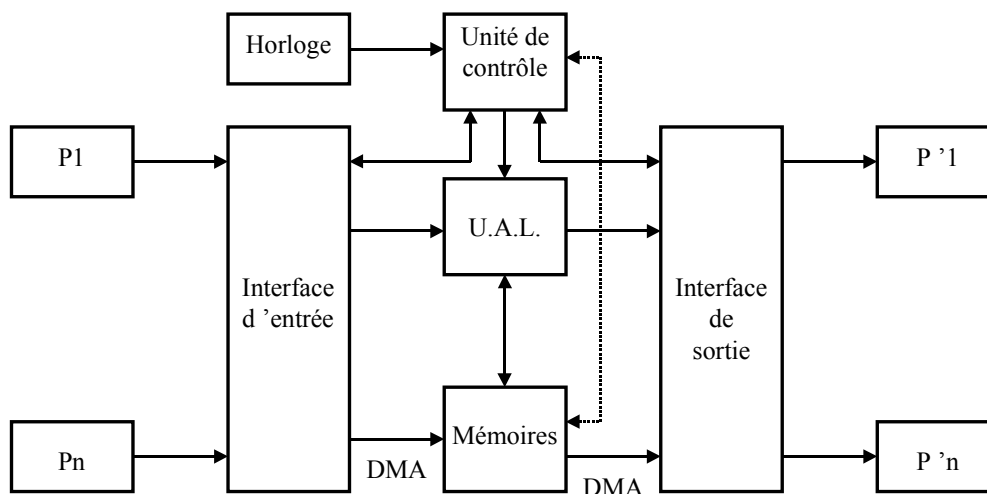


ARCHITECTURE D'UNE MACHINE INFORMATIQUE SYNCHRONISÉE PAR UNE HORLOGE

I) Fonctions de base d'une machine informatique

- Contenir de façon permanente les tâches à exécuter (mémoire programme) en ROM ou sur support magnétique.
- Contenir de façon temporaire des données (mémoire de travail) en RAM.
- Permettre un dialogue avec l'extérieur (circuit d'interface entrée/sortie) :
 - ↳ PIA- ACIA- TIMER
 - ↳ PIO- USART
 - ↳ SIO- UART.
- Effectuer des opérations arithmétiques et logiques élémentaires (UAL, en anglais ALU).
- Organiser des transits d'informations (unité de contrôle : U.C.)
- Cadencer les différentes informations (Horloge)
- Pointer l'étape du programme en cours (P.C. : compteur programme, en anglais Program Counter).

II) Schéma fonctionnel d'une machine informatique



A) Pn : Périphériques d'entrées

- | | |
|-----------------|------------------------------------|
| ➤ Clavier | ➤ Lecteur de bande magnétique |
| ➤ Souris | ➤ Lecteur de cartes |
| ➤ Joystick | ➤ Chaînes d'acquisition de données |
| ➤ Ecran tactile | ➤ Crayon optique |

B) Registres mémoires

A accès dit aléatoire c'est à dire, temps d'accès indépendant de l'emplacement de la donnée stockée.

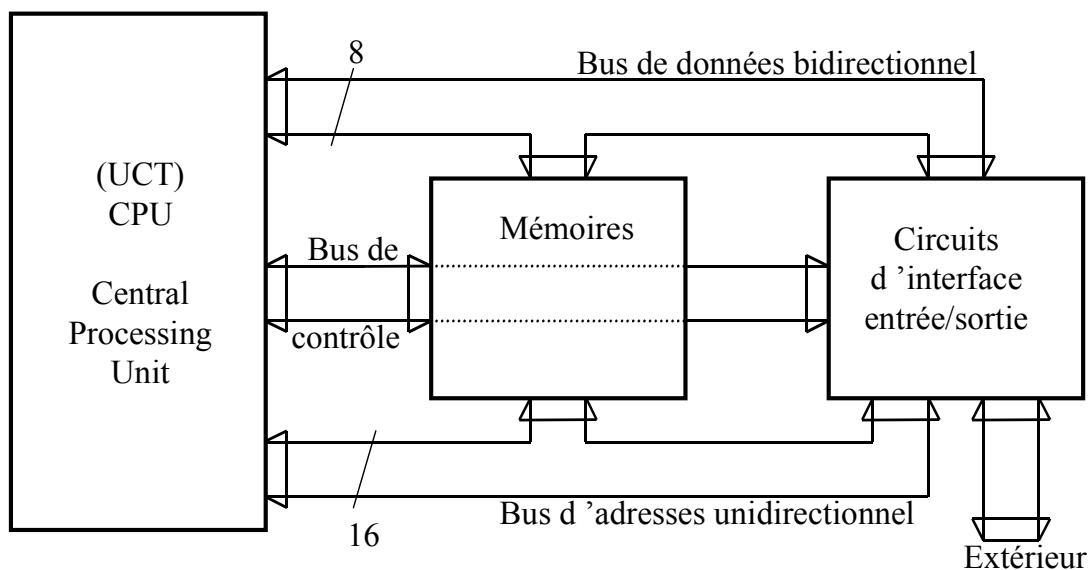
- | | | | |
|-------------------|--------|---------|-----------|
| ➤ Lecture seule : | ROM | ➤ RAM : | Statique |
| | PROM | | Dynamique |
| | REPROM | | |

C) Mémoires de masse

- Accès séquentiel : Bandes magnétiques
- Accès aléatoire : Disquettes
Disques durs
Cédéroms

D) P'n : Périphériques de sortie

- Dispositifs de visualisation : DELs
Affichage alphanumérique
Ecran vidéo
- Imprimante
- Table traçante
- Perforatrice de bandes
- Synthétiseur vocal
- Lecteur disquette

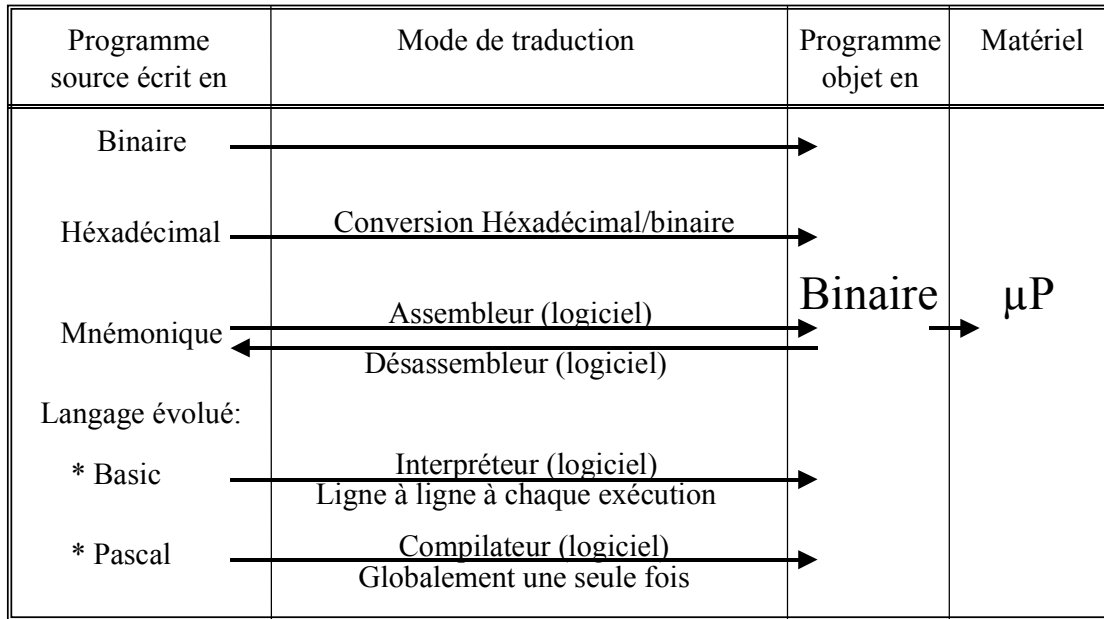
III) Architecture d'un système à microprocesseurA) Structure : μ P 68HC11 MOTOROLA

Les périphériques chez MOTOROLA sont considérés comme des accès mémoires.

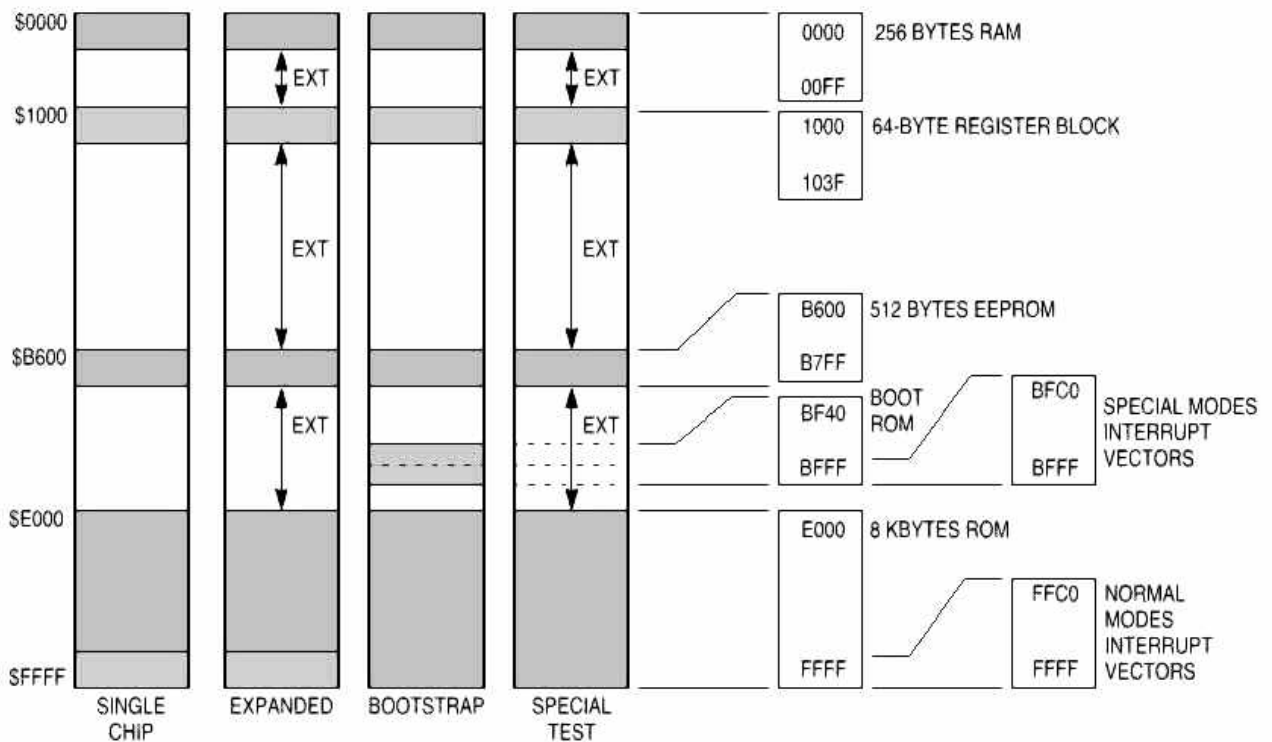
B) Notion de programme source et programme objet

- Programme source : Traduction de l'algorithme dans un langage compréhensible par la machine informatique (BASIC, PASCAL, C, FORTRAN, COBOL, ALGOL, PL1, ADA, LOGO, LSE, DELPHI, etc ...).
- Programme objet : Traduction du programme source en instructions codées en binaire, **seul langage exécutable par le microprocesseur.**

➤ Passage du programme source au programme objet :

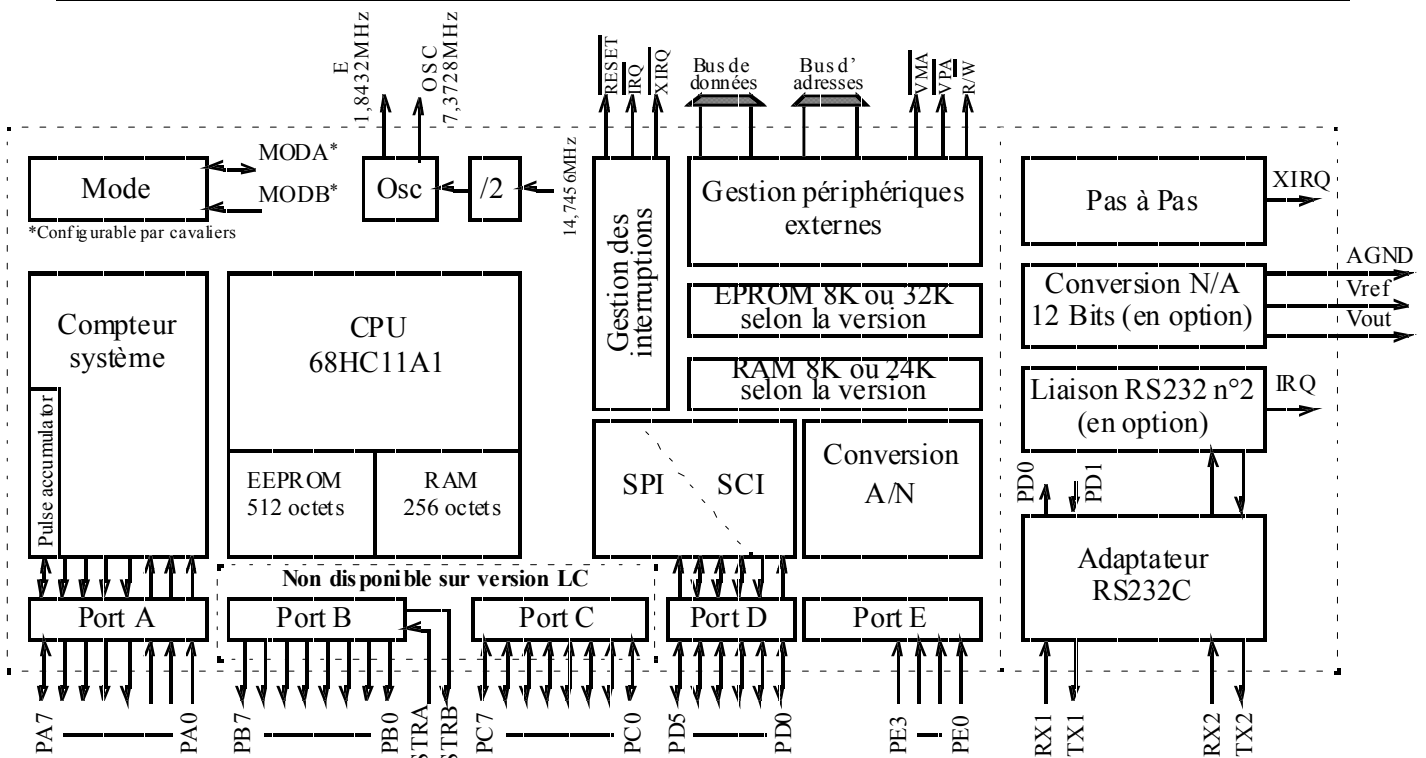


C) Structure interne 68HC11



Mapping mémoire du 68HC11.

Le 68HC11 est un microcontrôleur. C'est un microprocesseur qui possède de la RAM, ROM et divers périphériques d'entrées/sorties. Un microcontrôleur peut donc être utilisé seul, car il est à lui seul une machine informatique. Un microprocesseur, PENTIUM, 6809, Z80, etc..., ne peut fonctionner seul, il a nécessairement besoin de ROM et de RAM externes.



Architecture d'un microprocesseur de la famille 68HC11.

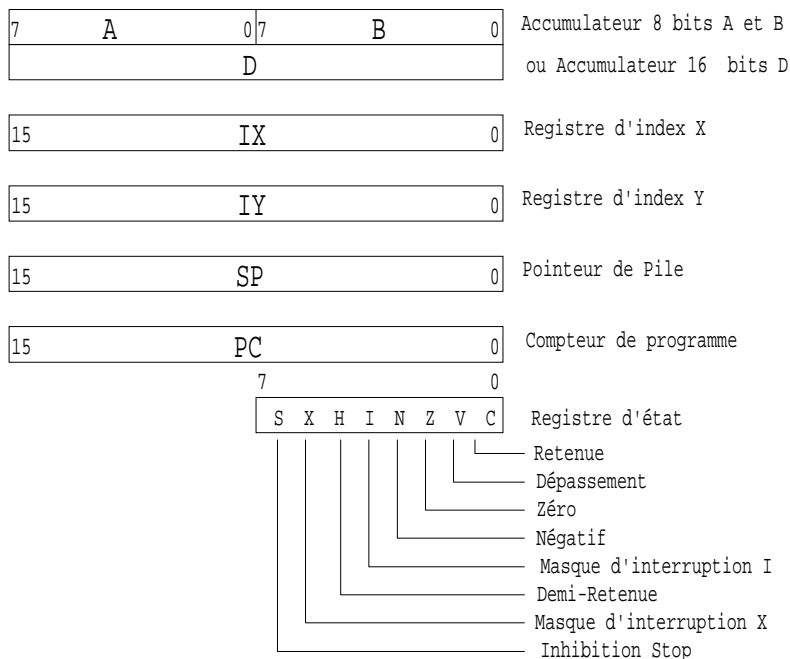
L'unité centrale 68HC11 est dérivée du « vieux » 6801. Il existe une grande variété de 68HC11, issus du 68HC11A8.

Autour de l'unité centrale du 68HC11, on trouve de la mémoire. Elle se subdivise au maximum en trois blocks distincts dont la taille et la présence varient selon les références exactes du circuit : RAM (256 octets minimum), ROM (présente ou absente) et EEPROM.

Cette unité centrale est entourée d'un certain nombre de ports parallèles baptisés port A à port E qui peuvent être bidirectionnels ou unidirectionnels selon le cas.

Un timer est également disponible. Il comporte plusieurs timers très évolués, ainsi qu'un accumulateur d'impulsions, une horloge temps réel et un chien de garde (ou COP : Computer Operating Properly) destiné à surveiller le fonctionnement du microcontrôleur.

Un convertisseur analogique/numérique à huit entrées complète les ressources internes du 68HC11.

D) Accumulateurs

Registres de travail du 68HC11

➤ Accumulateurs A, B et D :

Les registres A et B sont des accumulateurs 8 bits indépendants utilisés pour toutes opérations arithmétiques et logiques. A et B peuvent être concaténés et former un accumulateur D de 16 bits. Attention cet accumulateur n'est pas indépendant de A et B.

➤ Registres d'index X et Y :

Les registres d'index X et Y, indépendants, possèdent 16 bits chacun, puisque la capacité d'adressage du 68HC11 est de 16 bits d'adresses. Leur rôle premier est d'être utilisé pour l'adressage indexé, mais ils peuvent être utilisés pour le stockage temporaire de données, ou pour quelques opérations arithmétiques élémentaires.

➤ Pointeur de pile S :

Le pointeur de pile S est un registre 16 bits. Ce registre pointe sur la première adresse libre de la zone mémoire définie pour ranger différents paramètres, tel registre PC, registre CCR, etc ..., suivant la demande (interruption, appel à un sous-programme). Il est indispensable de l'initialiser à une adresse RAM, sinon à la première interruption ou appel à un sous-programme, le programme se plantera (bug).

➤ Compteur programme ou PC :

Le compteur programme, PC, indique l'adresse du prochain code binaire à 8 bits (instruction ou valeur de travail) à traiter. La taille du registre PC est donc de 16 bits.

➤ Registre d'état ou CCR :

Le registre d'état ou CCR (Condition Code Register) est un registre sur 8 bits. Chaque bit à une signification particulière.

↳ Le bit C (Carry) : C est mis à 1 lorsqu'une opération arithmétique génère une retenue. Il est également utilisé comme indicateur d'erreur lors d'une multiplication ou d'une division, et sert lors de certaines opérations de décalage ou rotation qui peuvent passer par son intermédiaire ou non.

↳ Le bit V (oVerflow) : V est mis à 1 lorsqu'une opération arithmétique génère un débordement de l'accumulateur.

↳ Le bit Z (Zero) : Z est **mis à 1** lorsque le **résultat** de l'instruction exécuté **est nul**.

↳ Le bit N (Negative) : N est **mis à 1** lorsque le **résultat** de la dernière opération arithmétique réalisée est **négatif** (bit de poids fort du résultat à 1).

↳ Le bit H (Half carry) : H, demi-retenue, est mis à 1 lors d'une retenue entre les bits 3 et 4 d'une opération arithmétique. Il n'est affecté que par les instructions ADD, ADC et ABA, et est ensuite exploité par l'instruction DAA pour réaliser de l'arithmétique DCB (en anglais BCD), c'est à dire codée en fait sur 2 groupes de 4 bits.

↳ Le bit I (Interrupt mask) : I interdit toute interruption masquable lorsqu'il est mis à 1. C'est à dire qu'il permet d'autoriser les interruptions du TIMER, ACIA et autres. Suite à un RESET, I est mis à 1.

↳ Le bit X (Xirq interrupt mask) : X interdit toute interruption masquable lorsqu'il est mis à 1. Suite à un RESET, X est mis à 1. Attention XIRQ est une interruption non masquable en général, mais dans le cas du 68HC11, il est possible de la masquer.

↳ Le bit S (Stop disable) : S est mis à 1 pour interdire l'exécution de l'instruction STOP. Elle sera considérée comme un simple NOP (No Operation).

Il existe certaines instructions pour manipuler directement certains bits du registre CCR.

E) Définitions des signaux de contrôle

↳ Alimentation (VDD et VSS) :
VSS=0V et VDD=+5V à 5%.

↳ $MODA/\overline{LIR}$ et MODB/Vstby :

MODB	MODA	Mode de fonctionnement
0	0	Normal- Circuit seul
0	1	Normal- Etendu
1	0	Spécial- Bootstrap
1	1	Spécial- Test

↳ EXTAL et XTAL :

Entrées du quartz pour générer le signal d'horloge E. On peut aussi appliquer seulement un signal d'horloge sur l'entrée EXTAL.

↳ E :

E est une sortie d'horloge de bus. Sa fréquence est égale au quart de la valeur du quartz. L'état logique de E, haut ou bas, indique si des données ou des adresses valides sont présentes sur le bus du 68HC11.

↳ RESET :

Entrée RESET active au niveau bas. Mais elle peut devenir une sortie, dans le cas du chien de garde ou COP, active là aussi au niveau bas.

↳ IRQ :

Entrée d'interruption masquable (voir bit I du registre CCR et bit IRQE du registre option).

↳ XIRQ :

Entrée d'interruption non masquable (voir bit X du registre CCR). Cette interruption est quand même masquable.

↳ VREFL et VREFH :

VREFL et VREFH sont respectivement des entrées de référence basse et haute du convertisseur analogique/numérique. Pour une bonne précision, la différence de potentiel entre VREFH et VREFL doit être au moins de 2,5V, mais restée dans les limites de VDD et VSS.

↳ Port E :

PE0 à PE7 sont des entrées du port parallèle E. Ces lignes sont unidirectionnelles et ne fonctionnent qu'en entrée. De plus elles sont partagées avec les entrées du convertisseur analogique/numérique. Il est possible de faire travailler le port E en numérique et en analogique en même temps (il faudra soigner le programme sous peine de légères perturbations).

↳ Port D :

PD0 à PD5 sont des entrées ou sorties du port parallèle D. Ces lignes sont programmables indépendamment en entrée ou sortie. La programmation du port D est réalisée par le registre DDRD, registre de programmation des lignes du port D.

↳ Port C :

PC0 à PC7 sont des entrées ou sorties du port parallèle C. Ces lignes sont programmables indépendamment en entrée ou sortie. La programmation du port C est réalisée par le registre DDRC, registre de programmation des lignes du port C.

En mode étendu, le port C devient bus de données (8 bits) ou bus de poids faible des adresses (8 bits) (multiplexage des différents bus).

↳ Port B :

PB0 à PB7 sont des sorties du port parallèle B.

En mode étendu, le port B devient bus de poids fort des adresses (8 bits).

↳ STRA/AS :

STRA est une entrée permettant de mémoriser des données présentes sur le port C en entrée. STRA est actif sur un front programmable.

↳ STRB/R/ \overline{W} :

STRB est une sortie indiquant lorsque des données sont présentes sur les lignes du port B et C qui ont été placées en sorties (fonction strobe).

↳ Port A :

PA0 à PA2 sont des entrées du port parallèle A, PA3 à PA6 sont les sorties de ce port et PA7 est l'entrée ou sortie de ce même port.

Ces lignes sont partagées par le timer. PA0 à PA2 sont les entrées de capture IC3 à IC1 du timer. PA3 à PA6 sont les sorties de comparaison OC1 à OC5 du timer. PA7 peut être configuré en entrée de capture ou sortie de comparaison.

IV) Structures algorithmiques fondamentales

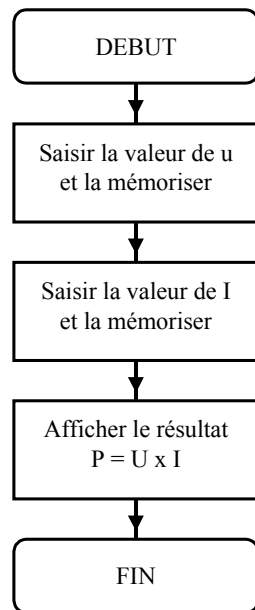
A) Définitions :

➤ Organigramme : (en anglais flowchart, se dit aussi ordinogramme) c'est un schéma synoptique qui analyse dans le détail les différentes phases du problème à traiter.

➤ Algorithme : C'est la transposition d'un organigramme en tâches simplifiées.

B) Structure linéaire ou séquentielle :

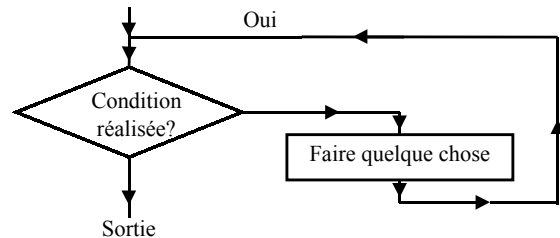
Ex: $P = U \times I$



C) Structure itérative :

1) Structure itérative : boucle « tant que » :

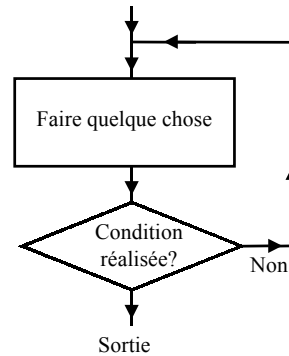
Tant que < condition réalisée >
Faire quelque chose
Répéter



Remarque : Le travail peut ne jamais être effectué.

2) Structure itérative : boucle « jusqu'à » :

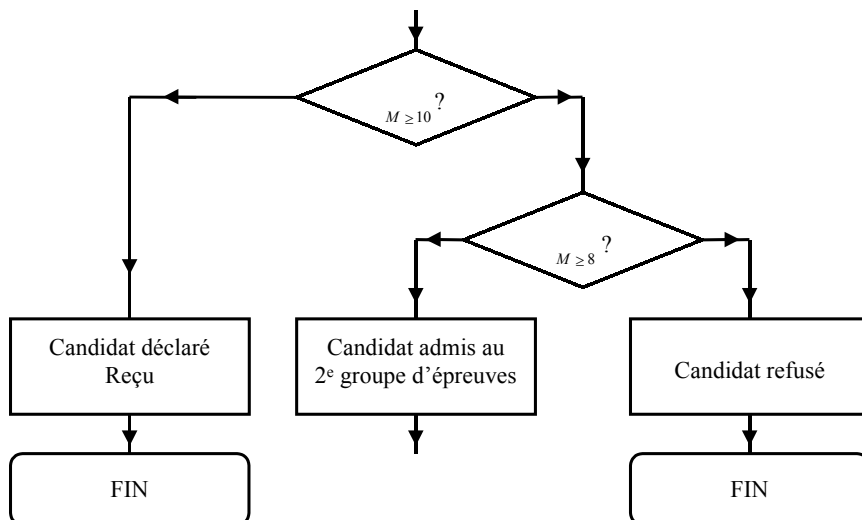
Faire quelque chose
Jusqu'à ce que < condition réalisée >



Remarque : Le travail à effectuer l'est au moins une fois.

D) Structure alternative :

Ex: Un candidat est reçu au bac si $M \geq 10$



V) Modes d'adressage

A) Adressage inhérent :

Ce mode n'est pas à proprement parler un mode d'adressage, mais tous les fabricants le décomptent comme tel.

Les instructions agissent sur les registres internes du microprocesseur. Les instructions comportent donc un code opérateur seul sans opérande.

- ABA Additionne le contenu de A avec celui de B. Le résultat se trouve dans A.
- ASLA Effectue un décalage vers la gauche des bits de A.

B) Adressage immédiat :

Dans ce mode d'adressage, le code opérateur est suivi par la donnée à manipuler, code opérande, qui est précédée par le symbole #. L'instruction est donc composée d'un code opérateur, d'un # et enfin d'un code opérande.

- LDAA #55 Stocke la valeur 55 en décimale dans le registre A.
- LDD #5A0 Stocke la valeur 5A0 en hexadécimale dans le registre D.

C) Adressage direct :

Ce mode d'adressage, que l'on peut confondre avec l'adressage immédiat, est le premier « vrai » mode d'adressage du 68HC11. Le code opérateur est suivi d'un octet non signé qui est l'adresse effective de la donnée, ou opérande, à manipuler.

Ne codant l'adresse que sur un octet, il n'est possible que d'adresser les 256 premiers octets, l'adresse \$00 à l'adresse \$FF, de l'espace adressable du microcontrôleur.

- LDAA \$55 Stocke la valeur contenue à l'adresse \$0055 dans le registre A.
- ADDA 55 Additionne la valeur contenue à l'adresse \$0037 avec la valeur contenue dans le registre A. Résultat de l'opération arithmétique dans le registre A.

D) Adressage étendu ou absolu :

Ce mode d'adressage est une évolution du précédent et permet d'atteindre n'importe quelle adresse mémoire car il autorise un codage de l'adresse effective sur deux octets, ou 16 bits. Le champs mémoire adressable est donc de \$0000 à \$FFFF.

L'opérande est donc codé sur 2 octets.

- LDAA \$B05A Stocke la valeur contenue à l'adresse \$B05A dans le registre A.
- ADDA \$B05A Additionne la valeur se trouvant à l'adresse \$B05A avec la valeur contenue dans le registre A. Résultat de l'opération arithmétique dans le registre A.

E) Adressage indexé avec déplacement d'une constante :

Ce mode d'adressage est particulièrement souple et fait intervenir deux facteurs dans le calcul de l'adresse effective : le contenu d'un des registres d'index X ou Y et une donnée appelée déplacement ou offset (non signé).

- LDAA \$33,X Stocke la valeur se trouvant à l'adresse X+\$33 dans le registre A.
- ADDA \$33,Y Additionne la valeur contenue à l'adresse Y+\$33 avec la valeur contenue dans le registre A. Résultat de l'opération arithmétique dans le registre A.
- STAB ,X Stocke la valeur contenue dans le registre B à l'adresse X, car l'offset est de 0 ici.

F) Adressage indexé avec déplacement accumulateur :

Ce mode fonctionne de la même manière que précédemment, cependant l'offset n'est plus une constante, mais un registre du 68HC11.

- LDAA B,X Stocke la valeur se trouvant à l'adresse X+B dans le registre A. X étant un registre sur 16 bits et B un registre sur 8 bits.

Remarque : *On peut conclure que l'offset est une donnée sur 8 bits non signée. Il ne peut donc varier que \$00 à \$FF.*

G) Adressage relatif :

Ce mode d'adressage se rencontre quasiment sur tous les microprocesseurs, mais n'est utilisé ici que pour les instructions de saut et branchement. Il spécifie l'adresse effective sous forme d'un déplacement qui est donc ajouté à la valeur courante du PC pour déterminer l'adresse de poursuite du programme. Comme lors de l'exécution d'une instruction, le PC pointe toujours sur l'instruction suivante, l'adresse effective est en fait égale à la valeur courante du PC+2+le déplacement.

Le déplacement est codé sur un octet signé, d'où un déplacement de -128 à +127 en base décimale.

- BEQ \$FB Après exécution de la ligne de commande le contenu du PC sera additionné à -5 si le bit Z est à 1, sinon le contenu du PC s'incrémentera de +2.
- BCC \$10 Après exécution de la ligne de commande le contenu du PC sera additionné à +16 si le bit C est à 0, sinon le contenu du PC s'incrémentera de +2.

VI) Jeux d'instructions du 68HC11

Cycle

- * Infinity or until reset occurs
- ** 12 Cycles are used beginning with the opcode fetch. A wait state is entered which remains in effect for an integer number of MPU E-Clc cycles (n) until an interrupt is recognized. Finally, two additional cycles are used to fetch the appropriate interrupt vector (14 + n total).

Operands

- dd = 8-Bit Direct Address (\$0000 - \$00FF) (High Byte Assumed to be \$00)
- ff = 8-Bit Positive Offset \$00 (0) to \$FF (255) (Is Added to Index)
- hh = High-Order Byte of 16-Bit Extended Address
- ii = One Byte of Immediate Data
- jj = High-Order Byte of 16-Bit Immediate Data
- kk = Low-Order Byte of 16-Bit Immediate Data
- ll = Low-Order Byte of 16-Bit Extended Address
- mm = 8-Bit Mask (Set Bits to be Affected)
- rr = Signed Relative Offset \$80 (-128) to \$7F (+127)
(Offset Relative to Address Following Machine Code Offset Byte)

Operators

- () Contents of register shown inside parentheses
- ← Is transferred to
- ↑ Is pulled from stack
- ↓ Is pushed onto stack
- Boolean AND
- +
- ⊕ Exclusive-OR
- *
- :
- Arithmetic subtraction symbol or Negation symbol (Two's Complement)

Condition Codes

- Bit not changed
- 0 Bit always cleared
- 1 Bit always set
- Δ Bit cleared or set, depending on operation
- ↓ Bit can be cleared, cannot become set

Table 3-2 Instruction Set (Sheet 1 of 6)

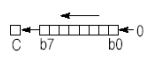
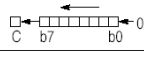
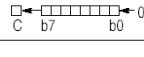
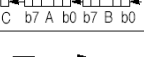
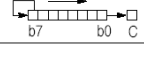
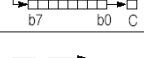
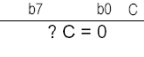
Mnemonic	Operation	Description	Addressing Mode	Instruction			Condition Codes							
				Opcode	Operand	Cycles	S	X	H	I	N	Z	V	C
ABA	Add Accumulators	$A + B \Rightarrow A$	INH	1B	—	2	—	—	Δ	—	Δ	Δ	Δ	Δ
ABX	Add B to X	$IX + (00 : B) \Rightarrow IX$	INH	3A	—	3	—	—	—	—	—	—	—	—
ABY	Add B to Y	$IY + (00 : B) \Rightarrow IY$	INH	18 3A	—	4	—	—	—	—	—	—	—	—
ADCA (opr)	Add with Carry to A	$A + M + C \Rightarrow A$	A IMM	89	ii	2	—	—	Δ	—	Δ	Δ	Δ	Δ
			A DIR	99	dd	3								
			A EXT	B9	hh ll	4								
			A IND,X	A9	ff	4								
			A IND,Y	A9	ff	5								
ADCB (opr)	Add with Carry to B	$B + M + C \Rightarrow B$	B IMM	C9	ii	2	—	—	Δ	—	Δ	Δ	Δ	Δ
			B DIR	D9	dd	3								
			B EXT	F9	hh ll	4								
			B IND,X	E9	ff	4								
			B IND,Y	E9	ff	5								
ADDA (opr)	Add Memory to A	$A + M \Rightarrow A$	A IMM	8B	ii	2	—	—	Δ	—	Δ	Δ	Δ	Δ
			A DIR	9B	dd	3								
			A EXT	BB	hh ll	4								
			A IND,X	AB	ff	4								
			A IND,Y	AB	ff	5								
ADDB (opr)	Add Memory to B	$B + M \Rightarrow B$	B IMM	CB	ii	2	—	—	Δ	—	Δ	Δ	Δ	Δ
			B DIR	DB	dd	3								
			B EXT	FB	hh ll	4								
			B IND,X	EB	ff	4								
			B IND,Y	EB	ff	5								
ADDD (opr)	Add 16-Bit to D	$D + (M : M + 1) \Rightarrow D$	IMM	C3	jj kk	4	—	—	—	—	Δ	Δ	Δ	Δ
			DIR	D3	dd	5								
			EXT	F3	hh ll	6								
			IND,X	E3	ff	6								
			IND,Y	E3	ff	7								
ANDA (opr)	AND A with Memory	$A \cdot M \Rightarrow A$	A IMM	84	ii	2	—	—	—	—	Δ	Δ	0	—
			A DIR	94	dd	3								
			A EXT	B4	hh ll	4								
			A IND,X	A4	ff	4								
			A IND,Y	A4	ff	5								
ANDB (opr)	AND B with Memory	$B \cdot M \Rightarrow B$	B IMM	C4	ii	2	—	—	—	—	Δ	Δ	0	—
			B DIR	D4	dd	3								
			B EXT	F4	hh ll	4								
			B IND,X	E4	ff	4								
			B IND,Y	E4	ff	5								
ASL (opr)	Arithmetic Shift Left		EXT	78	hh ll	6	—	—	—	—	Δ	Δ	Δ	Δ
			IND,X	68	ff	6								
			IND,Y	68	ff	7								
ASLA	Arithmetic Shift Left A		A INH	48	—	2	—	—	—	—	Δ	Δ	Δ	Δ
ASLB	Arithmetic Shift Left B		B INH	58	—	2								
ASLD	Arithmetic Shift Left D		INH	05	—	3								
ASR	Arithmetic Shift Right		EXT	77	hh ll	6	—	—	—	—	Δ	Δ	Δ	Δ
			IND,X	67	ff	6								
			IND,Y	67	ff	7								
ASRA	Arithmetic Shift Right A		A INH	47	—	2	—	—	—	—	Δ	Δ	Δ	Δ
ASRB	Arithmetic Shift Right B		B INH	57	—	2								
BCC (rel)	Branch if Carry Clear	? C = 0	REL	24	rr	3								
BCLR (opr) (msk)	Clear Bit(s)	$M \cdot (mm) \Rightarrow M$	DIR	15	dd mm	6	—	—	—	—	Δ	Δ	0	—
			IND,X	1D	ff mm	7								
			IND,Y	1D	ff mm	8								
BCS (rel)	Branch if Carry Set	? C = 1	REL	25	rr	3	—	—	—	—	—	—	—	
BEQ (rel)	Branch if = Zero	? Z = 1	REL	27	rr	3	—	—	—	—	—	—	—	
BGE (rel)	Branch if Δ Zero	? $N \oplus V = 0$	REL	2C	rr	3	—	—	—	—	—	—	—	

Table 3-2 Instruction Set (Sheet 2 of 6)

Mnemonic	Operation	Description	Addressing Mode	Instruction			Condition Codes							
				Opcode	Operand	Cycles	S	X	H	I	N	Z	V	C
BGT (rel)	Branch if > Zero	? Z + (N ⊕ V) = 0	REL	2E	rr	3	—	—	—	—	—	—	—	—
BHI (rel)	Branch if Higher	? C + Z = 0	REL	22	rr	3	—	—	—	—	—	—	—	
BHS (rel)	Branch if Higher or Same	? C = 0	REL	24	rr	3	—	—	—	—	—	—	—	
BITA (opr)	Bit(s) Test A with Memory	A • M	A IMM	85	ii	2	—	—	—	—	Δ	Δ	0	—
			A DIR	95	dd	3								
			A EXT	B5	hh ll	4								
			A IND,X	A5	ff	4								
			A IND,Y	A5	ff	5								
BITB (opr)	Bit(s) Test B with Memory	B • M	B IMM	C5	ii	2	—	—	—	—	Δ	Δ	0	—
			B DIR	D5	dd	3								
			B EXT	F5	hh ll	4								
			B IND,X	E5	ff	4								
			B IND,Y	E5	ff	5								
BLE (rel)	Branch if Δ Zero	? Z + (N ⊕ V) = 1	REL	2F	rr	3	—	—	—	—	—	—	—	
BLO (rel)	Branch if Lower	? C = 1	REL	25	rr	3	—	—	—	—	—	—		
BLS (rel)	Branch if Lower or Same	? C + Z = 1	REL	23	rr	3	—	—	—	—	—	—		
BLT (rel)	Branch if < Zero	? N ⊕ V = 1	REL	2D	rr	3	—	—	—	—	—	—		
BMI (rel)	Branch if Minus	? N = 1	REL	2B	rr	3	—	—	—	—	—	—		
BNE (rel)	Branch if not = Zero	? Z = 0	REL	26	rr	3	—	—	—	—	—	—		
BPL (rel)	Branch if Plus	? N = 0	REL	2A	rr	3	—	—	—	—	—	—		
BRA (rel)	Branch Always	? 1 = 1	REL	20	rr	3	—	—	—	—	—	—		
BRCLR(opr) (msk) (rel)	Branch if Bit(s) Clear	? M • mm = 0	DIR	13	dd mm rr	6	—	—	—	—	—	—	—	—
			IND,X	1F	ff mm rr	7								
			IND,Y	1F	ff mm rr	8								
BRN (rel)	Branch Never	? 1 = 0	REL	21	rr	3	—	—	—	—	—	—		
BRSET(opr) (msk) (rel)	Branch if Bit(s) Set	? (M) • mm = 0	DIR	12	dd mm rr	6	—	—	—	—	—	—	—	—
			IND,X	1E	ff mm rr	7								
			IND,Y	1E	ff mm rr	8								
BSET (opr) (msk)	Set Bit(s)	M + mm ⇒ M	DIR	14	dd mm	6	—	—	—	—	Δ	Δ	0	—
IND,X	1C	ff mm	7											
IND,Y	1C	ff mm	8											
BSR (rel)	Branch to Subroutine	See Figure 3–2	REL	8D	rr	6	—	—	—	—	—	—	—	
BVC (rel)	Branch if Overflow Clear	? V = 0	REL	28	rr	3	—	—	—	—	—	—		
BVS (rel)	Branch if Overflow Set	? V = 1	REL	29	rr	3	—	—	—	—	—	—		
CBA	Compare A to B	A – B	INH	11	—	2	—	—	—	—	Δ	Δ	Δ	Δ
CLC	Clear Carry Bit	0 ⇒ C	INH	0C	—	2	—	—	—	—	—	—	—	0
CLI	Clear Interrupt Mask	0 ⇒ I	INH	0E	—	2	—	—	—	0	—	—	—	
CLR (opr)	Clear Memory Byte	0 ⇒ M	EXT	7F	hh ll	6	—	—	—	—	0	1	0	0
			IND,X	6F	ff	6								
			IND,Y	6F	ff	7								
CLRA	Clear Accumulator A	0 ⇒ A	A INH	4F	—	2	—	—	—	—	0	1	0	0
CLRB	Clear Accumulator B	0 ⇒ B	B INH	5F	—	2	—	—	—	—	0	1	0	0
CLV	Clear Overflow Flag	0 ⇒ V	INH	0A	—	2	—	—	—	—	—	—	0	—
CMPA (opr)	Compare A to Memory	A – M	A IMM	81	ii	2	—	—	—	—	Δ	Δ	Δ	Δ
			A DIR	91	dd	3								
			A EXT	B1	hh ll	4								
			A IND,X	A1	ff	4								
			A IND,Y	A1	ff	5								
CMPB (opr)	Compare B to Memory	B – M	B IMM	C1	ii	2	—	—	—	—	Δ	Δ	Δ	Δ
			B DIR	D1	dd	3								
			B EXT	F1	hh ll	4								
			B IND,X	E1	ff	4								
			B IND,Y	E1	ff	5								

Table 3-2 Instruction Set (Sheet 3 of 6)

Mnemonic	Operation	Description	Addressing Mode	Instruction			Condition Codes							
				Opcode	Operand	Cycles	S	X	H	I	N	Z	V	C
COM (opr)	Ones Complement Memory Byte	$\$FF - M \Rightarrow M$	EXT IND,X IND,Y	73	hh ll	6	—	—	—	—	Δ	Δ	0	1
				63	ff	6								
				18 63	ff	7								
COMA	Ones Complement A	$\$FF - A \Rightarrow A$	A INH	43	—	2	—	—	—	—	Δ	Δ	0	1
COMB	Ones Complement B	$\$FF - B \Rightarrow B$	B INH	53	—	2	—	—	—	—	Δ	Δ	0	1
CPD (opr)	Compare D to Memory 16-Bit	$D - M : M + 1$	IMM DIR EXT IND,X IND,Y	1A 83	jj kk	5	—	—	—	—	Δ	Δ	Δ	Δ
				1A 93	dd	6								
				1A B3	hh ll	7								
				1A A3	ff	7								
				CD A3	ff	7								
CPX (opr)	Compare X to Memory 16-Bit	$IX - M : M + 1$	IMM DIR EXT IND,X IND,Y	8C	jj kk	4	—	—	—	—	Δ	Δ	Δ	Δ
				9C	dd	5								
				BC	hh ll	6								
				AC	ff	6								
				CD AC	ff	7								
CPY (opr)	Compare Y to Memory 16-Bit	$IY - M : M + 1$	IMM DIR EXT IND,X IND,Y	18 8C	jj kk	5	—	—	—	—	Δ	Δ	Δ	Δ
				18 9C	dd	6								
				18 BC	hh ll	7								
				1A AC	ff	7								
				18 AC	ff	7								
DAA	Decimal Adjust A	Adjust Sum to BCD	INH	19	—	2	—	—	—	—	Δ	Δ	Δ	Δ
DEC (opr)	Decrement Memory Byte	$M - 1 \Rightarrow M$	EXT IND,X IND,Y	7A	hh ll	6	—	—	—	—	Δ	Δ	Δ	—
				6A	ff	6								
				18 6A	ff	7								
DECA	Decrement Accumulator A	$A - 1 \Rightarrow A$	A INH	4A	—	2	—	—	—	—	Δ	Δ	Δ	—
DECB	Decrement Accumulator B	$B - 1 \Rightarrow B$	B INH	5A	—	2	—	—	—	—	Δ	Δ	Δ	—
DES	Decrement Stack Pointer	$SP - 1 \Rightarrow SP$	INH	34	—	3	—	—	—	—	—	—	—	—
DEX	Decrement Index Register X	$IX - 1 \Rightarrow IX$	INH	09	—	3	—	—	—	—	Δ	—	—	—
DEY	Decrement Index Register Y	$IY - 1 \Rightarrow IY$	INH	18 09	—	4	—	—	—	—	Δ	—	—	—
EORA (opr)	Exclusive OR A with Memory	$A \oplus M \Rightarrow A$	A IMM A DIR A EXT A IND,X A IND,Y	88	ii	2	—	—	—	—	Δ	Δ	0	—
				98	dd	3								
				B8	hh ll	4								
				A8	ff	4								
				18 A8	ff	5								
EORB (opr)	Exclusive OR B with Memory	$B \oplus M \Rightarrow B$	B IMM B DIR B EXT B IND,X B IND,Y	C8	ii	2	—	—	—	—	Δ	Δ	0	—
				D8	dd	3								
				F8	hh ll	4								
				E8	ff	4								
				18 E8	ff	5								
FDIV	Fractional Divide 16 by 16	$D / IX \Rightarrow IX; r \Rightarrow D$	INH	03	—	41	—	—	—	—	Δ	Δ	Δ	Δ
IDIV	Integer Divide 16 by 16	$D / IX \Rightarrow IX; r \Rightarrow D$	INH	02	—	41	—	—	—	—	Δ	0	Δ	Δ
INC (opr)	Increment Memory Byte	$M + 1 \Rightarrow M$	EXT IND,X IND,Y	7C	hh ll	6	—	—	—	—	Δ	Δ	Δ	—
				6C	ff	6								
				18 6C	ff	7								
INCA	Increment Accumulator A	$A + 1 \Rightarrow A$	A INH	4C	—	2	—	—	—	—	Δ	Δ	Δ	—
INCB	Increment Accumulator B	$B + 1 \Rightarrow B$	B INH	5C	—	2	—	—	—	—	Δ	Δ	Δ	—
INS	Increment Stack Pointer	$SP + 1 \Rightarrow SP$	INH	31	—	3	—	—	—	—	—	—	—	—
INX	Increment Index Register X	$IX + 1 \Rightarrow IX$	INH	08	—	3	—	—	—	—	Δ	—	—	—

Table 3-2 Instruction Set (Sheet 4 of 6)

Mnemonic	Operation	Description	Addressing Mode	Instruction			Condition Codes											
				Opcode	Operand	Cycles	S	X	H	I	N	Z	V	C				
INY	Increment Index Register Y	$IY + 1 \Rightarrow IY$	INH	18 08	—	4	—	—	—	—	—	—	—	—	—	—	—	—
JMP (opr)	Jump	See Figure 3-2	EXT IND,X IND,Y	7E	hh ll	3	—	—	—	—	—	—	—	—	—	—	—	—
				6E	ff	3												
				6E	ff	4												
JSR (opr)	Jump to Subroutine	See Figure 3-2	DIR EXT IND,X IND,Y	9D	dd	5	—	—	—	—	—	—	—	—	—	—	—	—
				BD	hh ll	6												
				AD	ff	6												
				AD	ff	7												
LDAA (opr)	Load Accumulator A	$M \Rightarrow A$	A A A A A	IMM	86 ii	2	—	—	—	—	—	—	—	—	—	—	—	—
				DIR	96 dd	3												
				EXT	B6 hh ll	4												
				IND,X	A6 ff	4												
				IND,Y	A6 ff	5												
LDAB (opr)	Load Accumulator B	$M \Rightarrow B$	B B B B B	IMM	C6 ii	2	—	—	—	—	—	—	—	—	—	—	—	—
				DIR	D6 dd	3												
				EXT	F6 hh ll	4												
				IND,X	E6 ff	4												
				IND,Y	E6 ff	5												
LDD (opr)	Load Double Accumulator D	$M \Rightarrow A, M + 1 \Rightarrow B$	IMM DIR EXT IND,X IND,Y	CC	jj kk	3	—	—	—	—	—	—	—	—	—	—	—	—
				DC	dd	4												
				FC	hh ll	5												
				EC	ff	5												
				EC	ff	6												
				EC	ff	6												
LDS (opr)	Load Stack Pointer	$M : M + 1 \Rightarrow SP$	IMM DIR EXT IND,X IND,Y	8E	jj kk	3	—	—	—	—	—	—	—	—	—	—	—	—
				9E	dd	4												
				BE	hh ll	5												
				AE	ff	5												
				AE	ff	6												
				AE	ff	6												
LDX (opr)	Load Index Register X	$M : M + 1 \Rightarrow IX$	IMM DIR EXT IND,X IND,Y	CE	jj kk	3	—	—	—	—	—	—	—	—	—	—	—	—
				DE	dd	4												
				FE	hh ll	5												
				EE	ff	5												
				EE	ff	6												
				EE	ff	6												
LDY (opr)	Load Index Register Y	$M : M + 1 \Rightarrow IY$	IMM DIR EXT IND,X IND,Y	18	jj kk	4	—	—	—	—	—	—	—	—	—	—	—	—
				18	DE	5												
				18	FE	6												
				1A	EE	6												
				18	EE	6												
				18	EE	6												
LSL (opr)	Logical Shift Left		EXT IND,X IND,Y	78	hh ll	6	—	—	—	—	—	—	—	—	—	—	—	—
				68	ff	6												
				68	ff	7												
				68	ff	7												
LSLA	Logical Shift Left A		A INH	48	—	2	—	—	—	—	—	—	—	—	—	—	—	—
LSLB	Logical Shift Left B		B INH	58	—	2	—	—	—	—	—	—	—	—	—	—	—	—
LSLD	Logical Shift Left Double		INH	05	—	3	—	—	—	—	—	—	—	—	—	—	—	—
LSR (opr)	Logical Shift Right		EXT IND,X IND,Y	74	hh ll	6	—	—	—	—	—	—	—	—	—	—	—	—
				64	ff	6												
				64	ff	7												
LSRA	Logical Shift Right A		A INH	44	—	2	—	—	—	—	—	—	—	—	—	—	—	—
LSRB	Logical Shift Right B		B INH	54	—	2	—	—	—	—	—	—	—	—	—	—	—	—
LSRD	Logical Shift Right Double		INH	04	—	3	—	—	—	—	—	—	—	—	—	—	—	—
MUL	Multiply 8 by 8	$A * B \Rightarrow D$	INH	3D	—	10	—	—	—	—	—	—	—	—	—	—	—	—
NEG (opr)	Two's Complement Memory Byte	$0 - M \Rightarrow M$	EXT IND,X IND,Y	70	hh ll	6	—	—	—	—	—	—	—	—	—	—	—	—
				60	ff	6												
				60	ff	7												
NEGA	Two's Complement A	$0 - A \Rightarrow A$	A INH	40	—	2	—	—	—	—	—	—	—	—	—	—	—	—
NEGB	Two's Complement B	$0 - B \Rightarrow B$	B INH	50	—	2	—	—	—	—	—	—	—	—	—	—	—	—

Table 3-2 Instruction Set (Sheet 5 of 6)

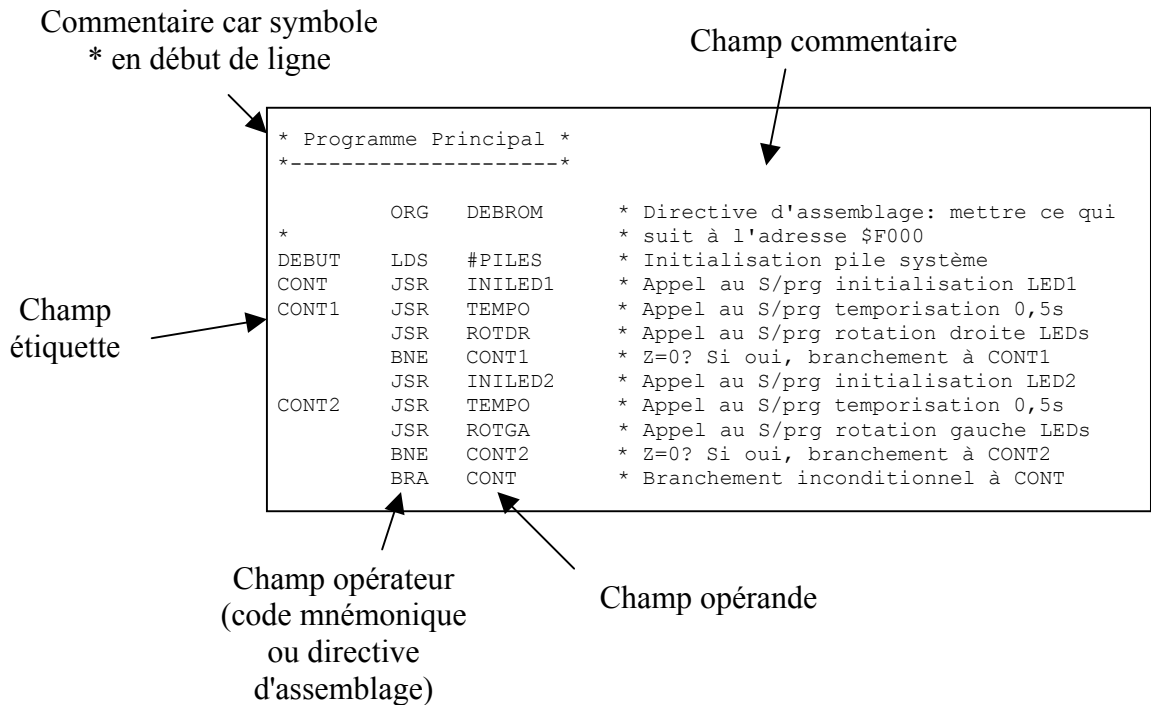
Mnemonic	Operation	Description	Addressing Mode	Instruction			Condition Codes									
				Opcode	Operand	Cycles	S	X	H	I	N	Z	V	C		
NOP	No operation	No Operation	INH	01	—	2	—	—	—	—	—	—	—	—	—	—
ORAA (opr)	OR Accumulator A (Inclusive)	$A + M \Rightarrow A$	A IMM	8A	ii	2	—	—	—	—	Δ	Δ	0	—	—	—
			A DIR	9A	dd	3										
			A EXT	BA	hh ll	4										
			A IND,X	AA	ff	4										
			A IND,Y	AA	ff	5										
ORAB (opr)	OR Accumulator B (Inclusive)	$B + M \Rightarrow B$	B IMM	CA	ii	2	—	—	—	—	Δ	Δ	0	—	—	—
			B DIR	DA	dd	3										
			B EXT	FA	hh ll	4										
			B IND,X	EA	ff	4										
			B IND,Y	EA	ff	5										
PSHA	Push A onto Stack	$A \Rightarrow \text{Stk}, SP = SP - 1$	A INH	36	—	3	—	—	—	—	—	—	—	—	—	—
PSHB	Push B onto Stack	$B \Rightarrow \text{Stk}, SP = SP - 1$	B INH	37	—	3	—	—	—	—	—	—	—	—	—	—
PSHX	Push X onto Stack (Lo First)	$IX \Rightarrow \text{Stk}, SP = SP - 2$	INH	3C	—	4	—	—	—	—	—	—	—	—	—	—
PSHY	Push Y onto Stack (Lo First)	$IY \Rightarrow \text{Stk}, SP = SP - 2$	INH	18 3C	—	5	—	—	—	—	—	—	—	—	—	—
PULA	Pull A from Stack	$SP = SP + 1, A \Leftarrow \text{Stk}$	A INH	32	—	4	—	—	—	—	—	—	—	—	—	—
PULB	Pull B from Stack	$SP = SP + 1, B \Leftarrow \text{Stk}$	B INH	33	—	4	—	—	—	—	—	—	—	—	—	—
PULX	Pull X From Stack (Hi First)	$SP = SP + 2, IX \Leftarrow \text{Stk}$	INH	38	—	5	—	—	—	—	—	—	—	—	—	—
PULY	Pull Y from Stack (Hi First)	$SP = SP + 2, IY \Leftarrow \text{Stk}$	INH	18 38	—	6	—	—	—	—	—	—	—	—	—	—
ROL (opr)	Rotate Left		EXT	79	hh ll	6	—	—	—	—	Δ	Δ	Δ	Δ	—	—
			IND,X	69	ff	6										
			IND,Y	69	ff	7										
ROLA	Rotate Left A		A INH	49	—	2	—	—	—	—	Δ	Δ	Δ	Δ	—	—
ROLB	Rotate Left B		B INH	59	—	2	—	—	—	—	Δ	Δ	Δ	Δ	—	—
ROR (opr)	Rotate Right		EXT	76	hh ll	6	—	—	—	—	Δ	Δ	Δ	Δ	—	—
			IND,X	66	ff	6										
			IND,Y	66	ff	7										
RORA	Rotate Right A		A INH	46	—	2	—	—	—	—	Δ	Δ	Δ	Δ	—	—
RORB	Rotate Right B		B INH	56	—	2	—	—	—	—	Δ	Δ	Δ	Δ	—	—
RTI	Return from Interrupt	See Figure 3-2	INH	3B	—	12	Δ	↓	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ
RTS	Return from Subroutine	See Figure 3-2	INH	39	—	5	—	—	—	—	—	—	—	—	—	—
SBA	Subtract B from A	$A - B \Rightarrow A$	INH	10	—	2	—	—	—	—	Δ	Δ	Δ	Δ	—	—
SBCA (opr)	Subtract with Carry from A	$A - M - C \Rightarrow A$	A IMM	82	ii	2	—	—	—	—	Δ	Δ	Δ	Δ	—	—
			A DIR	92	dd	3										
			A EXT	B2	hh ll	4										
			A IND,X	A2	ff	4										
			A IND,Y	A2	ff	5										
SBCB (opr)	Subtract with Carry from B	$B - M - C \Rightarrow B$	B IMM	C2	ii	2	—	—	—	—	Δ	Δ	Δ	Δ	—	—
			B DIR	D2	dd	3										
			B EXT	F2	hh ll	4										
			B IND,X	E2	ff	4										
			B IND,Y	E2	ff	5										
SEC	Set Carry	$1 \Rightarrow C$	INH	0D	—	2	—	—	—	—	—	—	—	—	—	1
SEI	Set Interrupt Mask	$1 \Rightarrow I$	INH	0F	—	2	—	—	—	1	—	—	—	—	—	—
SEV	Set Overflow Flag	$1 \Rightarrow V$	INH	0B	—	2	—	—	—	—	—	—	—	—	1	—
STAA (opr)	Store Accumulator A	$A \Rightarrow M$	A DIR	97	dd	3	—	—	—	—	Δ	Δ	0	—	—	—
			A EXT	B7	hh ll	4										
			A IND,X	A7	ff	4										
			A IND,Y	A7	ff	5										

Table 3-2 Instruction Set (Sheet 6 of 6)

Mnemonic	Operation	Description	Addressing Mode	Instruction			Condition Codes								
				Opcode	Operand	Cycles	S	X	H	I	N	Z	V	C	
STAB (opr)	Store Accumulator B	$B \Rightarrow M$	B DIR	D7	dd	3	—	—	—	—	Δ	Δ	0	—	
			B EXT	F7	hh ll	4									
			B IND,X	E7	ff	4									
			B IND,Y	E7	ff	5									
STD (opr)	Store Accumulator D	$A \Rightarrow M, B \Rightarrow M + 1$	DIR	DD	dd	4	—	—	—	—	Δ	Δ	0	—	
			EXT	FD	hh ll	5									
			IND,X	ED	ff	5									
			IND,Y	ED	ff	6									
STOP	Stop Internal Clocks	—	INH	CF	—	2	—	—	—	—	—	—	—		
STS (opr)	Store Stack Pointer	$SP \Rightarrow M : M + 1$	DIR	9F	dd	4	—	—	—	—	Δ	Δ	0	—	
			EXT	BF	hh ll	5									
			IND,X	AF	ff	5									
			IND,Y	AF	ff	6									
STX (opr)	Store Index Register X	$IX \Rightarrow M : M + 1$	DIR	DF	dd	4	—	—	—	—	Δ	Δ	0	—	
			EXT	FF	hh ll	5									
			IND,X	EF	ff	5									
			IND,Y	EF	ff	6									
STY (opr)	Store Index Register Y	$IY \Rightarrow M : M + 1$	DIR	18	DF	dd	5	—	—	—	—	Δ	Δ	0	—
			EXT	18	FF	hh ll	6								
			IND,X	1A	EF	ff	6								
			IND,Y	18	EF	ff	6								
SUBA (opr)	Subtract Memory from A	$A - M \Rightarrow A$	A IMM	80	ii	2	—	—	—	—	Δ	Δ	Δ	Δ	
			A DIR	90	dd	3									
			A EXT	B0	hh ll	4									
			A IND,X	A0	ff	4									
			A IND,Y	A0	ff	5									
SUBB (opr)	Subtract Memory from B	$B - M \Rightarrow B$	A IMM	C0	ii	2	—	—	—	—	Δ	Δ	Δ	Δ	
			A DIR	D0	dd	3									
			A EXT	F0	hh ll	4									
			A IND,X	E0	ff	4									
			A IND,Y	E0	ff	5									
SUBD (opr)	Subtract Memory from D	$D - M : M + 1 \Rightarrow D$	IMM	83	jj kk	4	—	—	—	—	Δ	Δ	Δ	Δ	
			DIR	93	dd	5									
			EXT	B3	hh ll	6									
			IND,X	A3	ff	6									
			IND,Y	A3	ff	7									
SWI	Software Interrupt	See Figure 3-2	INH	3F	—	14	—	—	—	1	—	—	—		
TAB	Transfer A to B	$A \Rightarrow B$	INH	16	—	2	—	—	—	—	Δ	Δ	0	—	
TAP	Transfer A to CC Register	$A \Rightarrow CCR$	INH	06	—	2	Δ	\downarrow	Δ	Δ	Δ	Δ	Δ	Δ	
TBA	Transfer B to A	$B \Rightarrow A$	INH	17	—	2	—	—	—	—	Δ	Δ	0	—	
TEST	TEST (Only in Test Modes)	Address Bus Counts	INH	00	—	*	—	—	—	—	—	—	—	—	
TPA	Transfer CC Register to A	$CCR \Rightarrow A$	INH	07	—	2	—	—	—	—	—	—	—	—	
TST (opr)	Test for Zero or Minus	$M - 0$	EXT	7D	hh ll	6	—	—	—	—	Δ	Δ	0	0	
			IND,X	6D	ff	6									
			IND,Y	6D	ff	7									
TSTA	Test A for Zero or Minus	$A - 0$	A INH	4D	—	2	—	—	—	—	Δ	Δ	0	0	
TSTB	Test B for Zero or Minus	$B - 0$	B INH	5D	—	2	—	—	—	—	Δ	Δ	0	0	
TSX	Transfer Stack Pointer to X	$SP + 1 \Rightarrow IX$	INH	30	—	3	—	—	—	—	—	—	—	—	
TSY	Transfer Stack Pointer to Y	$SP + 1 \Rightarrow IY$	INH	18 30	—	4	—	—	—	—	—	—	—	—	
TXS	Transfer X to Stack Pointer	$IX - 1 \Rightarrow SP$	INH	35	—	3	—	—	—	—	—	—	—	—	
TYS	Transfer Y to Stack Pointer	$IY - 1 \Rightarrow SP$	INH	18 35	—	4	—	—	—	—	—	—	—	—	
WAI	Wait for Interrupt	Stack Regs & WAIT	INH	3E	—	**	—	—	—	—	—	—	—	—	
XGDY	Exchange D with X	$IX \Rightarrow D, D \Rightarrow IX$	INH	8F	—	3	—	—	—	—	—	—	—	—	
XGDY	Exchange D with Y	$IY \Rightarrow D, D \Rightarrow IY$	INH	18 8F	—	4	—	—	—	—	—	—	—	—	

VII) Editeur assembleur :

A) Fenêtre de l'éditeur :



On remarque que l'écran se découpe en 4 champs :

- ↪ Champ étiquette
- ↪ Champ opérateur (code mnémorique) ou directive d'assemblage
- ↪ Champ opérande
- ↪ Champ commentaire

Pour passer d'un champ à un autre, il suffit de mettre un espace avec la touche espace ou TAB. Plusieurs espaces accolés correspondent à un seul et même espace. Attention on ne peut sauter des champs, c'est à dire passer du champ étiquette au champ opérande. Seul le champ commentaire est un peu particulier. Si un champ quelconque est rempli, on peut aller directement au champ commentaire. Il y a une exception, lorsqu'aucun champ n'est rempli, l'éditeur considère que toute la ligne est un champ commentaire, d'où la condition de mettre * en début de ligne.

On n'utilisera pas d'étiquette comportant plus de 8 caractères du code ASCII standard.

B) Directives d'assemblage :

- END** Fin du fichier contenant le programme.
- EQU** Permet de donner des équivalences.
Ex: PORTB EQU \$1004.
- ORG** Détermine où le compilateur doit mettre les codes qui suivent cette commande.
Ex: ORG \$1000.
- RMB** Réserve une quantité de mémoire (octet).
Ex: RMB 8
- FCB** Permet de mettre des valeurs d'octets fixes dans des zones mémoire.
- FCD** Permet de mettre des valeurs de mots de 16 bits dans des zones mémoire
- FCC** Permet de mettre des caractères de la table ASCII dans des zones mémoire

Remarque : *FCB, FCD, FCC et RMB s'utilisent avec la commande ORG afin de spécifier les zones mémoires affectées.*