# Architecture des ordinateurs
# Examen session principale
# ING 1 GI
# 2022 - 2023

**Modalités :**

- Durée : 2h
- Aucun document n'est autorisé, calculatrice autorisée
- Annexe inclue
- Pages 5 et 6 à compléter et à rendre

**Exercice 1 : QCM (Pour chaque question, une seule réponse est correcte) (2 pts)**

1. **Bus :**

a. Il y a quatre types de bus : données, adresse, commande et contrôle.
b. Un bus est un groupe de lignes électriques qui relie le CPU aux autres composantes.
c. Chaque ligne électrique d'un bus peut transférer deux bits d'information à la fois.
d. Le bus de données permet le transfert des données dans un seul sens.

2. **Adressage :**

a. Adressage registre est un adressage entre adresse mémoire et valeur implicite.
b. Adressage direct est un adressage entre deux registres.
c. Adressage immédiat est un adressage entre deux adresses mémoire.
d. Adressage indirect est un adressage entre un registre et une adresse mémoire.

3. **Soit le nombre -27 en base décimale. Comment écrire ce nombre en binaire complément à 2 sur 8 bits ?**

a. 1110 0101
b. 0001 1011
c. 1110 0100
d. 0001 1100

4. **L'unité de commande ne contient pas :**

a. Décodeur
b. Séquenceur
c. Accumulateur
d. Registre d'instruction

**Exercice 2 : Circuit Logique (5,5 pts)**

On désire effectuer la synthèse d'un compteur synchrone modulo 8 à base de bascule D. Un compteur modulo 8 est un type de compteur qui compte jusqu'à 7 puis recommence à 0.
On propose d'utiliser des bascules D à front montant.

1. Justifier le choix de **trois** bascules D.
   **Mod 8 → Comptage de 0 à 7 : $2^3 = 8$ → 3 bascules D sont nécessaires**
2. Compléter la table de transition (***Tableau 1***).

| Etat présent ($Q_n$) | | | Etat futur ($D_n = Q_{n+1}$) | | |
|---|---|---|---|---|---|
| $Q_2$ | $Q_1$ | $Q_0$ | $D_2$ | $D_1$ | $D_0$ |
| 0 | 0 | 0 | **0** | **0** | **1** |
| 0 | 0 | 1 | **0** | **1** | **0** |
| 0 | 1 | 0 | **0** | **1** | **1** |
| 0 | 1 | 1 | **1** | **0** | **0** |
| 1 | 0 | 0 | **1** | **0** | **1** |
| 1 | 0 | 1 | **1** | **1** | **0** |
| 1 | 1 | 0 | **1** | **1** | **1** |
| 1 | 1 | 1 | **0** | **0** | **0** |

**Tableau 1 -Table de transition d'un compteur modulo 8**

3. Remplir les tableaux de Karnaugh correspondant à ces fonctions $D_i$, puis donner l'équation réduite de chaque fonction.

**$D0 = \overline{Q0}$**

| Q0Q100 | 01 | 11 | 10 |
|---|---|---|---|
| **Q2** | | | |
| **0** | 1 | 1 | 0 | 0 |
| **1** | 1 | 1 | 0 | 0 |

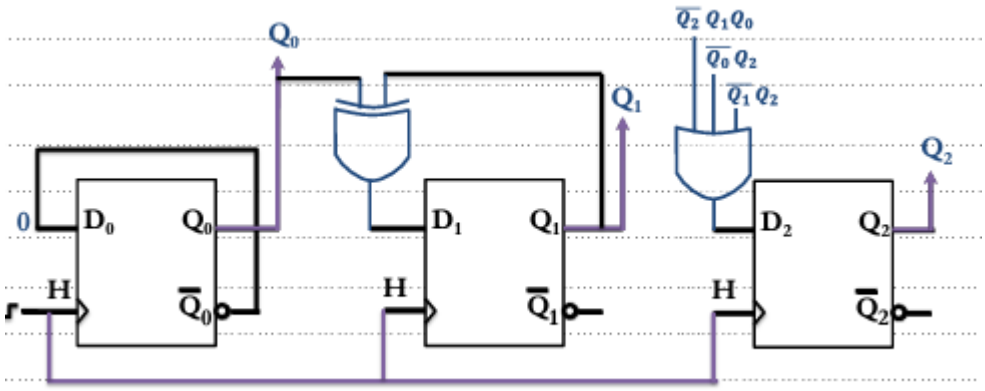
**$D1 = \overline{Q0} \cdot Q1 + Q0 \cdot \overline{Q1} = Q0 \text{ XOR } Q1$**

| Q0Q100 | 01 | 11 | 10 |
|---|---|---|---|
| **Q2** | | | |
| **0** | 0 | 1 | 0 | 1 |
| **1** | 0 | 1 | 0 | 1 |

**$D2 = Q0 \cdot Q1 \cdot \overline{Q2} + \overline{Q1} \cdot Q2 + \overline{Q0} \cdot Q2$**

| Q0Q100 | 01 | 11 | 10 |
|---|---|---|---|
| **Q2** | | | |
| **0** | 0 | 0 | 1 | 0 |
| **1** | 1 | 1 | 0 | 1 |

4. Compléter le schéma de câblage de ce compteur (***schéma 1***).




**OU**

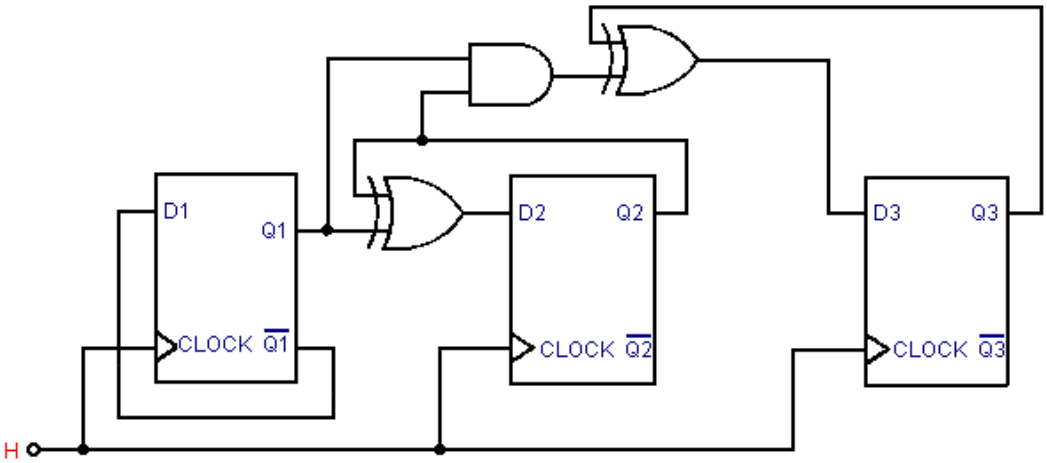
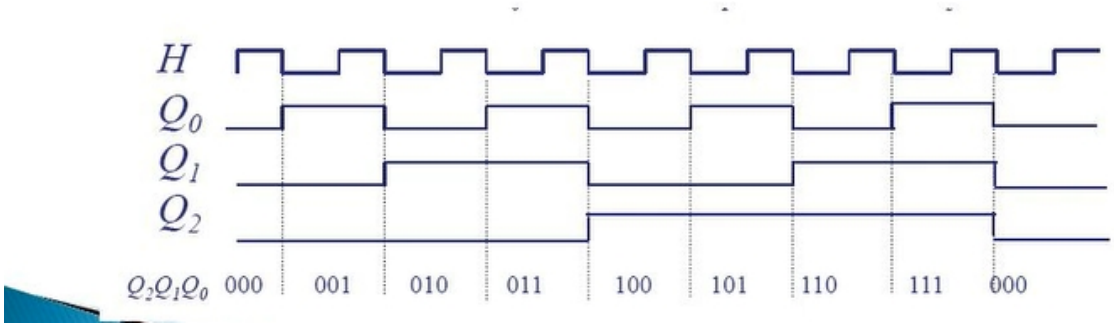



Fig. 24. - Compteur synchrone modulo 8.

5. Compléter le logigramme correspondant.




**Exercice 3 : (5,5 pts)**

Nous disposons d'un ordinateur disposant de 512 Mo de mémoire vive dont la taille du mot mémoire est de 32 bits.

**A.**

1. Déterminer l'adresse la plus haute (la plus grande) de cette mémoire.
   **Nombre de mots mémoire=512 MO/4 O=$2^7$*$2^{20}$=$2^{27}$ mots mémoire**

   **Donc l'adresse la plus haute est : $2^{27}$ – 1**

   **$2^{27}$ - 1 = $(1….1)_{27\ uns}$ en binaire ou bien $(7FFFFFF)_{16}$**

2. Convertir l'adresse trouvée **en octal.**
   **$2^{27}$ - 1 = $(7….7)_{9septs}$ en octal**

**B.** Un tableau de réels, selon la norme IEEE-754 double précision, est stocké dans la mémoire de cette machine à partir de l'adresse **0AE**.

3. Calculer l'adresse **en décimal** du 8ème élément de ce tableau.
   **$(0AE)_{16}$=$(174)_{10}$**
   **L'adresse du 8ème élément = Adresse du 1er élément + [(8-1) * nombre de mots d'un élément]**

   **= 174 + [(7*2)] = $(188)_{10}$ = $(0BC)_{16}$**

   **<u>Rappel :</u> la position du premier élément d'un tableau est 0, pour cela 8-1 pour avoir le 8ème élément du tableau**
   **Chaque mot = 32 bits, mais là nous demandons Double précision (64 bits) : alors chaque adresse (élément) aura 2mots, pour cela *2**

4. Combien **d'octets** précédent l'adresse de début de ce tableau.
   **Nbre d'octets = Adresse du 1er élément * taille d'un mot (en octets)**

   **= 174 * 4 = 696 OCTETS**

**C.** Nous désirons effectuer une addition sur deux éléments du tableau de réels.

6. Donner l'instruction en utilisant le mode d'adressage immédiat qui permet de modifier le contenu du registre **ESI**. **ESI** contiendra l'indice du cinquième élément du tableau.

   **MOV ESI, 8 [8 c'est 4 (pour 5eme élément) * 2 (nombre de mots par élément)]**

7. Donner l'instruction en utilisant le mode d'adressage indexé pour accéder au cinquième élément du tableau, sachant que le registre de base contenant l'adresse de début du tableau est **EBX.**

**MOV EAX, [EBX + ESI]**

**Exercice 4 : jeux d'instruction (3 pts)**

Soit l'extrait de programme ASSEMBLEUR INTEL 8086 suivant avec les valeurs initiales :
AX = $0000_H$, BX = $0000_H$, le Flag z = 0 et l'état de pile suivant : SP=$FFFC_H$,
$FFFE_H$ = 0002 $FFFC_H$ = 0001 $FFFA_H$ = 0000

Soit le code en assembleur suivant :

```
POP AX
MOV BX, 000A H
```

MUL AX
**Boucle** : ADD AX, $0005_H$
SUB BX, $0008_H$
CMP BX, 2
JNE **Boucle**
PUSH BX
PUSH AX

Compléter le tableau correspondant aux contenus des différents registres (***Tableau 2***) sachant que chaque ligne représente une étape d'exécution du code précédent.

**OBS : Chaque adresse de la pile (stack) FFFF, FFFE, FFFD, FFFC, FFFB, FFFA**
**« stock » 8 bits = 2 chiffres, exemple 01 ou 00**
**Dans cet exo le SP « pointe » sur FFFC.**
**POP AX copie 00 01 dans AX et libere la pile. Pour cela la deuxième ligne de la pile devient 00 00 à la place de 00 01. Et le pointeur de pile passe à FFFE**
**MUL AX multiple AX par lui même. Alors 0001 * 0001 = 0001**
**PUSH BX et PUSH AX copie l'information de registres sur la pile et bouge le pointeur.**

| Instruction | AX | BX | Flag z | SP | Stack :<br>FFFF, FFFE, FFFD, FFFC, FFFB, FFFA |
|---|---|---|---|---|---|
| Etat initial | 0000 | 0000 | 0 | FFFC | 00 02 00 01 00 00 |
| POP AX | 0001 | 0000 | 0 | FFFE | 00 02 00 00 00 00 |
| MOV BX, $000A_H$ | 0001 | 000A | 0 | FFFE | 00 02 00 00 00 00 |
| MUL AX | 0001 | 000A | 0 | FFFE | 00 02 00 00 00 00 |
| Boucle : ADD AX, $0005_H$ | 0006 | 000A | 0 | FFFE | 00 02 00 00 00 00 |
| SUB BX, $0008_H$ | 0006 | 0002 | 0 | FFFE | 00 02 00 00 00 00 |
| CMP BX, 2 | 0006 | 0002 | 1 | FFFE | 00 02 00 00 00 00 |
| JNE Boucle | 0006 | 0002 | 1 | FFFE | 00 02 00 00 00 00 |
| PUSH BX | 0006 | 0002 | 1 | FFFC | 00 02 00 02 00 00 |
| PUSH AX | 0006 | 0002 | 1 | FFFA | 00 02 00 02 00 06 |

**Tableau 2**

**Exercice 5 : Assembleur NASM (4 pts)**

Écrire un programme assembleur bien commenté qui calcule la somme des carrés des entiers de 1 à 10 et affiche le résultat.

```
section .data
   result db 0         ; Variable pour stocker le résultat
   format db "Le résultat est %d.", 10, 0  ; Chaîne de formatage pour l'affichage

section .text
   global _start

_start:
   mov ecx, 10         ; Initialiser le compteur à 10
   mov eax, 0          ; Initialiser le résultat à zéro

loop_start:
   add eax, ecx*ecx     ; Ajouter le carré du compteur au résultat
   dec ecx             ; Décrémenter le compteur
   cmp ecx, 0          ; Vérifier si le compteur est égal à zéro
   jne loop_start      ; Boucler tant que le compteur n'est pas égal à zéro

   ; Afficher le résultat
   push eax            ; Mettre le résultat sur la pile pour printf
   push format         ; Mettre la chaîne de formatage sur la pile pour printf
   call printf         ; Appeler la fonction printf pour afficher le résultat

   ; Terminer le programme
   mov eax, 1          ; Charger le code de sortie 1 dans EAX
   xor ebx, ebx        ; Mettre EBX à zéro pour signaler une sortie normale
   int 0x80            ; Appeler le système pour terminer le programme
```