Ing1 – Examen d'Architecture des Ordinateurs

Durée : 2H – Aucun document autorisé – Calculatrice Interdite.
Documentation assembleur Z80 en annexe.

Exercice 1 – Espace adressable

On considère une machine avec une CPU gérant des adresses de 24 bits ($a_{23}…a_0$) et des mots de 12 bits ($d_{11}…d_0$). La machine intègre une ROM, 3 emplacements pour des barrettes de RAM ainsi qu'une interface série et une interface parallèle pour les périphériques. Le circuit de décodage d'adresse utilise des décodeurs décimaux 4 bits (DEC_1 à DEC_5). L'organisation de la machine est donnée par les 2 plans aux pages suivantes. Seuls les signaux d'autorisation, de données et d'adresse sont représentés. Ils ne sont pas tous tracés pour ne pas alourdir le schéma ; les connexions sont alors nommées.

1.a) Quel est l'espace adressable maximal de cette machine (en mots) ?

1.b) Quelle est la plage d'adresses (en hexa) ?

1.c) Quelle est la taille de la ROM et de la RAM (en octets) ?

1.d) Donner les équations des signaux d'activation des 4 boîtiers mémoires et des 2 interfaces.

CS_{RO1} =

CS_{RA1} =

CS_{RA2} =

CS_{RA3} =

CS_{SE} =

CS_{PA} =

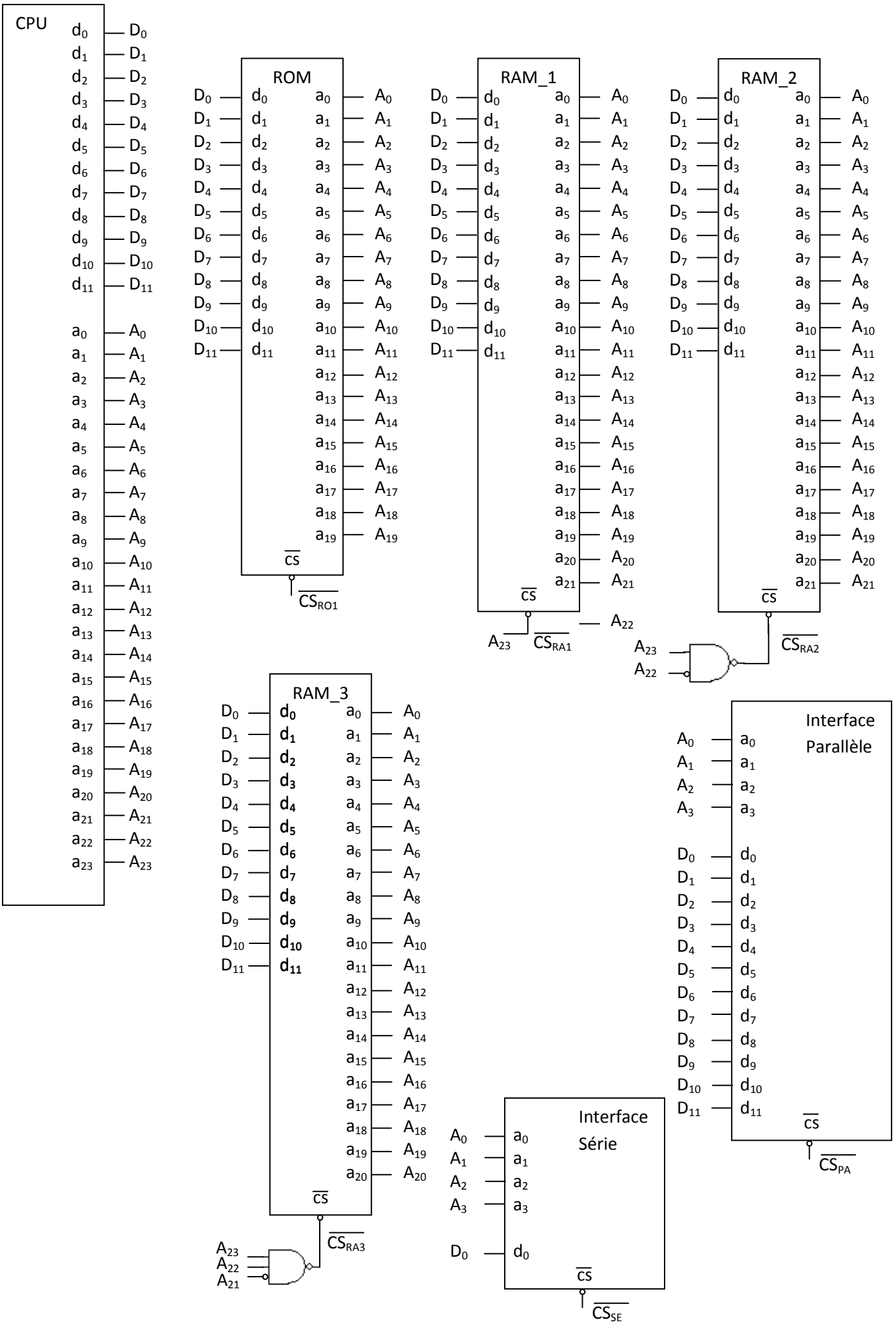
1.e) Compléter le tableau suivant :

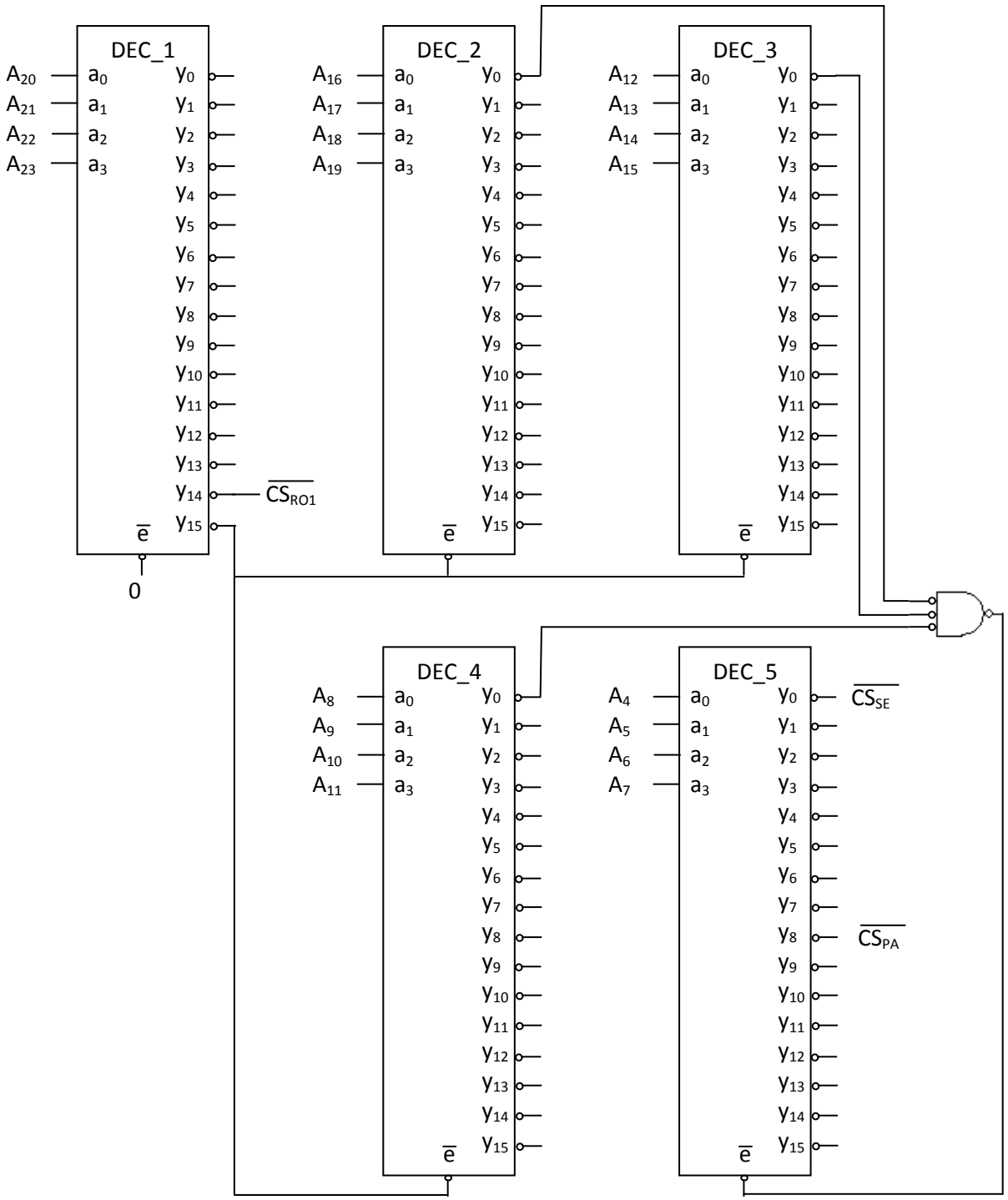
	Adresse minimale (hexa)	Adresse maximale (hexa)
ROM		
RAM 1		
RAM 2		
RAM 3		
Interface série		
Interface parallèle		

1.f) A quoi sert le fil d'adresse A22 non connecté à la RAM_1 ?

1.g) Pour résumer, dessiner la carte mémoire du système (mapping des adresses).

NB : rappel des unités de donnés : K, M, G, T, P, E, Z, Y


CPU
ROM
RAM_1
RAM_2
RAM_3
Interface Parallèle
Interface Série
CS
CS_RO1
CS_RA1
CS_RA2
CS_RA3
CS_PA
CS_SE


DEC_1
DEC_2
DEC_3
DEC_4
DEC_5
A_{20} A_{21} A_{22} A_{23}
A_{16} A_{17} A_{18} A_{19}
A_{12} A_{13} A_{14} A_{15}
A_8 A_9 A_{10} A_{11}
A_4 A_5 A_6 A_7
a_0 a_1 a_2 a_3
y_0 y_1 y_2 y_3 y_4 y_5 y_6 y_7 y_8 y_9 y_{10} y_{11} y_{12} y_{13} y_{14} y_{15}
$\overline{e}$
0
$\overline{CS_{RO1}}$
$\overline{CS_{SE}}$
$\overline{CS_{PA}}$

Exercice 2 – Entiers Relatifs

On considère les entiers relatifs codés en complément à 2 sur 4 bits.

2.a) Quel est l'intervalle des nombres codables dans ce système ?

2.b) Donner le code binaire des entiers relatifs suivants : +2, -2, +6, -6

3.c) Calculer les 3 additions -2 + 2, -2 + (-6), 2 + 6 en binaire complément à 2 (donner le détail des calculs). Vous préciserez, en le justifiant, si les résultats obtenus sont corrects ou s'il y a dépassement de capacité.

Exercice 3 - Assembleur

On s'intéresse à la traduction en assembleur Z80 du programme Java suivant :

```
1. int v1;       // données sur 16 bits à l'adresse 0x0100
2. int v2;       // données sur 16 bits à l'adresse 0x0102
3. int produit;  // données sur 16 bits à l'adresse 0x0104

4. produit = v1 * v2;
```

Donner la traduction du calcul du produit (ligne 4) en assembleur en Z80. Pour rappel : a * b = $\overbrace{a + a + \ldots + a}^{\text{b fois}}$

Remarque : Chaque entier est codé sur 2 mots ; les poids faibles sont rangés avant les poids forts. Par exemple, si v1 contient l'entier 0xA0, on a en mémoire

Adresse	Donnée
0x0100	0x0
0x0101	0xA

Une fiche récapitulative du langage assembleur Z80 est donnée en annexe.

Exercice 4 – VHDL et Chronogramme

On considère l'extrait de programme VHDL suivant, simulant le dialogue entre une CPU et une ALU :

```
signal clock, reset : std_logic;
signal a1, b1, s1, a2, b2, s2 : integer;

process(clock,reset) -- CPU demandant une somme à l'ALU
begin
     if reset = '0' then
          a1 <= 1;
          b1 <= 1;
          s1 <= 1;
          a2 <= 1;
          b2 <= 1;
     elsif clock'event and clock = '1' then
          -- changer les valeurs à additionner
          a1 <= b1;
          b1 <= s1;
          -- demande de calcul
          a2 <= a1;
          b2 <= b1;
          -- lecture d'un résultat
          s1 <= s2;
     end if ;
end process;
```

```
process(clock,reset) -- ALU calculant la somme a2 + b2
begin
     if reset = '0' then
          s2 <= 1;
     elsif clock'event and clock = '1' then
          s2 <= a2 + b2;
     end if ;
end process;

process -- process Horloge
begin
     clock <= '0';
     wait for 10 ns;
     clock <= '1';
     wait for 10 ns;
end process;

process -- pilotage de la simulation
begin
     reset <= '0';
     wait for 1 ns;
     reset <= '1';
     wait for 1000 ns;
end process;
```

Dessiner le chronogramme de simulation sur 10 cycles d'horloge pour les signaux clock, reset, a1, b1, s1, a2, b2, s2.