CORRIGE Examen Architecture des Ordinateurs – 17 Décembre 2015

Tous documents autorisés. Calculatrices interdites. Durée 2h

Pour toutes les questions, une explication **concise** est nécessaire pour que la réponse soit prise en compte. La représentation demandée est impérative.

1. Flottants [5 pts]

Q1. Donner la valeur (représentation **décimale**) du nombre réel représenté par 0x40880000 en format IEEE754 simple précision.

0x40880000 représente 4,25.
En effet 0x40880000 = 0b01000000100010…0
s = 0 e = E-127= 128+1-127=2 $f = 2^{-4}$
Donc $x = 2^2(1+2^{-4}) =$ **4,25**

Q2. Coder -5,75 (représentation **hexadécimale**) en format IEEE754 simple précision.

$5,75 = 2^2 + 1 + 2^{-1} + 2^{-2} = 2^2 (1 + 2^{-2} + 2^{-3} + 2^{-4})$

E = 127 +2 = 128 +1 = 0b10000001

Représentation : 0b 11000000101110…0 = **0xC0B80000**

Q3. Donner la représentation hexadécimale du plus petit nombre strictement supérieur à 1 représentable.

$1 = 2^0$
E = 127 $f = 2^{-23}$
Représentation 0b0011111110…01 = **0x3F800001**

Q4. Le résultat de l'opération $1,25x2^{-126} - 1x2^{-126}$ est-il représentable en format IEEE754 simple précision ? Une justification précise en deux lignes maximum est demandée.

Le résultat est $0,25x2^{-126}$ qui est égal à 2^{-128}, ou encore $2^{-127}x2^{-1}$.
Ce nombre n'est pas représentable en normalisé, car -126 ≤ e ≤ 127. Mais il est représentable en **dénormalisé** : E=0 ; e = -127 et $f = 2^{-1}$.

Q5. Quel est le résultat (représentation mathématique) de l'opération $(2^{-10} + 2^{20}) - 2^{20}$ effectuée dans l'additionneur flottant IEEE754 **double** précision ?

En double précision, la partie fractionnaire est sur 52 bits, donc le décalage ne produit pas d'arrondi pour l'addition $(2^{-10} + 2^{20})$. Le résultat est donc le résultat exact, 2^{-10}.

2. Fonctions booléennes et circuits combinatoires [5 pts]

Soit la fonction booléenne F des variables X_3, X_2, X_1, X_0 définie par la table de vérité de la table 1.

Q6. Donner l'écriture de F sous forme disjonctive normale (somme de minterms) et sous forme conjonctive normale (produit de maxterms).

F = m0 + m1 +m4 +m5 + m6 + m8 + m9 + m10 + m11 + m12 + m13 = M2.M3.M7.M14.M15
Avec par exemple
m1 = !X3. !X2. !X1.X0 et
M2 = X3 + X2 + !X1 + X0

C'est la somme des minterms pour lesquels la fonction F vaut 1 et le produit des maxterms pour lesquels elle vaut 0.

Q7. Donner une forme réduite de F ; justifier par un diagramme de Karnaugh.

De façon générale, les minterms pour un diagramme de karnaugh à 4 variables sont répartis ainsi :

	!X3		X3		
!X2	m0	m2	m10	m8	!X0
	m1	m3	m11	m9	
X2	m5	m7	m15	m13	X0
	m4	m6	m14	m12	!X0
	!X1	X1		!X1	

Pour la fonction F, on obtient :

	!X3		X3		
!X2	*		*	*	!X0
	*		*	*	
X2	*			*	X0
	*	*		*	!X0
	!X1	X1		!X1	

F = !X1 + X3. !X2 + !X3.X2.!X0

Q8. Implanter la fonction logique F avec un multiplexeur 8 entrées 1 sortie et un inverseur en complétant la figure 1. Les entrée 0 et 1 sont également disponibles ; si une entrée du multiplexeur doit être connectée à 0 ou à 1, on notera un 0 ou un 1 sur cette entrée.

La table 1 complétée donne les sorties du multiplexeur attendues pour chaque configuration de X_3, X_2, X_1. Et la figure 2 montre les connexions. NB : l'attribution correcte des **lignes de commande C0, C1, C2** est essentielle.

Q9. On veut implanter la fonction logique F avec un multiplexeur 4 entrées 1 sortie connecté suivant le schéma de la figure 2. Donner la table de vérité de C_1 et C_0 en fonction de X_3, X_2, X_1. **Aucun** schéma n'est demandé.

La table de vérité ci-dessous donne les sorties attendues du multiplexeur en fonction de X_3, X_2, X_1. Les configurations des commandes s'en déduisent, par exemple la commande doit sélectionner à l'entrée 2 lorsque !X0 est attendu.

X3	X2	X1	S	C1	C0
0	0	0	1	0	1
0	0	1	0	0	0
0	1	0	1	0	1
0	1	1	X0!	1	0
1	0	0	1	0	1

1	0	1	1	0	1
1	1	0	1	0	1
1	1	1	0	0	0

3. Bascules et Automates [3 pts]

Q10. En négligeant les temps de retard entre entrée et sortie, donner le signal Q pour les entrées de la figure 3 dans le cas de la bascule D (opaque) ; Q est initialement à 1.

Voir schéma

Q11. Réaliser un automate qui fonctionne comme un compteur par 6 ou par 8, selon une entrée E qui est à 1 en mode compteur par 6 et 0 en mode compteur par 8. On donnera le nombre de bascules D nécessaires, la table de transition (table de vérité des D_i en fonction des Q_i et de E) et un schéma de principe. L'expression réduite des fonctions de transition n'est **pas** demandée.

Nombre de bascules D nécessaires : 3, pour coder 8 états
Table de transition :

E	Q2	Q1	Q0	D2	D1	D0
0	0	0	0	0	0	1
0	0	0	1	0	1	0
0	0	1	0	0	1	1
0	0	1	1	1	0	0
0	1	0	0	1	0	1
0	1	0	1	0	0	0
0	1	1	0	**d**	**d**	**d**
0	1	1	1	**d**	**d**	**d**
1	0	0	0	1	1	1
1	0	0	1	0	0	0
1	0	1	0	0	0	1
1	0	1	1	0	1	0
1	1	0	0	0	1	1
1	1	0	1	1	0	0
1	1	1	0	1	0	1
1	1	1	1	1	1	0

Table 1

Une table de vérité doit être complète : toutes les configuration des entrées doivent être présentes. En fonctionnement compteur par 6, les sorties dans les états 6 et 7 sont indifférentes (d = don't care), puisque ces états ne se présentent pas.
Le schéma de principe est décrit dans le cours.

4. Microarchitecture [4 pts]

On considère la carte machine de la figure 4. Les réponses utiliseront le format de la table 2.

Q12. Définir les commandes associées à l'exécution de l'instruction SW Rd, Imm(Ra)

Q13. On veut étendre le jeu d'instruction MIPS avec les instructions

- Chargement pré-incrémenté

LWA Rd, Imm(Ra) qui effectue Rd <- Mem [Ra+ES(imm)] et Ra <- Ra+ES(imm)

- Rangement post-incrémenté

SWP Rd, Imm(Ra) qui effectue Mem [Ra] <- Rd et Ra <- Ra+ES(imm)

Définir les commandes associées à l'exécution de ces instructions.

1) SW Rd, Imm(Ra)

En langage transfert :

RAdm <- Ra + ES(Imm) via bus A, bus B, UAL et Bus R.

Mem(RAdM) <- Rd via bus A (pour l'adresse) et bus B (pour la donnée)

2) LWA Rd, Imm(Ra)

La différence avec LW est la modification de Ra (pré-incrémentation). Il faut donc sauvegarder la valeur du calcul d'adresse.

En langage transfert

RAdm et Ra <- Ra + ES(Imm) via bus A, bus B, UAL et Bus R.

Rd <- Mem(RAdM) via bus A (pour l'adresse) et bus R (pour la donnée)

On peut remarquer que l'utilisation de RAdM n'est pas nécessaire, puisque Ra contient l'adresse à la fin du premier cycle. Une solution également valide est donc:

Ra <- Ra + ES(Imm) via bus A, bus B, UAL et Bus R.

Rd <- Mem(Ra) via bus A (pour l'adresse) et bus B (pour la donnée)

3) SWP Rd, Imm(Ra)

La différence avec SW est l'utilisation de Ra sans calcul et sa modification ultérieure (post-incrémentation). Il faut donc effectuer l'incrémentation après l'accès mémoire. Ceci peut se faire en 2 cycles, car Ra peut être utilisé directement pour adresser la mémoire.

En langage transfert

Mem(Ra) <-Rd via bus A (pour l'adresse) et bus B (pour la donnée)

Ra <- Ra + ES(Imm) via bus A, bus B, UAL et Bus R.

La description en langage transfert est indiquée ici pour faciliter la compréhension du corrigé. Elle n'était pas demandée.

La table qui suit répond aux Q12 et Q13.

Inst	Cycle	Bus A	Bus B	Bus R	UAL	EXT	Comp	Mux1	MEM	WR
SW	1	Ra	EXT	UAL	+	ES	X	X	Rien	RAdM
	2	RAdM	Rd	X	X	X	X	X	ECR	aucun
LWA	1	Ra	EXT	UAL	+	ES	X	X	Rien	RAdM, **Ra**
Solution 1	2	RAdM	x	Mem	X	X	X	X	LECT	Rd
LWA	1	Ra	EXT	UAL	+	ES	X	X	Rien	**Ra**
Solution 2	2	Ra	x	Mem	X	X	X	X	LECT	Rd
SWP		Ra	Rd	X	X	X	X	X	ECR	aucun
		Ra	EXT	UAL	+	ES	X	X	Rien	Ra

X3	X2	X1	X0	F	S
0	0	0	0	1	1
0	0	0	1	1	
0	0	1	0	0	0
0	0	1	1	0	
0	1	0	0	1	1
0	1	0	1	1	
0	1	1	0	1	X0!
0	1	1	1	0	
1	0	0	0	1	1
1	0	0	1	1	
1	0	1	0	1	1
1	0	1	1	1	
1	1	0	0	1	1
1	1	0	1	1	
1	1	1	0	0	0
1	1	1	1	0	

Table 1

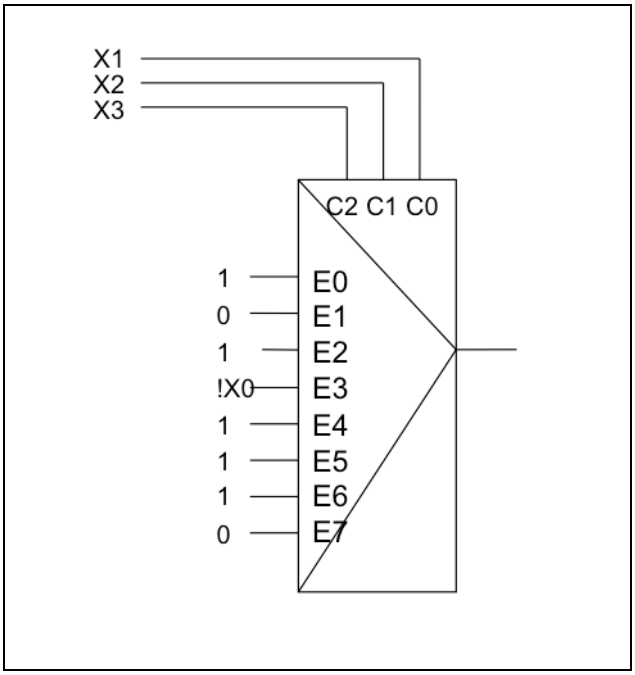

Figure 1

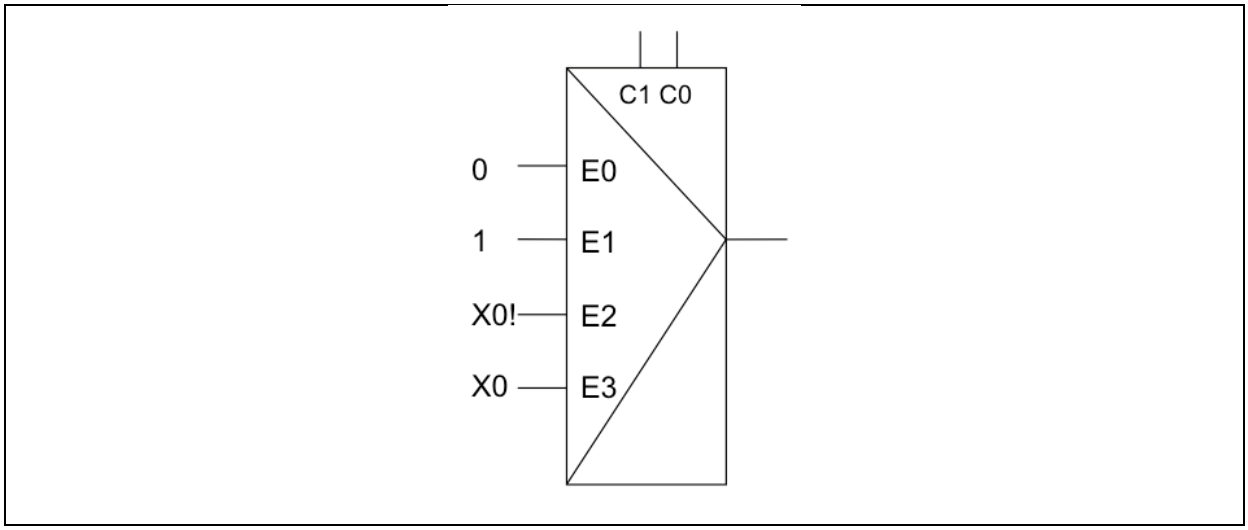

Figure 2

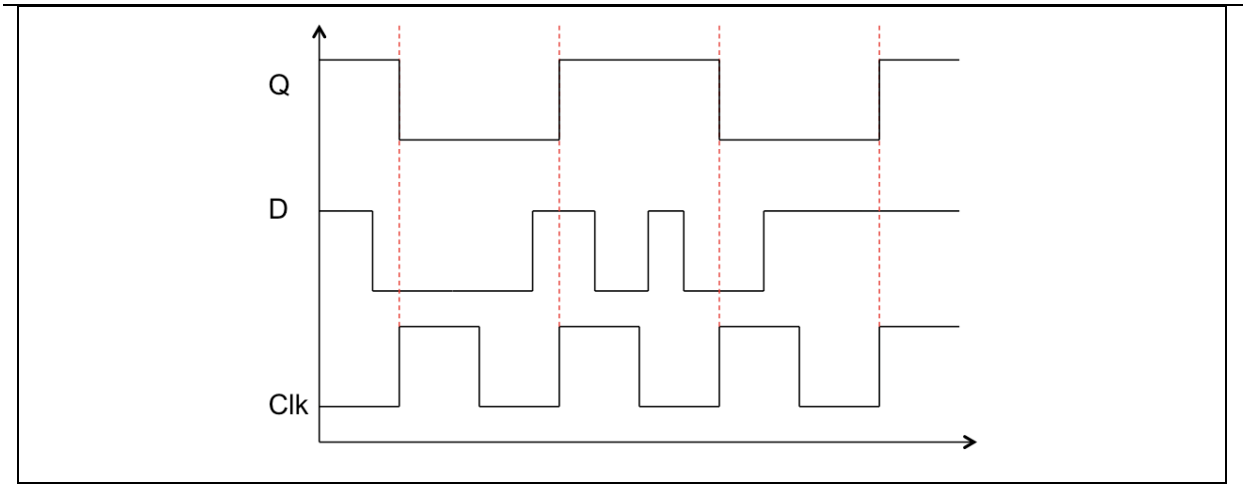

Figure 3

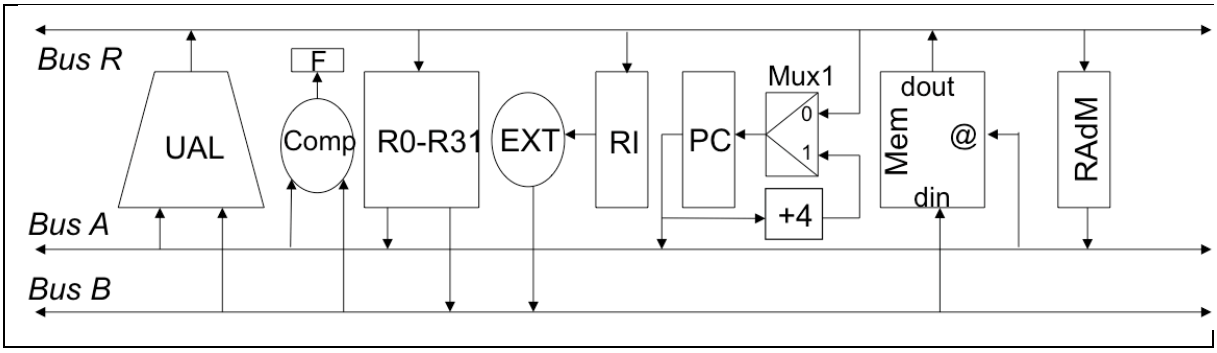

Figure 4

Inst	Cycle	Bus A	Bus B	Bus R	UAL	EXT	Comp	Mux1	MEM	WR

Table 2