Assembleur 68000

Je tiens à remercier chaleureusement Monsieur **Philippe Meyne**, enseignant à l'IUT GEII d'Evry pour sa participation importante à l'élaboration de ce cours.

Toute action informatique peut être décrite par une séquence d'instructions permettant :

- Transfert : allocation en mémoire et dans les registres internes du processeur.
  Allocation : écrire / lire
  $\alpha \leftarrow 2$ Écrire le nombre 2 dans l'emplacement mémoire réserve α.
- Opérations arithmétiques et logiques (fait en première année).
  Addition, multiplication, inversion de signe…
  ET, OU, NON, XOR, NAND, NOR…
- Branchement
  conditionnels : si <condition> alors aller à
  inconditionnels : aller à
- Appel à sous programme

L'objectif du chapitre est d'étudier quelques instructions de ces différents types.

I.Format d'une instruction assembleur

Assembleur = langage donc règles et syntaxes.

Étiquette ␣ Mnémonique.format ␣ source, destination ␣ commentaires

Étiquette : Facultative, permet de repérer une instruction dans un programme.

Mnémonique : Nom de l'instruction

Format : Taille des données manipulées.

.B	byte	octet
.W	word	mot
.L	long word	mot long

Source : Donnée de départ

Destination : Endroit d'arrivée

La source et la destination peuvent être confondues

Exemple :

DEB ␣ MOVE.B ␣ D0,D1	Transfert l'octet de poids faible du registre D0 dans le registre D1.
␣ NOT.B ␣ D1	Complément à 1 de l'octet de poids faible de D1. Remarque : La source et la destination sont confondues.
␣ BRA ␣ DEB	Saute à l'étiquette DEB

II.Instruction de transfert - Mode d'adressage

Instruction de transfert : MOVE.

Syntaxe : MOVE.<format> ␣ <source>, <destination>

II.1.Adressage direct

La donnée est contenue dans un registre.

MOVE.L "valeur", D0 destination (D0) en adressage direct
Le registre D0 est initialisé avec la "valeur".

MOVE.B "valeur", D0 seul l'octet de poids faible de D0 est modifié.
Si valeur = AF :

D0 avant	FA23FBED	par exemple
D0 après	FA23FBAF	

II.2.Adressage immédiat

La donnée est contenue dans l'instruction.

MOVE.W #523,D0 le mot de poids faible de D0 est initialisé avec 523.

immédiat↗ ↖direct

Dans l'exemple, quelle est la base de 532 ? Par défaut base 10.

Notations :	#	$	hexadécimal
	donnée ↗	%	binaire
		&	décimal

⚠ Piège : MOVE.B #$D0, D0 transfert de la donnée $D0=208 (décimal) dans le registre D0.

II.3.Adressage absolu

L'adresse est contenue dans l'instruction.

MOVE.B #$F0, $FF9001 Écrit la donnée F0 à l'adresse FF9001.

⚠ Pour un mot, l'adresse doit être paire : MOVE.W #$2A1D, $FF9002

.L

1D	FF9003
2A	FF9002

II.4.Adressage indirect

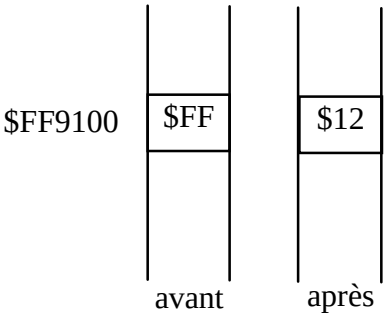
L'adresse de la source ou de la destination est contenue dans un pointeur d'adresses.

MOVE.B D0, (A_3)

La source est le contenu du registre D0. ↗ | ↖ La destination est l'adresse pointée par le pointeur A_3 (adresse effective).

D0 B21E4112

A3 00FF9100



Le contenu du pointeur d'adresses est l'adresse effective de la donnée transférée.

II.5.Adressage indirect avec déplacement

L'adresse effective est la somme du contenu du pointeur d'adresses et d'un déplacement fixe.

MOVE.W #$F14D, $70($A_4$).

A4 00FF9100

Adresse effective :	FF9100	(A4)
	+ 70	70
	FF9170	

II.6.Adressage indirect indexé avec déplacement

L'adresse effective est la somme du contenu du pointeur d'adresses, d'un registre d'index et d'un déplacement fixe.

déplacement ↘ ↙ Registre d'index .L ou .W, pas .B

MOVE.L D7, $70($A_2$, D_1.L)

Pointeur ↗

Adresse effective	00FF6000	A_2
	00001A0C	D_1.L
	00000070	déplacement
	00FF9A7C	l'adresse est bien paire

Déplacement : 8 bits si indexé + déplacement
16 bits si déplacement seul

II.7.Adressage indirect post incrémenté

L'adresse effective de la source ou de la destination est contenue dans un pointeur d'adresses. Le pointeur est incrémenté à la fin de l'instruction.

NOT.B (A_3)+ A_3 = FF9600 avant l'instruction = AE.
A_3 = FF9601 après l'instruction.

Incrémentation :
1 octet
2 mot
4 mot long

NOT.L (A_3)+ A_3 = FF9600 avant l'instruction = AE.
A_3 = FF9604 après l'instruction.

Sert à la gestion des piles.

II.8.Adressage indirect pré décrémenté

L'adresse effective de la source ou de la destination est contenue dans un pointeur d'adresses. Le pointeur est décrémenté au début de l'instruction.

Sert à la gestion des piles.

II.9.Principe d'extension du signe

Adresse effective sur 24 bits (A_{23}…A_0).

Pb : Si l'adresse n'est pas sur 24 bits, elle est complétée en extension de signe.

NOT.L $F000 AE : FFF000
NOT.L $7000 AE : 007000

 Piège : (An, Dm)

↖ contenu en .w, extension de signe

NOT.W (A_3, D_2) A_3 = FF9300 D_2 = FF0700

AE : FF9300
000700
FF9A00

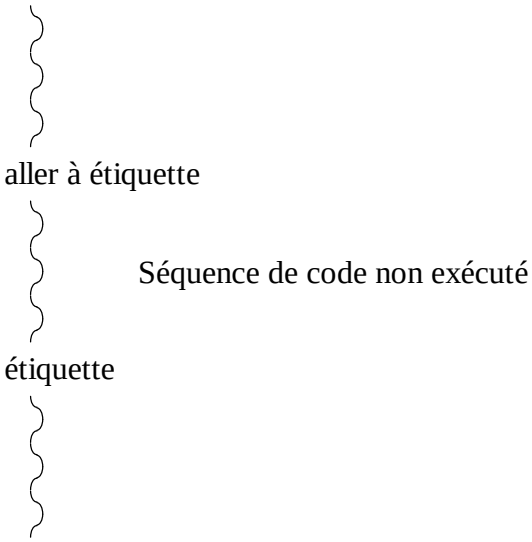
III.Instructions de branchement

Deux manières d'effectuer un branchement inconditionnel :

Inconditionnel JMP, BRA

Conditionnel Bcc, (DBcc)

III.1.Branchement inconditionnel



Instructions :

BRA étiquette Déroutement relatif.

PC ← PC + étiquette

↖ Déplacement : nombre d'instructions entre l'endroit du branchement et l'endroit où le programme doit se brancher.

JMP étiquette Déroutement absolu.

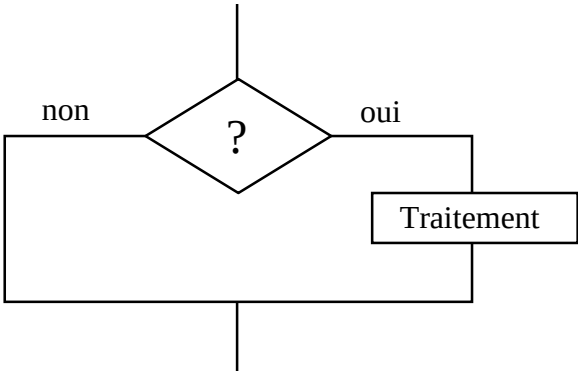
PC ← étiquette

↖ Adresse de branchement.

⚠ JMP génère un code non relogeable => déconseillé.

III.2.Branchement conditionnel

Intérêt : Dérouter un programme en fonction d'une condition.



Structure : si <condition> alors <traitement>.

III.2.1.Le registre code condition

Contient les informations qui vont permettre le test.

Poids faible du registre d'état : SR (Status Register).

Regsitre système								Registre code condition							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T		S			I_2	I_1	I_0				X	N	Z	V	C

Mode trace
Etat supervieur
Masque d'interruption
Extension
Retenue
Dépassement
Zéro
Négatif

Les bits du regsitre code condition sont positionnés par :

- les instructions arithmétiques et logiques,
- certaines instructions de transfert (calcul d'adresse),
- l'instruction de comparaison : CMP,
- l'instruction de test de bit : BTST.

Lecture de la doc : MOVE, ADD…

Exemple :

```
MOVE.B #$1, D0          * D0 ← 1
SUB.B  #$1, D0          * D0 ← D0-1
                              +  Z = 1
                              +  C = 0
SUB.B  #$1, D0          * D0 ← D0-1
                              +  Z = 0
                              +  C = 1
                              +  N = 1
```

Reste à tester ces bits pour effectuer ou non un branchement.

```
Bcc        cc =  EQ     (EQual)
                 NE     (Not Equal)
                 …      (voir §IV.III.2.4)
```

III.2.2.L'instruction CMP

Elle permet de comparer 2 données. Elle positionne les bits du registre code condition.

Syntaxe : CMP.<format> <source>, <destination>
<destination> - <source>, mais ne modifie pas la destination.

Exemple :

```
MOVE.B #$0A, D0
MOVE.B #$0B, D1
CMP.B  D1, D0
          D0 - D1 < 0
                +  N = 1
                +  Z = 0
                +  V = 0
                +  C = 1
CMP.B  #$0A, D0
          D0 - #$0A = 0
                +  N = 0
                +  Z = 1
                +  V = 0
                +  C = 0
```

CMP.B D0, D1

D1 - D0 > 0

\+ N = 0

\+ Z = 0

\+ V = 0

\+ C = 0

III.2.3.L'instruction BTST

Elle permet de tester un bit d'un regsitre. Elle postionne l'indicateur Z.

Syntaxe : BTST <n°de bit>, AE (Adresse Effective)

Si le bit est à 0 alors Z = 1.

III.2.4.Les instructions de branchement conditionnel

Elles permettent d'effectuer un branchement en fonction du registre de conditions.

Syntaxe : Bcc etiquette (cc : code condition)

Remarque : il s'agit d'un branchement relatif.

Les différentes conditions :

CC	retenue à 0	$\overline{C}$	
CS	retenue à 1	C	
EQ	égal	Z	
NE	non égal	$\overline{Z}$	
GE	supérieur ou égal	$N.V + \overline{N}.\overline{V}$	Arithmétique signée
GT	supérieur	$N.V.\overline{Z} + \overline{N}.\overline{V}.\overline{Z}$	
HI	plus grand	$\overline{C}.\overline{Z}$	Arithmétique non signée
LE	inférieur ou égal	$Z + N.\overline{V} + \overline{N}.V$	Arithmétique signée
LT	inférieur	$N.\overline{V} + \overline{N}.V$	
LS	plus petit ou égal	$C + Z$	Arithmétique non signée
MI	négatif	N	
PL	positif	$\overline{N}$	
VC	pas de dépassement	$\overline{V}$	
VS	dépassement	V	

( Pour la soustraction, le bit C (carry) est le complément de celui trouvé dans la soustraction avec la convention signée. Ce choix est cohérent avec la retenue utilisée pendant l'opération de soustraction.

Exemple 1 :

MOVE.B	#$0A,D0	X		
MOVE.B	#$0B,D1	Y		
CMP.B	D0,D1	Y-X = #$1	N=0, Z=0, V=0, C=0	
			BGT vrai	Y>X
			BLE faux	Y ≤ X
			BHI vrai	Y>X
			BLS faux	Y ≤ X
CMP.B	D1,D0	Y-X = #$FF	N=0, Z=0, V=0, C=0	
			BGT faux	~~Y>X~~
			BLE vrai	Y ≤ X
			BHI faux	~~Y>X~~
			BLS vrai	Y ≤ X

)

III.2.5.Les instructions de test et de branchement (détail)

Elles permettent d'effectuer un branchement en fonction du registre de conditions en intégrant un comptage..

Syntaxe : DBcc D_n,etiquette (cc : code condition)
Si (condition fausse) alors $D_n \leftarrow D_n$-1
Si ($D_n \neq -1$) alors PC ← PC + d (deplacement étiquette)
sinon PC ← PC + 2

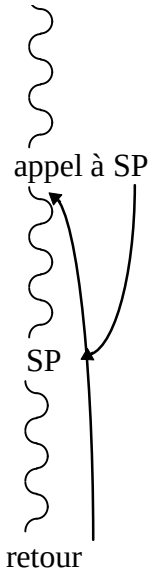
Exemple : Boucle de temporisation

```
tempo   MOVE.W   #$FFFF,D0
bou     DBEQ     D0,bou
        …

bou     Si (Z ≠ 0)      alors   D0 ← D0-1
        Si (Dn ≠ −1)    alors   bou
        Suite du programme
```

IV.Instruction d'appel de sous programme

Principe :



Branche au sous programme

Une fois le sous programme terminé, le programme revient juste après l'instruction d'appel.

Instruction précieuse : Conception modulaire.
Séquence de code répétitive.

Instructions :

BSR étiquette Branchement relatif.
PC retour sauvegardé
PC ← PC + étiquette
↖ Déplacement : nombre d'instructions entre l'endroit du branchement et l'endroit où le programme doit se brancher.

JSR étiquette Branchement absolu.
PC retour sauvegardé
PC ← étiquette
↖ Adresse de branchement.

 JSR génère un code non relogeable => déconseillé.

Instruction de retour : RTS.