CC-rattrapage – LIF6

Mathilde Noual – Nicolas Louvet

Aucun document autorisé. Les calculatrices, ordinateurs, téléphones et autres sont interdits.
Durée 1h00. Le barème est susceptible de modifications.

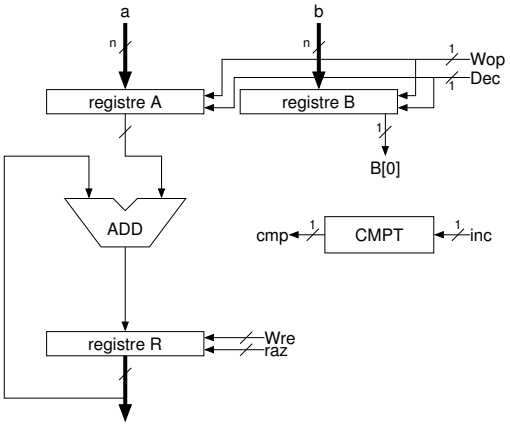
I Multiplieur séquentiel d'entiers naturels

Le but de cet exercice est d'ébaucher un circuit séquentiel permettant d'effectuer le produit de deux entiers naturels a et b codés dans leur représentation positionnelle binaire sur n bits.

Q.I.1) - Posez à la main la multiplication $a = (0011)_2$ par $b = (0111)_2$.

Q.I.2) - Justifez le fait que le produit $a \times b$ peut comporter jusqu'à $2n$ bits, et que $2n$ bits suffisent pour représenter le produit $a \times b$.

On considère l'ébauche de circuit ci-dessous :



Le circuit comporte trois registres :

- les registres à décalages A et B :
  - si au cours d'un cycle le signal Dec est activé, alors en fin de cycle le contenu de A subit un décalage à gauche, et le contenu de B subit un décalage à droite ;
  - si Wop est activé, les entrées a et b sont écrites dans les registres en fin de cycle ;
  - le signal B[0] donne le bit de poids faible de B.
- le registre résultat R, qui contiendra au final le produit $a \times b$:
  - si au cours d'un cycle le signal Wre est activé, alors en fin de cycle le registre R est écrit ;
  - si raz est activé, le registre R est remis à zéro.

Le circuit comporte un aditionneur ADD et un compteur CMPT :

- si au cours d'un cycle le signal inc est activé, le compteur est incrémenté ;
- si le compteur a atteint sa valeur maximale au cours d'un cycle, alors sa sortie cmp s'active immédiatement, et il repasse à zéro au début du cycle suivant ;
- dans la suite CMPT désignera la valeur du compteur.

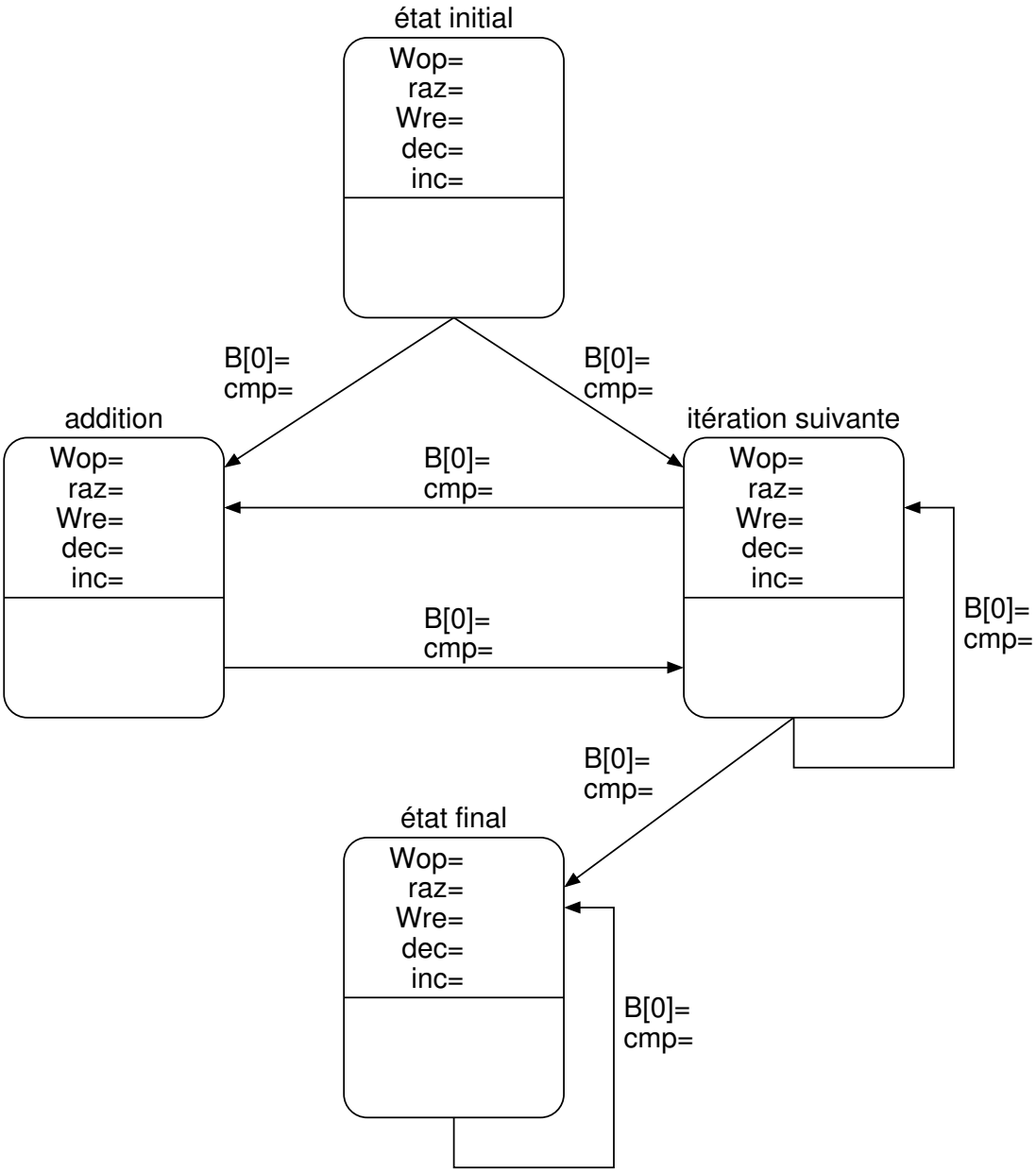
Le pseudo-code suivant écrit d'une manière encore assez informelle le fonctionnement du multiplieur :

```
A := a; B := b; R := 0
faire
  Si B[0] = 1 alors R := R + A; Fin si
  Si CMPTR = max alors passer en état final; Fin Si
  A = A<<1; B = B<<1; CMPTR = CMPTR + 1
fin faire
```

Q.I.3) - Complétez le circuit du multiplieur en indiquant la taille de chacun des vecteurs de signaux pour lequel l'information n'est pas encore donnée.

Q.I.4) - Combien de fois la partie faire du pseudo code doit-elle être répétée ? Quelle valeur doit prendre max ? Si $n = 2^k$ (k entier naturel, $k \geq 0$), quelle taille doit-on donner au compteur ?

Q.I.5) - Complétez le diagramme de l'automate de contrôle ci-dessous pour notre multiplieur séquentiel. Indiquez pour chaque état quelles sont les actions algorithmiques éffectuées.



Voir le cours d'Olivier Temam.

II Décompte de bits non-nuls

Ecrire une fonction dans l'assembleur du LC3 afin de compter le nombre de bits non-nuls dans un entier. Votre programme principal doit appeler votre fonction afin de compter le nombre de bits non-nuls de l'entier à l'adresse `n`, et placer le résulat à l'adresse `r`. Vous compléterez le code ci-dessous, en le commentant.

```
        .ORIG x3000
; Programme principal
        ...
; Données
n:      .FILL #78
r:      .BLKW #1

; Sous-routine pour calculer le nombre de bit non-nuls dans R1
; paramètre : R1, inchangé
; retour : R0, le nombre de bits non nuls dans R1
; registres temporaires : R2, R3, R4
```

```
cmptbits: ...
.END
```

On va implanter l'algorithme ci-dessous :

```
; R0=0
; R2=R1
; while(R2 != 0) {
;   if(R2 & x8000) c++;
;   n=n<<1;
; }
.END
```

Voici le résultat :

```
        .ORIG x3000
; Programme principal
        LD R1,n
        JSR cmptbits
        ST R0,r
        HALT

; Données
n:       .FILL #78
r:       .BLKW #1

; Sous-routine pour calculer le nombre de bit non-nuls dans R1
; paramètre : R1, inchangé
; retour : R0, le nombre de bits non nuls dans R1
; registres temporaires : R2, R3, R4
cmptbits:  AND R0,R0,#0  ; Mise à 0 du compteur
           LD R4,cste    ; Mise à 1 du bit de poids fort de R4
           ADD R2,R1,#0  ; On recopie R1 dans R2
boucle:    BRz finboucle ; Ici, NZP sont à jour d'après R2
           AND R3,R2,R4  ; On teste le bit de poids fort de R2
           BRz bitnul    ;   si le bit est nul, rien à faire
           ADD R0,R0,#1  ;   sinon, on incrémente le compteur
bitnul:    ADD R2,R2,R2  ; Décalage de R2 d'un bit à gauche
           BR boucle
finboucle: RET
cste:      .FILL x8000
```