

## MICROPROCESSEUR 6809

Le microprocesseur 6809 conçu par «MOTOROLA» est un circuit intégré de technologie MOS canal N comportant 40 broches compatibles TTL.

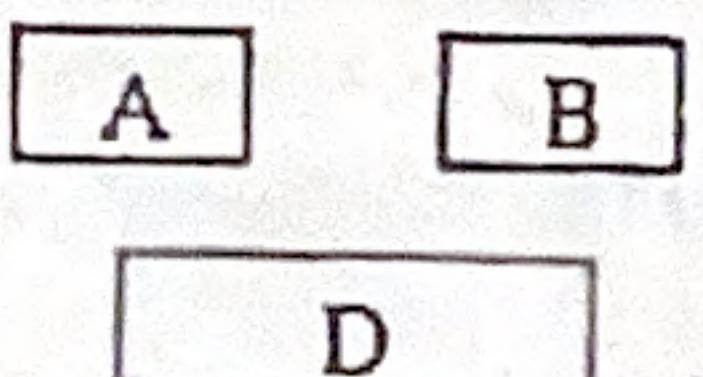
Bien qu'utilisant un bus de données de 8 bits ce microprocesseur permet certaines opérations arithmétiques en 16 bits telles que : addition, soustraction et multiplication 8 bits = 8 bits.

## L - REGISTRES INTERNES

Tous les registres internes 8 bits peuvent s'échanger, de même que tous les registres 16 bits.

### Accumulateurs A, B, D :

les registres A et B sont deux accumulateurs 8 bits pouvant intervenir séparément dans des opérations 8 bits ou pouvant être concaténés en un registre 16 bits lequel sera alors désigné sous le nom de «accumulateur D».



Les accumulateurs peuvent être chargés par les instructions «LOAD» désignées par les mnémoniques LDA (Load A), LDB (Load B), LDD (Load D). Cette dernière permet de charger par une seule instruction les deux registres A et B.

Les accumulateurs peuvent être transférés en mémoire par les instructions «STORE»:  
STA, STB, STD.

## Registres d'index X et Y:

ces registres de 16 bits sont utilisés pour l'adressage indexé. Ils pourront également servir de registre de travail 16 bits. Ils sont chargeables par l'instruction LOAD et transférables en mémoire par STORE.

### Pointeurs de pile U et S :

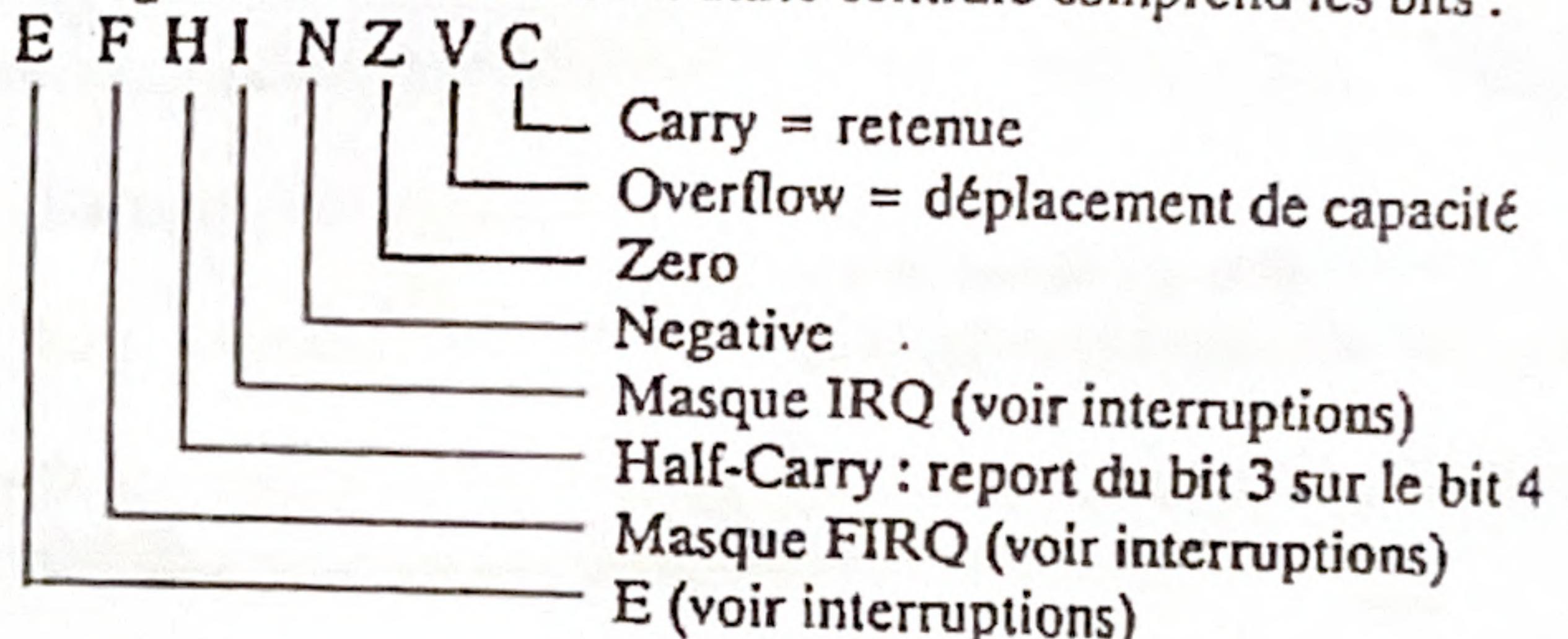
le pointeur de pile U est à disposition de l'utilisateur qui peut l'employer comme pointeur de pile ou comme registre d'index de la même façon que X et Y.

Compteur Ordinal «CO» ou «PC» (Program Counter) :

**Le compteur ordinal** permet au processeur de pointer l'adresse de la prochaine instruction à exécuter. Ce pointeur incrémenté automatiquement après la lecture de chaque octet d'instruction. Il est toutefois modifié par une instruction de saut ou de branchement ou par une interruption, ou encore par le transfert d'un registre 16 bits dans ce compteur ordinal.

**Registre d'état «CCR» (Condition Code Register) :**

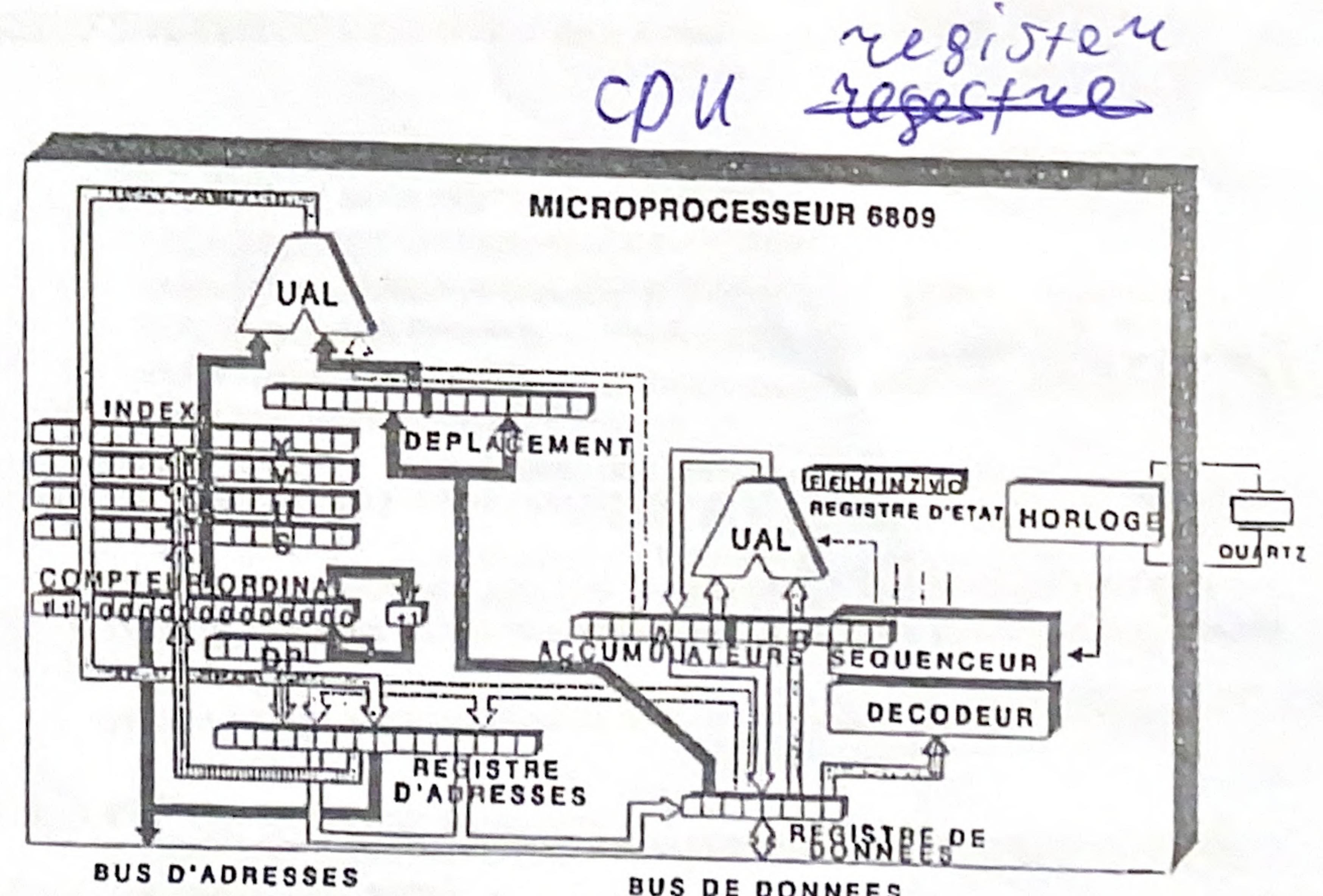
ce registre d'état associé à l'unité centrale comprend les bits :



Registre de page directe «DP» (Direct Page Register) :

Ce registre est utilisé pour former l'adresse en mode direct. Pour le charger il faut utiliser une instruction de transfert.

**LDA #\$3C** permet de mettre \$3C dans l'accumulateur A puis  
**TFR A,DP** dans DP



## 2 - UNITE ARITHMETIQUE ET LOGIQUE DU 6809

L'unité arithmétique et logique du microprocesseur 6809 permet des opérations 8 bits :

Arithmétiques :

Addition A + Mémoire  $\rightarrow$  A      ou B + Mémoire  $\rightarrow$  B

Soustraction

Multiplication A \* B  $\rightarrow$  D

Négation -A ou -B

Logiques :

Décalage

Rotation

Et

OU inclusif

OU exclusif

Cette unité arithmétique et logique permet également l'addition et la soustraction 16 bits

D + mémoire  $\rightarrow$  D

D - mémoire  $\rightarrow$  D

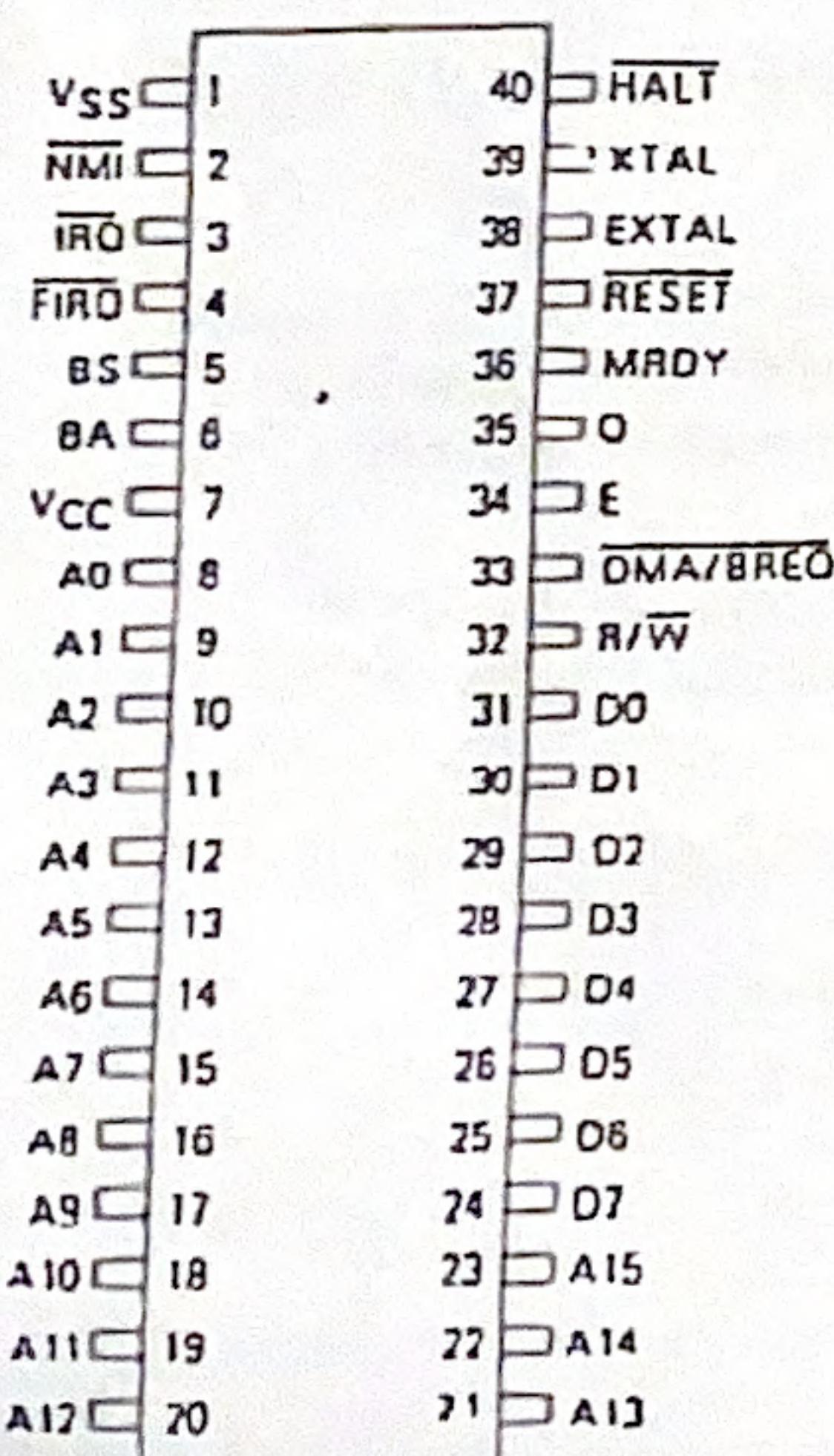
Il sera possible d'effectuer des additions telles que :

ABX                  X + B  $\rightarrow$  X

LEAX A,X            X + A  $\rightarrow$  X

LEAX \$1000,X        X + \$1000  $\rightarrow$  X

## 3 - BROCHAGE DU 6809 (version horloge extérieure)



Le circuit est alimenté par une alimentation de 5 volts appliquée sur la borne VCC(7) tandis que la borne VSS est à la masse.

Nous trouvons 16 lignes d'adresses et 8 lignes de données.

Les bornes XTAL (39) et EXTAL (38) sont à connecter à un quartz qui déterminera la fréquence de l'horloge interne au processeur. Il faut un quartz de fréquence 4 fois celle désirée pour le processeur.

L'horloge interne génère extérieurement les deux signaux Q (35) et E (33) en quadrature de phase. Ces signaux permettent de synchroniser les dispositifs extérieurs au processeur.

Le processeur crée le signal R/W qui est à 1 lors d'une lecture (Read) et passe à 0 lors d'une écriture (Write).

Les lignes RESET, NMI, IRQ, FIRO sont des lignes d'entrée d'interruption qui permettent de forcer le processeur à effectuer une rupture de séquence (voir chapitre correspondant).

La ligne HALT bloque le fonctionnement du processeur quand elle est à l'état bas.

MRDY tarde le fonctionnement du processeur pour les mémoires lentes.

DMA.BRQ permet le fonctionnement multiprocesseurs. Lorsque cette ligne vient au niveau 0 le processeur libère les bus au plus tôt pour un autre processeur.

Les lignes BA et BS renseignent les dispositifs extérieurs sur l'état du processeur selon le tableau :

BA	BS	état du 6809
0	0	fonctionnement normal
0	1	acquittement d'interruption
1	0	acquittement de synchronisation
1	1	mode halte : le bus est libéré.

*Bus available  
Bus status.*

## 4- MODES D'ADRESSAGES DU 6809

### 4- 1 Adressage implicite ou inhérent

Le code opération comprend implicitement l'adresse de l'opérande qui sera généralement interne au microprocesseur.

### 4- 2 Adressage immédiat

L'opérande est situé immédiatement après le code opération

Ex : LDA #\\$27      # symbolise l'adressage immédiat

mémoire
\$86
\$27

Code opération de LDA #  
Opérande

Après l'exécution l'accumulateur A vaut      Acc A = \\$27

Ex : LDX #\$2506

LDX
\$8E
\$25
\$06

Code opération de LDX #

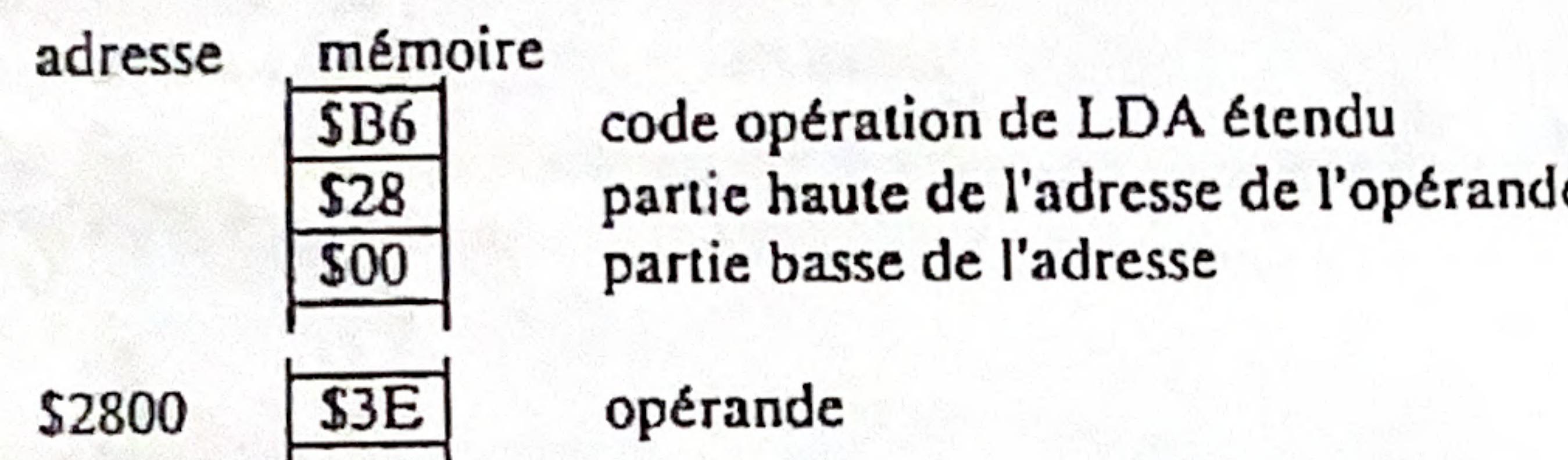
Après l'exécution l'index X vaut      Index X = \$2506

L'adressage immédiat est utilisé pour faire intervenir une constante

#### 4 - 3 Adressage étendu

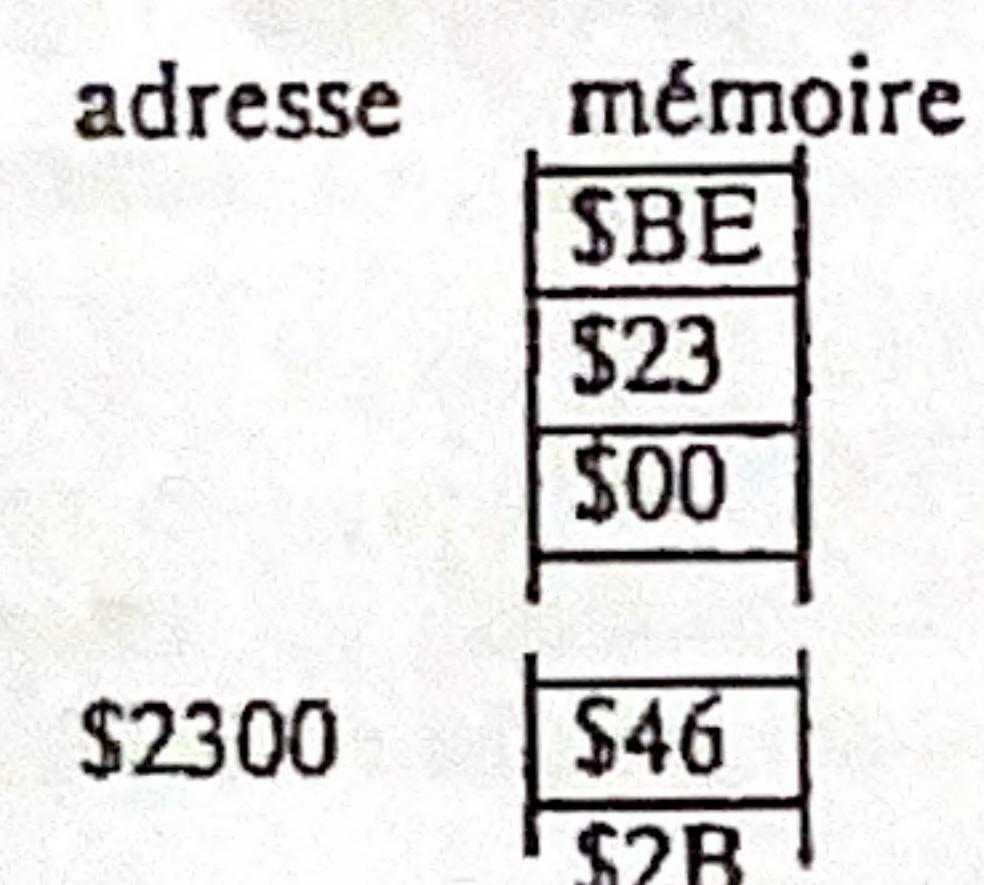
L'adresse de l'opérande suit le code opération.

Ex : LDA > \$2800 *(> symbolise l'adressage étendu optionnel si l'adresse est supérieure à \$FF)*



Après l'exécution l'accumulateur A vaut : Acc A = \$3E

Ex : LDX > \$2300



Après l'exécution l'index X vaut Index X = \$462B

Notons que X étant un registre de 16 bits, il sera chargé par 2 octets successifs, mais seul le premier sera désigné par l'adresse.

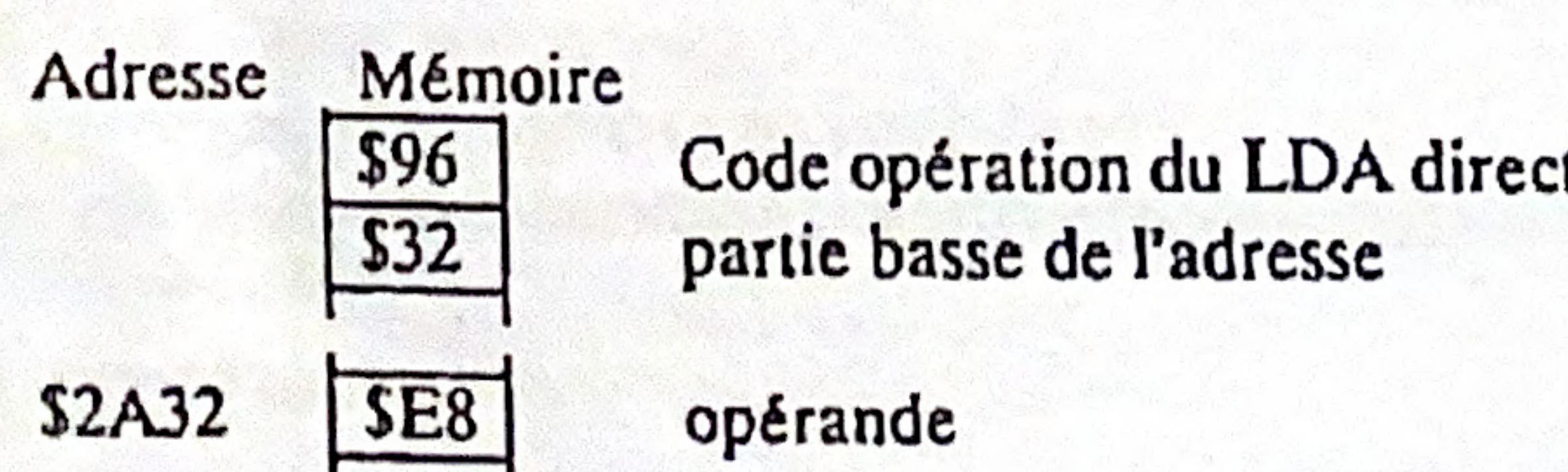
#### 4 - 4 Adressage direct

L'adresse de l'opérande est ainsi constituée :

- la partie basse de l'adresse suit le code opération,
- la partie haute de l'adresse se trouve dans le registre direct de page ou DPR (Direct Page Register) interne au processeur et chargeable par l'instruction TFR A,DP

Ex : LDA < \$32 *(< symbolise l'adressage direct)*

Si avant l'instruction DPR vaut \$2A



Après l'exécution l'accumulateur A vaut : Acc A = \$E8

#### 4 - 5 Adressage indexé

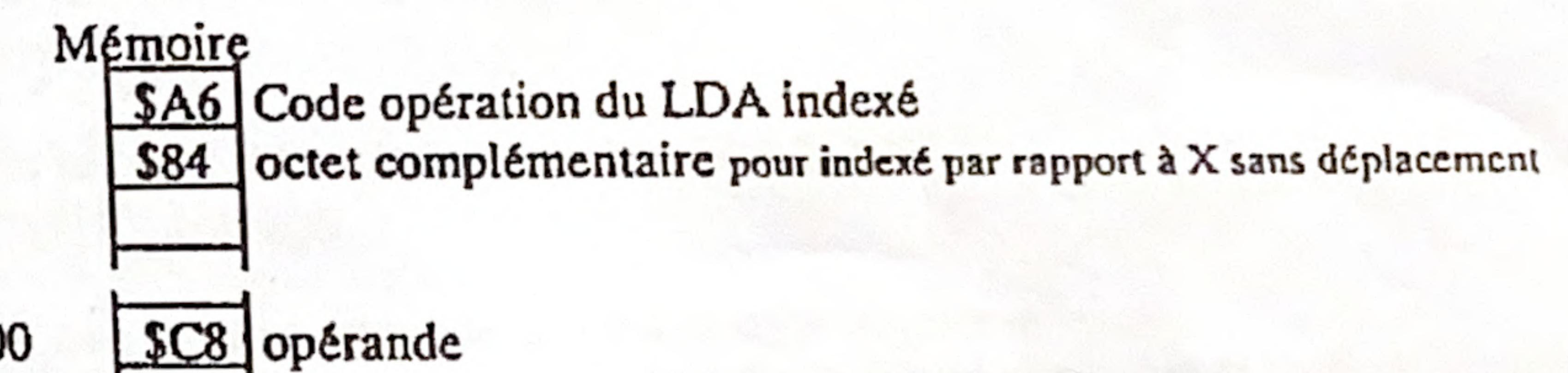
Dans l'adressage indexé l'adresse de l'opérande est obtenue par la somme du registre d'index choisi (X, Y, U ou S) et d'un déplacement («offset») qui suit le code opération. Cette adresse sera appelée «adresse effective». Le code opération est suivi d'un octet complémentaire spécifiant le mode indexé

##### 4 - 5 - 1 Adressage indexé sans déplacement

Dans ce cas le déplacement est nul.

Ex : LDA,X *(,X symbolise l'indexation par rapport à l'index X)*

Si X a été précédemment chargé avec la valeur \$2500 par LDX # \$2500 \$2500 sera l'adresse effective de l'opérande.



Après l'exécution l'accumulateur A vaut : Acc A = \$C8

##### 4 - 5 - 2 Adressage indexé avec auto-incréméntation ou auto-décréméntation

Le déplacement est toujours nul.

On peut avoir un auto-incrément de 1 ou de 2

Si l'adressage indexé est auto-incrémenté l'index incrémenté après être intervenu dans l'adressage. Par contre si l'adressage est auto-décrémenté l'index décrémenté avant être intervenu dans l'adressage.

exp : LDA,X+ X incrémenté de 1 après le chargement de A

LDD,Y++ Y incrémenté de 2 après le chargement de D

LDA,-X X décrémenté de 1 avant le chargement de A

LDD,-Y Y décrémenté de 2 avant le chargement de D

Remarque : l'auto-incrément dispense souvent de l'instruction de type LEAX 1,X

##### 4 - 5 - 3 Adressage indexé avec déplacement

Le déplacement peut être codé sur 5, 8 ou 16 bits. Pour chaque cas l'octet complémentaire est différent (voir tableau)

Ex : LDA \$30,X

Si X a été précédemment chargé avec la valeur \$2500 par LDX #\$2500  
\$2530 sera l'adresse effective de l'opérande.

Adresse Mémoire

\$A6	Code opération du LDA indexé
\$88	Octet complémentaire pour indexé par rapport à X avec déplacement 8 bits
\$30	Déplacement
\$2530	opérande

Après l'exécution l'accumulateur A vaut : Acc A = \$D4

#### 4 - 5 - 4 Adressage indexé dont le déplacement est le contenu d'un accumulateur

Pour ce faire les accumulateurs A, B ou D peuvent être utilisés.

Ex : LDA B,X

l'adresse effective est B+X

#### 4 - 5 - 5 Adressage indexé relatif au compteur ordinal

Le compteur ordinal est utilisé comme index.

L'adresse de l'opérande s'obtient en ajoutant au contenu du compteur ordinal la valeur du déplacement (en 8 ou 16 bits).

Ce type d'adressage permet de créer des programmes non localisés.

#### 4 - 5 - 6 Valeur de l'octet complémentaire en adressage indexé

RR étant fonction de l'index utilisé :

	RR
X	00
Y	01
U	10
S	11

Sans déplacement	1RR00100
Auto-incrément de 1	1RR00000
Auto-incrément de 2	1RR00001
Auto-décrément de 1	1RR00010
Auto-décrément de 2	1RR00010
Déplacement codé sur 5 bits	0RRnnnnn
Déplacement codé sur 8 bits	1RR01000
Déplacement codé sur 16 bits	1RR01001

nnnn étant les 5 bits (-16 à +15)

Déplacement est contenu dans l'accumulateur A	1RR00110
Déplacement est contenu dans l'accumulateur B	1RR00101
Déplacement est contenu dans l'accumulateur D	1RR01011
Relatif au compteur ordinal, 8 bits	10001100
Relatif au compteur ordinal, 16 bits	10001100

#### 4 - 6 Adressage indirect

L'adresse obtenue est l'adresse où se trouve l'adresse de l'opérande codée sur deux octets.

Ex : LDA [\$B800] symbolisant le mode indirect.  
L'adresse de l'opérande se trouve en \$B800.

Ex : LDA [\$30,X]

L'adresse de l'opérande se trouve en \$30 + X.

### 5 - SAUTS ET BRANCHEMENTS

#### 5 - 1 Sauts

Un saut est une rupture de séquence dans laquelle le compteur ordinal prend la valeur de l'adresse effective qui suit le code opération.

Ex : JMP >\$F800  
charge le compteur ordinal avec la valeur \$F800.

Ex : JMP ,X  
charge le compteur ordinal avec la valeur 16 bits pointée par X.

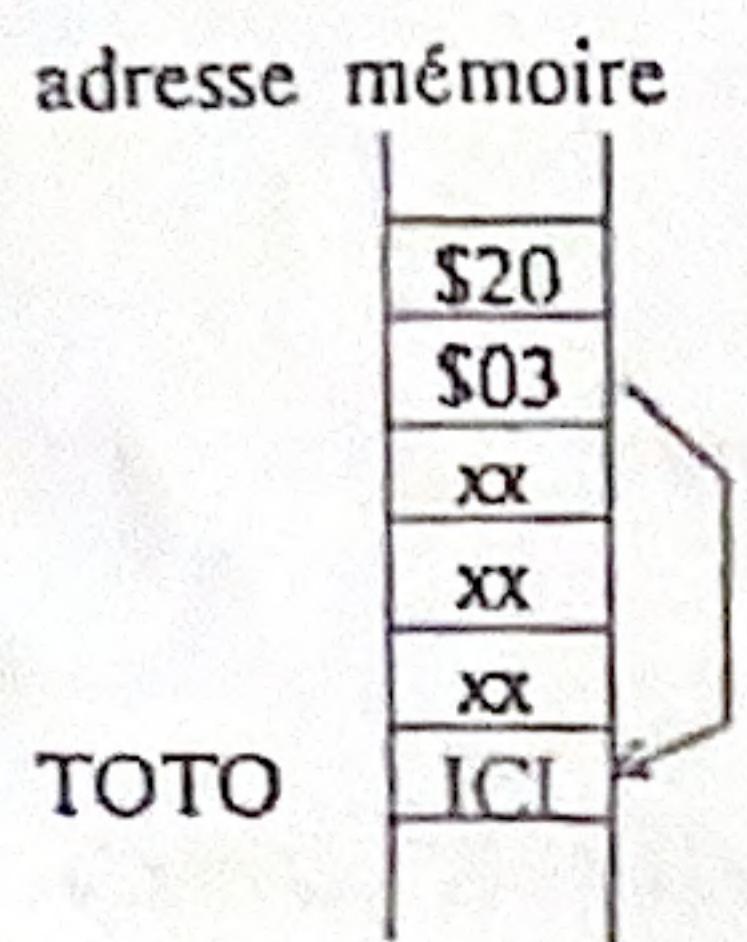
#### 5 - 2 Branchements inconditionnels

Un branchement est également une rupture de séquence mais dans ce cas le compteur ordinal s'augmente de la valeur signée du déplacement qui suit le code opération.

Le déplacement peut être codé sur 8 bits signés (branchement court) ou sur 16 bits (branchement long).

Sur 8 bits nous obtenons un branchement à +127 adresses plus loin ou à -128 adresses (en arrière). Le code opération est \$20. Le mnémonique est BRA.

EX : BRA TOTO TOTO étant une étiquette.



Sur 16 bits nous obtenons un branchement possible dans tout l'espace mémoire. Le code opération est \$16. Le mnémonique est LBRA.

### 5 -3 Branchements conditionnels

Le calcul du branchement sera identique au précédent mais son exécution sera conditionnelle :

- si la condition est vérifiée le branchement est effectué,
- si la condition n'est pas vérifiée le branchement n'est pas effectué.

Les conditions sont relatives aux bits Z, N, V, C du registre d'état.

Z = 1 si résultat ou chargement nul,

N = 1 si résultat ou chargement négatif (bit 7 = 1),

V = 1 si dépassement de capacité

C = 1 si retenue.

Ex : utilisation de BEQ (Branch If Equal)

LDA	\$B800	
CMPA	#\$41	soustrait A - \$41 et positionne Z = 1 si le résultat est nul, donc si A = \$41
BEQ	TOTO	branche si Z = 1, donc si A = \$41

*Branch*

### 6 - PILES ET POINTEURS DE PILE

Une pile est une zone de mémoire destinée à la sauvegarde temporaire d'informations.

Le dernier octet entré en sera le premier sorti. (Last Input First Output)

Le sommet de la pile sera pointé par le pointeur de pile (Stack Pointer) qui décrémente avant chaque sauvegarde d'octet et qui incrémente après sa récupération.

Le 6809 possède deux pointeurs de pile : le registre S (pointeur système) et le registre U (pointeur utilisateur).

Ces pointeurs doivent être définis par LDS ou LDU

Ex : LDS #SA078

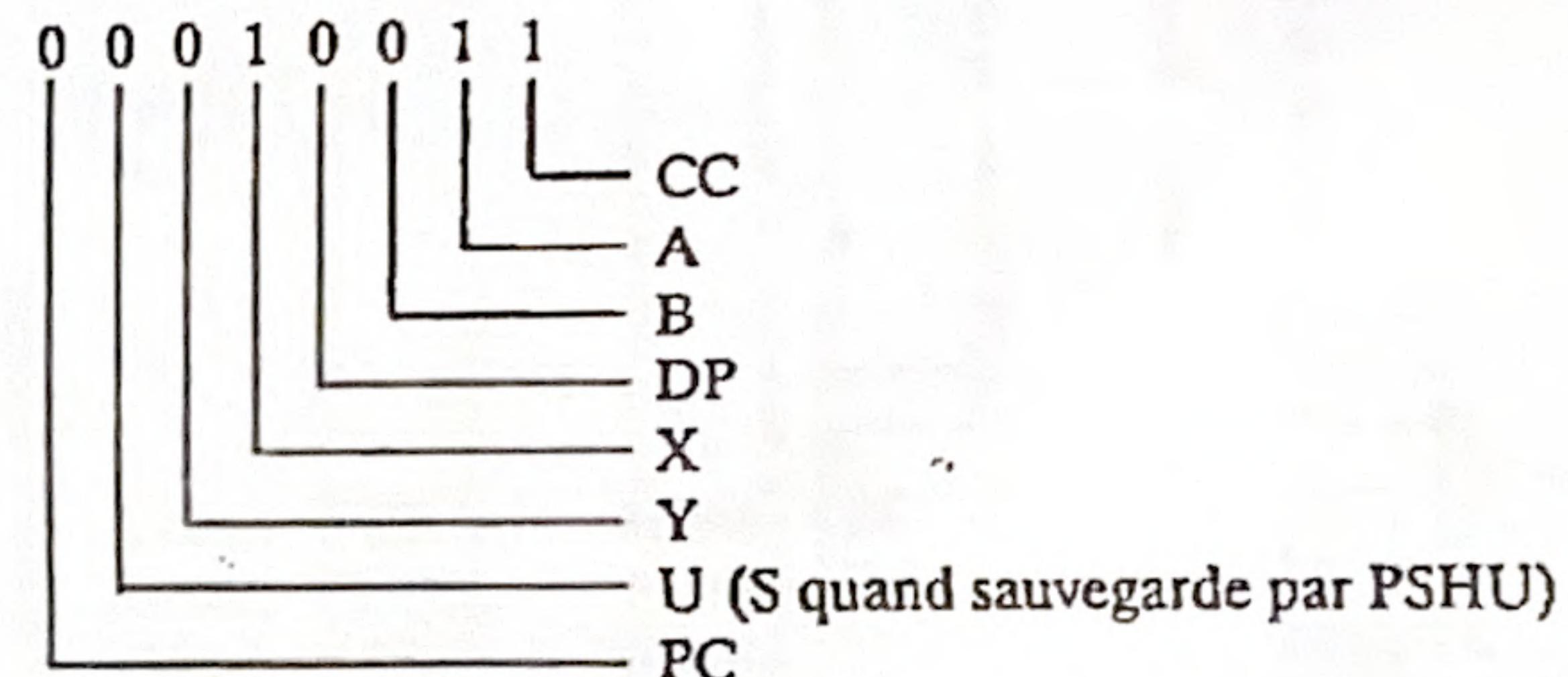
### 6 - 1 Sauvegarde en pile par PUSH

Pour sauvegarder dans la pile S un registre interne au 6809 on utilise l'instruction PSHS (PSHU pour la pile U)

Ex : PSHS X,A,CC

L'ordre des arguments n'importe peu :

Le code de PSHS est suivi de l'octet complémentaire



S décrémente avant la sauvegarde de chaque octet

Si S = \$A078

X = \$25C4

A = \$12

CC = \$81

Adresse Mémoire

\$A071	xxx
\$A072	xxx
\$A073	xxx
\$A074	xxx
\$A075	xxx
\$A076	xxx
\$A077	xxx
S = \$A078	\$00

Adresse Mémoire

\$A071	xxx
\$A072	xxx
\$A073	xxx
\$A074	\$81 CC
\$A075	\$12 A
\$A076	\$25 X haut
\$A077	\$C4 X bas
\$A078	\$00

pile  
vide  
avant  
sauvegarde

pile  
après  
sauvegarde

### 6 - 2 récupération par PULL

Les instructions PULS et PULU permettent la récupération d'un octet sauvegardé.

Ex : PULS X,A,CC récupère les registres précédemment sauvegardés par PSHS X,A,CC

## 7 - SOUS-PROGRAMMES

Le sous-programme est un segment de programme destiné à être appelé plusieurs fois par un programme que nous appellerons programme principal.

Le sous-programme est appelé par une instruction de saut ou de branchements à sous-programme. Ces sauts (JSR) et branchements (BSR) (LBSR) diffèrent de ceux précédemment vus par le fait qu'ils entraînent une sauvegarde automatique dans la pile système, du compteur ordinal ; c'est à dire de l'adresse de retour.

Exécution de «Jump To Subroutine» (JSR) :

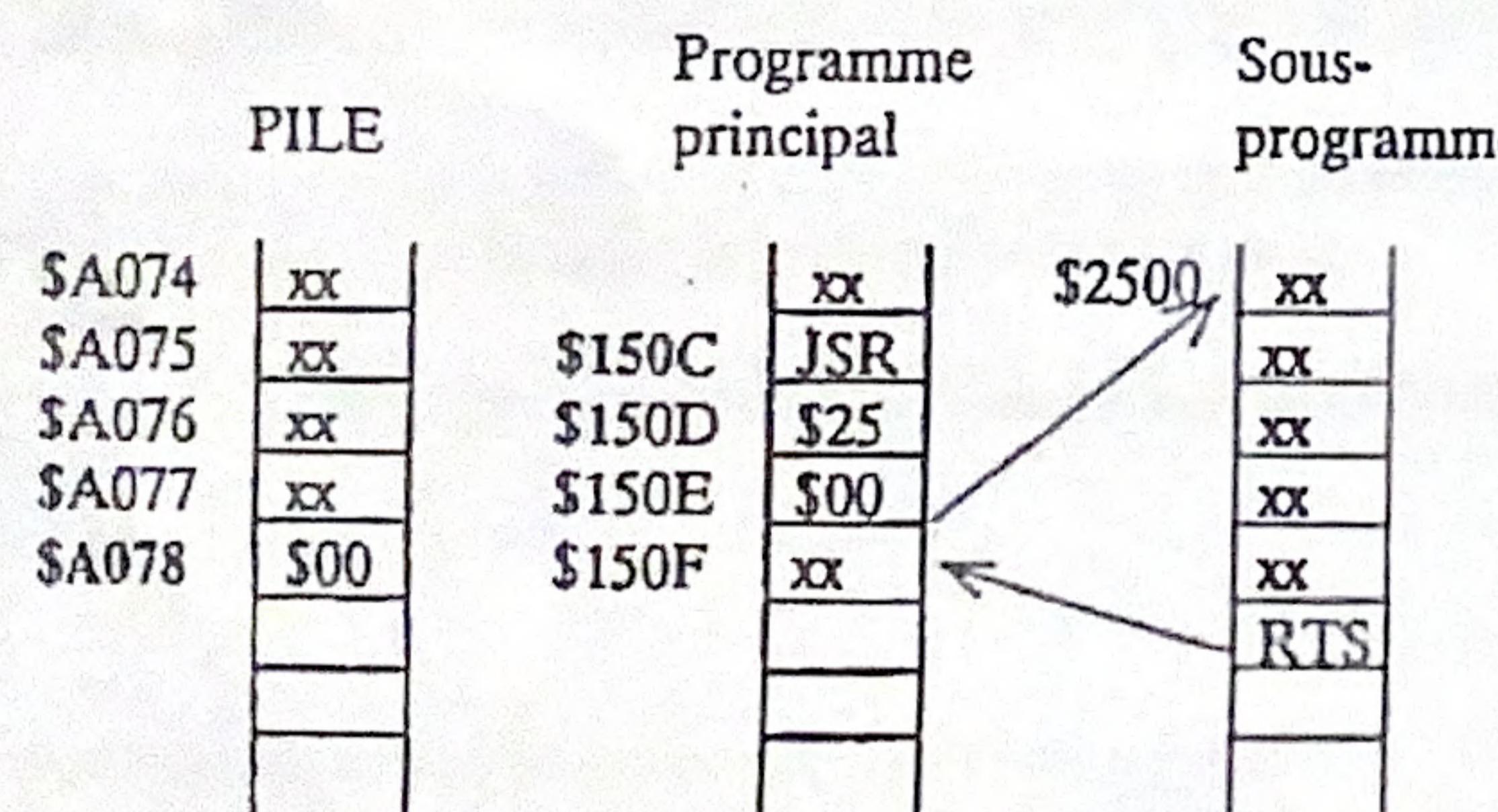
A la lecture de l'instruction  $JSR > \$2500$  le microprocesseur sauvegarde le compteur ordinal dans la pile par deux octets successifs (octet le moins significatif puis octet le plus significatif). Simultanément le pointeur de pile décrémente de deux. Après cela le compteur ordinal prend la valeur  $\$2500$ , entraînant le saut à cette adresse.

Le sous-programme doit terminer par l'instruction RTS (Return From Subroutine) laquelle force le processeur à récupérer dans la pile la valeur du compteur ordinal précédemment sauvegardée.

Notons que RTS est équivalent à PULS PC

Si avant la lecture de l'instruction  $JSR > \$2500$  les registres PC et S valent :

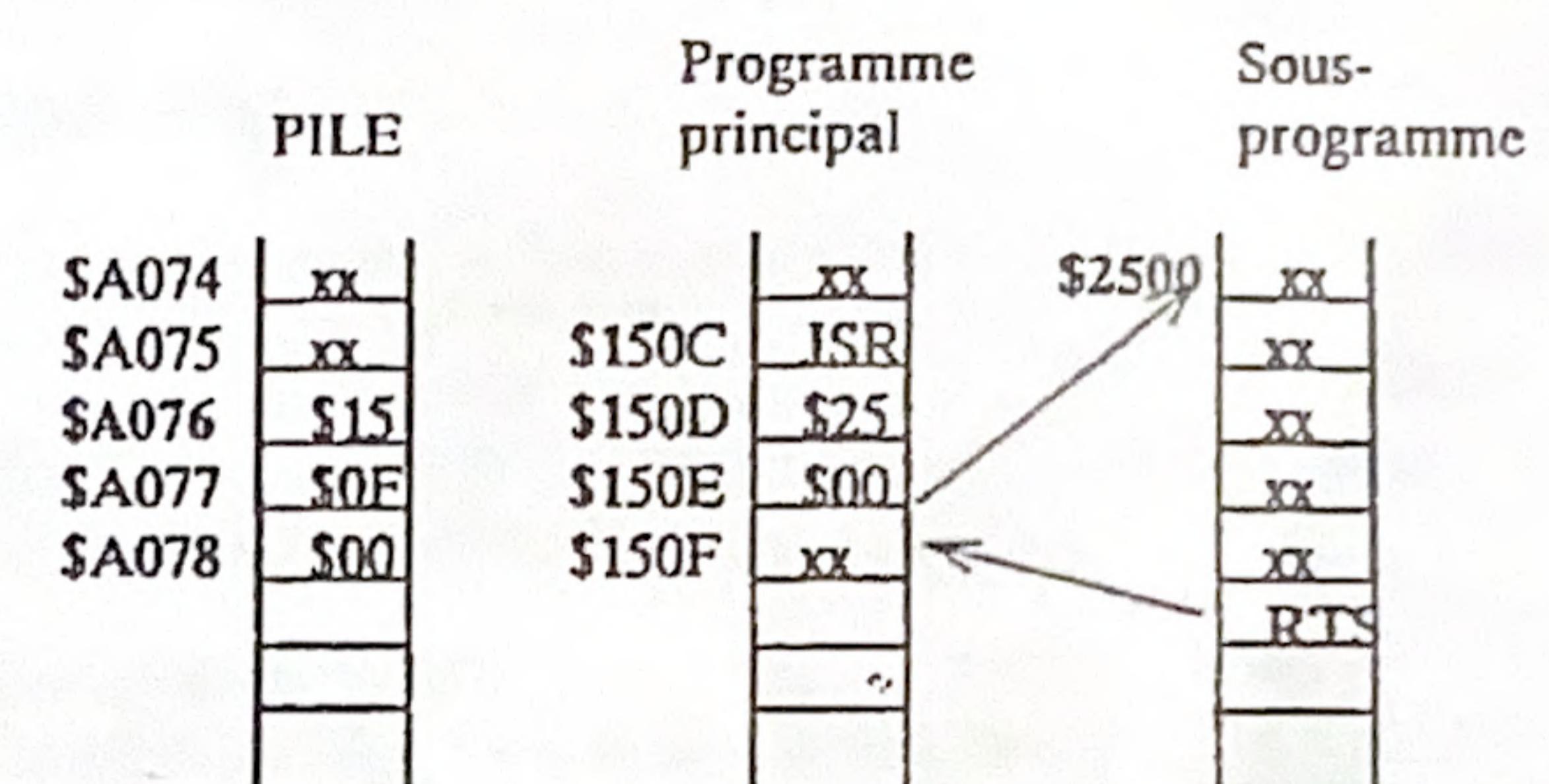
PC = \$150C  
S = \$A078



ETAT AVANT LE SAUT

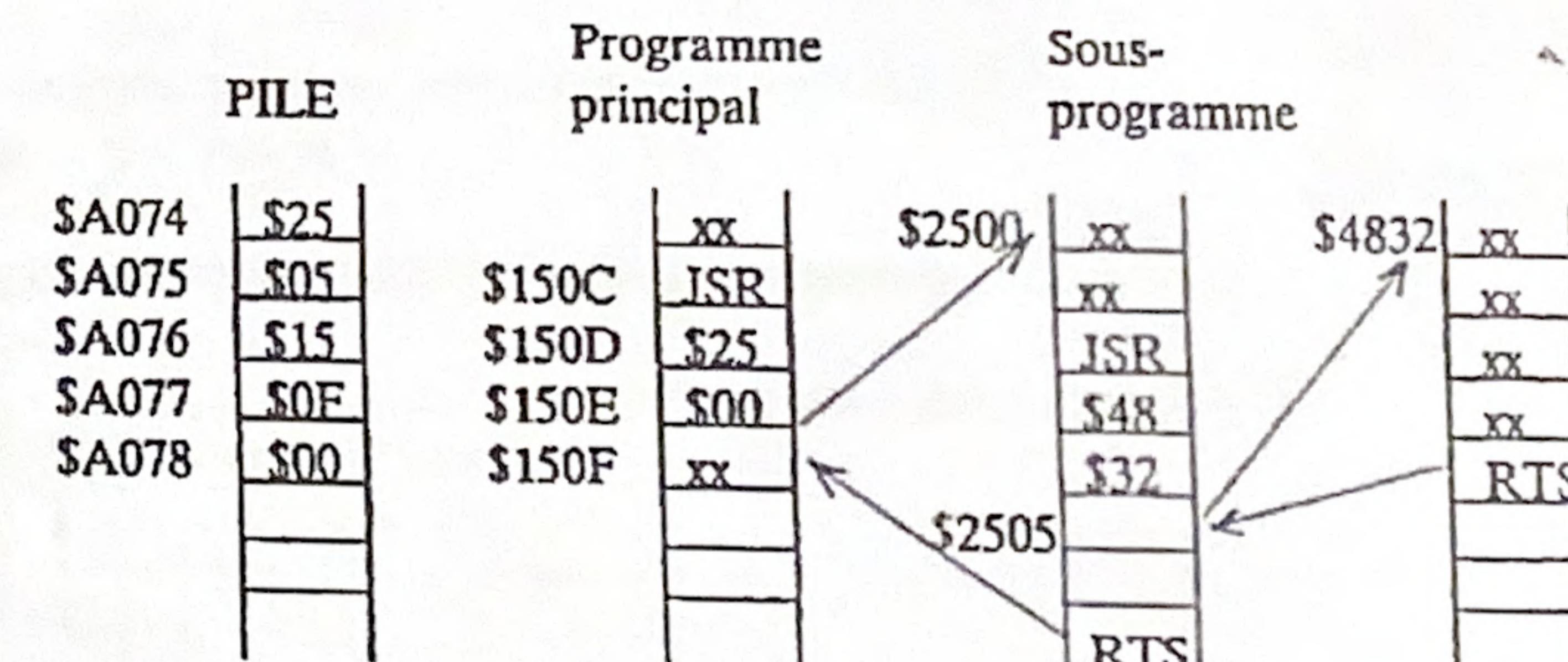
Après le saut :

PC = \$2500  
S = \$A076



ETAT APRES LE SAUT

Un sous-programme peut lui-même appeler un autre sous-programme. Nous avons alors des sous-programmes imbriqués.



## INSTRUCTIONS 6809

OPERATION	Mnémonique Instruction	Modes d'adressage du 6809										Indicateurs 5 3 2 1 0	
		INHERENT		DIRECT		EXTENDED		IMMEDIAT		INDEXE			
		OP	N	OP	N	OP	N	OP	N	OP	N		
Addition de l'accumulateur B & X (non signé)	ABX	3A	3	1								B + X - X	
Addition du contenu mémoire à l'accumulateur, avec retenue	ADCA	99	4	2	B9	5	3	B9	2	2	A9 <sub>1+2+</sub>	A + M + C - A	
	ADCB	D9	4	2	F9	5	3	C9	2	2	E9 <sub>1+2+</sub>	B + M + C - B	
Addition du contenu mémoire à l'accumulateur	ADDA	98	4	2	B8	5	3	B8	2	2	AB <sub>1+2+</sub>	A + M - A	
	ADDB	D8	4	2	FB	5	3	CB	2	2	EB <sub>1+2+</sub>	B + M - B	
	ADDO	D3	6	2	F3	7	3	C3	4	3	E3 <sub>1+2+</sub>	D + M + 1 - D	
ET logique entre mémoire et l'accumulateur	ANDA	94	4	2	B4	5	3	B4	2	2	A4 <sub>1+2+</sub>	A . M - A	
• ET logique avec le registre codes condition	ANDB	D4	4	2	F4	5	3	C4	2	2	E4 <sub>1+2+</sub>	B . M - B	
	ANDOC							1C	3	2		CC MM - CC	
Décalage arithmétique à gauche du contenu mémoire ou accumulateur	ASIA	48	2	1								A	
	ASLB	58	2	1	08	6	2	78	7	3	68 <sub>1+2+</sub>	B	
	ASL											M	
Décalage arithmétique à droite du contenu mémoire ou accumulateur	ASRA	47	2	1								A	
	ASRB	57	2	1	07	6	2	77	7	3	67 <sub>1+2-</sub>	B	
	ASR											M	
Branchement si pas de retenue	BCS										24	Si C = 0	
	LBCS										10 <sub>(5)</sub> 4		
Branchement si retenue											24		
Branchement si égal	BEO										25	Si C = 1	
	LBEQ										10 <sub>(5)</sub> 4		
Branchement si supérieur ou égal (signé)	BGE										25	Si z = 1	
	LBGE										10 <sub>(5)</sub> 4		
Branchement si supérieur (signé)	BGT										27	Si ≥ Zéro	
	LBGT										10 <sub>(5)</sub> 4		
Branchement si supérieur (non signé)	BHI										27	Si > Zéro	
	LBHI										10 <sub>(5)</sub> 4		
Branchement si supérieur ou égal (non signé)	BHS										27	Si > Zéro	
	LBHS										10 <sub>(5)</sub> 4		
Test de bit mémoire avec l'accumulateur	BITA	BS	4	2	BS	5	3	BS	2	2	AS4 <sub>1+2+</sub>	(M A A)	
	BITB	DS	4	2	FS	5	3	CS	2	2	ES4 <sub>1+2+</sub>	(M A B)	
Branchement si intérieur ou égal (signé)	BLE										2F	Si ≤ Zéro	
	LBLE										10 <sub>(5)</sub> 4		
Branchement si intérieur (non signé)	BLO										2F	Si ≤ Zéro	
	LBLO										10 <sub>(5)</sub> 4		
Branchement si intérieur ou égal (non signé)	BLS										25	Si < Zéro	
	LBLS										10 <sub>(5)</sub> 4		
Branchement si intérieur (signé)	BLT										25	Si < Zéro	
	LBLT										10 <sub>(5)</sub> 4		
Branchement si négatif	BMI										25	Si < Zéro	
	LBMI										10 <sub>(5)</sub> 4		
Branchement si non égal	BNE										25	Si Z = 0	
	LBNE										10 <sub>(5)</sub> 4		
Branchement si positif	BPL										25	Si Z = 0	
	LBPL										10 <sub>(5)</sub> 4		
Branchement inconditionnel	BRA										25	Si Z = 0	
	LBRA										10 <sub>(5)</sub> 4		

Tableau 1

## INSTRUCTIONS 6809

OPÉRATION	Mnémonique Instruction	Modes d'adressage du 6809												Indicateurs	
		INHERENT		DIRECT		EXTENDED		IMMÉDIAT		INDEXE		RELATIF			
		OP	N	OP	N	OP	N	OP	N	OP	N	OP	N		
Non branchement	BRN LBRN											21	3	2	
												10	5	4	
												21			
Branchement à un sous-programme	BSR LBSR											50	7	2	
												17	9	3	
Branchement si pas de débordement = 0	BVC LBVC											28	3	2	
												10	5(6)	4	
Branchement si débordement = 1	BVS LBVS											28	3	2	
												10	5(6)	4	
Mise à zéro du contenu de l'accumulateur ou de la mémoire	CLRA CLRB CLR	4F	2	1	OF	6	2	7F	7	3		6F5-2-			
		SF	2	1											
Comparaison du contenu mémoire avec l'accumulateur	CMPA CMPB CMPC	91	4	2	B1	5	3	B1	2	2	A1	4+2+			
		D1	4	2	F1	5	3	C1	2	2	E1	4+2+			
		10	7	3	10	B	4	10	5	4	10	7+3+			
		93			B3			83			A3				
Comparaison mémoire avec pointeur de pile	CMPS	11	7	3	11	8	4	11	5	4	11	7+3+			
	CMPU	9C	7	3	BC	8	4	11	5	4	AC				
	CMPX	93	6	2	BC	7	3	BC	4	3	AC	6+2+			
	CMPY	10	7	3	10	B	4	10	5	4	10	7+3+			
		9C			BC			BC			AC				
Comparaison mémoire avec registre index	COMA COMB COM	43	2	1	03	6	2	73	7	3		635+2+			
		53	2	1											
Complément à deux de l'accumulateur ou du contenu mémoire	CWAJ	3C	20	2									CC	XX→CC	
« ET » logique avec le registre codes condition puis attente d'interruption	DAA	19	2	1											
Ajustement décimal de l'accumulateur A	DECA DECB DEC	4A	2	1	0A	6	2	7A	7	3		6A5-2-			
Décrémentation du contenu mémoire ou de l'accumulateur	DECA DECB DEC	5A	2	1	0A	6	2	7A	7	3		6A5-2-			
« OU » exclusif du contenu mémoire avec l'accumulateur	FORA FORB	98	4	2	B8	5	3	B8	2	2	AB4+2+				
		D8	4	2	F8	5	3	C8	2	2	EB4+2+				
Echange de R1 avec R2 (R1, R2 = A, B, CC, DP)	EXG	R1, R2	1E	7	2								R1→R2 ②		
Incrémentation du contenu mémoire ou de l'accumulateur	INCA INC	4C	2	1	0C	6	2	7C	7	3		EC5+2+			
	INCB	5C	2	1											
Saut inconditionnel	JMP	0E	3	2	7E	4	3				6EB+2+				
Saut à un sous-programme	JSR	90	7	2	BD	8	3				AD7+2+				
Chargement de l'accumulateur avec le contenu mémoire	LDA	96	4	2	B6	5	3	B6	2	2	AB4+2+				
Chargement du pointeur de pile avec le contenu mémoire	LDB	D6	4	2	F6	5	3	C6	2	2	E64+2+				
Chargement du registre index avec le contenu mémoire	LDO	DC	5	2	FC	6	3	CC	3	3	EC5+2+				
	LDS	10	6	3	10	7	4	10	4	4	106+3+				
Chargement de l'adresse effective dans le pointeur de pile	LDU	DE	5	2	FE	6	3	CE	3	3	EE5+2+				
Chargement de l'adresse effective dans le registre index	LDX	9E	5	2	BE	6	3	BE	3	3	AE5+2+				
	LDY	10	6	3	10	7	4	10	4	4	106+3+				
		9E			BE			BE			AE				
Décalage logique à gauche du contenu mémoire ou de l'accumulateur	LEAS										324+2+				
	LEAU										334+2+				
	LEAX										304+2+				
	LEAY										314+2+				
Décalage logique à droite du contenu mémoire ou de l'accumulateur	LSLA	48	2	1									A		
	LSLB	58	2	1	08	6	2	78	7	3		686+2+			
	LSL												B		
													M		
Multiplication non signée	LSRA LSRB LSR	44	2	1	04	6	2	74	7	3		646+2+			
		54	2	1									A		
													B		
													M		
Complément à un du contenu mémoire ou de l'accumulateur	MUL	3D	11	1									A×B→D		
	NEGA	40	2	1	00	6	2	70	7	3		606+2+			
	NEG8	50	2	1									A+1+A		
	NEG												B+1+B		
													M+1+M		

Tableau 1 (suite)

# INSTRUCTIONS 6809

OPÉRATION	Minémomique Instruction	Modes d'adressage du 6809												Indicateurs S I 3 2 1 0	
		INHERENT		DIRECT		EXTENDUE		IMMÉDIAT		INDEX		RELATIF			
		OP	~	N	OP	~	N	OP	~	N	OP	~	N		
Non opération	NOP	12	2	1											
• OU = logique mémoire et accumulateur	DRA				9A	4	2	BA	5	3	BA	2	2	AAX+2+	
• OU = logique avec le registre codes condition	ORB				DA	4	2	FA	5	3	CA	2	2	EAN+2+	
Emplacement de tout(s) registre(s) (sauf S) sur la pile S	OROC							1A	3	2				CC IMM→CC	
Emplacement de tout(s) registre(s) (sauf U) sur la pile U	PSHS	34	5+	2											
Déplacement de tout(s) registre(s) de la pile S	PSHU	35	5+	2											
Déplacement de tout(s) registre(s) (sauf U) de la pile U	PULS	35	5+	2											
Déplacement de tout(s) registre(s) de la pile U	PULU	37	5+	2											
Décalage circulaire à gauche du contenu mémoire ou de l'accumulateur	ROLA	49	2	1											
	ROLB	50	2	1											
	ROL				09	6	2	79	7	3					
Décalage circulaire à droite du contenu mémoire ou de l'accumulateur	RORA	46	2	1											
	RORB	56	2	1											
	ROR				06	6	2	76	7	3					
Retour d'interruption	RTI	38	5/15	1											
Retour de sous-programme	RTS	39	5	1											
Soustraction du contenu mémoire de l'accumulateur	SBCA				92	4	2	B2	5	3	B2	2	2	A2X+2+	
	SBCB				D2	4	2	F2	5	3	C2	2	2	E2X+2+	
Extension de signe de l'accumulateur B à l'accumulateur A	SEX	1D	2	1											
Mise en mémoire du contenu de l'accumulateur	STA				97	4	2	B7	5	3				A→M	
	STB				D7	4	2	F7	5	3				B→M	
	STD				DD	5	2	FD	6	3				D→MM+1	
Mise en mémoire du pointeur de pile	STS				10	6	3	10	7	4				S→MM+1	
Mise en mémoire du registre index					DF			FF							
	STU				DF	5	2	FF	6	3				U→MM+1	
	STX				9F	5	2	BF	6	3				X→MM+1	
	STY				10	6	3	10	7	4				Y→MM+1	
Soustraction du contenu mémoire de l'accumulateur	SUBA				90	4	2	B0	5	3	B0	2	2	A04-2-	
	SUBB				DO	4	2	FO	5	3	CO	2	2	E04+2+	
	SUBD				93	6	2	B3	7	3	B3	4	3	A36+2+	
Interruption programmée	SW1	6	19	1											
	SW12	10	20	2											
	SW13	11	20	2											
		3F													
Synchronisation avec la ligne d'interruption	SYNC	213	2	1											
Transfer de R1 à R2	TFR R1, R2	1F	7	2										R1→R2 (2)	
Test mémoire ou accumulateur	TSTA	4D	2	1											
	TSTB	50	2	1											
	TST				DD	6	2	7D	7	3					
Tableau 1 (suite)															

**Légende**

OP code hexadécimal de l'opération	Z Zéro (octet)
~ nombre de cycles - horloge	V Dépassement
N nombre d'octets	C report du bit 7
+ plus arithmétique	I testé et mis à 1 si condition vraie, sinon mis à zéro
- moins arithmétique	• non affecté
×	CC registre d'indicateur d'état
M complément de M	:
→ transfert dans	concaténation
H demi-retenu du bit 3	V OU logique
N négatif (bit de signe)	Y OU logique Ex
	A ET logique

SOURCE	DESTINATION
0000 → D, A, B1	1000 → A
0001 → X	1001 → B
0010 → Y	1010 → CCR
0011 → Z	1011 → DPA
0100 → S	
0101 → PC	

Code registre

## Notes :

1. Dans le tableau sont donnés le nombre de cycles et d'octets de base. Pour déterminer le nombre total de cycles et d'octets ajouter les valeurs du tableau 2 des types d'adressage indexé.
2. R1 et R2 peuvent être une paire de registres 8 bits : A, B, CC, DP, ou 16 bits : X, Y, U, S, D, PC.  
Il faut alors ajouter au code-instruction un post-octet selon le code-registre

Tableau 1 (suite)

B3.5

## INSTRUCTIONS 6809

Type	Forms	Non Indirect				Indirect			
		Assembler Form	Postbyte OP Code	+	+	Assembler Form	Postbyte OP Code	+	-
Constant Offset From R (2's Complement Offsets)	No Offset	,R	1RR00100	0	0	[,R]	1RR10100	3	0
	5 Bit Offset	n, R	0RRnnnnn	1	0	defaults to 8-bit			
	8 Bit Offset	n, R	1RR01000	1	1	[n, R]	1RR11000	4	1
	16 Bit Offset	n, R	1RR01001	4	2	[n, R]	1RR11001	7	2
Accumulator Offset From R (2's Complement Offsets)	A Register Offset	A, R	1RR00110	1	0	[A, R]	1RR10110	4	0
	B Register Offset	B, R	1RR00101	1	0	[B, R]	1RR10101	4	0
	D Register Offset	D, R	1RR01011	4	0	[D, R]	1RR11011	7	0
Auto Increment/Decrement R	Increment By 1	,R +	1RR00000	2	0	not allowed			
	Increment By 2	,R + +	1RR00001	3	0	[,R + +]	1RR10001	6	0
	Decrement By 1	,- R	1RR00010	2	0	not allowed			
	Decrement By 2	,- - R	1RR00011	3	0	[, - - R]	1RR10011	6	0
Constant Offset From PC (2's Complement Offsets)	8 Bit Offset	n, PCR	1xx01100	1	1	[n, PCR]	1xx11100	4	1
	16 Bit Offset	n, PCR	1xx01101	5	2	[n, PCR]	1xx11101	8	2
Extended Indirect	16 Bit Address	-	-	-	-	(n)	10011111	5	2

R = X, Y, U or S  
x = Don't Care

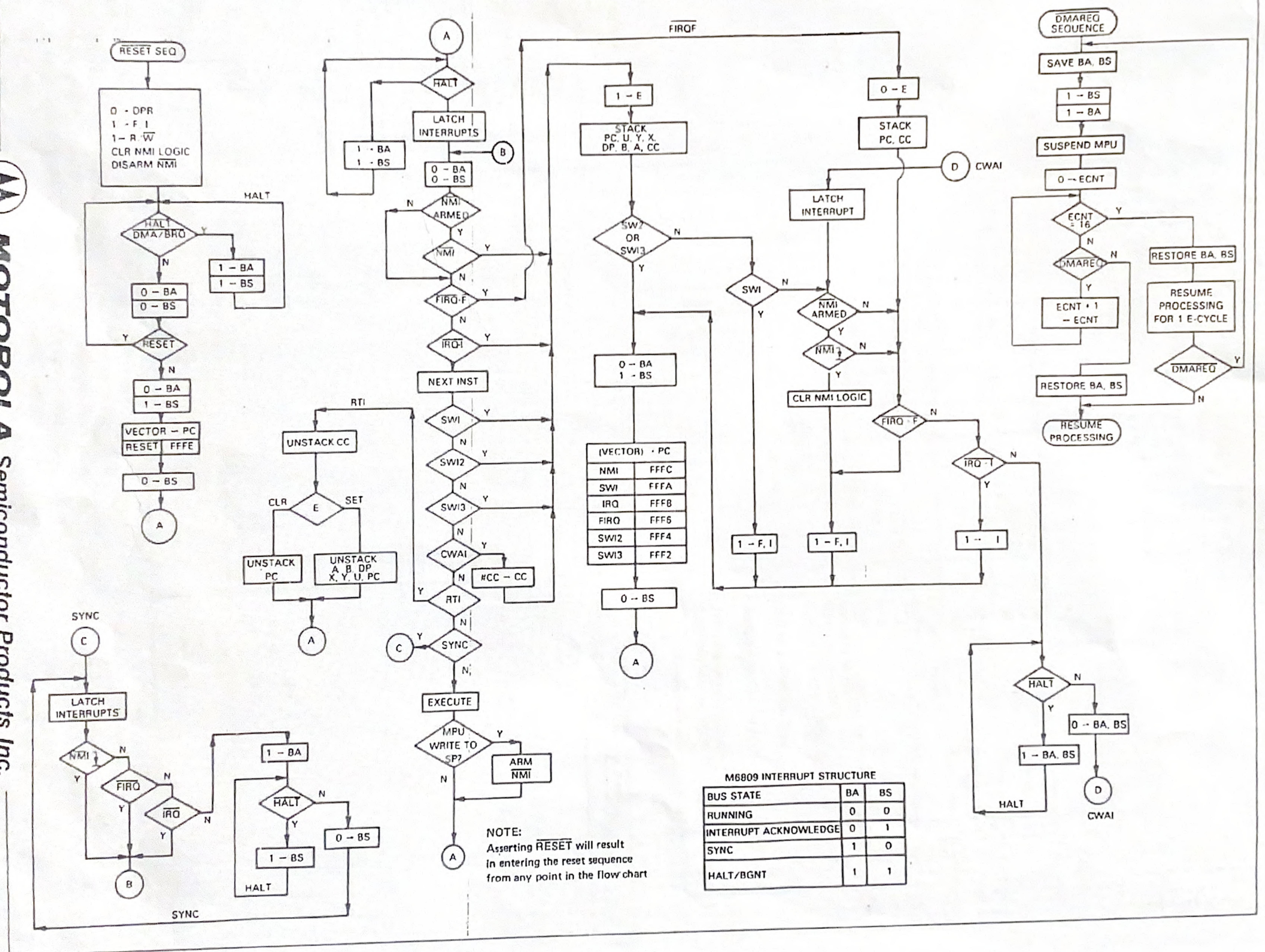
RR  
00=X  
01=Y  
10=U  
11=S

Tableau I (suite et fin)

RR



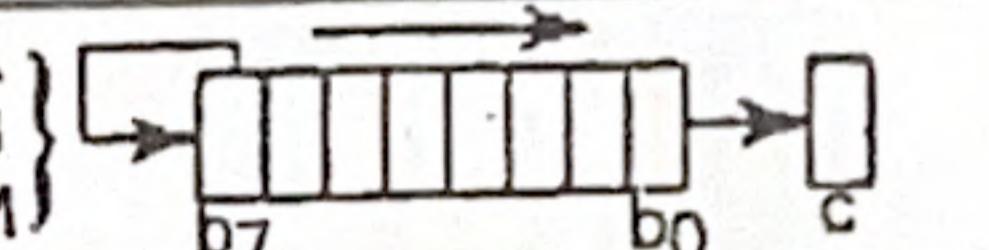
FIGURE 16 — MPU FLOWCHART



$\alpha$  - nombre cycle machines  
 $\#$  - nombre d'octets de commande

2t à partir de 2

**Table D-1. Programming Aid (Continued)**

Instruction	Forms	Addressing Modes												Description	5	3	2	1	0			
		Immediate			Direct			Indexed			Extended				H	N	Z	V	C			
		Op	-	#	Op	-	#	Op	-	#	Op	-	#	Op	-	#						
ABX														3A	3	1	B + X - X (Unsigned)					
ADC	ADCA	89	2	2	99	4	2	A9	4+	2+	B9	5	3					1	1	1	1	
	ADCB	C9	2	2	D9	4	2	E9	4+	2+	F9	5	3					1	1	1	1	
ADD	ADDA	8B	2	2	9B	4	2	AB	4+	2+	BB	5	3					1	1	1	1	
	ADDB	CB	2	2	DB	4	2	EB	4+	2+	FB	5	3					1	1	1	1	
	ADDD	C3	4	3	D3	6	2	E3	6+	2+	F3	7	3					•	1	1	1	
AND	ANDA	84	2	2	94	4	2	A4	4+	2+	B4	5	3					•	1	1	0	
	ANDB	C4	2	2	D4	4	2	E4	4+	2+	F4	5	3					•	1	1	0	
	ANDCC	1C	3	2														7				
ASL	ASLA													48	2	1		8	1	1	1	1
	ASLB													58	2	1		8	1	1	1	1
	ASL				08	6	2	68	6+	2+	78	7	3					8	1	1	1	1
ASR	ASRA A													47	2	1		8	1	1	•	1
	ASRB B													57	2	1		8	1	1	•	1
	ASR				07	6	2	67	6+	2+	77	7	3					8	1	1	•	1
BIT	BITA	85	2	2	95	4	2	A5	4+	2+	B5	5	3				Bit Test A (M & A)				•	
	BITB	C5	2	2	D5	4	2	E5	4+	2+	F5	5	3				Bit Test B (M & B)				•	
CLR	CLRA													4F	2	1	0-A				•	
	CLRB													5F	2	1	0-B				•	
	CLR				OF	6	2	6F	6+	2+	7F	7	3				0-M				•	
CMP	CMPA	81	2	2	91	4	2	A1	4+	2+	B1	5	3				Compare M from A				8	
	CMPB	C1	2	2	D1	4	2	E1	4+	2+	F1	5	3				Compare M from B				8	
	CMPD	10	5	4	10	7	3	10	7+	3+	10	8	4				Compare M:M+1 from D				•	
		83			93			A3			B3											
	CMPS	11	5	4	11	7	3	11	7+	3+	11	8	4				Compare M:M+1 from S				•	
		8C			9C			AC			BC											
	CMPU	11	5	4	11	7	3	11	7+	3+	11	8	4				Compare M:M+1 from U				•	
		83			93			A3			B3											
	CMPX	8C	4	3	9C	6	2	AC	6+	2+	BC	7	3				Compare M:M+1 from X				•	
COM	CMPY	10	5	4	10	7	3	10	7+	3+	10	8	4				Compare M:M+1 from Y				•	
		8C			9C			AC			BC											
	COMA													43	2	1	A-A				•	
COMB	COMB													53	2	1	B-B				•	
	COM				03	6	2	63	6+	2+	73	7	3				M-M				•	
																	CC & IMM-CC Wait for Interrupt				7	
CWAI		3C	≥20	2																		
DAA														19	2	1	Decimal Adjust A				•	
DEC	DECA													4A	2	1	A-1-A				•	
	DECB													5A	2	1	B-1-B				•	
	DEC				0A	6	2	6A	6+	2+	7A	7	3				M-1-M				•	
EOR	EORA	88	2	2	98	4	2	AB	4+	2+	B8	5	3				A+M-A				•	
	EORB	C8	2	2	D8	4	2	E8	4+	2+	F8	5	3				B+M-B				•	
EXG	R1, R2	1E	8	2													R1-R2 <sup>2</sup>				•	
INC	INCA													4C	2	1	A+1-A				•	
	INC B													5C	2	1	B+1-B				•	
	INC				0C	6	2	6C	6+	2+	7C	7	3				M+1-M				•	
JMP					OE	3	2	6E	3+	2+	7E	4	3				EA <sup>3</sup> -PC				•	
LD					9D	7	2	AD	7+	2+	BD	8	3				Jump to Subroutine				•	
	LDA	86	2	2	96	4	2	AB	4+	2+	B6	5										

Landon

**Legend:**

- OP Operation Code (Hexadecimal)
- Number of MPU Cycles
- # Number of Program Bytes
- + Arithmetic Plus
- Arithmetic Minus
- Multiply

## M Complement of M

## M Complement — Transfer Int

H Half-carry (from bit 3)

N Negative (sign)

Z Zero (Reset)

Z ZERO (Reset)  
V Overflow 2's comp

### C Carry from ALU

• Test and set if true, cleared otherwise

• Not Affected

CC Condition Code Register

## Concatenation

### V Logical or

## A Logical and

#### ⊕ Logical Exclusive or

Table D-1. Programming Aid (Continued)

Instruction	Forms	Addressing Modes												Description	5	3	2	1	0
		Immediate			Direct			Indexed <sup>1</sup>			Extended				H	N	Z	V	C
LSL	LSLA LSLB LSL				Op	-	#	Op	-	#	Op	-	#	Op	2	1			
					08	6	2	68	6+	2+	78	7	3	48	2	1			
														58	2	1			
LSR	LSRA LSRB LSR													44	2	1			
					04	6	2	64	6+	2+	74			54	2	1			
MUL														3D	11	1			
NEG	NEGA NEG B NEG													40	2	1	A × B - D (Unsigned)		
					00	6	2	60	6+	2+	70	7	3	50	2	1	$\bar{A} + 1 - A$		
																	$\bar{B} + 1 - B$		
																	$M + 1 - M$		
NOP														12	2	1	No Operation		
OR	ORA ORB ORCC	8A CA 1A	2 2 3	2 2 2	9A DA EA	4 4 +	2 2 +	AA + 2+	4+	2+	BA FA 5	5 5 3					A V M - A		
																	B V M - B		
																	CC V IMM - CC		
PSH	PSHS PSHU	34 36	5+ 5+	4	2												Push Registers on S Stack		
																	Push Registers on U Stack		
PUL	PULS PULU	35 37	5+ 5+	4	2												Pull Registers from S Stack		
																	Pull Registers from U Stack		
ROL	ROLA ROLB ROL													49	2	1			
					09	6	2	69	6+	2+	79	7	3	59	2	1			
ROR	RORA RORB ROR													46	2	1			
					06	6	2	66	6+	2+	76	7	3	56	2	1			
RTI														3B	6/15	1	Return From Interrupt		
RTS														39	5	1			
SBC	SBCA SBCB	B2 C2	2 2	2	92	4	2	A2	4+	2+	B2	5	3				Return from Subroutine		
					D2	4	2	E2	4+	2+	F2	5	3				A - M - C - A		
SEX																	B - M - C - B		
ST	STA STB STD STS STU STX STY				97	4	2	A7	4+	2+	B7	5	3				Sign Extend B into A		
<i>Stacker</i>					D7	4	2	E7	4+	2+	F7	5	3				A - M		
					DD	5	2	ED	5+	2+	FD	6	3				B - M		
					10	6	3	10	6+	3+	10	7	4				D - M : M + 1		
					DF			EF			FF						S - M : M + 1		
					DF	5	2	EF	5+	2+	FF	6	3				U - M : M + 1		
					9F	5	2	AF	5+	2+	BF	6	3				X - M : M + 1		
					10	6	3	10			10	7	4				Y - M : M + 1		
					9F			AF	6+	3+	BF								
SUB	SUBA SUBB SUBD	B0 CD B3	2 2 4	2 2 3	90	4	2	A0	4+	2+	B0	5	3				A - M - A		
					D0	4	2	E0	4+	2+	F0	5	3				B - M - B		
					A3	6+		B3	2+			7	3				D - M : M + 1 - D		
SWI	SWI <sup>6</sup> SWI2 <sup>6</sup> SWI3 <sup>6</sup>													3F	19	1	Software Interrupt 1		
														10	20	2	Software Interrupt 2		
														3F	20	1	Software Interrupt 3		
SYNC														13	$\geq 4$	1	Synchronize to Interrupt		
TFR	R1, R2	1F	6	2													R1 - R2 <sup>2</sup>		
TST	TSTA TSTB TST				OD	6	2	6D	6+	2+	7D	7	3	4D	2	1	Test A		
														5D	2	1	Test B		
																	Test M		

Notes:

1. This column gives a base cycle and byte count. To obtain total count, add the values obtained from the INDEXED ADDRESSING MODE table in Appendix F.
2. R1 and R2 may be any pair of 8 bit or any pair of 16 bit registers.  
The 8 bit registers are: A, B, CC, DP  
The 16 bit registers are: X, Y, U, S, D, PC
3. EA is the effective address.
4. The PSH and PUL instructions require 5 cycles plus 1 cycle for each byte pushed or pulled.
5. 5(6) means: 5 cycles if branch not taken, 6 cycles if taken (Branch instructions).
6. SWI sets I and F bits. SWI2 and SWI3 do not affect I and F.
7. Conditions Codes set as a direct result of the instruction.
8. Value of half-carry flag is undefined.
9. Special Case – Carry set if b7 is SET.

→ \* Étendu

LDA > \$2800  
7 octets

\$2800 7 bits

A - accumulateur

LDX > \$2400

X - registre

LDA #

LDA # \$AB A < \$AB

2400 2401 25 @

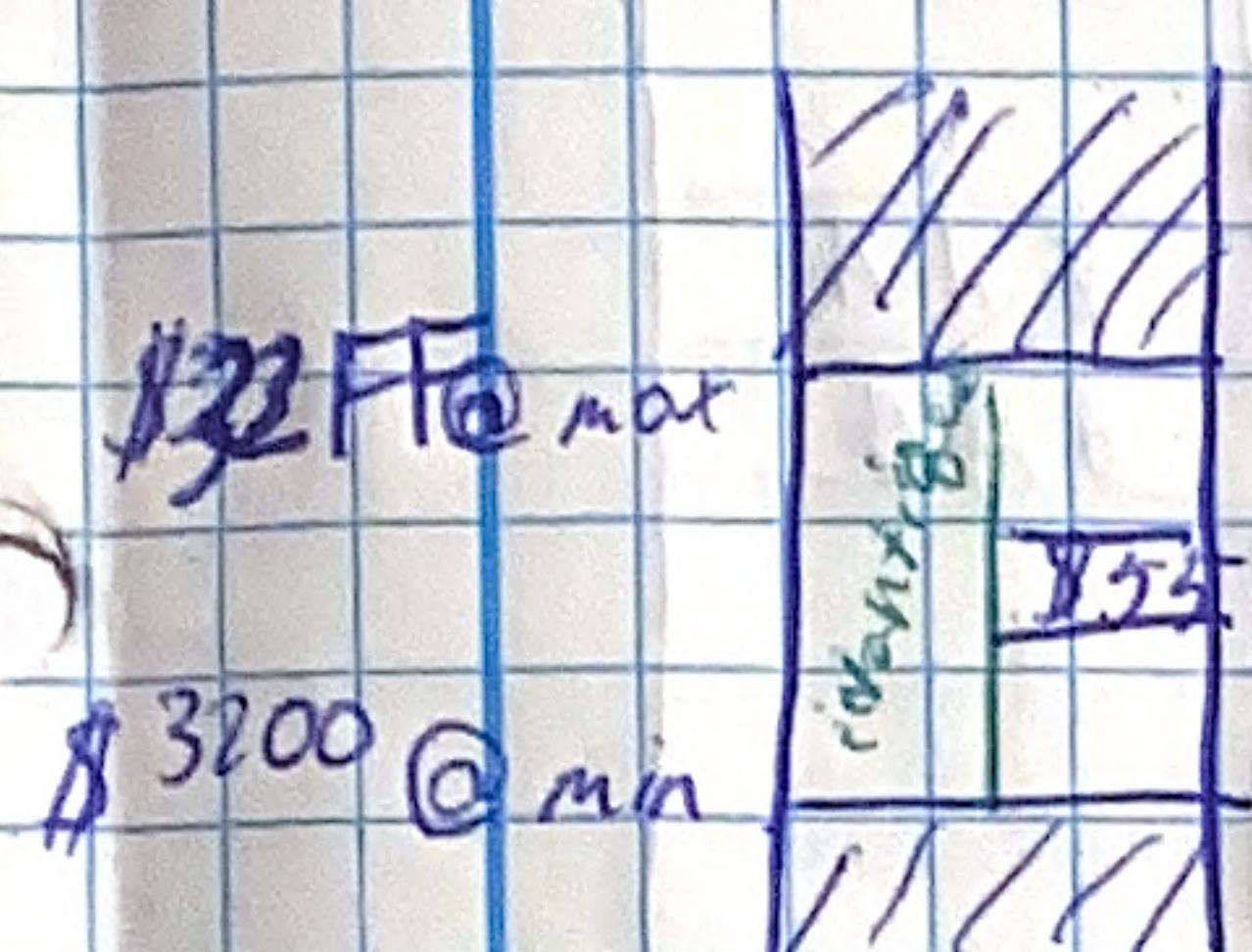
X < \$462B

→ Adressage direct

\* direct <

minimise l'envergure mémoire (exécutabile)

→ registre DP (Direct Page)



↳ mémorise l'octet de poids fort de l'adresse

DP < \$32

LDA < \$10

A < [\$3210] = 55

→ Adressage indexé

LDA , X

Lecture de données de RAM

quand on ne connaît pas où est contenu

LDX # \$2500

LDA, X → A & \$C8

X < \$12500

\$2500 & \$C8

avec déplacement

LDA , X + (+1)  
, X + + (+2)  
, - X (-1)  
, -- X (-2)

LDA, X A & [X]

X < X+1

déplacement fixe

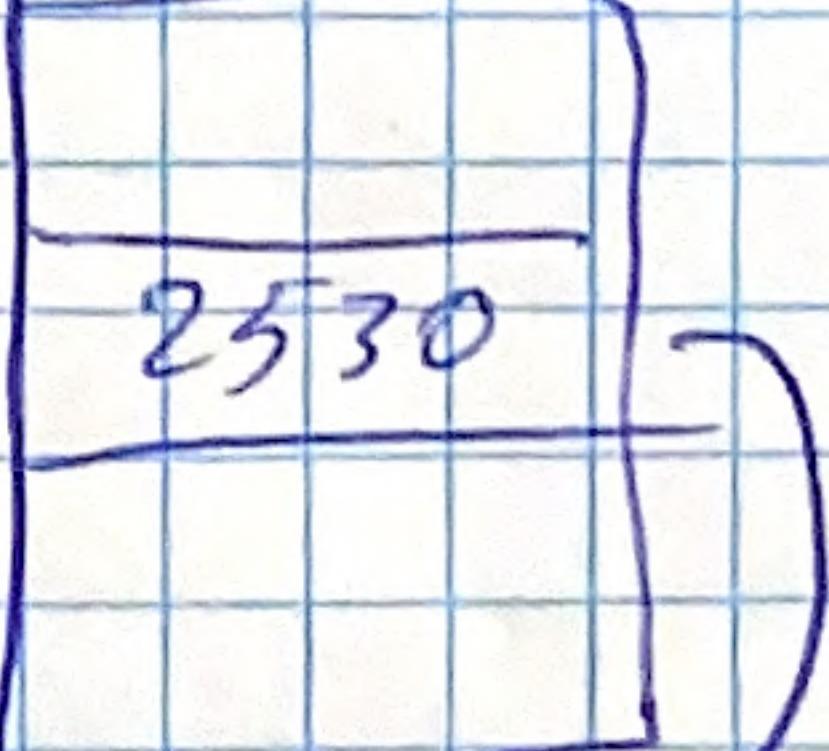
LDA \$30, X

A < [X + \$30]

si X = \$2500

@ = \$2500 + \$30 = \$2530

\$12500



A ←

déplacement modifiable

LDA B, X

$$A \leftarrow [X+B]$$
$$[\$2500 + \$40] = [\$2540]$$

→ nécessite d'utiliser 1 code supplémentaire  
dans l'écriture de l'action

LDA \$30, X

→ \* indirect []

LDA [\$B800]

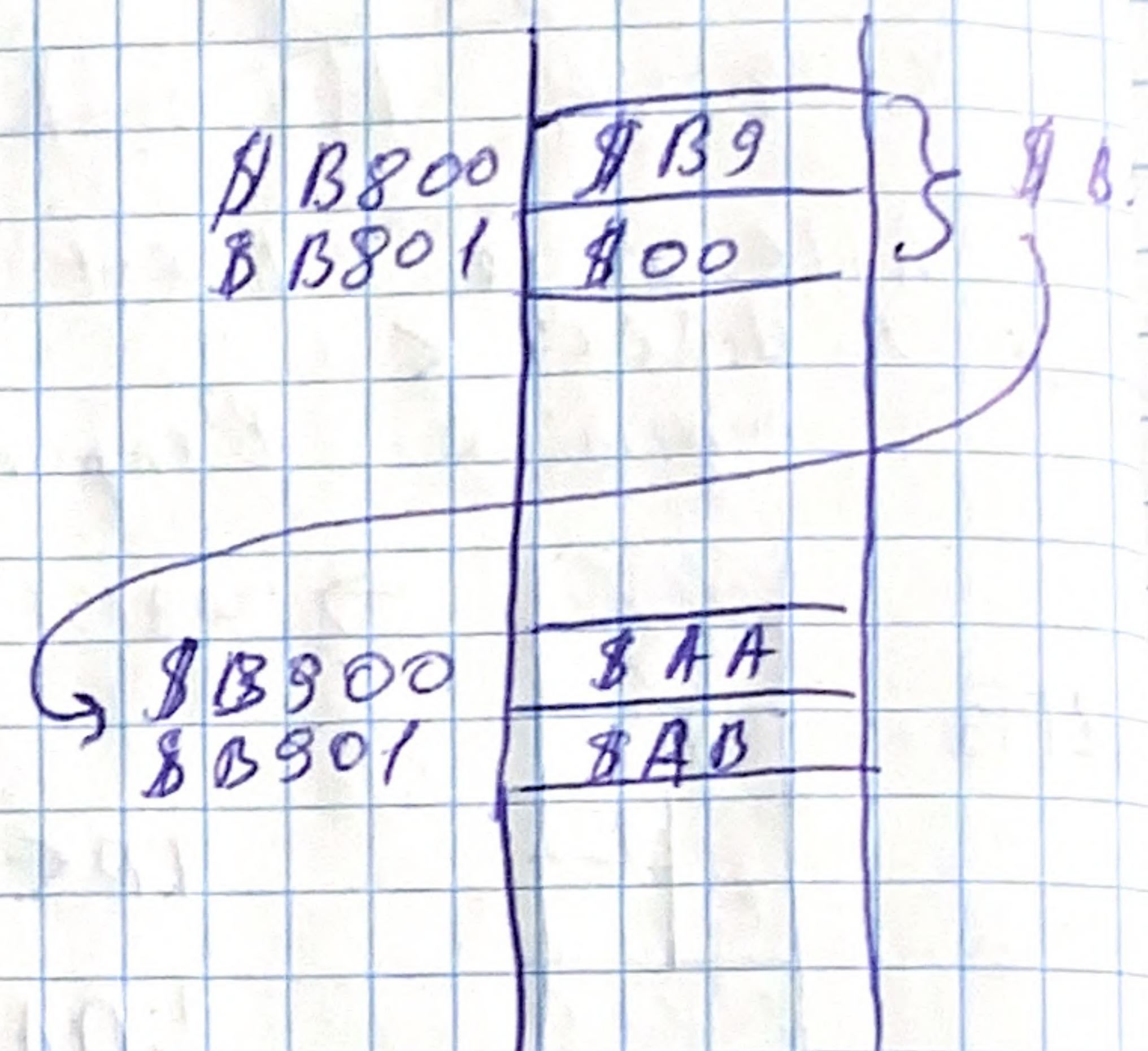
16 bits

$$A = \$AA$$

LDY [\$B800

16 bits

$$Y = \$AAAB$$



→ ADC addition avec retenue

CLR { A  
      B

$$A \leftarrow 0$$

$$B \leftarrow 0$$

COMA

$$A \leftarrow \bar{A}$$

DEC B

$$B \leftarrow B-1$$

INC A

$$A \leftarrow A+1$$

NEG B

$$B \leftarrow (\bar{B})_{\text{c}_2}$$

1010  
1000  
1000  
1010  
1010

LEAX 1,X X  $\leftarrow X+1$

load effect address

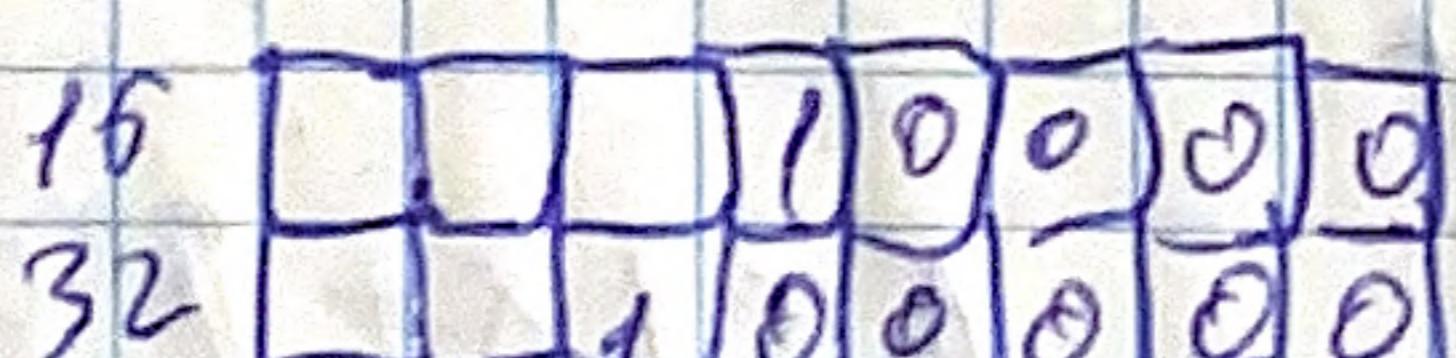
STB

[adresse]  $\leftarrow B$

ASL (x2)  $\ll 1$

arithmétique shift le

ASR (1/2)  $\gg 1$



$\ll$

Ecrire une program (algo) permettant de  
memoriser les 10 chiffres decimal à l'@ \$1000

~~prendre n chiffre~~

i < entier

~~tant que i < 10~~

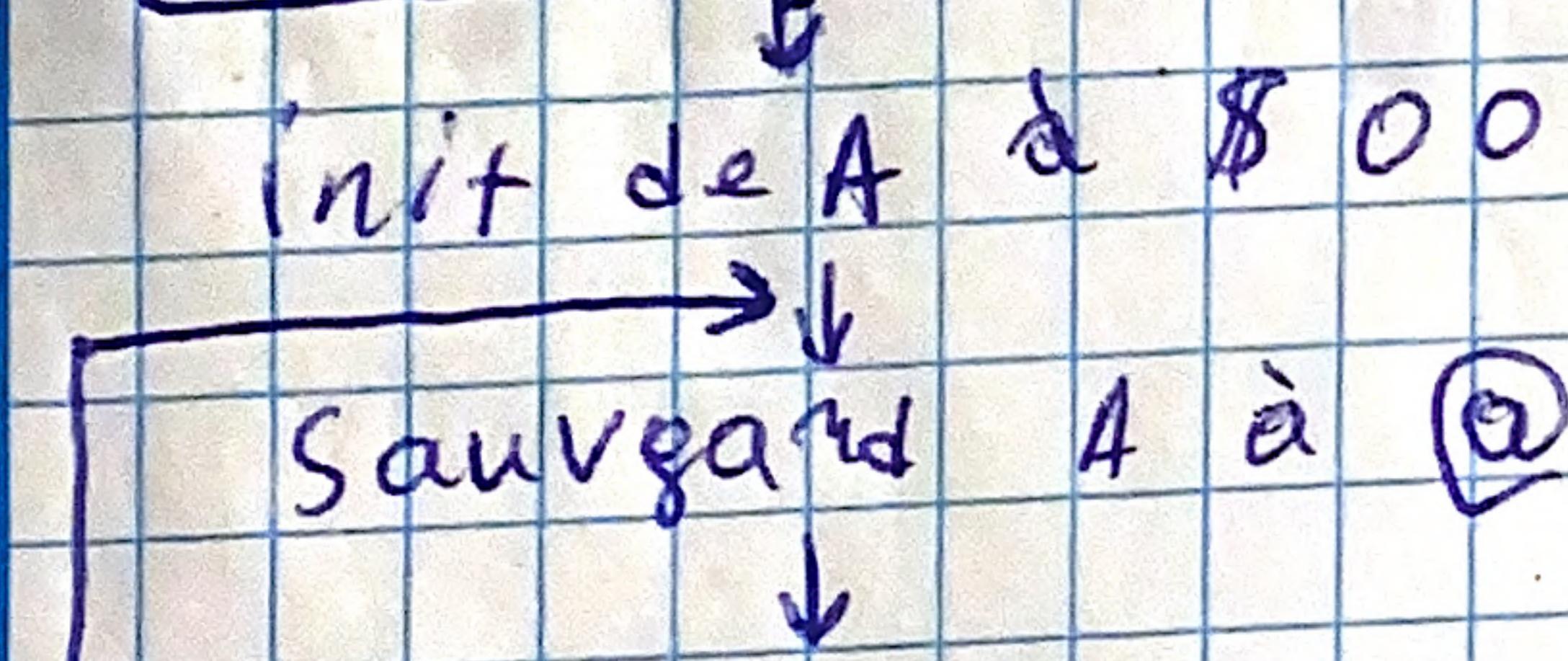
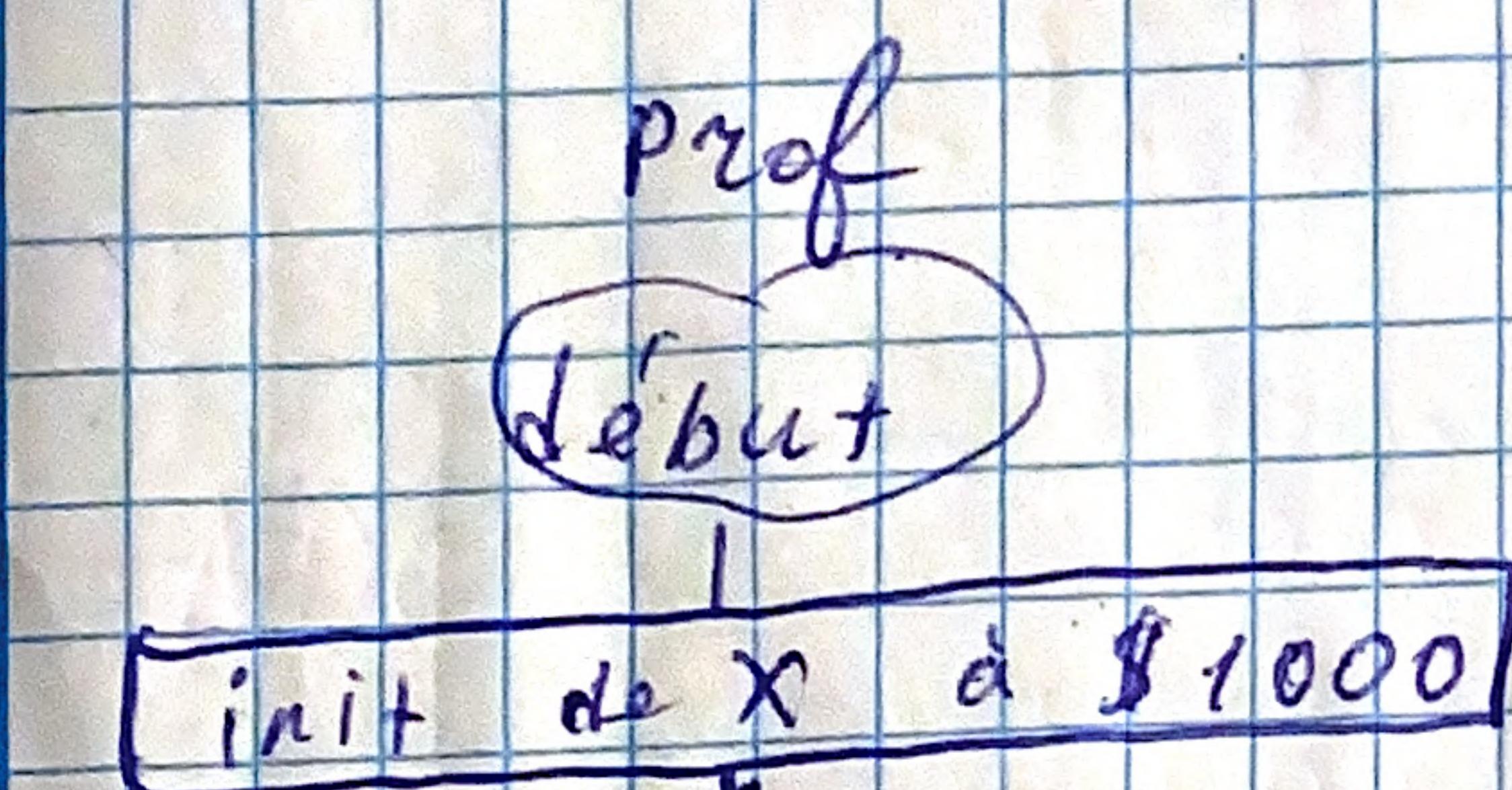
~~i~~

tab[i] sauvegarde dans @

@ + 1

i + 1

ORG - pour compilation



A ← A + 1

@ ← @ + 1

N

A = 10

fin

ORG \$C000

LDX # \$1000

LDA # \$00

STA , X

INCA

LEAX 1,X

CMPA # \$A

BEQ boucle

[A] = \$A

étagette