CHAPITRE III: COMPLÉMENTS SUR LES MÉCANISMES DE BASE

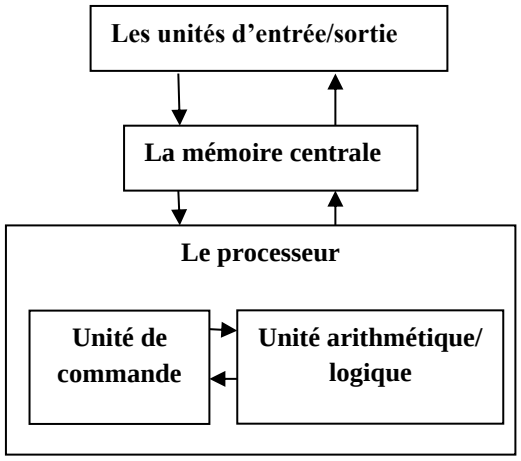
I/ Introduction :

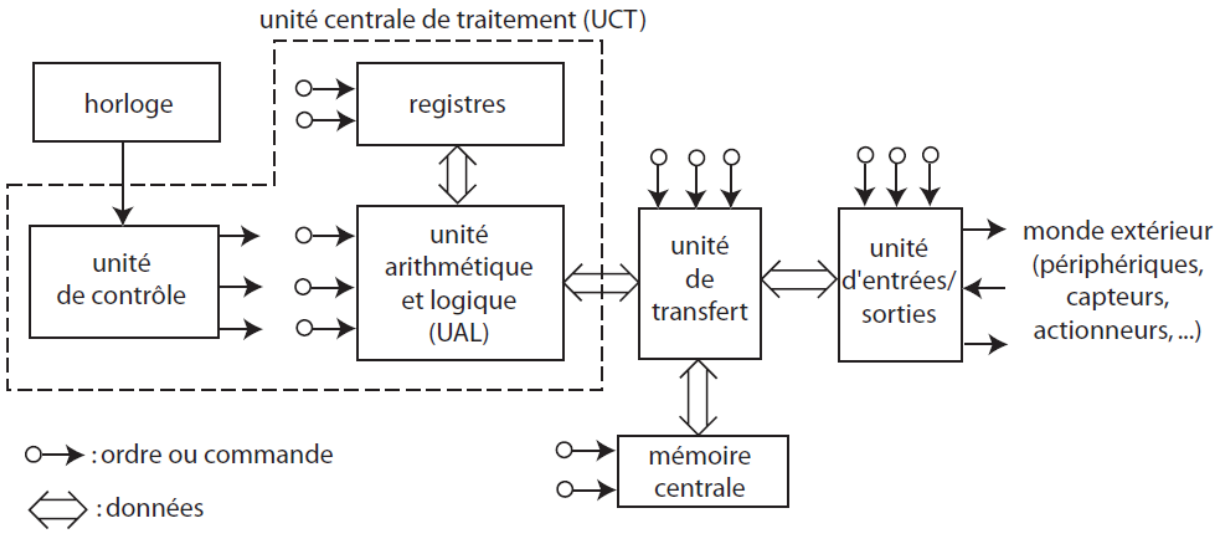
La mise en œuvre d'un système d'exploitation nécessite la connaissance des caractéristiques des éléments hardware, certains éléments participent dans l'exécution des programmes et autres interrompent leurs exécutions. Le système d'exploitation a comme fonctionnalité de les gérer pour conduire à une exécution fiable.
Dans ce chapitre, nous étudierons les composants du processeur et les concepts des interruptions.

II/ L'architecture Von Newmann :

La machine Von Newmann est constituée des éléments :

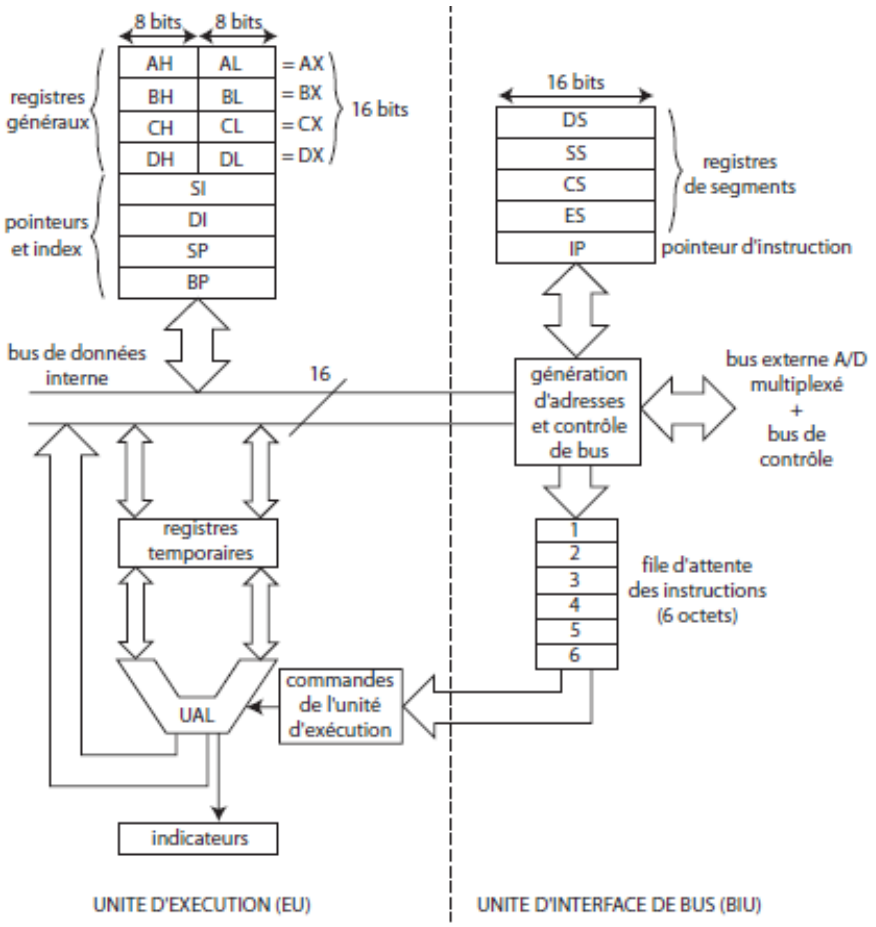
- La mémoire centrale pour sauvegarder les données et les programmes en cours d'exécution.
- Le processeur pour effectuer les calculs et les traitements nécessaires.
- Les périphériques pour l'échange de l'information.





III/ Le processeur :

Unité capable d'exécuter une instruction, elle se compose de :



1/ L'unité de contrôle et de commande :

Sert à contrôler le bon fonctionnement de traitement et commande le déroulement des instructions, elle se compose de :

- *Compteur ordinal ou Pointeur d'instruction (IP, CO)* : registre contenant l'adresse du mot mémoire où se trouve l'instruction suivante à exécuter.
- *Registre d'instruction (RI)* : conserve l'instruction courante pendant son interprétation.
- *Décodeur d'instruction (DI)* : analyse le code opération de l'instruction pour distribuer les différentes commandes élémentaires.
- *Séquenceur* : il ordonne l'ensemble des microcommandes à l'ensemble des unités (mémoire, UAL, …) pour réaliser le traitement de l'instruction.
- *Registre d'état (Processor Status Word PSW)*: contient des informations sur l'état du CPU et les instructions qui viennent d'être exécutées, c'est un masque de n bits contenant des indicateurs

nommés drapeaux (flags) comme : indicateur de débordement, indicateur de parité, indicateur du mode d'exécution, bit de protection, code d'interruptions, …

				O	D	I	T	S	Z		A		P		C
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

CF : indicateur de retenue (carry) ;

PF : indicateur de parité ;

AF : indicateur de retenue auxiliaire ;

ZF : indicateur de zéro ;

SF : indicateur de signe ;

TF : indicateur d'exécution pas à pas (trap) ;

IF : indicateur d'autorisation d'interruption ;

DF : indicateur de décrémentation ;

OF : indicateur de dépassement (overflow).

L'horloge : est un circuit qui transmet régulièrement selon une périodicité déterminée des impulsions électriques, elle définit le fonctionnement séquentiel du processeur de sorte que les cycles machines sont synchronisés avec l'horloge. (L'horloge est un signal carré avec une fréquence fixe comme 3Ghz).

❖ Le microprocesseur 8086 contient 14 registres répartis en 4 groupes :

- **Registres généraux** : 4 registres sur 16 bits

AX = (**AH**,**AL**) ;

BX = (**BH**,**BL**) ;

CX = (**CH**,**CL**) ;

DX = (**DH**,**DL**).

Registre 16 bits	Partie Haute bits 15 à 8	Partie Basse bits 7 à 0	Utilisation
ax	ah	al	accumulateur, multiplication, division
bx	bh	bl	accés mémoire
cx	ch	cl	compteur, repétition (rep), décalage
dx	dh	dl	in, out, multiplication, division
si	-	-	source index, lods, movs
di	-	-	destination index, stos, movs
bp	-	-	base pointer, pile
sp	-	-	stack pointer, sommet de pile

Ils peuvent être également considérés comme 8 registres sur 8 bits. Ils servent à contenir temporairement des données. Ce sont des registres généraux mais ils peuvent être utilisés pour des opérations particulières. Exemple : AX = accumulateur, CX = compteur.

- **Registres de pointeurs et d'index** : 4 registres sur 16 bits.

Pointeurs :

SP : Stack Pointer, pointeur de pile (la pile est une zone de sauvegarde de données en cours d'exécution d'un programme) ;

BP : Base Pointer, pointeur de base, utilisé pour adresser des données sur la pile.

Index :

SI : Source Index ;

DI : Destination Index. Ils sont utilisés pour les transferts de chaînes d'octets entre deux zones mémoire.

- Registres de segments : 4 registres sur 16 bits.

CS : Code Segment, registre de segment de code ;

DS : Data Segment, registre de segment de données ;

SS : Stack Segment, registre de segment de pile ;

ES : Extra Segment, registre de segment supplémentaire pour les données ;

Les registres de segments, associés aux pointeurs et aux index, permettent au microprocesseur 8086 d'adresser l'ensemble de la mémoire.

- ❖ Le microprocesseur 80x86 contient 14 registres répartis en 4 groupes en plus de IP et IR :

A/ Registres 32 bits : Les registres généraux ont une taille de 32 bits et les registres existants du 8086 sont toujours utilisables, mais ont été étendus.

Registre 32 bits	Partie Basse 16 bits	Partie Haute 8 bits	Partie Basse 8 bits
eax	ax	ah	al
ebx	bx	bh	bl
ecx	cx	ch	cl
edx	dx	dh	dl
esi	si	-	-
edi	di	-	-
ebp	bp	-	-
esp	sp	-	-

B/ Registres 64 bits

Registre 64 bits	Partie Basse 32 bits	Partie Basse 16 bits	Partie Haute 8 bits	Partie Basse 8 bits
rax	eax	ax	ah	al
rbx	ebx	bx	bh	bl
rcx	ecx	cx	ch	cl
rdx	edx	dx	dh	dl
rsi	esi	si	-	sil
rdi	edi	di	-	dil
rbp	ebp	bp	-	bpl
rsp	esp	sp	-	spl
r8	r8d	r8w	-	r8b
...	...	...	...	
r15	r15d	r15w	-	r15b

2/ L'unité arithmétique et logique :

L'UAL est composée des circuits dont le but est d'effectuer un traitement (les calculs) sur les opérandes sous le contrôle de l'unité de commande.

IV/ Le format d'une instruction machine :

Le microprocesseur n'est en fait capable de réaliser que 3 types d'opérations :

- **LOAD r,[mem]**, c'est à dire, charger dans un registre r une donnée située en mémoire à une adresse fournie en paramètre
- **STORE [mem],r** qui permet de stocker une donnée contenue dans un registre r dans la mémoire à une adresse fournie en paramètre
- **OP r3, r2, r1** où OP est une opération arithmétique ou logique et qui signifie mettre dans le registre r3 le résultat de r1 OP r2

Dans les microprocesseurs de type x86, on utilise seulement deux opérandes dans la plupart des instructions. On note donc **OP r1,r2** ce qui correspond à **r1 = r1 OP r2**. Dans ce cas l'opérande **r1** est appelée *destination* et l'opérande **r2** est qualifiée de *source*.

On distingue de classes de jeux d'instruction RISC et CISC

RISC Reduced Instruction Set Computer : le format d'instruction précédent et l'adressage mémoire reste simple (i.e. il n'existe que peu de manières différentes d'accéder à la mémoire).

CISC Complex Instruction Set Computer : pour ce type d'architecture on a tendance à combiner une instruction de chargement ou de stockage avec un calcul et l'adressage mémoire peut être complexe.

Exemple : *add [bx + cx * 4 + 8], ax*

V/ Les modes d'adressage :

Le mode d'adressage nous renseigne sur la manière dont l'instruction adresse ses opérandes, sachant que :

Champ adresse

- **Mot mémoire**
- **Registre**

1/ Adressage immédiat :

L'opérande est une constante contenue dans le champ adresse
Exemple : MOV AX, #100 → charger dans le registre AX la valeur 100.

2/ Adressage direct :

L'opérande se trouve en mémoire centrale, le champ adresse contient son adresse effective.
Exemple : MOV R1, 100 → charger dans R1 le contenu du mot mémoire 100

3/ Adressage indirect :

Le champ adresse contient l'adresse d'un pointeur vers la mémoire.
Exemple : MOV R1, (R2) → charger dans R1 le contenu d'une adresse localisé par le pointeur dont son adresse existe dans R2

4/ Adressage indexé :

L'adresse de l'opérande est calculée en additionnant le contenu du champ adresse au contenu du registre d'index (utilisation dans les tableaux).

5/ Adressage basé :

Le contenu de registre de base + le contenu du champ adresse.

6/ Adressage relatif :

Le contenu du compteur ordinal + le contenu du champ adresse (utilisation dans les branchements).

7/ Adressage base avec déplacement:

```
mov ah,[bx+123h] ; <=> mov ah,[ds:bx+123h] => Opcode 8AA72301
```

8/ Adressage indexé avec déplacement:

```
mov ah,[di+123h] ; <=> mov ah,[ds:di+123h] => Opcode 8AA52301
```

9/ Adressage basé indexé:

```
mov ah,[bx+di] ; <=> mov ah,[ds:bx+di] => Opcode 8A21
mov [bp+si],ah ; <=> mov [ss:bp+si],ah => Opcode 8822
```

10/ Adressage basé indexé avec déplacement:

```
mov ah,[bx+si+123h] ; <=> mov ah,[ds:bx+si+123h] => Opcode 8AA02301
```

VI/ L'exécution d'une instruction :

Pour qu'une instruction s'exécute, son cycle passe par deux étapes :

♣ Le cycle de recherche pour chercher l'instruction dans la mémoire puis la décoder.

♣ Le cycle d'exécution durant lequel l'opération spécifiée est effectuée par l'UAL. Alors, l'instruction s'exécute comme suit :

1/ Les instructions sont placées en mémoire centrale dans des adresses séquentielles (sauf dans le cas de branchement).

2/ Le compteur ordinal contient l'adresse de la nouvelle instruction.

3/ Une commande de lecture générée par l'unité de commande pour lire l'instruction dans la mémoire tout en envoyant son adresse dans le bus d'adresse.

4/ L'instruction est transférée dans le bus de données vers le registre d'instruction RI.

5/ Le compteur ordinal donne l'adresse de l'opérande qui est envoyé vers la MC.

6/ Le code opération est transmet au décodeur qui détermine le type de l'opération et le transmet au séquenceur (5, 6 sont faites en même temps).

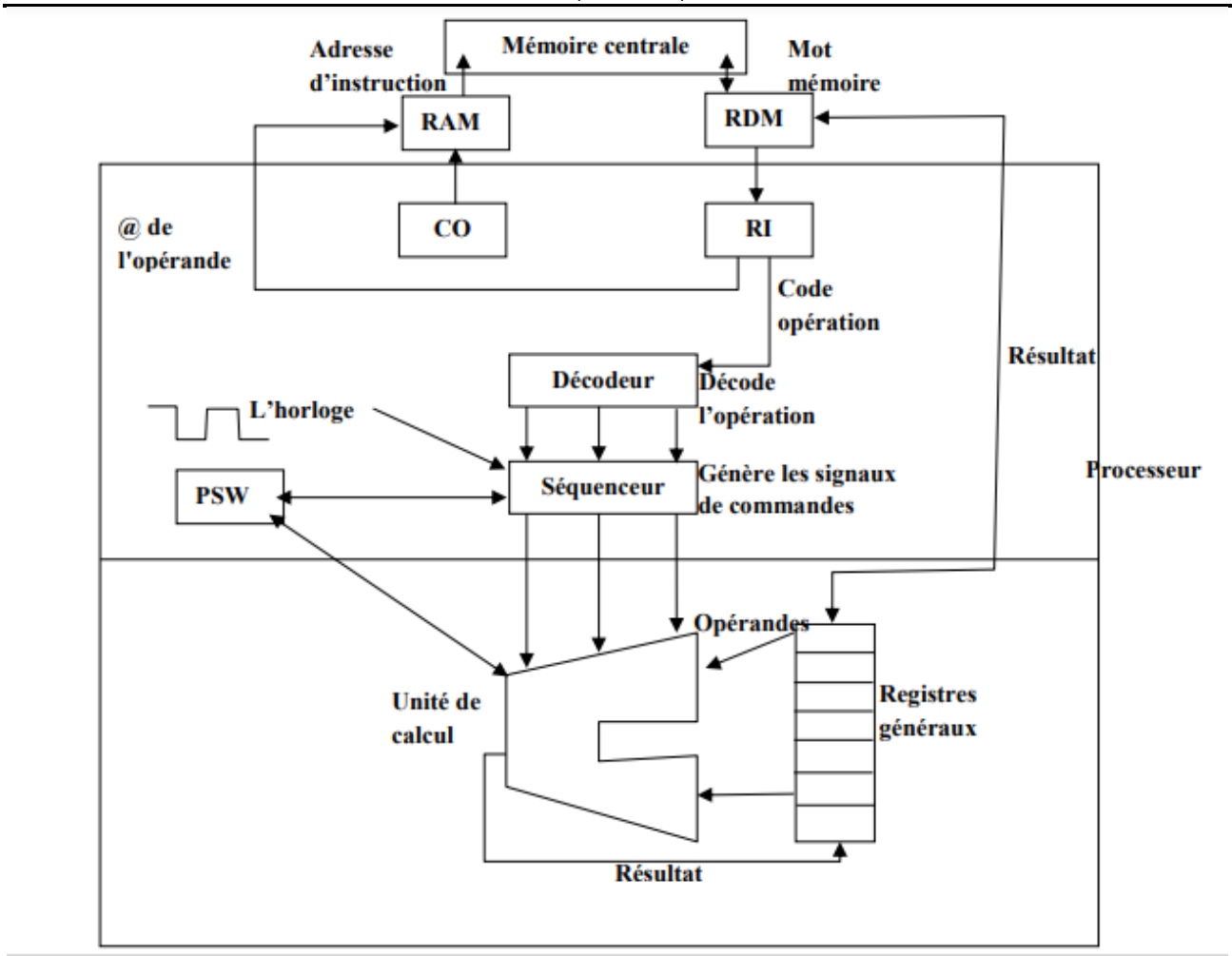
7/ Le séquenceur envoie les signaux de commandes vers la mémoire pour transmettre l'opérande

8/ L'opérande est transmis vers l'UAL.

9/ Le séquenceur envoie un signal de commande vers l'UAL pour l'exécution de l'opération choisie.

10/ Le compteur ordinal s'incrémente.

11/ Dans le cas de mémorisation de résultat, le séquenceur envoie un signal de commande d'écriture dans la mémoire.


Mémoire centrale
Adresse d'instruction
Mot mémoire
RAM
RDM
@ de l'opérande
CO
RI
Code opération
Résultat
Décodeur
Décode l'opération
L'horloge
PSW
Séquenceur
Génère les signaux de commandes
Processeur
Opérandes
Unité de calcul
Registres généraux
Résultat

VII/ Les interruptions :

Dans un ordinateur, existent deux types de programmes :

- Les programmes usagers, qui font un calcul utile.
- Les programmes du système, qui font un travail de supervision de tous les évènements arrivant à la machine.

Ces deux types de programmes se partagent le processeur de telle sorte de s'occuper des programmes utilisateurs et de prendre en considération les évènements produits.

Les évènements produits dans la machine sont deux types :

- Les évènements synchronisés liés aux programmes comme : la division par zéro, tentative d'accès à une zone interdite, …
- Les évènements asynchrones liés aux matériels comme une fin d'opération d'E/S, signal d'horloge, …

Ces évènements sont la manière dont le processeur contrôle continuellement l'état d'une ressource, sont nommés les interruptions.

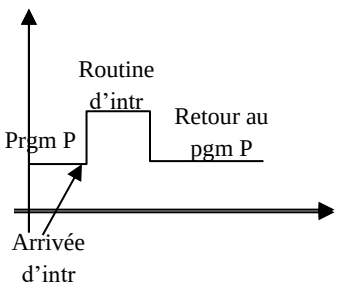
1/ Définition :

Une interruption constitue un mécanisme pour lequel les modules (E/S, mémoire, processus) peuvent interrompre le traitement normal du processeur.

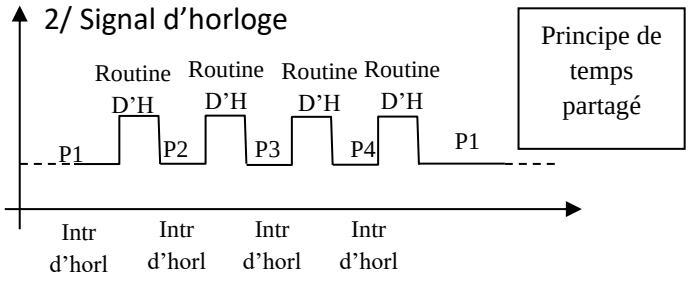
L'interruption est une réponse à un évènement qui interrompe l'exécution des programmes en cours à un point observable du processeur central, elle se traduit par un signal envoyé au processeur, elle permet de forcer le processeur à suspendre l'exécution du programme en cours et à déclencher l'exécution d'un programme prédéfini appelé « **routine d'interruption** ».

Exemple :

1/ Fin d'E/S



2/ Signal d'horloge



Principe de temps partagé

2/ <u>Les types des interruptions</u> :

On distingue deux types d'interruptions :

- Les interruptions externes liées aux matériels.
- Les interruptions internes liées aux programmes.

a/ *<u>Les interruptions externes</u>* causées par des organes externes au processeur comme les unités d'E/S, on peut citer :

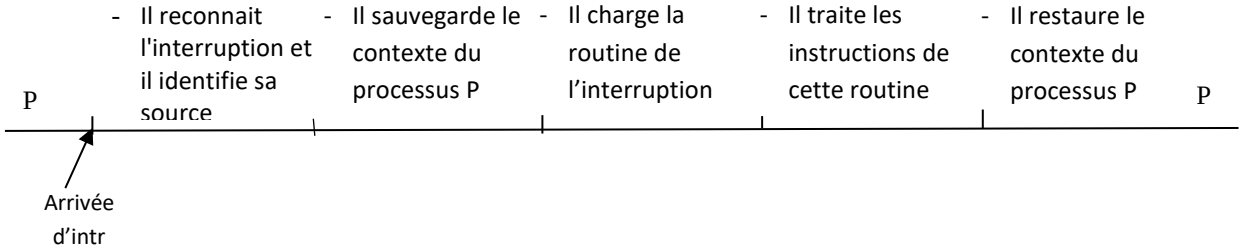
- Un périphérique devient prêt.
- Réinitialisation du système.
- Une erreur durant une E/S.
- Une fin d'E/S.

b/ *<u>Les interruptions internes</u>* dont on peut les diviser en deux types :

- *Les interruptions de l'horloge* : à travers ces interruptions, le SE gère le temps d'exécution entre l'ensemble des processus dans le système.
- *Les déroutements* : sont causées par des erreurs lors de l'exécution des programmes, un déroutement ne doit pas être masqué ou retardé, comme exemple :
  - Tentative d'exécution d'une opération interdite ou invalide.
  - Violation d'accès à une zone protégée ou inexistante.
  - Division par zéro.
  - Débordement arithmétique.
- *Les appels au superviseur* : sont causés par les instructions d'appel à des fonctions d'E/S système comme l'instruction de lecture à partir du clavier, l'instruction d'impression, …

3/ <u>Mécanisme de gestion des interruptions</u> :

Quand un processus utilisateur **p** est en cours d'exécution, dans un temps t, et le processeur reçoit une interruption quelconque plus prioritaire, alors :



Plus précisément, la connaissance d'une interruption peut avoir lieu à des moments différents :

- *Pour une interruption externe* : le processeur possède d'un bit dit bit d'interruption qui prend la valeur 1 par le hardware quand une interruption est déclenchée par un périphérique. Le processeur teste la valeur de ce bit avant de commencer l'instruction suivante (alors les interruptions externes sont prises en compte au début de chaque instruction).
- *Pour une interruption interne* : elle aura lieu le moment de l'exécution d'une instruction (surtout un déroutement):
  - Dans l'étape de décodage (comme la tentative d'exécution d'une instruction non existante, ou une instruction privilégiée en mode esclave).
  - Dans l'étape de recherche de l'opérande (dans le cas de violation de mémoire).
  - Dans l'étape de déclenchement de l'opération (dans le cas de dépassement de capacité).

Pour identifier la source d'une interruption, la technique utilisée couramment est de consulter un vecteur de bits où chaque bit est associé à un périphérique.

N.B : sachant que :

- **Un contexte** du programme c'est l'ensemble des informations nécessaires pour la reprise d'un processus après son interruption, il contient : son compteur ordinal, son mot d'état, l'accumulateur et le contenu des registres généraux.
- L'opération de sauvegarde de contexte d'un processus et le chargement ou la restauration du contexte d'un autre processus est nommée : « **Commutation de contexte** ».

4/ Les conditions d'arrivée d'une interruption :

Pour que le processeur prenne en charge une interruption juste après son arrivée, il faut vérifier les conditions suivantes :

- Le processeur doit être dans un point observable, d'un autre terme, le code, qui est en train d'être exécuté, doit être un code divisible où on peut l'interrompre à tout moment et dans n'importe quelle instruction. (On dit un code indivisible dans le cas inverse)
- Le système d'interruption doit être actif, dont le processeur contient un mécanisme d'activation et de désactivation du système d'interruption. Il désactive ce système s'il est en train d'exécuter des instructions protégées contre les interruptions (exemple : les instructions de sauvegarde de contexte).
- L'interruption doit être plus prioritaire que le programme en cours.
- L'interruption doit être démasquée, où on peut masquer une interruption à cause de sa priorité. Quand une interruption masquée arrive, elle sera retardée.

5/ Les priorités entre les interruptions :

Durant un temps déterminé, on peut avoir plusieurs demandes d'interruptions simultanées, et pour les traiter d'une manière équitable le système définit un certain ordre de priorité entre les niveaux d'interruptions, ces priorités peuvent être fixes ou modifiables.