LE MICROPROCESSEUR 16 BITS

MOTOROLA

68000

Le Microprocesseur Motorola 68000 est le fruit de l'évolution technologique des concepteurs Motorola. Après le 6800 apparu en 1974, le 6802 en 1976, le 6809 en 1977, les premiers échantillons du 68000 ont été diffusés en 1979. Bien que son bus de données soit de 16 bits, la structure interne du 68000 est 32 bits, c'est le circuit charnière entre les 8 bits de la famille 6800 et les 32 bits de la famille 68020.

L'idée maîtresse des concepteurs Motorola a toujours été la simplicité, en effet les circuits de ce fabricant ont toujours suivi la même logique. Cependant le 68000 a dû se plier notamment à l'évolution de la conception des modes d'échange qui sont passés du mode synchrone unique pour la famille 6800 au mode asynchrone pour le 68020. Le 68000 assure ces deux modes d'échange lui permettant d'utiliser les périphériques de l'un comme de l'autre. Le mode d'échange asynchrone utilise la technique dite de la poignée de main (handshake):

- Le microprocesseur place la donnée sur le bus
- Validation de la donnée par le microprocesseur
- Acquisition de la donnée par le périphérique
- Acquittement par le périphérique
- Le microprocesseur passe à la tâche suivante

Le 68000 est un pas vers la rationalité qui cependant ne sera pleinement atteinte qu'avec le 68020 ainsi, le 68000 peut travailler sur :

- Un bit
- Un octet
- Un mot de 16 bits
- Un mot long de 32 bits

cependant il n'accepte pas que les mots et mots longs soient situés à des adresses impaires.

Le 68000 facilite également la création de machines complexes telles que des stations de travail,

- d'une part en possédant deux modes de fonctionnement hiérarchiquement distincts le mode superviseur et le mode utilisateur
- d'autre part en étendant la notion d'interruption à la notion d'exception ce qui permet:
  . de faciliter l'analyse des erreurs de manipulation et améliorer de la convivialité des machines
  . de faciliter la conception de systèmes de développement
  . de faciliter le fonctionnement en mémoire virtuelle

L'extension de son jeu d'instructions et de ses modes d'adressage en font un circuit performant et facile à programmer.

I - MODÈLE DE PROGAMMATION

Le modèle de programmation est l'ensemble des registres auxquels le programmeur peut se référer

8 **REGISTRES DE DONNÉES** de 32 bits
qui peuvent être réduits à l'octet de poids faible (bits de 0 à 7) ou le mot de poids faible (bits de 0 à 15)

31 16	15 8	7 0	Appellation
			D0
			D1
			D2
			D3
			D4
			D5
			D6
			D7

7 **REGISTRES D'ADRESSES** de 32 bits
qui peuvent être réduits (partiellement) au mot de poids faible (bits de 0 à 15) **mais pas à l'octet**

31 16	15 0	Appellation
		A0
		A1
		A2
		A3
		A4
		A5
		A6

Un registre 32 bits à la fois **POINTEUR DE PILE UTILISATEUR (USP)** et registre d'adresses A7

31 0	Appellation
	USP=A7

Un registre 32 bits à la fois **POINTEUR DE PILE SUPERVISEUR (SSP)** et registre d'adresses A7'

31 0	Appellation
	SSP=A7'

Un **COMPTEUR DE PROGRAMME** 32 bits
dont seuls les 24 bits de poids faible sont utilisables

31 23	0	Appellation
		PC

Un **REGISTRE D'ÉTAT (SR)** de 16 Bits incluant le **REGISTRE DE CODE CONDITION (CCR)**
dans ses 8 bits de poids faible

15 8	7 0	Appellation
	(CCR)	**SR**

I - 1 - Registre d'état (Status register - SR)
C'est un registre 16 bits dont les 8 bits de poids faible constituent le Registre de Codes Condition(CCR). **Les 8 bits de poids fort ne sont accessibles qu'en mode superviseur.**

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

Etat du registre après un RESET (0x2700) -
(0x signifie que le nombre qui suit est exprimé en hexadécimal)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**0**	**1**	**0**	**0**	**1**	**1**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

Fonctions des bits du Registre d'état:

Bit 15 **T** **Validation du mode trace**

Avant chaque instruction le bit T est testé, si celui-ci est à 1, le MPU entrera dans le traitement d'une exception **après** l'exécution de l'instruction. Ceci facilite la mise en œuvre d'un programme d'aide au développement incluant le fonctionnement pas à pas du microprocesseur. L'état du bit T ne peut être modifié qu'en mode superviseur.

Bit 13 **S** **Modes de fonctionnement**

L'état du bit S détermine le niveau de hiérarchie dans lequel travaille le microprocesseur

S = 1 Le MPU(Micro Processor Unit) est dans le mode superviseur, il est dans le mode de niveau hiérarchique le plus élevé, il peut utiliser tout le jeu d'instructions. Son pointeur de pile est A7' (SSP)

S = 0 Le MPU est dans le mode Utilisateur, il ne peut exécuter les instructions privilégiées son pointeur de pile est A7 (USP).

Modification du bit S

Comme l'ensemble des bits de l'octet de poids fort, le bit S ne peut être modifié que dans le mode superviseur, c'est à dire que l'on ne peut que faire passer le bit **S de 1 à 0**. Ceci pourra se faire à l'aide des instructions agissant sur **SR** telles que MOVE, ANDI, EORI To SR. L'instruction RTE, qui restitue le registre d'état après le traitement d'une exception peut également modifier S. Le passage du bit **S de 0 à 1** ne peut se faire qu'à la faveur d'une exception en effet nous verrons plus loin que les exceptions sont traitées en mode superviseur. En résumé:

S=0 → S=1 ne peut se faire qu'à la faveur d'une exception

S=1 → S=0 à l'aide des instructions MOVE, ANDI, EORI To SR et RTE.

Bits 10, 9 et 8 **I2, I1, I0** **Masque d'interruption**

Le masque d'interruption fixe le niveau au-dessus duquel les demandes d'interruptions seront acceptées par le microprocesseur. Une demande d'interruption s'effectue par l'intermédiaire des broches IPL 2,1,0 du microprocesseur (actives à 0), pour qu'une demande d'interruption soit prise en compte par le microprocesseur il faut que le nombre (complémenté) appliqué sur les broches IPL soit strictement supérieur à celui du masque. Seule une interruption de niveau 7 sera prise en compte quelque soit l'état du masque.

L'interruption de niveau 7 est une interruption non masquable

Après un RESET le masque d'interruption est positionné au niveau 7 (1 1 1) , il peut être modifié par programme en mode superviseur. Lors de la prise en compte d'une exception le masque se positionne au niveau de l'interruption traitée pendant la durée du traitement de celle-ci. Il revient au niveau antérieur après le retour d'exception.

Octet accessible à l'utilisateur, Registre de code condition (CCR)

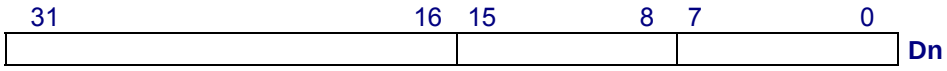
Le registre de code condition est influencé par l'exécution des instructions. Il n'est influencé que par les bits utiles de l'opérande, c'est à dire que si l'opérande est un octet dans un registre 32 bits, seuls les 8 bits de l'octet sont pris en compte, le bit N par exemple sera l'image du bit7 :

Bit 4	**X**	EXTENSION	Bit de report, voisin de C, il n'est influencé que par les opérations à caractère arithmétique
Bit 3	**N**	NEGATIF	Bit de signe en complément à deux, plus généralement, le bit de poids fort du résultat
Bit 2	**Z**	ZERO	Passe à l'état actif (1) lorsque le résultat donne 0
Bit 1	**V**	OVERFLOW	Indicateur de dépassement en code complément à 2. Quatre cas peuvent positionner le bit V à 1 dans son fonctionnement classique: Addition d'un nombre positif à un nombre positif résultat négatif Addition d'un nombre négatif à un nombre négatif résultat positif Soustraction d'un nombre négatif à un nombre positif résultat négatif Soustraction d'un nombre positif à un nombre négatif résultat positif D'autres instructions telles que la division peuvent avoir un effet significatif sur le bit V
Bit 0	**C**	CARRY	Report ou retenue

I - 2 - Registres de Données (Data registers - D0 à D7)

Le 68000 possède 8 registres de données, ces registres ont une dimension de 32 bits.
Les instructions peuvent porter sur:

- un octet, les bits concernés sont alors de 0 à 7, les autres bits du registre ne sont pas influencés.
- un mot, bits de 0 à 15, les autres bits du registre ne sont pas influencés.
- un mot long, bits de 0 à 31



I - 3 - Registres d'adresses (Address registers - A0 à A7)

Il y a 8 registres d'adresses, cependant le registre A7 est particulier puisqu'il sert de pointeur de pile, nous en parlerons au paragraphe suivant. Les registres d'adresses servent comme leur nom l'indique à l'adressage. Certaines instructions peuvent s'appliquer au contenu de ces registres cependant elles ne pourront pas porter sur un octet. Dans la plupart des cas, lorsque l'instruction porte sur un mot qui doit être reçu par un registre d'adresses, il y a extension du bit de signe. L'extension du signe consiste à recopier dans les bits 16 à 31 la valeur du bit de signe du résultat c'est à dire le bit 15.

I - 4 - Pointeurs de pile (Stack pointer - A7 et A7')

Le registre d'adresses A7 est utilisé comme pointeur de pile. En fait ce registre est double l'un est utilisé en mode utilisateur, il s'appellera dans ce cas **A7 ou USP**. Il sera utilisé systématiquement lors des appels à sous-programme ou l'exécution des instructions LINK ou PEA en mode utilisateur.

> *Contrairement à ce que l'on pourrait penser à priori, l'instruction **MOVE USP est une instruction privilégiée** et ne peut être utilisée qu'en mode Superviseur. C'est une curiosité de ce microprocesseur puisque l'instruction Move de ou vers A7 en mode utilisateur est parfaitement possible et assure la même fonction.*

Le registre **A7' ou SSP** est le pointeur de pile Superviseur, on ne peut y accéder que dans ce mode. Il est le seul utilisé dans le traitement des exceptions qui s'exécutent toujours dans le mode superviseur. Ce registre est chargé à la suite d'un Reset matériel par le mot long contenu à l'adresse 0. Son contenu pourra comme précédemment être modifié grâce à l'instruction Move vers A7 exécuté en mode superviseur.

I - 5 - Compteur de programme (Program Counter - PC)

C'est un registre de 32 bits qui a la fonction habituelle du compteur de programme ou compteur ordinal (appellation qui a tendance à disparaître). Il est chargé lors d'un Reset matériel par le mot long situé à l'adresse 4. Son contenu peut être modifié par les instructions de saut ou de branchement. Seuls les bits de A1 à A23 sont répercutés sur le bus d'adresse le bit A0 pilote les lignes LDS et UDS (Lower ou Upper Data Strobe) ces lignes sont actives à 0

parité de l'adresse Æ	A0 = 0		A0 = 1	
Ø taille du transfert	LDS	UDS	LDS	UDS
OCTET	1	0	0	1
MOT	0	0		
MOT LONG	0	0		

II- PRÉSENTATION PHYSIQUE DU 68000

II - 1 - Les boitiers

Le 68000 se présente sous différents boîtiers

- Quad pack 68 connexions, boîtier carré de 25 millimètres de coté pour le montage en surface. Les connexions sont réparties sur la périphérie du composant au pas de 1,27 mm.
- Pin Grid Array (PGA ou boîtier fakir) 68 connexions, boîtier carré de 27 millimètres de coté. Le boîtier retourné se présente comme un lit de broches celles-ci étant disposées suivant une grille au pas de 2,54 mm.
- Dual In Line (DIL ou DIP) 64 broches boîtier rectangulaire de 81 mm sur 20 mm muni d'une double rangée de broches au pas de 2,54 mm.

Les quatre broches supplémentaires des boîtiers carrés sont imposées par la géométrie et ne sont pas connectées.

Nous décrivons dans le paragraphe suivant le brochage du boîtier DIL. Les caractéristiques mécaniques sont décrites page 127.

II - 2 - Le brochage (voir le boîtier page 127)

N°	E/S	NOM	FONCTION
1 à 5	E/S	**D4 à D0**	5 bits de poids faible du bus de données (le reste de 54 à 64)
6	S 3états	**$\overline{AS}$**	Address Strobe, à l'état 0 signale une adresse valide sur le bus
7	S 3états	**$\overline{UDS}$**	Upper Data Strobe Validation de l'octet de poids fort du bus de données
8	S 3états	**$\overline{LDS}$**	Lower Data Strobe Validation de l'octet de poids faible du bus de données
9	S 3états	**R/$\overline{W}$**	Read / Write indique si le MPU fait une opération de lecture (1) ou d'écriture (0) R/W UDS LDS FONCTION 0 0 0 Écriture d'un mot, deux octets valides sur le bus 0 0 1 Écriture de l'octet de poids fort (*) 0 1 0 Écriture de l'octet de poids faible (*) 1 0 0 Lecture d'un mot 1 0 1 Lecture de l'octet de poids fort 1 1 0 Lecture de l'octet de poids faible X 1 1 Pas de donnée valide sur le bus (*) Le même octet est présent sur les parties haute et basse du bus de données
10	E	**$\overline{DTACK}$**	Data Transfer Acknowledge, reconnaissance de transfert de donnée. Actionnée par le circuit périphérique indique: R/$\overline{W}$ =1 $\overline{DTACK}$ =0 la donnée venant du périphérique est valide R/$\overline{W}$ =0 $\overline{DTACK}$ =0 la donnée a été lue par le périphérique
11	S	**$\overline{BG}$**	Bus Grant le MPU signale que les bus seront disponibles à la fin du cycle en cours pour un accès direct à mémoire ou pour un autre MPU
12	E	**$\overline{BGACK}$**	Bus Grant Acknowledge le MPU ou le contrôleur DMA indique qu'il prend possession des bus
13	E	**$\overline{BR}$**	Bus Request, un MPU ou un contrôleur DMA demande la maîtrise des bus
14 et 49		**+Vcc (*)**	Alimentation 5 Volts En version ordinaire pour F clk de 8 à 12 MHz; La puissance absorbée est de 1,1 à 1,7 W pour la technologie MOS et 2,2 W à 16 MHz. Elle tombe à 0,13 et 0,26 W pour la technologie HCMOS

E = Entrée **S =** Sortie **S3états =** Sortie 3 états (0, 1, X-haute impédance-)

N°	E/S	NOM	FONCTION
15	E	**Clk**	Clock - Horloge - Charge 1 unité TTL F de 4 à 16 MHz suivant le modèle
16 et 53		**Gnd (*)**	0 Électrique - Masse -
17	E/S	**$\overline{\text{HALT}}$** Entrée sortie à drain ouvert	Mise à 0 seule, le MPU termine le cycle en cours puis place ses lignes 3 états en Haute Impédance. Les lignes de contrôle deviennent inactives sauf BR, BG, BGACK. Mise à 0 avec la broche Reset, initialisation matériel du 68000 En sortie cette broche à 0 signale que le MPU est arrêté
18	E/S	**$\overline{\text{RESET}}$** Entrée sortie à drain ouvert	Mise à 0 avec la broche Halt, initialisation matériel du 68000 (voir le traitement de l'exception reset page 28) En sortie, cette broche peut être mise à 0 par l'exécution de l'instruction reset
19	S	**$\overline{\text{VMA}}$**	Sortie utilisée en mode synchrone, indique qu'une adresse est valide
20	S	**E**	Signal de Synchronisation sa Fréquence est F Clk ⌈⌉ 10 (6 périodes à 0, 4 périodes à 1) utilisé en particulier avec les périphériques 6800
21	E	**$\overline{\text{VPA}}$**	Un état actif placé sur cette broche signale au MPU qu'il communique avec un périphérique synchrone
22	E	**$\overline{\text{BERR}}$**	Bus Error un état actif sur cette broche signale au MPU une erreur bus, l'instruction avorte et une exception est déclenchée
23, 24 et25	E	**$\overline{\text{IPL2}}$** **$\overline{\text{IPL1}}$** **$\overline{\text{IPL0}}$**	Entrées de demande d'interruption 0 état actif: IPL2 IPL1 IPL0 = 111 pas de demande d'interruption 110 demande de niveau 1 000 demande de niveau 7
26, 27 et 28	S	**FC2, FC1, FC0**	Sorties indiquant la tâche en cours d'exécution **FC** **FC** **2 1 0** FONCTION **2 1 0** FONCTION 0 0 0 Indéfinie 1 0 0 Indéfinie 0 0 1 Donnée Utilisateur 1 0 1 Donnée Superviseur 0 1 0 Progr Utilisateur 1 1 0 Progr Superviseur 0 1 1 Indéfinie 1 1 1 Traitement Interruption
29 à 52	S 3états	**A1 à A23**	Bus d'adresses - A0 est remplacé par UDS et LDS (voir page 7) Capacité d'adressage 2^{24} Octets soit 16 MégaOctets
53 et 16		**GND ***	0 Électrique de l'alimentation
54 à 64	E/S 3états	**D15 à D5**	Onze bits de poids fort du bus de données

* Les alimentations Vcc et Gnd multiples doivent en principe toutes être connectées de façon à bien irriguer la puce. Il ne faut pas perdre de vue que la version MOS peut absorber des pointes de courant de 1,5 Ampères.

III - ORGANISATION DES DONNÉES EN MÉMOIRE

Le 68000 admet les trois tailles de données ci-dessous.

- Octet
- Mot (2 octets)
- Mot Long (4 octets)

et données décimales. Certaines instructions peuvent ne concerner qu'un seul bit cependant l'opération s'effectuera par la lecture et la réécriture de l'opérande modifié contenant ce bit (l'opérande devra donc être accessible en lecture et en écriture).
L'adresse N d'un Mot Long est l'adresse de son octet de poids fort ainsi que de son mot de poids fort. L'adresse du mot de poids faible prend la valeur N+2, l'octet de poids faible N+3.

> ***Une donnée mot ou mot long, de même qu'une instruction est nécessairement alignée sur les frontières de mot (adresses multiples de 2)***

Considérons le mot long d'adresse 0, il est constitué de 2 mots, les mots d'adresse 0 et d'adresse 2 ou de quatre octets d'adresse 0, 1, 2, 3 comme représenté ci-dessous:

31 24	23 16	15 8	7 0
MOT LONG 0 = mot long d'adresse 0x000000 qui contient les mots ou les octets ci-dessous			
MOT 0 = mot d'adresse 0x000000		**MOT 1** = mot d'adresse 0x000002	
OCTET 0 adresse 0x000000	**OCTET 1** adresse 0x000001	**OCTET 2** adresse 0x000002	**OCTET 3** adresse 0x000003
MOT LONG 1 = mot long d'adresse 0x000004 qui contient les mots ou les octets ci-dessous			
MOT 2 = mot d'adresse 0x000004		**MOT 3** = mot d'adresse 0x00000006	
OCTET 4 adresse0x000004	**OCTET 5** adresse 0x000005	**OCTET 6** adresse 0x000006	**OCTET 7** adresse 0x000007

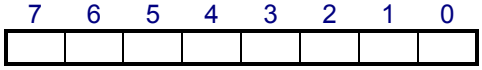
Le bus d'adresses ne comporte pas le bit A0, celui-ci est remplacé par les lignes UDS et LDS. Les mémoires étant généralement structurées en mots de 8 bits, un système 68000 possèdera en général deux mémoires RAM, l'une chargée d'accueillir les octets de poids fort, d'adresse paire, dont le Chip Select sera commandé par la ligne UDS et l'autre les octets de poids faible, d'adresse impaire, commandée par LDS (voir page suivante)

Lorsqu'une opération porte sur un octet d'adresse paire, UDS est actif, sur un octet d'adresse impaire LDS est actif à son tour. Si l'opération porte sur un mot les deux octets sont alors concernés UDS et LDS doivent alors être actifs simultanément. On voit que **cette technique interdit des opérations sur des mots d'adresse impaire puisque UDS et LDS ne peuvent plus faire la différence**.

Lors de l'écriture d'un octet en mémoire, cet octet est placé par le microprocesseur sur la partie haute et sur la partie basse du bus de données, c'est UDS ou LDS qui sélectionne la mémoire d'accueil.

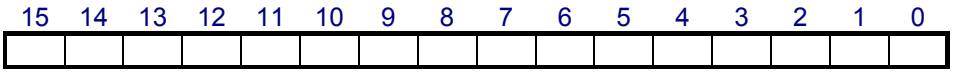
III - 1 - Structure de la donnée: OCTET

(situé à l'intérieur d'un registre l'opérande octet concerne uniquement l'octet de poids faible).

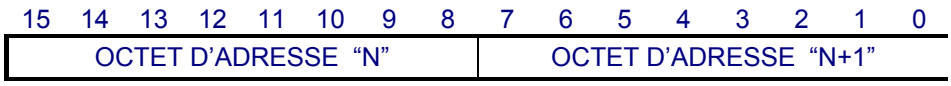


III - 2 - Structure de la donnée: MOT

Soit le **mot d'adresse N**



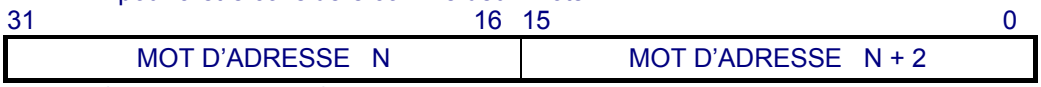
Nous pourrons également le considérer comme 2 octets:



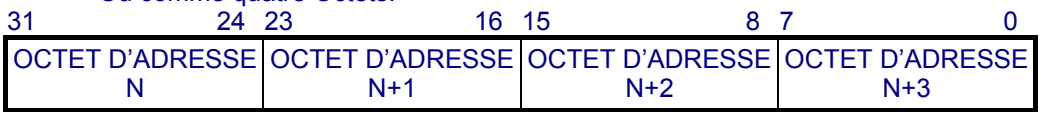
III - 3 - Structure de la donnée: MOT LONG

Soit le mot long d'adresse N

Il pourra être considéré comme deux mots:

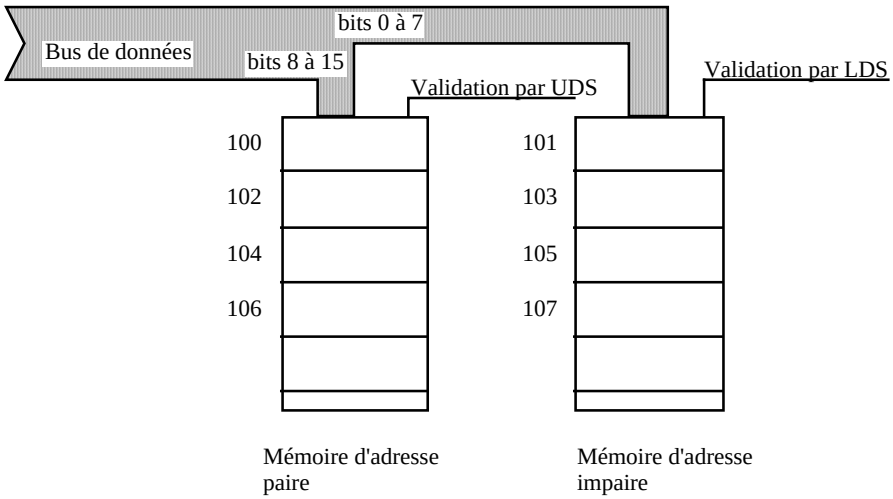


Ou comme quatre Octets:



III - 4 - Structure des données en mémoire

Soient les deux mémoires structurées en mots de 8 bits l'une connectée sur la partie haute et l'autre sur la partie basse du bus de données du microprocesseur. Chacune reçoit sur ses entrées d'adresse les mêmes lignes du bus d'adresse et leurs Chip Select sont pilotés par Lds pour l'une et Uds pour l'autre



Action d'une adresse contenue dans le programme ou élaborée par le programme sur les lignes du bus d'adresses et les broches de la mémoire:

Exemple: Transfert à l'adresse 0x100 (adresse paire)

Adresse (dans le programme)	0	0	0	1	0	0	0	0	0	0	0	0
id en hexadécimal	1				0				0			
Broches du bus d'adresse	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	Uds/Lds
sur le bus pour Opérande Octet	0	0	0	1	0	0	0	0	0	0	0	0/1
sur le bus pour Opérande Mot	0	0	0	1	0	0	0	0	0	0	0	0/0
broches d'adresse de la mémoire	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	Cs
	0			8				0				

Dans cet exemple, les adresses 0x100 et 0x101 qui ne diffèrent que par le bit A0 sélectionnent dans chaque mémoire la même case (adresse 0x080). Lorsque le transfert concerne le mot d'adresse 0x100, l'octet de poids fort se présente sur la partie haute du bus de données et l'octet de poids faible sur la partie basse , le 68000 active UDS et LDS et les deux octets sont transférés simultanément.

Lorsque le transfert concerne un octet, celui-ci se présente sur les deux parties du bus de données, les deux même cases dans chacune des mémoires sont sélectionnées mais si l'adresse est impaire le 68000 active Lds et le transfert se fait de ou dans la mémoire validée si l'adresse est paire c'est UDS qui est actif

Il est facile de voir que la différence entre Adresse paire et impaire qui se fait par le bit A0 à l'intérieur du micro, se fait à l'extérieur avec LDS et UDS or lorsque le micro écrit un mot, il doit sélectionner en même temps mémoire paire et impaire il ne peut plus alors faire la distinction entre paire et impaire.

Avec le microprocesseur 68000, l'écriture ou la lecture d'un mot (ou d'un mot long) à une adresse impaire n'est pas admise.
Une tentative de ce type provoque une exception "error address" (p30)

III - 5 - Les données dans une instruction (Opérandes)

Les instructions du 68000 se présentent de façon beaucoup plus souple que pour ses ancêtres ainsi dans la plupart des instructions nous trouverons la désignation de l'opérande source et de l'opérande destination. Le premier opérande désigné dans l'instruction constitue la source le second la destination. De même chaque instruction précisera la taille des opérandes.

Par exemple, l'instruction:

```
MOVE.W  D0,D1
```

signifie:

```
MOVE    = transfert
.W      = Taille de l'opérande le mot (word)
D0      = Opérande source
D1      = Opérande destination
```

TRANSFERT du mot de poids faible contenu dans D0 dans le mot de poids faible de D1

Second exemple:

```
ADD.L   D1,D7
```

ADDITION du mot long (L) contenu dans D1 avec celui de D7 résultat dans D7

Troisième exemple:

```
AND.B   D7,D1
```

ET logique entre l'octet (B) de poids faible de D7 et celui de D1 résultat dans D1

Quatrième exemple:

les modes d'adressage pourront être différents pour la recherche de l'opérande source et de l'opérande destination

```
OR.B    D0,0x1001
```

OU logique entre l'octet de poids faible de D0 et l'octet d'adresse hexadécimale 1001 résultat à l'adresse 1001.

IV - LES MODES D'ADRESSAGE

IV - 1 - Vue d'ensemble des modes d'adressage

Afin de rendre ce tableau plus compréhensible, nous avons donné un exemple d'instruction pour chacun des modes d'adressage. L'instruction présentée est toujours un MOVE.W c'est à dire le transfert d'un mot. La source est désignée à l'aide de l'adressage étudié, la destination est toujours le registre D1.
Les mots extension sont des mots qui accompagnent l'instruction permettant de déterminer les opérandes ou contenant les opérandes (adressage immédiat).
Le codage FC est l'état des broches FC1 et FC0 lorsque le microprocesseur lit ou écrit à l'adresse en question:

01 espace "données" (mémoire RAM, registres internes des circuits périphériques etc.)
10 espace "programme" (Mémoire ROM ou RAM d'accueil du programme)

MODE D'ADRESSAGE	FORME	EXEMPLE	MOTS extension	CODAGE (FC 1,0)
A - DIRECT PAR REGISTRE				
- Direct par registre de données	Dn	MOVE.W D2,D1	0	01
- Direct par registre d'adresses	An	MOVE.W A2,D1	0	01
B - INDIRECT PAR REGISTRE D'ADRESSES				
- Indirect par registre d'adresses	(An)	MOVE.W (A2),D1	0	01
- Indirect par registre d'adresses post-incrémenté	(An)+	MOVE.W (A2)+,D1	0	01
- Indirect par registre d'adresses pré-décrémenté	-(An)	MOVE.W -(A2),D1	0	01
- Indirect par registre d'adresses avec déplacement	$d_{(16)}$(An)	MOVE.W (0x1000,A2),D1 ou MOVE.W 0x1000(A2),D1	1	01
- Indirect par registre d'adresses avec dépl et index	$d_{(8)}$(An,Rx)	MOVE.W 0x10(A2,D0.W),D1	1	01
C - ABSOLU				
- Absolu court	$N_{(16)}$	MOVE.W 0x1000,D1	1	01
- Absolu long	$N_{(32)}$	MOVE.W 0x0011223344,D1	2	01
D - INDIRECT PAR COMPTEUR DE PROGRAMME (RELATIF)				
- Indirect par PC avec déplacement	$d_{(16)}$(PC)	MOVE.W 0x0020(PC),D1	1	10
- Indirect par PC avec déplacement et index	$d_{(8)}$(PC,Rx)	MOVE.W 0x20(PC,D0.W),D1	1	10
E - IMMEDIAT				
- Immédiat	#$N_{(8,16 ou 32 bits)}$	MOVE.W #0x1000,D1	1 ou 2	01
- Immédiat rapide	#$N_{(<à 3 ou 8 bits)}$	MOVEQ #1,D1	0	01

Abréviations utilisées:

$d_{(16)}$ = déplacement sur 16 bits
$N_{(16)}$ = nombre de 16 bits
Rx = Registre quelconque (A ou D)

IV - 2 - Détail des modes d'adressage

IV - 2 - 1 - Adressage direct par registre

IV - 2 - 1 - 1 - ADRESSAGE DIRECT PAR REGISTRE DE DONNÉES.

L'opérande est contenu dans le registre de données spécifié dans l'instruction.

Localisation de l'opérande	l'opérande est le contenu de Dn
Syntaxe Assembleur	Dn
Codification dans le mot Instruction	
Mode	
Registre	000 N° du registre (n)
Codification sur FC 1 0	01
Mémoire	mémoire de données
Nombre de Mot extension	0

Exemple: Instruction **MOVE.L D2,D1**

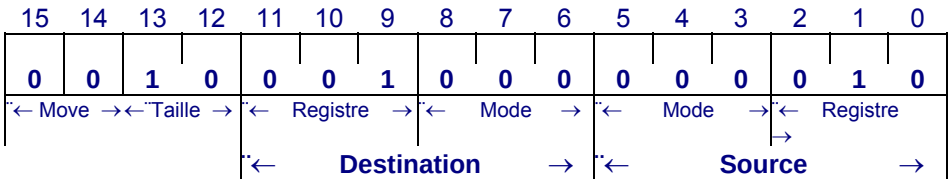
AVANT EXECUTION

D2	11	22	33	44
D1	00	00	00	00

APRES EXECUTION

D2	11	22	33	44
D1	11	22	33	44

Codage de l'Instruction



Mot Instruction:

0x2202

Ce mode d'adressage ne nécessite pas de mot extension

IV - 2 - 1 - 2 - ADRESSAGE DIRECT PAR REGISTRE D'ADRESSES.

L'opérande est contenu dans le registre d'adresses spécifié dans l'instruction.

Localisation de l'opérande	l'opérande est le contenu de An
Syntaxe Assembleur	An
Codification dans le mot Instruction	
Mode	001
Registre	N° du registre (n)
Codification sur FC 1 0	01
Mémoire	mémoire de données
Nombre de Mot extension	0

Exemple : **MOVE.L A2,D1**

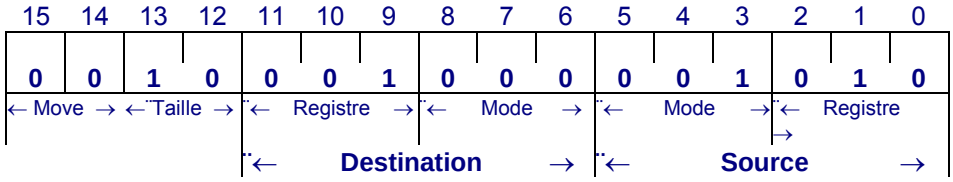
AVANT EXECUTION

A2	11	22	33	44
D1	00	00	00	00

APRES EXECUTION

A2	11	22	33	44
D1	11	22	33	44

Codage de l'Instruction



Mot Instruction:
0x220A

Ce mode d'adressage ne nécessite pas de mot extension

IV - 2 - 2 - Adressage Indirect par registre d'adresses

IV - 2 - 2 - 1 - ADRESSAGE INDIRECT PAR REGISTRE D'ADRESSES.

L'opérande est contenu dans l'emplacement mémoire dont l'adresse est spécifiée dans le registre d'adresses. Le Numéro du registre d'adresses est précisé dans l'instruction.

Localisation de l'opérande	l'opérande est le contenu de l'adresse logée dans An
Syntaxe Assembleur	(An)
Codification dans le mot Instruction	
Mode	010
Registre	N° du registre (n)
Codification sur FC 1 0	01
Mémoire	mémoire de données
Nombre de Mot extension	0

Exemple: **MOVE.L (A2),D1**

AVANT EXECUTION

Mémoire: 0x10000 = 1122
0x10002 = 3344
0x10004 = 5566

Registres:

A2	00	01	00	00
D1	00	00	00	00

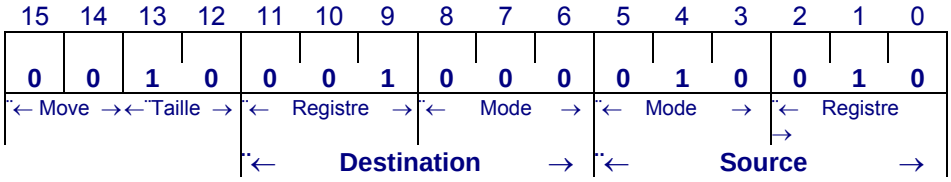
APRES EXECUTION

Mémoire: 0x10000 = 1122
0x10002 = 3344
0x10004 = 5566

Registres

A2	00	01	00	00
D1	11	22	33	44

Codage de l'Instruction



Mot Instruction:
0x2212

Cette instruction ne nécessite pas de mot extension

IV - 2 - 2 - 2 - ADRESSAGE INDIRECT PAR REGISTRE D'ADRESSES AVEC POST-INCRÉMENTATION

L'opérande est contenu dans l'emplacement mémoire dont l'adresse est spécifiée dans le registre d'adresses précisé dans l'instruction. Après exécution de l'instruction l'adresse contenue dans le registre est incrémentée de 1 pour le transfert d'un octet (.B), de 2 pour le transfert d'un mot (.W) ou de 4 pour un mot long (.L).

Localisation de l'opérande	L'opérande est le contenu de l'adresse pointée par An Après exécution le contenu de An est incrémenté de 1,2,4 suivant la taille de l'opérande
Syntaxe Assembleur	(An)+
Codification dans le mot Instruction	
Mode	011
Registre	N° du registre (n)
Codification sur FC 1 0	01
Mémoire	mémoire de données
Nombre de Mot extension	0

Exemple: **MOVE.L (A2)+,D1**

AVANT EXECUTION

Mémoire: 0x10000 = 1122
0x10002 = 3344
0x10004 = 5566

Registres:

A2	00	01	00	00
D1	00	00	00	00

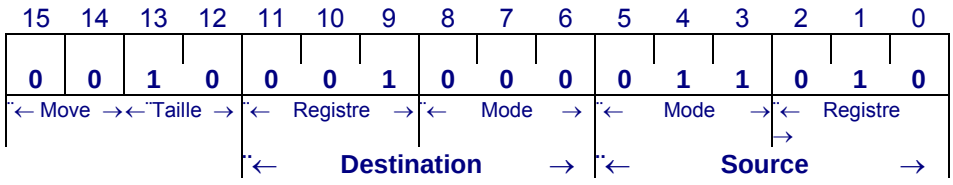
APRES EXECUTION

Mémoire: 0x10000 = 1122
0x10002 = 3344
0x10004 = 5566

Registres

A2	00	01	00	04
D1	11	22	33	44

Codage de l'Instruction



Mot Instruction:
0x221A
Cette instruction ne nécessite pas de mot extension

IV - 2 - 2 - 3 - ADRESSAGE INDIRECT PAR REGISTRE D'ADRESSES AVEC PRÉ-DÉCRÉMENTATION

L'adresse de l'opérande est obtenue en retranchant au contenu du registre d'adresses spécifié dans l'instruction le nombre d'octets, taille de l'opérande, 1 pour le transfert d'un octet (.B), 2 pour le transfert d'un mot (.W) ou 4 pour un mot long (.L).

Localisation de l'opérande	L'opérande est le contenu de l'adresse logée dans An préalablement décrémentée de 1,2,4 suivant la taille de l'opérande
Syntaxe Assembleur	-(An)
Codification dans le mot Instruction	
Mode	100
Registre	N° du registre (n)
Codification sur FC 1 0	01
Mémoire	mémoire de données
Nombre de Mot extension	0

Exemple: **MOVE.L -(A2),D1**

AVANT EXECUTION

Mémoire: 0x10000 = 1122
0x10002 = 3344
0x10004 = 5566

Registres:

A2	00	01	00	04
D1	00	00	00	00

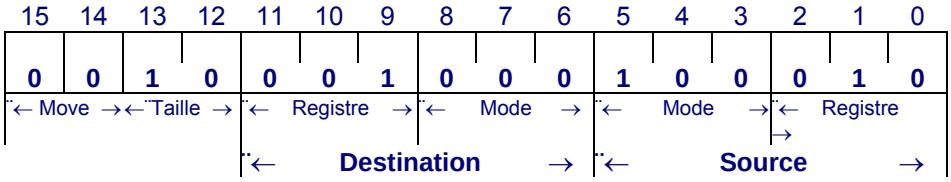
APRES EXECUTION

Mémoire: 0x10000 = 1122
0x10002 = 3344
0x10004 = 5566

Registres

A2	00	01	00	00
D1	11	22	33	44

Codage de l'Instruction



Mot Instruction:
0x2222

Cette instruction ne nécessite pas de mot extension

IV - 2 - 2 - 4 - ADRESSAGE INDIRECT PAR REGISTRE D'ADRESSES AVEC DEPLACEMENT

L'adresse de l'opérande est obtenue en additionnant le déplacement mentionné dans l'instruction au contenu du registre d'adresses spécifié. Le déplacement est un nombre de 16 bits dont le signe sera étendu à 32 bits lors du calcul de l'adresse

Localisation de l'opérande	L'opérande est le contenu de l'adresse obtenue en additionnant le contenu de An au déplacement spécifié dans l'instruction
Syntaxe Assembleur	d(An)
Codification dans le mot Instruction Mode	101
Registre	N° du registre (n)
Codification sur FC 1 0 Mémoire	01 mémoire de données
Nombre de Mot extension	1

Le déplacement est donné sur 16 bits en mode complément à 2 ce qui permet d'accéder à une zone de ± 32KO. Exemples de déplacement:

Déplacement décimal	Mot extension	Valeur ajoutée à (An)
0	0x0000	0x0000 0000
+1	0x0001	0x0000 0001
+32 767	0x7FFF	0x0000 7FFF
-32 767	0x8001	0xFFFF 8001
-1	0xFFFF	0xFFFF FFFF

Exemple: **MOVE.L (0x1000,A2),D1**

Calcul de l'adresse (0x1000,A2)

(A2) + 0x1000 = 0x00010000+0x00001000 = 0x00011000

AVANT EXECUTION

Mémoire: 0x11000 = 1122
0x11002 = 3344
0x11004 = 5566

APRES EXECUTION

Mémoire: 0x11000 = 1122
0x11002 = 3344
0x11004 = 5566

Registres:

A2	00	01	00	00
D1	00	00	00	00

Registres

A2	00	01	00	00
D1	11	22	33	44

Codage de l'Instruction

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**0**	**1**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**1**	**0**	**1**	**0**	**1**	**0**
← Move →		←Taille →		← Registre →			← Mode →			← Mode →			← Registre →		
				← **Destination** →						← **Source** →					

Mot Extension

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

Mot Instruction: **0x222A**

Mot Extension **0x1000**

IV - 2 - 2 - 5 - ADRESSAGE INDIRECT PAR REGISTRE D'ADRESSES AVEC DÉPLACEMENT SUR 8 BITS ET INDEX

L'adresse de l'opérande est obtenue en faisant la somme du contenu du registre d'adresses spécifié, du déplacement sur 8bits dont le signe est étendu à 32 bits et du contenu de l'index sur la taille spécifiée.
Le champ utile à l'intérieur du registre d'index est précisé dans l'instruction (mot, mot long), le signe est étendu à 32 bits de le même façon que le déplacement.

Localisation de l'opérande	L'opérande est le contenu de l'adresse obtenue en additionnant le contenu de An au déplacement spécifié dans l'instruction et à l'index
Syntaxe Assembleur	$d_{(8)}$(An,Rx)
Codification dans le mot Instruction	
Mode	110
Registre	N° du registre (n)
Codification sur FC 1 0	01
Mémoire	mémoire de données
Nombre de Mot extension	1

Exemple: **MOVE.L 0x10(A2,D0.W),D1**

Calcul de l'Adresse de l'opérande source :
Index D0.W = 0x0002 ⇒ D0.W Signe étendu = 0x00000002
Déplacement = 0x10 ⇒ Déplacement Signe étendu 0x00000010
Adresse = D0.W + Dép + A2 = 00000002 + 00000010 + 00000100 = 00000112

AVANT EXECUTION

Mémoire: 0x000112 = 1122
0x000114 = 3344
0x000116 = 5566

Registres:

A2	00	00	01	00
D0	11	11	00	02
D1	00	00	00	00

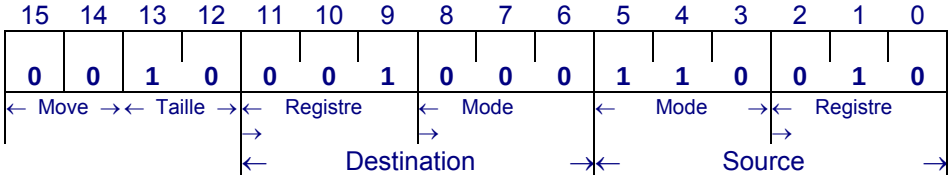
APRES EXECUTION

Mémoire: 0x000112 = 1122
0x000114 = 3344
0x000116 = 5566

Registres:

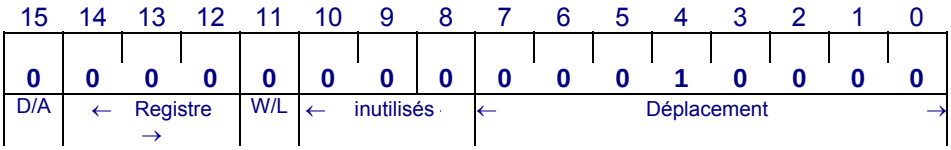
A2	00	00	01	00
D0	11	11	00	02
D1	11	22	33	44

Codage de l'Instruction



Voir le contenu du mot extension page suivante

Mot Extension pour adressage indexé



Mot instruction **0x2232**

Mot extension **0x0010**

Champ de définition de l'index:

D/A Registre de Données ou d'Adresses
- 0 Registre de données
- 1 Registre d'adresses

REGISTRE
N° du registre sur 3 bits

W/L SIZE
- 0 (.W) Le registre est utilisé sur ses 16 bits de poids faible
- 1 (.L) Le registre est utilisé sur ses 32bits

DEPLACEMENT
Valeur du déplacement sur 8 bits

IV - 2 - 3 - Adressages absolus

Les deux adressages regroupés dans ce chapitre sont dés plus simples puisqu'il suffit de mentionner en clair l'adresse de l'opérande, soit sur un mot, adressage absolu court, soit sur un mot long, adressage absolu long.

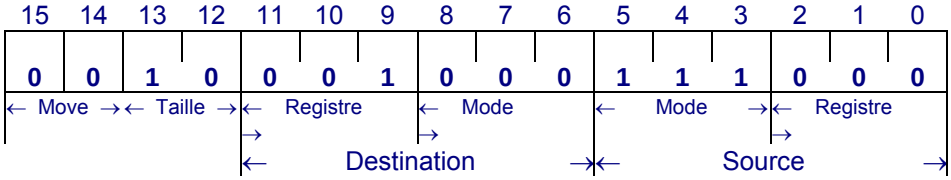
IV - 2 - 3 - 1 - ADRESSAGE ABSOLU COURT

L'adresse de l'opérande est fournie sur un mot, l'assembleur étendra le bit de signe à 32 bits.

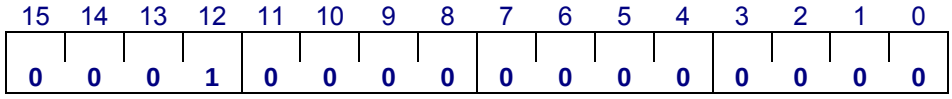
Localisation de l'opérande	L'opérande est le contenu de l'adresse spécifiée dans l'instruction
Syntaxe Assembleur	$N_{(16)}$
Codification dans le mot Instruction	
Mode	111
Registre	000
Codification sur FC 1 0	01
Mémoire	mémoire de données
Nombre de Mot extension	1

Exemple: **MOVE.L 0x1000,D1**

Le mot long contenu dans l'emplacement mémoire d'adresse 0x001000 sera tranféré dans le registre D1
Codage de l'Instruction



Mot Extension pour Adresse de l'opérande source (0x1000)



Mot instruction **0x2238** Mot extension **0x1000**

IV - 2 - 3 - 2 - ADRESSAGE ABSOLU LONG

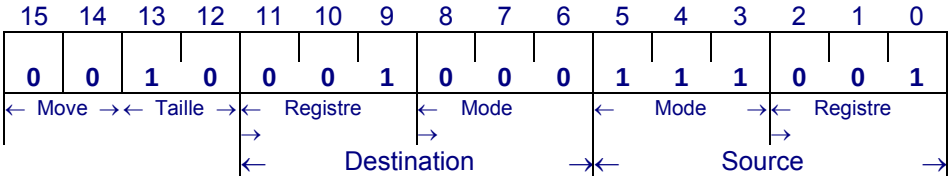
L'adresse de l'opérande est fournie sur un mot long.

Localisation de l'opérande	L'opérande est le contenu de l'adresse spécifiée dans l'instruction
Syntaxe Assembleur	$N_{(32)}$
Codification dans le mot Instruction	
Mode	111
Registre	001
Codification sur FC 1 0	01
Mémoire	mémoire de données
Nombre de Mot extension	2

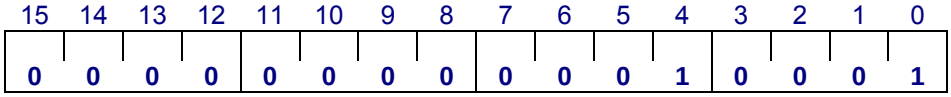
Exemple: **MOVE.L 0x00112233,D1**

Le mot long contenu dans l'emplacement mémoire d'adresse 0x112233 sera transféré dans le registre D1.

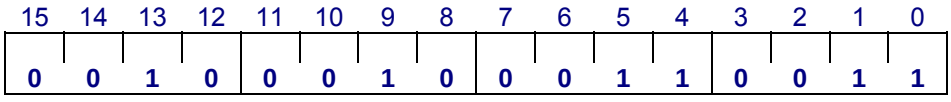
Codage de l'Instruction



Premier Mot Extension pour la partie haute de l'adresse de l'opérande source (0x0011)



Deuxième Mot Extension pour la partie basse de l'adresse de l'opérande source (0x2233)



Mot instruction	1er Mot extension	2ème Mot extension
0x2239	**0x0011**	**0x2233**

IV - 2 - 4 - Adressages indirects par compteur de Programme

Ce chapitre regroupe deux modes d'adressage similaires à certains déjà étudiés dans les modes indirect par rapport à An, mais le registre de base qui dans ces adressages étaient obligatoirement un registre d'adresse est ici remplacé par le compteur de programme. On pourrait dénommer ces modes d'adressages relatifs
Ces adressages sont classifiés "programme".

IV- 2 - 4 - 1 - ADRESSAGE INDIRECT PAR PC AVEC DÉPLACEMENT SUR 16 BITS

C'est un mode très simple puisqu'il consiste à additionner au contenu de PC un déplacement exprimé sur 16 bits. Le déplacement est contenu dans un mot extension, son bit de signe est étendu à 32 bits lors du calcul de l'adresse de l'opérande, la valeur de PC à prendre en compte est l'adresse du mot extension.
Cet adressage est classifié programme.

Localisation de l'opérande	L'opérande est le contenu de l'adresse obtenue en additionnant le déplacement à l'adresse du mot extension
Syntaxe Assembleur	$d_{(16)}$(PC) ou Étiquette(PC)
Codification dans le mot Instruction	
Mode	111
Registre	010
Codification sur FC 1 0	10
Mémoire	mémoire de programme
Nombre de Mot extension	1

Exemple: **MOVE.L 0x0020(PC),D1**
Calcul de l'Adresse de l'opérande source: Supposons l'instruction logée à l'adresse 0x1000 le compteur de programme sera pointé en 1002 pendant l'exécution de l'instruction l'adresse de l'opérande sera = PC + d
Déplacement 0x0020 fi Déplacement signe étendu 0x00000020
PC + d = 00001002 + 00000020 = 00001022

AVANT EXECUTION			APRES EXECUTION		
Mémoire: 0x001000	=	223A**	Mémoire: 0x001000	=	223A**
0x001002	=	0020*	0x001002	=	0020*
0x001004	=	1234	0x001004	=	1234
.....			.....		
0x001020	=	0000	0x001020	=	0000
0x001022	=	1122	0x001022	=	1122
0x001024	=	3344	0x001024	=	3344

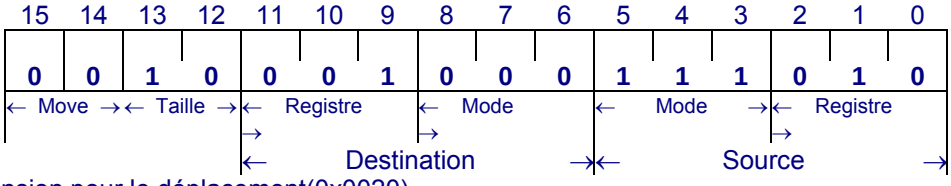
Registres:

D1	00	00	00	00

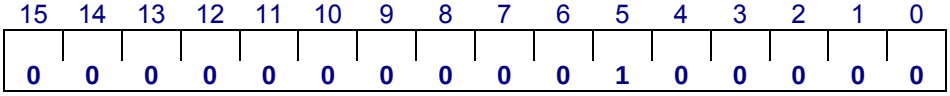
Registres:

D1	11	22	33	44

** = Mot instruction * = Mot extension (déplacement)
Codage de l'Instruction



Mot Extension pour le déplacement(0x0020)



mot instruction **0x223A** mot extension **0x0020**

IV- 2 - 4 - 2 - ADRESSAGE INDIRECT PAR PC AVEC DÉPLACEMENT SUR 8 BITS, INDEXE

L'adresse de l'opérande est obtenue en faisant la somme du contenu de PC avec le déplacement sur 8 bits, signe étendu à 32 et l'index en tenant compte de sa dimension .

Localisation de l'opérande	L'opérande est le contenu de l'adresse obtenue en additionnant le déplacement à l'adresse du mot extension et à l'index
Syntaxe Assembleur	$d_{(8)}$(PC,Rn) ou Étiquette(PC,Rn)
Codification dans le mot Instruction	
Mode	111
Registre	011
Codification sur FC 1 0	10
Mémoire	mémoire de programme
Nombre de Mot extension	1

Exemple: **MOVE.L 0x20(PC,D0.W),D1**

Calcul de l'Adresse de l'opérande source : Supposons l'instruction logée à l'adresse 0x1000 le compteur de programme sera pointé en 1002 pendant l'exécution de l'instruction l'adresse de l'opérande sera:
Déplacement 0x20 fi Déplacement signe étendu 0x00000020
Index D0.W=0x0002 fi D0.W signe étendu 0x00000002
d +PC + X = 00000020 + 00001002 + 00000002 = 00001024

AVANT EXECUTION			APRES EXECUTION		
Mémoire: 0x001000	=	223B**	Mémoire: 0x001000	=	223B**
0x001002	=	0020*	0x001002	=	0020*
0x001004	=	1234	0x001004	=	1234
.....			.....		
0x001022	=	0000	0x001022	=	0000
0x001024	=	1122	0x001024	=	1122
0x001026	=	3344	0x001026	=	3344

Registres:

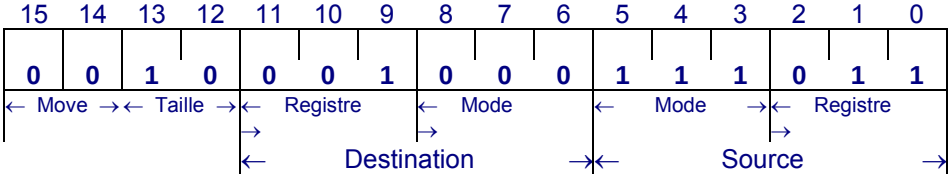
D0	11	11	00	02
D1	00	00	00	00

Registres:

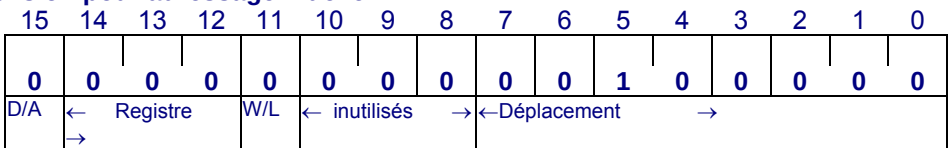
D0	11	11	00	02
D1	11	22	33	44

** = Mot instruction * = Mot extension

Codage de l'Instruction



Mot Extension pour adressage indexé



(Voir description des champs page 18)

Mot instruction **0x223B** Mot extension **0x0020**

IV - 2 - 5 - ADRESSAGE IMMÉDIAT

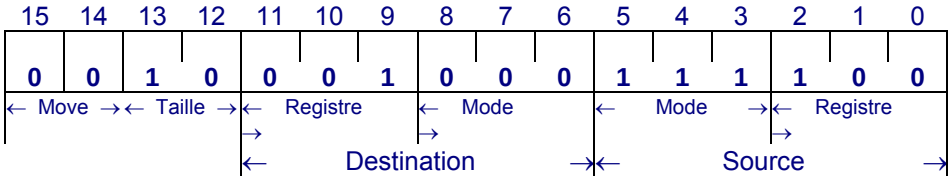
Il ne s'agit pas, à proprement parler, d'un mode d'adressage puisque ce n'est pas l'adresse de l'opérande qui est fournie dans l'instruction mais l'opérande lui même.
Quelques instructions ne nécessitent pas de mot extension (telles que ADDQ ou SUBQ) lorsque la taille de l'opérande est réduite (de 1 à 8). Pour les autres si l'opérande porte sur un octet il sera constitué de l'octet de poids faible du mot extension, pour un mot ce sera tout le mot extension, pour un mot long deux mots extension.

Localisation de l'opérande	L'opérande est fourni dans l'instruction
Syntaxe Assembleur	#N ou #Étiquette
Codification dans le mot Instruction Mode Registre	 111 100
Codification sur FC 1 0 Mémoire	01 mémoire de données
Nombre de Mot extension	0,1 ou 2

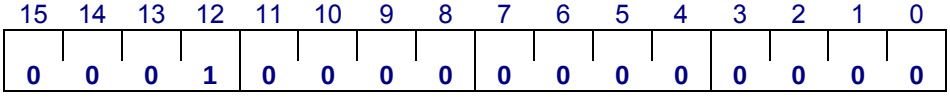
Exemple **MOVE.L #0x1000,D1**

La valeur 0x1000 sera transférée dans D1 sous la forme d'un mot long

Codage de l'Instruction



Mot Extension pour l'opérande source lui même (0x1000)



Mot Instruction **0x223C** Mot extension **0x1000**

V - LES MODES DE FONCTIONNEMENT DU 68000

Le 68000 peut fonctionner dans 2 niveaux de privilège, selon l'état du bit S du registre d'état:
Le mode **Utilisateur** ou niveau de moindre privilège (bit S de SR à 0)
Le mode **Superviseur** ou niveau de privilège maximum (bit S à 1).

Le niveau de privilège dans lequel travaille le processeur à un instant donné est répercuté sur la broche FC2. Le décodage de ce bit peut permettre, notamment, de scinder les zones mémoire utilisateur et superviseur. Il est possible aussi, dans un système multi-utilisateurs d'interdire l'accès des utilisateurs à la mémoire réservée à l'**Operating System** (OS). L'Operating System (logiciel qui permet la gestion de la

machine informatique) travaille en mode superviseur, il doit pouvoir accéder à toutes les ressources et coordonner les activités des utilisateurs, par contre ceux-ci ne doivent pas pouvoir modifier la configuration du système. Une PMMU (Unité de Gestion Mémoire) pourra utiliser ces niveaux de privilège pour la gestion mémoire.

V - 1 - Le mode Superviseur

Le mode superviseur est obtenu en positionnant le bit S du registre d'état à 1, la broche FC2 se positionne à 1. Le passage du mode utilisateur au mode superviseur ne peut se faire qu'à la faveur d'une exception.

Toutes les interruptions sont traitées en mode superviseur et utilisent la pile pointée par SSP quel que soit l'état du bit S avant l'entrée en exception

En mode superviseur il est possible d'utiliser l'ensemble du jeu d'instructions

V - 2 - Mode Utilisateur

Le mode utilisateur est obtenu en positionnant le bit S du registre d'état à 0 , la broche FC2 se positionne à 0. C'est l'état de moindre privilège.

L'ensemble du jeu d'instructions ne peut être utilisé dans ce mode de fonctionnement. Les restrictions d'utilisation portent essentiellement sur les instructions permettant d'influer sur la configuration d'un système. Les instructions non autorisées en mode utilisateur sont repérées par un astérix (*) dans le tableau des pages 30 et 31, ce sont essentiellement les instructions qui permettraient à un utilisateur de modifier le registre d'état (octet de poids fort) ou le pointeur de pile superviseur (SSP).

Dans l'état utilisateur, toute référence au pointeur de pile soit implicite soit explicite, registre d'adresse A7, concerne USP.

V - 3 - Changement de niveau de privilège

V - 3 - 1 - Passage de l'état Utilisateur à l'état Superviseur.

La seule façon de passer de l'état utilisateur à l'état superviseur est de rencontrer une exception.
Le traitement de l'exception sauve l'état courant des bits du registre d'état dans la pile superviseur active et met le bit S à 1 forçant ainsi le microprocesseur à l'état superviseur. Il est possible, si le programme de traitement de l'exception l'a prévu, de revenir dans le programme antérieur à l'aide de l'instruction RTR qui chargera PC avec l'adresse de l'instruction qui suit l'exception et en ne restaurant que la zone CCR du registre d'état, l'utilisateur reste alors en l'état superviseur. Il est possible également de modifier le contenu de la pile de telle sorte que l'instruction RTE réinstalle un registre dans lequel le bit S aura été mis à 1.

V - 3 - 2 - Passage de l'état Superviseur à l'état Utilisateur

Le passage de l'état superviseur à l'état utilisateur se fait en positionnant le bit S du registre d'état à 0 ce qui ne pose pas de problème puisqu'en l'état superviseur toutes les instructions peuvent être utilisées. Ceci pourra se faire à l'aide des instructions telles que MOVE, ANDI, EORI to SR.

V - 4 - Les zones mémoires

Les broches FC 2,1,0 donnent une information sur l'état dans lequel travail le microprocesseur.

FC2	FC1	FC0	Zone Mémoire
0	0	0	Indéfinie
0	0	1	Données Utilisateur
0	1	0	Programme Utilisateur
0	1	1	Indéfinie
1	0	0	Indéfinie
1	0	1	Données Superviseur
1	1	0	Programme Superviseur
1	1	1	Reconnaissance d'interruption, pas de zone mémoire définie

L'utilisation de ces informations peut être incluse dans le décodage des adresses et ainsi permettre la cohabitation de différentes zones mémoire réservées aux différents modes de fonctionnement.

V - 5 - Les Exceptions

V - 5 - 1 - Énumération des exceptions

Les exceptions résultant d'une action sur les broches du 68000 dites **exceptions externes**

- Interruptions
- Erreur Bus
- Reset

Exceptions résultant de l'exécution d'une instruction ou **exceptions internes**

- Erreur adresse
- Trace
- Trappe, Trappe V
- Chk
- Division par 0
- Instruction Illégale
- Viol de privilège

V - 5 - 2 - Classement des exceptions

GROUPES DE PRIORITES	PRIORITES RELATIVES	CARACTERISTIQUES
0	0-0 Reset	Fait avorter le processus en cours rien n'est sauvegardé
	0-1 Erreur bus 0-2 Erreur adresse	Suspend le traitement de l'instruction en cours et sauve le contexte interne
1	1-0 Trace 1-1 Interruption	Le traitement de l'exception commence après l'exécution de l'instruction
	1-2 Instruction illégale ou inexistante, viol de privilège	Le processus de traitement de l'exception commence avant l'exécution de l'instruction
2	2-0 Chk, Div par 0 Trappe	Le traitement de l'exception fait partie de l'exécution de l'instruction

V- 5 - 3 - Séquence type de traitement d'une exception

La séquence type de traitement est constituée de 4 étapes:

a) Sauvegarde interne du registre d'état tel qu'il était avant la reconnaissance de l'exception puis les bits S et T du registre d'état en activité sont mis respectivement à 1 et 0 (pour les interruptions et reset , mise à jour du masque d'interruption).

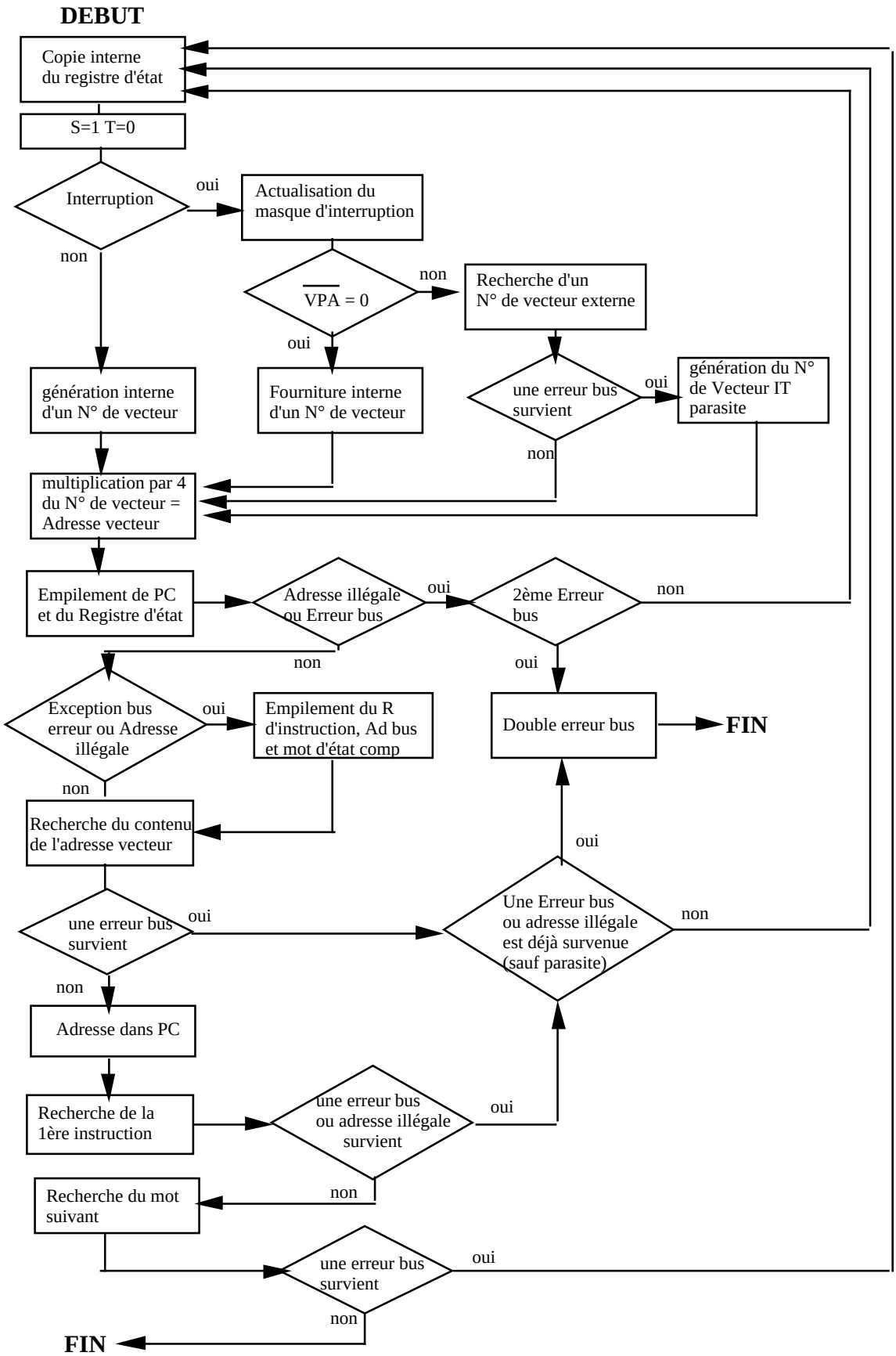
b) Détermination du numéro de vecteur. Pour les interruptions vectorisées, le N° est lu pendant que les lignes FC2,1,0 sont à 1 ce qui constitue un cycle de reconnaissance d'interruption. Pour toutes les autres exceptions une logique interne fournit le numéro vecteur

c) Pour toutes les exceptions sauf reset, la troisième étape consiste à sauvegarder l'état interne du processeur . Le contexte d'exception est logé dans la pile. Le contexte est très réduit puisqu'il ne se compose que de PC et du registre d'état sauvegardé initialement. Un contexte spécial pour les erreurs bus et erreurs adresse composé de 7 mots est sauvegardé afin de permettre une analyse plus fine de l'exception (voir page 30)

d) La dernière étape est identique pour toutes les exceptions, l'adresse vecteur est déterminée en multipliant par 4 le N° vecteur. Le vecteur d'exception est le mot long logé dans l'espace superviseur à l'adresse calculée. Il est ensuite chargé dans PC. Le microprocesseur amorce ensuite l'exécution du programme.

Il est a noter que le vecteur reset qui est constitué de deux mots longs, le premier pour charger SSP et le second pour PC. Ces vecteurs sont logés **dans l'espace programme superviseur alors** que tous les autres sont dans **l'espace données superviseur.**

V- 5 - 4 - Organigramme de traitement des exceptions



V - 5 - 5 - Numérotation des vecteurs d'exception

Numéro Vecteur	Adresse Vecteur	Affectation	Zone mémoire (broches FC)
00	0x000	Reset Initialisation de SSP	Prog Super
01	0x004	Reset Initialisation de PC	Prog Super
02	0x008	Erreur Bus	Données Super
03	0x00C	Erreur Adresse	Données Super
04	0x010	Instruction Illégale	Données Super
05	0x014	Division par zéro	Données Super
06	0x018	CHK	Données Super
07	0x01C	TrapV	Données Super
08	0x020	Viol de privilège	Données Super
09	0x024	Trace	Données Super
10	0x028	Émulateur Ligne A	Données Super
11	0x02C	Émulateur Ligne F	Données Super
12	0x030	Non affecté (réservé)	
13	0x034	Non affecté (réservé)	
14	0x038	Non affecté (réservé)	
15	0x03C	Interruption non initialisée	Données Super
16	0x040	----------	
à	à	Non affectés (réservés)	
23	0x05C	----------	
24	0x060	Interruption parasite	Données Super
25	0x064	Auto-vecteur Interruption Niveau 1	Données Super
26	0x068	Auto-vecteur Interruption Niveau 2	Données Super
27	0x06C	Auto-vecteur Interruption Niveau 3	Données Super
28	0x070	Auto-vecteur Interruption Niveau 4	Données Super
29	0x074	Auto-vecteur Interruption Niveau 5	Données Super
30	0x078	Auto-vecteur Interruption Niveau 6	Données Super
31	0x07C	Auto-vecteur Interruption Niveau 7	Données Super
32	0x080	----------	
à	à	Instructions Trap de #0 à #15	Données Super
47	0x0BC	----------	
48	0x0C0	----------	
à		Non affectés (réservés)	Données Super
63	0x0FF	----------	
64	0x100	----------	
à	à	Affectation utilisateur	Données Super
255	0x3FC	----------	

Seuls les vecteurs N° 0 et 1 sont dans l'espace **Programme Superviseur** tous les autres sont dans l'espace **Données Superviseur**

V - 5 - 6 - Traitement détaillé des exceptions

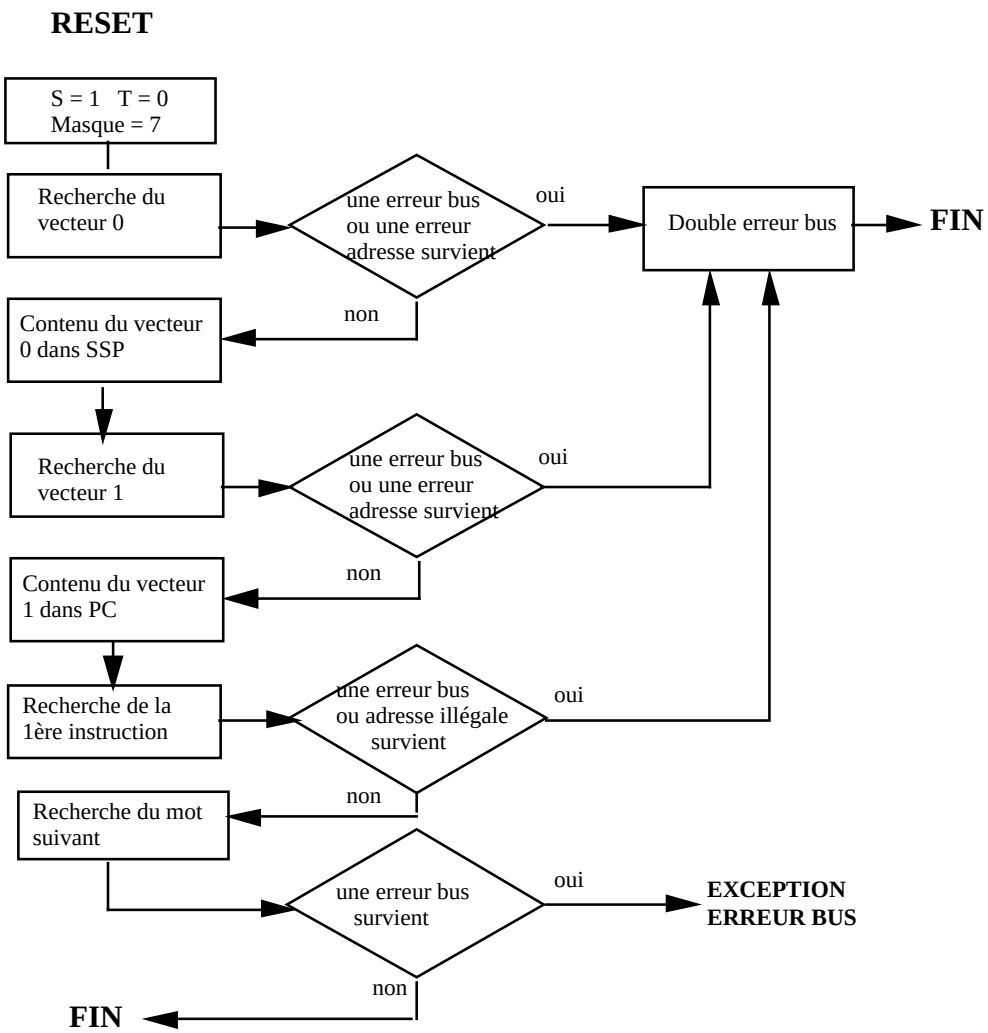
V - 5 - 6 - 1 - Traitement détaillé des exceptions: reset

RESET

C'est l'exception de niveau prioritaire maximum. Elle est utilisée pour initialiser le système à la mise sous tension ou en cas d'erreur majeure. Elle est causée par l'arrivée d'états actifs simultanés sur les entrées Reset et Halt. (Ne pas confondre avec l'instruction Reset destinée à réinitiliser les périphériques). Tout traitement en cours est avorté, rien n'est sauvegardé.

- Le registre d'état est initialisé:T est mis à 0, S est mis à 1, Le masque d'interruption au niveau 7
- Le vecteur 0 est fourni en interne et, exceptionnellement le microprocesseur va chercher deux mots longs dans la zone Programme Superviseur
  - Le premier à l'adresse 0 pour en charger le pointeur de pile superviseur (SSP)
  - Le second à l'adresse 4 pour initialiser le compteur de programme (PC)
- Rien ne peut être dit ni assuré pour les autres registres.

Le temps nécessaire pour amorcer l'exécution du programme est de 40 périodes d'horloge (le 68000 effectue 6 opérations de lecture en mémoire) à compter du moment où reset et halt sont devenus inactifs.



V - 5 - 6 - 2 - Traitement détaillé des exceptions:Erreur Bus

ERREUR BUS

Cette exception est causée par un état actif appliqué sur la broche BERR par une logique externe faisant avorter un cycle bus externe.
Si le processeur est dans l'espace données, il amorce immédiatement le traitement de l'exception.
Si le cycle bus avorté est une pré-recherche d'instruction, l'exception ne sera traitée que lorsque le processeur aurait dù exécuter l'instruction qui est cause de l'erreur bus.
Le Registre d'état est copié, les bits du registre actif sont positionnés de telle sorte que T = 0 et S = 1. Un numéro de vecteur est généré en interne (0x02). PC, la copie de SR, le contenu du registre d'instruction (celui-ci ne fait pas partie du modèle de programmation), la dernière adresse accédée (qui peut être celle de l'instruction ou d'un opérande) ainsi que l'état de certaines broches du microprocesseur (voir ci dessous le contexte empilé) sont sauvés dans la pile superviseur. La valeur de PC sauvegardée est l'adresse de l'instruction en cours d'exécution au moment ou est survenue l'erreur bus, ce n'est pas forcément cette instruction qui a causé l'erreur. En effet le 68000 va systématiquement chercher deux mots en mémoire de programme consécutivement, de nombreuses instructions ne nécessitant qu'un mot, il est possible que l'exécution d'une instruction s'amorce après la recherche de l'instruction suivante. Il peut donc y avoir ambiguïté sur les adresses.
Le temps nécessaire pour accéder au programme de traitement est de 50 périodes d'horloge, le 68000 effectue 4 opérations de lecture en mémoire et 7 d'écriture.
Deux cas particuliers peuvent se présenter:

1er Cas:

États actifs simultanés sur BERR et HALT
Le 68000 effectue dans ce cas un "RERUN" c'est à dire que, dès que l'action sur ces broches a cessé (d'abord sur BERR et ensuite sur HALT) le MPU ré-exécute la dernière instruction.

2ème Cas:

Le 68000 a reconnu une interruption, il est en attente de réception d'un N° de vecteur signalé par DTACK ou d'un signal VPA précisant que l'interruption est auto-vectorisée. Recevant un signal BERR, une logique interne lui fournit le N° de vecteur "interruption parasite" .

Schéma du contexte mis en pile:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
												R/W	I/N	FC 2,1,0		¨ SSP X - 14
DERNIERE ADRESSE ACCEDEE PARTIE HAUTE																¨ SSP X - 12
DERNIERE ADRESSE ACCEDEE PARTIE BASSE																¨ SSP X - 10
REGISTRE D'INSTRUCTION																¨ SSP X - 8
REGISTRE D'ETAT																¨ SSP X - 6
COMPTEUR DE PROGRAMME PARTIE HAUTE																¨ SSP X - 4
COMPTEUR DE PROGRAMME PARTIE BASSE																¨ SSP X - 2
																¨ SSP = X

Le pointeur de pile reste sur le dernier mot empilé à l'adresse X - 14 (10)

Pointeur de pile avant sauvegarde

Les informations rangées en SP-14 sont l'état des broches code fonction FC, de la ligne R/W et un bit I/N qui indique lorsqu'il est à 1 que le processeur traitait une exception du groupe 2 au moment de l'erreur.

V - 5 - 6 - 3 - Traitement détaillé des exceptions: Erreur Adresse

ERREUR ADRESSE

Cette exception survient lorsque le microprocesseur tente de rechercher ou de loger un mot ou un mot long à une adresse impaire. Ceci peut être assimilé à une Erreur Bus interne. Le cycle bus n'est pas exécuté, la procédure est identique à celle de l'erreur bus qui est développée dans le paragraphe précédent, cependant le numéro de vecteur est différent 0x0C.
Le temps nécessaire pour accéder au programme de traitement est de 50 périodes d'horloge, le 68000 effectue 4 opérations de lecture en mémoire et 7 d'écriture.

Si une erreur adresse survient pendant le traitement d'une erreur bus, d'une erreur adresse,ou d'un reset, le microprocesseur est mis en Halt, il ne pourra sortir de ce mode que par un reset matériel

V - 5 - 6 - 4 - Traitement détaillé des exceptions: Trace

TRACE

Le mode trace est destiné à aider à la mise au point des programmes en permettant de les exécuter instruction par instruction.
Ce mode utilise le bit T du registre d'état .

Si , au début de l'exécution d'une instruction le bit T est à 1, une exception trace sera générée après l'exécution de l'instruction.

De façon générale, une exception trace peut être vue comme l'extension de l'instruction, celle-ci ne sera pas considérée comme achevée tant que le processus lié à l'exception n'a pas été complètement exécuté.

Quelques cas particuliers:
- L'instruction subit une erreur bus ou une erreur adresse, l'exception Trace est différée jusqu'à ce que l'exécution de l'instruction suspendue soit achevée (par le RTE qui lui est associé).
- L'instruction est exécutée et une interruption est pendante, le trace est exécuté avant l'interruption.
- Pendant l'exécution de l'instruction, une exception est forcée par l'instruction, l'exception forcée est exécutée avant le trace.
- Si le processeur est en mode trace lors d'une tentative d'exécution d'instruction illégale ou inexistante, l'instruction ne causera pas de trace puisqu'elle n'est pas exécutée. Ceci est particulièrement important pour une routine d'émulation d'instruction qui réalise la fonction de l'instruction, ajuste la valeur empilée de PC à l'adresse de l'instruction suivante et retourne ensuite au programme. Avant que le retour ne soit exécuté, SR dans la pile pourrait être contrôlé pour déterminer si le mode trace est actif et, si tel est le cas, l'exception trace pourrait être émulée également.

Traitement de l'exception Trace:

L'exception Trace est initiée à la fin du traitement normal de l'instruction et avant le début de la prochaine instruction. Une copie interne du Registre d'état est faite, le passage à l'état superviseur est effectué (S=1), le bit T est mis à 0, un N° de vecteur est généré (N°9). PC et la copie de SR sont empilés. La valeur sauvée de PC est l'adresse de l'instruction suivante. L'exécution des instructions reprend dans la routine de traitement de Trace.
Le temps nécessaire pour accéder au programme de traitement est de 34 périodes d'horloge, le 68000 effectue 4 opérations de lecture en mémoire et 3 d'écriture.

Si une instruction STOP commence sont exécution avec T1=1, une Trace sera prise après que l'instruction stop ait chargé SR. Sur le retour du programme de traitement de trace, l'exécution continuera avec l'instruction qui suit le Stop et le microprocesseur ne sera jamais en condition STOP C'est le seul cas où Trace affecte l'exécution d'une instruction.

V - 5 - 6 - 5 - Traitement détaillé des exceptions: Interruptions

INTERRUPTIONS

Les demandes d'interruption sont effectuées en appliquant un ou plusieurs états actifs sur les broches IPL 2,1,0. L'interruption est alors pendante (en instance). La demande ne sera satisfaite qu'après l'exécution de l'instruction en cours. Si le niveau de priorité de la demande est strictement supérieur au masque d'interruption situé dans le registre d'état celle-ci sera acceptée par le processeur.
Une exception est faite pour **l'interruption de niveau 7** qui ne peut être masquée, elle sera toujours acceptée quelle que soit la valeur introduite dans le masque d'interruption, c'**est une interruption non masquable.**

Une copie interne de SR est effectuée, le processeur est mis en l'état superviseur, le mode Trace est supprimé, le masque d'interruption est mis au niveau de l'interruption prise en compte. Le processeur cherche le N° de vecteur du circuit interrompant en effectuant une lecture alors que les broches de code

fonction (FC2,1,0) sont à 111, R/W à1, AS, LDS, UDS à 0, A5 et A4 à 1 et les trois bits d'adresse A 3,2,1 positionnés au niveau de l'interruption en cours de traitement

Un circuit de décodage devra exploiter ces informations de telle sorte que plaçant un état actif sur la ligne IACK du circuit interrompant (cas d'un périphérique 68000) celui-ci fournisse le N° vecteur (de 64 à 255) sur les 8 lignes de poids faible du bus de données et tire la ligne DTACK du processeur à 0.

Si le N° Vecteur n'est pas généré par le circuit interrompant, une logique externe devra activer la ligne VPA du 68000 pour en demander la génération interne. Les N° fournis iront alors de 25 pour le niveau 1, à 31 pour le niveau 7.
Si la logique externe active la ligne BERR le processeur génère le vecteur Interruption Parasite (N°24)

Lorsque le N° vecteur est obtenu, le processeur sauve PC et SR sur la pile superviseur. La valeur de PC empilée est l'adresse de l'instruction qui aurait été exécutée en l'absence d'interruption.

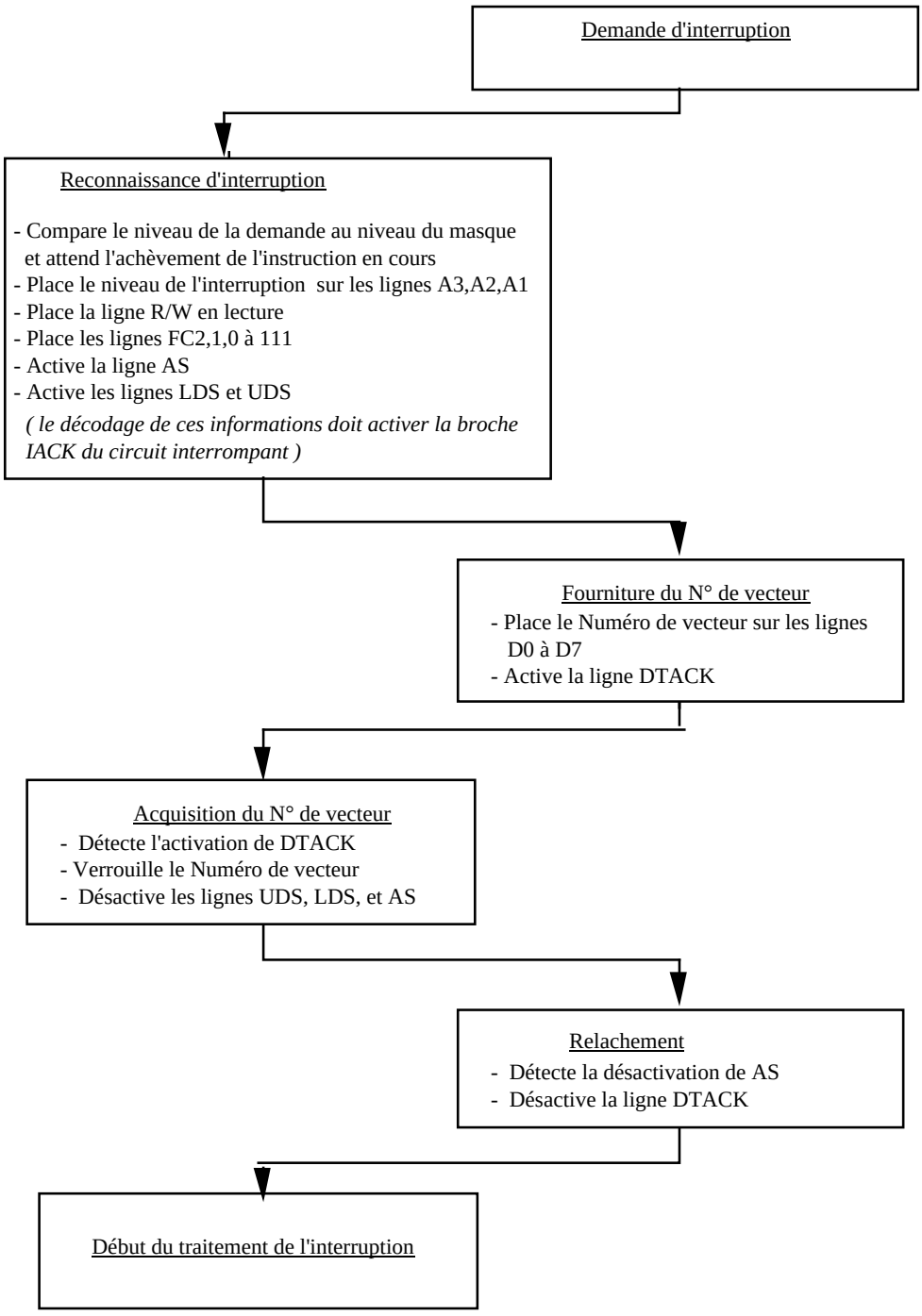
Si le circuit qui demande l'interruption est de la famille 68000 et que cette demande survient avant son initialisation, le vecteur fourni est 0x0F

Le temps nécessaire pour accéder au programme de traitement est de 44 périodes d'horloge, le 68000 effectue 5 opérations de lecture en mémoire et 3 d'écriture. Le cycle de reconnaissance d'interruption étant supposé de 4 périodes.

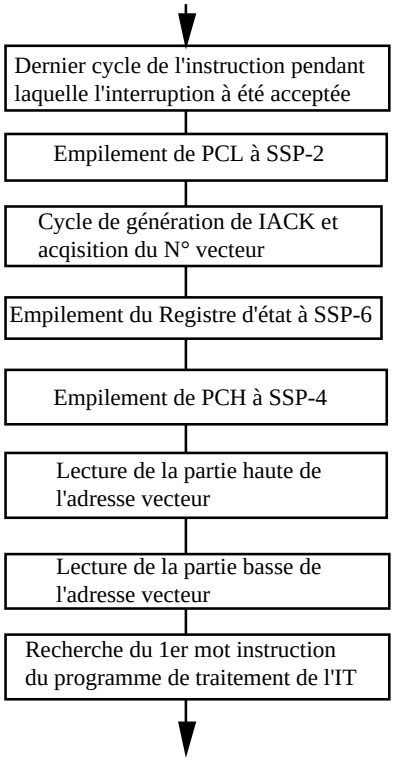
Chronogramme de la reconnaissance d'interruption

Échanges entre le circuit interrompant et le microprocesseur

MICROPROCESSEUR **CIRCUIT INTERROMPANT**



Chronologie d'entrée dans le programme de traitement de l'interruption



V - 5 - 6 - 6 - Traitement détaillé des exceptions: Instruction illégale

INSTRUCTION ILLEGALE

Une instruction illégale est une instruction dont les bits caractéristiques à l'intérieur du mot ne correspondent pas à une instruction du 68000 .
Les instructions dont les 4bits de poids fort sont à 1010 (A-line) et 1111 (F-Line) sont considérées comme instructions inexistantes Lorsqu'une tentative d'exécution déclenche une exception pour instruction illégale, les instructions inexistantes utilisent un vecteur séparé permettant une efficacité de traitement plus grande .

N° vecteur pour instruction illégale 0x04

Le processus de traitement est identique à la précédente

Le temps nécessaire pour accéder au programme de traitement est de 34 périodes d'horloge, le 68000 effectue 4 opérations de lecture en mémoire et 3 d'écriture.

V - 5 - 6 - 7 - Traitement détaillé des exceptions: Instruction inexistante

EMULATEUR LIGNES A et F

Le microprocesseur 68000 préfigure le 68020 vrai microprocesseur 32 bits prévu pour travailler en relation avec ses deux coprocesseurs dont les mots instructions commencent par F. Il est possible de concevoir des systèmes performants équipés de coprocesseurs et ces mêmes systèmes à performances réduites dans lesquels, pour des raisons de coût, le coprocesseur est remplacé par un programme qui émulera les instructions du coprocesseur. Dans ces systèmes une instruction commençant par F déclenchera une exception dont le programme analysera le mot instruction et effectuera la tâche du coprocesseur en un temps évidemment beaucoup plus important. Le 6800 comme le 68020 est prévu pour émuler des instructions ou pseudo instructions commençant par F ou A

N° vecteur pour instruction inexistante A-Line 0xA
N° vecteur pour instruction inexistante F-Line 0xB
Le processus de traitement est identique à la précédente

Le temps nécessaire pour accéder au programme de traitement est de 34 périodes d'horloge, le 68000 effectue 4 opérations de lecture en mémoire et 3 d'écriture.

V - 5 - 6 - 8 - Traitement détaillé des exceptions: Viol de privilège

VIOL DE PRIVILEGE

Lorsque le processeur tente l'exécution d'une instruction privilégiée en mode utilisateur, il y a déclenchement d'une exception pour viol de privilège. Le traitement est identique à une Trappe.
Le Numéro de vecteur est 0x8
Le temps nécessaire pour accéder au programme de traitement est de 34 périodes d'horloge, le 68000 effectue 4 opérations de lecture en mémoire et 3 d'écriture.

V - 5 - 6 - 9 - Traitement détaillé des exceptions: Instructions Trap

INSTRUCTION TRAP

Le processus de traitement des Trappes est le même que celui des autres exceptions. SR est copié, le Registre d'état actif est positionné S = 1,T = 0. Si le mode trace était validé , l'exception Trap sera traitée d'abord normalement puis le mode Trace sera exécuté. Pour les Trappes inconditionnelles, il y en a 16 possibles de Trap 0 à Trap 0xF, le numéro de vecteur est obtenu en ajoutant 32 au N° trap. Pour TrapV trappe conditionnée par l'état du bit V le N° vecteur est 7

PC et la copie de SR sont sauvés dans la pile superviseur . La valeur de PC est l'adresse de l'instruction suivant celle qui a généré la Trappe. PC est chargé avec l'adresse trouvée puis le traitement du programme Trap commence.
Le temps nécessaire pour accéder au programme de traitement est de 38 périodes d'horloge, le 68000 effectue 4 opérations de lecture en mémoire et 3 d'écriture. pour les trappes et de 34 pour trapV

V - 5 - 6 - 10 - Traitement détaillé des exceptions: Division par 0

DIVISION PAR 0

Cette exception est déclenchée par l'exécution d'une division lorsque le diviseur est nul
Le N° vecteur est 0x5.
Le temps nécessaire pour accéder au programme de traitement est de 42 périodes d'horloge

V - 5 - 6 - 11 - Traitement détaillé des exceptions: Instruction check

INSTRUCTION CHECK

Cette exception est déclenchée par le traitement de l'instruction. check lorsque le registre testé sort des limites assignées.
Le N° vecteur est 0x06
Le temps nécessaire pour accéder au programme de traitement est de 44 périodes d'horloge, le 68000 effectue 5 opérations de lecture en mémoire et 3 d'écriture.