Département GEII

Structure des systèmes minimums de traitement de l'information

Philippe Hoppenot

hoppenot@lsc.univ-evry.fr

http://lsc.univ-evry.fr/~hoppenot/presentationfrancaise.html

Toutes les remarques sur le fond et la forme de ce document sont les bienvenues.

Structure des systèmes minimums de traitement de l'information

Le cœur d'un système industriel est le traitement d'information. Le cours d'électronique présente le traitement des informations analogiques. Le cours d'informatique industrielle présente le traitement des données numériques. Il existe deux manières de mettre en œuvre un traitement numérique. La solution câblée, les composants "programmables" en VHDL de type CPLD remplaçant depuis le milieu des années 90 les composants discrets, donne des systèmes optimisés mais peu évolutifs dans lesquels toutes les fonctions sont décrites par le concepteur. La solution programmée, basée sur les microprocesseurs, permet la mise en œuvre de solutions plus complexes et plus évolutives. C'est à cette seconde solution que s'atèle ce cours.

L'architecture de ces systèmes est présentée dans la section I. La section suivante (II) présente le médium de communication entre le processeur et ces périphériques. Le fonctionnement de ces échanges est expliqué dans la section III. Enfin, la section IV aborde la notion fondamentale de l'interruption, interface indispensable entre le programme et le matériel.

I. Architecture d'un système minimum

I.1. Eléments constitutifs

Processeur : Cœur du système, il accueille et met en œuvre le traitement décrit sous la formes d'une suite d'instructions.

ROM : Read Only Memory (mémoire à lecture seule), de type PROM, EPROM ou EEPROM, elle contient le programme et l'environnement nécessaire à son exécution (moniteur, debogueur, boot loader, table de vecteurs d'interruption…). Elle est non volatile.

RAM : Random Access Memory (mémoire à accès aléatoire), de type SRAM (statique RAM) ou DRAM (Dynamique RAM), elle contient les données temporaires utiles au bon déroulement des programmes. Elle est volatile.

Entrées – sorties : Dispositifs permettant au système de communiquer avec l'extérieur : port parallèle, port série, contrôleur de réseau… en fonction des utilisations.

Alimentation : Tous les circuits électronique ont besoin d'une alimentation.

Quartz : Tous les systèmes évoluant dans le temps on besoin d'une référence. Elle est donnée par un quartz qui définit l'unité de temps de base.

I.2. Communication - les bus

L'échange d'information entre le processeur et ses périphériques s'effectue en deux temps. Dans la première phase, le processeur impose l'adresse à laquelle la donnée va être lue ou écrite. Durant la seconde phase, la donnée est échangée.

Un échange d'information s'effectue à travers un bus, ensemble de fils. Chaque fils peut se trouver dans trois états : 0, 1, haute impédance. Trois bus permettent le fonctionnement d'un système minimum : - le bus de données
- le bus d'adresses
- le bus de contrôle

I.2.1. Bus de données

Bus sur lequel circulent les données nécessaires au traitement et les instructions qui réalisent ces traitements. Il est bidirectionnel et commun au processeur et à tous les périphériques.

La taille du bus de données varie de 8 bits à 64 bits voire 128 bits. Cela correspond au nombre de bits transféré en un seul échange.

I.2.2. Bus d'adresses

Bus sur lequel circule l'adresse de la donnée échangée. Il est unidirectionnel : c'est le processeur qui choisit l'adresse.

La taille du bus d'adresses donne le nombre de données que le processeur peut traiter. En combinant la taille du bus de données et la taille du bus d'adresses, on obtient la capacité d'adressage du processeur.

Exemples : 8051 Bus d'adresses : 16 bits => 2^{16} adresses = 2^6x2^{10} adresses = 64 K
Bus de données : 8 bits => 64 Koctets
68360 Bus d'adresses : 32 bits => 2^{32} adresses = 2^2x2^{10} $x2^{10}x2^{10}$ adresses = 4 G
Bus de données : 16 bits => 4 Gmots = 8 Goctets

I.2.3. Bus de contrôle

Bus sur lequel circulent les signaux permettant le fonctionnement du système. Les signaux les plus classiques sont R pour la lecture et W pour l'écriture (Intel), RW pour la lecture-écriture (Motorola), LDS et UDS pour le choix entre le poids fort et le poids faible des données (Motorola). Les et donnent des exemples de lecture et d'écriture avec le 8051.

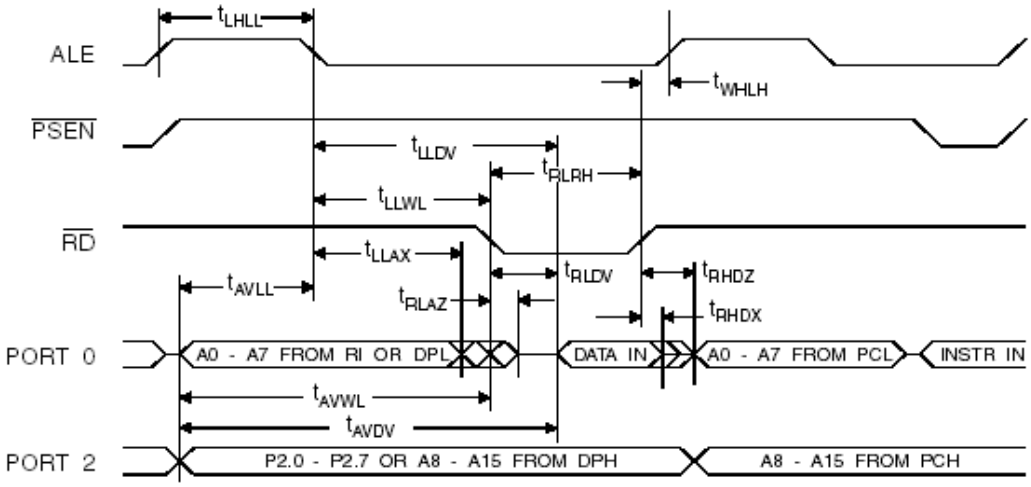


Figure I.1 : *Cycle de lecture dans une mémoire de données avec le 8051.*

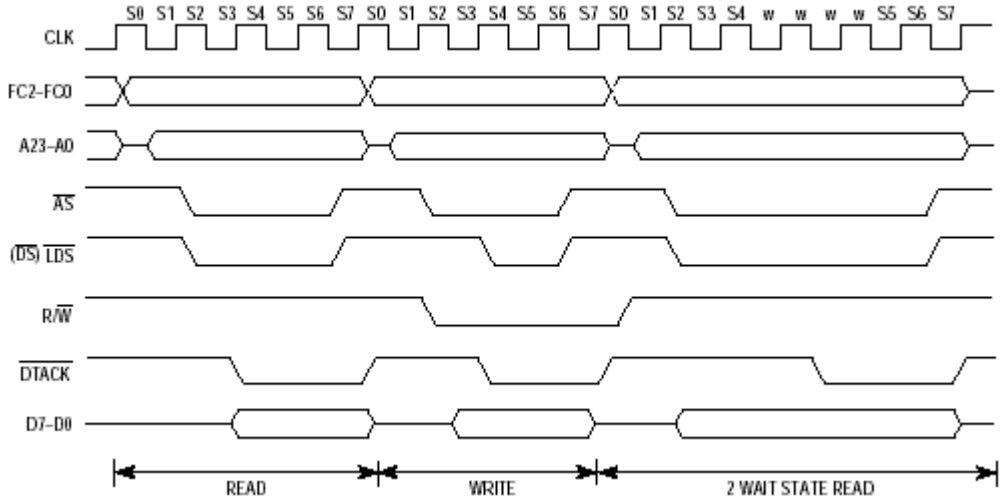


Figure I.2 : *Cycles de lecture et d'écriture avec le 68000.*

D'autres signaux de contrôle seront abordés au fur et à mesure.

I.3. Structure d'un composant périphérique

Un composant périphérique a deux caractéristiques :

- Il réalise une fonction : mémoire, comptage…
- Il s'interface avec le processeur.

Exemples : EEPROM 27C256

- Fonction : mémoire morte effaçable électriquement, non volatile.

Figure I.3 : *Schéma de câblage DIP de la mémoire EEPROM 27C256.*

- Valeur par défaut de chaque point mémoire : 1 logique
- Programmation de la mémoire : mise à 0 de certains points logiques.
  - PROM : définitif
  - EPROM : Impulsion sur le drain d'un transistor MOS. L'effet d'avalanche provoque la conduction. Un bombardement aux ultraviolets permet de rétablir l'état bloqué.
  - EEPROM : La grille du transistor MOS est placé à un potentiel électrique. La mémoire est effaçable électriquement.
- Interfaçage avec le bus : réalisé avec des "buffers" trois états.
  - 0 logique représenté par un potentiel entre 0 V et 0,8 V
  - 1 logique représenté par un potentiel entre 2 V et 5 V
  - Haute impédance représenté par une broche déconnectée du bus (potentiel flottant).

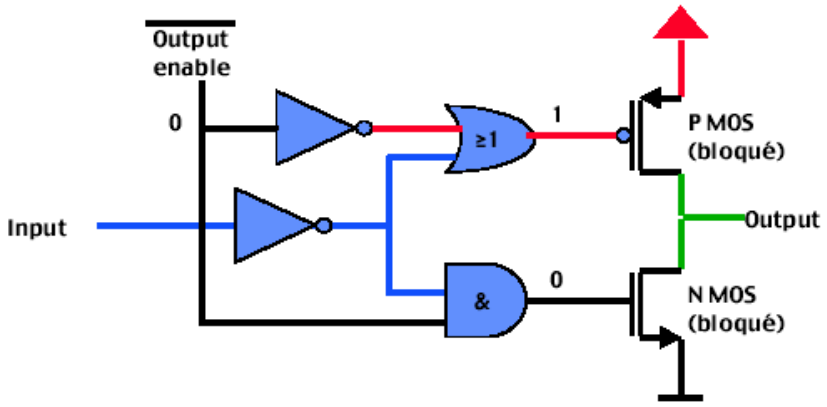


Figure I.4 : *Structure physique du buffer trois états CMOS.*

- Le bus est unidirectionnel.

RAM 5256

- Fonction : mémoire vive, volatile
- Structure d'un point mémoire :
  - SRAM (Static RAM)

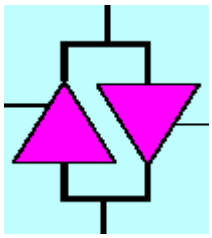


Figure I.5 : *Structure d'un point mémoire SRAM.*

Dans ce cas, deux buffers se rafraîchissent mutuellement.

-DRAM (Dynamic RAM)

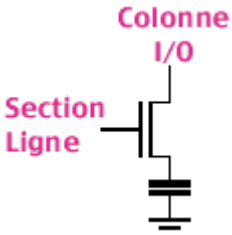


Figure I.6 :*Structure d'un point mémoire DRAM.*

Ici, la décharge du condensateur nécessite une rafraîchissement actif. Plus lentes que les mémoires statiques, elles sont plus denses et donc largement utilisées.

- Interfaçage avec le bus : réalisé avec des "buffers" trois états.
  - 0 logique représenté par un potentiel entre 0 V et 0,8 V
  - 1 logique représenté par un potentiel entre 2 V et 5 V
  - Haute impédance représenté par une broche déconnectée du bus (potentiel flottant).
- Le bus est bidirectionnel. Par convention, le sens du bus est donné vu du processeur. Ainsi, en écriture le bus est en sortie pour le processeur et en entrée pour la mémoire ; en lecture le bus est en entrée pour le processeur et en sortie pour la mémoire.

II. Bus d'un système minimum

Le bus est l'interface entre le processeur et ses périphériques. Il existe deux familles de bus d'adresses et de données : multiplexés et non multiplexés. Dans les deux cas, les deux phases de l'échange (adresses puis données) sont respectées. C'est l'implantation physique des bus qui diffère. Il convient aussi de se demander comment les conflits d'accès au bus sont gérés.

II.1. Bus non multiplexés – Exemple de la famille 68000 de Motorola

Les bus d'adresses et de données du 68000 sont physiquement séparés (Figure II.1). On dit qu'il s'agit de bus non multiplexés.

Le bus d'adresses comprend 23 bits et le bus de données 16 bits. On peut donc adresser 2^{23} mots de 16 bits soit 8 Mmots ou 16 Moctets. On remarque que le bit A0 n'existe pas. Il est en fait remplacé par LDS et UDS qui désignent le poids faible ou le poids fort de la donnée échangée (Figure I.2).

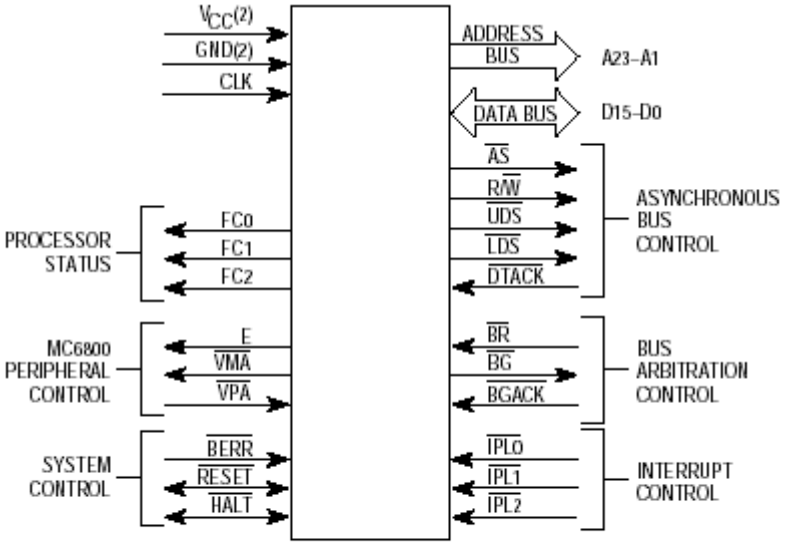


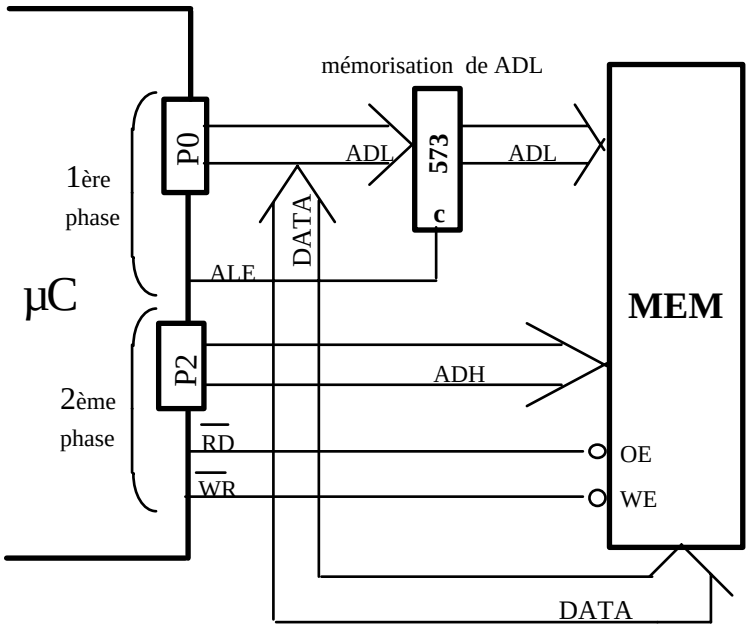
Figure 3-1. Input and Output Signals (MC68000, MC68HC000 and MC68010)

Figure II.1 : *Description des entrées/sorties du 68000.*

L'échange est piloté par les signaux suivants du bus de contrôle :

- AS : Validation des adresses
- R/W : Sens du bus de données : lecture ou écriture
- DTACK : acquittement des données (voir échange asynchrone, III.2)

II.2. Bus multiplexés - Exemple du 8051 d'Intel



Les ports P0 et P2 ne sont plus utilisables en tant que ports

Figure II.2 : *Exemple de bus multiplexé – Cas du 8051.*

Dans le cas du 8051, les bus d'adresses et de données utilisent des ressources physiques communes. En effet, le bus d'adresse est composé de 16 bits et le bus de données de 8 bits soit 24 bits, le tout devant se partager 2 ports (P0 et P2) de 8 bits soit 16 bits. Cette apparente contradiction se résout en tirant profit des deux étapes dans l'échange de données (I.2). Dans la première phase, l'adresse est

présente sur les 16 bits disponibles, les poids faibles sur P0 et les poids forts sur P2. A la fin de cette phase, identifiée par ALE, les poids faibles des adresses sont mémorisées dans une latch (573). Ainsi, le port P0 est-il libéré pour permettre aux données de circuler. La Figure II.2 illustre ce type d'échange.

II.3. Les conflits d'accès au bus

Les bus d'adresses, de données et de contrôle sont communs au processeur et à ses différents périphériques. Dans tous ces échanges, c'est le processeur qui choisit l'adresse de la donnée à transférer et gère l'échange. Ainsi, seul le processeur écrit sur les bus d'adresses et de contrôle. En revanche, tous les périphériques peuvent être amenés à écrire sur le bus de données. Lorsque le processeur initie une lecture, quel périphérique doit répondre ?

Le principe consiste à découper l'espace d'adressage du processeur. Par exemple, pour un système à base de 68000, on peut imaginer le découpage suivant de la mémoire :

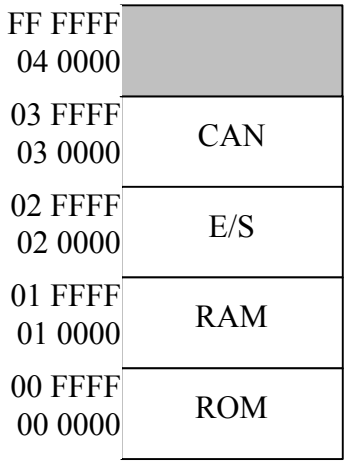


Figure II.3 : *Exemple d'espace d'adressage avec le 68000.*

La Table II.1 donne les adresses correspondant aux zones mémoires définies dans la Figure II.3. On remarque que les bits A17 et A16 sont déterminants pour désigner une zone. On obtient la table d'un décodeur démultiplexeur. La validation de ce circuit est réalisé grâce à la patte AS du 68000. Chaque composant périphérique possède une entrée de type CS (Chip Select, sélection du circuit) qui le met en mode de fonctionnement. Ainsi, un seul périphérique est validé pour un échange donné.

A23-A18	A17	A16	A15-A0	CAN	E/S	RAM	ROM
000000	1	1	0000000000000000	0	1	1	1
000000	1	1	FFFFFFFFFFFFFFFF	0	1	1	1
000000	1	0	0000000000000000	1	0	1	1
000000	1	0	FFFFFFFFFFFFFFFF	1	0	1	1
000000	0	1	0000000000000000	1	1	0	1
000000	0	1	FFFFFFFFFFFFFFFF	1	1	0	1
000000	0	0	0000000000000000	1	1	1	0
000000	0	0	FFFFFFFFFFFFFFFF	1	1	1	0

Table II.1 : *Exemple de décodage d'adresses avec le 68000.*

Dans l'exemple présenté, seuls les bits A17 et A16 sont pris en compte dans le décodage d'adresses. Ainsi, les adresses 10 0000 et 00 0000 sélectionnent le même boîtier. Il s'agit d'adresses image. Un décodage d'adresses est toujours un compromis entre la complexité du décodage et le nombre d'adresses images.

III. Echanges processeur – périphérique

On a vu qu'un échange de données s'effectue en deux étapes :

- une étape d'adressage,
- une étape de transfert de données.

Le processeur est maître de la phase d'adressage. Le transfert de données fait intervenir un périphérique. Deux modes d'échange sont possibles. Dans le mode synchrone, est défini une fois pour toute. Il doit donc tenir compte du périphérique le plus lent. Dans le mode asynchrone, le périphérique informe le processeur qu'il est prêt à recevoir des données ou qu'il a fini de les émettre. Ce mode est quant à lui un peu plus difficile à mettre en œuvre.

III.1. Echange synchrone

Lors d'un échange synchrone, le rythme de l'échange est défini à l'avance, une fois pour toutes. Ainsi, une fois la phase d'adressage terminée, la phase d'échange de donnée commence et s'arrête automatiquement au bout d'un temps fixé. La Figure III.1 illustre l'échange synchrone dans le cas du 8051. Une fois le signal ALE passé à 0, la phase de donnée commence (après t_{LLWL}) et se termine au bout de t_{RLRH}. Le signal RD indique qu'il s'agit d'une lecture. Puis un nouveau cycle peut commencer.

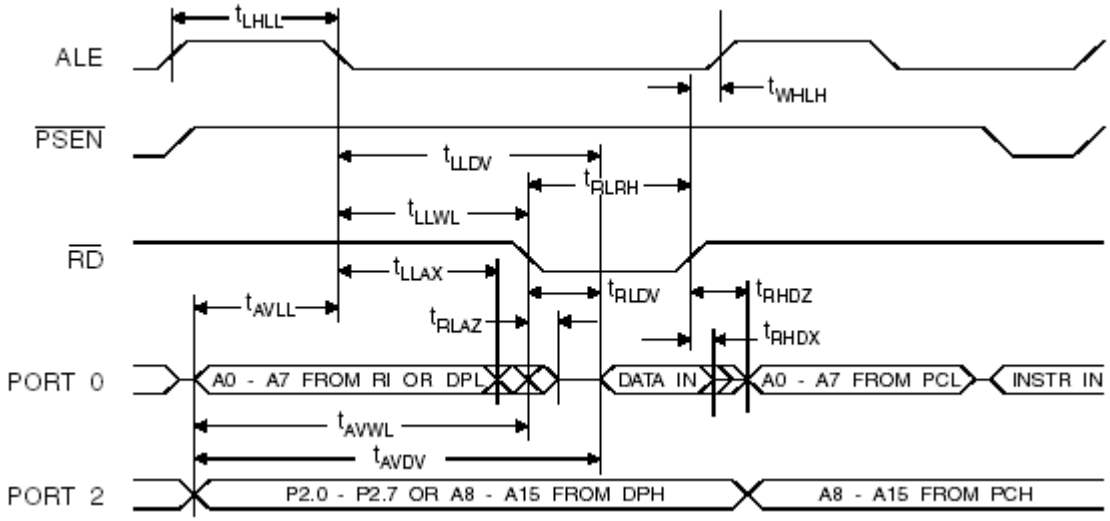


Figure III.1 : *Echange de données synchrone - Exemple du 8051.*

III.2. Echange asynchrone

Lors d'un échange asynchrone, la phase d'adressage est toujours pilotée par le processeur. En revanche, la durée de la phase de transfert de données est indiquée par le périphérique à l'aide d'un signal spécifique. La Figure III.2 illustre l'échange asynchrone lors d'un transfert de données avec le 68000. Une fois la phase d'adressage terminée, la phase de transfert de données commence. Le signal R/W indique qu'il s'agit d'une écriture. Le processeur positionne donc le bus de données D15-D0. Quand le périphérique a fini de lire les données, il l'indique au processeur à l'aide du signal DTACK. Alors seulement le processeur enclenche la procédure de fin de transmission.

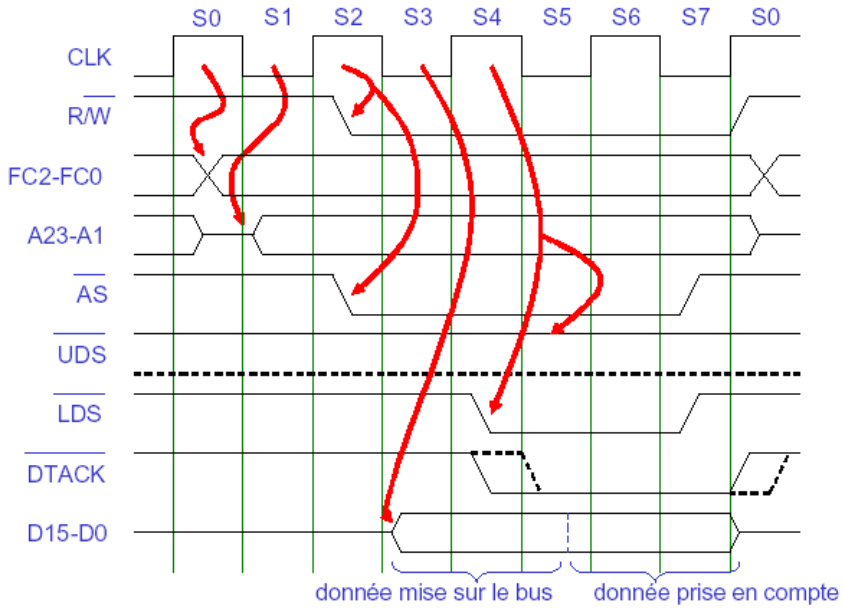


Figure III.2 : *Echange de données asynchrone- Exemple du 68000.*

Que se passe-t-il si le périphérique choisi ne répond pas ? Le processeur attend… Et si le périphérique ne répond toujours pas ? Un mécanisme est prévu pour débloquer la situation : un niveau bas sur l'entrée BERR du processeur initie un traitement particulier (interruption) permettant au processeur de ne pas rester bloquer (Figure III.3). C'est au concepteur du système de prévoir la génération de ce signal à chaque échange. Par exemple, au bout de 16 tops d'horloge, BERR peut être placé à l'état bas interrompre le cycle d'attente.

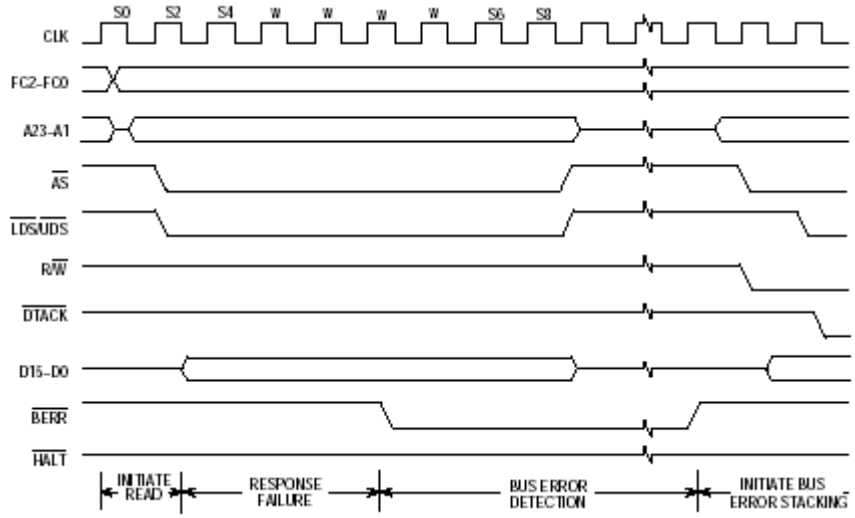


Figure III.3 : *Cycle d'erreur de bus sur le 68000 – signal BERR.*

IV. Gestion d'un système minimum

Une fois le système décrit, il faut pouvoir l'utiliser. On dispose de registres internes (IV.1) servant à contrôler le processeur. Il faut ensuite pouvoir accéder aux données suivant divers modes d'adressage (IV.2). On peut alors écrire des programmes (IV.3). La notion d'interruption permet la gestion d'événements extérieurs (IV.4).

IV.1. Registres internes

Ils permettent de contrôler le processeur. La Figure IV.1 donne les registres du 68000, la Figure IV.2 donne ceux du 8051 et la Figure IV.3 donne ceux du 80386.

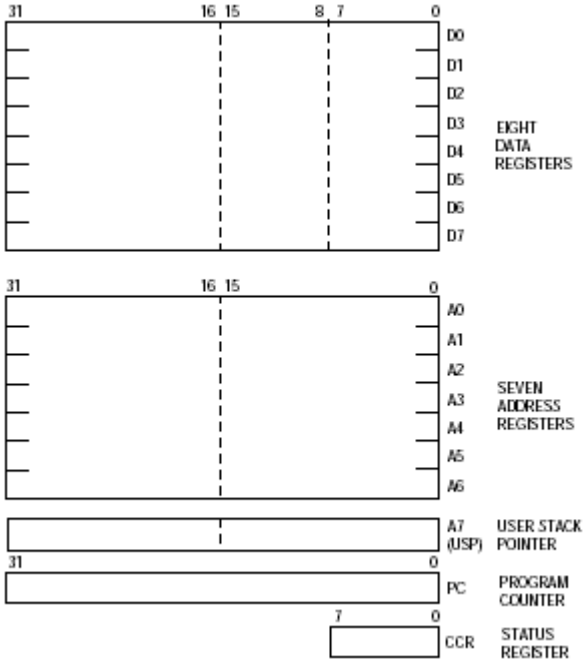


Figure IV.1 : *Registres internes du 68000.*

Pour le 68000, on dispose d'un registre d'état, de huit pointeurs d'adresses et de huit registres de données.

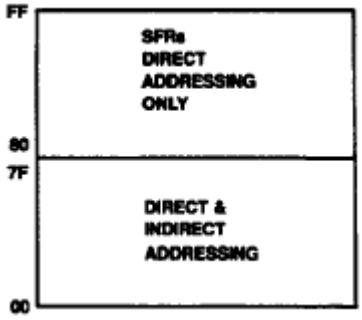


Figure IV.2 : *Registres internes du 8051.*

Le 8051 dispose de 128 octets de mémoire de données (00h-7Fh) et de registres spéciaux (80h-FFh). Parmi ceux-ci, on retrouve un registre d'état, un pointeur d'adresses, deux registres de données…

On retrouve des registres équivalents sur les 80386 (Figure IV.3).

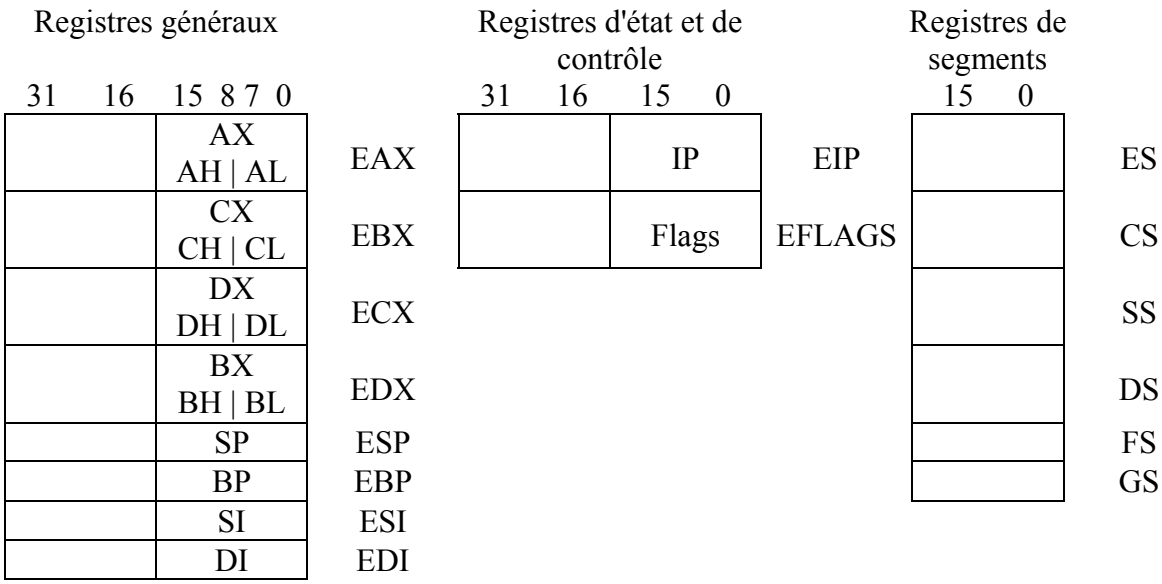


Figure IV.3 : *Les registres du 80386.*

IV.2. Accès aux données – Modes d'adressage

Un processeur ne peut fonctionner qu'avec des données. Il faut pouvoir y accéder. Différents modes sont disponibles, les principaux étant immédiat, direct et indirect.

IV.2.1. Immédiat

La donnée est contenue dans l'instruction.

```
Exemples : 8051      mov      A, #2fh
           68000     move.w   D0, #$4F52
```

IV.2.2. Direct

La donnée est contenue dans un registre.

```
Exemples : 8051      mov      A, R0
           68000     move.b   D1, D3
```

IV.2.3. Indirect

L'adresse est contenue dans l'instruction.

```
Exemples : 8051      mov      @R0, A
           68000     move.l   D0, (A2)
```

IV.3. Notion de programme

Pour traiter les données, on utilise des programmes. Il s'agit de suites d'instructions (IV.3.1), organisées dans uns structure spécifique (IV.3.2). Le tout est codé (IV.3.3) et stocké en mémoire de programme pour pouvoir être exécuté (IV.3.4).

IV.3.1. Instructions

Une instruction permet de manipuler des données. Il en existe de plusieurs types :

- Arithmétique (addition, soustraction…)
- Logique (et, ou…)
- Transfert
- Sauts inconditionnels et conditionnels
- Appel à sous programme

Une instruction comprend au moins :

- Un nom ou mnémonique
- Un ou plusieurs opérandes

IV.3.2. Structure d'un programme

Un programme est une suite d'instructions qui s'exécutent les unes à la suite des autres. Si on tombe sur une instruction de saut, c'est le code se trouvant à l'adresse précisée qui s'exécute.

Un sous-programme est une suite d'instructions qui peut être appelée de plusieurs endroits dans le programme. Il débute par une étiquette pour le nommer et une instruction spécifique qui permet un retour au niveau de l'appel. Pour cela, l'adresse de l'instruction d'appel doit avoir été stockée avant le détournement du programme. La Figure IV.4 montre un exemple d'appel d'un sous-programme en 8051.

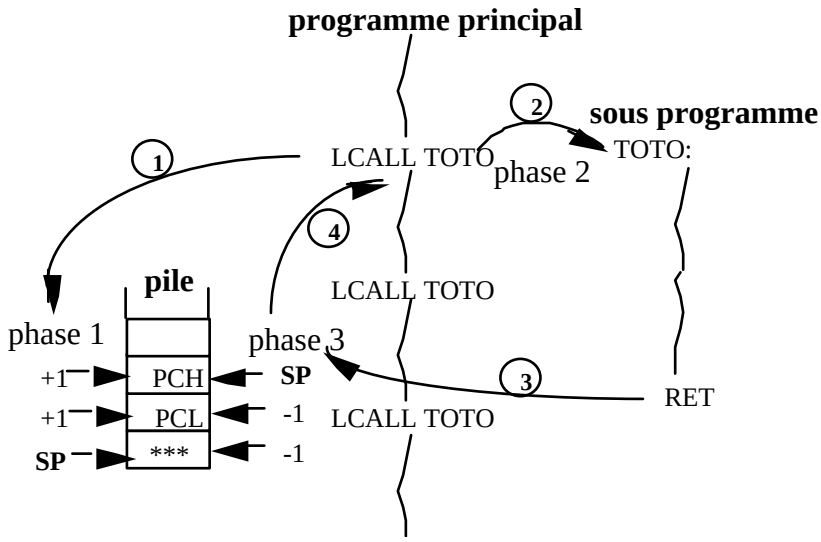


Figure IV.4 : *Exécution d'un sous-programme en 8051.*

IV.3.3. Codage

L'écriture d'un programme se réalise en trois phases principales :

- Edition du code — Fichiers sources
- Compilation, assemblage — Fichiers objets
- Edition des liens — Fichier exécutable, en langage machine

IV.3.4. Exécution d'un programme

Le code machine doit être transféré dans la mémoire de programme du système. On obtient ainsi une liste d'octets décrivant le fonctionnement du programme. Pour exécuter le programme, le compteur de programme est initialisé à l'adresse du premier octet. Le code de l'instruction correspondante est chargé dans l'unité centrale qui effectue l'action associée. Le compteur de programme pointe alors sur l'instruction suivante.

Bien qu'une instruction (en assembleur) soit la plus petite entité accessible au programmeur, son exécution est réalisée en plusieurs phases. Pour le 486 et le Pentium, elles sont au nombre de 5 (Figure IV.5). Dans la phase PF, l'instruction est recherchée dans la mémoire et transférée dans le processeur. Lors du premier décodage (D1), l'instruction est examinée pour déterminer le type d'action à déclencher. La seconde phase de décodage (D2) sert à compléter la première par exemple en déterminant les opérandes associés à l'instruction. La phase d'exécution (EX) met en œuvre l'instruction avec les accès mémoire nécessaires. La dernière phase met à jour les registres internes du processeur. Un cycle machine est consommé par phase (parfois 2 dans D2 et EX).

PF	PreFetch
D1	Decode 1
D2	Decode 2
EX	EXecute
WB	Write Back

Figure IV.5 : *Phases d'exécution d'une instruction.*

La structure de pipeline part de cette décomposition (Figure IV.5) : à chaque cycle machine, toutes les phases décrites ci-dessus sont actives. Cela revient à traiter en permanence cinq instructions, chacune dans une phase différente (Figure IV.6). Une fois le mécanisme enclenché, le rythme d'exécution revient à une instruction par cycle.

Pgm	PF	D1	D2	EX	WB	Etat	Cycle
mov ax,1	mov ax,1						1
add ax,bx	add ax,bx	Mov ax,1					2
cmp ax,15	cmp ax,15	add ax,bx	mov ax,1				3
int 123	int 123	cmp ax,15	add ax,bx	mov ax,1			4
Shl ax,1	shl ax,1	int 123	cmp ax,15	add ax,bx	mov ax,1	Fini	5
		shl ax,1	int 123	cmp ax,15	Add ax,bx	Fini	6
			shl ax,1	int 123	cmp ax,15	Fini	7

Figure IV.6 : *Exécution d'un programme avec la structure de pipeline.*

Le Pentium utilise même deux pipelines en parallèle (U et V). C'est l'architecture de pipeline superscalaire. On arrive théoriquement à un doublement de la cadence d'instructions réalisées. En réalité, deux instructions successives ne sont pas forcément indépendantes : la seconde peut devoir utiliser le résultat de la première. C'est dans la phase D1 que la décision est prise d'une exécution parallèle quand elle est possible : la seconde instruction est traitée par le pipeline V. Sinon, la seconde instruction suit la première dans le pipeline U. Certains compilateurs pour Pentium sont dotés d'une fonction d'optimisation qui organise le programme de telle façon que le plus d'instructions possibles soient traitées en parallèle.

IV.4. Interruptions

L'interruption est un mécanisme fondamental de tout processeur. Il permet de prendre en compte des événements extérieurs au processeur et de leur associer un traitement spécifique. La Figure IV.7 donne le déroulement d'une interruption.

La séquence classique de fonctionnement d'une interruption est la suivante :

0- Détection de l'événement déclencheur
1- Sauvegarde de l'adresse de retour
2- Déroutement vers la routine d'interruption
3- Sauvegarde du contexte
4- Acquittement de l'interruption
L'événement est alors traité.
5- Restauration du contexte
6- Récupération de l'adresse de retour
7- Retour au programme initial

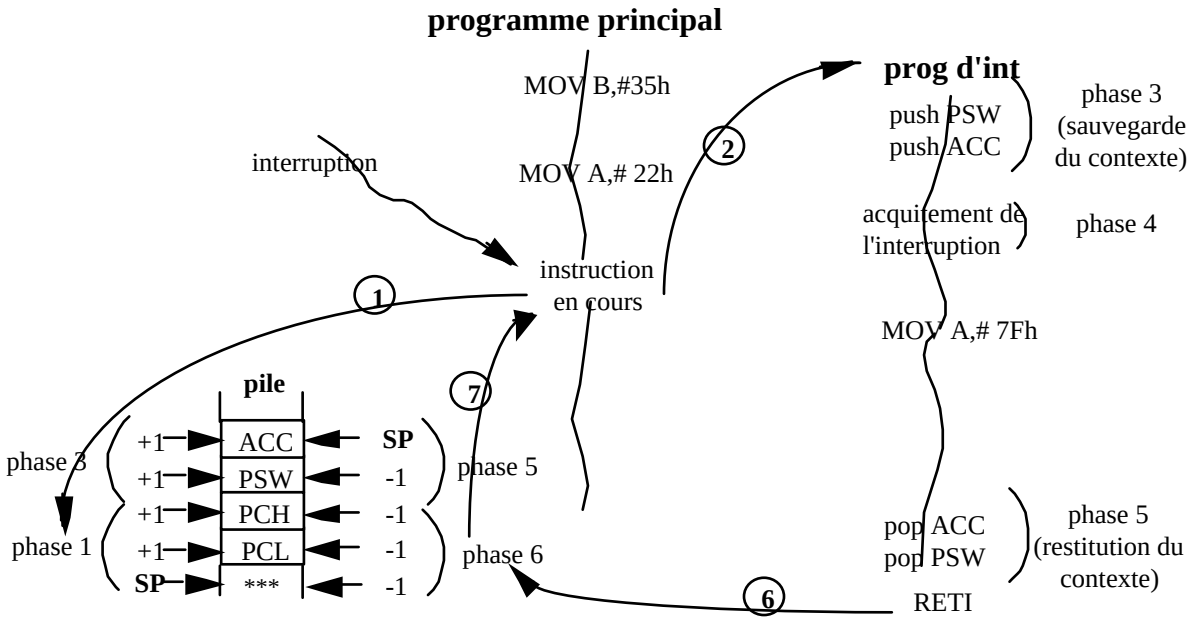


Figure IV.7 : *Déroulement d'une interruption.*

Un point important n'a pas été abordé : quelle est l'adresse de la routine d'interruption ? En effet, pour un sous-programme, l'instruction d'appel est suivie de l'adresse du sous-programme. Pour chaque source d'interruption, cette adresse est donnée par le système. Pour le 68000, une table des adresses d'interruption est disponible à une adresse donnée par le système : VBR (Figure IV.8). Pour le 8051, une adresse figée est affectée à chaque source d'interruption :

- RESET : Saut à l'adresse 0000h de la mémoire de programme
- $\overline{\text{INT0}}$: Saut à l'adresse 0003h de la mémoire de programme
- TIMER0 :saut à l'adresse 000Bh de la mémoire de programme
- $\overline{\text{INT1}}$: Saut à l'adresse 0013h de la mémoire de programme
- TIMER1 :Saut à l'adresse 001Bh de la mémoire de programme
- TI ou RI : Saut à l'adresse 0023h de la mémoire de programme

Vectors Numbers		Address			
Hex	Decimal	Dec	Hex	Space[6]	Assignment
0	0	0	000	SP	Reset: Initial SSP[2]
1	1	4	004	SP	Reset: Initial PC[2]
2	2	8	008	SD	Bus Error
3	3	12	00C	SD	Address Error
4	4	16	010	SD	Illegal Instruction
5	5	20	014	SD	Zero Divide
6	6	24	018	SD	CHK Instruction
7	7	28	01C	SD	TRAPV Instruction
8	8	32	020	SD	Privilege Violation
9	9	36	024	SD	Trace
A	10	40	028	SD	Line 1010 Emulator
B	11	44	02C	SD	Line 1111 Emulator
C	12[1]	48	030	SD	(Unassigned, Reserved)
D	13[1]	52	034	SD	(Unassigned, Reserved)
E	14	56	038	SD	Format Error[5]
F	15	60	03C	SD	Uninitialized Interrupt Vector
10–17	16–23[1]	64	040	SD	(Unassigned, Reserved)
		92	05C		—
18	24	96	060	SD	Spurious Interrupt[3]
19	25	100	064	SD	Level 1 Interrupt Autovector
1A	26	104	068	SD	Level 2 Interrupt Autovector
1B	27	108	06C	SD	Level 3 Interrupt Autovector
1C	28	112	070	SD	Level 4 Interrupt Autovector
1D	29	116	074	SD	Level 5 Interrupt Autovector
1E	30	120	078	SD	Level 6 Interrupt Autovector
1F	31	124	07C	SD	Level 7 Interrupt Autovector
20–2F	32–47	128	080	SD	TRAP Instruction Vectors[4]
		188	0BC		—
30–3F	48–63[1]	192	0C0	SD	(Unassigned, Reserved)
		255	0FF		—
40–FF	64–255	256	100	SD	User Interrupt Vectors
		1020	3FC		—

NOTES:
1. Vector numbers 12, 13, 16–23, and 48–63 are reserved for future enhancements by Motorola. No user peripheral devices should be assigned these numbers.
2. Reset vector (0) requires four words, unlike the other vectors which only require two words, and is located in the supervisor program space.
3. The spurious interrupt vector is taken when there is a bus error indication during interrupt processing.
4. TRAP #n uses vector number 32+ n.
5. MC68010 only. This vector is unassigned, reserved on the MC68000 and MC68008.
6. SP denotes supervisor program space, and SD denotes supervisor data space.

Figure IV.8 : *Listes des vecteurs d'interruption en 68000.*

Pour qu'une interruption puisse être prise en compte, elle doit être validée. Au reset, elles sont toutes masquées, ce qui permet de réaliser des programmes simples sans tenir compte de ce mécanisme. Les interruptions sont rangées par ordre de priorité. Pour le 68000, on valide toutes les interruptions à partir d'un niveau de priorité. Pour le 8051, chaque interruption est validée séparément grâce au registre IE (Figure IV.9). Les cinq bits de poids faible de ce registre permettent d'autoriser les cinq sources d'interruption. Le bit EA est une validation générale qui doit aussi être positionné pour que l'interruption puisse se dérouler. Un bit de IE à "1" valide l'interruption. Un bit à "0" l'invalide.

AFh	AEh	Adh	ACh	ABh	AAh	A9h	A8h	adresse
EA	**-**	**-**	**ES**	**ET1**	**EX1**	**ET0**	**EX0**	A8h

Figure IV.9 : *Registre IE de validation des interruptions en 8051.*