A.Exercice 1 : Machine de Turing (4 pts)

Soit la machine de Turing définie par :

M = ({0, 1, 2}, {a, b}, {a,b,#}, E, 0, {2}, #} et les transitions suivantes :

- 0, b → 0, b ,R
- 0, a → 1, a, R
- 1, a → 0, a, R
- 1, a → 1, a, R
- 1, b → 1, b, R
- 1, # → 2, #, R

(a) Donnez le diagramme des transitions.

(b) Soit **w = baab**, la donnée fournie à la machine, faite un calcul de la machine de Turing et qualifie le résultat (calcul acceptant, calcul non acceptant).

B.Exercice 2 : Fonctionnent de l'unité centrale de traitement (4 pts)

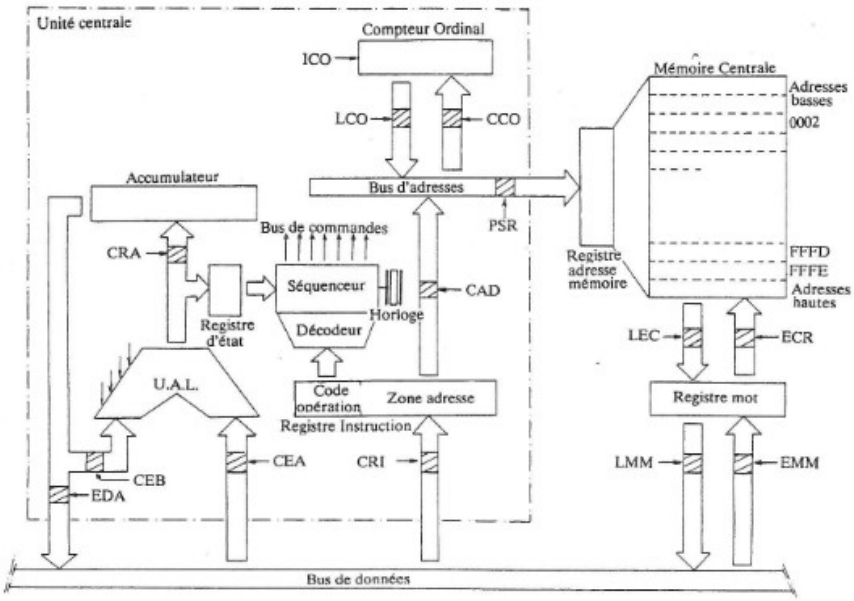


Schéma complet d'une unité centrale

LCO *Lecture Compteur Ordinal*	(Compteur Ordinal) --> Bus adresses
CCO *Chargement Compteur Ordinal*	(Bus adresses) --> Compteur Ordinal
PSR *Pointage Sur Registre*	(Bus d'adresses) --> Registre Adresse Mémoire
LEC *LECture*	(Mémoire) --> Registre Mot
ECR *ÉCRiture en mémoire*	(Registre Mot) --> Mémoire
LMM *Lecture Mot Mémoire*	(Registre Mot) --> Bus de Données
EMM *Écriture Mot Mémoire*	(Bus de Données) --> Registre Mot
CAD *Chargement ADresse*	(Reg Instr adresse) --> Bus d'Adresses
CRA *Chargement Registre Accumulateur*	(UAL sortie) --> Accumulateur
CRI *Chargement Registre Instruction*	(Bus de Données) --> Registre Instruction
CEB *Chargement Entrée B*	(Accumulateur) --> Entrée B de l'UAL
CEA *Chargement Entrée A*	(Bus de Données) --> Entrée A de l'UAL
EDA *Envoi de Donnée Accumulateur*	(Accumulateur) --> Bus de données
ICO *Incrémentation du Compteur Ordinal*	(Compteur Ordinal) + 1
NOP *No OPeration*	la donnée passe de l'entrée A à la sortie sans opération
ADD *Addition*, **SUB** *Soustraction*, **ET**, **OU** logique, etc	

Description des microcomandes

Exemple :
Soit un programme chargé de l'addition de deux nombres :

•0x8 -> stocké en 0xF800

•0x4 -> stocké en 0xF810
Le résultat 0xC est stocké en 0xF820.

•Étape 1 : chargé 0x8 dans le registre l'accumulateur (A)

•Étape 2 : Faire l'addition du contenu de A avec la seconde donnée (0x4) et le résultat est remis dans A.

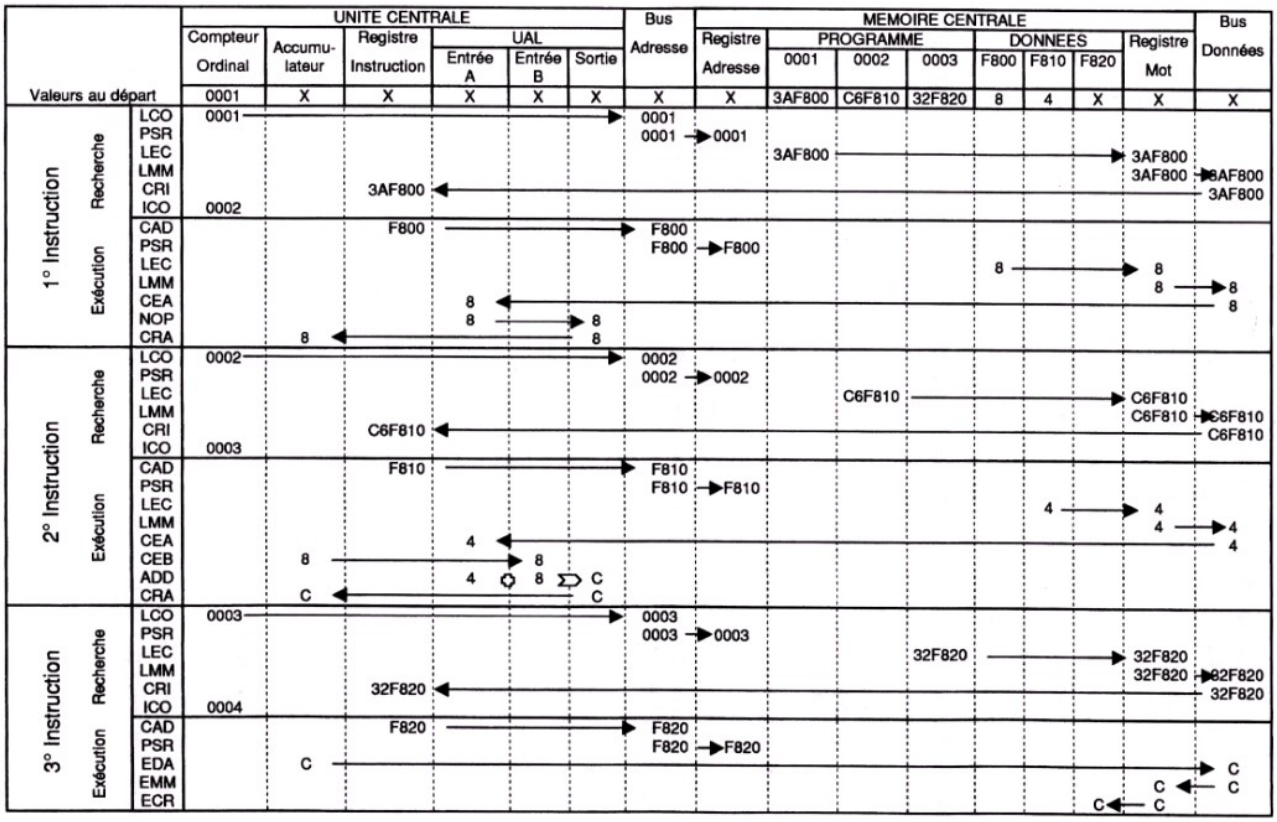
•Étape 3 : Ranger le résultât (0xC) en mémoire à l'adresse 0xF820

En langage d'assemblage cela donne :

```
LD A,(F800H)          Le code machine généré          3A F8 00
ADD A,(F810H)                                         C6 F8 10
LD (F820H),A                                          32 F8 20
```

On stocke ses instruction en 0001H, 0002H et 0003H. (un mot devrait contenir la même quantité d'information, donc l'instruction 3A F8 00, devrait être en réalité stockée sur trois mots mémoires.
Mais par souci de simplification, on stocke toute l'instruction dans un seul mot mémoire.

Tableau des commandes générées par le programme d'addition



Question : En vous inspirant de l'exemple, réaliser un tableau de fonctionnement pour le programme qui soustrait le nombre **3H** , rangé à l'adresse F820H du nombre **9H** , rangé à l'adresse F810H et range le résultat à l'adresse F820H.

Considérons pour l'exercice que les instructions en pseudo-assembleur et leur équivalent en langage machine sont :

```
LD A,(F800H)          Le code machine généré          3A F8 00
SUB A,(F810H)                                         D6 F8 10
LD (F820H),A                                          32 F8 20
```

On stocke ses instruction en 0001H, 0002H et 0003H.

C.Exercice 3 : Ordonnancement de processus (6 pts)

Nous désirons connaître l'heure à laquelle les 4 processus A, B, C et D (dont les caractéristiques sont données dans le tableau plus loin) vont **terminer leur exécution**, et dans quel ordre ces processus sont exécutés au cours du temps suivant les différents algorithmes d'ordonnancement donnée ci-après :

(2 pts) a) l'algorithme du *tourniquet* avec un quantum de 200ms

(4 pts) b) l'algorithme d'ordonnancement à 2 niveaux basé sur le modèle **Unix**, ou :

•le quantum est de 100ms,

•un top toutes les 25ms,

•toutes les secondes les priorités des processus sont réévaluées suivant la formule suivante : nouvelle_priorité = priorité_initiale + nb_top/2.

Les processus ont les caractéristiques suivantes :

Processus	Priorité initiale	Heure d'arrivée	Blocage après son propre temps d'exécution (comme en TD)	Heure de reprise	Durée d'exécution
A	10	10h00'00''	Après 0,3s	10h00'01,4	1,2s
B	10	10h00'00''	-	-	0,9s
C	10	10h00'00''	Après 0,3s	10h00'01,4	0,6s
D	16	10h00'01''	-	-	1,3s

Remarque :
quand un processus est en attente il est dans une file d'attente, et si 2 processus sont prêts en même temps alors le processus « nouveau » (par rapport à celui qi vient de s'exécuter) rentrera avant le processus qui vient d'être exécuté ; ces processus seront « mis » en file d'attente.
Exemple : on a une file d'attente avec BC et D qui vient de s'exécuter si A arrive en même temps que D finit (son quantum) alors la file d'attente devient BC**AD**.

D.Exercice 4 : Gestion de la mémoire (6 pts)

D.I.Pagination et segmentation (4 pts)

Nous supposerons dans tout cet exercice qu'un processus dispose de 4 segments 0, 1, 2, et 3 de taille respective 16394, 8250, 32768 et 4084.

Pour chaque question, trouver l'adresse physique des adresses virtuelles suivantes où le terme de gauche dans chaque couple représente le numéro du segment concerné, et le terme de droite le déplacement à l'intérieur de chaque segment :

1.(0, 8196)

2.(1, 1024)

3.(2, 30000)

4.(3, 4090)

(a) Segmentation pure (les segments ne sont pas paginés et sont donc stockés complètement en mémoire)

Donnez l'adresse physique de chaque adresse virtuelle si la table des segments est la suivante :

N° segment	Adresse physique
0	8500
1	10
2	29990
3	24900

(b) Segmentation paginée (chaque segment est paginé)

La mémoire physique est paginée (donc les segments sont eux-mêmes divisés en pages) avec la taille de 4Ko pour les pages. Calculez les adresses physiques des adresses virtuelles précédentes sous cette nouvelle hypothèse avec la table des correspondances suivante :

N ° segment	N ° page virtuelle	N° page physique	Bit de présence
0	4	0	1
0	3	1	1
2	7	2	1
1	1	3	1
0	0	4	1
1	0	5	1
1	2	6	1
0	1	7	1
3	0	8	1
2	6	9	1
2	0	10	1
2	4	11	1
2	2	12	1
2	5	13	1
0	3	14	1
0	2	15	1

D.II.Remplacement de pages (2 pts)

Soit la liste des pages virtuelles référencées aux instants

t = 1, 2, 3, …, 11 : 4 5 6 8 4 9 6 12 4 6 10.

La mémoire centrale est composée de 4 cases initialement vides. Représentez l'évolution de la mémoire centrale au fur et à mesure des accès pour chacune des deux politiques de remplacement de pages ***FIFO*** et ***LRU***. Notez les défauts de pages éventuels.

<u>Rappel</u> :

•Remplacement ***FIFO***
Avec cet algorithme, c'est la page la plus anciennement chargée qui est remplacée.

•Remplacement **LRU**
Avec cet algorithme, c'est la page la moins récemment utilisée qui est remplacée.