Architecture et programmation des ordinateurs :

6- Microprocesseurs & Jeu d'instruction

Architecture des ordinateurs :

Microprocesseur & jeu d'instructions – (1)

1- Description d'un microprocesseur

Un microprocesseur se présente sous la forme d'un circuit intégré muni d'un nombre généralement important de broches.

Exemples :

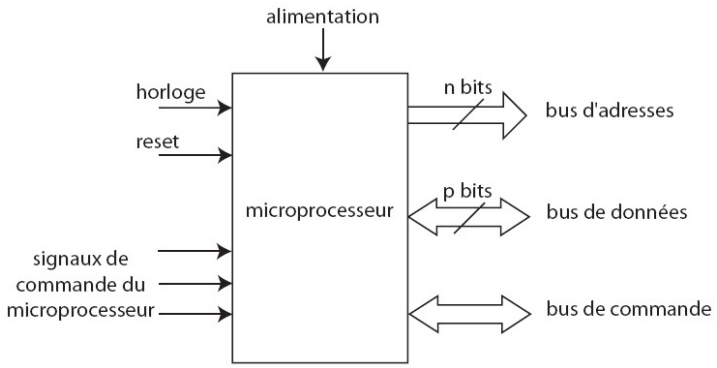
- *Intel 8085, 8086 : 40 broches, DIP (Dual In-line Package) ;*
- *Motorola 68000 : 64 broches, DIP ;*
- *Intel 80386 : 196 broches, PGA (Pin Grid Array).*

Technologies de fabrication : NMOS, PMOS, **CMOS**.

Architecture des ordinateurs :

Microprocesseur & jeu d'instructions – (2)

2- Organisation fonctionnelle externe d'un microprocesseur



Architecture des ordinateurs :

Microprocesseur & jeu d'instructions – (3)

3- Organisation fonctionnelle interne d'un microprocesseur

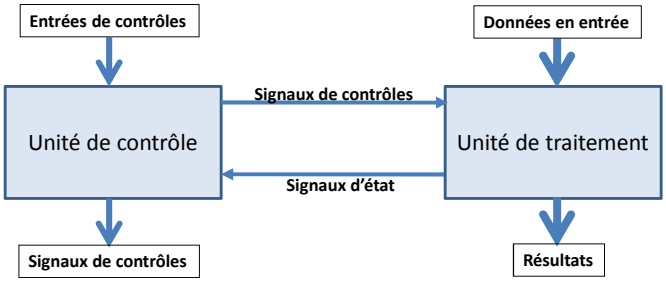
LI	DI	LO	EX	RR

LI : **L**ecture de l'**I**nstruction codée
DI : **D**écodage de l'**I**nstruction
LO : **L**ecture de des **O**pérandes
EX : **E**xécution
RR : **R**angement du résultat

Remarque :
Unité de contrôle a en charge les différentes phases de l'exécution d'une instruction

Architecture des ordinateurs :
Microprocesseur & jeu d'instructions – (4)

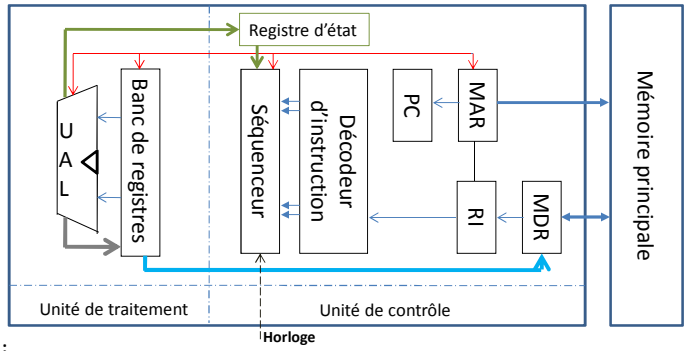
3- Organisation fonctionnelle interne d'un microprocesseur



Remarques :
L'unité de traitement = éléments de mémorisation (registres internes) + UAL
Il s'agit d'une machine séquentielle

Architecture des ordinateurs :
Microprocesseur & jeu d'instructions – (5)

3- Organisation fonctionnelle interne d'un microprocesseur

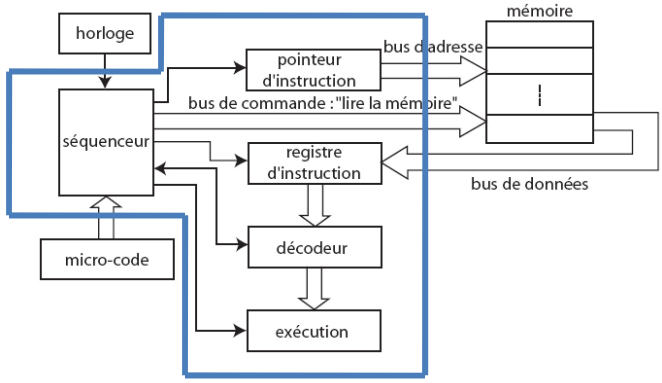


Remarques :
Banc de registres :: opérandes ; MAR -> RI :: adresse opérande
PC :: adresse instruction ; MDR -> MP :: Résultats

Architecture des ordinateurs :

Microprocesseur & jeu d'instructions – (6)

3- Organisation fonctionnelle interne d'un microprocesseur



Architecture des ordinateurs :

Microprocesseur & jeu d'instructions – (9)

4- Fonctionnement d'un microprocesseur - BASIC

Un microprocesseur exécute un programme. Le programme est une suite d'instructions stockées dans la mémoire.

Une instruction peut être codée sur un ou plusieurs octets.

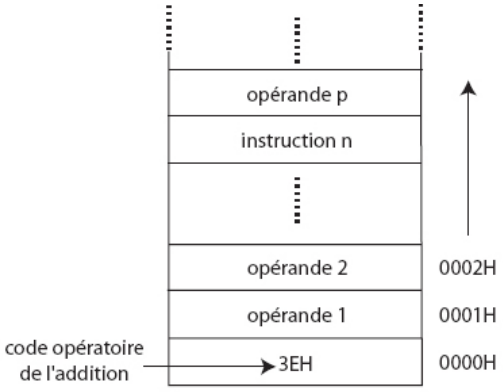
Format d'une instruction :

Opération à effectuer	Opérande 1 , Opérande 2
ADDITIONNER	Case mémoire 1 , Case mémoire 2

Architecture des ordinateurs :

Microprocesseur & jeu d'instructions– (10)

5- Fonctionnement d'un microprocesseur



Architecture des ordinateurs :

Microprocesseur & jeu d'instructions – (11)

5- Fonctionnement d'un microprocesseur

Pour exécuter les instructions dans l'ordre établi par le programme, le microprocesseur doit savoir à chaque instant l'adresse de la prochaine instruction à exécuter.

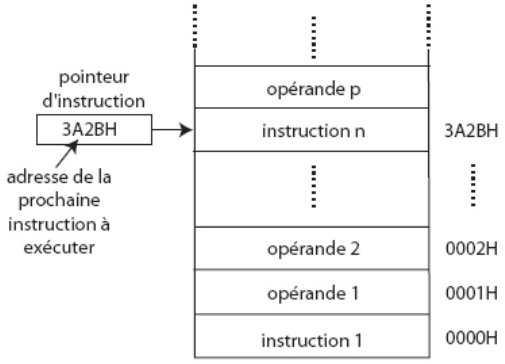
Le microprocesseur utilise un registre contenant cette information.

Ce registre est appelé **pointeur d'instruction** (**IP : Instruction Pointer**) **ou compteur d'instructions ou compteur ordinal.**

Architecture des ordinateurs :

Microprocesseur & jeu d'instructions – (12)

5- Fonctionnement d'un microprocesseur



Architecture des ordinateurs :

Microprocesseur & jeu d'instructions – (13)

5- Fonctionnement d'un microprocesseur

Remarque : la valeur initiale du pointeur d'instruction est fixée par le constructeur du microprocesseur.

Elle vaut une valeur bien définie à chaque mise sous tension du microprocesseur ou bien lors d'une remise à zéro (reset).

Pour savoir quel type d'opération doit être exécuté (addition, soustraction, ...),
le microprocesseur lit le premier octet de l'instruction pointée par le pointeur d'instruction (**code opératoire**) et le range dans un registre appelé **registre d'instruction**.

Architecture des ordinateurs :
Microprocesseur & jeu d'instructions – (13)

5- Fonctionnement d'un microprocesseur

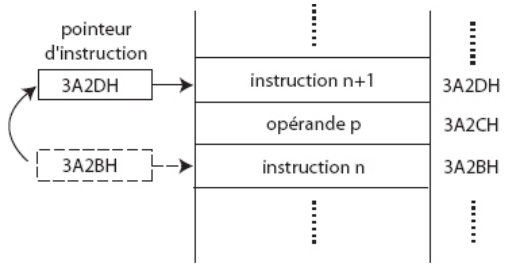
Le **code opératoire** est décodé par des circuits de décodage contenus dans le microprocesseur. Des signaux de commande pour l'**UAL** sont produits en fonction de l'opération demandée qui est alors exécutée.

Remarque : pour exécuter une instruction, l'UAL utilise **des registres de travail**.

Architecture des ordinateurs :
Microprocesseur & jeu d'instructions – (14)

5- Fonctionnement d'un microprocesseur

Pendant que l'instruction est décodée, le pointeur d'instruction est incrémenté de façon à pointer vers l'instruction suivante :



Puis le processus de lecture et de décodage des instructions recommence.

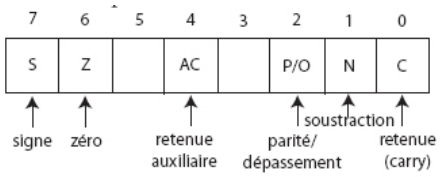
Architecture des ordinateurs :

Microprocesseur & jeu d'instructions – (15)

5- Fonctionnement d'un microprocesseur

A la suite de chaque instruction, un registre du microprocesseur est actualisé en fonction du dernier résultat : c'est le **registre d'état** du microprocesseur.

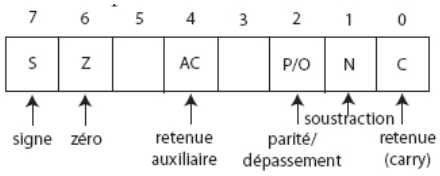
Chacun des bits du registre d'´etat est un **indicateur d'état** ou **flag (drapeau)**.



Architecture des ordinateurs :

Microprocesseur & jeu d'instructions – (16)

5- Fonctionnement d'un microprocesseur



Exemple : registre d'état du **microprocesseur Z80**
Les indicateurs d'état sont activés lorsqu'une certaine condition est remplie,
Exemple :
le flag **Z** est mis à **1** lorsque la dernière opération a donné un résultat nul,
le flag **C** est mis à **1** lorsque le résultat d'une addition possède une retenue

Architecture des ordinateurs :

Microprocesseur & jeu d'instructions – (17)

6- Le microprocesseur MOTOROLA 68000 - Introduction

Le **M68000** est un microprocesseur CISC 16/32 bits sorti en 1979.

Le processeur le plus prisé de sa génération. Le formidable succès des machines comme le Mac, l'Amiga ou l'Atari est en grande partie due à cet innovant processeur.

Voici ces caractéristiques :

- Mémoire adressable de **16Mo** (24 lignes d'adresse).
- Jeu d'instruction très complet (**56 Instructions**).
- **14** modes d'adressage différents.
- **8** registres de donnée 32 bits.
- **7** registres d'adresse 32 bits.
- Gestion de privilèges (Utilisateur / Superviseur).
- Deux pointeurs de pile **32 bits** différents.

Architecture des ordinateurs :

Microprocesseur & jeu d'instructions – (17-1)

6- Le microprocesseur MOTOROLA 68000 - Introduction

Le terme **16/32** bits résume la façon dont le circuit manipule les données.
Le bus de données interne est sur **32 bits** mais le processeur communique avec le reste du monde sur un bus externe de **16 bits**.

La série "microprocesseur 680x0" se compose du 68010, 68020, 68030, 68040 et 68060. Ces processeurs constituent la base des versions microcontrôleurs 683xx.

Architecture des ordinateurs :

Microprocesseur & jeu d'instructions – (17-2)

6- Le microprocesseur MOTOROLA 68000 - Introduction

Tous les registres du 68000 sont sur **32 bits** excepté le registre d'état **CCR** **16 bits**.

Petit rappel sur les formats de données :

- Quartet = 4 bits
- Octet = 8 bits
- Mot / Word = 16 bits
- Long mot / Long Word = 32 bits

Les registres de données et d'adresses sont banalisés à l'exception de A7 aussi appelé **SP** (Stack pointer) qui représente le *pointeur de pile*.

En interne, il existe en fait deux pointeurs différents l'**USP** (***User Stack Pointer***) en mode utilisateur et le ***SSP*** (***Supervisor Stack Pointer***) en mode superviseur.

Il s'agit toujours du registre **A7**, le 68000 utilisant **implicitement** l'un ou l'autre en fonction de son mode de fonctionnement.

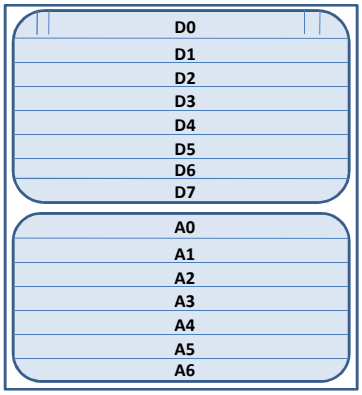
Architecture des ordinateurs :

Microprocesseur & jeu d'instructions – (17-3)

6- Le microprocesseur MOTOROLA 68000 – Les registres

b31 : bit de poids fort

8 Registre de données
sur 32 bits



b0 : bit de poids faible

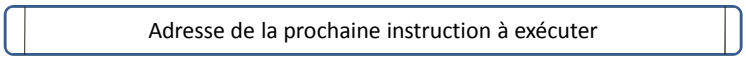
7 Registre d'adresse
sur 32 bits

Architecture des ordinateurs :

Microprocesseur & jeu d'instructions – (17-4)

6- Le microprocesseur MOTOROLA 68000 – Les registres

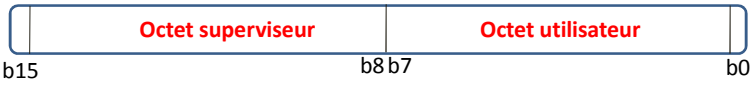
Registre PC : Programm Counter ou compteur de programme / CO



b31 : bit de poids fort

b0 : bit de poids faible

Registre SR : Status Register ou registre d'état



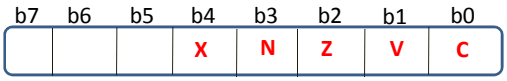
Architecture des ordinateurs :

Microprocesseur & jeu d'instructions – (17-5)

6- Le microprocesseur MOTOROLA 68000 – Les registres

Registre SR : Status Register ou registre d'état

1. Octet utilisateur / Registre CCR (condition Code Register)



Bit C: positionné si une retenue est générée.
Bit V: positionné s'il y a dépassement arithmétique.
Bit Z: positionné si le résultat d'une opération est nul.
Bit N: positionné si le résultat d'une opération est négatif.
Bit X: bit de retenue pour les opérations de calculs étendues.

Architecture des ordinateurs :

Microprocesseur & jeu d'instructions – (17-6)

6- Le microprocesseur MOTOROLA 68000 – Les registres

Registre SR : Status Register ou registre d'état

2. Octet Superviseur :

b15	b14	b13	b12	b11	b10	b9	b8
T		S			I2	I1	I0

bit T : s'il est positionné met le processeur en mode trace.
Dans ce mode, le processeur s'arrête après l'exécution de chaque instruction et génère une exception permettant, par exemple, la mise en place d'un programme de débogage.

bit S : définit le mode de fonctionnement du processeur,
0 pour le **mode Utilisateur** et **1** pour le **mode Superviseur**.

Architecture des ordinateurs :

Microprocesseur & jeu d'instructions – (17-7)

6- Le microprocesseur MOTOROLA 68000 – Les registres

Registre SR : Status Register ou registre d'état

2. Octet Superviseur :

b15	b14	b13	b12	b11	b10	b9	b8
T		S			I2	I1	I0

Les bits **I0** , **I1** et **I2** représentent le niveau du masque d'interruption.
Les interruptions dont le niveau est inférieur ou égal au masque ne sont pas prisent en compte.

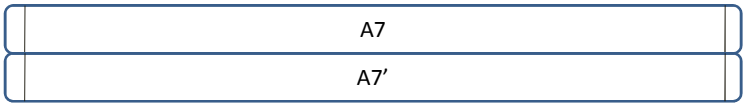
une exception est un évènement matériel (interruptions) ou logiciel qui déroute le processeur vers une procédure particulière.

Architecture des ordinateurs :

Microprocesseur & jeu d'instructions – (17-8)

6- Le microprocesseur MOTOROLA 68000 – Les registres

Registre SP : Stack Pointer ou registre pointeur de pile (A7)



Si S=1, le microprocesseur est en mode superviseur
-> le pointeur de la pile utilisé est **A7'**, appelé **SSP** : **Supervisor Stack Pointer**

Si S=0, le microprocesseur est en mode utilisateur
-> le pointeur de la pile utilisé est **A7**, appelé **USP** : **User Stack Pointer**

Architecture des ordinateurs :

Microprocesseur & jeu d'instructions – (17-9)

6- Le microprocesseur MOTOROLA 68000 – Gestion de la pile

Le fonctionnement de la **pile** est un élément fondamental pour le fonctionnement du processeur.

La **pile** est en fait une portion de mémoire dont l'adresse se trouve dans **A7** et qui permet la sauvegarde temporaire de données.

On parle d'**empilement** lors d'une **écriture** dans la pile et de **dépilement** lors d'une **lecture**.

Ce processus est utilisé, soit automatiquement par le microprocesseur lors de l'appel d'un sous-programme par exemple, soit par le programmeur.

Architecture des ordinateurs :

Microprocesseur & jeu d'instructions – (17-9)

6- Le microprocesseur MOTOROLA 68000 – Gestion de la pile

Pour bien comprendre le fonctionnement d'un pile, on peut imaginer la pile comme une pile d'assiettes de différentes couleurs.

En prenant comme condition initiale **A7=10**, on empile une assiette bleu, une assiette **rouge** puis une assiette **verte**.

L'assiette bleu sera stockée en position **10**, la rouge en **9** et la verte en **8**, au final A7 vaut **8**.

L'opération de dépilement est identique à l'empilement mais à l'envers, en récupère l'assiette **verte**, la **rouge** puis la **bleu** et A7 revient à sa position initiale soit 10.

Architecture des ordinateurs :

Microprocesseur & jeu d'instructions – (17-10)

6- Le microprocesseur MOTOROLA 68000 – Gestion de la pile

Lors du processus d'empilement la première assiette empilé est **bleu**, lors du dépilement c'est la dernière.

C'est pour cette raison que l'on parle de pile **LIFO** (**Last In First Out**) le dernier élément empilé sera le premier dépilé.

Pour information, il existe un autre type de pile appelée pile **FIFO** (**First In First Out**). Dans ce cas le premier élément empilé sera le premier dépilé.

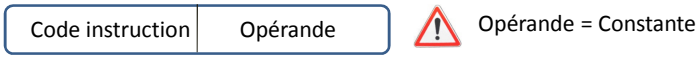
Architecture des ordinateurs :

Microprocesseur & jeu d'instructions – (17-11)

6- Le microprocesseur MOTOROLA 68000 – Mode d'adressage – Contexte général

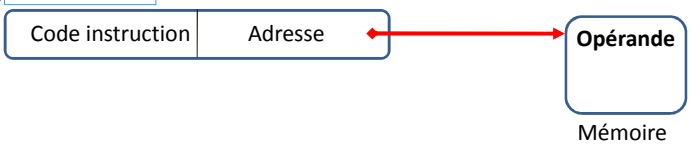
Le type de donnée contenu dans le champ opérandes caractérise le mode d'adressage.

1 **Immédiat**



Opérande = Constante

2 **Direct absolu**

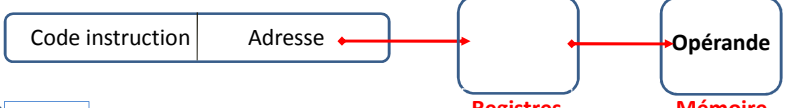


Architecture des ordinateurs :

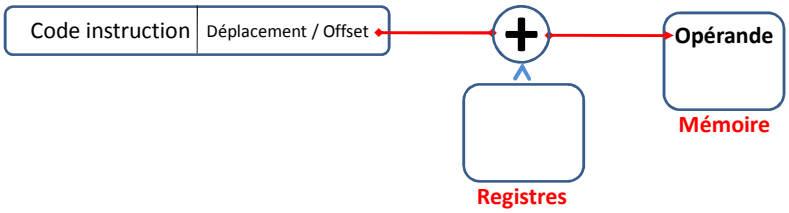
Microprocesseur & jeu d'instructions – (17-12)

6- Le microprocesseur MOTOROLA 68000 – Mode d'adressage – Contexte général

3 **Indirect par registre**



4 **Relatif**

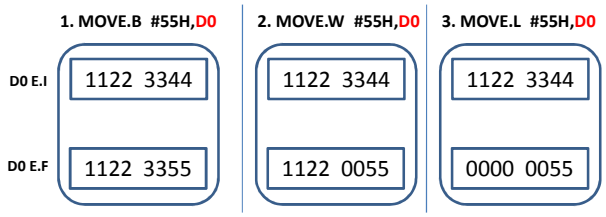


Architecture des ordinateurs :

Microprocesseur & jeu d'instructions – (17-14)

6- Le microprocesseur MOTOROLA 68000 – Mode d'adressage

A **Adressage immédiat** — **Destination : registre de données**



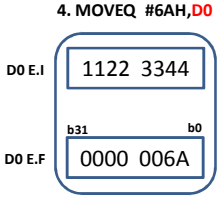
Taille de données transférées : .B : Byte : un octet / .W : Word : Mot de 16 bits / .L : Long word : Mot de 32 bits

Architecture des ordinateurs :

Microprocesseur & jeu d'instructions – (17-15)

6- Le microprocesseur MOTOROLA 68000 – Mode d'adressage

A **Adressage immédiat rapide** — **Destination : registre de données**



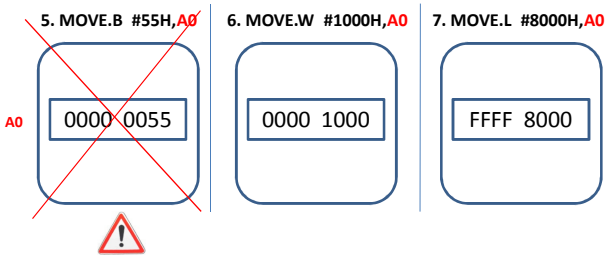
Taille de données transférées : .B : Byte : un octet UNIQUEMENT

Architecture des ordinateurs :

Microprocesseur & jeu d'instructions – (17-16)

6- Le microprocesseur MOTOROLA 68000 – Mode d'adressage

A **Adressage immédiat** — **Destination : registre d'adresse**



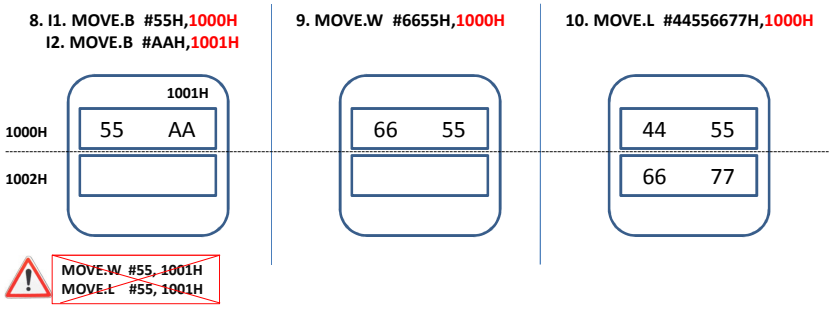
Taille de données transférées : .W : Word : Mot de 16 bits / .L : Long word : Mot de 32 bits UNIQUEMENT

Architecture des ordinateurs :

Microprocesseur & jeu d'instructions – (17-17)

6- Le microprocesseur MOTOROLA 68000 – Mode d'adressage

A **Adressage immédiat** — **Destination : mémoire principale**



Taille de données transférées : .B : Byte : un octet / .W : Word : Mot de 16 bits / .L : Long word : Mot de 32 bits

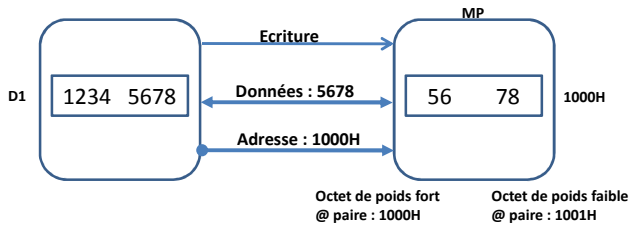
Architecture des ordinateurs :

Microprocesseur & jeu d'instructions – (17-18)

6- Le microprocesseur MOTOROLA 68000 – Mode d'adressage

B **Adressage Direct** — **Destination : mémoire principale**

11. MOVE.W D1,1000H



Taille de données transférées : .B : Byte : un octet / .W : Word : Mot de 16 bits / .L : Long word : Mot de 32 bits

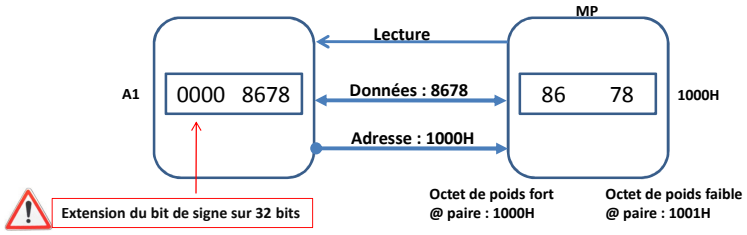
Architecture des ordinateurs :

Microprocesseur & jeu d'instructions – (17-19)

6- Le microprocesseur MOTOROLA 68000 – Mode d'adressage

B **Adressage Direct** — **Destination : registre d'adresse**

12. MOVE.W 1000H, A1



Taille de données transférées : :.W : Word : Mot de 16 bits / .L : Long word : Mot de 32 bits UNIQUEMENT

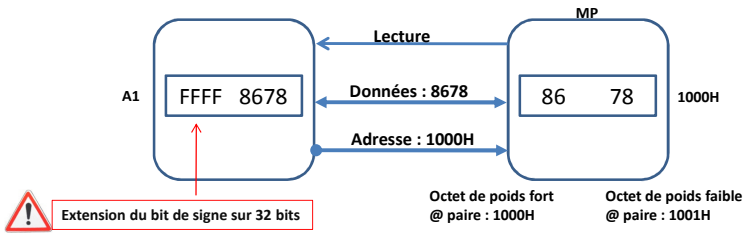
Architecture des ordinateurs :

Microprocesseur & jeu d'instructions – (17-19)

6- Le microprocesseur MOTOROLA 68000 – Mode d'adressage

B **Adressage Direct** — **Destination : registre d'adresse**

12. MOVE.L 1000H, A1



Taille de données transférées : :.W : Word : Mot de 16 bits / .L : Long word : Mot de 32 bits UNIQUEMENT

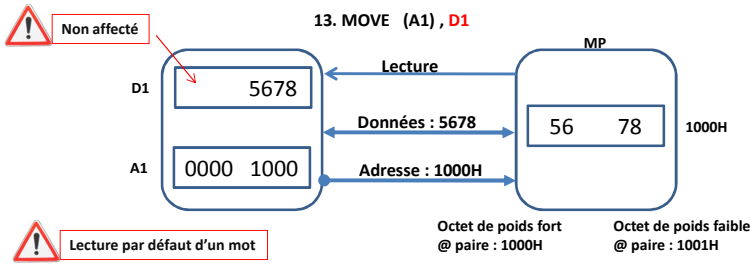
Architecture des ordinateurs :

Microprocesseur & jeu d'instructions – (17-20)

6- Le microprocesseur MOTOROLA 68000 – Mode d'adressage

B **Adressage Indirect par Registre**

13. MOVE (A1) , D1



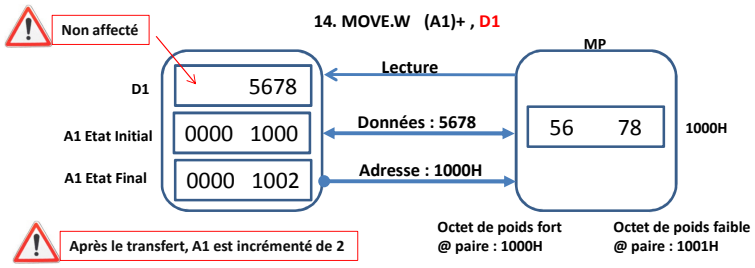
Taille de données transférées : .B : Byte : un octet / .W : Word : Mot de 16 bits / .L : Long word : Mot de 32 bits

Architecture des ordinateurs :

Microprocesseur & jeu d'instructions – (17-21)

6- Le microprocesseur MOTOROLA 68000 – Mode d'adressage

B **Adressage Indirect par Registre / Pos-incrémentation**



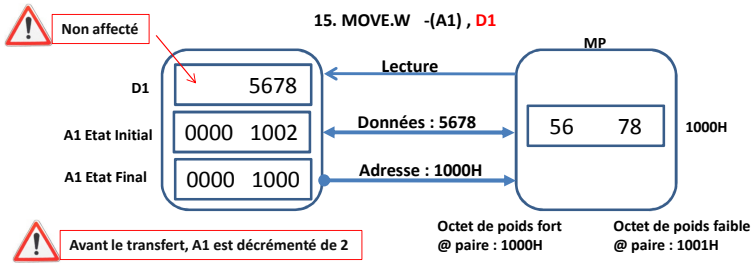
Taille de données transférées : .B : Byte : un octet / .W : Word : Mot de 16 bits / .L : Long word : Mot de 32 bits

Architecture des ordinateurs :

Microprocesseur & jeu d'instructions – (17-22)

6- Le microprocesseur MOTOROLA 68000 – Mode d'adressage

B **Adressage Indirect par Registre / Pré-décrémentation**



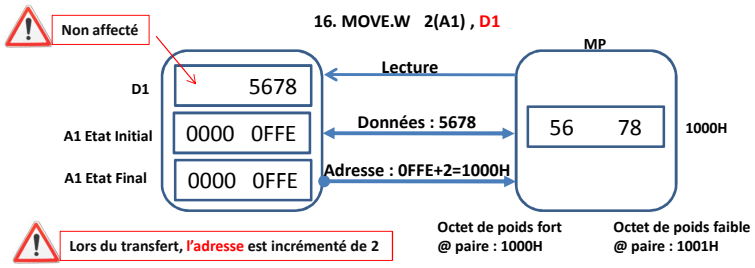
Taille de données transférées : .B : Byte : un octet / .W : Word : Mot de 16 bits / .L : Long word : Mot de 32 bits

Architecture des ordinateurs :

Microprocesseur & jeu d'instructions – (17-23)

6- Le microprocesseur MOTOROLA 68000 – Mode d'adressage

B **Adressage Indirect par Registre avec déplacement**



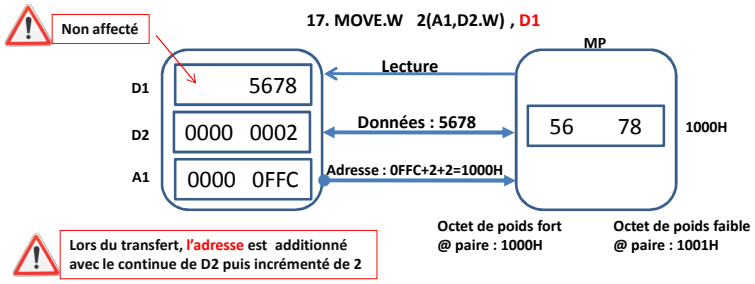
Taille de données transférées : .B : Byte : un octet / .W : Word : Mot de 16 bits / .L : Long word : Mot de 32 bits

Architecture des ordinateurs :

Microprocesseur & jeu d'instructions – (17-24)

6- Le microprocesseur MOTOROLA 68000 – Mode d'adressage

B **Adressage Indirect par Registre avec déplacement sur 8 bits signés et index court :**



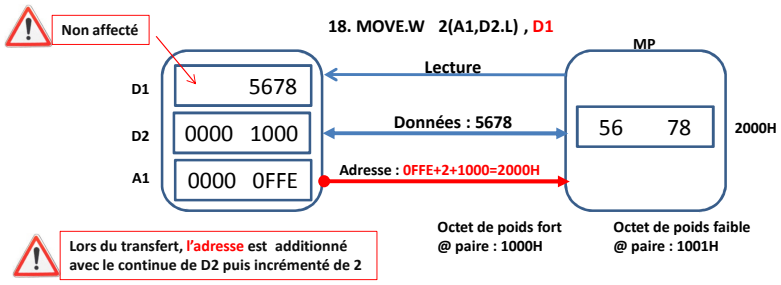
Taille de données transférées : .B : Byte : un octet / .W : Word : Mot de 16 bits / .L : Long word : Mot de 32 bits

Architecture des ordinateurs :

Microprocesseur & jeu d'instructions – (17-25)

6- Le microprocesseur MOTOROLA 68000 – Mode d'adressage

B **Adressage Indirect par Registre avec déplacement sur 8 bits signés et index long :**



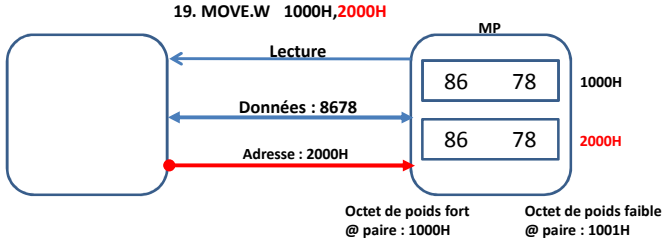
Taille de données transférées : .B : Byte : un octet / .W : Word : Mot de 16 bits / .L : Long word : Mot de 32 bits

Architecture des ordinateurs :

Microprocesseur & jeu d'instructions – (17-26)

6- Le microprocesseur MOTOROLA 68000 – Mode d'adressage

C **Adressage Absolu Court :**



Taille de données transférées : .B : Byte : un octet / .W : Word : Mot de 16 bits / .L : Long word : Mot de 32 bits

Architecture des ordinateurs :

Microprocesseur & jeu d'instructions – (17-27)

6- Le microprocesseur MOTOROLA 68000 – Jeu d'instructions – transfert de données

INSTRUCTION	TAILLE DE L'OPÉRANDE	OPÉRATION
EXG	32	Rx ⟷ Ry
LEA	32	EA → An
LINK	–	An → SP @ –
		SP → An
		SP+d → SP
MOVE	8, 16, 32	(EA)s → EAd
MOVEM	16, 32	(EA) → An, Dn
		An, Dn → EA
MOVEP	16, 32	(EA) → Dn
		Dn → EA
MOVEQ	8	#xxx → Dn
PEA	32	EA → SP @ –
SWAP	32	Dn [31:16] ⟷ D [15:0]
UNLK	–	An → SP
		SP @ + → An

Architecture des ordinateurs :

Microprocesseur & jeu d'instructions – (17-28)

6- Le microprocesseur MOTOROLA 68000 – Jeu d'instructions – Instruction arithmétique

INSTRUCTION	TAILLE DE L'OPÉRANDE	OPÉRATION
ADD	8, 16, 32	DN + (AE) → DN
		(AE) + DN → AE
		(AE) + # DONNÉE → AE
		AN + (AE) → AN
ADX	8, 16, 32	Dx + Dy + X → Dx
	16, 32	[–(Ax)] + [–(Ay)] + X → – (Ax)
CLR	8, 16, 32	0 → AE
CMP	8, 16, 32	DN – (AE)
		(AE) – # DONNÉE
		[(Ax)+] – [(Ay)+]
	16, 32	AN – (AE)
DIVS	32 ÷ 16	DN ÷ (AE) → DN
DIVU	32 ÷ 16	DN ÷ (AE) → DN
EXT	8 → 16	EXTENSION BIT 7 SUR BITS 8 A 15
	16 → 32	EXTENSION BIT 15 SUR BITS 16 A 31
MULS	16 × 16 → 32	DN × (AE) → DN
MULU	16 × 16 → 32	DN × (AE) → DN
NEG	8, 16, 32	0 – (AE) → AE
NEGX	8, 16, 32	0 – (AE) – X → AE
SUB	8, 16, 32	DN – (AE) → DN
		(AE) – DN → AE
		(AE) – # DONNÉE → AE
		AN – (AE) → AN
SUBX	16, 32	
	8, 16, 32	Dx – Dy – X → Dx
		[–(Ax)] – [–(Ay)] – X → – (Ax)
TAS	8	(AE) – 0 et BIT 7 de AE – 1
TST	8, 16, 32	(AE) – 0

Architecture des ordinateurs :

Microprocesseur & jeu d'instructions – (17-29)

6- Le microprocesseur MOTOROLA 68000 – Jeu d'instructions – Opérations logiques

INSTRUCTION	TAILLE DE L'OPÉRANDE	OPÉRATION
AND	8, 16, 32	DN ET (AE) → DN (AE) ET DN → AE (AE) ET # DONNÉE → AE
OR	8, 16, 32	DN OU (AE) → DN (AE) OU DN → AE (AE) OU # DONNÉE → AE
EOR	8, 16, 32	(AE) OU exclusif Dy → AE (AE) OU exclusif # donnée → AE
NOT	8, 16, 32	COMPLÉMENT A UN DE (AE) → AE

Architecture et programmation des ordinateurs :

~Microprocesseur & jeu d'instructions~

Travaux Dirigés

Architecture des ordinateurs :

Travaux dirigés

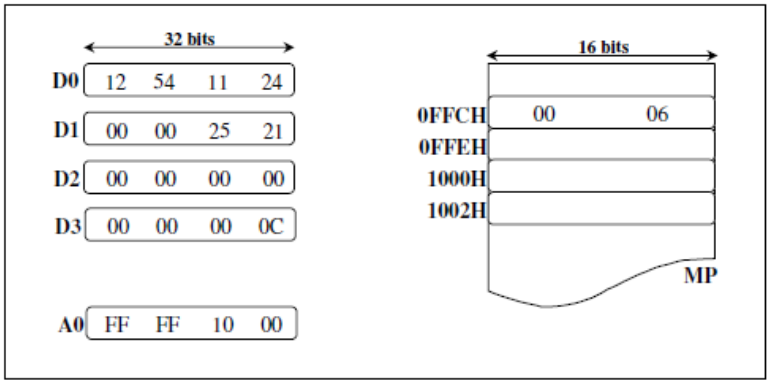
Exercice 1 : On désire analyser une partie du code source « **exam.s** »

```
MOVE.W   #02H,D0
MOVE.L   #0FFCH,A0
MOVE.W   #2135H,1000H
MOVE.L   #05H,D2
MOVE.W   (A0)+ ,D1
ADD.B    D1,D2
MOVE.W   D2,1002H
MOVE.W   2(A0,D0.W),D3
MOVE.W   D0,D2
MULU     D1,D2
MOVE.W   D2,1000H
.
.
```


Architecture des ordinateurs :

Travaux dirigés

Exercice 1 : L'état initial des registres et de la mémoire est indiqué par la figure suivante :



Architecture des ordinateurs :
Travaux dirigés

Exercice 1 : Compléter le tableau, en décrivant l'exécution de chaque instruction. Préciser également le contenu des registres ou de la mémoire suivant le cas.

I1	MOVE.W #02H,D0	Description : ▪ D0 reçoit 02h ▪ Seuls les deux premiers octets sont modifiés. ▪ D0 = 1254**0002**
I2	MOVE.L #0FFCH,A0	Description : ▪ …… ▪ A0 = ……
I3	MOVE.W #2135H,1000H	Description : ▪ …… ▪ @1000H = ……
I4	MOVE.L #05H,D2	Description : ▪ …… ▪ D2=……
I5	MOVE.W (A0)+ ,D1	Description : ▪ …… ▪ D1=…… ▪ A0=……

Architecture des ordinateurs :
Travaux dirigés

Exercice 1 : Compléter le tableau, en décrivant l'exécution de chaque instruction. Préciser également le contenu des registres ou de la mémoire suivant le cas.

I6	ADD.B D1,D2	Description : ▪ …… ▪ D2=……
I7	MOVE.W D2,1002H	Description : ▪ …… ▪ @1002H=
I8	MOVE.W 2(A0,D0.W),D3	Description : ▪ …… ▪ D3=…… ▪ A0=……
I9	MOVE.W D0,D2	Description : ▪ …… ▪ D2=……
I10	MULU D1,D2	Description : ▪ …… ▪ D2=……
I11	MOVE.W D2,1000H	Description : ▪ …… ▪ @1000H=……

Architecture et programmation des ordinateurs :
Motorola 68000

Travaux pratiques Easy68K

Architecture des ordinateurs :
Travaux pratiques

Exercice 1 : On souhaite calculer la somme de 45 et 85.

```
        ORG     $1000
START:                          ; first instruction of program
        MOVE.B  #45,D0
        ADD.B   #85,D0

        MOVE.B  #9,D0
        TRAP    #15             ; halt simulator

* Variables and Strings

        END     START           ; last line of source
```

Architecture des ordinateurs :

Travaux pratiques

Exercice 2 : Calculer la somme de (45+12-48+18).

```
            ORG     $1000
START:                              ; first instruction of program

            MOVE.W  #45,D0
            ADD.W   #12,D0
            ADD.W   #-48,D0
            ADD.W   #18,D0

            MOVE.B  #9,D0
            TRAP    #15             ; halt simulator

•Variables and Strings

            END     START           ; last line of source
```


Architecture des ordinateurs :

Travaux pratiques

Exercice 3 : Calculer la même somme en mettant les données dans un tableau puis les additionner. Pour construire ce tableau, on utilise la directive DC.

```
            ORG     $1000
START:                              ; first instruction of program
            LEA TAB(PC),A0          ; A0 pointe à la première donnée
            MOVE.W (A0)+,D0         ; D0 contient 45
            ADD.W (A0)+,D0          ; D0 contient 57
            ADD.W (A0)+,D0          ; D0 contient 9
            ADD.W (A0)+,D0          ; D0 contient 27

            MOVE.B  #9,D0
            TRAP    #15             ; halt simulator

TAB         DC.W 45,12,-48,18

            END     START           ; last line of source
```

Architecture des ordinateurs :

BIBLIOGRAPHIE

1. Aho A, Concepts fondamentaux de l'informatique, Dunod, Paris, 1993
2. Tanenbaum A., Architecture des ordinateurs, InterEditions, Paris, 1996
3. Vieillfond C., Mise en œuvre du 68000, Sybex, 1984