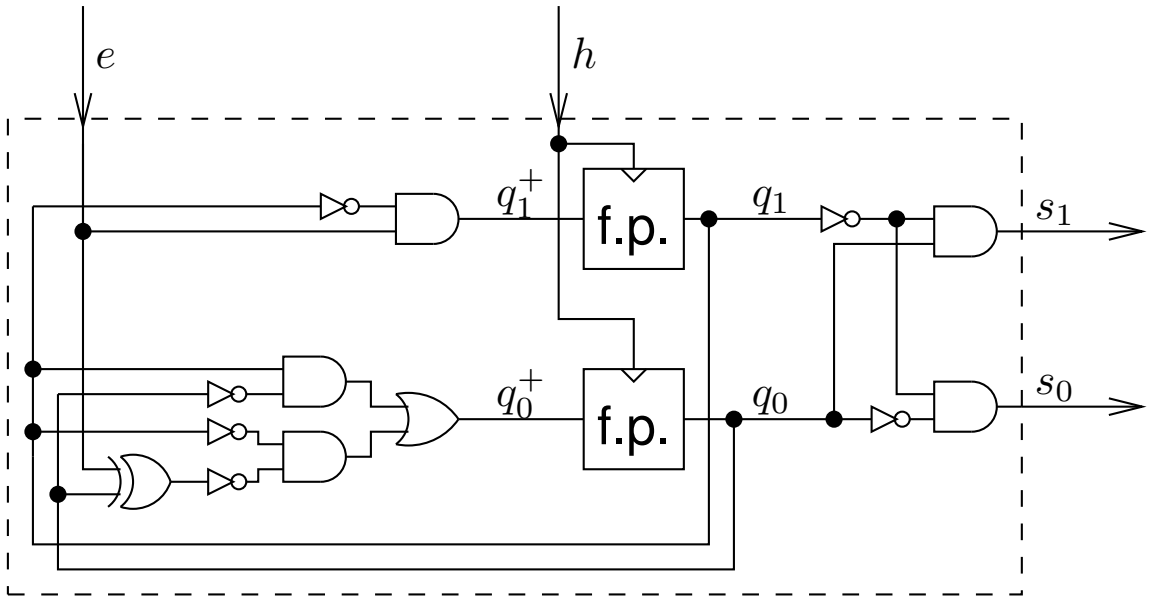
CC2 - LIF6 Automne 2013

Durée prévue : 2h00

Aucun document autorisé. Calculatrices, téléphones et ordinateurs portables interdits. N'oubliez pas de noter sur le sujet qui est à rendre votre numéro de copie. Justifiez brièvement et précisément vos réponses, tout en respectant les emplacements prévus. Le barème est donné à titre indicatif, et pourra être modifié.

1 Analyse d'un circuit séquentiel (5 pts)

On considère le circuit séquentiel ci-dessous. Ce circuit reçoit un signal d'horloge h, prend une entrée e, et produit une sortie $s = (s_1 s_0)$. Il comporte 2 bascules flip-flops, régies par le front montant de l'horloge h, qui stockent l'état courant $(q_1 q_0)$ du circuit.



La fonction de transition du circuit est la fonction f telle que $(q_1^+, q_0^+) = f(q_1, q_0, e)$. La fonction de sortie g est telle que $s = g(q_1, q_0, e)$.

1. (0.5 points) Exprimez q_1^+ et q_0^+ en fonction de q_1, q_0 et e par des expressions booléennes.

   .....

   .....

   .....

   .....

2. (0.5 points) Exprimez s_1 et s_2 en fonction de q_1 et q_0 par des expressions booléennes.

   .....

   .....

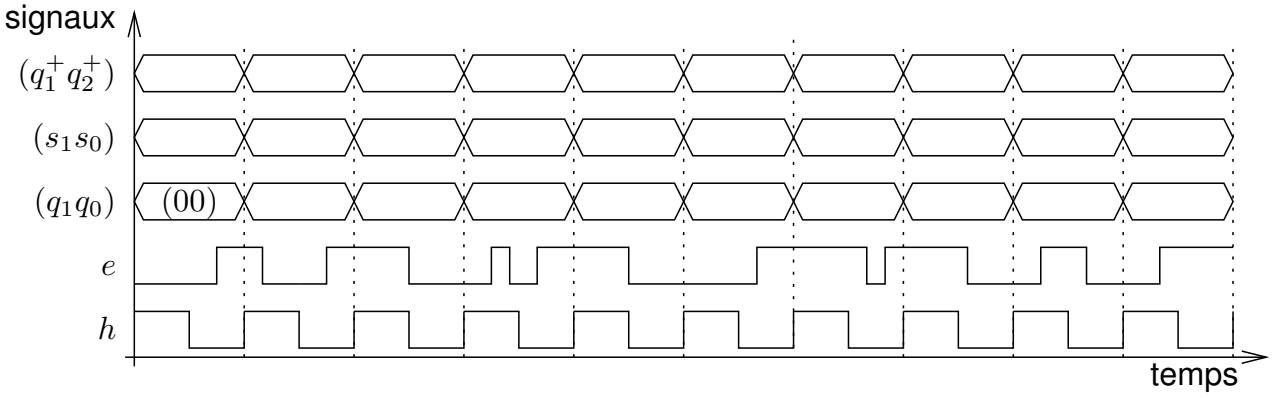
   .....

3. (0.5 points) Complétez la table de vérité suivante.

q_1	q_0	e	q_1^+	q_0^+	s_1	s_0
0	0	0				
0	0	1				
0	1	0				
0	1	1				
1	0	0				
1	0	1				
1	1	0				
1	1	1				

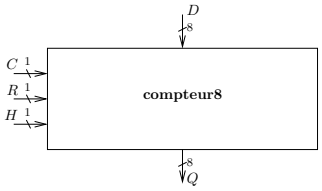
4. (2 points) Représentez le comportement du circuit sous la forme d'un automate fini séquentiel. Indiquez bien sur chaque fléche indiquant une transition la valeur de e provoquant cette transition, ainsi que la valeur de la sortie s pour chaque état.

5. (1.5 points) Complétez le chronogramme suivant.



2 Un autre circuit compteur (4 pts)

On souhaite mettre au point un circuit compteur qui maintient un entier Q codé en binaire sur 8 bits. Ce circuit prend comme entrées : un signal d'horloge créneau-périodique H, un entier codé en binaire sur 8 bits D, un signal R de remise à zéro du compteur, et un signal de commande C. Il présente en sortie la valeur actuelle Q du compteur, sous la forme d'un entier codé en binaire sur 8 bits. On suppose que le compteur maintient un décompte modulo $2^8 = 256$ (on ne se préoccupe donc pas des éventuels dépassements de capacité du compteur).



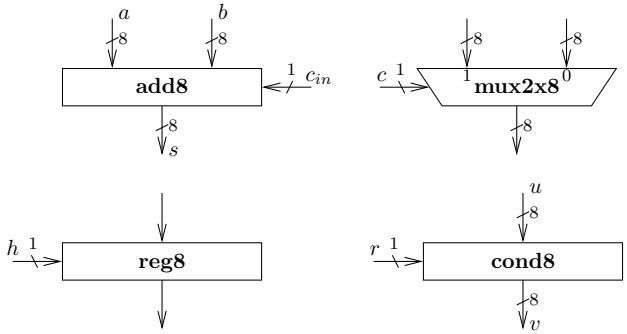
A chaque cycle d'horloge :

- Soit $R = 1$ en fin de cycle, et dans ce cas le compteur est remis à zéro.
- Soit $R = 0$ en fin de cycle, et dans ce cas :
  - Si $C = 0$ alors au cycle suivant $Q \leftarrow Q + 1 \mod 2^8$.
  - Si $C = 1$ alors au cycle suivant $Q \leftarrow Q + 1 + D \mod 2^8$.

On vous demande dans cet exercice de proposer un circuit réalisant un tel compteur. Vous disposez pour cela *a priori* des composants suivants :

- Un additionneur 8 bits (`add8`), qui prend en entrée deux entiers a et b codés sur 8 bits, ainsi qu'une retenue entrant c_{in}, et ressort $s = a + b + c_{in} \mod 2^8$ codé sur 8 bits.
- Un multiplexeur 2×8 vers 8 (`mux2x8->8`).
- Un registre 8 bits (`add8`) à base de bascules flip-flops, qui prend en entrée un signal d'horloge h, ainsi que le mot sur 8 bits à stocker, et ressort le mot stocké sur 8 bits
- Un circuit `cond8`, qui prend en entrée un signal de commande r, et un mot de 8 bits u, et produit sur sa sortie le mot de 8 bits v tel que : $v = 0$ si $r = 0$, $v = u$ sinon.

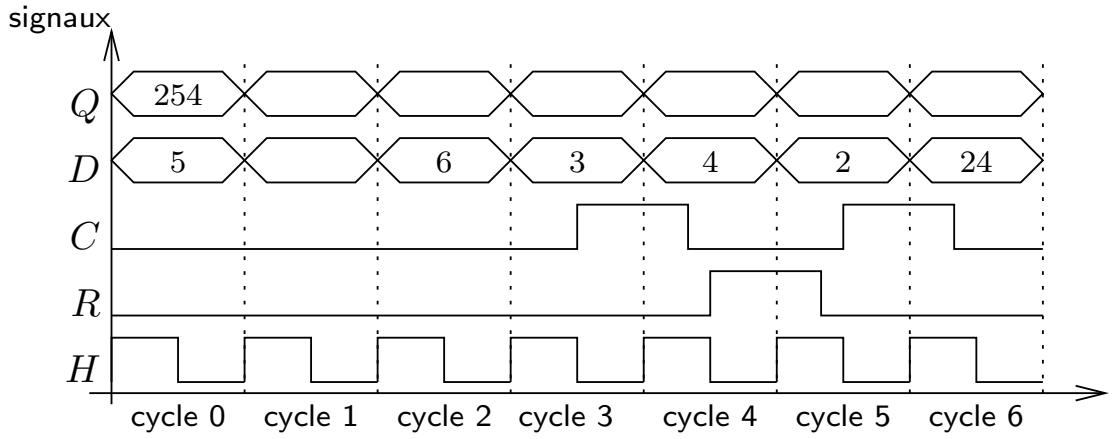
Voici les schémas représentant ces composants :



Vous pouvez en outre utiliser les portes logiques de base (AND, OR, NOT, XOR...), et générer les constantes que vous jugerez nécessaires dans votre circuit.

1. (1 point) Proposez un circuit réalisant le composant `cond8` en n'utilisant que des portes AND.

2. (1 point) Complétez le chronogramme suivant en vous conformant aux spécifications du compteur données dans l'énoncé.



3. (2 points) Proposez un circuit réalisant le compteur demandé. Il est nécessaire d'annoter soigneusement votre circuit, en indiquant bien les entrées et les sorties, leurs noms, ainsi que le nombre de bits pour chaque vecteur de signaux que vous représentez.

3 Saisie d'un entier naturel décimal (7 pts)

Le but est ici d'écrire une routine `saisie` permettant de saisir un entier en décimal sur le LC3. Pour cela, vous utiliserez la méthode de Horner, donc il va d'abord falloir implanter une multiplication par la constante 10. Ensuite, vous écrirez une routine `aff` permettant d'afficher naïvement la valeur d'un entier naturel n, en affichant n étoiles. Voici l'allure d'un programme mettant en œuvre les routines `saisie` et `aff` :

```
          .ORIG x3000
; Programme principal
main:     LD R6,spinit  ; on initialise le pointeur de pile
          JSR saisie    ; R1 <- lecture d'un entier naturel au clavier
          JSR aff       ; affichage de R1 étoiles à l'écran
          HALT

; partie qui va être complétée dans la suite de l'exercice

; Gestion de la pile
spinit:   .FILL stackend
          .BLKW #256
stackend: .BLKW #1      ; adresse du fond de la pile
          .END
```

1. (1 point) Complétez la routine `mul10` ci-dessous, qui doit permettre de multiplier par $(10)_{10}$ le contenu du registre $R1$. Vous procéderez sans utiliser de boucle, en remarquant que si n est un entier naturel, alors $n \times 10 = 2 \times n + 2 \times 2 \times 2 \times n$. Vous n'utiliserez que le registre `R2` comme registre temporaire.

```
; routine mul10 : effectue R1 <- R1 * 10.
; paramètre donnée-résultat : R1
; registre temporaire (écrasé par l'appel) : R2
; On note n la valeur d'entrée de R1 dans les commentaires.
mul10:    ..............  ; ......................................
          ..............  ; ......................................
          ..............  ; ......................................
          ..............  ; ......................................
          ..............  ; ......................................
          ..............  ; ......................................
          ..............  ; ......................................
          ..............  ; ......................................
```

2. (3 points) Pour la routine de saisie d'un entier naturel, vous allez traduire le pseudo-code suivant, qui utilise la méthode de Horner :

```
routine saisie(paramètre résultat : R1) {
  R1 <- 0;
  R0 <- readcar();        // lecture d'un caractère au clavier (IN)
  while(R0 - '\n' != 0) { // le code ascii de '\n' est 13
    R2 <- '0';            // le code ascii de '0' est 48
    R0 <- R0 - '0';       // conversion du caractère en un chiffre
    R1 <- R1 * 10;        // utiliser ici un appel à mul10
    R1 <- R1 + R0;
    R0 <- readcar();      // lecture d'un caractère au clavier (IN)
  }
  return;
}
```

Complétez, d'après le pseudo-code de la question précédente, le code de la routine `saisie`. Comme vous allez faire appel à `mul10` pour multiplier `R1` par 10, et à l'appel système `IN` pour effectuer des saisies de caractères, il est important de sauvegarder l'adresse de retour sur la pile en début de routine, puis de la restaurer en fin de routine. Veillez à bien commenter la condition de sortie de boucle.

```
; routine saisie : pour saisir un entier naturel décimal.
; paramètre résultat : R1
; registres temporaires (écrasés par l'appel) : R0, R2
saisie:   ..............  ; ......................................
          ..............  ; ......................................
          ..............  ; ......................................
          ..............  ; ......................................
          ..............  ; ......................................
```

```
            ..................  ; ....................................
            ..................  ; ....................................
            ..................  ; ....................................
            ..................  ; ....................................
            ..................  ; ....................................
            ..................  ; ....................................
            ..................  ; ....................................
            ..................  ; ....................................
            ..................  ; ....................................
            ..................  ; ....................................
            ..................  ; ....................................
            ..................  ; ....................................
            ..................  ; ....................................
            ..................  ; ....................................
    ; Constante pour la routine saisie :
    carzero:  .FILL 48       ; code ascii du caractère '0'
```

3. (1 point) Si on regarde le contenu du registre `R1` comme un entier naturel, la routine `aff` doit, lorsqu'elle est appelée, afficher `R1` étoiles à l'écran. Proposez un pseudo-code pour la routine `aff` (contentez-vous appeler pour l'instant `printchar('*')` pour indiquer l'affichage d'une étoile).

```
routine aff(paramètre donnée : R1) {
  ..................................................................
  ..................................................................
  ..................................................................
  ..................................................................
  ..................................................................
  ..................................................................
  ..................................................................
}
```

4. (2 points) Complétez la routine `aff` ci-dessous, en traduisant le pseudo-code que vous avez donné à la question précédente. Vous allez faire des appels à `OUT` pour afficher des caractères : veillez donc à sauvegarder l'adresse de retour sur la pile en début de routine, puis de la restaurer en fin de routine. Veillez aussi à bien commenter la condition de sortie de boucle.

```
; routine aff : pour afficher n étoiles.
; paramètre donnée : R1
; registre temporaire (écrasé par l'appel) : R0, R2
aff:    .................. ; ....................................
        .................. ; ....................................
        .................. ; ....................................
        .................. ; ....................................
        .................. ; ....................................
        .................. ; ....................................
        .................. ; ....................................
        .................. ; ....................................
        .................. ; ....................................
        .................. ; ....................................
        .................. ; ....................................
        .................. ; ....................................
        .................. ; ....................................
        .................. ; ....................................
        .................. ; ....................................
; Constante pour la routine aff :
caretoile:  .FILL 42       ; code ascii du caractère '*'
```

4 Somme des éléments d'un tableau d'entiers (4pts)

On veut ajouter les entiers compris entre les adresses `debut` et `fin` à l'aide d'une boucle (les entiers aux adresses `debut` et `fin` inclus), et placer le résultat de la sommation à l'adresse `resultat`. Il vous est demandé de commencer par donner un pseudo-code pour votre programme, en respectant les directives d'utilisation des registres données ci-dessous, avant de le traduire en langage d'assemblage.

```
           .ORIG x3000     ; adresse de début de programme
; partie dédiée au code
; R0 sera utilisé comme un pointeur pour une case du tableau
; R1 contiendra l'opposé de l'adresse fin, pour comparaison avec R0
; R2 servira à l'accumulation des entiers du tableau
; R3 servira de registre temporaire
           ### A COMPLETER ###
; partie dédiée au résultat et aux données
resultat:  .BLKW #1        ; espace pour stocker le résultat (résultat attendu ici : 157 soit x9D)
debut:     .FILL #12
           .FILL #45
           .FILL #16
           .FILL #06
fin:       .FILL #78
           .END
```

1. (1 point) Donnez le pseudocode de votre programme.

   .....

   .....

   .....

   .....

   .....

   .....

   .....

   .....

   .....

   .....

   .....

2. (3 points) Donnez le code de votre programme en langage d'assemblage du LC3. Veillez à bien commenter la condition de sortie de boucle.

   .....

   .....

   .....

   .....

   .....

   .....

   .....

   .....

   .....

   .....

   .....

   .....

   .....

   .....