentés.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LIA 06	02 06
	xx02	LIB 20	03 20
	xx04	LP 04	84
	xx05	LIQ 02	13 02
	xx07	LIJ 01	01 01
	xx09	MVB	0A
	xx0A	RTN	37

Description: Après exécution du programme, le contenu de l'accumulateur (&06) et du registre B (&20) se trouve dans le registre x (&2006).  
Le registre P à la valeur &06 et le registre Q à la valeur &04.

## MVBD

Cette instruction écrit le contenu de d+1 octets, à partir de l'adresse pointée par le registre DP, aux adresses situées d+1 octets après l'adresse référencée par le registre P.

J -> d  
Répéter:  
(DP) -> (P), P+1, DP+1, d-1  
Jusqu'à ce que:  
d=FF

Code HEX: 1A                      Code DEC: 26  
Code BIN: 0 0 0 1 1 0 1 0      Indic. :  
Cycles : 5 + 4d                  Octets : 01

Remarques: La valeur du registre J est placée dans le registre d.  
Les registres P et DP sont incrémentés.  
Cette instruction permet de placer une valeur de la RAM externe dans la RAM interne sans échanger les valeurs (cf. EXWD)

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LP 02	82
	xx01	LIDP xx20	10 xx 20
	xx04	LIJ 01	01 01
	xx06	MVBD	1A
	xx07	RTN	37

Description: Après exécution du programme, le contenu de l'adresse de la RAM externe &xx20 se trouve dans l'accumulateur et celui de l'adresse &xx21 dans le registre B.  
Le registre P à la valeur &06 et le registre DP à la valeur &xx22.



## MVDM

Cette instruction écrit le contenu de l'adresse pointée par le registre P, à l'adresse référencée par le registre DP.

(P) -> (DP)

Code HEX: 53                      Code DEC: 83  
Code BIN: 0 1 0 1 0 0 1 1      Indic. :  
Cycles : 3                      Octets : 01

Remarques: Cette instruction permet de placer une valeur de la RAM interne dans la RAM externe.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LIDP xx20	10 xx 20
	xx03	LP 08	88
	xx04	LIA FF	02 FF
	xx06	EXAM	DB
	xx07	MVDM	53
	xx08	RTN	37

Description: Après exécution du programme, le contenu du registre K (&FF) se trouve à l'adresse &xx20 de la RAM externe.

## MVMD

Cette instruction écrit le contenu de l'adresse pointée par le registre DP, à l'adresse référencée par le registre P.

(DP) -> (P)

Code HEX: 55                      Code DEC: 85  
Code BIN: 0 1 0 1 0 1 0 1      Indic. :  
Cycles : 3                      Octets : 01

Remarques: Cette instruction permet de placer une valeur de la RAM externe dans la RAM interne.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LIDP xx20	10 xx 20
	xx03	LP 08	88
	xx04	MVMD	55
	xx05	RTN	37

Description: Après exécution du programme, le registre K (&FF) renferme le contenu de l'adresse &xx20 de la RAM externe.

## MVW

Cette instruction écrit le contenu de d+1 octets, à partir de l'adresse pointée par le registre Q, aux adresses situées d+1 octets après l'adresse référencée par le registre P.

I -> d  
Répéter:  
(Q) -> (P), P+1, Q+1, d-1  
Jusqu'à ce que:  
d=FF

Code HEX: 08                      Code DEC: 08  
Code BIN: 0 0 0 0 1 0 0 0      Indic. :  
Cycles : 5 + 2d                  Octets : 01

Remarques: La valeur du registre I est placée dans le registre d.  
Les registres P et Q sont incrémentés.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LIA 06	02 06
	xx02	LIB 20	03 20
	xx04	LP 04	84
	xx05	LIQ 02	13 02
	xx07	LII 01	00 01
	xx09	MVW	08
	xx0A	RTN	37

Description: Après exécution du programme, le contenu de l'accumulateur (&06) et du registre B (&20) se trouve dans le registre x (&2006). Le registre P à la valeur &06 et le registre Q à la valeur &04.

## MVWD

Cette instruction écrit le contenu de d+1 octets, à partir de l'adresse pointée par le registre DP, aux adresses situées d+1 octets après l'adresse référencée par le registre P.

I -> d  
Répéter:  
(DP) -> (P), P+1, DP+1, d-1  
Jusqu'à ce que:  
d=FF

Code HEX: 18                      Code DEC: 24  
Code BIN: 0 0 0 1 1 0 0 0      Indic. :  
Cycles : 5 + 4d                  Octets : 01

Remarques: La valeur du registre I est placée dans le registre d.  
Les registres P et DP sont incrémentés.  
Cette instruction permet de placer une valeur de la RAM externe dans la RAM interne sans échanger les valeurs (cf. EXWD)

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LP 02	82
	xx01	LIDP xx20	10 xx 20
	xx04	LII 01	00 01
	xx06	MVWD	18
	xx07	RTN	37

Description: Après exécution du programme, le contenu de l'adresse de la RAM externe &xx20 se trouve dans l'accumulateur et celui de l'adresse &xx21 dans le registre B.  
Le registre P à la valeur &06 et le registre DP la valeur &xx22.

## NOPT

Cette instruction ne fait rien pendant 3 cycles.

NOPT

Code HEX: CE                      Code DEC: 206  
Code BIN: 1 1 0 0 1 1 1 0      Indic. :  
Cycles : 3                      Octets : 01

**Remarques:** Cette instruction est très utile dans les applications où la durée et la vitesse d'exécution jouent un rôle important.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LIA 02	02 02
	xx02	NOPT	CE
	xx03	RTN	37

**Description:** A la suite du chargement de la constante &02 dans l'accumulateur, le programme ne fait rien pendant 3 cycles.

## NOPW

Cette instruction ne fait rien pendant 2 cycles.

NOPW

Code HEX: 4D                      Code DEC: 77  
Code BIN: 0 1 0 0 1 1 0 1      Indic. :  
Cycles : 2                      Octets : 01

**Remarques:** Cette instruction est très utile dans les applications où la durée et la vitesse d'exécution jouent un rôle important.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LIA 02	02 02
	xx02	NOPW	4D
	xx03	RTN	37

**Description:** A la suite du chargement de la constante &02 dans l'accumulateur, le programme ne fait rien pendant 2 cycles.

## ORIA n

Cette instruction effectue un OU logique entre l'accumulateur et la constante n (8 bits).

A v n -> A Z

Code HEX: 65                      Code DEC: 101  
Code BIN: 0 1 1 0 0 1 0 1      Indic. : Z  
Cycles : 4                      Octets : 02

Remarques: Comme toutes les opérations logiques, cette instruction peut positionner l'indicateur de nullité.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LIA 2F	02 2F
	xx02	ORIA 12	65 12
	xx04	RTN	37

Description: Ce programme effectue un Ou logique entre la valeur de l'accumulateur (&2F) et la constante &12. Le résultat &3F est placé dans l'accumulateur.

## ORID n

Cette instruction effectue un OU logique entre le contenu de l'adresse mémoire référencée par le registre DP et la constante n (8 bits). Le résultat est placé dans l'octet mémoire pointé par le registre DP.

(DP) v n -> (DP) Z

Code HEX: D5                      Code DEC: 213  
Code BIN: 1 1 0 1 0 1 0 1      Indic. : Z  
Cycles : 6                      Octets : 02

Remarques: Comme toutes les opérations logiques, cette instruction peut positionner l'indicateur de nullité.  
R-1 est utilisé.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LIDP xx50	10 xx 50
	xx03	LIA 15	02 15
	xx05	STD	52
	xx06	ORID 63	D5 63
	xx08	RTN	37

Description: Ce programme effectue un OU logique entre les constantes &15 et &63. Pour débiter, la valeur &15 est placée à l'adresse &xx50. Un OU logique est ensuite effectué avec la constante &63. Le résultat &77 est rangé à l'adresse &xx50 et peut être lu par l'instruction Basic PEEK.

## ORIM n

Cette instruction effectue un OU logique entre le contenu de l'adresse mémoire référencée par le registre P et la constante n (8 bits). Le résultat est placé dans l'octet mémoire pointé par le registre P.

(P) v n -> (P) Z

Code HEX: 61                      Code DEC: 97  
Code BIN: 0 1 1 0 0 0 0 1      Indic. : Z  
Cycles : 4                      Octets : 02

**Remarques:** Comme toutes les opérations logiques, cette instruction peut positionner l'indicateur de nullité.

Exemple:	Adresse	Assembleur	Code HEX
xx00	LP	0A	8A
xx01	LIA	15	02 15
xx03	EXAM		DB
xx04	ORIM	63	61 63
xx06	RTN		37

**Description:** Ce programme effectue un OU logique entre les constantes &15 et &63. Pour débiter, la valeur &15 est placée à l'adresse &0A (registre M). Un OU logique est ensuite effectué avec la constante &63. Le résultat &77 est rangé à l'adresse &0A (registre M).

## ORMA

Cette instruction effectue un OU logique entre le contenu de l'adresse référencée par le registre P et l'accumulateur. Le résultat est placé à l'adresse pointée par le registre P.

(P) v A -> (P) Z

Code HEX: 47                      Code DEC: 71  
Code BIN: 0 1 0 0 0 1 1 1      Indic. : Z  
Cycles : 3                      Octets : 01

**Remarques:** Comme toutes les opérations logiques, cette instruction peut positionner l'indicateur de nullité.

Exemple:	Adresse	Assembleur	Code HEX
xx00	LP	0A	8A
xx01	LIA	15	02 15
xx03	EXAM		DB
xx04	LIA	63	02 63
xx06	ORMA		47
xx07	RTN		37

**Description:** Ce programme effectue un OU logique entre les constantes &15 et &63. Pour débiter, la valeur &15 est placée à l'adresse &0A (registre M) de la RAM interne. Un OU logique est ensuite effectué avec la constante &63 contenue dans l'accumulateur. Le résultat &77 est rangé à l'adresse &0A (registre M).

## OUTA

Cette instruction sort l'octet d'adresse &5C de la RAM interne par le port IA.

(5C) -> port IA

Code HEX: 5D                      Code DEC: 93  
Code BIN: 0 1 0 1 1 1 0 1      Indic. :  
Cycles : 3                      Octets : 01

Remarques: Chaque bit est associé à une ligne. Le diagramme donné en annexe permet de déterminer la spécificité de chaque ligne. La figure ci-dessous montre les relations ligne/octet:

8	7	6	5	4	3	2	1	bit (5C)
↓	↓	↓	↓	↓	↓	↓	↓	
IA8	IA7	IA6	IA5	IA4	IA3	IA2	IA1	

Exemple: cf. l'instruction INA !

## OUTB

Cette instruction sort l'octet d'adresse &5D de la RAM interne par le port IB.

(5D) -> port IB

Code HEX: DD                      Code DEC: 221  
Code BIN: 1 1 0 1 1 1 0 1      Indic. :  
Cycles : 2                      Octets : 01

Remarques: Chaque bit est associé à une ligne. Le diagramme donné en annexe permet de déterminer la spécificité de chaque ligne. La figure ci-dessous montre les relations ligne/octet:

8	7	6	5	4	3	2	1	bit (5D)
↓	↓	↓	↓	↓	↓	↓	↓	
IB8	IB7	IB6	IB5	IB4	IB3	IB2	IB1	

Exemple: cf. l'instruction INB !

## OUTC

Cette instruction sort l'octet d'adresse &5F de la RAM interne par le port de contrôle.

(5F) -> port de contrôle

Code HEX: DF                      Code DEC: 223  
Code BIN: 1 1 0 1 1 1 1 1      Indic. :  
Cycles : 2                      Octets : 01

Remarques: Il est utile de conserver la valeur de chaque bit lorsque l'octet 1 est modifié. La figure ci-dessous montre l'agencement de l'octet &5F:

8	7	6	5	4	3	2	1	bit (5F)
↓	↓	↓	↓	↓	↓	↓	↓	
NC	BZ3	BZ2	BZ1	OFF	HLT	CL	DIS	

Ces abréviations correspondent aux termes suivants:

BZ1-BZ3 sont les bits de contrôle de XOUT et XIN. Leur combinaison à la valeur suivante:

BZ3	BZ2	BZ1	XOUT	XIN
0	0	0	LOW	inactif
0	0	1	HIGH	inactif
0	1	0	2 KHz	inactif
0	1	1	4 KHz	inactif
1	0	0	LOW	actif
1	0	1	HIGH	actif
1	0	X	XIN->XOUT	actif

OFF: hors tension  
HLT: arrêt horloge  
CL: compteur à 0  
DIS: 1=LCD on, 0=LCD off

Exemple:

Adresse	Assembleur	Code HEX
xx00	LIP 5F	12 5F
xx02	LIA 01	02 01
xx04	ORMA	47
xx05	OUTC	DF
xx06	RTN	37

Description: Ce programme initialise le mode graphique. (Indispensable à la gestion par points des ordinateurs de poche 14xx, 12xx).

## OUTF

Cette instruction sort l'octet d'adresse &5E de la RAM interne par le port F0.

(5E) -> port F0

Code HEX: 5F                      Code DEC: 95  
Code BIN: 0 1 0 1 1 1 1 1      Indic. :  
Cycles : 3                      Octets : 01

**Remarques:** Les 5 derniers bits sont associés à une ligne. Le diagramme donné en annexe permet de déterminer la spécificité de chaque ligne. La figure ci-dessous montre les relations ligne/octet:

8	7	6	5	4	3	2	1	bit (5E)
↓	↓	↓	↓	↓	↓	↓	↓	
NC	NC	NC	F05	F04	F03	F02	F01	

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LIP 5E	12 5E
	xx02	LIA 04	02 04
	xx04	ORMA	47
	xx05	OUTF	5F
	xx06	RTN	37

**Description:** Ce programme émet un signal HIGH sur la ligne F03. Pour ce faire, il effectue un OU logique entre l'adresse &5E de la RAM interne et la constante &04. Le résultat est ensuite envoyé au port F0.

## POP

Cette instruction charge l'accumulateur avec l'octet référencé par le pointeur de pile. ce dernier est ensuite incrémenté.

(R) -> A  
R+1 -> R

Code HEX: 5B                      Code DEC: 91  
Code BIN: 0 1 0 1 1 0 1 1      Indic. :  
Cycles : 2                      Octets : 01

**Remarques:** Le registre R est incrémenté d'une unité.

**Exemple:** Cf. l'instruction PUSH.



## PTJ

L'instruction PTJ (Prepare Table Jump) définit une table des sauts comportant le nombre de branchements ainsi que l'adresse de retour. Celle-ci est empilée dans la pile système.

l = nombre de branchements  
n = octet de poids fort de l'adresse de retour  
m = octet de poids faible de l'adresse de retour

n m	-> (R-1, R-2)
(R-2)	-> R

Code HEX: 7A                      Code DEC: 122  
Code BIN: 0 1 1 1 1 0 1 0      Indic. :  
Cycles : 9                      Octets : 04

Remarques: Cette instruction ne fait pas partie du jeu d'instructions standard communiqué par Sharp.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	PTJ 03	7A 03
	xx02	- xx12	xx 12
	xx04	LIA 01	02 01
	xx06	DTJ	69
	xx07	-	01 xx 13
	xx0A	-	02 xx 16
	xx0D	-	03 xx 19
	xx10	-	xx 1C
	xx12	RTN	37
	xx13	LIB 50	03 50
	xx15	RTN	37
	xx16	LIB 80	03 80
	xx18	RTN	37
	xx19	LIB A0	03 A0
	xx1B	RTN	37
	xx1C	LIB FF	03 FF
	xx1E	RTN	37

Description: En fonction du contenu de l'accumulateur, diverses constantes seront placées dans le registre B. Ces valeurs sont:

A=01 => B=50  
A=02 => B=80  
A=03 => B=A0  
défaut => B=FF

Le contenu du registre A est chargé à l'adresse xx04. Le branchement au sous-programme s'effectue en fonction du contenu de l'accumulateur. Cf. l'instruction DTJ.

## PUSH

Cette instruction empile le contenu de l'accumulateur.

R - 1 -> R  
A -> (R)

Code HEX: 34                      Code DEC: 52  
Code BIN: 0 0 1 1 0 1 0 0      Indic. :  
Cycles : 3                      Octets : 01

**Remarques:** Le registre R est décrémenté d'une unité.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LIA 03	02 03
	xx02	PUSH	34
	xx03	LIA 00	02 00
	xx05	POP	5B
	xx06	RTN	37

**Description:** L'instruction PUSH place le contenu de l'accumulateur (&03) dans la pile. POP replace cette constante dans l'accumulateur.

## RA

Cette instruction met à zéro le contenu de l'accumulateur.

0 -> A

Code HEX: 23                      Code DEC: 35  
Code BIN: 0 0 1 0 0 0 1 1      Indic. :  
Cycles : 2                      Octets : 01

**Remarques:** Il s'agit là de la façon la plus rapide de rafraichir le contenu de l'accumulateur.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LIA 03	02 03
	xx02	RA	23
	xx03	RTN	37

**Description:** Bien que la constante &03 ait été chargée dans l'accumulateur, après exécution du programme le contenu de l'accumulateur est nul.

## RC

Cette instruction met à zéro l'indicateur de retenue et positionne l'indicateur de nullité

1 -> Z
0 -> C

Code HEX: D1                      Code DEC: 209  
Code BIN: 1 1 0 1 0 0 0 1      Indic. : CZ  
Cycles : 2                      Octets : 01

Exemple:	Adresse	Assembleur	Code HEX
	xx00	SC	D0
	xx01	RC	D1
	xx02	RTN	37

**Description:** Bien que l'instruction SC positionne les deux indicateurs, après exécution du programme l'indicateur de retenue est à zéro.

## READ

Cette instruction charge l'accumulateur avec le contenu de l'adresse mémoire référencée par le registre PC+1.

(PC+1) -> A
-------------

Code HEX: 56                      Code DEC: 86  
Code BIN: 0 1 0 1 0 1 1 0      Indic. :  
Cycles : 3                      Octets : 01

**Remarques:** Cette instruction ne fait pas partie du jeu d'instructions standard communiqué par Sharp.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	READ	56
	xx01	LIB FE	03 FE
	xx03	RTN	37

**Description:** Après exécution du programme, l'accumulateur contient la valeur &03 (code hex de l'instruction LIB).

## READM

Cette instruction range le contenu de l'adresse mémoire référencée par le registre PC+1, à l'adresse de la RAM interne pointée par le registre P.

(PC+1) -> (P)

Code HEX: 54                      Code DEC: 84  
Code BIN: 0 1 0 1 0 1 0 0      Indic. :  
Cycles : 3                      Octets : 01

**Remarques:** Cette instruction ne fait pas partie du jeu d'instructions standard communiqué par Sharp.

**Exemple:**

<u>Adresse</u>	<u>Assembleur</u>	<u>Code HEX</u>
xx00	LP 08	88
xx01	READM	54
xx02	LIB FE	03 IE
xx04	RTN	37

**Description:** Après exécution du programme, la valeur &03 (code hex de l'instruction LIB) est dans le registre K.

## RTN

Cette instruction termine un programme ou un sous-programme écrit en langage machine.

(R) -> PCL  
(R+1) -> PCH  
R+2 -> R

Code HEX: 37                      Code DEC: 55  
Code BIN: 0 0 1 1 0 1 1 1      Indic. :  
Cycles : 4                      Octets : 01

**Remarque:** Il n'est pas nécessaire d'inclure un exemple.

## SBB

Cette instruction effectue une soustraction sur 16 bits. Le contenu de l'adresse mémoire référencée par le registre P est soustrait de la paire d'accumulateur AB. Le résultat est placé à l'adresse pointée par le registre P. (P+1) complémente le registre P pour former 16 bits.

(P+1,P)-(BA) -> (P+1,P) C,Z P+1 → P

Code HEX: 15                      Code DEC: 21  
Code BIN: 0 0 0 1 0 1 0 1      Indic. : C Z  
Cycles : 5                      Octets : 01

**Remarques:** Comme toute opération arithmétique, cette instruction positionne l'indicateur de nullité en cas de résultat égal à zéro et l'indicateur de retenue en cas de débordement. Notez que le registre P est incrémenté de une unité.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LP 08	88
	xx01	LIA 30	02 30
	xx03	EXAM	DB
	xx04	LP 09	89
	xx05	LIA 21	02 21
	xx07	EXAM	DB
	xx08	LIB 10	03 10
	xx0A	LIA 40	02 40
	xx0C	LP 08	88
	xx0D	SBB	15
	xx0E	RTN	37

**Description:** Ce programme soustrait &1040 de &2130. Pour pouvoir être soustrait, ces nombres sont d'abord chargés dans les registres de la CPU.

La constante &08 est placée dans le registre P car le nombre &2130 doit être rangé dans les

registres K et L. L'octet de poids faible (&30) est ensuite placé dans l'accumulateur et transféré dans le registre K à l'aide de l'instruction EXAM. Le même procédé est appliqué au registre L, d'adresse &09, qui lui, reçoit l'octet de poids fort (&21).

Le nombre &1040 est réparti entre les deux accumulateurs. Le résultat est placé dans les registres K et L.

Après exécution de l'instruction SBB, le contenu du registre P est égal à 9.

Pour pouvoir lire le résultat, il suffit de transférer le contenu des registres K et L dans la RAM externe. Le résultat peut alors être lu à l'aide de l'instruction Basic PEEK.

## SBCM

Cette instruction soustrait le contenu de l'accumulateur et l'indicateur de retenue du contenu de l'adresse référencée par le registre P. Le résultat est placé à l'adresse pointée par le registre P.

(P)-A-C -> (P) C,Z

Code HEX: C5                      Code DEC: 197  
Code BIN: 1 1 0 0 0 1 0 1      Indic. : C Z  
Cycles : 3                      Octets : 01

Remarques: Comme toute opération arithmétique, cette instruction positionne l'indicateur de nullité en cas de résultat égal à zéro et l'indicateur de retenue en cas de débordement.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LP 08	88
	xx01	LIA 30	02 30
	xx03	EXAM	DB
	xx04	LIA 04	02 04
	xx06	SC	D0
	xx07	SBCM	C5
	xx08	RTN	37

Description: Ce programme soustrait &04 du nombre &30 et positionne l'indicateur de retenue. La constante &30 est placée dans le registre K (&08). &04 est chargé dans l'accumulateur. L'instruction SC positionne l'indicateur de retenue. Le résultat &2B est placé dans le registre K.

## SBIA n

Cette instruction soustrait la constante n du contenu de l'accumulateur

A-n -> A C,Z

Code HEX: 75                      Code DEC: 117  
Code BIN: 0 1 1 1 0 1 0 1      Indic. : C Z  
Cycles : 4                      Octets : 02

Remarques: Comme toute opération arithmétique, cette instruction positionne l'indicateur de nullité en cas de résultat égal à zéro et l'indicateur de retenue en cas de débordement.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LIA 2F	02 2F
	xx02	SBIA 12	75 12
	xx04	RTN	37

Description: Ce programme soustrait &12 du contenu de l'accumulateur (&2F). Le résultat &1D est placé dans l'accumulateur;

## SBIM n

Cette instruction soustrait la constante n du contenu de l'adresse référencée par le registre P.

(P)-n -> (P) C,Z

Code HEX: 71                      Code DEC: 113  
Code BIN: 0 1 1 1 0 0 0 1      Indic. : C Z  
Cycles : 4                      Octets : 02

Remarques: Comme toute opération arithmétique, cette instruction positionne l'indicateur de nullité en cas de résultat égal à zéro et l'indicateur de retenue en cas de débordement.

Exemple:

Adresse	Assembleur	Code HEX
xx00	LIA 70	02 70
xx02	LP 08	88
xx03	EXAM	DB
xx04	SBIM 0F	71 0F
xx06	RTN	37

Description: Ce programme soustrait la constante &0F de &70. La valeur &70 est chargée dans l'accumulateur et transférée dans le registre K à l'aide de l'instruction EXAM. Le résultat &61 est placé dans le registre K.

## SBM

Cette instruction soustrait l'accumulateur du contenu de l'adresse de la RAM interne référencée par le registre P.

(P)-A -> (P) C,Z

Code HEX: 45                      Code DEC: 69  
Code BIN: 0 1 0 0 1 0 0 1      Indic. : C Z  
Cycles : 3                      Octets : 01

Remarques: Comme toute opération arithmétique, cette instruction positionne l'indicateur de nullité en cas de résultat égal à zéro et l'indicateur de retenue en cas de débordement.

Exemple:

Adresse	Assembleur	Code HEX
xx00	LIA 30	02 30
xx02	LP 08	88
xx03	EXAM	DB
xx04	LIA 04	02 04
xx06	SBM	45
xx07	RTN	37

Description: Ce programme soustrait la valeur &04 de &30. Le registre K est utilisé. La valeur &30 est chargée dans l'accumulateur et transférée dans le registre K à l'aide de l'instruction EXAM. Le résultat &2C est placé dans le registre K.

## SBN

Cette instruction soustrait d+1 fois le contenu de l'accumulateur à partir de l'adresse référencée par le registre P. Au début de l'instruction, le registre P pointe l'adresse la plus haute.

I -> d  
 Répéter:  
 (P)-A -> (P) (DCB),P-1,d-1 C,Z  
 jusqu'à ce que:  
 d=FF

Code HEX: 0D                      Code DEC: 13 13  
 Code BIN: 0 0 0 0 1 1 0 1      Indic. : CZ  
 Cycles : 7+3d                    Octets : 01

Remarques: L'indicateur de nullité Z est mis à 1 si le résultat est égal à 0. L'indicateur de retenue C est mis à 1 si l'opération produit une retenue.

Le registre I contient la valeur d.  
 Les registres D et P sont décrémentés.  
 Le nombre de cycles dépend bien évidemment du nombre d'itérations de la boucle.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LP 08	88
	xx01	LIA 10	02 10
	xx03	EXAM	DB
	xx04	LP 09	89
	xx05	LIA 99	02 99
	xx07	EXAM	DB
	xx08	LP 0A	8A
	xx09	LIA 99	02 99
	xx0B	EXAM	DB
	xx0C	LP 0B	8B
	xx0D	LIA 99	02 99
	xx0F	EXAM	DB
	xx10	LII 03	00 03

xx12	LIA 14	02 14
xx14	SBN	0D
xx15	RTN	37

Description: Ce programme soustrait le nombre 14 de 10999999 sous forme DCB. le nombre 10999999 est placé aux adresses mémoire 08-0B (xx00-xx0F). Le nombre 14 est stocké dans l'accumulateur. Le registre I est chargé avec la valeur 3 (d-1=3) car les adresses 08-0B correspondent à des registres sur 4 bits. Au sein de l'instruction SBN, le contenu du registre I est transféré dans le registre d. le résultat 10999985 est placé dans les registres 08-0B. Chaque registre contient la valeur suivante:

K (08) = 10  
 L (09) = 99  
 M (0A) = 99  
 N (0B) = 85  
 P = 07



## SBW

Cette instruction effectue également une soustraction sous forme DCB. Elle soustrait le contenu de d+1 octets, à partir de l'adresse référencée par le registre Q, aux d+1 octets, à partir de l'adresse pointée par le registre P. Au départ, les registres P et Q contiennent les adresses les plus hautes. Le résultat est placé aux adresses référencées par le registre P.

I -> d  
 Répéter:  
 (P)-(Q) -> (P) (DCB),P-1,Q-1,d-1 C,Z  
 Jusqu'à ce que:  
 d=FF

Code HEX: 0F                      Code DEC: 15  
 Code BIN: 0 0 0 0 1 1 1 1      Indic. : CZ  
 Cycles : 7+3d                    Octets : 01

Remarques: L'indicateur de nullité Z est mis à 1 si le résultat est égal à 0. L'indicateur de retenue C est mis à 1 si l'opération produit une retenue.

Le registre I contient la valeur d.

Les registres d, P et Q sont décrémentés.

Le nombre de cycles dépend bien évidemment du nombre d'itérations de la boucle.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LP 08	88
	xx01	LIA 10	02 10
	xx03	EXAM	DB
	xx04	LP 09	89
	xx05	LIA 99	02 99
	xx07	EXAM	DB
	xx08	LP 0A	8A
	xx09	LIA 99	02 99
	xx0B	EXAM	DB
	xx0C	LP 0B	8B

xx0D	LIA 99	02 99
xx0F	EXAM	DB
xx10	LII 01	00 01
xx12	LIQ 09	13 09
xx14	SBW	OF
xx15	RTN	37

**Description:** Ce programme soustrait 9999 de 1099 sous forme DCB. Le nombre 1099 est stocké dans les registres K et L tandis que la valeur 9999 est rangée dans les registres M et N. Pour que l'opération de soustraction puisse s'effectuer, il faut que le registre P référence l'adresse 0B et que le registre Q pointe l'adresse 09. L'opération de soustraction positionne l'indicateur de retenue à 1 car le résultat (1099-9999=-8900) ne peut être représenté sur deux octets. Chaque registre contient la valeur suivante:

K (08) = 10  
 L (09) = 99  
 M (0A) = 89  
 N (0B) = 00

P = 09  
 Q = 06

## SC

Cette instruction positionne l'indicateur de nullité et de retenue.

1 -> C
1 -> Z

Code HEX: D0	Code DEC: 208
Code BIN: 1 1 0 1 0 0 0 0	Indic. : CZ
Cycles : 2	Octets : 01

Exemple: cf. l'instruction RC !

## SL

Cette instruction effectue une rotation arithmétique à gauche de 1 bit de l'accumulateur. Le bit de gauche est mis dans l'indicateur de retenue et l'indicateur de retenue devient le bit de droite.

A8 <----- A1	<- C
--------------	------

Code HEX: 5A	Code DEC: 90
Code BIN: 0 1 0 1 1 0 1 0	Indic. : CZ
Cycles : 2	Octets : 01

Remarques: Le résultat de cette opération correspond à une multiplication par 2.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LIA OD	02 0D
	xx02	SL	5A
	xx03	RTN	37

Description: L'instruction SL permet de multiplier le contenu de l'accumulateur (&0D=13) par deux. Après exécution du programme, le contenu de l'accumulateur est égal à &1A (26).

## SLW

Cette instruction effectue d+1 rotations arithmétiques à gauche de 4 bits (1 position DCB) à partir de l'adresse de la RAM interne pointée par le registre P.

I -> d  
Répéter:  
rotation (P) de 4 bits à gauche,  
P-1, d-1  
Jusqu'à ce que:  
d=FF

Code HEX: 1D                      Code DEC: 29  
Code BIN: 0 0 0 1 1 1 0 1      Indic. :  
Cycles : 5+d                      Octets : 01

Remarques: Cette instruction décrémente le registre P.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LP 08	88
	xx01	LIA 12	02 12
	xx03	EXAM	DB
	xx04	LP 09	89
	xx05	LIA 34	02 34
	xx07	EXAM	DB
	xx08	LP 0A	8A
	xx09	LIA 56	02 56
	xx0B	EXAM	DB
	xx0C	LP 0B	8B
	xx0D	LIA 78	02 78
	xx0F	EXAM	DB
	xx10	LII 03	00 03
	xx12	SLW	1D
	xx13	RTN	37

Description: Le nombre DCB 12345678 est placé dans les registres K à N. Chaque registre prend la valeur suivante:

K=12; L=34; M=56; N=78

La constante 3 est placée dans le registre I (taille du bloc: 4 octets-1). L'instruction SLW effectue une rotation à gauche. La position à droite prend la valeur 0. Après exécution du programme, les registres ont la valeur suivante:

K=23; L=45; M=67; N=80

P=07

## SR

Cette instruction effectue une rotation arithmétique à droite de 1 bit de l'accumulateur. Le bit de droite est mis dans l'indicateur de retenue et l'indicateur de retenue devient le bit de gauche.

A8 -----> A1 -> C

Code HEX: D2                      Code DEC: 210  
Code BIN: 1 1 0 1 0 0 1 0      Indic. : C  
Cycles : 2                      Octets : 01

Remarques: Le résultat de cette opération correspond à une division par 2.

Exemple:

Adresse	Assembleur	Code HEX
xx00	LIA D0	02 D0
xx03	SR	D2
xx04	RTN	37

Description: L'instruction Sk permet de diviser le contenu de l'accumulateur (&D0=208) par deux. Après exécution du programme, le contenu de l'accumulateur est égal à &68 (104).

## SRW

Cette instruction effectue d+1 rotations arithmétiques à droite de 4 bits (1 position DCB) à partir de l'adresse de la RAM interne pointée par le registre P.

I -> d  
Répéter:  
rotation (P) de 4 bits à droite,  
P+1, d-1  
Jusqu'à ce que:  
d=FF

Code HEX: 1C                      Code DEC: 28  
Code BIN: 0 0 0 1 1 1 0 0      Indic. :  
Cycles : 5+d                      Octets : 01

Remarques: Cette instruction décrémente le registre P.

Exemple:

Adresse	Assembleur	Code HEX
xx00	LP 08	88
xx01	LIA 12	02 12
xx03	EXAM	DB
xx04	LP 09	89
xx05	LIA 34	02 34
xx07	EXAM	DB
xx08	LP 0A	8A
xx09	LIA 56	02 56
xx0B	EXAM	DB
xx0C	LP 0B	8B
xx0D	LIA 78	02 78
xx0F	EXAM	DB
xx10	LII 03	00 03
xx12	LP 08	88
xx13	SRW	1C
xx14	RTN	37

Description: Le nombre DCB 12345678 est placé dans les registres K à N. Chaque registre prend

la valeur suivante:

K=12; L=34; M=56; N=78

La constante 3 est placée dans le registre I (taille du bloc: 4 octets-1). L'instruction SLW effectue une rotation à droite. La position à gauche prend la valeur 0. Après exécution du programme, les registres ont la valeur suivante:

K=01; L=23; M=45; N=67

P=0C

## STD

Cette instruction range le contenu de l'accumulateur à l'adresse pointée par le registre DP.

A -> (DP)

Code HEX: 52                      Code DEC: 82  
Code BIN: 0 1 0 1 0 0 1 0      Indic. :  
Cycles : 2                      Octets : 01

**Remarques:** Cette instruction permet de ranger une valeur dans la RAM externe.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LIDP xx02	10 xx 02
	xx03	LIA F0	02 F0
	xx05	STD	52
	xx06	RTN	37

**Description:** Le contenu de l'accumulateur est rangé à l'adresse xx02.

## STP

Cette instruction range le contenu de l'accumulateur dans le registre P.

A -> P

Code HEX: 30                      Code DEC: 48  
Code BIN: 0 0 1 1 0 0 0 0      Indic. :  
Cycles : 2                      Octets : 01

Remarques: Le registre P est un registre sur 7 bits.

Exemple:

Adresse	Assembleur	Code HEX
xx00	LIA 08	02 08
xx02	STP	30
xx03	RTN	37

Description: La constante &08 est placée dans le registre P.

## STQ

Cette instruction range le contenu de l'accumulateur dans le registre Q.

A -> Q

Code HEX: 31                      Code DEC: 49  
Code BIN: 0 0 1 1 0 0 0 1      Indic. :  
Cycles : 2                      Octets : 01

Remarques: Le registre Q est un registre sur 7 bits.

Exemple:

Adresse	Assembleur	Code HEX
xx00	LIA 08	02 08
xx02	STQ	31
xx03	RTN	37

Description: La constante &08 est placée dans le registre Q.

## STR

Cette instruction range le contenu de l'accumulateur dans le registre R.

A -> R

Code HEX: 32                      Code DEC: 50  
Code BIN: 0 0 1 1 0 0 1 0      Indic. :  
Cycles : 2                      Octets : 01

Remarques: Le registre R est un registre sur 7 bits.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LIA 50	02 50
	xx02	PUSH	34
	xx03	STR	32
	xx04	RTN	37

Description: La constante &50 est placée dans le registre R. Même après l'instruction PUSH.

## SWP

Cette instruction échange les 4 bits de droite de l'accumulateur avec les 4 bits de gauche.

A8 - A5 <-> A4 - A1

Code HEX: 58                      Code DEC: 88  
Code BIN: 0 1 0 1 1 0 0 0      Indic. :  
Cycles : 2                      Octets : 01

Remarques: Cette instruction est utilisée pour les opérations en DCB.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LIA 50	02 50
	xx02	SWP	58
	xx03	RTN	37

Description: Après exécution du programme, le contenu de l'accumulateur est égal à &05.

## TEST n

Cette instruction teste certains signaux de la CPU. Le résultat est placé dans l'indicateur de nullité.

n -> TEST, Z

Code HEX: 6B                      Code DEC: 107  
Code BIN: 0 1 1 0 1 0 1 1      Indic. : Z  
Cycles : 4                      Octets : 02

**Remarques:** L'argument ne peut prendre que certaines valeurs bien spécifiques car chaque bit de l'argument est associé à une fonction déterminée. En réalité, cette instruction effectue un ET et place le résultat dans l'indicateur de nullité. Les bits sont associés de la manière suivante:

8   7   6   5   4   3   2   1  
XIN RST NC NC KON NC CT2 CT1

XIN: si XIN = 1 => Z=0  
      si XIN = 0 => Z=1

RST: test reset

KON: si KON = 1 => Z=0  
      si KON = 0 => Z=1

Sur PC-1350, KON est associé à la touche BREAK. Ainsi, TEST 08 permet de tester uniquement l'état de la touche BREAK.

CT2: 2 msec countertest  
CT1: 512 msec countertest

**Exemple:**

Adresse	Assembleur	Code HEX
xx00	TEST 08	6B 08
xx02	JRZM 03	39 03

xx03

RTN

37

**Description:** Ce programme attend l'appui de la touche BREAK.



## TSIA n

Cette instruction effectue un ET entre la constante n (8 bits) et l'accumulateur. Le résultat est placé dans l'indicateur de nullité.

$A^n \rightarrow Z$

Code HEX: 66                      Code DEC:102  
Code BIN: 0 1 1 0 0 1 1 0      Indic. : Z  
Cycles : 4                      Octets : 02

Remarques: Le contenu de l'accumulateur n'est pas modifié.

Exemple:

Adresse	Assembleur	Code HEX
xx00	LIA F0	02 F0
xx02	TSIA 0F	66 0F
xx04	RTN	37

Description: Ce programme effectue un ET entre le contenu de l'accumulateur (&F0) et la constante &0F. Comme le résultat est nul, l'indicateur de nullité est positionné.

## TSID n

Cette instruction effectue un ET entre le contenu de l'adresse référencée par le registre DP et la constante n (8 bits). Le résultat est placé dans l'indicateur de nullité.

$(DP) \wedge n \rightarrow Z$

Code HEX: D6                      Code DEC:214  
Code BIN: 1 1 0 1 0 1 1 0      Indic. : Z  
Cycles : 6                      Octets : 02

Remarques: Le contenu de l'adresse référencée par DP n'est pas modifié. R-1 est utilisé.

Exemple:

Adresse	Assembleur	Code HEX
xx00	LIDP xx50	10 xx 50
xx03	LIA 0F	02 0F
xx05	STD	52
xx06	TSID F0	D6 F0
xx08	RTN	37

Description: Les "xx" indiquent une adresse de la RAM externe qui servira à la comparaison. Le programme effectue un ET entre les valeurs &0F et &F0. La valeur &0F est placée à l'adresse xx50 et comparée (ET). Le résultat positionne ou non l'indicateur de nullité.

## TSIM n

Cette instruction effectue un ET entre le contenu de l'adresse référencée par le registre P et la constante n (8 bits). Le résultat est placé dans l'indicateur de nullité.

$(P) \wedge n \rightarrow Z$

Code HEX: 62                      Code DEC: 98  
Code BIN: 0 1 1 0 0 0 1 0      Indic. : Z  
Cycles : 4                      Octets : 02

Remarques: Le contenu de l'adresse référencée par P n'est pas modifié.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LP 0A	8A
	xx01	LIA 0F	02 0F
	xx03	EXAM	DB
	xx04	TSIM F0	62 F0
	xx06	RTN	37

Description: Ce programme effectue un ET entre les constantes &0F et &F0. La valeur &0F est placée à l'adresse du registre M (&0A) et comparée (ET). Le résultat positionne ou non l'indicateur de nullité.

## TSMA (TSIP)

Cette instruction effectue un ET entre le contenu de l'adresse référencée par le registre P et l'accumulateur. Le résultat est placé dans l'indicateur de nullité.

$(P) \wedge A \rightarrow Z$

Code HEX: C6                      Code DEC: 198  
Code BIN: 1 1 0 0 0 1 1 0      Indic. : Z  
Cycles : 3                      Octets : 01

Remarques: Cette instruction ne fait pas partie du jeu standard publié par Sharp. Elle portait la désignation TSIP.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LP 0A	8A
	xx01	LIA 0F	02 0F
	xx03	EXAM	DB
	xx04	LIA F0	02 F0
	xx06	TSMA	C6
	xx07	RTN	37

Description: Ce programme effectue un ET entre les constantes &0F et &F0. La valeur &0F est placée à l'adresse du registre M (&0A) et comparée (ET). Le résultat positionne ou non l'indicateur de nullité.

## WAIT n

Cette instruction est ineffective. La CPU ne fait aucune opération pendant 6+n cycles.

NOP de 6+n

Code HEX: 4E                      Code DEC: 78  
Code BIN: 0 1 0 0 1 1 1 0      Indic. :  
Cycles : 6+n                      Octets : 02

Remarques: Cette instruction possède les mêmes fonctionnalités que les instructions NOPT et NOPW.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LIA 02	02 02
	xx02	WAIT 60	4E 60
	xx04	RTN	37

Description: Ce programme place la constante &02 dans l'accumulateur et ne fait rien pendant &66 (102 cycles).

## WAITI (WAITJ)

Cette instruction est ineffective. La CPU ne fait aucune opération pendant 5+4\*d cycles.

I -> d  
NOP 5+4\*d

Code HEX: 4F                      Code DEC: 79  
Code BIN: 0 1 0 0 1 1 1 1      Indic. : Z  
Cycles : 5+4\*d                      Octets : 01

Remarques: Cette instruction ne fait pas partie du jeu standard publié par Sharp. Après quelques recherches, il a été possible de déterminer que cette instruction modifie le registre P et positionne l'indicateur de nullité. Il est possible qu'elle recelle encore d'autres fonctions.

Exemple:	Adresse	Assembleur	Code HEX
	xx00	LII 02	00 02
	xx02	WAITI	4F
	xx03	RTN	37

Description: Ce programme ne fait rien pendant x cycles. "x" dépend du registre I. Après exécution du programme, le registre P à la valeur &02 et l'indicateur de nullité est positionné.

## APPLICATIONS PRATIQUES PROGRAMMES EN LM

La description de chaque instruction est suivie d'un exemple de programme en langage machine. Cependant, ces derniers ne peuvent être exécutés en mode Basic ou RUN.

Toutefois, ils détaillent de façon précise le fonctionnement de chaque instruction.

Pour exécuter ces programmes, il faut les intégrer aux programmes Basic. Nous allons, à présent décrire cette méthode.

**1er exemple:** Le contenu de l'accumulateur est copié dans une cellule mémoire de la RAM externe.

Tout d'abord, il nous faut une zone libre de la mémoire où nous pourrons "poker" la routine en langage machine. Cette zone libre est définie dans la table donnée au chapitre 1. Dans tous nos exemples, l'octet de poids fort de la zone sera représenté par les caractères "xx" car il dépend du modèle d'ordinateur de poche utilisé.

Programme en langage machine:

Adresse	Assembleur	Code hex	Description
xx00	LIA 06	02 06	:06 -> accu
xx02	LIDP xx10	10 xx 10	:xx10 -> reg DP
xx05	STD	52	:06 -> (xx10)
xx06	RTN	37	,Retour

La constante &06 est chargée dans l'accumulateur et transférée à l'adresse xx10 à l'aide de l'instruction STD. Elle peut ensuite être lue par l'instruction Basic PEEK.

Le codage de la routine en mémoire s'effectue à l'aide de l'instruction Basic POKE suivie du code hexadécimal de toutes les instructions. La routine de notre exemple prend alors la forme suivante:

10: POKE &xx00,&02,&06,&10,&xx,&10,&52,&37

nous sauvegarderons la ligne 10 dans la zone des programmes et pourrons, éventuellement, y apporter des modifications ultérieures.

Les codes sont pokés par RUN. Lancez le programme par CALL &xx00 et examinez le contenu de l'adresse &xx10 au moyen de l'instruction Basic PEEK. Le contenu de l'adresse doit être égal à &06.

**2ème exemple:** Addition de deux nombres (8 bits).

L'exemple ci-dessous effectue l'addition de deux nombres sur 8 bits, c'est-à-dire compris entre 0 et 255. En effet, la représentation d'un nombre inférieur ou égal à 255 nécessite 8 bits ( $2^8=256$ ). Les deux membres de l'addition sont placés en mémoire. Le premier à l'adresse &xx00 et le second à l'adresse &xx01. Le résultat est placé à l'adresse &xx02.

Programme en langage machine:

Adresse	Assembleur	Code hex	Description
xx00	-		;1er membre
xx01	-		;2eme membre
xx02	-		;résultat
xx03	LIDP xx00	10 xx 00	;xx00 -> reg DP
xx06	LDD	57	;(xx00) -> accu
xx07	LP 03	83	;03 -> reg P
xx08	EXAB	DA	;(A) <-> (B)
xx09	LIDL 01	11 01	;01 -> reg DPL
xx0B	LDD	57	;(xx01) -> accu
xx0C	ADM	44	;(03)+A -> (03)
xx0D	EXAB	DA	;(A) <-> (03=B)
xx0E	LIDL 02	11 02	;02 -> reg DPL
xx10	STD	52	;A -> (xx02)
xx11	RTN	37	;retour

Tout d'abord, le premier nombre est transféré de

l'adresse &xx00 dans l'accumulateur. La constante &03 est placée dans le registre P afin que celui-ci pointe le registre B, dans lequel est placé le premier membre par EXAB. A présent, le second membre est chargé à partir de l'adresse xx01 dans l'accumulateur. ADM additionne le contenu du registre B (pointé par P) à l'accumulateur et place le résultat dans le registre B. EXAB transfère le résultat dans l'accumulateur, tandis que STD le place à l'adresse xx02.

La routine en langage machine peut être intégrée au programme Basic suivant:

```
10 : CLEAR
20 : POKE &xx03,&10,&xx,&00,&57,
    &83,&DA,&11,&01,&57,&44,&DA,
    &11,&02,&52,&37
30 : INPUT "1. MEMBRE: ";A,
    "2. MEMBRE: ";B
40 : POKE &xx00,A,B: CALL &xx03
50 : PRINT "SOMME=>", PEEK &xx02
60 : END
```

CALL &xx03 lance l'exécution de la routine en langage machine.

### 3. exemple: Addition de deux nombres (16 bits)

Cette fois-ci, nous allons additionner deux nombres compris entre 0 et 65535 (16 bits). La répartition des nombres entre octet de poids fort et octet de poids faible constitue la difficulté majeure de ce programme.

Le premier nombre est placé aux adresses &xx00 et &xx01. Le second est rangé aux adresses &xx02 et &xx03. Le résultat réside aux adresses &xx04 et &xx05.

Programme en langage machine:

Adresse	Assembleur	Code hex	Description
xx00	-		;1er membre LB

xx01	-		;1er membre HB
xx02	-		;2ème membre LB
xx03	-		;2ème membre HB
xx04	-		;résultat LB
xx05	-		;résultat HB
xx06	LIDP xx00	10 xx 00	;xx00 -> reg DP
xx09	LII 03	00 03	;03 -> reg I
xx0B	LP 02	82	;02 -> reg P
xx0C	MVWD	18	;(xx00-03)-> 2-5
xx0D	LP 04	84	;04 -> reg DPL
xx0E	ADB	14	;addition
xx0F	LIDL 04	11 04	;04 -> reg DPL
xx11	LII 01	00 01	;01 -> reg I
xx13	LP 04	84	;04 -> reg P
xx14	EXWD	19	;(4,5) -> RAM ext.
xx15	RTN	37	;retour

L'instruction MVWD copie le contenu des adresses &xx00 à &xx03 de la RAM externe aux adresses &02 - &05 (registres A,B,XL,XH) de la RAM interne.

L'instruction ADB additionne les registres 16 bits (XH,XL) et (B,A). Le résultat est placé dans les registres XL et XH. L'instruction EXWD copie le contenu des registres d'adresses &04 et &05 (XL,XH) aux adresses &xx04 et &xx05.

La routine en langage machine peut être intégrée au programme Basic suivant:

```
10 : CLEAR
20 : POKE &xx06,&10,&xx,&00,&00,&03,
    &82,&18,&84,&14,&11,&04,&00,&01,
    &84,&19,&37
30 : INPUT "1. MEMBRE: ";A: GOSUB 100
40 : POKE &xx00,L,H
50 : INPUT "2. MEMBRE: ";A: GOSUB 100
60 : POKE &xx02,L,H: CALL &xx06
70 : PRINT "SOMME=>", PEEK &xx05*256
    + PEEK &xx04
80 : END
100 : REM répartition octet fort - faible
110 : H=INT (A/256): L=A-H*256
120 : RETURN
```

CALL &xx06 (LIGNE 60) lance l'exécution de la routine en langage machine.

#### 4. exemple: Production de sons

Sur la majorité des modèles, le Beeper est associé à la ligne XOUT (vérifiez que c'est effectivement le cas de votre modèle à l'aide du diagramme donné en annexe). La fréquence du signal sonore peut être modulée. Le programme ci-dessous crée diverses fréquences en combinaison avec l'instruction WAIT. La tonalité varie entre 0 et 65535, ce qui permet de coupler fréquence et tonalité.

Programme en langage machine:

Adresse	Assembleur	Code hex	Description
xx00	LP09	89	;09 -> reg P
xx01	LIA 0D	02 0D	;13 -> accu
xx03	EXAM	DB	;13 -> reg L
xx04	LP 08	88	;08 -> reg P
xx05	LIA FE	02 FE	;254 -> accu
xx07	EXAM	DB	;254 -> reg K
xx08	LIB 00	03 00	;00 -> reg B
xx0A	LIA 01	02 01	;01 -> accu
xx0C	LIP 5F	12 5F	;95 -> reg P
xx0E	ORIM 10	61 10	;OU logique
xx10	OUTC	DF	;(5F) -> port C
xx11	WAIT FF	4E 0F	;att. 261 cycles
xx13	ANIM E9	60 E9	;ET logique
xx15	OUTC	DF	;(5F) -> port C
xx16	LP 08	88	;08 -> reg P
xx17	SBB	15	;soustraction
xx18	JRNZM 0D	29 0D	;saut -> xx0C
xx1A	RTN	37	;retour

Les adresses &xx0C à &xx15 créent le son. L'adresse mémoire &xx12 contient la fréquence. La boucle externe soustrait 1 du nombre sur 16 bits placé dans les registres K et L, jusqu'à ce que sa valeur soit égale à 0 et que l'indicateur de nullité soit positionné. Les adresses mémoire &xx02 (octet de poids

fort) et &xx06 (octet de poids faible) gèrent la durée.

La routine en langage machine peut être intégrée au programme Basic suivant:

```

10 : CLEAR
20 : POKE &xx00,&89,&02,&0D,&DB,&88,
    &02,&FE,&DB,&03,&00,&02,&01,&12,
    &5F,&61,&10,&DF
30 : POKE &xx11,&4E,&0F,&60,&E9,&DF,
    &88,&15,&29,&0D,&37
40 : INPUT "TONALITE: ";T: POKE &xx02,
    INT (T/256): POKE &xx06,T-INT(T/
    256)*256
50 : INPUT "FREQUENCE: ";F: POKE &xx12,F
60 : CALL &xx00
70 : GOTO 40
80 : END

```

Il est possible d'augmenter la durée de la boucle créée par OUTC.

#### 5ème exemple: Renumerotation

En Basic, il est souvent nécessaire de renuméroter les lignes d'un programme. La routine en langage machine ci-dessous renumérote les lignes d'un programme Basic. Malheureusement, ce programme ne gère pas les instructions GOTO et GOSUB. Les ouvrages de la série "Trucs et astuces" traitent de la structure des programmes Basic.

Programme en langage machine:

Adresse	Assembleur	Code hex	Description
xx00	LIDP 6F01	10 6F 01	;6F01 -> reg DP
xx03	LP 04	84	;04 -> reg P
xx04	MVMD	55	;LB-déb.BASIC ->XL
xx05	LIDL 02	11 02	;6F02 -> reg DP
xx07	LP 05	85	;05 -> reg P

xx08	MVMD	55	;HB-déb.BASIC -> XH
xx09	LP 08	88	;08 -> reg P
xx0A	LIA 10	02 10	;LB-N ligne -> A
xx0C	EXAM	DB	;accu <-> reg K
xx0D	LP 09	89	;09 -> reg P
xx0E	LIA 00	02 00	;HB-n ligne -> A
xx10	EXAM	DB	;accu <-> reg L
xx11	IXL	24	;x=x+1,(x)->accu
xx12	CPIA FF	67 FF	;A=ff?-> fin Basic
xx14	JRNZP	28 02	;non -> xx17
xx16	RTN	37	;retour
xx17	LP 09	89	;09 -> reg P
xx18	MVDM	53	;(L) -> (DP)
xx19	DECP	51	;08 -> reg P
xx1A	IX	04	;x=x+1;X->reg Dp
xx1B	MVMD	53	;(K) -> (DP)
xx1C	LIA 10	02 10	;LB-pas -> accu
xx1E	LIB 00	03 00	;HB-pas -> reg B
xx20	ADB	14	;addition (16 bits)
xx21	IXL	24	;x=x+1,(x)->accu
xx22	LP 04	84	;04 -> reg P
xx23	LIB 00	03 00	;00 -> reg B
xx25	ADB	14	;X = X + B,A
xx26	JRM 16	2D 16	;saut -> xx11

La constante &6F01 est placée dans le registre DP. Elle correspond au pointeur de début de Basic sur PC-1350 et doit être modifiée en fonction du modèle utilisé.

Le contenu de ce pointeur est ensuite placé dans le registre X. Le nouveau numéro de ligne (16 bits) est lui placé dans les registres K et L.

La renumérotation s'effectue tant que le contenu de ces adresses diffère de &FF. &FF correspond au caractère de fin de programme Basic.

Le numéro de ligne est placé aux octets suivants et incrémenté de la valeur du pas (STEP). Le registre X est incrémenté de la longueur de la ligne Basic.

La structure des programmes Basic est brièvement décrite ci-dessous:

xx00	FF	Début Basic
xx01	LB	N de ligne
xx02	HB	N de ligne
xx03	LB	nombre de carac. par ligne
xx04	xx	1er octet du programme Basic
xx05	xx	2eme octet du programme Basic

La routine en langage machine peut être intégrée à un programme Basic de la façon suivante. A correspond à l'adresse à partir de laquelle le programme sera "poké". P correspond au pointeur de début de Basic.

Listing:

```

50000:"Z" A=&6000:P=&6F01
50001:H= INT (P/256):L=P-H*256
50002:POKE A,&10,H,L,&84,&55,&11,L+1,&8
      5,&55,&88,&02,&0A
50003:POKE A+12,&DB,&89,&02,&00,&DB,&24
      ,&67,&FF,&28,&02,&37,&89
50004:POKE A+24,&53,&51,&04,&53,&02,&0A
      ,&03,&00,&14,&24,&84,&03,&0,&14,&
      2D,&16
50005:IF PEEK A<>16 GOTO "Z"
50006:"A" INPUT "DEBUT LIGNE:";Z,"LONG.
      LIGNE:";W
50007:POKE A+11,Z- INT (Z/256)*256:
      POKE A+15, INT (Z/256): POKE A+29
      ,W- INT (W/256)*256,3, INT (W/256
      )
50008:CALL A

```

6ème exemple: Conversion BIN-HEX et désassembleur

Le désassembleur utilise deux routines écrites en langage machine qui lisent une valeur de la ROM interne et effectuent la conversion binaire ->

hexadécimal. Le programme ne nécessite que la saisie de deux adresses système pour fonctionner sur tous les modèles de la gamme Sharp. Ces adresses sont reprises à la ligne 11 du programme Basic. M correspond à l'adresse où le programme en langage machine doit être "poké". Nous avons choisi l'adresse &5667 pour un PC-1450. Le nombre N=&5C2F donne l'adresse de fin de la mémoire Basic.

Le désassembleur décode les instructions machine en tenant compte des instructions sur 1,2,3 ou 4 octets. Il donne également les adresses absolue ou relatives des branchements. Les codes sont rangés dans une table B\$(255)\*5 et sauvegardés sur cassette à la suite du programme par l'instruction PRINT# B\$(\*)).

Après avoir tapé RUN, l'ordinateur charge la table, lance le programme. L'édition peut avoir lieu sur une imprimante. Ensuite, le programme demande l'adresse de début et de fin du désassemblage. La réponse doit impérativement correspondre à une adresse contenant un code machine (surtout pas à un argument!).

Programme en langage machine:

566F	02	LIA	6F
5671	03	LIB	56
5673	00	LII	04
5675	A0	LP	20
5676	35	DATA	
5677	A0	LP	20
5678	10	LIDP	56BB
567B	19	EXWD	
567C	37	RTN	
567D	86	LP	06
567E	02	LIA	2F
5680	DB	EXAM	
5681	50	INCP	
5682	02	LIA	5C

5684	DB	EXAM	
5685	02	LIA	F5
5687	26	IYS	
5688	37	RTN	
5689	78	CALL	567D
568C	10	LIDP	56BD
568F	57	LDD	
5690	34	PUSH	
5691	58	SWP	
5692	78	CALL	56AF
5695	5B	POP	
5696	78	CALL	56AF
5699	02	LIA	00
569B	26	IYS	
569C	37	RTN	
569D	78	CALL	567D
56A0	10	LIDP	56BC
56A3	57	LDD	
56A4	34	PUSH	
56A5	58	SWP	
56A6	78	CALL	56AF
56A9	5B	POP	
56AA	78	CALL	56AF
56AD	2D	JRM	22 <568C>
56AF	64	ANIA	0F
56B1	75	SBIA	0A
56B3	3A	JRCP	03 <56B7>
56B5	74	ADIA	07
56B7	74	ADIA	3A
56B9	26	IYS	
56BA	37	RTN	

De &566F à &567C, 5 constantes sont transférées à partir de l'adresse B,A (16 bits) vers les adresses &56BB-&56C0.

Le programme ci-dessous effectue la conversion binaire -> hexadécimal et possède deux adresses de



début: &5689 convertit un nombre à deux positions et &569D convertit un nombre à quatre positions. Dans tous les cas, le résultat est placé aux adresses de la variable Z\$ qui débutent à la suite du programme Basic.

Programme Basic:

```

1:REM M=ADR(ML)
10:" " DIM B$(255)*5: INPUT #B$(*)
11:M=&566F:N=&5C2F
12:H= INT (M/256):L=M-H*256:S= INT (N/256):T=N-S*256
13:IF L>170 PRINT "RENOUVELER M": END
20:POKE M,&2,,3,,0,4,&A0,&35,&A0,&10,
  H,L+76,&19,&37,&86,&2,T,&DB,&50,&2,S
  ,&DB,&2
22:POKE M+23,&F5,&26,&37,&78,H,L+14,&10
  ,H,,&57,&34,&58,&78,H,L+64,&5B,&78,
  H,L+64,&2
23:POKE M+43,0,&26,&37,&78,H,L+14,&10,H
  ,,&57,&34,&58,&78,H,L+64,&5B,&78,H,
  L+64
24:POKE M+62,&2D,&22,&64,&F,&75,&3A,&3A,
  &3,&74,&7,&74,&3A,&26,&37: GOTO 29
25:H= INT (I/256):W=I-H*256: POKE M+1,W
  ,3,H: CALL M:X= PEEK (M+76)
26:POKE M+31,L+1: POKE M+51,L+3: CALL M
  +46:B$=Z$
27:POKE M+31,L+76: CALL M+26:C$=Z$
28:RETURN
29:"Z"R=1: INPUT "SORTIE IMPRIM. ?",Q$:
  IF Q$="0" LET R=0: PRINT = LPRINT
30:"A" INPUT "ADR. DEBUT: ";A,"ADR.FIN: ";
  E
40:FOR I=A TO E
50:GOSUB 25

```

```

60:IF X<4 OR (X>16 AND X<20) OR (X>39
  AND X<48) OR (X>55 AND X<60) OR (X>2
  11 AND X<215) LET I=I+1: GOTO 200
70:IF X>95 AND X<118 AND X<>105 OR X=78
  LET I=I+1: GOTO 200
80:IF X>119 AND X<128 AND X<>122 OR X=1
  6 LET I=I+2: GOTO 300
85:IF X=105 GOTO 500
90:IF X=122 LET I=I+3: GOTO 400
95:IF X>224 LET F$= LEFT$ (B$(X),3)+
  "H$="": POKE M+31,L+77: CALL M+26:G
  $= RIGHT$ (B$(X),2)+Z$:I=I+1: GOTO 1
  00
96:IF X>127 AND X<192 LET F$= LEFT$ (B$
  (X),2)+ "G$= RIGHT$ (B$(X),2):H$
  ="": GOTO 100
98:F$=B$(X):G$="":H$=""
100:PRINT B$+" "+C$+" "+F$+" "+G$+" "+H$
110:IF R=1 AND LEN G$>2 OR R=1 AND H$<>"
  " PRINT F$+" "+G$+" "+H$
120:NEXT I: GOTO 30
200:POKE M+31,L+77: CALL M+26:G$=Z$:F$=B
  $(X):H$=""
210:IF X>39 AND X<48 OR X>55 AND X<60
  THEN 250
220:GOTO 100
250:D=I+(( INT (X/2)=X/2)*2-1)* PEEK (M+
  77)
255:POKE M+80, INT (D/256),D- INT (D/256
  )*256: POKE M+51,L+80: POKE M+31,L+8
  1: CALL M+46
260:H$="<"+Z$+">": GOTO 100
300:POKE M+31,L+78: POKE M+51,L+77: CALL
  M+46:G$=Z$:F$=B$(X):H$="": GOTO 100
400:U= PEEK (M+77): POKE M+31,L+77: CALL
  M+26:G$=Z$: POKE M+31,L+79: POKE M+5
  1,L+78: CALL M+46:H$=">"+Z$+"<"

```

```

410:F$=B$(X): GOTO 100
500:IF U=0 PRINT "PTJ ERROR": END
505:PRINT B$+" "+C$+" "+B$(X):I=I+1:
      GOSUB 25
510:GOSUB 25
520:POKE M+51,L+77: POKE M+31,L+78: CALL
      M+46:G$=Z$:F$="=A? >"
525:PRINT B$+" "+C$+" "+F$+" "+G$
530:U=U-1:I=I+3: IF U<>0 GOTO 510
540:POKE M+31,L+77: CALL M+26:G$=C$+Z$
550:PRINT B$+"      ELSE> ";G$
560:I=I+1: GOTO 120

```

B\$(0)=LII	B\$(128)=LP 00
B\$(1)=L1J	B\$(129)=LP 01
B\$(2)=LIA	B\$(130)=LP 02
B\$(3)=LIB	B\$(131)=LP 03
B\$(4)=IX	B\$(132)=LP 04
B\$(5)=DX	B\$(133)=LP 05
B\$(6)=IY	B\$(134)=LP 06
B\$(7)=DY	B\$(135)=LP 07
B\$(8)=MUW	B\$(136)=LP 08
B\$(9)=EXW	B\$(137)=LP 09
B\$(10)=MVB	B\$(138)=LP 0A
B\$(11)=EXB	B\$(139)=LP 0B
B\$(12)=ADN	B\$(140)=LP 0C
B\$(13)=SBN	B\$(141)=LP 0D
B\$(14)=ADW	B\$(142)=LP 0E
B\$(15)=SBW	B\$(143)=LP 0F
B\$(16)=LIDP	B\$(144)=LP 10
B\$(17)=LIDL	B\$(145)=LP 11
B\$(18)=LIP	B\$(146)=LP 12
B\$(19)=LIQ	B\$(147)=LP 13
B\$(20)=ADB	B\$(148)=LP 14
B\$(21)=SBB	B\$(149)=LP 15
B\$(22)=?022?	B\$(150)=LP 16
B\$(23)=?023?	B\$(151)=LP 17
B\$(24)=MUWD	B\$(152)=LP 18
B\$(25)=EXWD	B\$(153)=LP 19
B\$(26)=MUBD	B\$(154)=LP 1A
B\$(27)=EXBD	B\$(155)=LP 1B
B\$(28)=SRW	B\$(156)=LP 1C
B\$(29)=SLW	B\$(157)=LP 1D
B\$(30)=FILM	B\$(158)=LP 1E
B\$(31)=FILD	B\$(159)=LP 1F
B\$(32)=LDP	B\$(160)=LP 20

B\$(33)=LDQ	B\$(161)=LP 21
B\$(34)=LDR	B\$(162)=LP 22
B\$(35)=RA	B\$(163)=LP 23
B\$(36)=IXL	B\$(164)=LP 24
B\$(37)=DXL	B\$(165)=LP 25
B\$(38)=IYS	B\$(166)=LP 26
B\$(39)=DYS	B\$(167)=LP 27
B\$(40)=JRNZP	B\$(168)=LP 28
B\$(41)=JRNZM	B\$(169)=LP 29
B\$(42)=JRNCP	B\$(170)=LP 2A
B\$(43)=JRNCM	B\$(171)=LP 2B
B\$(44)=JRP	B\$(172)=LP 2C
B\$(45)=JRM	B\$(173)=LP 2D
B\$(46)=?046?	B\$(174)=LP 2E
B\$(47)=LOOP	B\$(175)=LP 2F
B\$(48)=STP	B\$(176)=LP 30
B\$(49)=STQ	B\$(177)=LP 31
B\$(50)=STR	B\$(178)=LP 32
B\$(51)=?051?	B\$(179)=LP 33
B\$(52)=PUSH	B\$(180)=LP 34
B\$(53)=DATA	B\$(181)=LP 35
B\$(54)=?054?	B\$(182)=LP 36
B\$(55)=RTN	B\$(183)=LP 37
B\$(56)=JRZP	B\$(184)=LP 38
B\$(57)=JRZM	B\$(185)=LP 39
B\$(58)=JRCP	B\$(186)=LP 3A
B\$(59)=JRCM	B\$(187)=LP 3B
B\$(60)=?060?	B\$(188)=LP 3C
B\$(61)=?061?	B\$(189)=LP 3D
B\$(62)=?062?	B\$(190)=LP 3E
B\$(63)=?063?	B\$(191)=LP 3F
B\$(64)=INCI	B\$(192)=INCJ
B\$(65)=DECI	B\$(193)=DECJ

B\$(66)=INCA	B\$(194)=INCB
B\$(67)=DECA	B\$(195)=DECB
B\$(68)=ADM	B\$(196)=ADCM
B\$(69)=SBM	B\$(197)=SBCM
B\$(70)=ANMA	B\$(198)=TSIP
B\$(71)=ORMA	B\$(199)=CPMA
B\$(72)=INCK	B\$(200)=INCL
B\$(73)=DECK	B\$(201)=DECL
B\$(74)=INCM	B\$(202)=INCN
B\$(75)=DECM	B\$(203)=DECN
B\$(76)=INA	B\$(204)=INB
B\$(77)=NOPW	B\$(205)=?205?
B\$(78)=WAIT	B\$(206)=NOPT
B\$(79)=WAITJ	B\$(207)=?207?
B\$(80)=INCP	B\$(208)=SC
B\$(81)=DECP	B\$(209)=RC
B\$(82)=STD	B\$(210)=SR
B\$(83)=MU DM	B\$(211)=WRIT
B\$(84)=READM	B\$(212)=ANID
B\$(85)=MUMD	B\$(213)=ORID
B\$(86)=READ	B\$(214)=TSID
B\$(87)=LDD	B\$(215)=?215?
B\$(88)=SWP	B\$(216)=LEAVE
B\$(89)=LDM	B\$(217)=?217?
B\$(90)=SL	B\$(218)=EXAB
B\$(91)=POP	B\$(219)=EXAM
B\$(92)=?092?	B\$(220)=?220?
B\$(93)=OUTA	B\$(221)=OUTB
B\$(94)=?094?	B\$(222)=?222?
B\$(95)=OUTF	B\$(223)=OUTC
B\$(96)=ANIM	B\$(224)=CAL00
B\$(97)=ORIM	B\$(225)=CAL01
B\$(98)=TSIM	B\$(226)=CAL02

B\$(99)=CPIM	B\$(227)=CAL03
B\$(100)=ANIA	B\$(228)=CAL04
B\$(101)=ORIA	B\$(229)=CAL05
B\$(102)=TSIA	B\$(230)=CAL06
B\$(103)=CPIA	B\$(231)=CAL07
B\$(104)=?104?	B\$(232)=CAL08
B\$(105)=DTJ	B\$(233)=CAL09
B\$(106)=?106?	B\$(234)=CAL0A
B\$(107)=TEST	B\$(235)=CAL0B
B\$(108)=?108?	B\$(236)=CAL0C
B\$(109)=?109?	B\$(237)=CAL0D
B\$(110)=?110?	B\$(238)=CAL0E
B\$(111)=?111?	B\$(239)=CAL0F
B\$(112)=ADIM	B\$(240)=CAL10
B\$(113)=SBIM	B\$(241)=CAL11
B\$(114)=?114?	B\$(242)=CAL12
B\$(115)=?115?	B\$(243)=CAL13
B\$(116)=ADIA	B\$(244)=CAL14
B\$(117)=SBIA	B\$(245)=CAL15
B\$(118)=?118?	B\$(246)=CAL16
B\$(119)=?119?	B\$(247)=CAL17
B\$(120)=CALL	B\$(248)=CAL18
B\$(121)=JP	B\$(249)=CAL19
B\$(122)=PTJ	B\$(250)=CAL1A
B\$(123)=?123?	B\$(251)=CAL1B
B\$(124)=JPNZ	B\$(252)=CAL1C
B\$(125)=JPNC	B\$(253)=CAL1D
B\$(126)=JPZ	B\$(254)=CAL1E
B\$(127)=JPC	B\$(255)=CAL1F

## ANNEXES

Résumé des Instructions :

HEX DEC Instruction Fonction Octets Cycl.

00	000	LIJ n	n -> I-Register	2	4
01	001	LIJ n	n -> J-Register	2	4
02	002	LIA n	n -> Akku	2	4
03	003	LIB n	n -> B-Register	2	4
04	004	IX	X+1 -> X ; X -> DP	1	6
05	005	DX	X-1 -> X ; X -> DP	1	6
06	006	IY	Y+1 -> Y ; Y -> DP	1	6
07	007	DY	Y-1 -> Y ; Y -> DP	1	6
08	008	MVW	(Q) -> (P)	1	5+2*d
09	009	EXW	(P) <-> (Q)	1	6+3*d
0A	010	MVB	(Q) -> (P)	1	5+2*d
0B	011	EXB	(P) <-> (Q)	1	6+3*d
0C	012	ADN	(P)+A -> (P) (BCD)	1	7+3*d
0D	013	SBN	(P)-A -> (P) (BCD)	1	7+3*d
0E	014	ADW	(P)+(Q) -> (P) (BCD)	1	7+3*d
0F	015	SBW	(P)-(Q) -> (P) (BCD)	1	7+3*d
10	016	LIDP nm	n -> DPH, m -> DPL	3	8
11	017	LIDL n	n -> DPL	2	5
12	018	LIP n	n -> P-Register	2	4
13	019	LIQ n	n -> Q-Register	2	4
14	020	ADB	(P+1,P)+(BA)->(P+1,P)	1	5
15	021	SBB	(P+1,P)-(BA)->(P+1,P)	1	5
18	024	MVWD	(DP) -> (P)	1	5+4*d

19	025	EXWD	(DP) <-> (P)	1	7+6*d
1A	026	MVBD	(DP) -> (P)	1	5+4*d
1B	027	EXBD	(DP) <-> (P)	1	7+6*d
1C	028	SRW	4 bit shift right	1	5+1*d
1D	029	SLW	4 bit shift left	1	5+1*d
1E	030	FILM	A -> (P), P+1 -> P	1	5+1*d
1F	031	FILD	A ->(DP), DP+1->DP	1	4+3*d
20	032	LDP	P -> A	1	2
21	033	LDQ	Q -> A	1	2
22	034	LDR	R -> A	1	2
23	035	RA	O -> A	1	2
24	036	IXL	X+1->X, X->DP, (DP)->A	1	7
25	037	DXL	X-1->X, X->DP, (DP)->A	1	7
26	038	IYS	Y+1->Y, Y->DP, A->(DP)	1	6
27	039	DYS	Y-1->Y, Y->DP, A->(DP)	1	6
28	040	JRNZP n	IF Z=0 PC+1+n -> PC ELSE PC+2 -> PC	2	7/4
29	041	JRNZM n	IF Z=0 PC+1-n -> PC ELSE PC+2 -> PC	2	7/4
2A	042	JRNCP n	IF C=0 PC+1+n -> PC ELSE PC+2 -> PC	2	7/4
2B	043	JRNCM n	IF C=0 PC+1-n -> PC ELSE PC+2 -> PC	2	7/4
2C	044	JRP n	PC+1+n -> PC	2	7
2D	045	JRM n	PC+1-n -> PC	2	7
2F	047	LOOP n	(R)-1 -> (R)	2	10/7

			IF C=0 PC+1-n -> PC ELSE PC+2 -> PC		
30	048	STP	A -> P	1	2
31	049	STQ	A -> Q	1	2
32	050	STR	A -> R	1	2
34	052	PUSH	R-1 -> R, A -> (R)	1	3
35	053	DATA	(BA) -> (P)	1	11+4d
37	055	RTN	(R)->PCL, (R+1)->PC R+2 -> R	1	4
38	056	JRZP n	IF Z=1 PC+1+n -> PC ELSE PC+2 -> PC	2	7/4
39	057	JRZM n	IF Z=1 PC+1-n -> PC ELSE PC+2 -> PC	2	7/4
3A	058	JRCP n	IF C=1 PC+1+n -> PC ELSE PC+2 -> PC	2	7/4
3B	059	JRCM n	IF C=1 PC+1-n -> PC ELSE PC+2 -> PC	2	7/4
40	064	INCI	I=I+1 -> I	1	4
41	065	DECI	I=I-1 -> I	1	4
42	066	INCA	A=A+1 -> A	1	4
43	067	DECA	A=A-1 -> A	1	4
44	068	ADM	(P)+A -> (P)	1	3
45	069	SBM	(P)-A -> (P)	1	3
46	070	ANMA	(P)A A -> (P)	1	3
47	071	ORMA	(P)A A -> (P)	1	3
48	072	INCK	K=K+1 -> K	1	4

49	073	DECK	K=K-1 -> K	1	4
4A	074	INCM	M=M+1 -> M	1	4
4B	075	DECM	M=M-1 -> M	1	4
4C	076	INA	IA-Port -> A	1	2
4D	077	NOPW	No Operation	1	2
4E	078	WAIT n	No Operation	2	6+n
4F	079	WAITI	No Operation	1	5+4*d
50	080	INCP	P=P+1 -> P	1	2
51	081	DECP	P=P-1 -> P	1	2
52	082	STD	A -> (DP)	1	2
53	083	MVDM	(P) -> (DP)	1	3
54	084	READM	(PC+1) -> (P)	1	3
55	085	MVMD	(DP) -> (P)	1	3
56	086	READ	(PC+1) -> A	1	3
57	087	LDD	(DP) -> A	1	3
58	088	SWP	Akku: (1-4)<->(5-8)	1	2
59	089	LDM	(P) -> A	1	2
5A	090	SL	1 bit shift left	1	2
5B	091	POP	(R) -> A, R+1 -> R	1	2
5D	093	OUTA	(5C) -> IA-Port	1	3
5F	095	OUTF	(5E) -> FO-Port	1	3
60	096	ANIM n	(P)A n -> (P)	2	4
61	097	ORIM n	(P)V n -> (P)	2	4
62	098	TSIM n	(P)A n Z	2	4
63	099	CPIM n	(P)-n Z,C	2	4
64	100	ANIA n	A^n -> A	2	4

65	101	ORIA n	A^n -> A	2	4
66	102	TSIA n	A^n Z	2	4
67	103	CPIA n	A-n Z, C	2	4
69	105	DTJ	Do Table Jump	-	-
6B	107	TEST n	n -> TEST	2	4
70	112	ADIM n	(P)+n -> (P)	2	4
71	113	SBIM n	(P)-n -> (P)	2	4
74	116	ADIA n	A+n -> A	2	4
75	117	SBIA n	A-n -> A	2	4
78	120	CALL nm	(PC+3) -> (R-1,R-2) R-3 -> R, nm -> PC	3	8
79	121	JP nm	n -> PCH, m -> PCL	3	6
7A	122	PTJ	Prepare Table Jump	4	9
7C	124	JPNZ nm	IF Z=0 n->PCH,m->PCL ELSE PC+3 -> PC	3	6
7D	125	JPNC nm	IF C=0 n->PCH,m->PCL ELSE PC+3 -> PC	3	6
7E	126	JPZ nm	IF Z=1 n->PCH,m->PCL ELSE PC+3 -> PC	3	6
7F	127	JPC nm	IF C=1 n->PCH,m->PCL ELSE PC+3 -> PC	3	6
80	128	LP 1	1 -> P (1=00 - 3F)	1	2
+1	+1				
C0	192	INCJ	J=J+1 -> J	1	4
C1	193	DECJ	J=J-1 -> J	1	4
C2	194	INCB	B=B+1 -> B	1	4

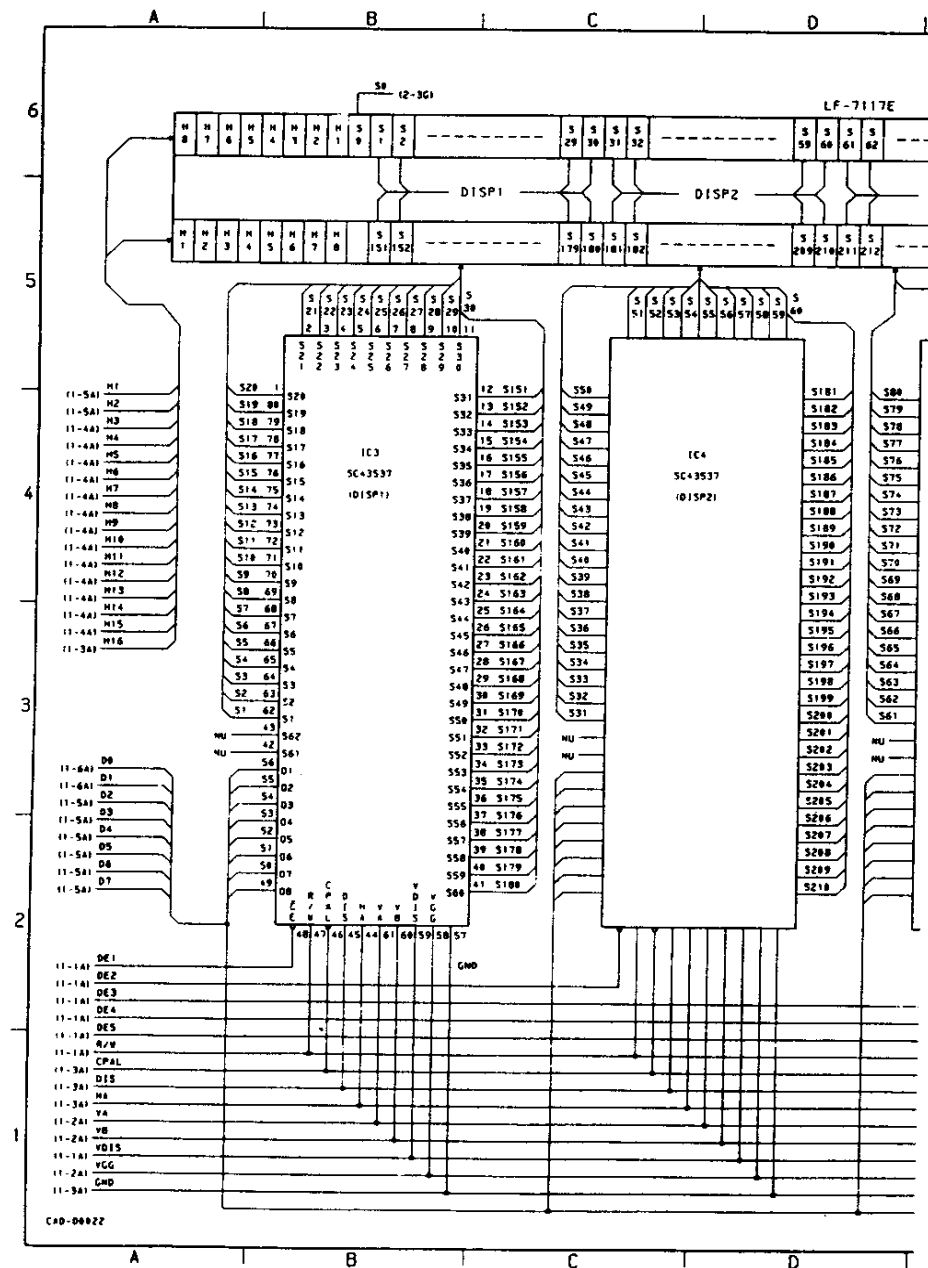


**SC60220 IC Pinout and Connections:**

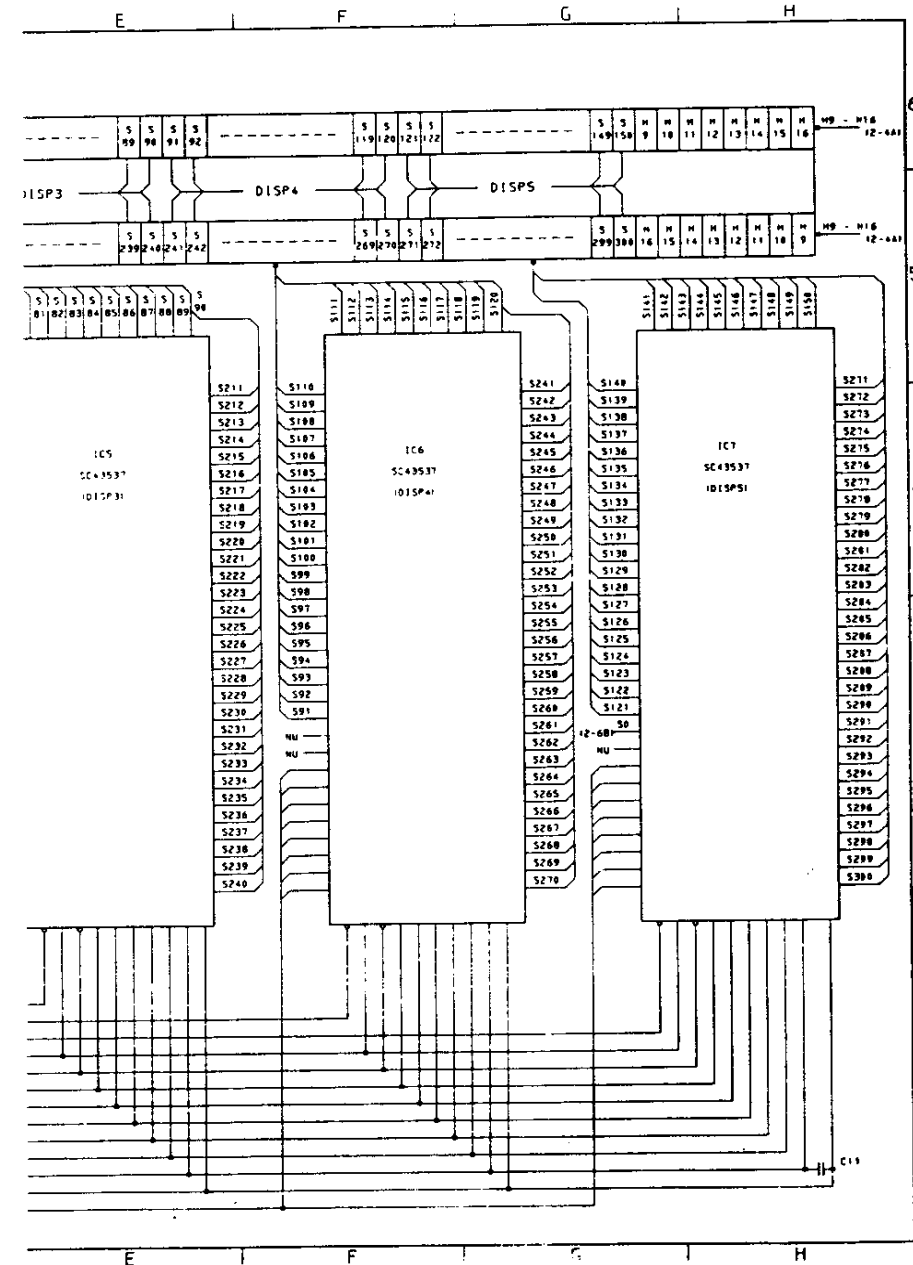
- Pin 1:** VCC
- Pin 2:** D0
- Pin 3:** D1
- Pin 4:** D2
- Pin 5:** D3
- Pin 6:** D4
- Pin 7:** D5
- Pin 8:** D6
- Pin 9:** D7
- Pin 10:** A14
- Pin 11:** A13
- Pin 12:** A12
- Pin 13:** A11
- Pin 14:** A10
- Pin 15:** A9
- Pin 16:** A8
- Pin 17:** A7
- Pin 18:** A6
- Pin 19:** A5
- Pin 20:** A4
- Pin 21:** A3
- Pin 22:** A2
- Pin 23:** A1
- Pin 24:** A0
- Pin 25:** VCC
- Pin 26:** RAN1
- Pin 27:** RAN2
- Pin 28:** QPCE
- Pin 29:** QUT5
- Pin 30:** QUT4
- Pin 31:** QUT3
- Pin 32:** QUT2
- Pin 33:** QUT1
- Pin 34:** QUT0
- Pin 35:** QUT7
- Pin 36:** QUT6
- Pin 37:** QUT5
- Pin 38:** QUT4
- Pin 39:** QUT3
- Pin 40:** QUT2
- Pin 41:** QUT1
- Pin 42:** QUT0
- Pin 43:** QUT7
- Pin 44:** QUT6
- Pin 45:** QUT5
- Pin 46:** QUT4
- Pin 47:** QUT3
- Pin 48:** QUT2
- Pin 49:** QUT1
- Pin 50:** QUT0
- Pin 51:** QUT7
- Pin 52:** QUT6
- Pin 53:** QUT5
- Pin 54:** QUT4
- Pin 55:** QUT3
- Pin 56:** QUT2
- Pin 57:** QUT1
- Pin 58:** QUT0
- Pin 59:** QUT7
- Pin 60:** QUT6
- Pin 61:** QUT5
- Pin 62:** QUT4
- Pin 63:** QUT3
- Pin 64:** QUT2
- Pin 65:** QUT1
- Pin 66:** QUT0
- Pin 67:** QUT7
- Pin 68:** QUT6
- Pin 69:** QUT5
- Pin 70:** QUT4
- Pin 71:** QUT3
- Pin 72:** QUT2
- Pin 73:** QUT1
- Pin 74:** QUT0
- Pin 75:** QUT7
- Pin 76:** QUT6
- Pin 77:** QUT5
- Pin 78:** QUT4
- Pin 79:** QUT3
- Pin 80:** QUT2
- Pin 81:** QUT1
- Pin 82:** QUT0
- Pin 83:** QUT7
- Pin 84:** QUT6
- Pin 85:** QUT5
- Pin 86:** QUT4
- Pin 87:** QUT3
- Pin 88:** QUT2
- Pin 89:** QUT1
- Pin 90:** QUT0
- Pin 91:** QUT7
- Pin 92:** QUT6
- Pin 93:** QUT5
- Pin 94:** QUT4
- Pin 95:** QUT3
- Pin 96:** QUT2
- Pin 97:** QUT1
- Pin 98:** QUT0
- Pin 99:** QUT7
- Pin 100:** QUT6
- Pin 101:** QUT5
- Pin 102:** QUT4
- Pin 103:** QUT3
- Pin 104:** QUT2
- Pin 105:** QUT1
- Pin 106:** QUT0
- Pin 107:** QUT7
- Pin 108:** QUT6
- Pin 109:** QUT5
- Pin 110:** QUT4
- Pin 111:** QUT3
- Pin 112:** QUT2
- Pin 113:** QUT1
- Pin 114:** QUT0
- Pin 115:** QUT7
- Pin 116:** QUT6
- Pin 117:** QUT5
- Pin 118:** QUT4
- Pin 119:** QUT3
- Pin 120:** QUT2
- Pin 121:** QUT1
- Pin 122:** QUT0
- Pin 123:** QUT7
- Pin 124:** QUT6
- Pin 125:** QUT5
- Pin 126:** QUT4
- Pin 127:** QUT3
- Pin 128:** QUT2
- Pin 129:** QUT1
- Pin 130:** QUT0
- Pin 131:** QUT7
- Pin 132:** QUT6
- Pin 133:** QUT5
- Pin 134:** QUT4
- Pin 135:** QUT3
- Pin 136:** QUT2
- Pin 137:** QUT1
- Pin 138:** QUT0
- Pin 139:** QUT7
- Pin 140:** QUT6
- Pin 141:** QUT5
- Pin 142:** QUT4
- Pin 143:** QUT3
- Pin 144:** QUT2
- Pin 145:** QUT1
- Pin 146:** QUT0
- Pin 147:** QUT7
- Pin 148:** QUT6
- Pin 149:** QUT5
- Pin 150:** QUT4
- Pin 151:** QUT3
- Pin 152:** QUT2
- Pin 153:** QUT1
- Pin 154:** QUT0
- Pin 155:** QUT7
- Pin 156:** QUT6
- Pin 157:** QUT5
- Pin 158:** QUT4
- Pin 159:** QUT3
- Pin 160:** QUT2
- Pin 161:** QUT1
- Pin 162:** QUT0
- Pin 163:** QUT7
- Pin 164:** QUT6
- Pin 165:** QUT5
- Pin 166:** QUT4
- Pin 167:** QUT3
- Pin 168:** QUT2
- Pin 169:** QUT1
- Pin 170:** QUT0
- Pin 171:** QUT7
- Pin 172:** QUT6
- Pin 173:** QUT5
- Pin 174:** QUT4
- Pin 175:** QUT3
- Pin 176:** QUT2
- Pin 177:** QUT1
- Pin 178:** QUT0
- Pin 179:** QUT7
- Pin 180:** QUT6
- Pin 181:** QUT5
- Pin 182:** QUT4
- Pin 183:** QUT3
- Pin 184:** QUT2
- Pin 185:** QUT1
- Pin 186:** QUT0
- Pin 187:** QUT7
- Pin 188:** QUT6
- Pin 189:** QUT5
- Pin 190:** QUT4
- Pin 191:** QUT3
- Pin 192:** QUT2
- Pin 193:** QUT1
- Pin 194:** QUT0
- Pin 195:** QUT7
- Pin 196:** QUT6
- Pin 197:** QUT5
- Pin 198:** QUT4
- Pin 199:** QUT3
- Pin 200:** QUT2
- Pin 201:** QUT1
- Pin 202:** QUT0
- Pin 203:** QUT7
- Pin 204:** QUT6
- Pin 205:** QUT5
- Pin 206:** QUT4
- Pin 207:** QUT3
- Pin 208:** QUT2
- Pin 209:** QUT1
- Pin 210:** QUT0
- Pin 211:** QUT7
- Pin 212:** QUT6
- Pin 213:** QUT5
- Pin 214:</**



# 2. PC-1550 display circuit



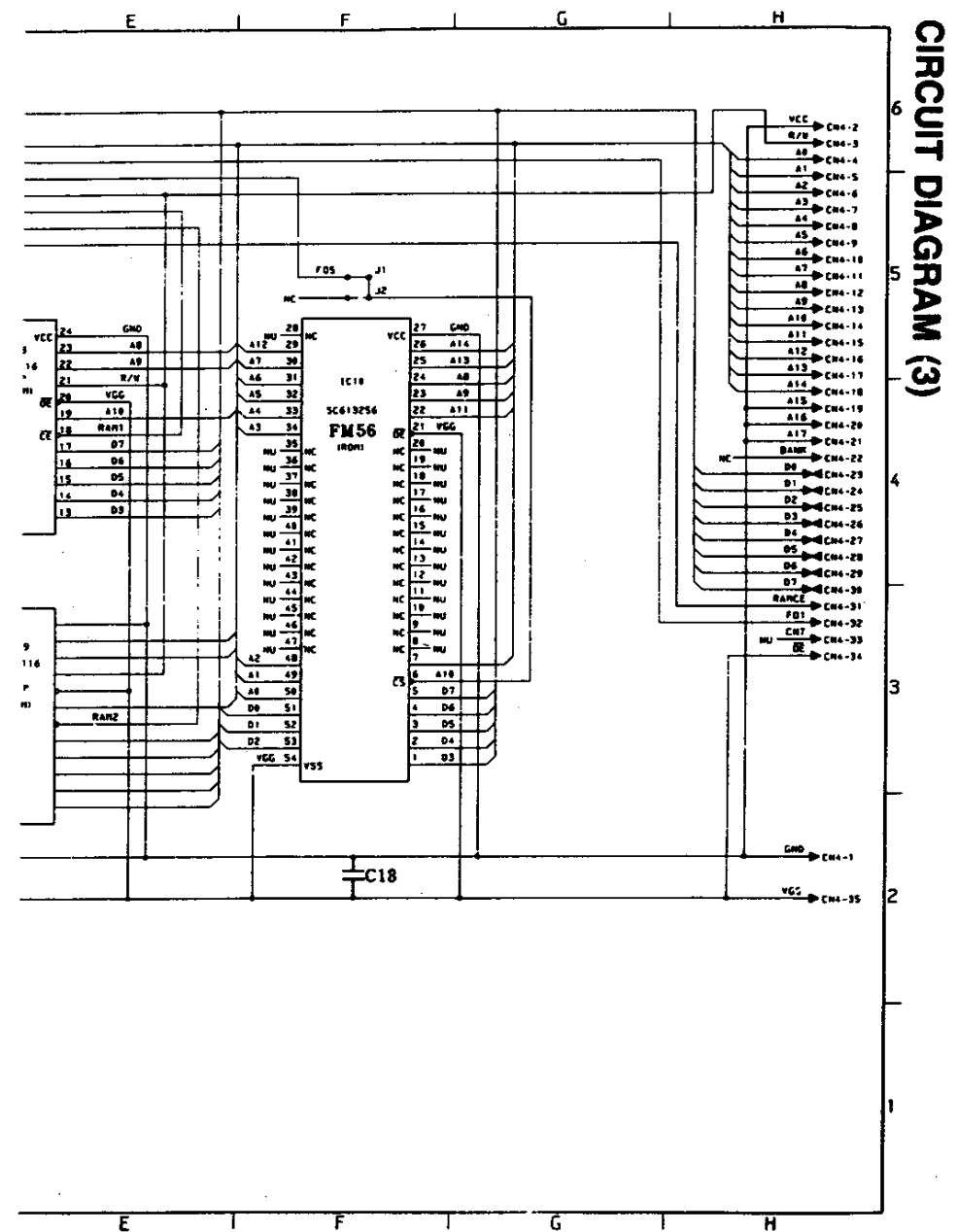
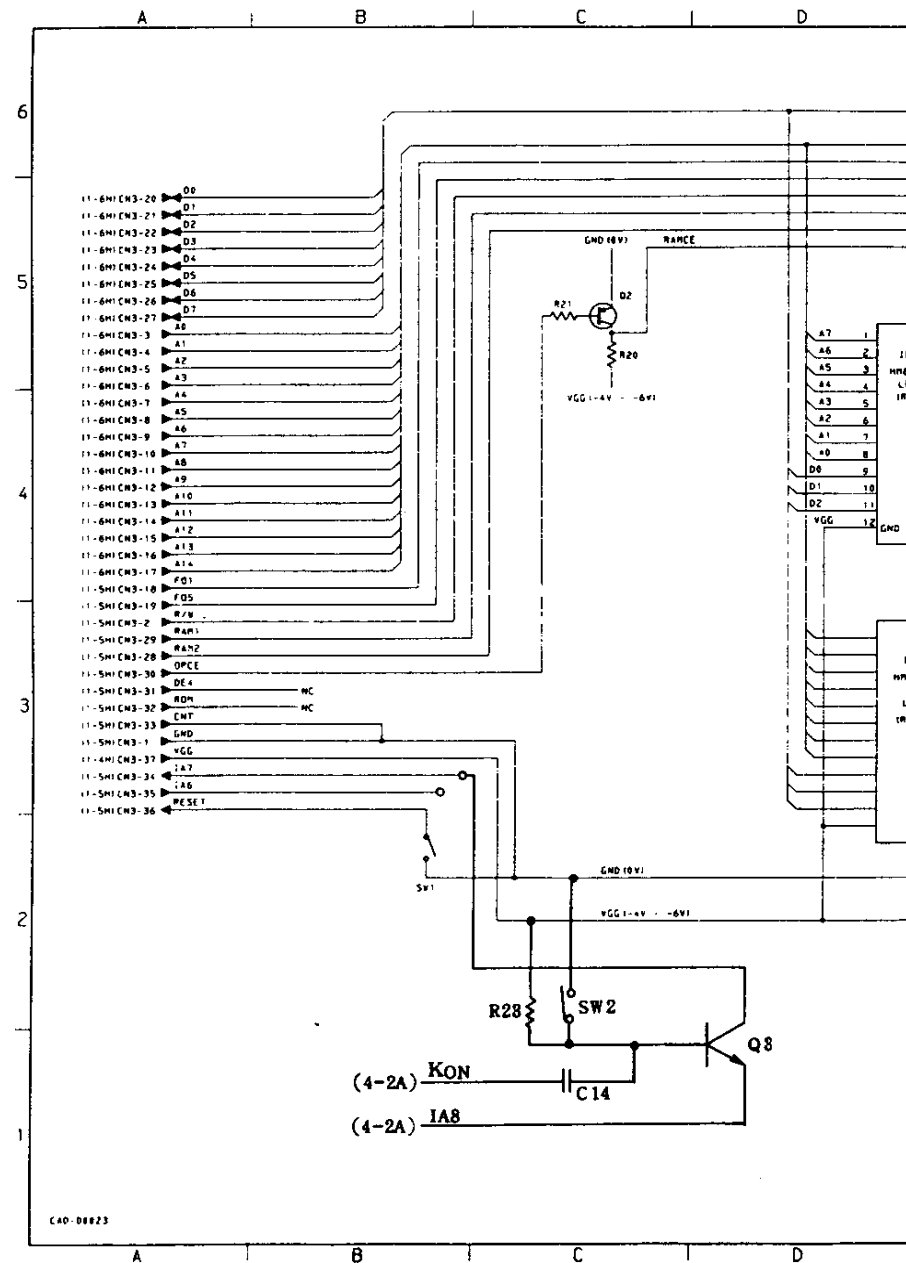
220



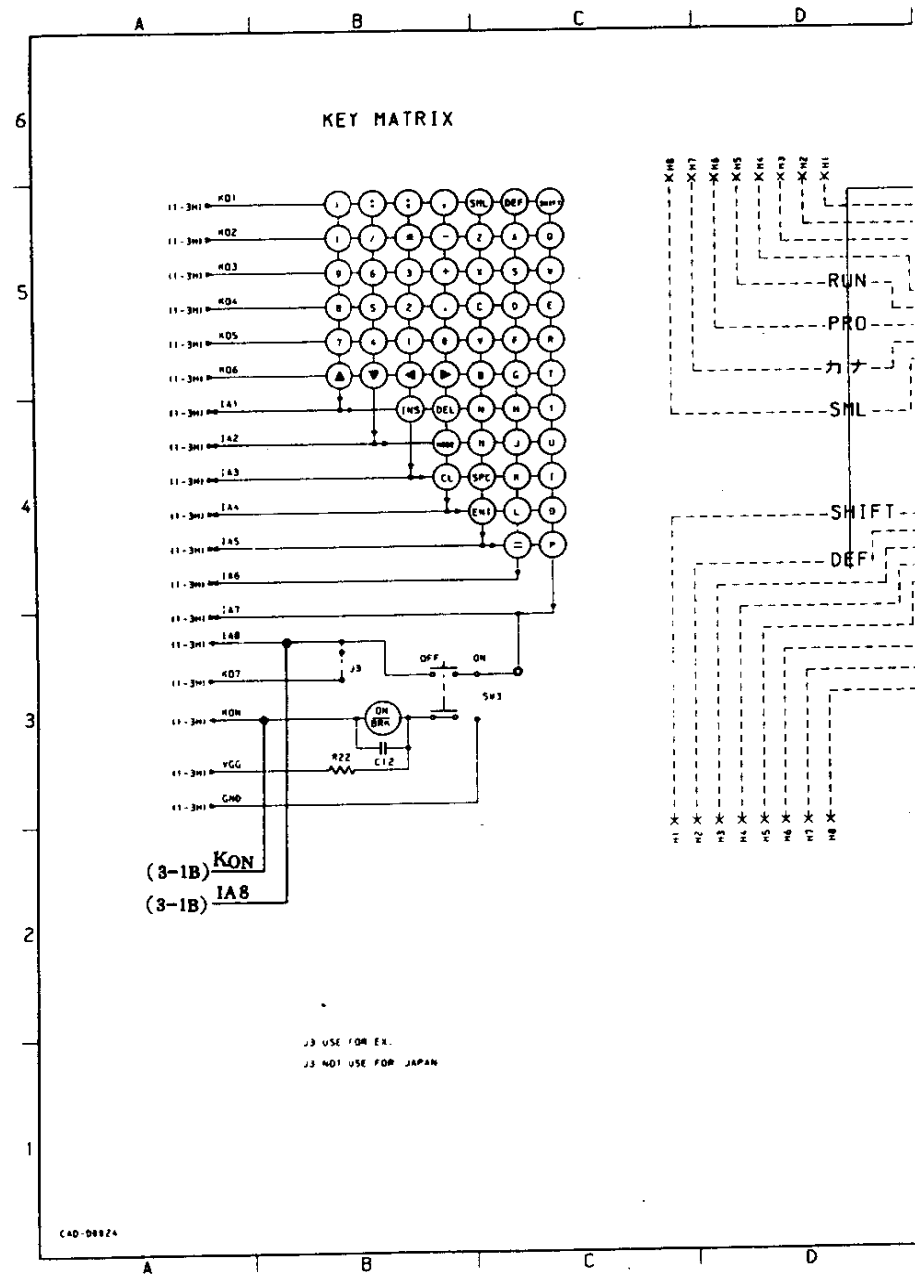
## CIRCUIT DEAGRAM (2)

221

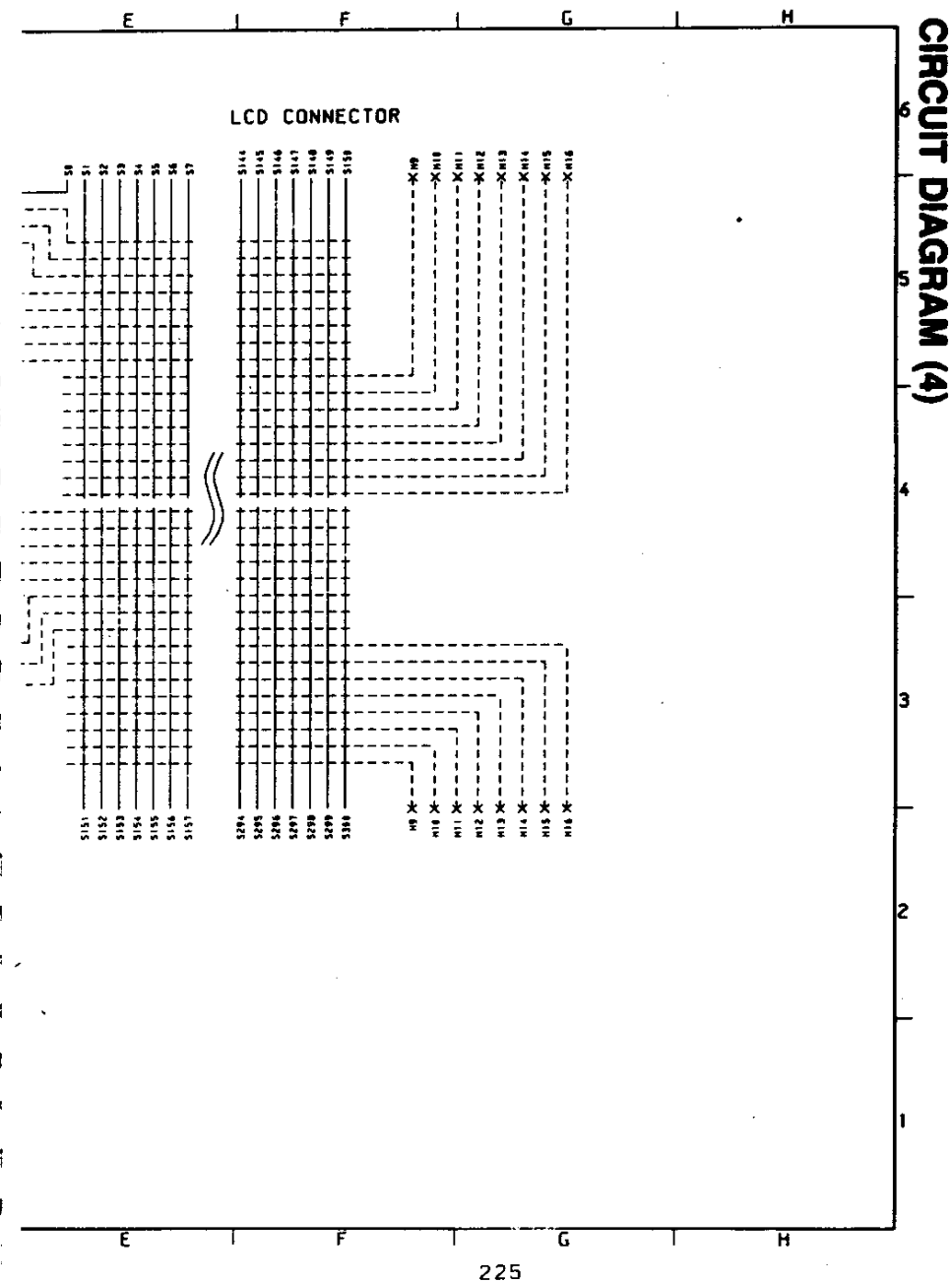
### 3. PC-1350 memory circuit (ROM: SC613256)



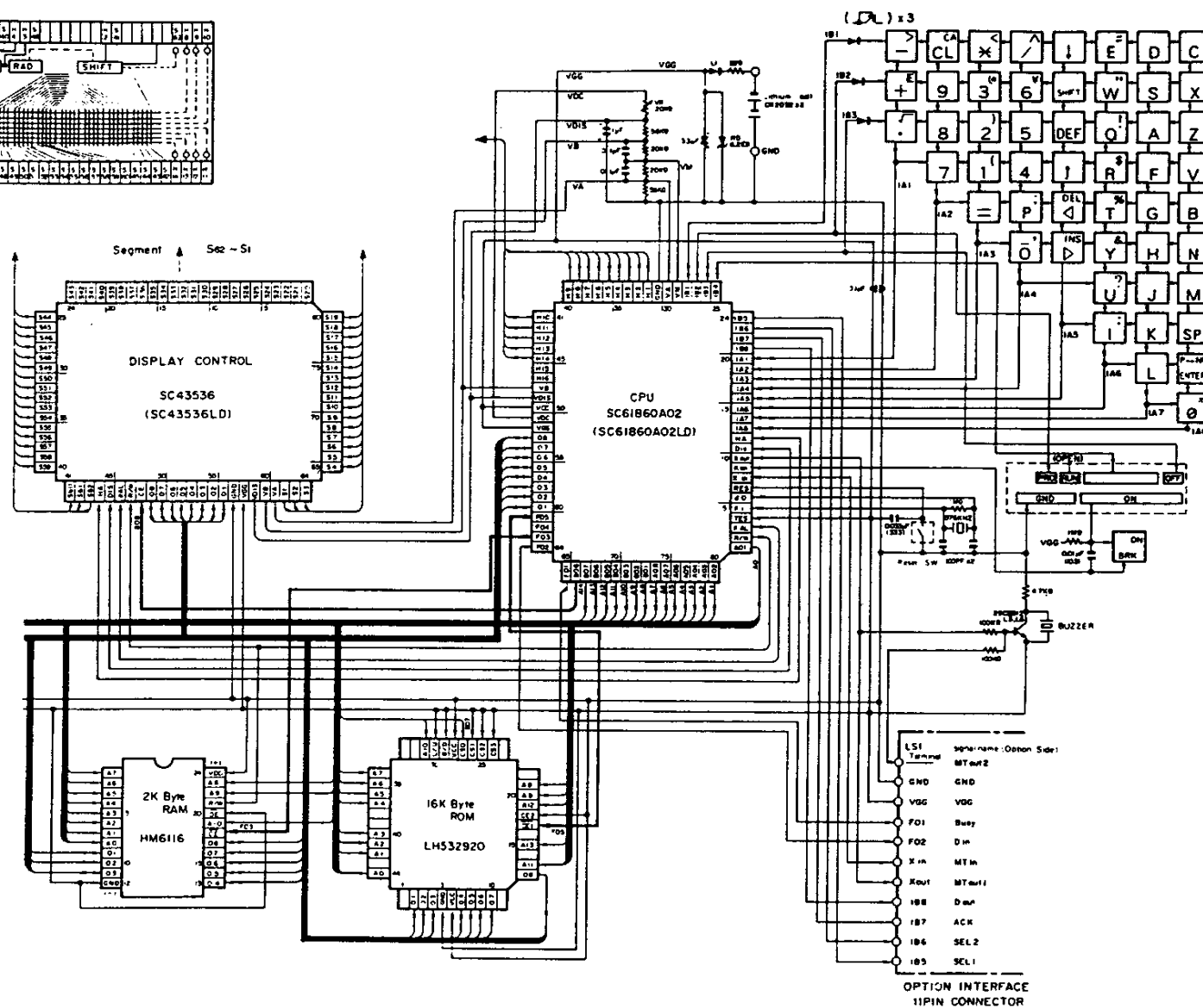
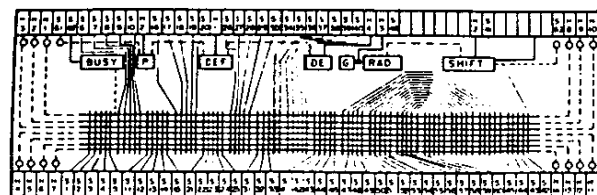
#### 4. PC-1350 key/LCD matrix circuits

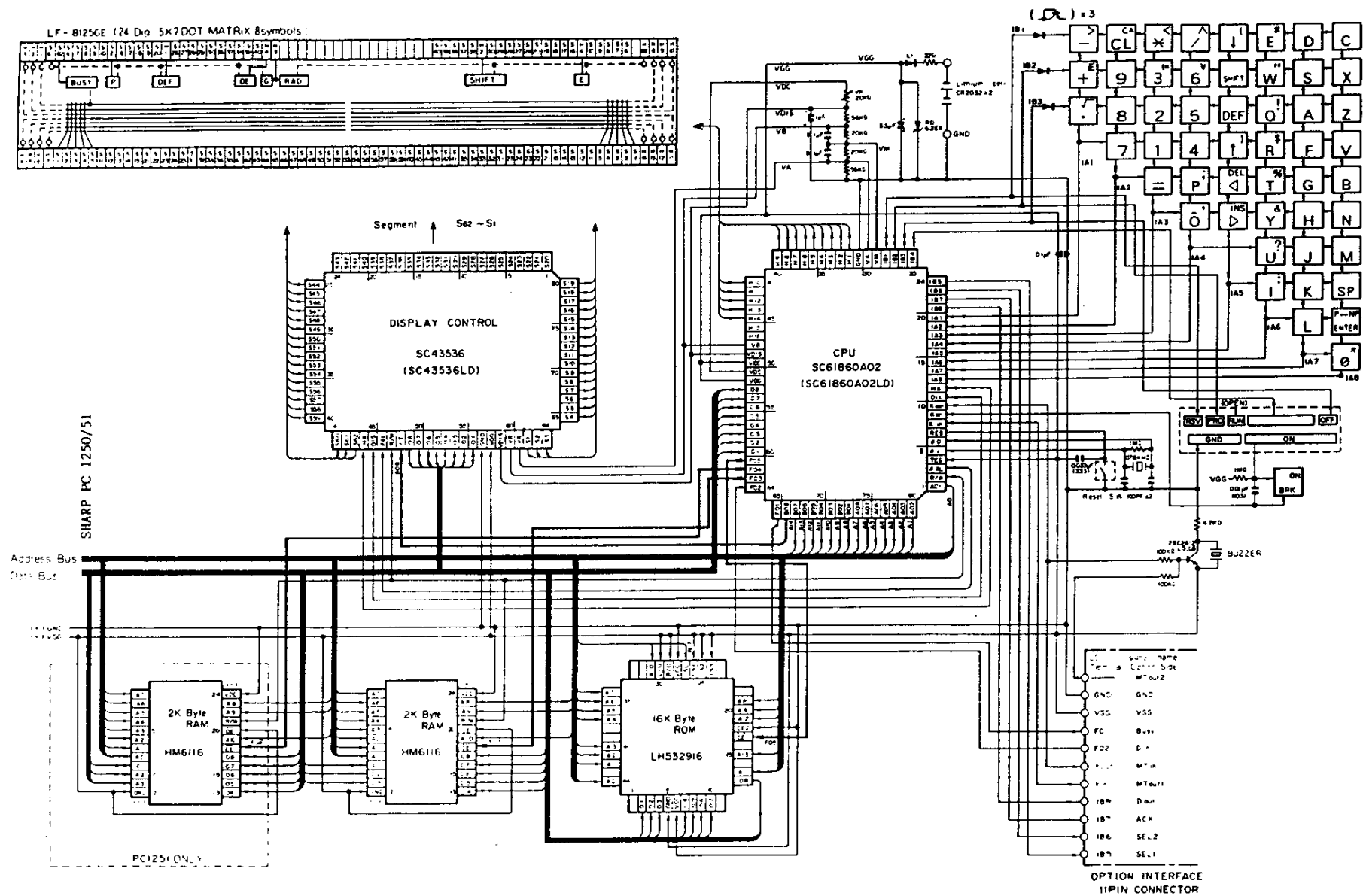


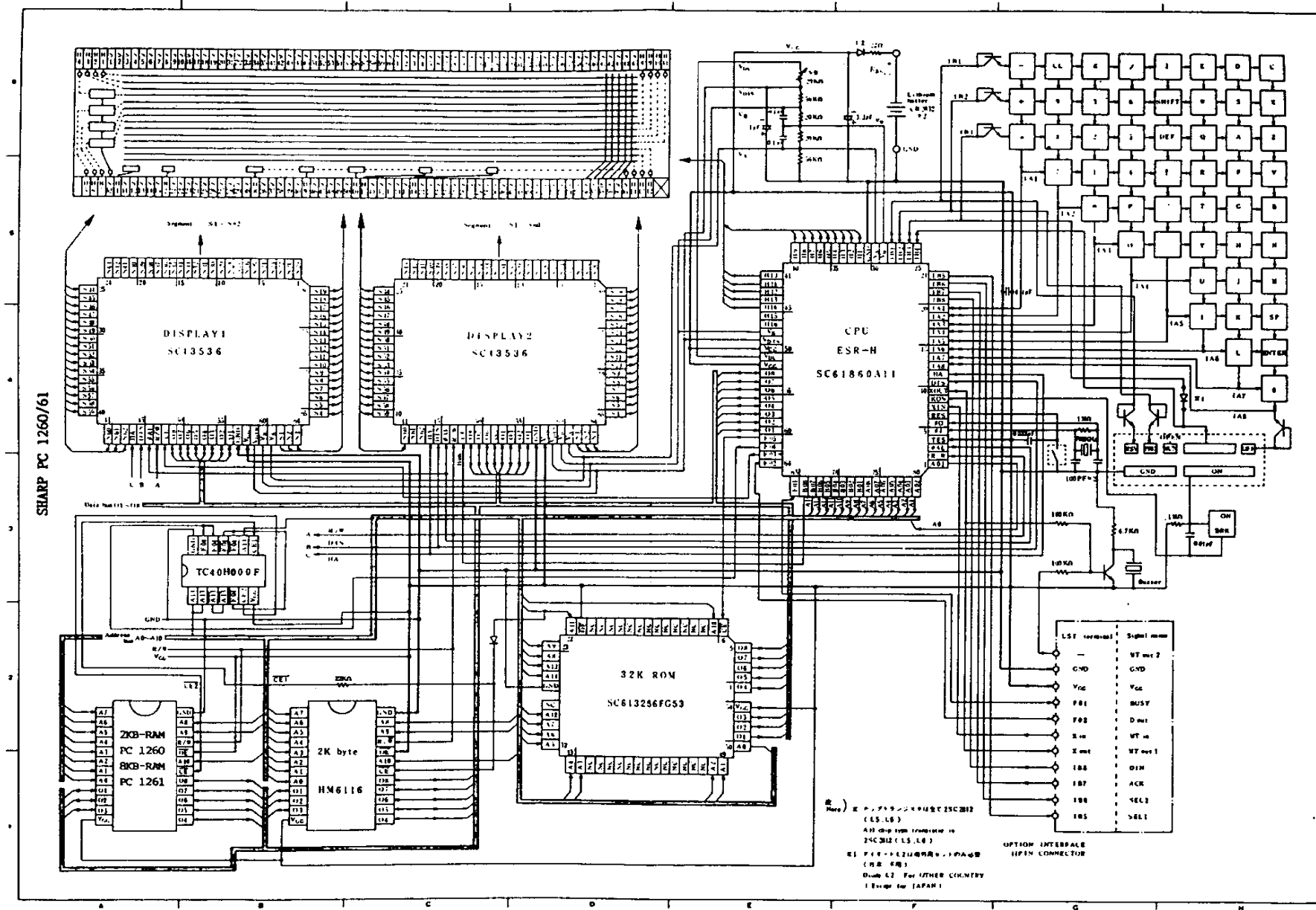
224

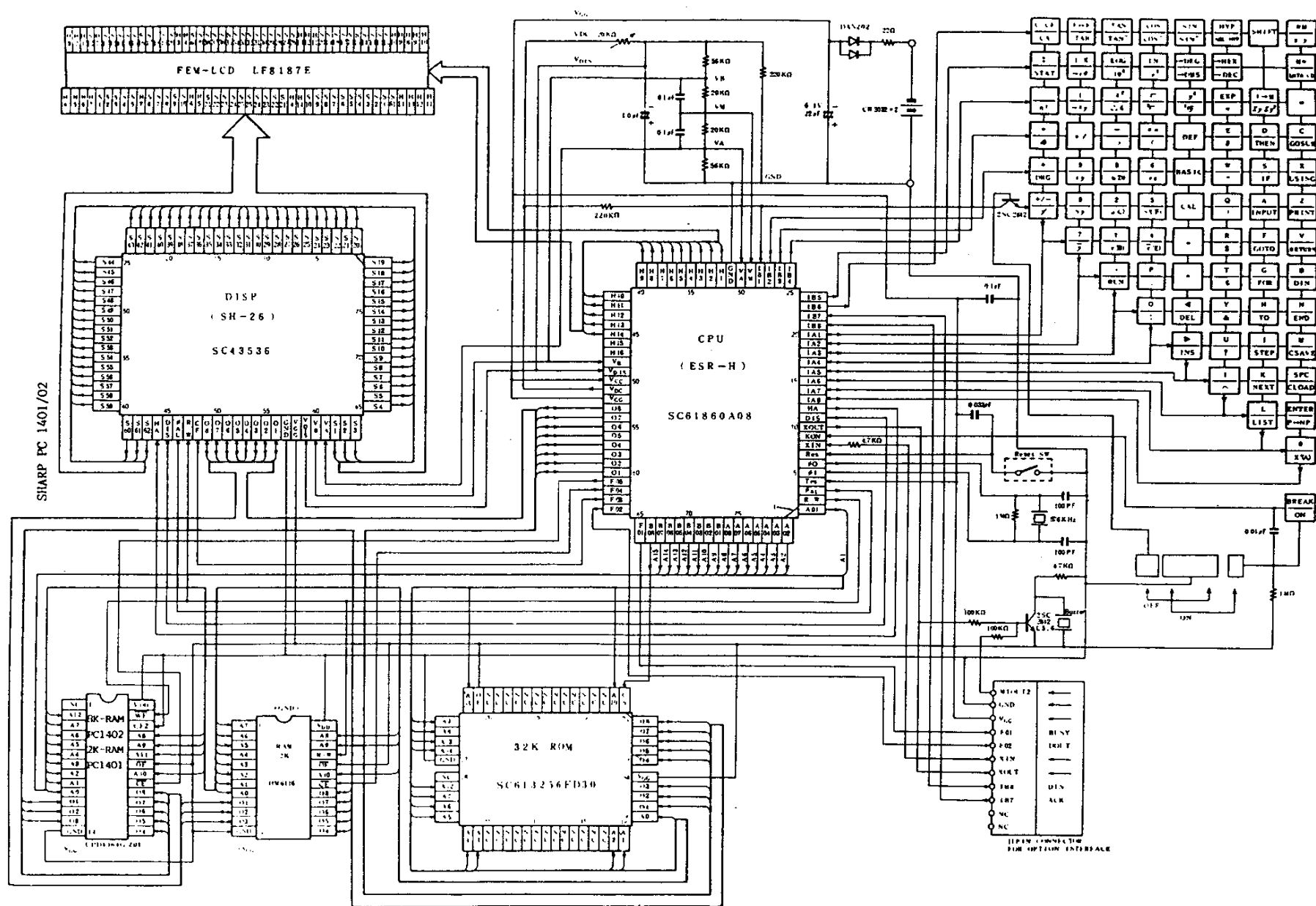


225

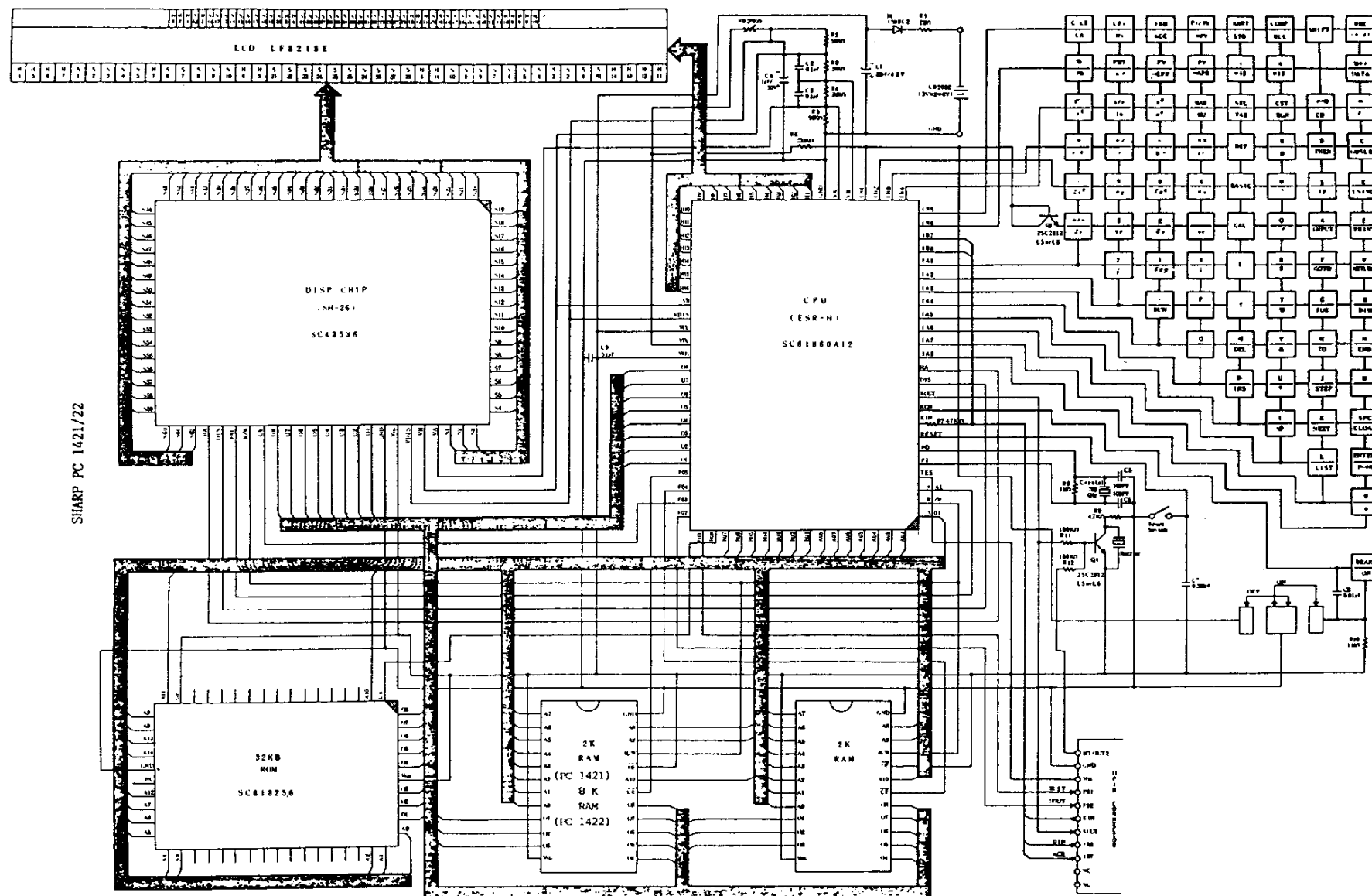




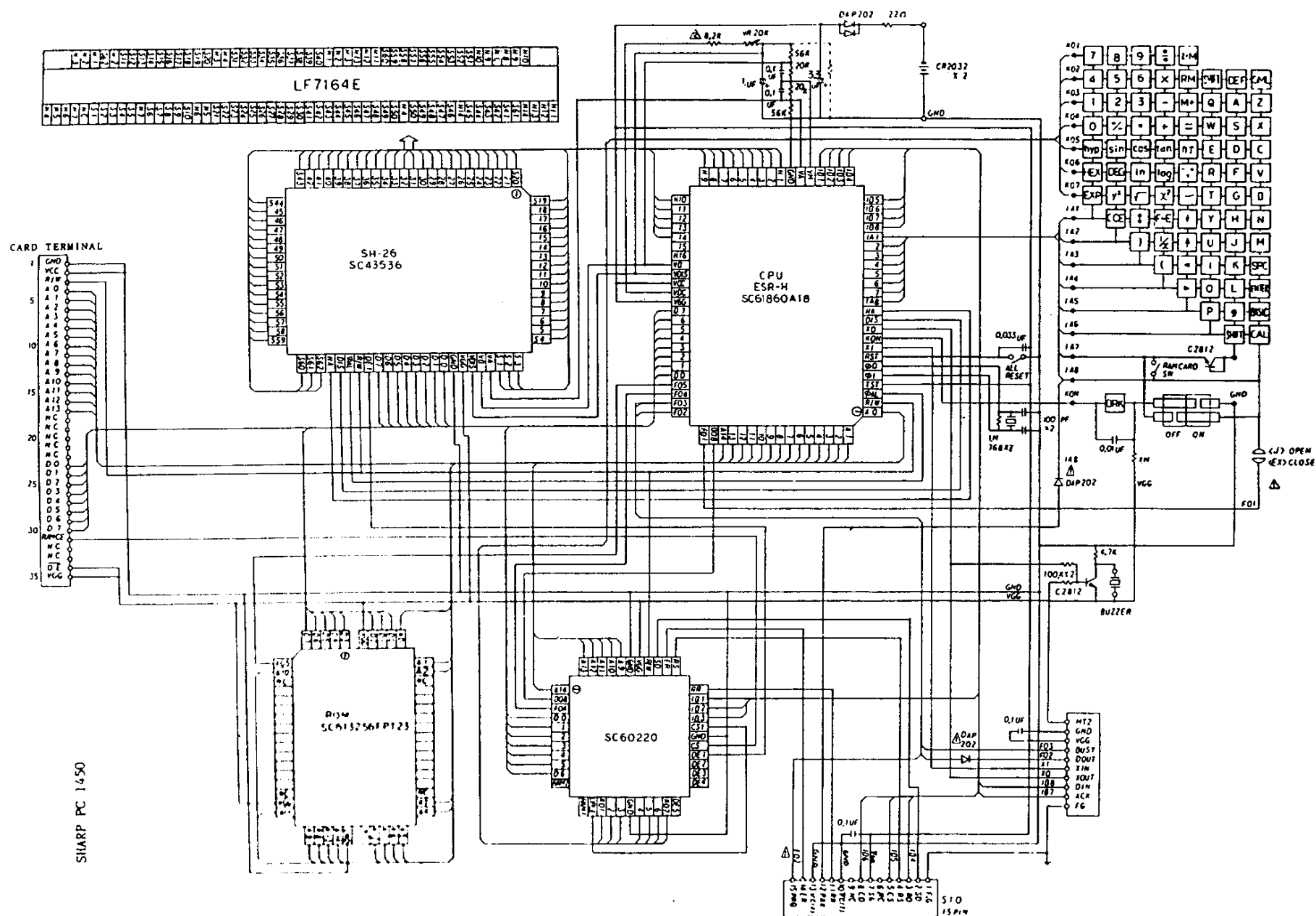


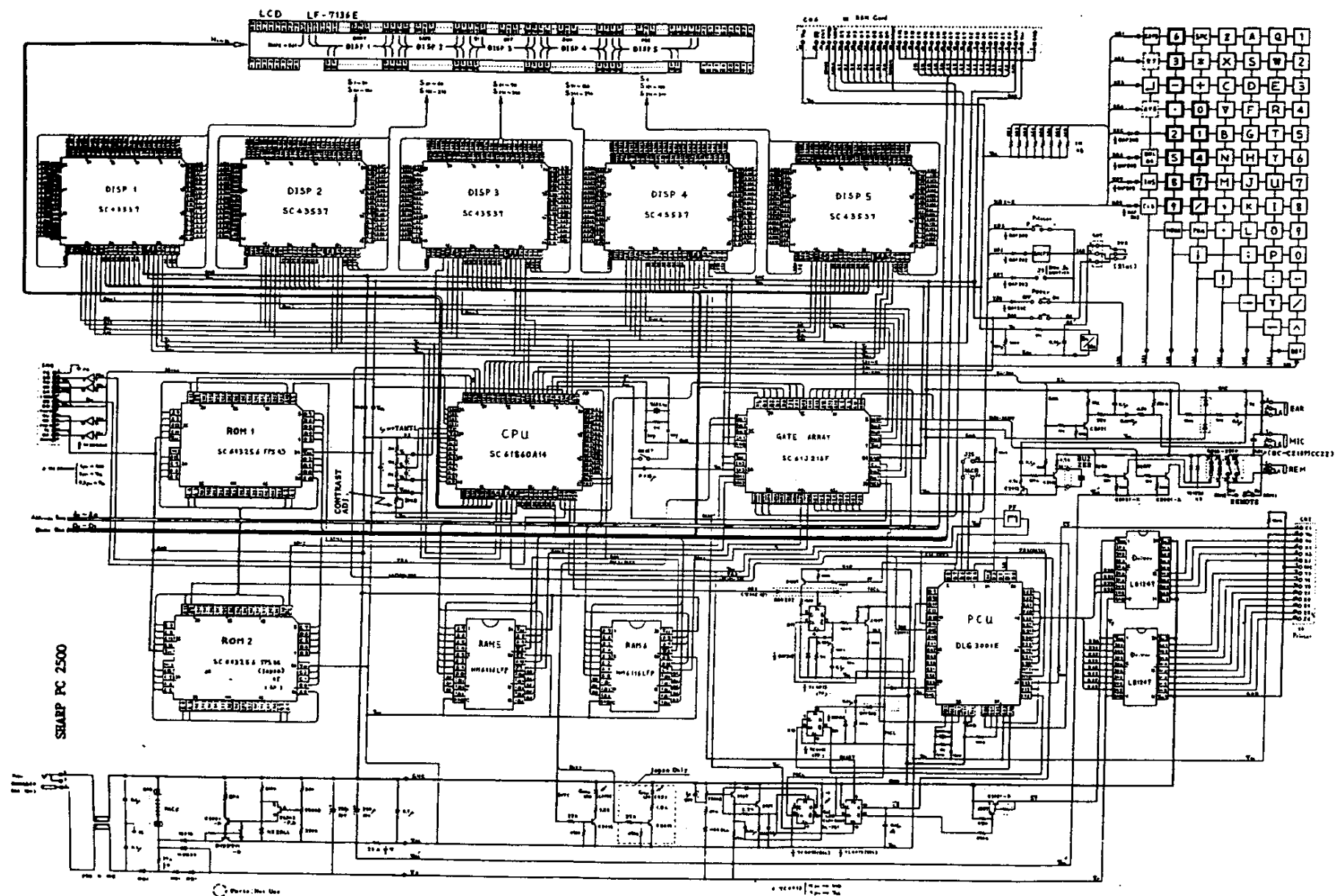


SHARP PC 1421/22









	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0 LII	16 LIDP	32 LDP	48 STP	64 INCI	80 INCP	96 ANIM	112 ADIM	128 LP 00	144 LP 10	160 LP 20	176 LP 30	192 INCJ	208 SC	224 CAL00	240 CAL10
1	1 LIJ	17 LIDL	33 LDQ	49 STQ	65 DECI	81 DECP	97 ORIM	113 SBIM	129 LP 01	145 LP 11	161 LP 21	177 LP 31	193 DECJ	209 RC	225 CAL01	241 CAL11
2	2 LIA	18 LIP	34 LDR	50 STR	66 INCA	82 STD	98 TSIH	114 ?	130 LP 02	146 LP 12	162 LP 22	178 LP 32	194 INCB	210 SR	226 CAL02	242 CAL12
3	3 LIB	19 LIO	35 RA	51 ?	67 DECA	83 MVDM	99 CPIM	115 ?	131 LP 03	147 LP 13	163 LP 23	179 LP 33	195 DECB	211 WRIT?	227 CAL03	243 CAL13
4	4 IX	20 ADB	36 IXL	52 PUSH	68 ADM	84 READM	100 ANIA	116 ADIA	132 LP 04	148 LP 14	164 LP 24	180 LP 34	196 ADCM	212 ANID	228 CAL04	244 CAL14
5	5 DX	21 SBB	37 DXL	53 DATA	69 SBM	85 MVMD	101 ORIA	117 SBIA	133 LP 05	149 LP 15	165 LP 25	181 LP 35	197 SBCM	213 ORID	229 CAL05	245 CAL15
6	6 LY	22 ?	38 IYS	54 ?	70 ANMA	86 READ	102 TSIA	118 ?	134 LP 06	150 LP 16	166 LP 26	182 LP 36	198 TSMA	214 TSID	230 CAL06	246 CAL16
7	7 DY	23 ?	39 DYS	55 RTN	71 ORMA	87 LDD	103 CPIA	119 ?	135 LP 07	151 LP 17	167 LP 27	183 LP 37	199 CPMA	215 ?	231 CAL07	247 CAL17
8	8 MVW	24 MVWD	40 JRNZP	56 JRZP	72 INCK	88 SWP	104 ?	120 CALL	136 LP 08	152 LP 18	168 LP 28	184 LP 38	200 INCL	216 LEAVE	232 CAL08	248 CAL18
9	9 EXW	25 EXWD	41 JRNZM	57 JRZM	73 DECK	89 LDM	105 DTJ	121 JP	137 LP 09	153 LP 19	169 LP 29	185 LP 39	201 DECL	217 ?	233 CAL09	249 CAL19
A	10 MVB	26 MVBD	42 JRNCP	58 JRCP	74 INCM	90 SL	106 ?	122 PTJ	138 LP 0A	154 LP 1A	170 LP 2A	186 LP 3A	202 INCN	218 EXAB	234 CAL0A	250 CAL1A
B	11 EXB	27 EXBD	43 JRNCH	59 JRCH	75 DECH	91 POP	107 TEST	123 ?	139 LP 0B	155 LP 1B	171 LP 2B	187 LP 3B	203 DECH	219 EXAM	235 CAL0B	251 CAL1B
C	12 ADN	28 SRW	44 JRP	60 ?	76 INA	92 ?	108 ?	124 JPNZ	140 LP 0C	156 LP 1C	172 LP 2C	188 LP 3C	204 INB	220 ?	236 CAL0C	252 CAL1C
D	13 SBN	29 SLW	45 JRM	61 ?	77 NOPW	93 OUTA	109 ?	125 JPNC	141 LP 0D	157 LP 1D	173 LP 2D	189 LP 3D	205 ?	221 OUTB	237 CAL0D	253 CAL1D
E	14 ADM	30 FILM	46 ?	62 ?	78 WAIT	94 ?	110 ?	126 JPZ	142 LP 0E	158 LP 1E	174 LP 2E	190 LP 3E	206 NOPT	222 ?	238 CAL0E	254 CAL1E
F	15 SBW	31 FILD	47 LOOP	63 ?	79 WAITI	95 OUTF	11 ?	127 JPC	143 LP 0F	159 LP 1F	175 LP 2F	191 LP 3F	207 ?	223 OUTC	239 CAL0F	255 CAL1F