

**Modalités :**

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

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

**1. Quelles méthodes pouvons-nous utiliser pour simplifier les fonctions booléennes ?**

- ☐ Théorèmes et propriétés de l'Algèbre de Boole ET Tableaux de Karnaugh
- ☐ Table de vérité ET Tableaux de Karnaugh
- ☐ Table de vérité ET Portes logiques
- ☐ Portes logiques ET Théorèmes et propriétés de l'Algèbre de Boole

**2. La mémoire peut posséder au maximum :**

- ☐  $2^n - 1$  cases mémoires
- ☐  $2^{n+1}$  cases mémoires
- ☐  $2^n$  cases mémoires
- ☐  $2^{n-1}$  cases mémoires

**3. L'unité de commande contient :**

- ☐ Décodeur et Accumulateur
- ☐ Accumulateur et Séquenceur
- ☐ Compteur Ordinal et l'Unité Arithmétique et Logique
- ☐ Séquenceur et Compteur Ordinal

**4. Soit le nombre représenté en virgule flottante selon la norme IEEE754 simple précision (32 bits). Quelle est la valeur décimale du nombre suivant : 1 1000 0011 1001110000000000000000 ?**

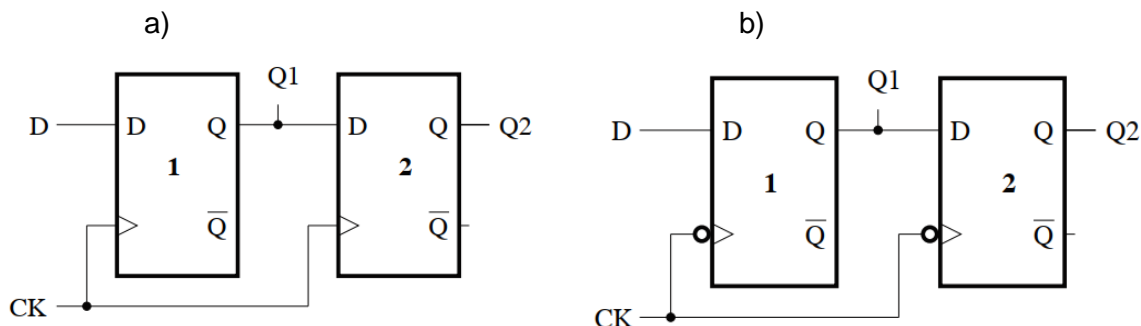
- ☐ 0,2575
- ☐ -25,75
- ☐ 25,75
- ☐ -0,2575

**Exercice 2 : Circuit Logique (6 pts)**

1. On désire effectuer un compteur synchrone modulo 7 à base de bascules JK synchronisées sur front montant. Un compteur modulo 7 est un type de compteur qui compte jusqu'à 6 puis recommence à 0.

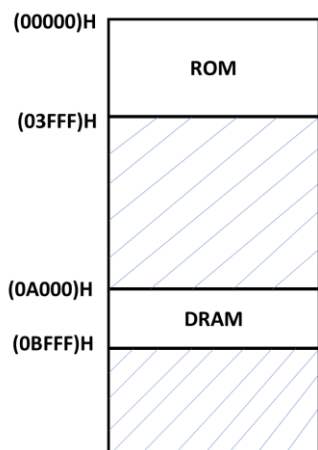
- a) Compléter la table de transition (**Tableau 1 – page 5**).
- b) Remplir les tableaux de Karnaugh correspondant aux fonctions  $J_i$  et  $K_i$ , puis donner l'équation réduite de chaque fonction.
- c) Compléter le schéma de câblage de ce compteur (**Schéma 1 - page 5**).

2. Soit les circuits logiques ci-dessous utilisant deux bascules D. Compléter le chronogramme de chaque circuit (**Schéma 2 (a) et Schéma 3 (b) – pages 5 et 6**).



### Exercice 3 : Mémoire (5 pts)

Un microprocesseur a un bus d'adresse de 20 bits et un bus de données de 8 bits. La figure ci-dessous illustre la mémoire formée par une ROM et une DRAM, ainsi que les adresses de chaque partie de la mémoire.



1. Quel est l'espace adressable du processeur ?
2. Quelle est la capacité totale de cette mémoire ?
3. Quels sont les adresses de début et fin de cette mémoire ?
4. Quel est le nombre de bits permettant d'adresser la ROM ?
5. Quelle est la capacité de la ROM ?
6. Quelle est la capacité de la DRAM ?

### Exercice 4 : Jeux d'instruction (3 pts)

Soit l'extrait de programme ASSEMBLEUR INTEL 8086 suivant avec les valeurs initiales :  
 AX = 0001<sub>H</sub>, BX = 0000<sub>H</sub>, le Flag z = 0 et l'état de pile suivant : SP = FFFC<sub>H</sub>,  
 FFFE<sub>H</sub> = 0001, FFFC<sub>H</sub> = 0002 et FFFA<sub>H</sub> = 0000.

Soit le code en assembleur suivant :

```
POP AX
MOV BX, 000BH
Boucle : ADD AX, 0002H
SUB BX, 0009H
CMP BX, 2
JNE Boucle
PUSH BX
PUSH AX
```

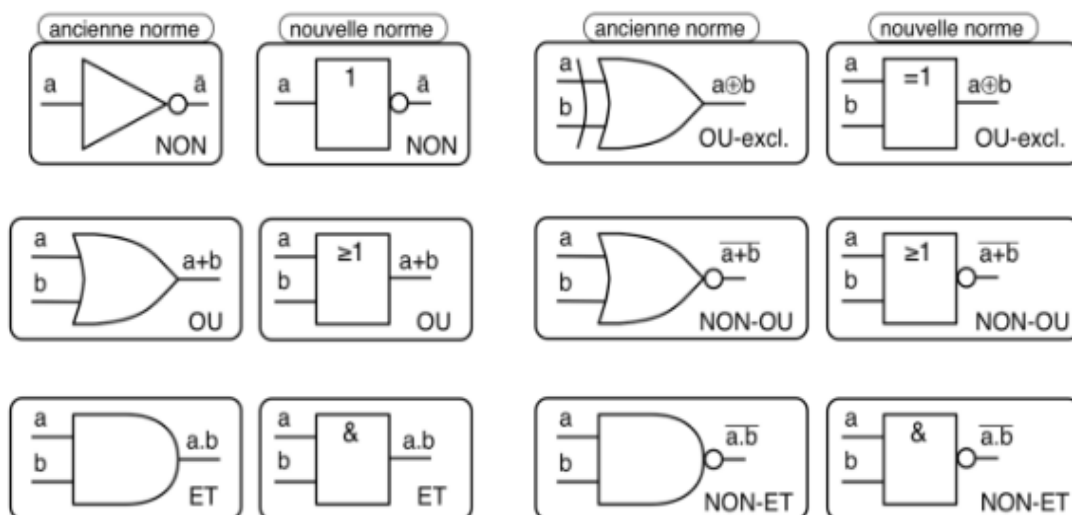
Compléter le tableau correspondant aux contenus des différents registres (**Tableau 2- page 6**) sachant que chaque ligne représente une étape d'exécution du code précédent.

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

Ecrire un programme qui affiche une chaîne de caractères saisie à partir du clavier.

## Annexe portes logiques et assembleur x86

### Portes logiques



Symboles des portes logiques.

### Assembleur

#### ● Empiler : push

Syntaxe

push [source](#)

- ☐ copie le contenu de [source](#) au sommet de la pile
- ☐ commence par décrémenter [esp](#) de 4 puis effectue la copie
- ☐ [source](#): adresse, constante ou registre

#### ● Dépiler : pop

Syntaxe

pop [destination](#)

- ☐ copie les 4 octets qui se trouvent au sommet de la pile dans [destination](#)
- ☐ commence par effectuer la copie puis incrémente [esp](#) de 4
- ☐ [destination](#): un registre ou une adresse

- **Comparaison : cmp**

Syntaxe

cmp destination, source

- ☐ Effectue l'opération destination - source
- ☐ destination: registre ou adresse
- ☐ source: constante, registre ou adresse
- ☐ Les valeurs des flags ZF , SF et CF sont éventuellement modifiées.

- **Incrémentation: Inc**

Syntaxe

Inc destination

- ☐ incrémente destination

- **Décrementation: Dec**

Syntaxe

Dec destination

- ☐ décremente destination

**Instruction de saut inconditionnel : JMP**

Syntaxe

jmp adr

- ☐ Va à l'adresse/ étiquette adr

**Instructions de saut conditionnel**

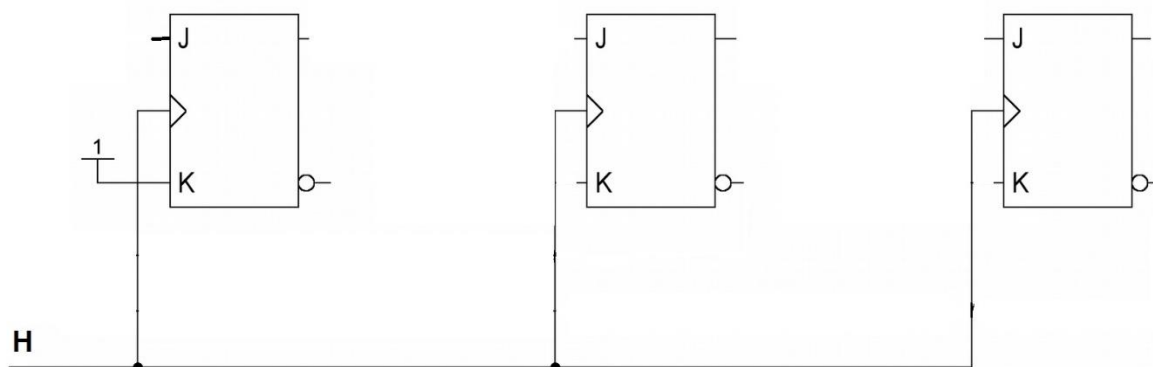
	Instruction	Description	Indicateurs
<b>Valeurs non signées</b>	JB	Jump below	CF = 1
	JBE	Jump below or equal	CF = 1 et ZF=1
	JA	Jump above	CF =0 et ZF=0
	JAЕ	Jump above or equal	CF=0
	JE/JZ	Jump if equal /Jump if zero	ZF=1
	JNE/JNZ	Jump if not equal/ Jump if not zero	ZF=0
<b>Valeurs signées</b>	JL	Jump less	SF=NOT OF
	JLE	Jump less or equal	ZF=1 ou SF=NOT OF
	JG	Jump greater than	ZF=0 et SF=OF
	JGE	Jump greater or equal	SF=OF

**Rappel:** ZF (zero flag), SF (sign flag), CF (carry flag), OF (overflow flag)

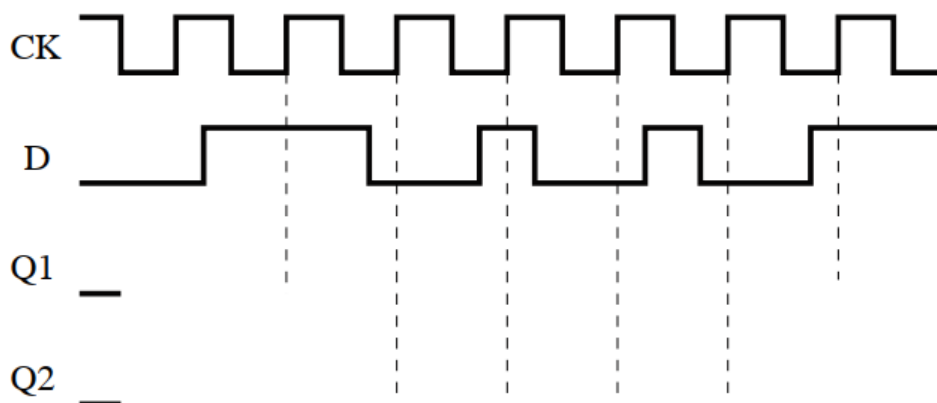
**Pages à compléter et à rendre**

	État présent ( $Q_n$ )			État futur ( $Q_{n+1}$ )					
	$Q_2$	$Q_1$	$Q_0$	$J_2$	$K_2$	$J_1$	$K_1$	$J_0$	$K_0$
0	0	0	0						
1	0	0	1						
2	0	1	0						
3	0	1	1						
4	1	0	0						
5	1	0	1						
6	1	1	0						

**Tableau 1**



**Schéma 1 (utilisez des couleurs si nécessaire)**



**Schéma 2 (utilisez des couleurs si nécessaire)**

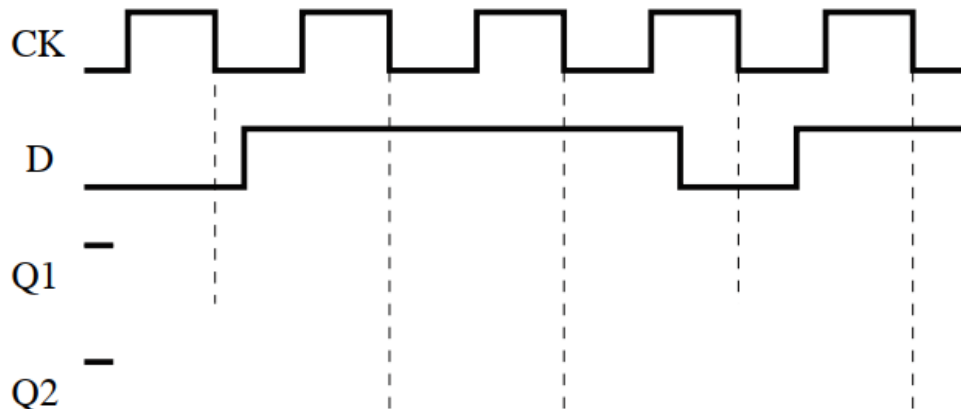


Schéma 3 (utilisez des couleurs si nécessaire)

	AX	BX	Flag z	SP	Stack: FFFF, FFFE, FFFD, FFFC, FFFB, FFFA
État initial					

Tableau 2