Exercices en préparation de l'examen final du cours INF3610 Systèmes embarqués

Chapitre 3 – Bloc processeur/ architectures RISC: parallélisme d'instruction et autres optimisations

QUESTION #1 Architecture RISC et aléas

Soit un pipeline à 5 niveaux semblable au DLX :

- LI : lecture d'instruction
- DI : décodage de l'instruction et lecture des registres
- EX : exécution et calcul de l'adresse effective
- MEM : accès mémoire ou fin de branchement
- ER : écriture du résultat dans le banc de registres

Soit la boucle suivante (Figure 1.1) avec la spécification du tableau 1.1:

```
etiq :  LW      R1, 10(R2)
        ADDI    R1, R1, #1
        SW      R1, 10(R2)
        ADDI    R2, R2, #4
        SUB     R4, R3, R2
        LW      R5, 10(R6)
        BNZ     R4, etiq
```

Figure 1.1

Instructions pouvant être pipelinées	**Signification**	**Nombre de cycles dans EX**	**Cycle du pipeline où l'opération termine (le résultat étant disponible 1 cycle plus tard)**
LW R1, 10(R2)	R1 ← Mem[10 + R2]	**1**	ER (pas d'accumulateur)
ADDI R1, R1, #1	R1 ← R1 + 1	**1**	ER "
SW R1, 10(R2)	Mem[10 + R2] ← R1	**1**	MEM
SUB R4, R3, R2	R4 ← R3 – R2	**1**	ER "
BNZ R3, etiq	Branch. si non zéro	**1**	EX

Table 1.1 Spécifications du DLX

a) À l'aide de la figure 1.1 et du tableau 1.1, complétez le tableau 1.2 en utilisant au besoin des suspensions (bulles). Considérez un modèle M2 de pipeline. Pour l'instruction de branchement, considérez que le pipeline suspend tous les instructions jusqu'à ce que la destination[1]. Autrement dit, le branchement n'est pas effectué et le corps de boucle s'exécute une seule fois.

LW	R1, 10(R2)	LI	DI	EX	MEM	ER		
ADDI	R1, R1, #1		LI	DI	…			
SW	R1, 10(R2)			LI	…			
ADDI	R2, R2, #4				…			
SUB	R4, R3, R2					…		
LW	R5, 10(R6)						…	
BNZ	R4, etiq							…

Tableau 1.2

b) Refaire a) mais en supposant le tableau 1.3 à la place du tableau 1.1 (toujours avec un modèle M2). De plus, considérez maintenant une politique de *prédiction de branchement pris* (avec toujours une destination qui est l'instruction LW R1, 10(R2)).

Instructions pouvant être pipelinées	**Signification**	**Cycle du pipeline où l'opération termine et entre parenthèse cycle ou la donnée est disponible grâce aux chemins d'envoi et d'un accumulateur.**
LW R1, 10(R2)	R1 ← Mem[10 + R2]	ER (MEM)
ADDI R1, R1, #1	R1 ← R1 + 1	ER (EX)
SW R1, 10(R2)	Mem[10 + R2] ← R1	MEM
SUB R4, R3, R2	R4 ← R3 – R2	ER (EX)
BNZ R3, etiq	Branch. si non zéro	EX

Tableau 1.3 Spécifications du DLX

[1] On parle alors d'un branchement avec délai (*branch delay*)

QUESTION #2 Architecture RISC, aléas et superscalaire

Soit comme à la question précédente un pipeline à 5 niveaux semblable au DLX et soit la boucle suivante (Figure 2.1) qui réalise l'opération vectorielle (double mot) Z = X - Y + Z sur des vecteurs de longueur 16:

```
etiq :  LD      F10, 0(R1)          ; charge  X(i)
        LD      F20, 0(R2)          ;    ″    Y(i)
        SUBF    F30, F10, F20       ; X(i) - Y(j)
        LD      F40, 0(R3)          ; charge Z(i)
        ADDF    F50, F40, F30       ; X(i) – Y(i) + Z(i)
        SD      0(R3), F50          ; range Z(i)
        ADDI    R1, R1, #8          ; incrémente l'indice  X
        ADDI    R2, R2, #8          ;      ″          ″    Y
        ADDI    R3, R3, #8          ;      ″          ″    Z
        SGTI    R5, R1, R4          ; teste si plus grand que 16*8
        BEQZ    R5, etiq            ; boucle sinon
```

où

- X(0), Y(0) et Z(0) se trouve à partir de l'adresse R1, R2 et R3
- R1, R2 et R3 sont initialisés à 0, 128 et 256 respectivement.
- R4 est initialisé à 128.

Figure 2.1

Détails des d'instructions pouvant être pipelinées	**Nom de l'instruction**	**Nombre de cycles dans EX**	**Cycle du pipeline où l'opération termine**
LD F10, 0(R1)	**Charg. mot dans F10**	**1**	**ER (le résultat est dans l'accumulateur après MEM)**
ADDF F20, F0, F10 **SUBF F20, F0, F10**	**Additionne F0 et F10** **Soustrait F0 de F10**	**4**	**ER (le résultat est mis dans l'accumulateur après EX4)**
ADDI R1, R1, #8	**Add immédiat**	**1**	**ER (le résultat est mis dans l'accumulateur après EX)**
SGTI R3, R1, R4	**si (R1 > R4) alors R3 <= 1** **sinon R3 <= 0**	**1**	**ER ((le résultat est mis dans l'accumulateur après EX)**
SD 0(R2), F6	**Rang. d'un mot à partie de F6**	**1**	**MEM**
BEQZ R3, etiq	**Branch. si zéro**	**1**	**EX**

Table 2.1 Instructions

À partir de cette boucle et à l'aide de la table 2.1:

a) Donnez la boucle non ordonnancée, en tenant compte des cycles de suspension ou d'attente. Considérez un modèle de pipeline de type M4. Donnez le temps d'exécution.

b) À partir du résultat obtenu en a), dépliez la boucle 4 fois et ordonnancé les instructions afin de minimiser le nombre de cycles par itération. Au besoin vous pouvez ajouter des registres Ri ou Fi. Donnez le nombre de cycles par itération résultant.

c) Soit une machine superscalaire à deux pipelines distincts: un pipeline pour les instructions entière (et les accès mémoire) et un pipeline pour le point flottant. Ordonnancez la boucle dépliée obtenue en b). Donnez le nombre de cycles par itération résultant.

QUESTION #3 Pipeline et parallélisme d'instructions

Soit une boucle qui réalise l'opération vectorielle Y=a*X+Y/b pour un vecteur de longueur de 100 selon le jeu d'instructions du tableau 3.1:

```
L:    LD      F10, 0(R1)
      MULT F  F20, F10, F0
      LD      F30, 0(R2)
      DIVF    F40, F30, F1
      ADDF    F40, F40, F20
      SD      0(R2), F40
      ADDI    R1, R1, 8
      ADDI    R2, R2, 8
      STGI    R3, R1, R4
      BEQZ    R3, L
```

où F0 = a, F1=b, R4 = 100 et finalement R1 = 0 au départ

Instructions pipelinées	Signification	Nombre de cycles dans EX	Cycle du pipeline où l'opération termine (le résultat étant disponible 1 cycle plus tard)
LD F0, 0(R3)	**F0 ← Mem[0 + R3]**	**1**	**ER (le résultat est dans l'accumulateur après MEM)**
ADDF F8, F6, F1	**F8 ← F6 + F1**	**2**	**ER (le résultat est mis dans l'accumulateur après EX2)**
MULTF F4, F2, F0	**F4 ← F2 * F0**	**3**	**ER (le résultat est mis dans l'accumulateur après EX3)**
DIVF F8, F6, F1	**F8 ← F6 + F1**	**4**	**ER (le résultat est mis dans l'accumulateur après EX4)**
ADDI R1, R1, 8	**R1 ← R1 + 8**	**1**	**ER (le résultat est mis dans l'accumulateur après EX)**
SD 0(R2), F8	**Mem[0 + R2] ← F8**	**1**	**MEM**
STGI R3, R1, R4	**si R1 > R4 alors R3=1 sinon 0**	**1**	**EX**
BEQZ R3, etiq	**Branchement si zéro**	**1**	**EX**

Tableau 3.1 Instructions pipelinées

a) Complétez la trace de ce programme (Tableau 3.2, page suivante) pour l'exécution d'une seule boucle. Considérez le modèle M3. Donnez le nombre de cycles pour une itération.

b) Montrez l'avantage du modèle M4 par rapport à M3, lorsqu'on désire dérouler le code donné en a) autant de fois que nécessaire. Montrez l'ordonnancement et le gain par rapport à a).

##		Instruction/Cycl	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
1	L:	LD F10, 0(R1)																										
2		MULTF F20,F10,F0																										
3		LD F30,0(R2)																										
4		DIVF F40,F30,F20																										
5		ADDF F40, F40, F20																										
6		SD 0(R2),F40																										
7		ADDI R1, R1, 8																										
8		ADDI R2, R2, 8																										
9		STGI R3,R1,R4																										
10		BEQZ R3, L																										

Tableau 3.2 À compléter pour la question a)

Question 4 Architecture RISC de base et ordonnancement

Soit le programme DLX suivant pour un filtre en point flottant 1D FIR TAP où n = 4. En langage C on aura:

```
for (i = 3 ; i  < 100; i++)
   y[i] = a0 * x[i-3] + a1 * x[i-2] + a2 * x[i-1] + a3 * x[i]
```

De plus, on a:

- 4 registres F0, F10, F20 et F30 pour les constantes a0, a1, a2 et a3 respectivement;
- 4 registres F40, F50, F60 et F70 pour chaque échantillon représentant x[i], x[i1-1], x[i-2] et x[i-3] dans un vecteur de dimension 100,
- n varie de 3 à 99 via R4, et
- 1 registre résultant F80 (initialement à 0) pour réaliser un multiplicateur/accumulateur (MAC).

En assembleur DLX cela correspond au programme suivant:

```
L:      LD      F80, #0
        LD      F40, -24(R4)    ; traitement de i-3
        MULT    F40, F40, F0
        ADDF    F80, F80, F40
        LD      F50, -16(R4)    ; traitement de i-2
        MULT    F50, F50, F10
        ADDF    F80, F80, F50
        LD      F60, -8(R4)     ; traitement de i-1
        MULT    F60, F60, F20
        ADDF    F80, F80, F60
        LD      F70, 0(R4)      ; traitement de i
        MULT    F70, F70, F30
        ADDF    F80, F80, F70
        SD      1000(R6), F80
        ADDI    R4, R4, #8
        ADDI    R6, R6, #8
        SEQI    R5, R4, #800
        BEQZ    R5, L
```

avec au départ R4=24, de plus le résultat est placé dans un nouveau vecteur indexé à partir de l'adresse 1000

Le Tableau 4.1 explique les différents types d'instructions assembleurs.

Détails des instructions pouvant être pipelinées	Nom de l'instruction	Nombre de cycles dans EX	Cycle du pipeline où l'opération termine
LD F10, 0(R1)	**Charge le mot dans F10**	**1**	**ER (le résultat est dans l'accumulateur après MEM)**
LD F10, #0	**Charge directement à 0**	**1**	**ER**
ADDF F20, F0, F10	**Additionne F0 et F10**	**3**	**ER (le résultat est mis dans l'accumulateur après EX3)**
MULT F20, F0, F10	**Multiplie F0 et F10**	**5**	**ER (le résultat est mis dans l'accumulateur après EX5)**
ADDI R1, R1, #8	**Addition entière**	**1**	**EX (le résultat est mis dans l'accumulateur après EX)**
SEQI R5, R4, #val	**si (R4 = #val) alors R5 <= 1 sinon R5 <= 0**	**1**	**ER (le résultat est mis dans l'accumulateur après EX)**
SD 0(R2), F6	**Rang. d'un mot à partir de F6**	**1**	**MEM**
BEQZ R3, etiq	**Branch. si zéro**	**1**	**EX**

Tableau 4.1

a) Donnez la trace du programme en complétant le Tableau 4.2 pour l'exécution d'une seule boucle, en tenant compte des cycles de suspension ou d'attente. Considérez un modèle de pipeline de type M4. Donnez le temps d'exécution.

b) À partir de la trace obtenue en a), proposez une réorganisation (ré-ordonnancement) du programme en déroulant la boucle 2 fois afin d'accélérer au maximum l'exécution du filtre FIR. Vous devez préserver la fonctionnalité et au besoin vous pouvez ajouter des registres. *Remarque : le choix des numéros de registre (p.e. F10) a volontairement été espacé pour que vous puissiez, au besoin, en introduire de nouveaux (p.e. F11, F12, etc.).*Vous devez donc :
   b.1) Complétez le Tableau 4.3
   b.2) Montrez le gain de l'ordonnancement résultant et comparez le gain par rapport à a).

c) Soit une architecture superscalaire de type M5 (Voir Annexe). Proposez une façon d'améliorer au maximum l'exécution de ce programme sur cette architecture en ordonnançant le code obtenu en b). Vous devez donc :

   c.1) Complétez le Tableau 4.4
   c.2) Montrez le gain de l'ordonnancement résultant et comparez le gain par rapport à a).

d) Soit une architecture VLIW (Very Long Instruction World) avec des mots de longueur 160 bits. Considérez 2 unités de transfert (accès mémoires), 2 unités de calcul flottant et 1 unité de calcul entière (incluant les instructions de branchement). Proposez une façon d'améliorer au maximum l'exécution de ce programme sur cette architecture en ordonnançant le code obtenu en b). Vous devez donc :

   d.1) **Complétez le Tableau 1.5 de la page 5/14**
   d.2) Comparez le gain par rapport à a).

Instructions	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44
LD F80, #0																																												
LD F40, -24(R4)																																												
MULT F40, F40, F0																																											
ADDF F80, F80, F40																																											
LD F50, -16(R4)																																											
MULT F50, F50, F10																																											
ADDF F80, F80, F50																																											
LD F60, -8(R4)																																											
MULT F60, F60, F20																																											
ADDF F80, F80, F60																																											
LD F70, 0(R4)																																											
MULT F70, F70, F30																																											
ADDF F80, F80, F70																																											
SD F80, 0(R4)																																											
ADDI R4, R4, #8																																											
ADDI R6, R6, #8																																											
SEQI R5, R4, #800																																											
BEQZ R5, L																																											

Tableau 4.2 À compléter

Cycle	Pipeline 1
1 L :	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	
16	
17	
18	
19	
20	
21	
22	
23	
24	
25	
26	
27	
28	
29	
30	
31	
32	
33	
34	
35	

Tableau 4.3 À compléter

Cycle	Pipeline 1	Pipeline 2
1 L :		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		
16		
17		
18		
19		
20		
21		
22		
23		
24		
25		
26		
27		
28		
29		
30		

Tableau 4.4 À compléter

	UNITÉ TRANSFERT 1	UNITÉ TRANSFERT 2	UNITÉ FLOTTANTE 1	UNITÉ FLOTTANTE 2	UNITÉ ENTIÈRE
1bL:					
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					
14					
15					
16					
17					
18					
19					
20					

Tableau 4.5 À compléter

QUESTION #5 FIR et architecture VLIW

Soit l'application du filtre 1D FIR N TAP :

$y(n) = a0*x(n) + a1*x(n-1) + a2*x(n-2) + ... + aN-1*x(n-N-1)$

où on a[2]

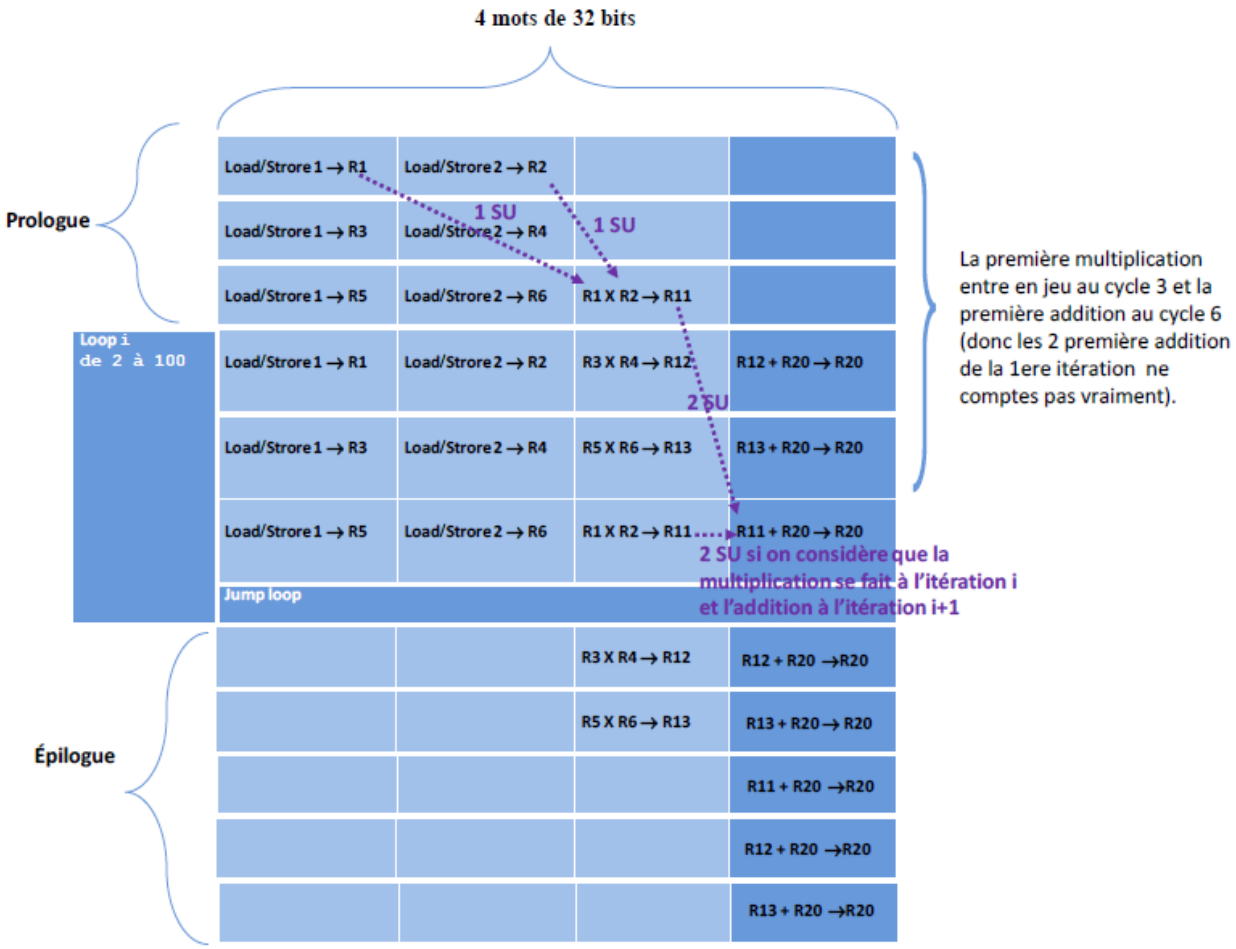
- une table de N coefficients, a0 à aN-1
- une table de N échantillons, allant de l'échantillon courant x(n) à l'échantillon le plus ancien x(n-N-1)
- pour chaque échantillon résultat y(n) courant, la nécessitée d'effectuer N opérations MAC.
- une structure itérative, l'échantillon d'entrée x(n) courant devenant l'échantillon précédant à chaque calcul du nouvel échantillon de sortie y(n)

Soit d'autre part une machine VLIW composé de 2 unités pour lecture/écriture (1 cycle pour EX), 1 unité de multiplication (3 cycles pour EX) et 1 unité d'addition/soustraction (1 cycle pour EX). Expliquez ce que représente le code de la Figure 5.1 et la technique de programmation employée. Indiquez aussi l'endroit où se trouvent le prologue et l'épilogue ainsi que leurs raisons d'être.

Finalement, on suppose les latences d'exécution suivantes :

1) 1 cycle pour le load
2) 3 cycles pour la mult (flottante)
3) 3 cycles pour le add (flottant)

[2] Pour plus d'information référez à http://b.l.free.fr/Page7.html


4 mots de 32 bits
Prologue
Load/Strore 1 → R1
Load/Strore 2 → R2
Load/Strore 1 → R3
Load/Strore 2 → R4
Load/Strore 1 → R5
Load/Strore 2 → R6
R1 X R2 → R11
1 SU
1 SU
Loop i
de 2 à 100
Load/Strore 1 → R1
Load/Strore 2 → R2
R3 X R4 → R12
R12 + R20 → R20
2 SU
Load/Strore 1 → R3
Load/Strore 2 → R4
R5 X R6 → R13
R13 + R20 → R20
Load/Strore 1 → R5
Load/Strore 2 → R6
R1 X R2 → R11
R11 + R20 → R20
2 SU si on considère que la multiplication se fait à l'itération i et l'addition à l'itération i+1
Jump loop
Épilogue
R3 X R4 → R12
R12 + R20 →R20
R5 X R6 → R13
R13 + R20 → R20
R11 + R20 →R20
R12 + R20 →R20
R13 + R20 →R20
La première multiplication entre en jeu au cycle 3 et la première addition au cycle 6 (donc les 2 première addition de la 1ere itération ne comptes pas vraiment).