République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
L'université Mohamed Boudiaf - M'Sila –

Faculté de mathématiques et d'informatique

2ème Année Licence (2L)

Système d'exploitation 1 (SE 1)

Semestre : 04

2023/2024

Réalisé par

Dr. DABBA ALI

➢ Contactez-nous

alidabba@gmail.com

ali.dabba@univ-msila.dz

- En cas de problèmes ou de difficultés, me contacter ou contacter votre enseignant TD / TP
- Nous sommes à votre disposition pour vous aider

CHAPITRE 2

Mécanismes de Base d'Exécution des Programmes

Plan

I. Introduction

II. Architecture et technologie des ordinateurs

III. Machine de VON-NEUMANN

IV. Les étapes de cheminement d'un programme dans un système

V. Le processus

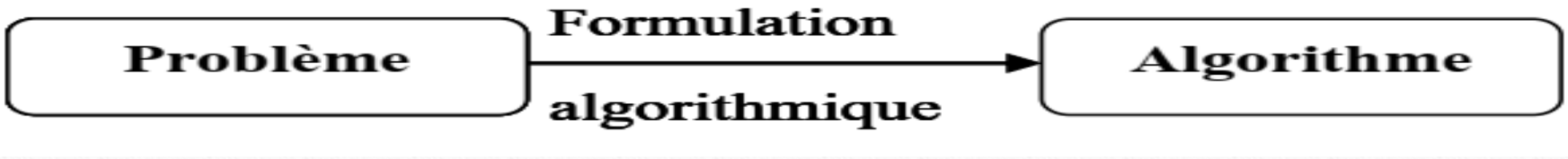
VI. Les Systèmes d'Interruption

I. Introduction

L'homme a utilisé l'ordinateur pour résoudre ses problèmes d'une manière automatique. Ces problèmes sont formulés en programmes, puis exécute par la machine matérielles.

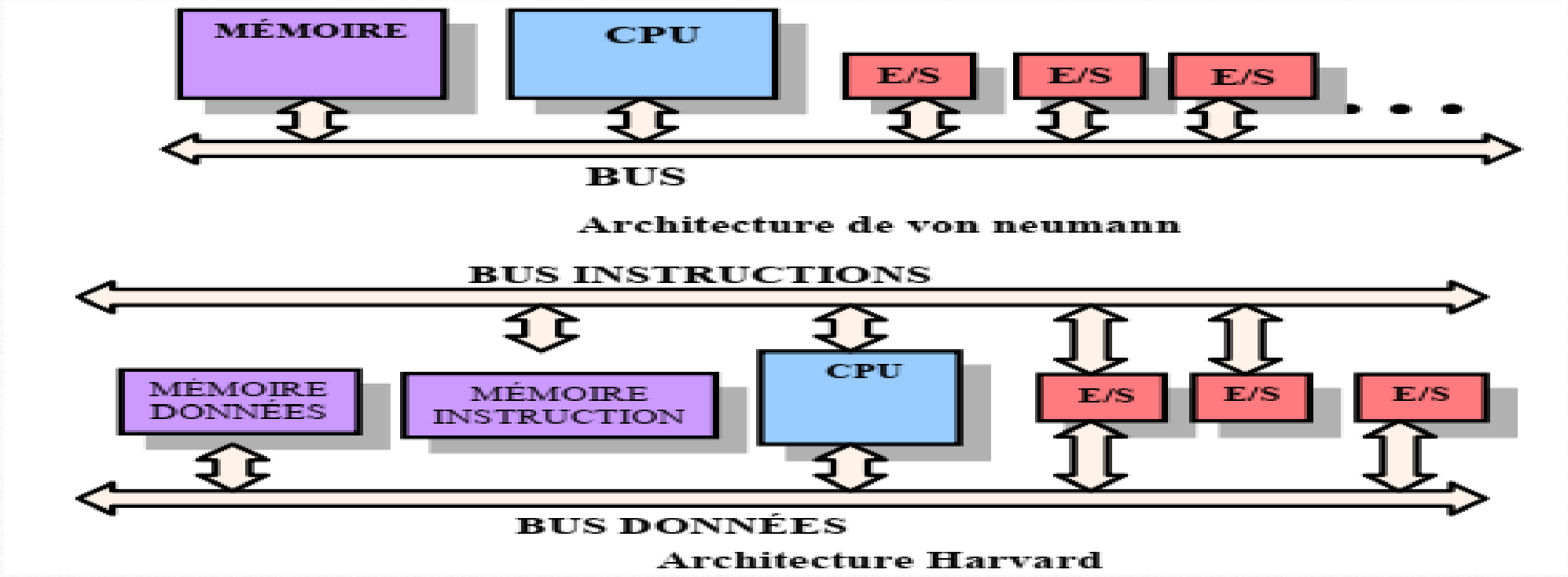
Pour qu'un problème puisse être interprété, exécuté et donne les résultats attendus par le processeur de la machine, il faut qu'il soit formulé dans un langage compréhensible par la machine et il faut que le système d'exploitation pilote son exécution. Pour le faire, le programmeur commence au début d'écrire son problème d'une manière structurée sous forme d'un algorithme.

I. Introduction



II. Architecture et technologie des ordinateurs

- HARVARD & VON NEUMANN.



III. Machine de VON-NEUMANN

Définition 2.1

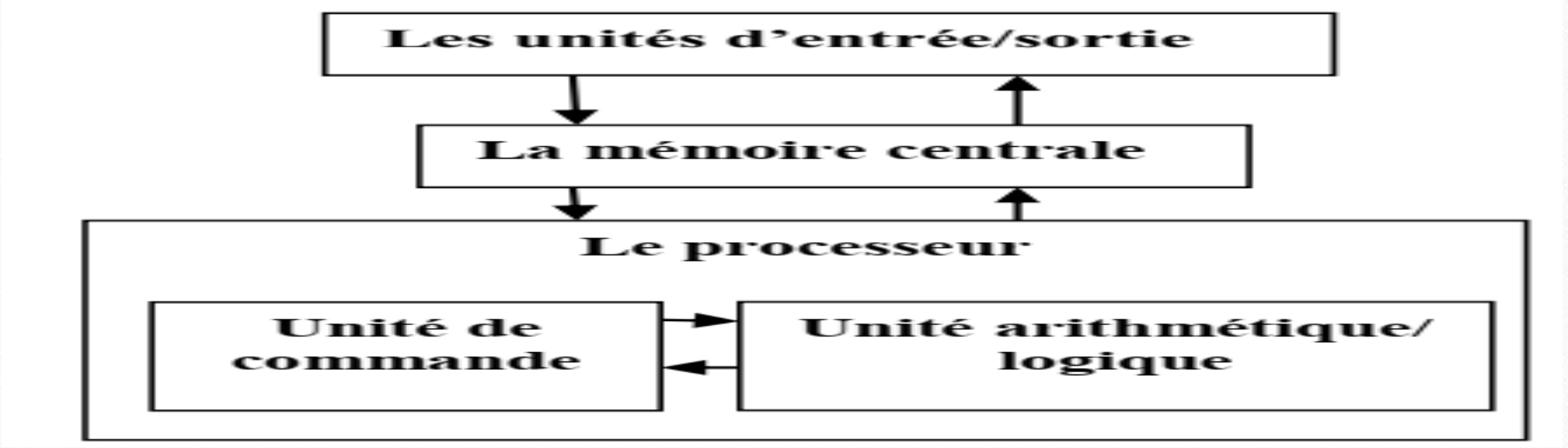
Une machine de **Von-Neumann** est un calculateur électronique **à base de mémoire** dont les composants sont :

→ Mémoire Centrale (MC).

→ Processeur ou Unité Centrale (UC) ; pour effectuer les calculs et exécuter les instructions.

→ Unités périphériques ou d'Entrée/Sortie (E/S).

III. Machine de VON-NEUMANN



- Ces dispositifs permettent la mise en œuvres des fonctions de base d'un ordinateur : le stockage de données, le traitement des données, le mouvement des données et le contrôle.

1) Unité Centrale de Traitement (CPU, Central Process Unit)

Appelée aussi "**Processeur Central**" (PC), elle est composée de :

- L'Unité de Commande (UC).
- L'Unité Arithmétique et Logique (UAL).
- Les Registres (Données, Adresses, Contrôle, Instruction).
- Bus Interne.
- Le processeur exécute des instructions ou actions d'un programme. C'est le cerveau de l'ordinateur. Une **action** fait passer, en un temps fini, le processeur et son environnement d'un **état initial** à un **état final**. Les actions sont constituées de suites d'instructions **élémentaires**.

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.

L’unité de contrôle et de commande (UC)

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

- **Compteur Ordinal (CO)**
- **Registre d’Instruction (RI)**
- **Décodeur d’instruction (DI)** :
- **Séquenceur**
- **Registre Mot d’Etat (PSW, Program Status Word)**

Registre Mot d'Etat (PSW, Program Status Word)

- Les valeurs courantes des **codes conditions** (**Flags**) qui sont des bits utilisés dans les opérations arithmétiques et comparaisons des opérandes.
- Le **mode d'exécution**.
  - *Mode **privilégié** ou **maître** (ou **superviseur**)*.
  - *Mode **non privilégié** ou **esclave***
- **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).

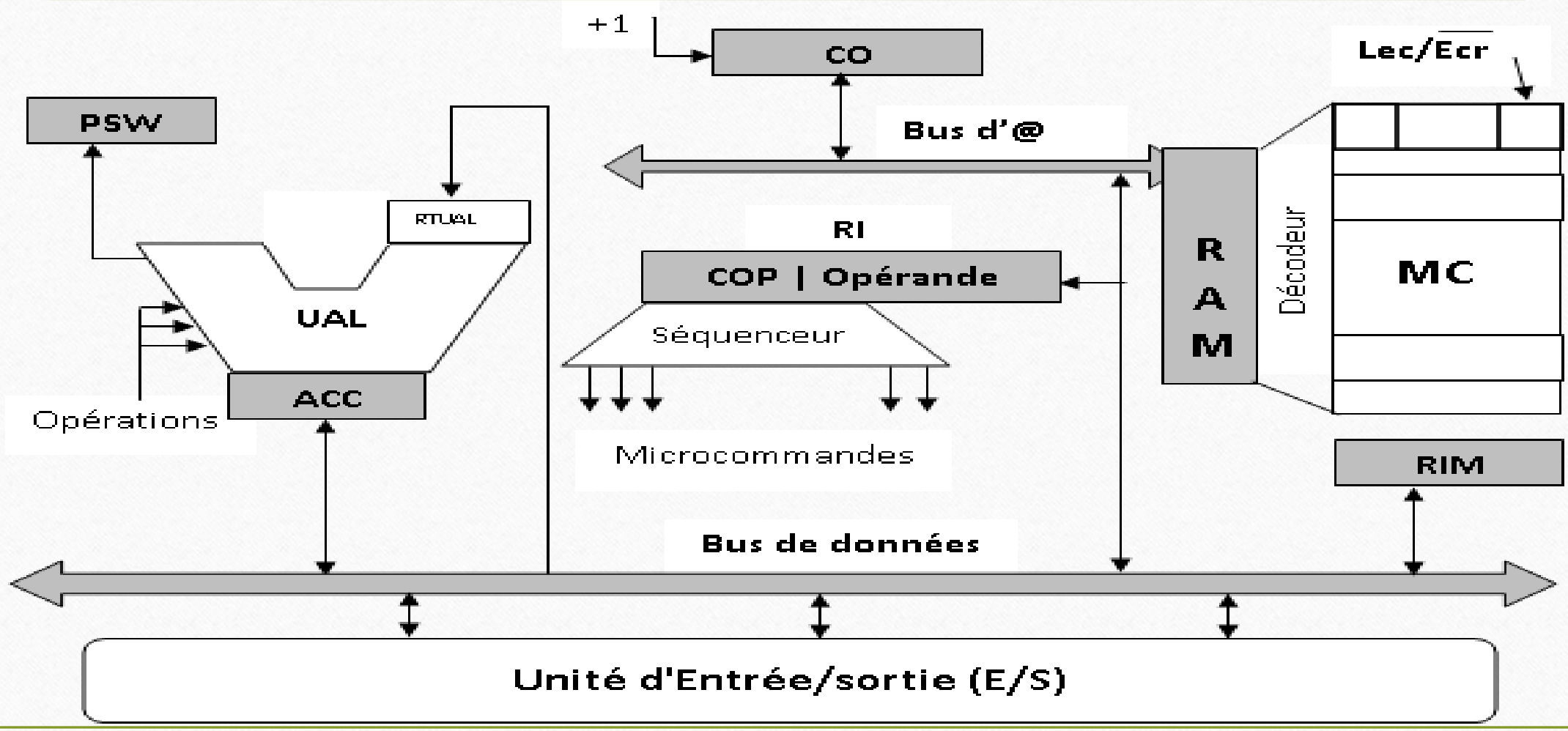
Registres du processeur central

- Les registres sont une sorte de mémoire interne à la CPU, à accès très rapide qui permettent de stocker des résultats temporaires ou des informations de contrôle.
- Le nombre et le type des registres implantés dans une unité centrale font partie de son architecture et ont une influence importante sur la programmation et les performances de la machine.
- Le processeur contient d'autres registres dans la plupart des machines, citons :

Registres du processeur central

- **Les registres généraux**
- **Registre d'index**
- **Registre de base**
- **Registre pointeur de pile** : utilisé pour l'accès à la pile.
- **Accumulateur** : utilisé pour sauvegarder les résultats surtout dans la multiplication et l'addition
- **Registres Banalisés**

Machine de VON-NEUMANN



Cycle d’exécution d'une instruction

Pratiquement, toutes les instructions élémentaire fait suivant un cycle comprenant trois phases principales :

Fetch → Decode → Execute.

Cycle d’exécution d'une instruction

Définition 2.2

- Le cycle d’exécution du processeur est divisé en deux niveau :

  cycle de recherche (Fetch) + cycle d’exécution (Execute) = cycle indivisible l’arrêt ne peut se faire qu’à la fin de la phase Execute (Points observable qui correspondent en général au début et fin d’instruction.)

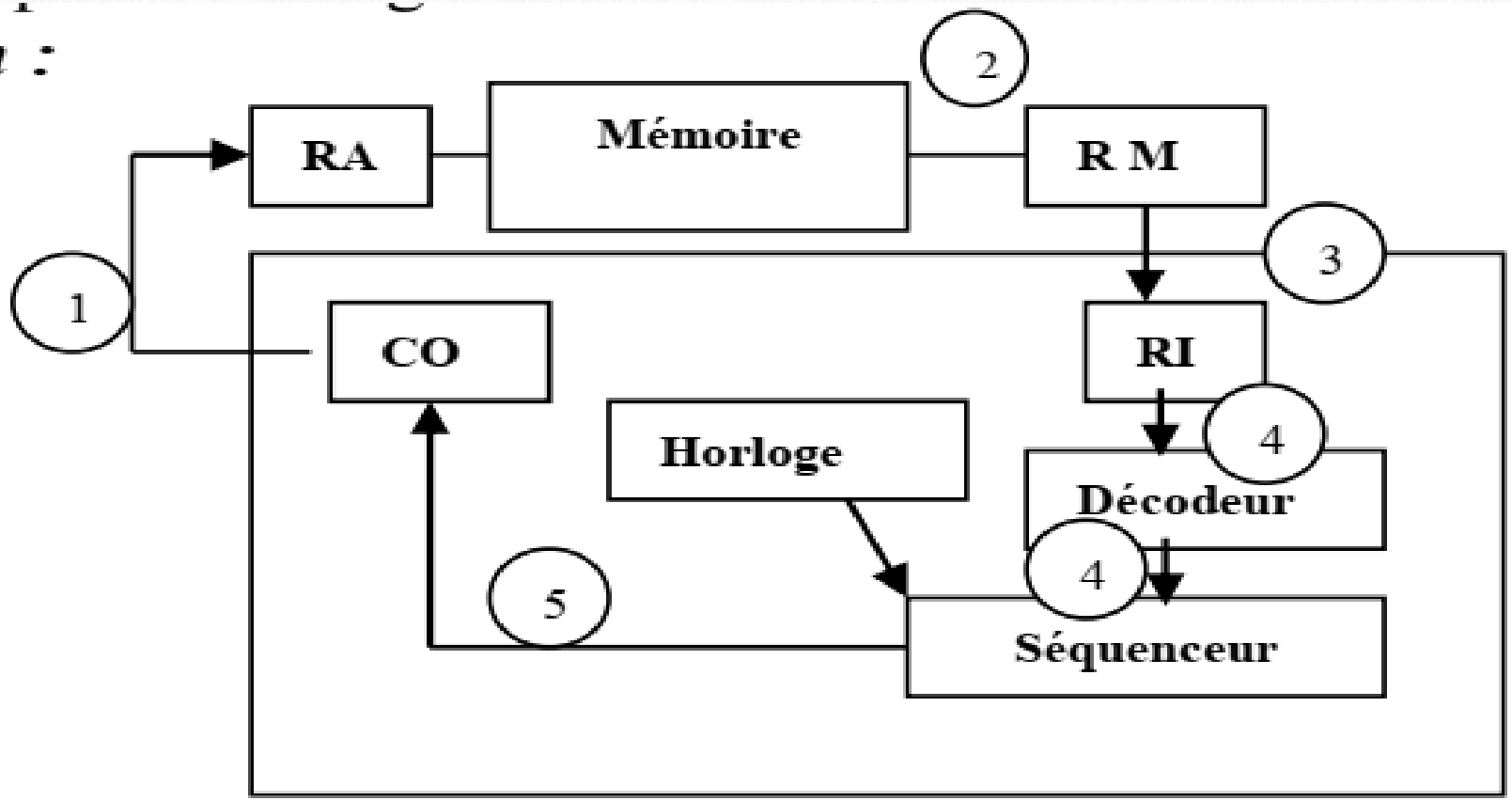
Cycle d’exécution d'une instruction

Phase 1 (Recherche de l'instruction à traiter)

1) Le CO contient l'adresse de l'instruction suivante du programme. Cette valeur est recopiée dans le registre d’adresse (RA) en transitant par le bus «adresse»
2) Une impulsion de lecture généré par l’UC qui copier le contenu de la case mémoire sélectionnée dans le registre de donnée (RM) par le bus «données» (RM fonctionnait comme un registre tampon pour toutes les lectures ou écritures en mémoire)
3) L'instruction du registre de donnée (RM) est stockée dans le registre instruction du processeur (RI) en transitant par le bus «instruction».

Cycle d’exécution d'une instruction

Fetch :



Cycle d’exécution d'une instruction

Phase 2 (Décodage de l’instruction et recherche de l'opérande)

1) Cette instruction courante est décodée à destination de l'UAL ; Le registre d'instruction contient maintenant le premier mot de l'instruction qui peut être codée sur plusieurs mots. Ce premier mot contient le code opératoire qui définit la nature de l'opération à effectuer (addition, rotation,...) et le nombre de mots de l'instruction.
2) L'unité de commande transforme l'instruction en une suite de commandes élémentaires nécessaires au traitement de l'instruction.

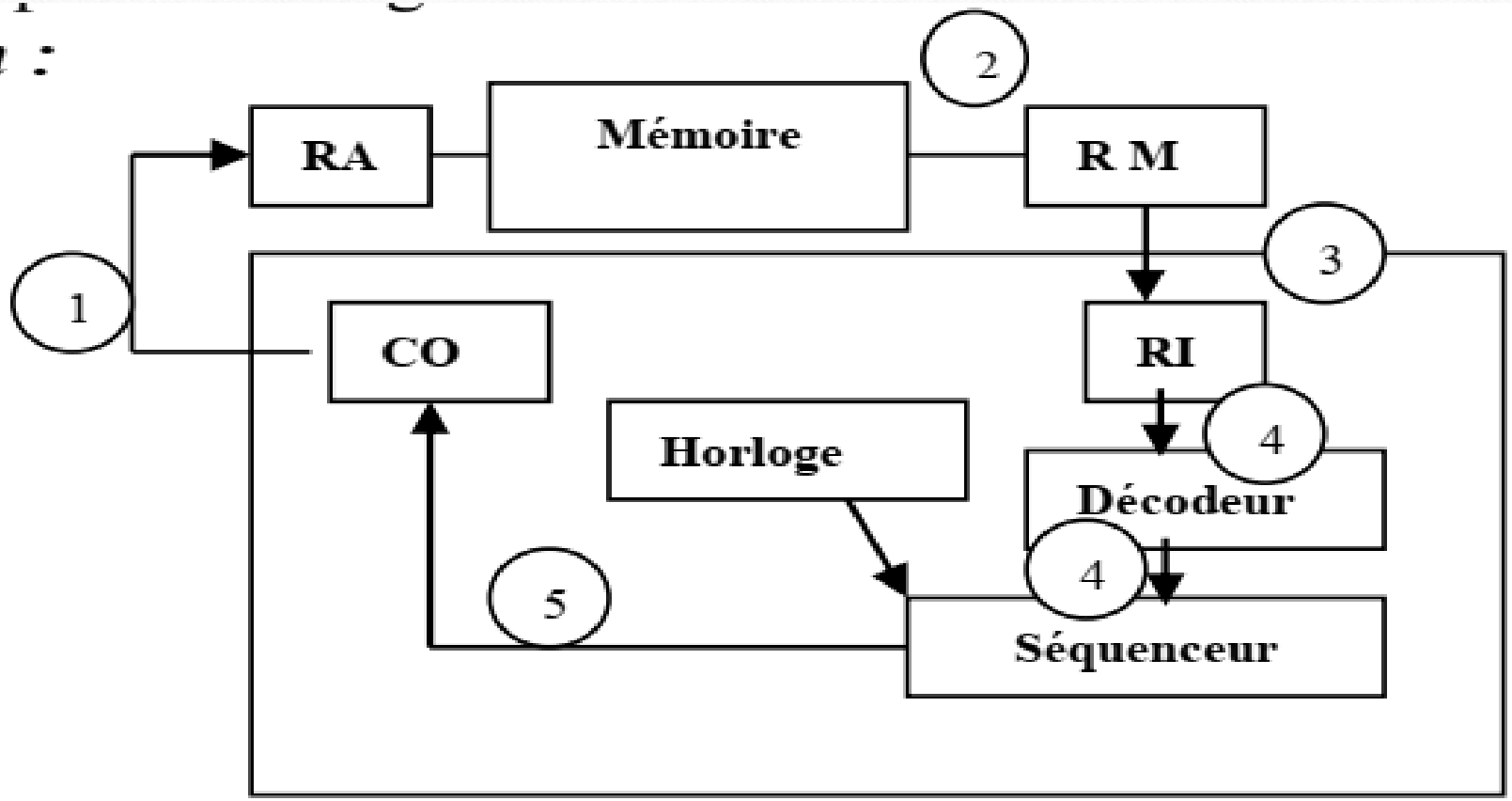
Cycle d’exécution d'une instruction

Phase 2 (Décodage de l’instruction et recherche de l'opérande)

3) Si l'instruction nécessite une donnée en provenance de la mémoire, l'unité de commande récupère sa valeur dans le registre de donnée (RIM) sur le bus «donnée».
4) L’opérande est stockée dans le registre tampon de l’UAL (RTUAL), qui stocke temporellement l’un des deux opérandes d’une instruction arithmétique.
5) Le code opération est transmis au décodeur qui détermine le type d’opération et le transmis en séquenceur

Cycle d’exécution d'une instruction

Fetch :



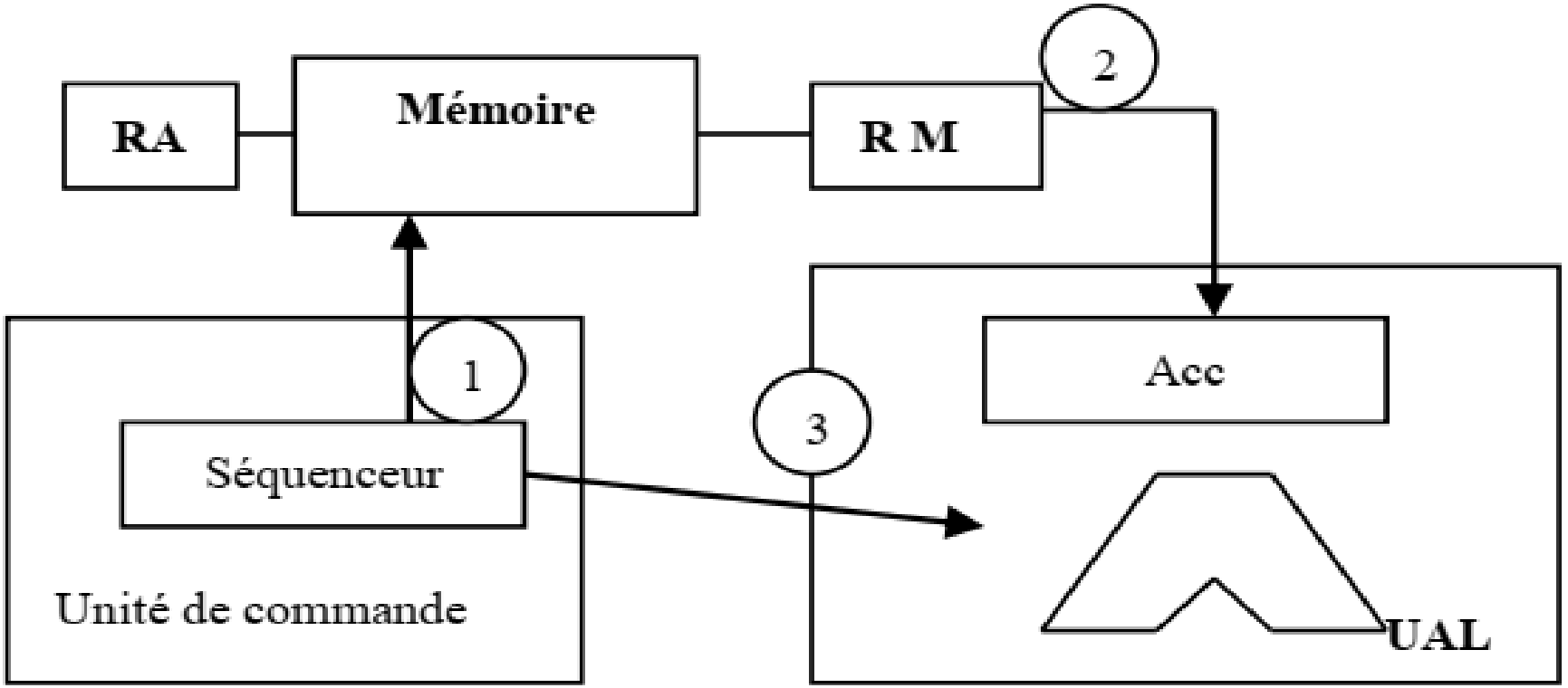
Cycle d’exécution d'une instruction

Phase 3 (Exécution de l'instruction)

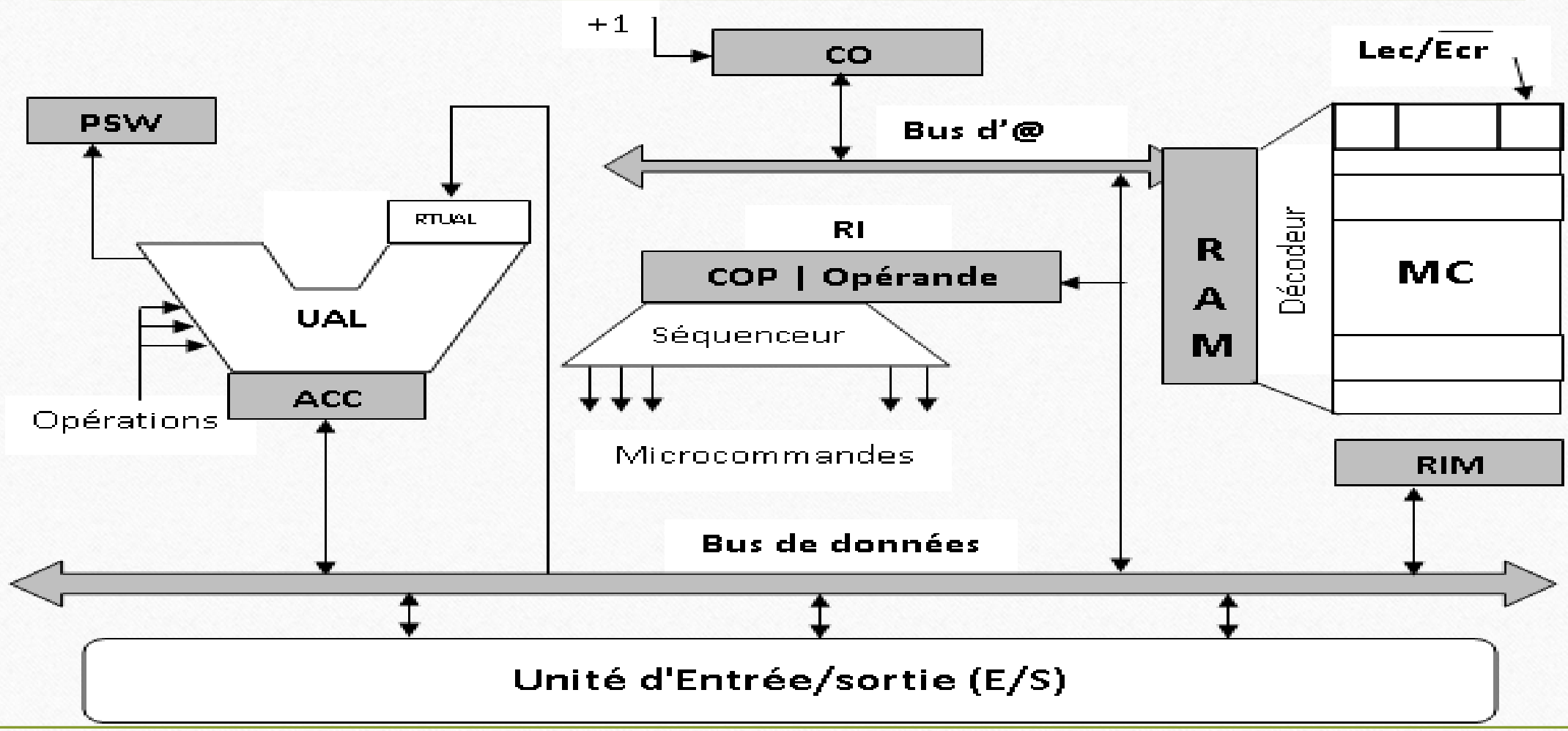
1) Le séquenceur commence à envoyer des signaux de commandes vers la mémoire pour lire l’opérande à l’adresse déjà stockée dans le RIM et transmet le contenu du RIM à l’UAL [ACC, (RTUAL), …]
2) L'UAL exécute l'opération qui lui est demandée en mettant à jour son registre résultat «Accumulateur» et transfère ce résultat dans la mémoire centrale, à l'adresse référencée dans l'instruction, en utilisant le bus « données/résultats »;
3) Par ailleurs, le CO est automatiquement incrémentée.

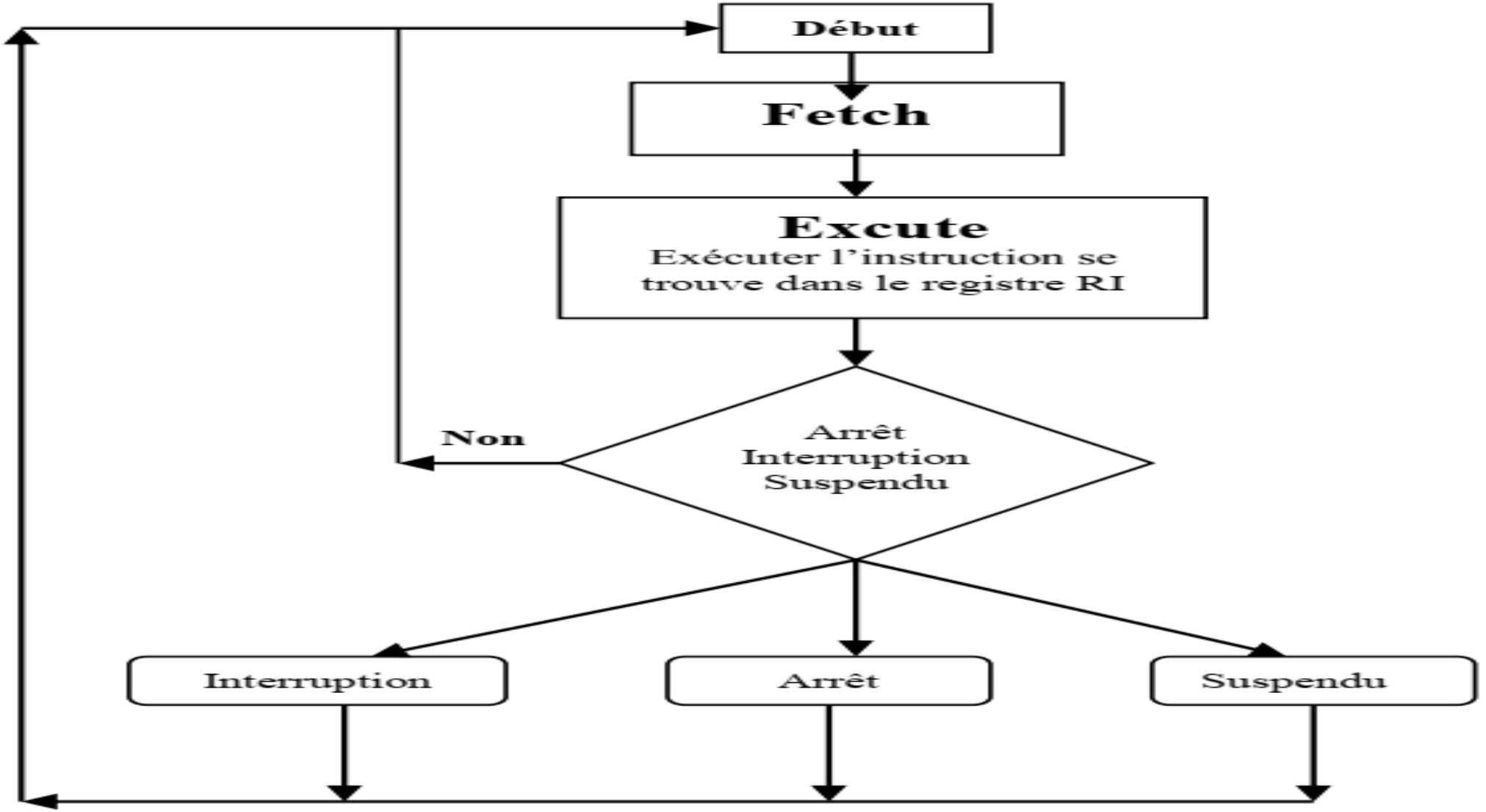
Cycle d’exécution d'une instruction

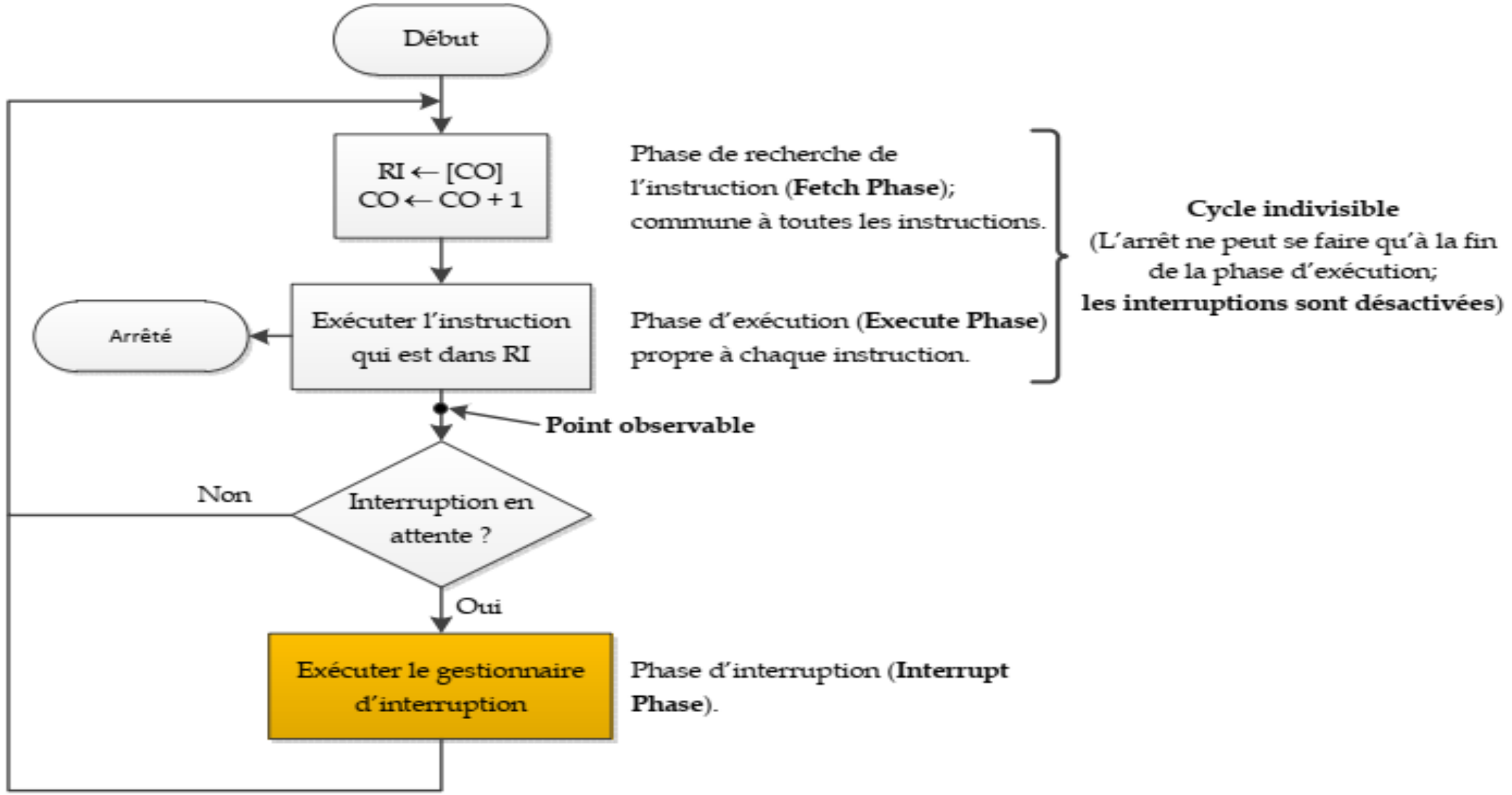
Execute :



Machine de VON-NEUMANN




Début
Fetch
Excute
Exécuter l'instruction se trouve dans le registre RI
Non
Arrêt
Interruption
Suspendu
Interruption
Arrêt
Suspendu


Début
RI ← [CO]
CO ← CO + 1
Phase de recherche de l'instruction (Fetch Phase); commune à toutes les instructions.
Arrêté
Exécuter l'instruction qui est dans RI
Phase d'exécution (Execute Phase) propre à chaque instruction.
Cycle indivisible
(L'arrêt ne peut se faire qu'à la fin de la phase d'exécution; les interruptions sont désactivées)
Point observable
Non
Interruption en attente ?
Oui
Exécuter le gestionnaire d'interruption
Phase d'interruption (Interrupt Phase).

Etat du processeur (Processor State Information)

En général, il est nécessaire d'empêcher le programme d'un utilisateur de perturber le fonctionnement global du système ou les programmes des autres utilisateurs. Cela implique qu'il n'ait pas accès à l'ensemble des ressources de la machine, mais seulement à celles qui lui ont été allouées en propre. La solution la plus couramment adoptée est de distinguer deux modes de fonctionnement du processeur, **le mode maître et le mode esclave (asservi).**

Etat du processeur (Processor State Information)

Mode utilisateur (ou esclave) : réservé aux programmes usagers qui ont des droits d'actions limités. Certaines opérations sont interdites dans ce mode (manipulation directes des E/S, masquage d'IT, accès à une zone système, modification de priorité d'un process, modification directe de l'heure, manipulation de l'horloge, …).

Etat du processeur (Processor State Information)

Mode superviseur (maître, noyau ou privilégié) : réservé au noyau du système d'exploitation, c.à.d. le propriétaire du programme en cours d'exécution est le SE. Il a tous les droits d'actions sur les objets du SIQ (le processeur a accès à toutes les ressources de la machine) (modifier les variables systèmes, faire les E/S, actionner le CPU, l'heure système, …)

Etat du processeur (Processor State Information)

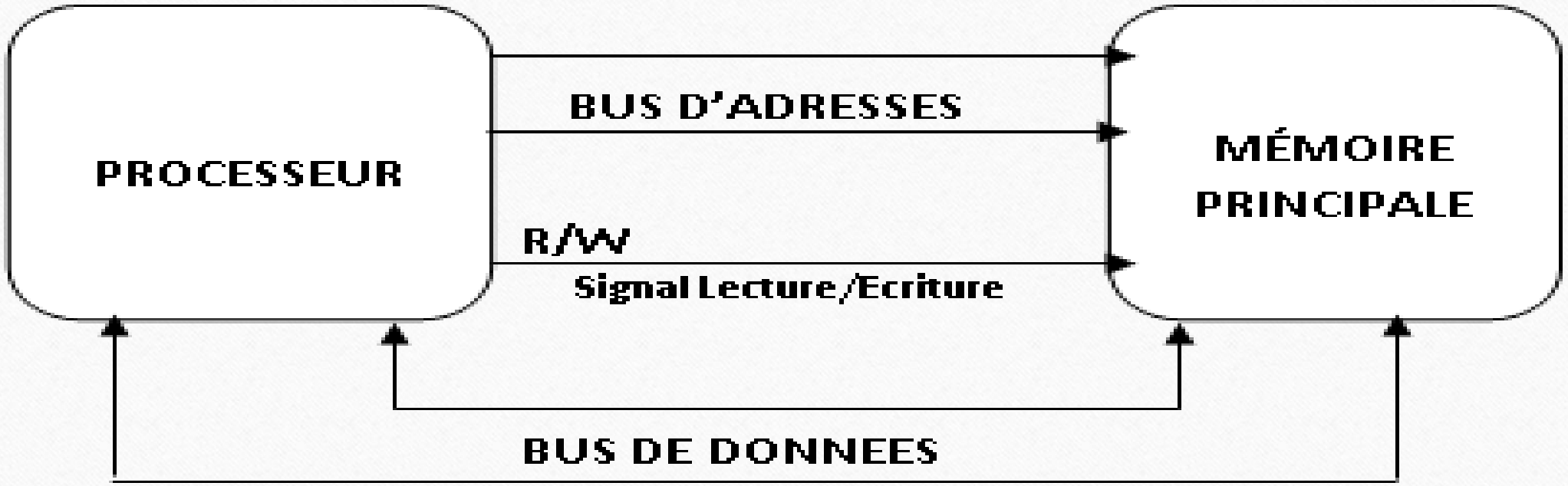
Remarque :

- L'indicateur du mode de fonctionnement maître/esclave fait partie du mot d'état programme PSW.
- Quand un processus est interrompu, l'information contenue dans les registres du processeur doit être sauvegardée afin qu'elle puisse être restaurée quand le processus interrompu reprend son exécution.
- L'état d'un processeur n'est observable qu'entre deux cycles du processeur.

2) Mémoire Centrale (MC)

Une mémoire adressable est un ensemble de supports d'informations de même capacité appelé **Emplacement**. Chaque emplacement est repéré dans la mémoire par une information appelée **adresse** qui permet de le désigner et de le distinguer des autres emplacements. L'interface entre un processeur et une mémoire adressable est schématisé par la figure ci-dessus.

2) Mémoire Centrale (MC)



Cycle mémoire : c'est le temps minimal qui doit séparer deux accès successifs à une mémoire

3) Les Unités périphériques ou d'Entrée/Sortie (E/S)

L'unité d'E/S est un dispositif capable de transférer une quantité d'information entre la mémoire d'un ordinateur (mémoire centrale) et un support d'information externe (périphérique). Le principe élémentaire mis en œuvre pour l'échange de données entre deux constituants physiques est représenté dans la figure suivante

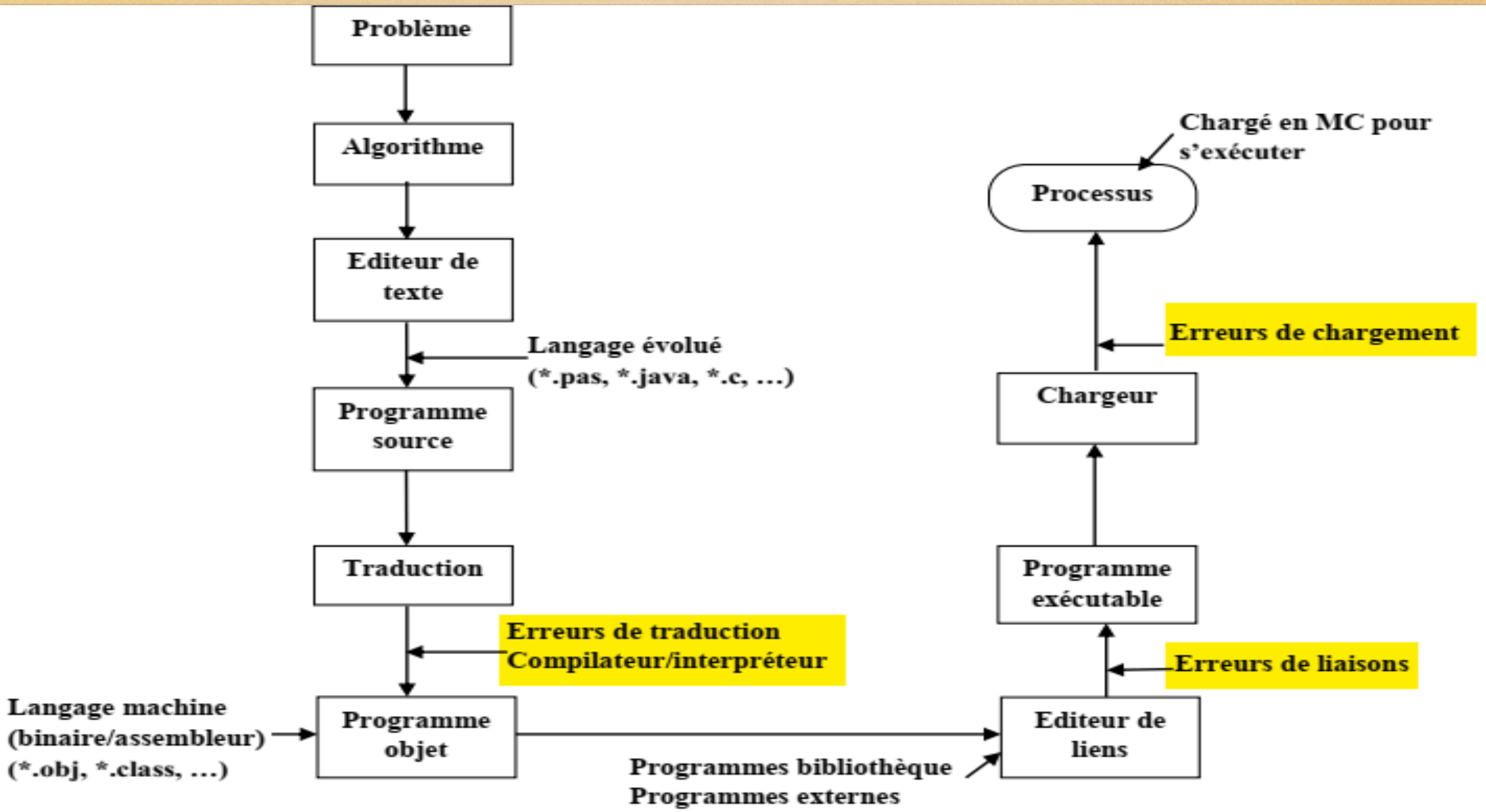
3) Les Unités périphériques ou d'Entrée/Sortie (E/S)

Différents modes d'entrée/sortie peuvent être distingués :

- Entrée/ Sortie Directe (programmée)
- E/S par Accès Direct à la Mémoire (DMA).
- Les entrées-sorties par processeur spécialisé (Canal d'E/S).

IV. Les étapes de cheminement d'un programme

Le développement d'un programme, depuis l'analyse du problème jusqu'à sa mise au point, nécessite de nombreux outils logiciels qui constituent un environnement de programmation. Pour fonctionner, ces outils utilisent les services du système d'exploitation. La chaine de production de programme transforme un programme écrit dans un langage de haut niveau (Pascal, C, VB, Java, etc.) en un programme dit **exécutable**, écrit en langage machine. Cette transformation s'effectue à l'aide de plusieurs étapes (Voir Figure 2.3):


Problème
Algorithme
Editeur de texte
Langage évolué
(*.pas, *.java, *.c, …)
Programme source
Traduction
Erreurs de traduction
Compilateur/interpréteur
Langage machine
(binaire/assembleur)
(*.obj, *.class, …)
Programme objet
Programmes bibliothèque
Programmes externes
Editeur de liens
Erreurs de liaisons
Programme exécutable
Chargeur
Erreurs de chargement
Processus
Chargé en MC pour s'exécuter

1) L'Éditeur de texte ou l'édition de programme

C'est l'étape dont on saisit l'algorithme établit pour résoudre le problème tout en le traduisant à un programme écrit dans un langage évolué (exemple : Java, Pascal, Delphi, C++, …) ou dans le langage Assembleur. L'éditeur de texte est l'outil qui nous permet de réaliser cette étape, c'est un logiciel interactif soit associé au système, soit associé au langage de programmation. Il nous permet d'appliquer toutes les opérations nécessaires sur un fichier nommé « **Code source** ». Ce code source à une extension spécifique tout dépend du langage utilisé (exemple : *.pas, *.java, *.cpp, …)

Exemple : Bloc-notes, Wordpad, Edit de MS-DOS, Editeur du compilateur C, JEDPlus pour java, …

2) La traduction en langage machine

Le traducteur est un outil système qui permet de traduire les instructions écrites dans un langage de programmation vers des instructions écrites dans le langage machine (Binaire), on distingue deux types de traducteurs :

- **Le compilateur** : Est un logiciel de complexité grandissante, permettant de traduire un programme source vers le langage machine (programme objet) en passant par :
- **L'interpréteur :** L'interpréteur est un logiciel qui permet de traduire le programme vers le langage machine et de l'exécuter en même temps.

Le compilateur

- **Le compilateur** : Est un logiciel de complexité grandissante, permettant de traduire un programme source vers le langage machine (programme objet) en passant par :
  - **L'analyse lexicale** :
  - **L'analyse syntaxique :**
  - **L'analyse sémantique** :
  - **L'optimisation de code.**
  - **Transformation en code objet**

Le compilateur

Lors de ces étapes, le compilateur signale des erreurs à corriger qui peuvent apparaître dans chaque étape. Le résultat de cette phase est un fichier nommé **code objet** a une extension spécifique selon le langage de programmation (**Exemple** : *.obj Pascal, *.class Java, …)

L’interpréteur

L’interpréteur : L’interpréteur est un logiciel qui permet de traduire le programme vers le langage machine et de l’exécuter en même temps.

Ce type de traducteur ne résulte pas un fichier contenant le code objet ce qui impose l’interprétation à chaque fois d’exécution (**Exemple** : l’interpréteur HTML).

2) La traduction en langage machine

N.B : le compilateur définit deux types de codes dans le code objet :

- Le code **absolu** qui a une adresse fixe dans la mémoire.
- Le code **translatable** qui peut se mettre dans n'importe quel emplacement dans la mémoire.

L'opération de **translation** consiste à ajouter à **chaque emplacement** qui contient une **adresse**, la valeur de l'adresse effective de l'implantation finale dans la mémoire.

3) L’édition de liens

Lors de la traduction d’un programme en code objet, le compilateur associe à chaque instruction son type, dont on peut avoir :

- Des données variables
- Des données constantes
- Des instructions
- Des appels à des procédures, des modules ou des variables externes.

3) L'édition de liens

Donc il associe à chaque référence soit :

- Un lien interne au programme, mais qui peut être accédé par d'autre modules → **lien utilisable (définition externe).**
- Un lien externe appartient à un autre module appelé dans ce programme → **lien à satisfaire (référence externe).**

3) L'édition de liens

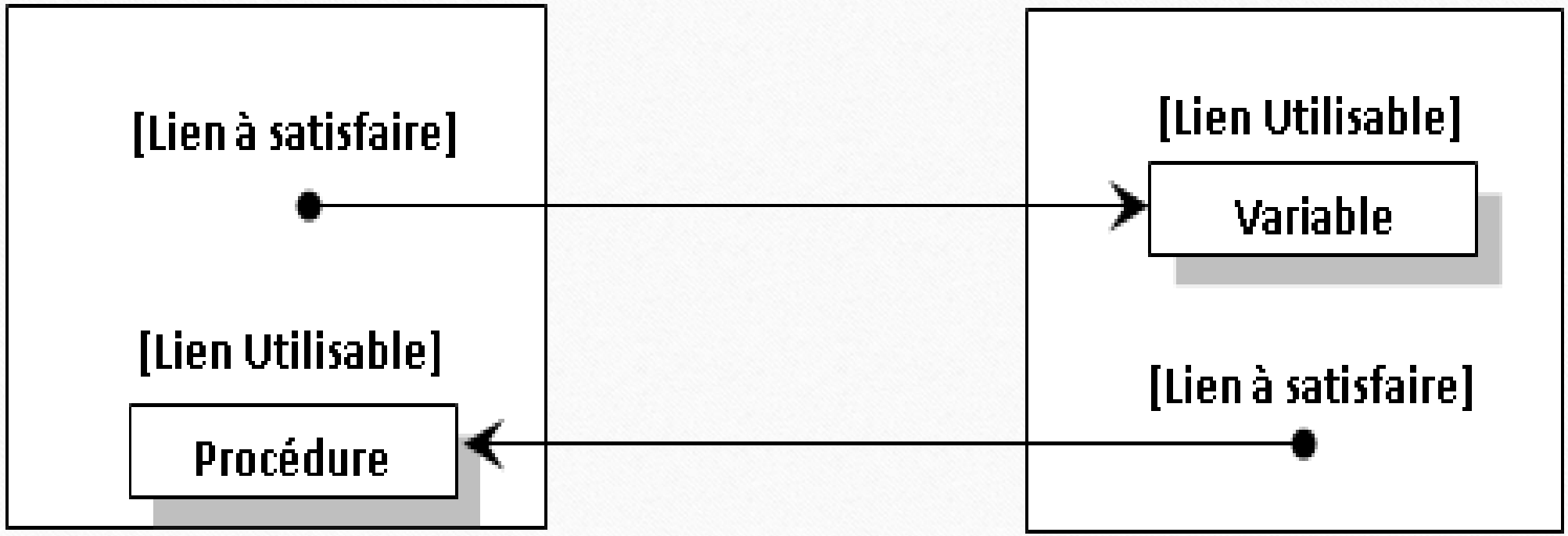
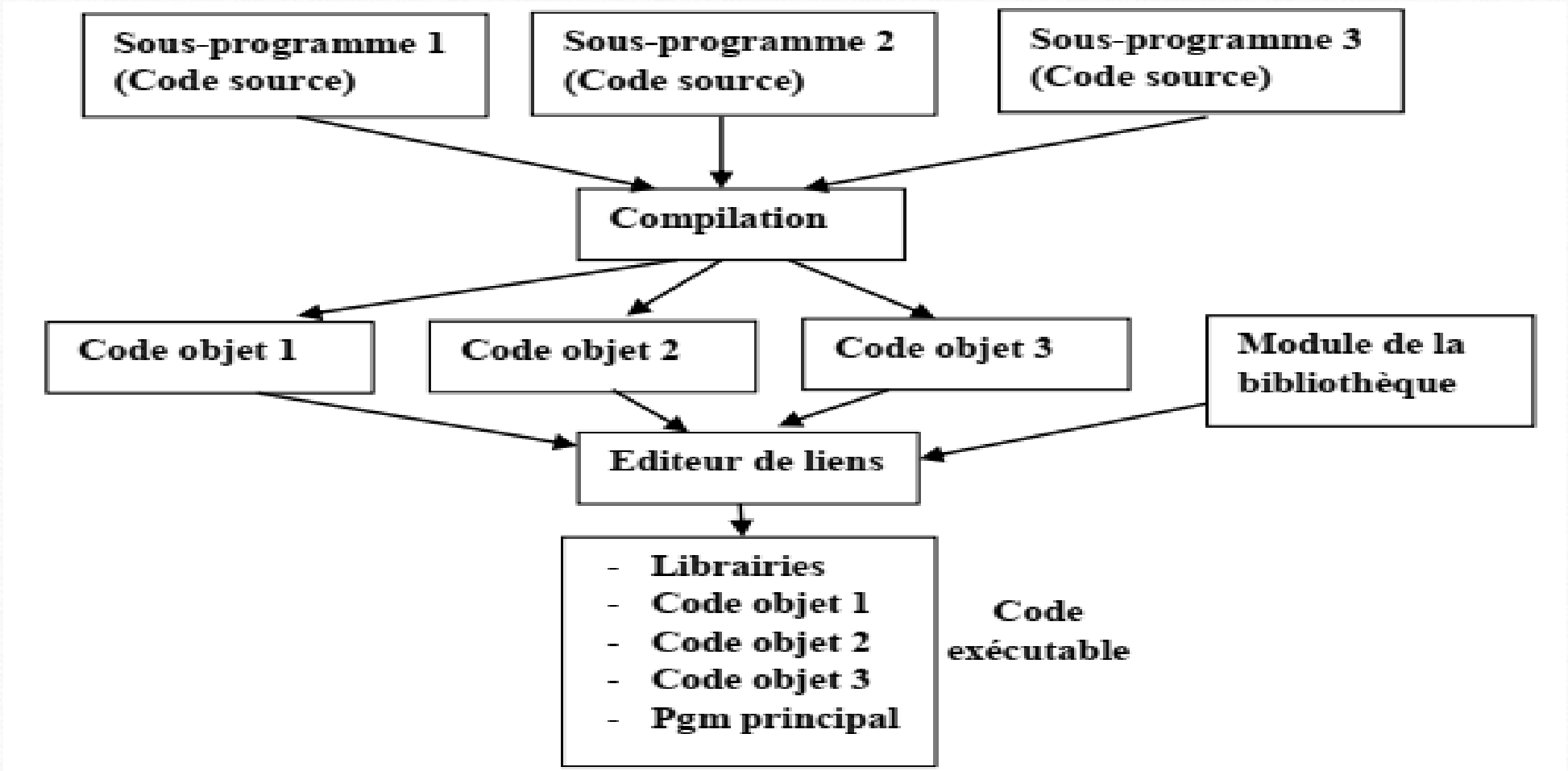


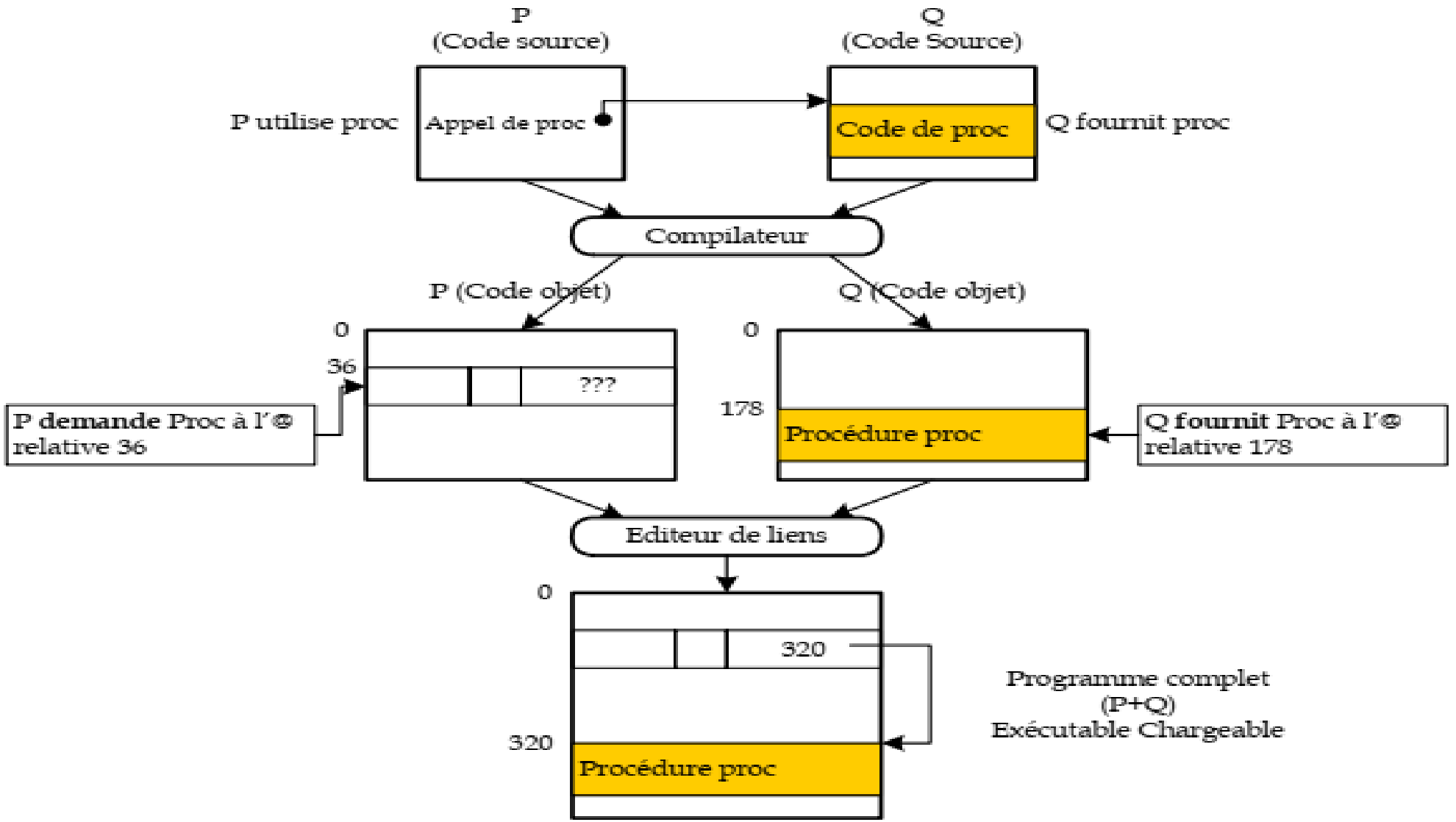
Figure 2.4 : La notion de lien

3) L’édition de liens

L’éditeur de lien est **un outil système** qui a comme objectif **d’accomplir** la tâche du compilateur en ajoutant les codes objets de tous les liens à satisfaire dans le code objet principal. Alors l’éditeur de liens cherche l’origine de chaque référence externe et établit une table globale des modules contenant les informations : **nom du module, taille et adresse d’implantation.**

3) L'édition de liens




P
(Code source)
Q
(Code Source)
P utilise proc
Appel de proc
Code de proc
Q fournit proc
Compilateur
P (Code objet)
Q (Code objet)
0
36
???
0
178
Procédure proc
P demande Proc à l'@ relative 36
Q fournit Proc à l'@ relative 178
Editeur de liens
0
320
320
Procédure proc
Programme complet
(P+Q)
Exécutable Chargeable

3) L'édition de liens

On peut distinguer entre deux types des éditeurs de liens :

- **Editeur de lien statique** : c'est l'édition de liens qui s'établit une fois pour toute et elle engendre un fichier résultat liant toutes les références externes avec le programme principal, nommé le fichier exécutable. Ce type des éditeurs de liens ne se refait pas à chaque exécution.
- **Exemple :** l'éditeur de liens du langage Pascal → des fichiers *.exe.

3) L’édition de liens

On peut distinguer entre deux types des éditeurs de liens :

- **Editeur de liens dynamique** : c’est l’édition de liens qui s’établit à chaque demande d’exécution du programme, elle n’engendre pas un fichier exécutable.
- **Exemple :** l’éditeur de liens du langage Java

3) L’édition de liens

On peut distinguer entre deux types des éditeurs de liens :

- **Editeur de liens dynamique** : c’est l’édition de liens qui s’établit à chaque demande d’exécution du programme, elle n’engendre pas un fichier exécutable.
- **Exemple :** l’éditeur de liens du langage Java

3) L'édition de liens

Exemple :

Prog1.c

```
extern int v1 ;
extern int proc1(int x) ;
float v2 ;
main()
{
int a, x ;
x = v1 + proc1(a) ;
}
```

Prog2.c

```
int v1 ;
extern float v2 ;
int proc1(int x) ;
main()
{
float b ;
b = v2 *2;
}
int proc1(int x)
{
int y ;
return (x * y /2) ;
}
```

3) L'édition de liens

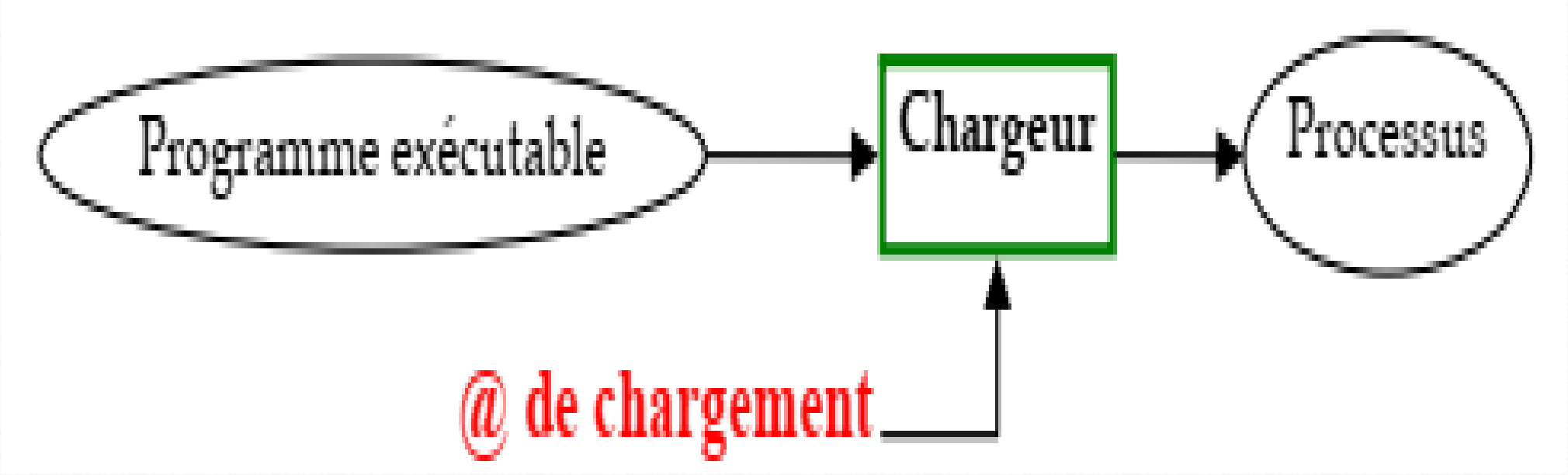
On remarque dans cet exemple que pour le programme **Prog1.c** les liens à satisfaire sont la variable **v1** et la procédure **proc1**, alors que le lien utilisable est la variable **v2**. par contre pour **Prog2.c**, le lien à satisfaire est la variable **v2** alors que les liens utilisables sont la variable **v1** et la procédure **proc1**.

Module	Liens à satisfaire	Liens utilisables
Prog1.c	Variable v1 Procédure Proc1	Variable v2
Prog2.c	Variable v2	Variable v1 Procédure proc1

4) Le chargement

Après l'édition de liens, le programme est prêt à s'exécuter, pour faire, il faut le charger dans la mémoire centrale. Cette phase est faite par un outil système nommé le **chargeur**. Cette étape consiste à lire les instructions en mémoire secondaire, et les transférer en mémoire centrale tout en lisant au début l'adresse de chargement fournit par l'éditeur de liens. Ainsi, le chargeur est un programme qui installe ou charge un exécutable en MC à partir d'une adresse déterminée par le S.E. Dans ce cas le programme devient un **processus**.

4) Le chargement



4) Le chargement

On distingue deux types de chargement :

- **Le chargement absolu** : le code chargé ne doit être pas met dans un autre bloc de données différent de celui indiqué dans le code objet, (d'une autre façon recopier le code).
- **Le chargement relogeable (translatable)** : le chargement peut se réaliser à n'importe quelle adresse dans la mémoire centrale, la façon de translater les adresses est basée sur les informations fournies par l'éditeur de liens.

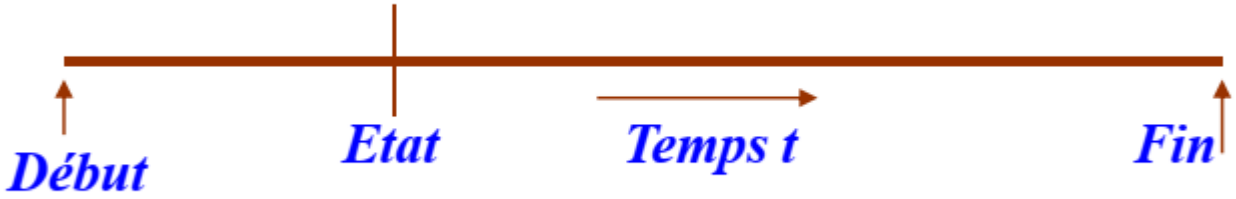
5) Le débogueur

C’est un logiciel qui permet d’exécuter le programme pas à pas, d’afficher les valeurs des variables à tout moment et mettre en place des points d’arrêts sur des conditions ou des lignes du programme. Il offre au programmeur la possibilité de contrôler l’exécution et de détecter l’origine des erreurs non corrigées.

V. Notion de processus

- Qu'est-ce qu'un processus ?
  - Entité **dynamique** représentant l'exécution d'un programme sur un processeur

  ➔ créée à un instant donné, a un état qui évolue au cours du temps et qui disparaît, en général, au bout d'un temps fini



V. Notion de processus

Définition 2.3

Un processus est un programme en cours d'exécution qui est exécuté par un processeur. Aussi, plusieurs processus peuvent-ils être associés à un programme. Chaque processus possède un espace de travail en mémoire, son compteur ordinal et ses registres.

V. Notion de processus

Définition 2.4

On peut définir un processus (process) comme un programme en exécution. Autrement dit, un programme par lui-même n'est pas un processus. Un programme est une entité passive, comme le contenu d'un fichier stocké sur disque, tandis qu'un processus est une entité active, avec un compteur d'instructions spécifiant l'instruction suivante à exécuter et un ensemble de ressources associées.

V. Notion de processus

Définition 2.5

Un processus est une entité dynamique correspondant à l'exécution d'une suite d'instructions : un programme qui s'exécute, ainsi que ses données, sa pile, son compteur ordinal, son pointeur de pile, ses ressources et les autres contenus de registres nécessaires à son exécution.

1) Ce qui se passe après le lancement d'un programme

- Le système crée un **Job** au niveau de la mémoire virtuelle tout en chargeant le programme.
- Après, selon un critère ou autre ce job est admis dans le système tout en le chargeant vers la mémoire centrale dans sa totalité ou une partie, ce chargement crée ce qu'on appelle un **processus**.
- Le processus est dirigé, après, vers l'exécution dans le processeur selon des conditions spécifiques, où ce processus peut changer son état, durant cette exécution, plusieurs fois jusqu'à sa terminaison.

2) Caractéristiques du processus

Un processus est caractérisé par

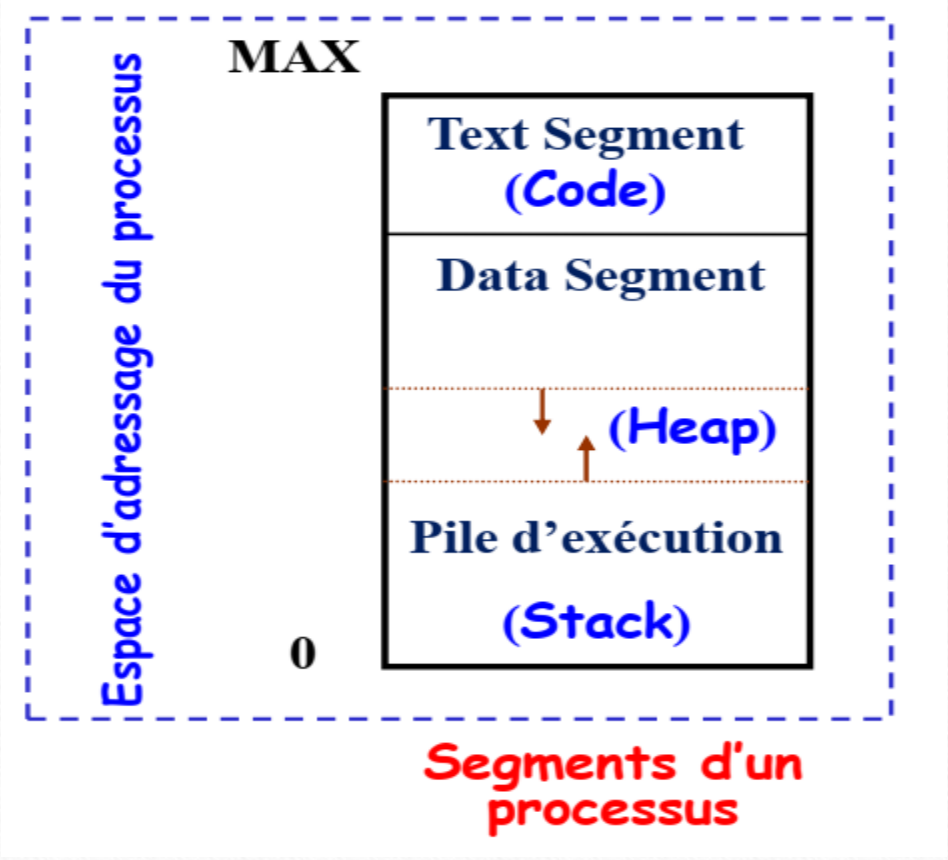
1. Un processus possède un identifiant unique (PID : Process Identifier)
2. Les instructions à exécuter sont stockées dans une pile de données
3. Un espace d'adressage (code, données, piles d'exécution);
4. Un état principal (prêt, en cours d'exécution (élu), bloqué, …);
5. Les valeurs des registres lors de la dernière suspension (CO, sommet de Pile...);
6. Une priorité;
7. Les ressources allouées (fichiers ouverts, mémoires, périphériques …);
8. Les signaux à capter, à masquer, à ignorer, en attente et les actions associées;
9. Autres informations

3) Contexte d’exécution d’un processus

- Le contexte d'un processus est l'ensemble des données qui permettent de reprendre l'exécution d'un processus qui a été interrompu.
- Quand il y a changement de processus courant, il y a réalisation d'une **commutation de mot d'état** et d'un **changement de contexte.** Le noyau s'exécute alors dans le nouveau contexte.

4) Image mémoire d'un processus

- Image mémoire d'un processus



5) Descripteur de Processus (PCB)

Identificateur processus

Etat du processus

Compteur ordinal

Contexte pour reprise (Registres, Pointeurs piles, …)

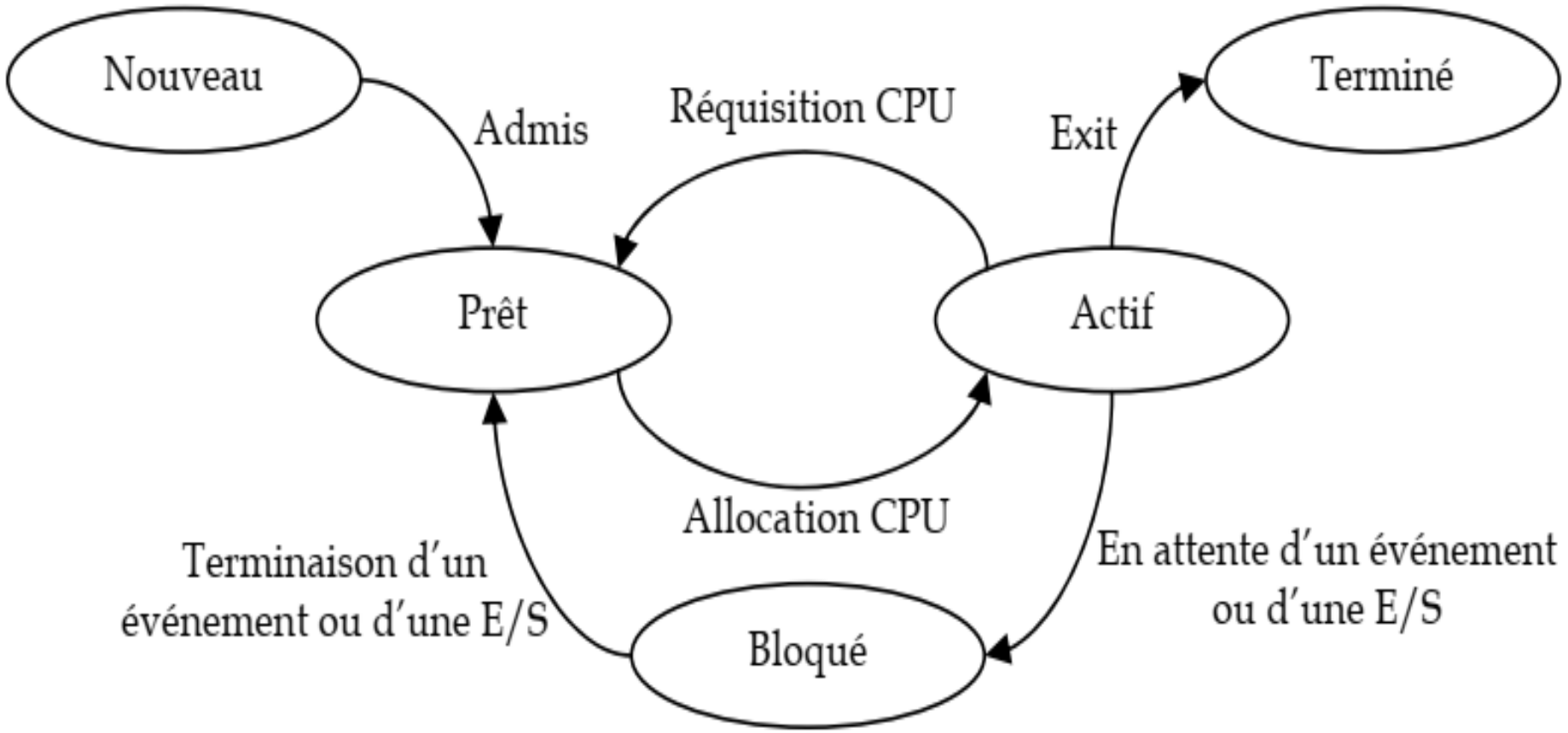
Pointeurs sur file d'attente et priorité (Ordonnancement)

Information mémoire (limites et tables pages/segments)

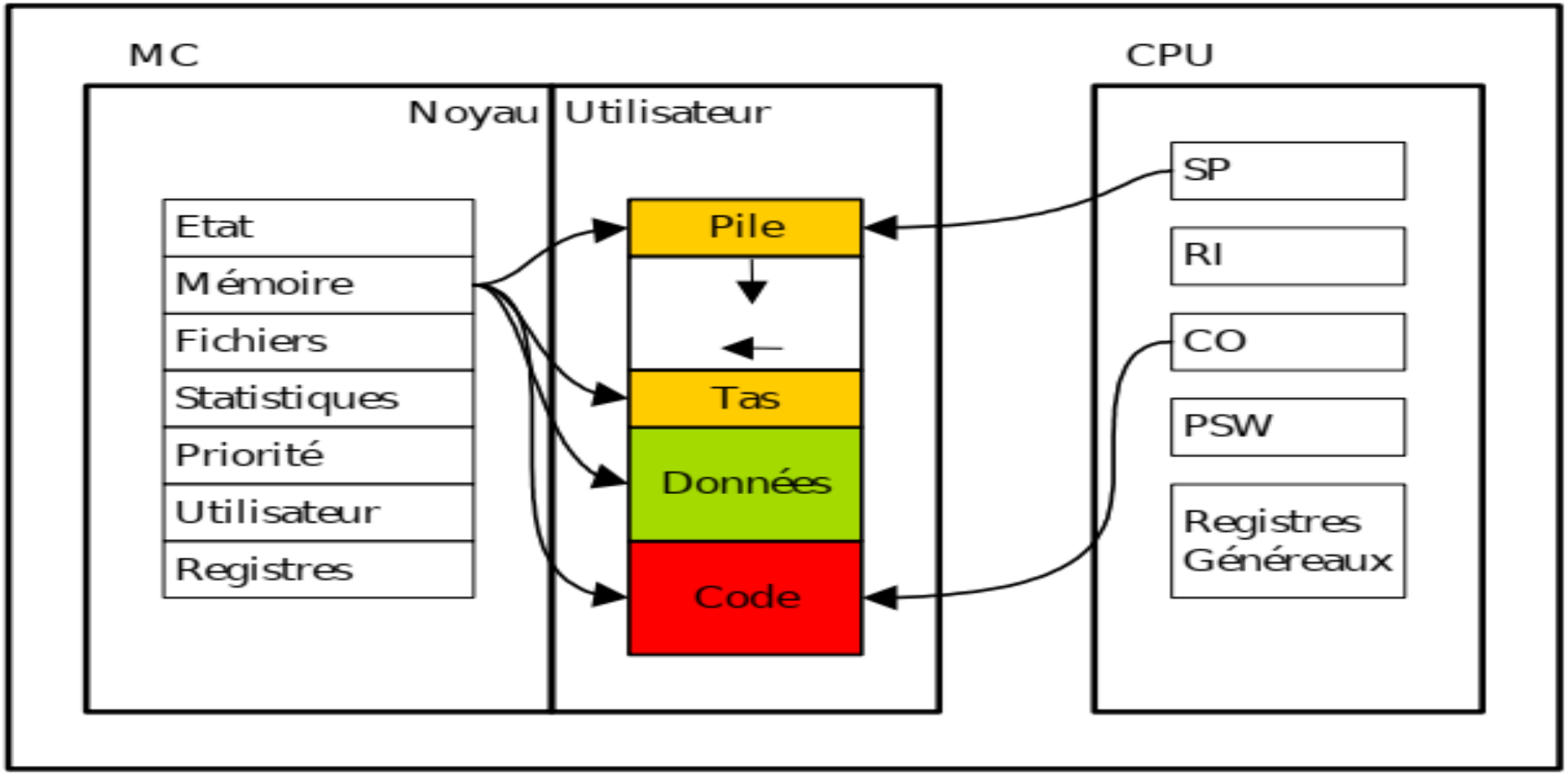
Informations des comptabilisation ets sur les E/S, périphériques alloues, fichiers ouverts, ...

Pointeurs des PCBs suivant et précèdent

6) Etats de processus



II. Notion de processus



7) Les primitives de manipulation de processus

La manipulation d'un processus se fait par des primitives nécessaires qui utilisent son PCB et se sont des procédures systèmes. Ces primitives d'exécutent d'une manière indivisible :

- La création d'un processus.
- Activation d'un processus prêt.
- Suspension d'un processus.
- Destruction d'un processus

8) Mécanisme de commutation de contexte

Dans un système multiprogrammé, le processeur assure l'exécution de plusieurs processus en parallèle (pseudo-parallélisme). Le passage dans l'exécution d'un processus à un autre nécessite une opération de **sauvegarde du contexte du processus arrêté**, et le chargement de celui du nouveau processus. Ceci s'appelle la **commutation du contexte** (**Context Switch**).

8) Mécanisme de commutation de contexte

- La commutation de contexte est le mécanisme qui permet au système d'exploitation **de remplacer le processus élu par un autre processus éligible.** Une commutation de contexte se produit quand l'exécution d'un processus s'interrompt, ou reprend.
- La commutation de contexte consiste à changer les contenus des registres du processeur central par les informations de contexte du nouveau processus à exécuter.

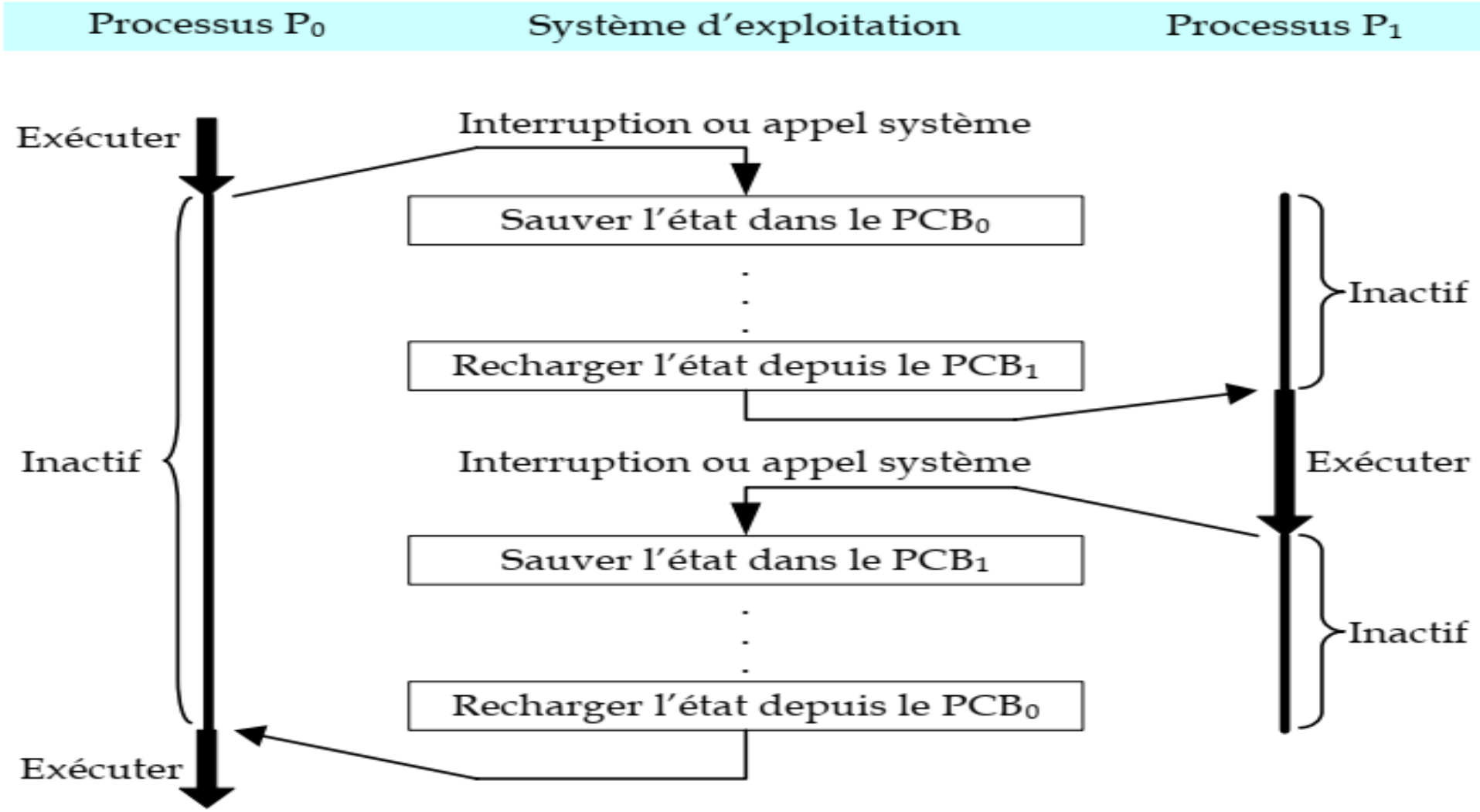


Figure 2.9 : Le Mécanisme de commutation de contexte

8) Mécanisme de commutation de contexte

La commutation du contexte se fait en deux phases :

- La **première phase** consiste à commuter le petit contexte (CO, PSW) par une instruction **indivisible**.
- La **deuxième phase** consiste quant à elle à commuter le grand contexte par celui du nouveau processus.

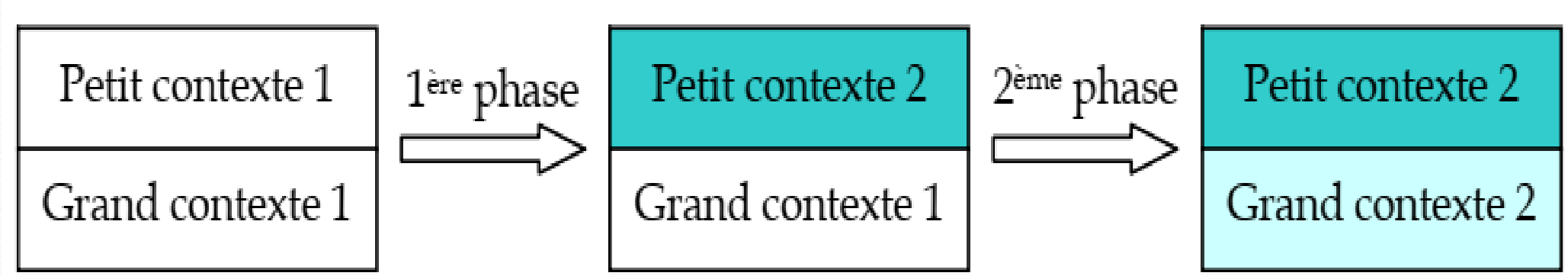


Figure 2.10 : Les phases d'une commutation de contexte.

8) Mécanisme de commutation de contexte

Remarque :

La commutation du contexte est déclenchée suivant l'état d'un indicateur qui est consulté par le processeur à chaque point observable.

VI. Les Systèmes d'Interruption

Dans un ordinateur, deux types de programmes coexistent :

- Les **programmes usagers** qui font un calcul utile.
- Les **programmes du S.E** qui font un travail de **superviseur** de tous les événements qui arrivent à la machine.

VI. Les Systèmes d'Interruption

Lorsqu'un programme est en cours d'exécution, plusieurs événements peuvent arriver

- **Les événements synchrones** qui sont liés à l'exécution du programme en cours, comme :
  - ✓ Division par zéro.
  - ✓ Exécution d'une instruction inexistante ou interdite.
  - ✓ Tentative d'accès à une zone protégée.
  - ✓ Appel à une fonction du S.E.
- **Les événements asynchrones** qui ne sont pas liés à l'exécution du programme en cours :
  - ✓ Fin d'opération d'E/S.
  - ✓ Signal d'horloge.

1) Problématique

Question : Par quel mécanisme peut-on réduire le temps de supervision du S.E. ?

1) Problématique

Question : Par quel mécanisme peut-on réduire le temps de supervision du S.E. ?

Solution : Au lieu que ça soit le processeur qui contrôle continuellement l'état d'une ressource, c'est plutôt à la ressource d'informer le processeur central sur son état au moment significatif. C'est **le principe des interruptions** de programmes.

VI. Les Systèmes d'Interruption

Définition 2.6

Une **interruption** est un signal déclenché par un événement interne à la machine ou externe, provoquant l'arrêt d'un programme en cours d'exécution à la fin de l'opération courante, au profit d'un programme plus prioritaire appelé programme d'interruption. Ensuite, le programme interrompu reprend son exécution à l'endroit où il avait été interrompu ou un autre programme.

VI. Les Systèmes d'Interruption

Définition 2.6

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 (interruptible) 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** ».

VI. Les Systèmes d'Interruption

Définition 2.8

Point de vue matérielle : Une interruption est un signal physique (électronique) émis par un périphérique au CPU pour lui signaler l'occurrence d'un évènement.

Point de vue logicielle : Une interruption est un évènement extérieur provoquant la suspension temporaire (la préemption) du process actif (lui retirer le CPU) au profit d'un programme spécial appelé programme (**routine** ou **handler**) d'interruption chargé de l'évènement arrivé. Ou un appelle à un service de système d'exploitation (**appel système**)

2) Causes d'interruptions

On distingue plusieurs types d'événements pouvant provoquer des interruptions, ils peuvent être regroupés en deux principales causes :

- **les interruptions externes ou matérielles**
- **Les interruptions internes ou logicielles**

2) Causes d'interruptions

Les interruptions externes ou matérielles

Elles sont provoquées par les périphériques connectés à la machine suite à une fin d'E/S ou à une anomalie ou pour signaler d'autres événements.

A titre d'exemples : Branchement ou retrait d'un Flash disque au port USB, le signal émis par le clavier suite à une frappe d'une touche, signal émis par l'horloge suite à un tick ou encore le signal généré par un capteur de température dans une installation industrielle.

2) Causes d'interruptions

Les interruptions internes ou logicielles

La cause principale de ce type d'IT est le process actif (le programme en cours d'exécution). On en distingue deux principales causes :

- **les déroutements**
- **les Appels au superviseur (SVC)**

2) Causes d'interruptions

les déroutements : c'est une exception levée suite à une anomalie (une erreur) générée par l'exécution de l'instruction courante. Il peut être provoqué par :

- une erreur arithmétique : division par zéro, débordement, …
- référence d'une adresse mémoire hors inexistante (limites de la mémoire disponible).
- une violation de la protection mémoire.
- une instruction illégale
- tentative d'exécution d'une instruction privilégiée en mode esclave.
- les défauts de page, débordement de pile, …

A la suite d'un déroutement, l'utilisateur sera averti de l'erreur commise et le process actif est arrêté immédiatement.

2) Causes d'interruptions

les Appels au superviseur (SVC)

Un appel au superviseur (SVC : SuperVisor Call) est une instruction qui a pour fonction de provoquer le changement d'état du processeur. Un appel au superviseur provoque l'exécution d'un programme spécifié avec un changement du mot d'état du processeur et sauvegarde de l'état au moment de l'appel. Cette instruction est principalement utilisée pour réaliser l'appel, à partir d'un programme utilisateur exécuté en mode esclave (asservi), par exemple des fonctions d'accès aux fichiers (ouvrir, lire, fermer…etc.). ces appels offrent des services de système d'exploitation.

2) Causes d'interruptions

Un appel au superviseur (SVC) est une instruction au moyen de laquelle un programme qui s'exécute en mode esclave peut solliciter du système une fonction ou un service auquel il n'a pas le droit d'accéder directement.

2) Causes d'interruptions

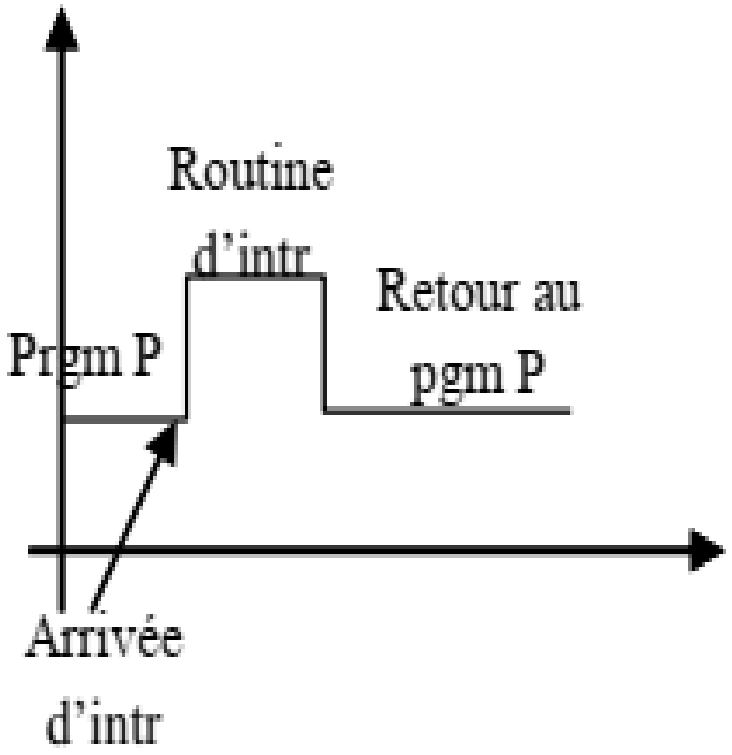
Remarque 2.1

Pour certains auteurs, un appel SVC est considéré comme étant un déroutement. La différence principale est qu'un **déroutement** exprime **une erreur non volontaire** alors qu'un **SVC** est une **instruction voulue en elle-même par l'utilisateur**.

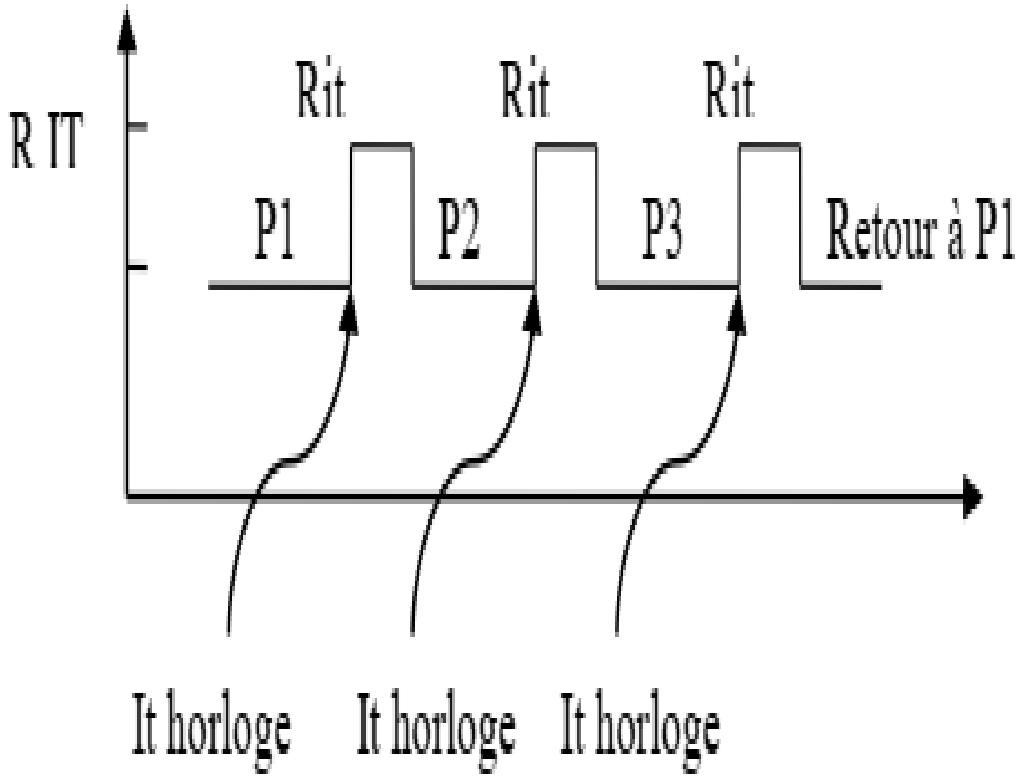
La distinction entre interruption, déroutement, appel au superviseur est **fondée sur la fonction** et **non sur le mécanisme** qui est commun.

2) Causes d'interruptions (Exemple)

1) Fin d'E/S



2) Signal d'horloge



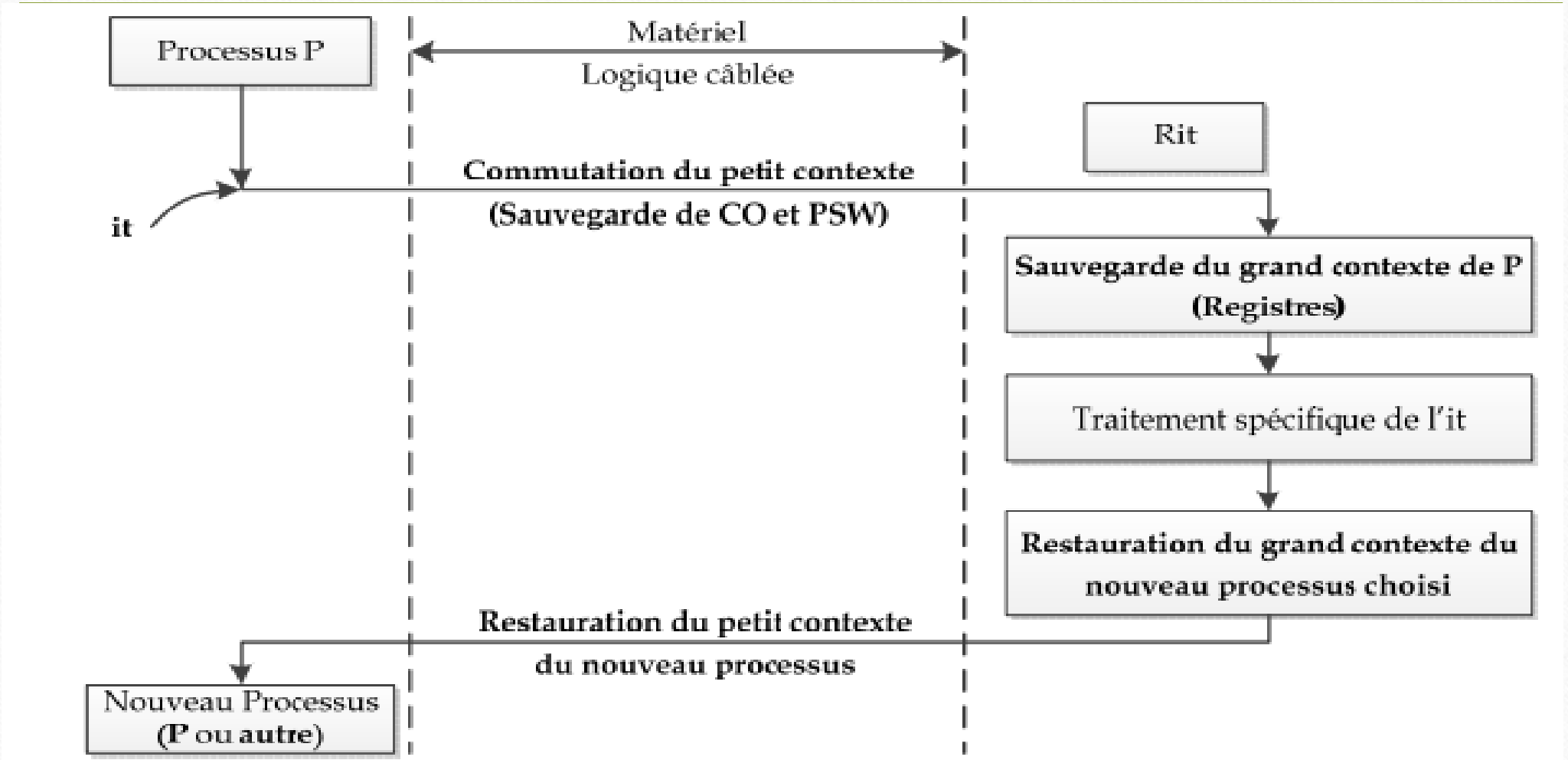
Principe de temps partagé

3) Mécanismes de gestion des interruptions

a) Le traitement d'une interruption

1) Suspendre le process **P** en cours et sauvegarder son contexte dans une pile :
2) Identifier la cause d'interruption : pour détecter la source émettrice du signal d'IT.
3) Chargement du contexte du programme d'interruption (contexte actif) et passage en mode système (ou superviseur)
4) Exécuter le programme de traitement de cette **IT**
5) Restaurer le contexte du programme **P** suspendu et en repassant en mode utilisateur ou le choix d'un autre, et continuer son exécution.

3) Mécanismes de gestion des interruptions



3) Mécanismes de gestion des interruptions

b) Conditions d'arrivée d'une interruption

Une interruption ne peut arriver au processeur que dans les conditions suivantes :

1) Le système d'interruption est **actif**.
2) L'UC est à un point observable (**interruptible**).
3) L'interruption est **armée**.
4) L'interruption est **démasquée**.
5) L'interruption est plus prioritaire que le programme en cours.

3) Mécanismes de gestion des interruptions

c) Les priorités des interruptions

Une fois que le signal est détecté dans **un point observable**, il faut déterminer la cause de l'interruption. Pour cela on utilise un indicateur, pour les différentes causes, on parle alors du **vecteur d'interruptions**.

3) Mécanismes de gestion des interruptions

c) Les priorités des interruptions

Pour cela on peut envisager diverses méthodes :

1) si l'indicateur d'interruption est unique, le code de l'interruption est stocké quelque part dans la mémoire et doit être lu par le programme de traitement.
2) s'il existe des indicateurs multiples, chacun est appelé niveau d'interruption. On attache un programme différent de traitement à chacun de ces niveaux.
3) on peut utiliser simultanément ces deux méthodes. Chaque niveau d'interruption est accompagné d'un code qui est lu par le programme de traitement de ce niveau.

3) Mécanismes de gestion des interruptions

c) Les priorités des interruptions

A chaque interruption, est associée une priorité (système d'interruptions hiérarchisées) qui permet de regrouper les interruptions en classes. Chaque classe est caractérisée par un degré d'urgence d'exécution de son programme d'interruption.

Règle : Une interruption de priorité **j** est **plus prioritaire** qu'une interruption de niveau **i** si **j > i**.

3) Mécanismes de gestion des interruptions

c) Les priorités des interruptions

On peut schématiser les différents niveaux d'interruptions comme dans le schéma suivant. La priorité va en décroissant du haut vers le bas.

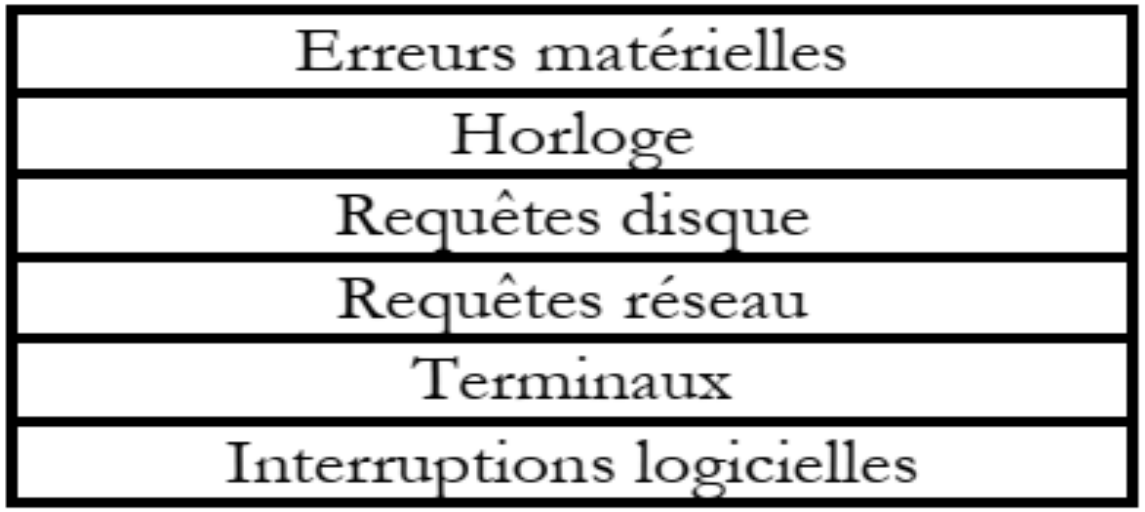


Figure 2.10 : les différents niveaux d'interruptions

3) Mécanismes de gestion des interruptions

c) Les priorités des interruptions

L'intérêt de ce système est la solution de problèmes tels que :

- arrivée de plusieurs signaux d'interruption pendant l'exécution d'une instruction,
- arrivée d'un signal d'interruption pendant l'exécution du signal de traitement d'une interruption précédente.
- On peut utiliser un contrôleur d'interruptions pour regrouper les fils d'interruptions, gérer les priorités entre les interruptions et donner les éléments de calcul d'adresse au processeur.

3) Mécanismes de gestion des interruptions

d) Hiérarchie des interruptions

Les interruptions peuvent être **hiérarchisées**, c'est-à-dire classées par ordre de priorités respectives. Un traitant d'interruption peut donc être lui-même interrompu par une demande d'interruption de niveau de priorité supérieure. Il passe alors à l'état d'attente. La figure suivante représente l'activité des programmes dans le temps pour un système hiérarchisé à 8 niveaux où le niveau 0 est le plus prioritaire, le niveau 7 correspondant au programme d'arrière-plan (Figure 11).

3) Mécanismes de gestion des interruptions

d) Hiérarchie des interruptions

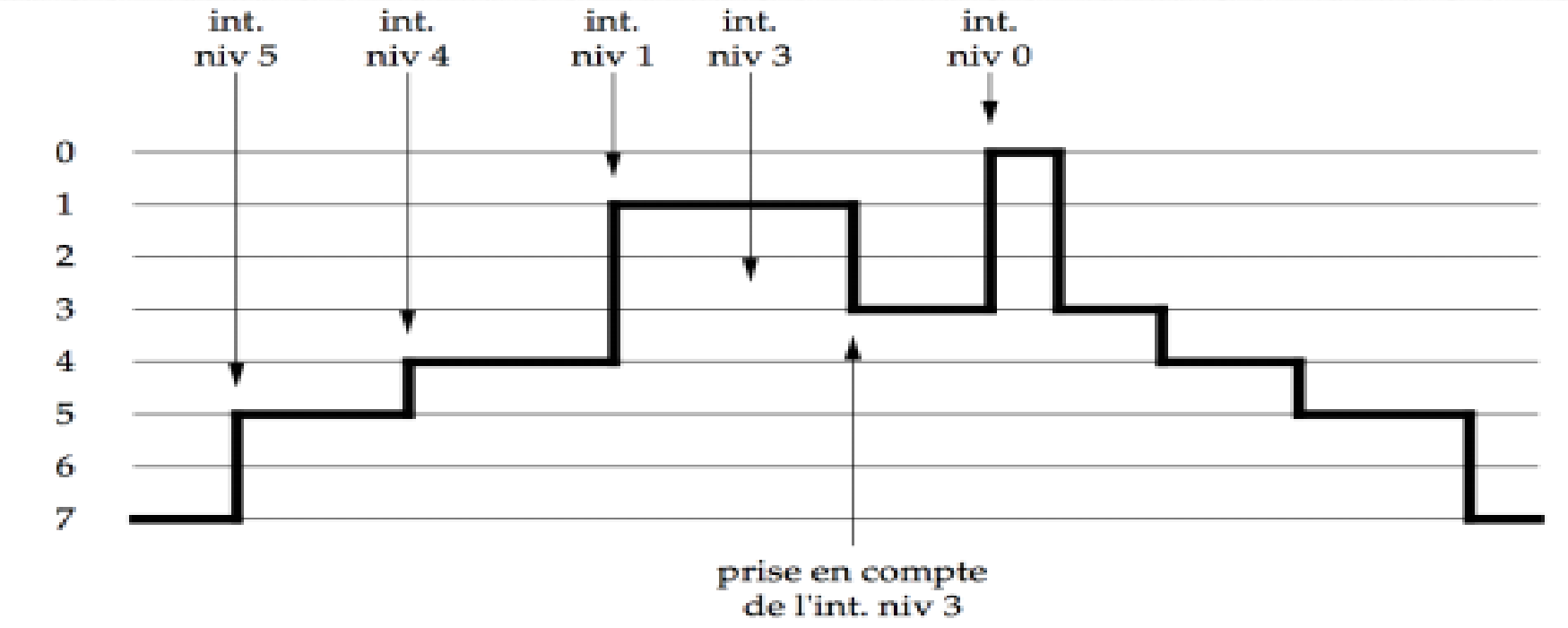


Figure 2.11 : Hiérarchie des interruptions.

3) Mécanismes de gestion des interruptions

e) Les opérations sur les interruptions

- Masquage des interruptions : Certaines interruptions présentent tellement d'importance qu'il ne doit pas être possible d'interrompre leur traitement. On masquera alors les autres interruptions pour empêcher leur prise en compte. Certaines interruptions sont non-masquables : on les prend obligatoirement en compte. **Une interruption masquée n'est pas ignorée : elle est prise en compte dès qu'elle est démasquée**.
- Désarmer une interruption :elle sera ignorée. Par défaut, les interruptions sont évidemment armées.

3) Mécanismes de gestion des interruptions

Exemple

Il est possible à un processus superviseur de masquer une interruption. Elle peut être démasquée après. À un instant donné, les interruptions sont soit masquées soit autorisées, suivant l'état d'un indicateur spécial du registre d'état, IF (Interrupt Flag).

- Si IF = 1 (interruptions autorisées), alors le processeur accepte les demandes d'interruptions masquables, c'est-à-dire qu'il les traite immédiatement ;
- si IF = 0 (interruptions masquées), alors le processeur ignore ces interruptions. L'état de l'indicateur IF peut être modifié à l'aide de deux instructions, CLI (CLear IF pour la mise à 0 de IF), et STI (SeT IF, pour la mise à 1 de IF).

3) Mécanismes de gestion des interruptions

f) La table des vecteurs d'interruption

Le vecteur d'interruption est une table qui contient les adresses des routines d'interruptions. Les interruptions sont numérotées et ces numéros (allant de 0 à 255 (FFh) dans un PC) servent d'index pour rechercher dans le vecteur d'interruption l'adresse de la routine à exécuter.

3) Mécanismes de gestion des interruptions

f) La table des vecteurs d'interruption

L'adresse de début d'une routine d'interruption est appelée **Vecteur d'interruption**. Toutes les adresses (logique CS:IP) de début de ces vecteurs sont rangées dans une table et constituent donc la Table des Vecteurs d'Interruption. Cette table se charge en **RAM** à partir de l'adresse la plus basse (0000h). Chaque vecteur d'interruption est stocké sur **4 octets** (2 pour CS et 2 pour IP),

Numéro de vecteur	Usage
0	Division par zéro
1	Exception de débogage
2	Non-Maskable Interrupts (NMI)
3	Point d'arrêt
4	Dépassement de capacité de registre interne
5	Dépassement de limite
6	Code d'opération non valide
7	Périphérique indisponible
8	Erreur sur nombre double précision
9	réservé : gestion coprocesseur
10	segment d'état de tâche invalide
11	segment absent
12	Erreur dans la pile
13	Erreur de protection
14	Défaut de page
15	réservé
16	Erreur dans nombre en virgule flottante
17	Contrôle alignement
18	Contrôle hardware
19 - 31	réservé
32 - 255	Interruptions masquables

Questions ?

Exercice 2.1 (Questions de Compréhension)

1. Nommez quatre éléments contenus dans le Process Control Block (PCB)?

   Parmi ses attributs, on cite : PID, pointeurs sur des processus parents (PPID) ou enfants, Etat, program counter, registres, pointeurs de mémoire, info pour l'ordonnancement des processus (priorité), pointeurs sur des ressources utilisés, statistiques sur le PCB, Compteur Ordinal, Pointeurs, Temps d'exécution passé…

Exercice 2.1 (Questions de Compréhension)

2. Qu'est-ce qu'une interruption? Quand cela se produit-il?

   Une interruption est un évènement généralement imprévu qui interrompt la séquence normale d'exécution des instructions par le microprocesseur. Les interruptions se produisent quand un périphérique signale un évènement, quand une erreur matérielle ou logicielle survient ou quand le programmeur appelle une interruption.

Exercice 2.1 (Questions de Compréhension)

3. Quelles sont les principales tâches effectuées dans l'interruption du système d'exploitation?
    1) Sauvegarder l'information nécessaire à la reprise ultérieure du processus en cours d'exécution.
    2) Gérer le temps et les processus en attente.
    3) Déterminer quel sera le prochain processus à exécuter et gérer l'état des processus.
    4) Recharger les informations nécessaires à l'exécution du prochain processus à exécuter, puis l'exécuter.

Exercice 2.1 (Questions de Compréhension)

4. D'un point de vue logiciel, que fait généralement un microprocesseur lorsqu'il détecte un signal d'interruption provenant d'un périphérique?
   1) Le microprocesseur termine l'instruction (ou les instructions) en cours
   2) Le microprocesseur vérifie si l'interruption peut être traitée en fonction de sa priorité et si elle est permise.
   3) Le microprocesseur sauvegarde l'adresse de retour et toute l'information nécessaire à la reprise normale de l'exécution après le traitement de l'interruption.
   4) Le microprocesseur détermine l'adresse de la routine à exécuter afin de traiter l'interruption.
   5) Le microprocesseur fait un saut vers la routine qui traite l'interruption

Exercice 2.1 (Questions de Compréhension)

5. Qu'est-ce qu'un "déroutement" ? Donnez en un exemple.

   Le déroutement est un type d'interruption interne où le contrôle passe au système d'exploitation pour traiter l'interruption.

   Exemple : Un défaut de page (tentative d'accès à une page n'existant pas en mémoire)

Exercice 2.1 (Questions de Compréhension)

6. Qu'est-ce qu'une interruption masquée ?

Une interruption masquée est une interruption dont l'effet est temporairement retardé parce qu’elle est désactivée.

Exercice 2.1 (Questions de Compréhension)

7. A quoi sert la table des vecteurs d'interruption? Cette table est-elle en mémoire vive, dans le microprocesseur, dans une mémoire non-volatile ou sur le disque dur?

   La table des vecteurs d'interruption associe les interruptions à la routine qu'il faudra exécuter afin de les traiter. Située dans la ROM initialement (peut être déplacée en mémoire RAM), elle a pour index le numéro de l'interruption et pour contenu l'adresse absolue de la routine qui traitera l'interruption.

Exercice 2.1 (Questions de Compréhension)

8. Les processus peuvent être dans un de trois états : Actif, Prêt, ou Bloqué. Dans quel état est le processus pour chacun des deux cas suivants ?

   (a) Attente des données d'être lues à partir d'un disque. **Bloqué**

   (b) Avoir juste accompli une E/S et attendre d'être ordonnancé encore sur le processeur. **Prêt**

Exercice 2.1 (Questions de Compréhension)

9. Un programme d'éditeur de liens
   a) place le programme dans la mémoire afin de l'exécution.
   b) traduit un programme source en un programme objet.
   c) lie le programme avec d'autres programmes nécessaires pour son exécution.
   d) Est une interface d'un programme avec les entités produisant ses données d'entrée.

Exercice 2.1 (Questions de Compréhension)

10. Quel est l'effet d'une interruption sur le registre PC?

PC prend une nouvelle valeur fournie à partie du numéro de l'interruption et de la table des vecteurs d'interruptions.

Exercice 2.1 (Questions de Compréhension)

11. Dans le système UNIX, les véritables appels système sont effectués à partir
    - d'un programme utilisateur
    - d'une commande shell
    - d'une procédure de la bibliothèque standard

    Sont-ils exécutés en mode superviseur ou en mode utilisateur ?

    A partir de la bibliothèque standard des appels système (instruction TRAP). Ils sont exécutés en mode superviseur (Leurs codes constituent le système d'exploitation).