Université Joseph Fourier – Grenoble 1
ALM MIAGE L3
Examen première session
Décembre 2009
Durée : 1h30, tout document manuscrit et calculette autorisés
Sujet sur 1 recto-verso.

I. Synthèse de fonctions par comparaison (barème indicatif : 6 points)

Q1. Donner la table de vérité de la fonction $Sup_{\geq 5}$ définie pour les entiers sur 32 bits à valeur booléenne qui dit si un nombre est supérieur ou égal à 5. $Sup_{\geq 5}$: x → x ≥ 5.
Dessiner un circuit réalisant cette fonction (ou une partie significative).

Q2. En supposant que deux circuits sont donnés pour réaliser respectivement $Sup_{\geq 5}$ et $Sup_{\geq 10}$, définir un circuit réalisant la fonction $Appartient_{[5...9]}$ pour les entiers sur 32 bits à valeur booléenne telle que $Appartient_{[5...9]}$: x→ x ∈ [5…9]

Q3. En supposant que des circuits sont donnés pour réaliser $Sup_{\geq n}$ quel que soit n, définir un circuit réalisant la fonction $Appartient_{[5...9]\cup[14...27]}$ pour les entiers sur 32 bits à valeur booléenne telle que $Appartient_{[5...9]\cup[14...27]}$: x→ x ∈ [5…9]∪[14...27]

Q4. Pour une fonction F quelconque, décrire une méthode pour obtenir un circuit réalisant F à partir de circuit de comparaison $Sup_{\geq n}$.
Appliquer votre méthode pour définir un circuit réalisant la fonction Log_2Log_2 qui donne pour un entier x sur 32 bits le nombre de chiffres de l'écriture en base 2 du nombre de chiffre de l'écriture en base 2 de x.
Exemple : pour x = 1 milliard, x a une écriture en base 2 qui nécessite 29 chiffres, 29 a une écriture en base 2 qui nécessite 5 bits, donc Log_2Log_2 (x)=5.

II. Comparaison (barème indicatif : 6 points)

Q1. Donner la table de vérité de la fonction binaire de comparaison ($Sup_{\geq}$(A,B)) pour des entrées A, B sur 1 bit : $Sup_{\geq}$(A,B) = A≥B.

Q2. A partir de ce qui précède, en supposant que vous disposez de circuits $Sup_{\geq,1bit}$ et en s'inspirant de l'un des algorithmes ci-après proposer un circuit combinatoire qui réalise la comparaison entre 2 nombres de 32 bits.

Algorithme de comparaison : pour A, B deux entiers représentés sur n=31 bits (A_n, ... A_0), (B_n, ... B_0)
Formulation récursive :
$Sup_{\geq}$(A,B,n) : si n=0 alors retourne $A_0 \geq B_0$
sinon si $A_n=B_n$ alors retourne $Sup_{\geq}$(A,B,n-1)
sinon retourne $A_n \geq B_n$.
Appel : $Sup_{\geq}$(A,B,32).

Formulation itérative :
$Sup_{\geq}$(A,B) : i=31 ; tant que i>0 et $A_i=B_i$ faire : i ← i-1 ; fin tant que ; retourne $A_i \geq B_i$.

Q3. A partir de ce qui précède proposer un circuit séquentiel qui réalise la comparaison entre 2 nombres sur 32 bits.

Q4. Comparer les 2 circuits obtenus.

III. Instruction Add à 3 adresses (barème indicatif : 8 points)

Prenons la machine vue en cours. Cette machine possède le langage machine suivant :

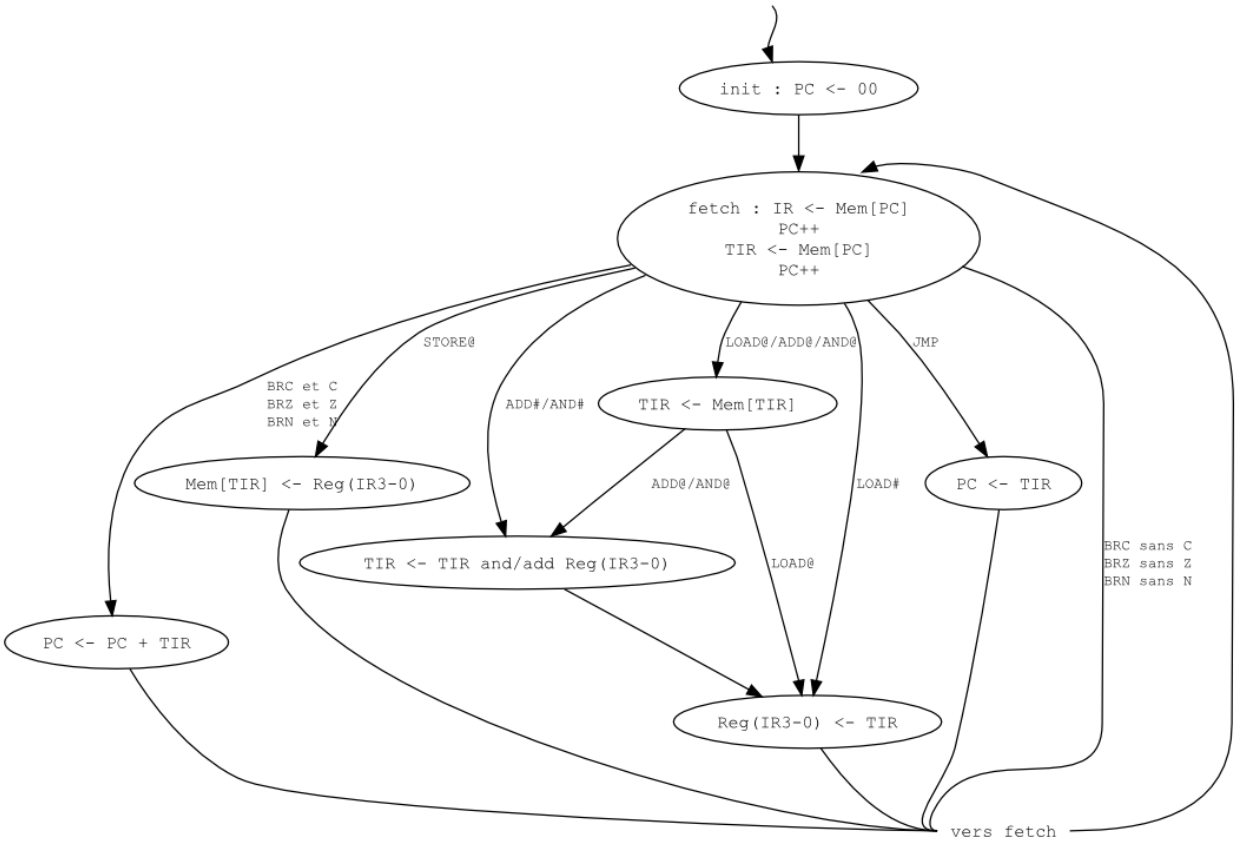
- LOAD#/ADD#/AND# en mode d'adressage immédiat, codage sur 2 octets, avec resp. les codes binaires 1, 0, 2 dans le quartet de poids fort du premier octet de l'instruction, le numéro du registre paramètre de l'instruction dans le quartet de poids faible de cet octet et la valeur immédiate dans le second octet de l'instruction
- LOAD@/STORE@/ADD@/AND@ en mode d'adressage direct, codage sur 2 octets, avec resp. les codes binaires 5, 7, 4, 6 dans le quartet de poids fort du premier octet de l'instruction, le numéro du registre paramètre de l'instruction dans le quartet de poids faible de cet octet et l'adresse de l'opérande dans le second octet de l'instruction

- et divers sauts JMP/BRC/BRZ/BRN en adressage absolu ou relatif, codage sur 2 octets, avec resp. les codes binaires C0, C1, C9, CD dans le premier octet de l'instruction et l'adresse du saut ou la valeur de décalage dans le second octet de l'instruction

Rappel de quelques instructions :

Mode d'adressage	**Mnémo**	**Codage binaire**	**Interprétation**
Immédiat	LOAD Reg, #i	1(N°Reg) puis i	Reg ←i, PC ← PC + 2
Absolu	JMP @ad	C0 puis ad	Pc ←ad

Rappel de l'automate de contrôle pour l'interprétation de ce langage machine :



Cet exercice porte sur l'introduction de nouvelles instructions assembleurs dans le langage interprété par l'automate de contrôle de cette machine.
Les instructions en question sont des instructions permettant de faire la somme de deux registres, le résultat étant stocké dans un troisième registre. Le mnémonique choisi pour cette instruction est Ad3. Dans une première version, les trois registres sont implicitement A, B, C et l'interprétation de l'instruction « $Ad3_1$ » en pseudo langage est le suivant :
A←B+C ; Pc←Pc+1 (ou 2 selon la taille de l'instruction)
Q1. Proposer un codage pour cette instruction $Ad3_1$, code binaire de l'instruction, éventuellement nombre d'opérandes, place des opérandes
Q2. Modifier le graphe de contrôle pour prendre en compte cette instruction.
Seconde version, les trois registres doivent pouvoir être définis dans l'instruction « $Ad3_2$ »
Q3. Proposer un codage pour cette instruction $Ad3_2$, code binaire de l'instruction, éventuellement nombre d'opérandes, place des opérandes
Q4. Modifier le graphe de contrôle pour prendre en compte cette instruction.
Q5. Donner le temps d'exécution des instructions $Ad3_1$ et $Ad3_2$ (discuter d'éventuelles hypothèses à prendre sur la mise en œuvre du graphe de contrôle)
Q6. Ces instructions assembleurs pouvaient-elles être simulées avec l'ensemble d'instructions disponibles pour la machine initiale ? Si oui, comparer les mises en œuvre.

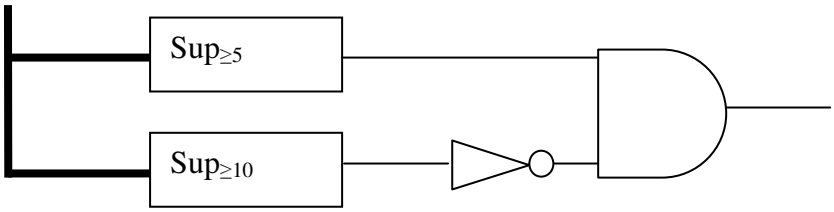
Université Joseph Fourier – Grenoble 1
ALM MIAGE L3
Examen première session - Décembre 2009 - Durée : 1h30
Eléments de solution

I. Synthèse de fonctions par comparaison

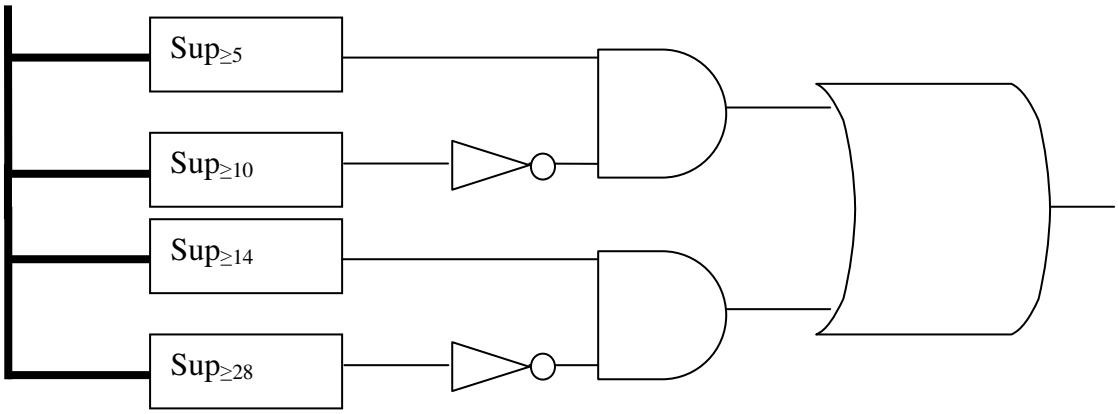
Q1. Table de vérité de la fonction $Sup_{\geq 5}$: x → x ≥ 5.

N	$Sup_{\geq 5}(N)$
00…0000	0
00…0001	0
00…0010	0
00…0011	0
00…0100	0
00…0101	0
00…0110	1
00…0111	1
…	…

Q2. Circuit pour $Appartient_{[5…9]}$: x→ x ∈ [5…9]



Q3. Circuit pour $Appartient_{[5…9]\cup[14…27]}$: x→ x ∈ [5…9]∪[14...27]



Q4. Méthode pour obtenir un circuit réalisant F à partir de circuit de comparaison $Sup_{\geq n}$: c'est une généralisation de ce qui précède à un nombre quelconque d'intervals (où la fonction F est vraie).
Pour Log_2Log_2 : le résultat peut valoir 1, .., 5 :

N	$Log_2Log_2(N)$
Pour 0 et 1	1
De 10 à 111	2
De 1 000 à 1 111 111	3
De 10 000 000 à 111 111 111 111 111	4
Au delà	5

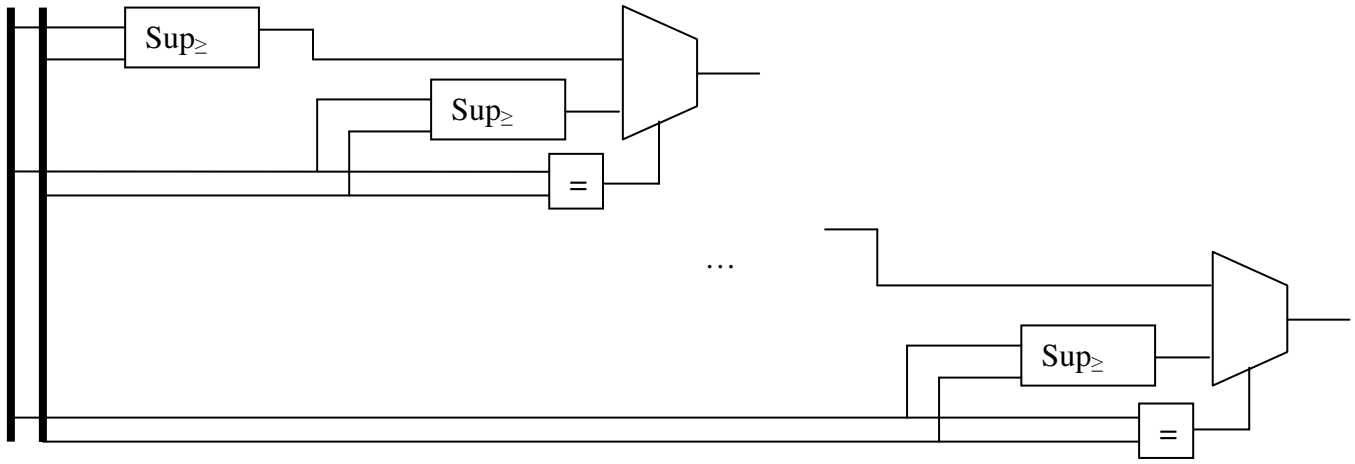
D'où il suit, une fonction dont le résultat est sur 3 bits, par exemple pour le bit de poids faible (vrai si le résultat est impair) : $Appartient_{[0…1]\cup[1\,000…1\,111\,111]\cup[1\,000\,000\,000\,000\,000…11\,111\,111\,111\,111\,111\,111\,111\,111\,111]}$

II. Comparaison

Q1. Table de vérité de la fonction binaire de comparaison :

A	B	$Sup_{\geq}(A,B) = A \geq B$
0	0	1
0	1	0
1	0	1
1	1	1

Q2. Circuit combinatoire de comparaison. Il est constitué de 32 couches, la première ne comporte que $Sup_{\geq}$, les 31 autres comportent un circuit '$Sup_{\geq}$', un circuit '='et un multiplexeur, la $i^{ème}$ couche a en entrée les fils A_i et B_i et la sortie de la couche précédente.



Q3. Circuit séquentiel de comparaison. Il s'agit de boucler la couche répétée 31 fois de la question 2 avec 1 registre 1 bit pour le résultat courant, et 2 registres 32 bits pour A et B pour lesquels dans la boucle il y a un décaleurs de telle sorte que l'entrée de la couche prenne toujours en entrée A_0 et B_0 mais que, avec les décaleurs, A_0 et B_0 soient au $i^{ème}$ top d'horloge les valeurs A_i et B_i des A et B initiaux.

III. Instruction Add à 3 adresses

Q1. Codage de $Ad3_1$: à priori il n'y a pas d'opérande, le code de l'instruction peut tenir sur 1 octet (même moins), le code 30 est disponible. Le 0 pourrait être remplacée par une valeur constante à additionner en plus.

Q2. Ajout dans le graphe : entre les deux premières micro-instructions du fetch et les suivantes, on peut ajouter une transition si le code 30 de $Ad3_1$ a été observé, il même à l'état A←B+C à partir duquel une transition ramène à fetch.

Q3. Codage pour $Ad3_2$: ici il y a 3 opérandes, chacun étant un registre, 4 bits suffisent pour donner le numéro du registre. Avec 2 octets pour le codage de l'instruction complète on peut prendre 3R puis IJ ou I, J sont les numéros des registres à additionner et R le numéros du registre où est stocké le résultat.

Q4. Ajout dans le graphe : après le fetch (complet), on peut ajouter un état qui fait l'addition : $Reg(IR_{3\text{-}0}) \leftarrow Reg(TIR_{3\text{-}0}) + Reg(TIR_{7\text{-}4})$, puis de cet état on ajoute une transition vers Fetch.

Q5. $Ad3_1$ s'exécute en 3 tops d'horloge + le temps de décryptage de l'instruction. Pour $Ad3_2$, il faut 2 temps de plus pour finir le fetch et beaucoup de temps pour obtenir $Reg(IR_{3\text{-}0})$, $Reg(TIR_{3\text{-}0})$, $Reg(TIR_{7\text{-}4})$ via les labus, lbbus, lcbus.

Q6. Simulation en assembleur, pour $Ad3_1$, en utilisant l'adresse 81 pour stocker temporairement des valeurs :

```
Load# A, 0
Store@ B, @81
Add@ A, @81
Store@ C, @81
Add@ A, @81
```