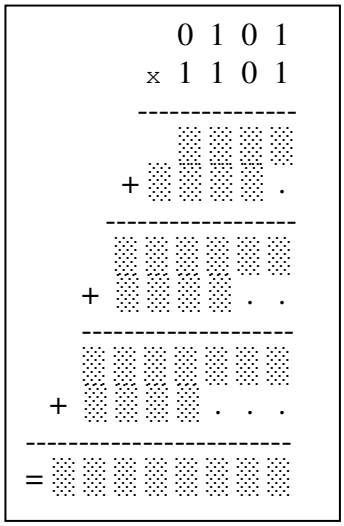
Université Joseph Fourier – Grenoble 1
Architecture des ordinateurs - MIAGE L3

Examen d'Architecture des ordinateurs - première session

Décembre 2014

Durée : 3h00 en tout (~1h00 Architecture des ordinateurs, ~1h Système, ~1h Réseau).
1 document A4 commun ASR autorisé. Sujet Architecture des ordinateurs sur 1 recto-verso.

1. Circuit combinatoire pour la multiplication
(barème indicatif sur 7 points : 3 points)



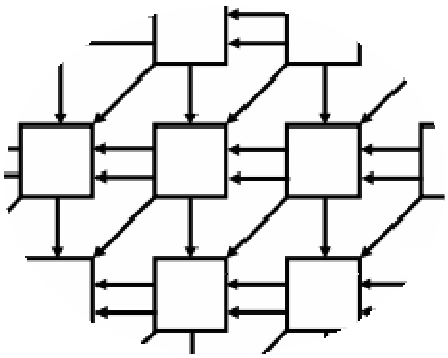
Q1. Effectuez la multiplication 0101 x 1101 (en binaire), comme à l'école primaire, avec un algorithme naïf, c'est à dire :

- en notant tous les bits de résultat intermédiaires (même les 0),
- en calculant toutes les multiplications individuelles (même celle par 0) et
- en procédant aux additions le plus tôt possible (ne pas attendre d'avoir 4 opérandes sur 4 bits à additionner. Dès que vous avez 2 opérandes, même sur 1 bit, faites l'addition.)

Votre multiplication aura donc la forme ci-contre.
(vous pouvez répondre sur le sujet en remplissant les « ▒ »)

Q2. L'algorithme naïf employé permet d'isoler une opération sur 1 bit complexe, i.e. combinaison d'un transfert ou décalage de données sur 1 bit, d'une multiplication 1 bit par 1 bit et d'une addition sur 1 bit. Opération complexe mais unique à effectuer tout au long de la multiplication.

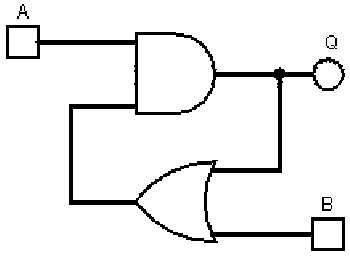
On retrouve cette opération dans les schémas de multiplieurs combinatoires (cf. l'extrait d'un dessin de multiplieur combinatoire ci-contre). Cette cellule est appelée cellule de multiplication.



Identifiez les entrées et les sorties de cette cellule combinatoire de multiplication, donner sa table de vérité et son schéma logique.

Donnez le circuit global pour une multiplication combinatoire sur 4 bits (c'est-à-dire : 4bits x 4 bits ➔ 8 bits) utilisant cette cellule de multiplication.

2. Cellule de mémorisation ?
(barème indicatif sur 7 points : 2 points)



Q1. Analysez le schéma ci-contre. Identifiez les entrées et les sorties potentielles.

Est-ce un circuit capable de mémoriser un bit ? (ou un circuit combinatoire, ou un circuit oscillant, ou autre chose encore ?)

Si c'est un circuit capable de mémoriser un bit, précisez comment fixer « ce bit » à 0, ou à 1 et quelle configuration des entrées conserve l'état précédent ?

Sinon, quel est son comportement ?

Dans tous les cas comparez son comportement aux comportements des circuits similaires connus.

3. Automate de contrôle
(barème indicatif sur 7 points : 2 points)

Une machine à accumulateurs est donnée par un automate d'interprétation dont une partie est donnée ci-après.
Dans cette machine les instructions peuvent être données sur 1 octet ou 2 octets. Dans les deux cas, le premier octet (stocké dans le registre IR lors de l'exécution de l'automate de contrôle) comporte le code de l'instruction sur les 4 bits de poids fort (noté IR::I dans la suite), un bit pour indiquer un registre accumulateur à utiliser (noté IR::A dans la suite) et un numéro de registre (noté IR::R dans la suite) sur les 3 bits de poids faible. Le premier octet peut donc être représenté par IR = [IR::I ; IR::A, IR::R]. Le second octet (quand il existe), comporte un argument sur 8 bits (valeur immédiate, adresse, …).
Cette machine possède une mémoire (noté MEM dans la suite), 2 accumulateurs et 8 registres banalisés. La donnée stockée en mémoire à une adresse fournie par un registre R peut être notée MEM[R]. Les 2 registres accumulateurs (ACC0 et ACC1) peuvent être notés Acc(N) où N vaut 0 ou 1. Les registres banalisés peuvent être notés Reg(N) où N est un nombre entre 0 et 7.

Format des instructions dans IR :

Bits : 7 6 5 4 3 2 1 0

IR::I	IR::A	IR::R

Mémoire :

Adresse	Contenu (en hexadécimal)
00	00
01	19
02	FF
03	49
04	00
05	00
…	…
FE	00
FF	00

Les instructions disponibles dans l'automate de contrôle donné (ci-après) sont au nombre de 2. Une instruction d'addition avec un accumulateur (avec 2 mode d'adressages : par registre, ou avec une valeur immédiate) et une instruction de saut conditionnel.

Le code de l'addition par registre est 0, celui de l'addition avec valeur immédiate est 1, celui du saut est 4.

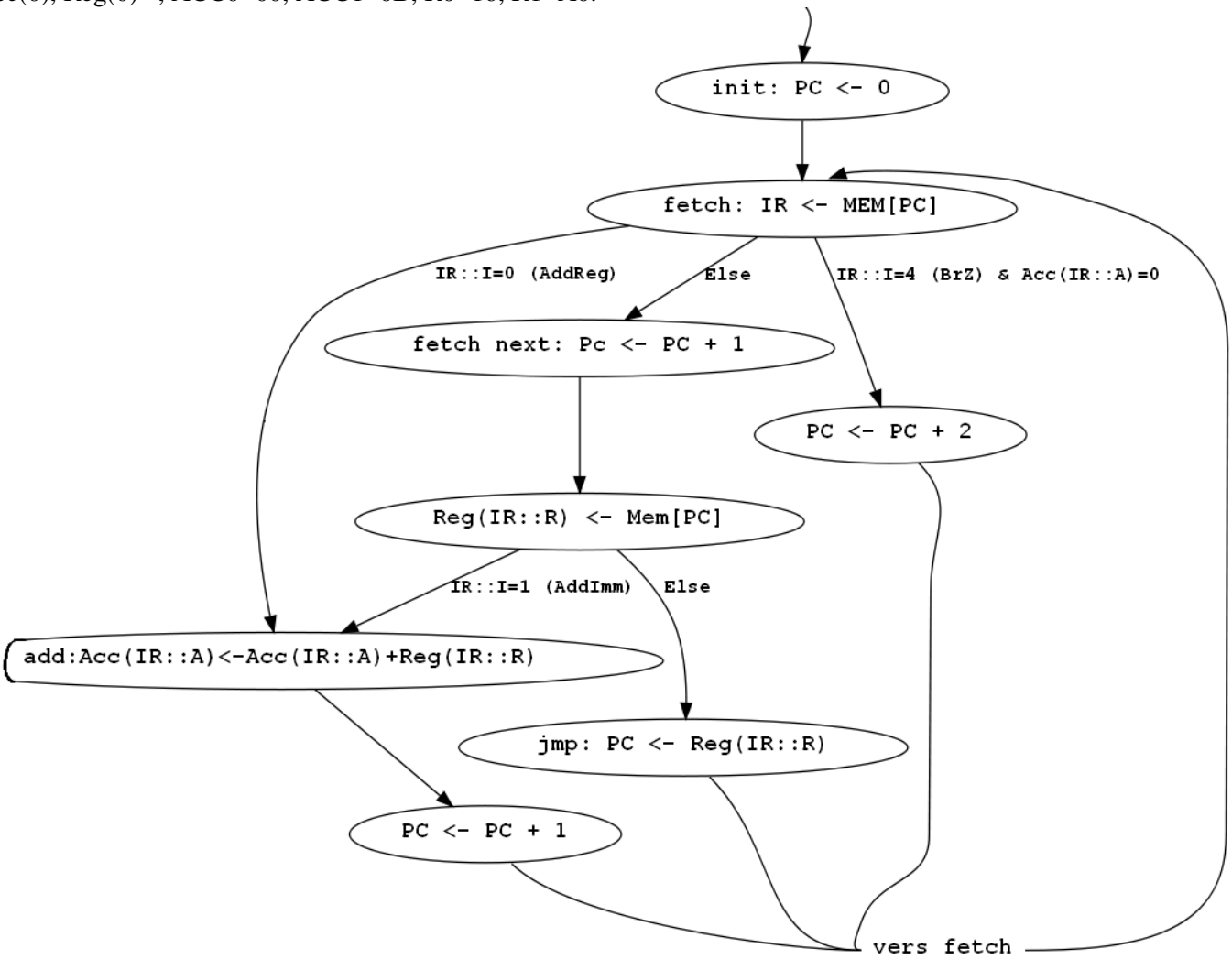
L'instruction d'addition par registre est codée sur 1 octet, ex. : 03 = (en binaire 0000 0011) = [0 ; 0,3], c'est-à-dire : « AddReg Acc(0), Reg(3) ».

L'addition avec valeur immédiate est codée sur 2 octets, ex. : 1A suivi de la valeur immédiate FD = (en binaire 0001 1010) suivi de FD = [1 ; 1,2] suivi de FD, c'est-à-dire : « AddImm Acc(1) Reg(2) » suivi de FD.

Le saut conditionnel est codé sur 2 octets (ex. : 40 suivi de 00 = (en binaire 1000 0000) suivi de 00 = [4 ; 0,0] suivi de 00, c'est-à-dire : « BrZ, Acc(0) Reg(0) » suivi de 00.

L'automate de contrôle donne le détail de l'exécution de ces instructions.

Q1. Simulez l'exécution des 4 premières instructions du programme donné en mémoire (cf. ci-après). Indiquez pour chaque état, quand elles changent, les valeurs de PC, IR, Acc0, Acc1, Reg0, Reg1. Vous supposerez que la machine, à l'état initial, est dans la configuration suivante (en hexadécimal) : PC=00, IR=00=[0 ; 0,0], c'est-à-dire « AddReg Acc(0), Reg(0) », ACC0=00, ACC1=0B, R0=16, R1=A0.



Q2. Modifiez l'automate pour permettre une addition avec un accumulateur et une donnée en mémoire désignée par son adresse. Indiquez un codage possible pour cette instruction et les modifications de l'automate de contrôle.

Éléments de corrections

Exercice 1. Multiplication

La multiplication de 5 par 13 donne bien 65 :

```
        A : 0 1 0 1
        B : 1 1 0 1
        ---------------
          0 1 0 1
        + 0 0 0 0 .
        ------------------
          0 0 1 0 1
      +   0 1 0 1 . .
      --------------------
      0 0 1 1 0 0 1
    + 0 1 0 1 . . .
  -------------------------
= 0 1 0 0 0 0 0 1
```

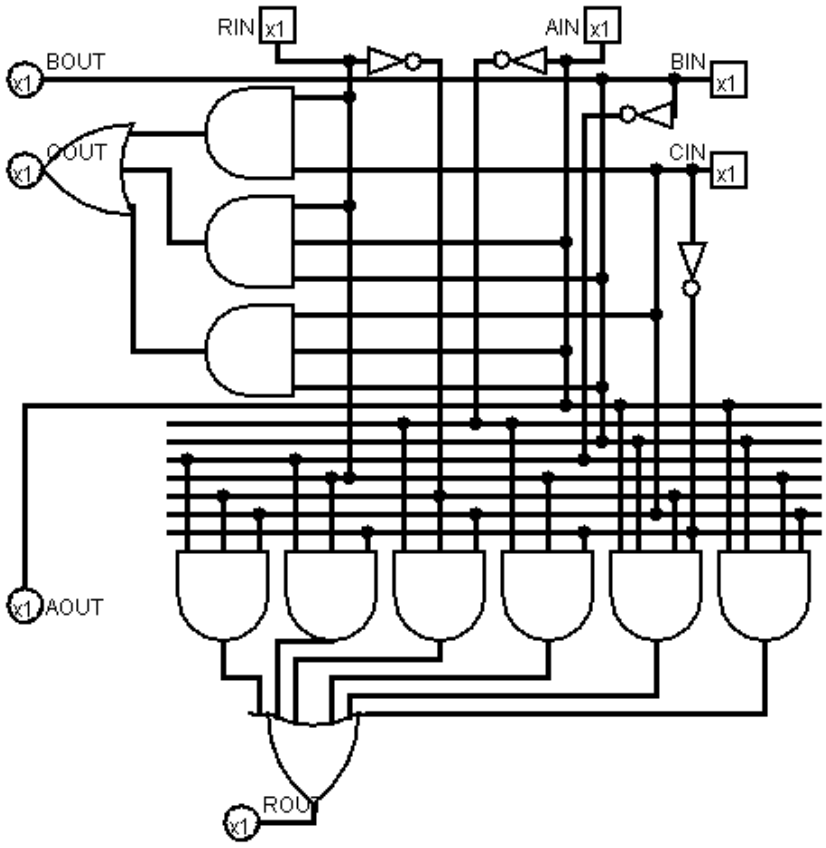
Après analyse, on peut imaginer avoir 4 entrées et 4 sorties : 2 pour des transits, 2 pour l'addition-couplée-multiplication.

Dans le détail :
A : transite en diagonal, i.e. : $A_{out} = A_{in}$
B : transite à l'horizontal, i.e. : $B_{out} = B_{in}$
(C_{out}, R_{out}) : $R_{in} + A*B + C_{in}$

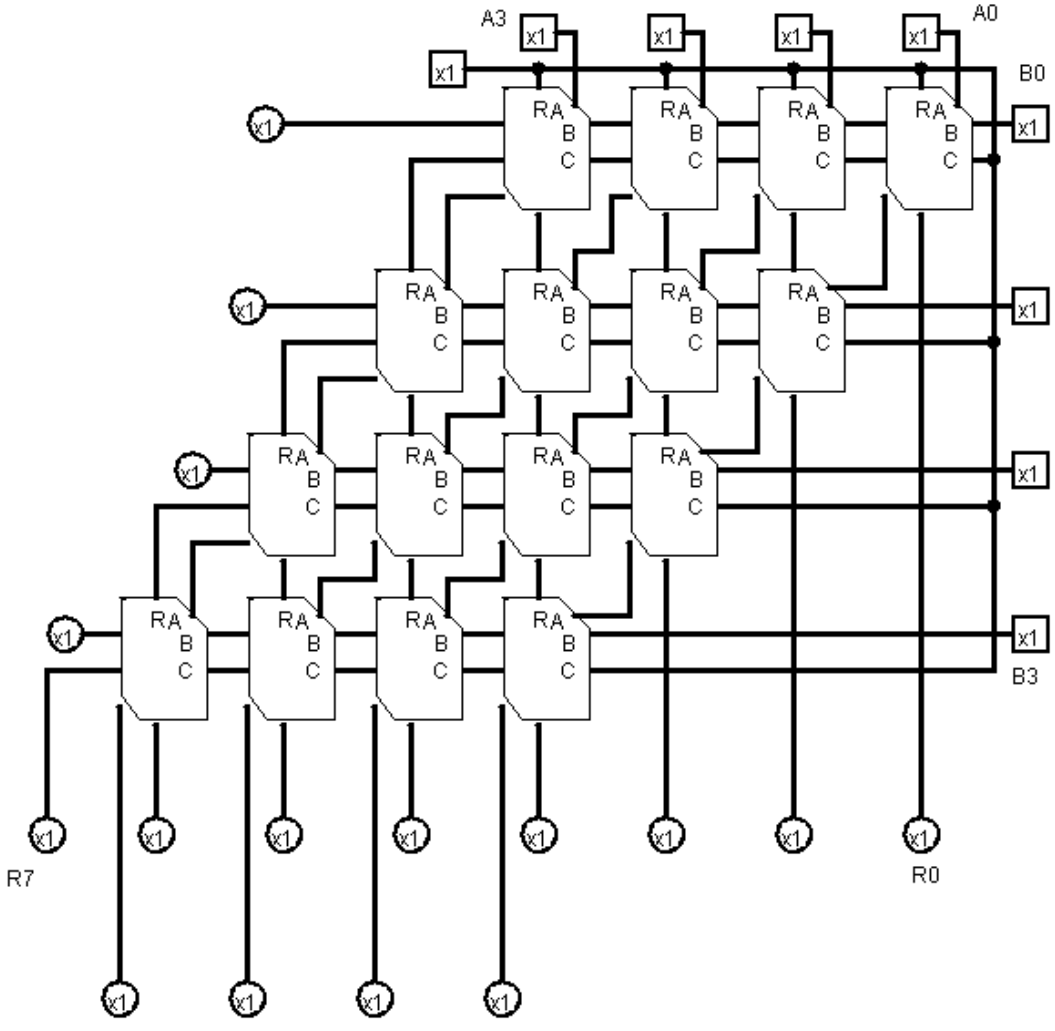
Ce qui donne la table de vérité :

A	B	R_{in}	C_{in}	C_{out}	R_{out}
0	0	0	0	0	0
0	0	0	1	0	1
0	0	1	0	0	1
0	0	1	1	1	0
0	1	0	0	0	0
0	1	0	1	0	1
0	1	1	0	0	1
0	1	1	1	1	0
1	0	0	0	0	0
1	0	0	1	0	1
1	0	1	0	0	1
1	0	1	1	1	0
1	1	0	0	0	1
1	1	0	1	1	0
1	1	1	0	1	0
1	1	1	1	1	1

Le circuit correspondant, un peu simplifié, est donné par :



En assemblant 16 cellules, on obtient le circuit de multiplication 4 bits suivants :



Exercice 2. Cellule de mémorisation

A	B	Q
0	0	0
0	1	0
1	0	Sauvegarde d'un bit (Q)
1	1	1

Conclusion :
A = 0 ➔ Q = 0 (reset)
A=B=1 ➔ Q = 1 (set)
A=1, B=0 ➔ conservation de la valeur précédente de Q.

Exercice 3. Automate de contrôle

L'exécution s'effectue en une dizaine de temps :

t	PC	IR	AC0	AC1	R0	R1
0	0	0	0	0B	16	A0
1	0					
2		0				
3			16			
4	1					
5		19				
6	2					
7						FF
8				0A		
9	3					
10		49				
11	4					00
12	0					
13		0				
14			2C			
15	1					

Pour modifier l'automate afin d'accepter une donnée en mémoire, il faut ajouter un Reg(IR ::R)←Mem[Reg(IR ::R)] à la suite de Reg(IR ::R)←Mem[PC] et menant à add, en différenciant le cas I=1 et le cas Else pour mener à cet état avec un troisième cas qui dépendra du code de l'instruction, par exemple en prenant I=2 (le reste de l'instruction pourra être comme le add immédiat, à la différence que l'octet suivant l'instruction comportera l'adresse de la donnée au lieu de la donnée).