TD 5 — 4MMARCA, Ensimag 2A/S6, ISI/SEOC

Conception d'un processeur pipeline

Architecture avancée, 2017-2018

Cet exercice est largement inspiré de l'exercice A5, page A-82 du Hennessy/Patterson 3ème édition.

Ces questions ont pour but de concevoir une machine dont l'un des opérandes peut être une case mémoire, ce que l'on appelle une machine registre/mémoire. La machine possède 2 formats d'instruction : un format registre/registre et un format registre mémoire. Il n'y a par contre qu'un seul mode d'adressage, le mode indirect registre plus déplacement identique à celui du MIPS que nous avons vu jusqu'à présent.

Les instructions utilisant l'ALU (opérations ADD, SUB, AND, OR ...) s'utilisent comme suit :

```
OPR Rdst, Rsrc1, Rsrc2
OPM Rdst, Rsrc1, CST(Radr)
```

On peut par ailleurs effectuer des accès mémoires sans calculs avec les instructions suivantes :

```
LW Rdst, CST(Radr)
SW Rsrc1, CST(Radr)
```

Les branchements permettent de faire tous les types de comparaison entre deux registres (égalité, supériorité stricte, ...) et sont relatifs au PC.

```
Bcc Rsrc1, Rsrc2, label
```

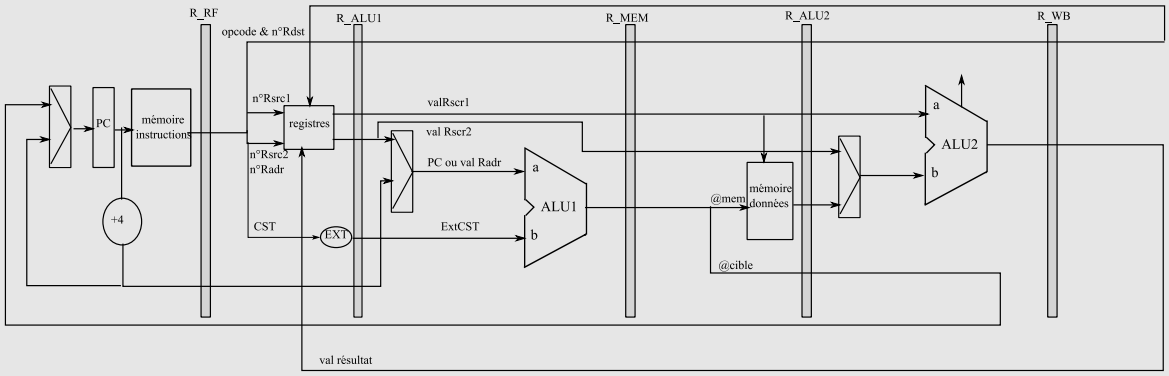
La machine est pipelinée en 6 étages (dans l'ordre IF, RF, ALU1, MEM, ALU2 et WB) de sorte qu'une instruction est commencée à chaque cycle. L'étage IF procède au chargement de l'instruction via la mémoire d'instruction. L'étage RF (*Registers Fetch*) réalise à la fois le décodage de l'instruction et la récupération des valeurs des registres. L'étage ALU1 est utilisé pour le calcul de l'adresse effective des accès mémoire et des branchements. L'étage MEM permet d'accéder à la mémoire de données. L'étage ALU2 est utilisé pour les opérations de l'ALU et pour les comparaisons utilisées par les branchements.

On fait de plus l'hypothèse que si le même registre est lu *et* écrit au même cycle, alors la lecture retourne la valeur qu'aura l'écriture à la fin du cycle.

Question 0 - Donnez le schéma bloc d'un banc de registres ayant le comportement décrit ci-dessus en supposant que l'on dispose d'un banc de registres « normal », c'est-à-dire qui peut-être écrit et lu dans le même cycle, mais pour lequel si le même registre est la cible d'une écriture et d'une lecture dans le même cycle, on récupérera en lecture la valeur précédent l'écriture. Le banc de registre « normal » ne sera pas détaillé, on se basera uniquement sur ses entrées/sorties pour bâtir la nouvelle version.

Question 1 - Dessinez sommairement le pipeline en positionnant étage par étage les ressources principales de la machine (mémoires, banc, ALU...). Précisez le nombre maximum d'opérateurs pouvant réaliser une addition dont a besoin cette architecture. Précisez aussi le nombre d'accès et le type (lecture/écriture) de chaque accès nécessaire au banc de registres et aux mémoires.

En supposant que l'ALU2 fait un nop sur son entrée b pour l'instruction lw, le pipeline ressemble à :



3 opérateurs : un dans IF pour incrémenter PC, un dans ALU1 pour faire Radr + CST ou PC + CST, un dans ALU2 pour la vraie ALU.

Pour le banc de registres : 2 ports en lecture et 1 en écriture. A noter : si on avait codé le SW par SW Rsrc1, CST(Radr), il aurait fallu 3 ports de lecture : 1 par champ registre, à moins de décoder partiellement l'instruction au début de RF pour savoir si il faut lire $Rsrc_1$ ou Rdst.
Pour la mémoire I : 1 port R
Pour la mémoire D : 1 port R et 1 W

Pour lister de manière exhaustive les aléas de données, il faut détecter les cas de dépendances lecture après écriture (RaW, Read after Write). Pour le processeur décrit, les instructions accèdent à :

- 1 registre en écriture (Rdst) et 2 en lecture (Rsrc1, Rsrc2) pour les instructions OPR
- 1 registre en écriture (Rdst) et 2 en lecture (Radr, Rsrc1) pour les instructions OPM
- 1 registre en écriture (Rdst) et 1 en lecture (Radr) pour l'instruction LW
- 2 en lecture (Radr, Rsrc1) pour l'instruction SW
- 2 en lecture (Rsrc1, Rsrc2) pour les instructions Bcc

Avec 3 types d'accès en écriture et 9 d'accès en lecture, on peut distinguer 27 cas de dépendances RaW. Pour chacun de ces cas et selon la distance en nombre d'instructions séparant l'instruction impliquée dans l'écriture de celle impliquée dans une lecture, on peut dire rapidement s'il y a un aléa et s'il nécessite une attente.

Question 2 - Chaque situation correspond à une case du tableau fourni. Remplissez-le avec au choix : OK, court-circuit (bypass), bulles (stall) et leur nombre.

dépendance RaW				Distance			
Inst. W	Reg W	Inst. R	Reg. R	1	2	3	4
OPR	Rdst	OPR	Rsrc1				
		OPR	Rsrc2				
		OPM	Rsrc1				
		OPM	Radr				
		LW	Radr				
		SW	Radr				
		SW	Rsrc1				
		Bcc	Rsrc1				
		Bcc	Rsrc2				
OPM	Rdst	OPR	Rsrc1				
		OPR	Rsrc2				
		OPM	Rsrc1				
		OPM	Radr				
		LW	Radr				
		SW	Radr				
		SW	Rsrc1				
		Bcc	Rsrc1				
		Bcc	Rsrc2				
LW	Rdst	OPR	Rsrc1				
		OPR	Rsrc2				
		OPM	Rsrc1				
		OPM	Radr				
		LW	Radr				
		SW	Radr				
		SW	Rsrc1				
		Bcc	Rsrc1				
		Bcc	Rsrc2				

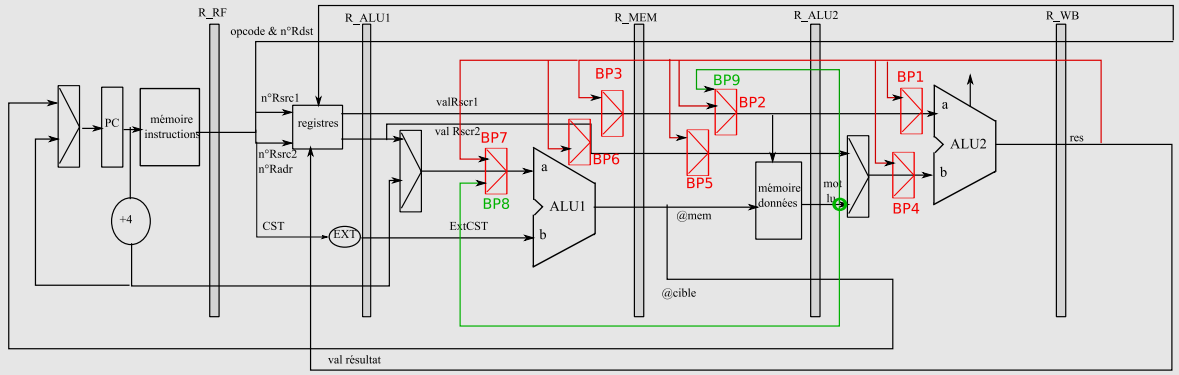
	dépendance RaW				Distance			
	Inst. W	Reg W	Inst. R	Reg. R	1	2	3	4
1	OPR	Rdst	OPR	Rsrc1	bypass BP1	bypass BP2	bypass BP3	OK
2			OPR	Rsrc2	bypass BP4	bypass BP5	bypass BP6	OK
3			OPM	Rsrc1	bypass BP1	bypass BP2	bypass BP3	OK
4			OPM	Radr	stall 2	stall 1	bypass BP7	OK
5			LW	Radr	stall 2	stall 1	bypass BP7	OK
6			SW	Radr	stall 2	stall 1	bypass BP7	OK
7			SW	Rsrc1	stall 1	bypass BP2	bypass BP3	OK
8			Bcc	Rsrc1	bypass BP1	bypass BP2	bypass BP3	OK
9			Bcc	Rsrc2	bypass BP4	bypass BP5	bypass BP6	OK
10	OPM	Rdst	OPR	Rsrc1	bypass	bypass	bypass	OK
11			OPR	Rsrc2	bypass	bypass	bypass	OK
12	idem		OPM	Rsrc1	bypass	bypass	bypass	OK
13	cas		OPM	Radr	stall	stall	bypass	OK
14	1 à 9		LW	Radr	stall	stall	bypass	OK
15			SW	Radr	stall	stall	bypass	OK
16			SW	Rsrc1	stall	bypass	bypass	OK
17			Bcc	Rsrc1	bypass	bypass	bypass	OK
18			Bcc	Rsrc2	bypass	bypass	bypass	OK
19	LW	Rdst	OPR	Rsrc1	bypass BP1	bypass BP2	bypass BP3	OK
20			OPR	Rsrc2	bypass BP4	bypass BP5	bypass BP6	OK
21			OPM	Rsrc1	bypass BP1	bypass BP2	bypass BP3	OK
22			OPM	Radr	stall 1	bypass BP8	bypass BP7	OK
23			LW	Radr	stall 1	bypass BP8	bypass BP7	OK
24			SW	Radr	stall 1	bypass BP8	bypass BP7	OK
25			SW	Rsrc1	bypass BP9	bypass BP2	bypass BP3	OK
26			Bcc	Rsrc1	bypass BP1	bypass BP2	bypass BP3	OK
27			Bcc	Rsrc2	bypass BP4	bypass BP5	bypass BP6	OK

Question 3 - Ajoutez sur votre dessin de pipeline le matériel permettant de résoudre les courts-circuits. On ne demande pas ici la détection des aléas.

Il faut ajouter les court-circuits suivants :

- BP1 : chemin entre le champ res de R_WB et le port a de l'ALU2.
- BP2 : chemin entre le champ res de R_WB et le champ Rsrc1 de R_ALU2 ou le port écriture de données de la mémoire.
- BP3 : chemin entre le champ res de R_WB et le champ Rsrc1 de R_MEM
- BP4 : chemin entre le champ res de R_WB et le port b de l'ALU2.
- BP5 : chemin entre le champ res de R_WB et le champ Rsrc2 de R_ALU2
- BP6 : chemin entre le champ res de R_WB et le champ Rsrc2 de R_MEM
- BP7 : chemin entre le champ res de R_WB et le port port a de l'ALU1
- BP8 : chemin entre le champ motlu de R_ALU2 et le port a de l'ALU1
- BP9 : chemin entre le champ motlu de R_ALU2 et le port écriture de données de la mémoire

Sur la PO précédente, on obtient :



Question 4 - Indiquez pour chaque court-circuit l'équation de détection associée, ainsi que celle pour la détection des cycles d'attente à insérer.

Equation de BP1 :

$$R\_WB.nRdst == R\_ALU2.nRsrc1 \& R\_WB.opcode == (OPR|OPM|LW) \\ \& R\_ALU2.opcode == (OPR|OPM|Bcc)$$

Equation de BP2 :

$$R\_WB.nRdst == R\_MEM.nRsrc1 \& R\_WB.opcode == (OPR|OPM|LW) \\ \& R\_MEM.opcode == (OPR|OPM|Bcc|SW)$$

Equation de BP3 :

$$R\_WB.nRdst == R\_ALU1.nRsrc1 \& R\_WB.opcode == (OPR|OPM|LW) \\ \& R\_ALU1.opcode == (OPR|OPM|Bcc|SW)$$

Equation de BP4 :

$$R\_WB.nRdst == R\_ALU2.nRsrc2 \& R\_WB.opcode == (OPR|OPM|LW) \\ \& R\_ALU2.opcode == (OPR|Bcc)$$

Equation de BP5 :

$$R\_WB.nRdst == R\_MEM.nRsrc2 \& R\_WB.opcode == (OPR|OPM|LW) \\ \& R\_MEM.opcode == (OPR|Bcc)$$

Equation de BP6 :

$$R\_WB.nRdst == R\_ALU1.nRsrc2 \& R\_WB.opcode == (OPR|OPM|LW) \\ \& R\_ALU1.opcode == (OPR|Bcc)$$

Equation de BP7 :

$$R\_WB.nRdst == R\_ALU1.nRadr \& R\_WB.opcode == (OPR|OPM|LW) \\ \& R\_ALU1.opcode == (OPM|LW|SW)$$

Equation de BP8 :

$$R\_ALU2.nRdst == R\_ALU1.nRadr \& R\_ALU2.opcode == (LW) \\ \& R\_ALU1.opcode == (OPM|LW|SW)$$

Equation de BP9 :

$$R\_ALU2.nRdst == R\_MEM.nRsrc1 \& R\_ALU2.opcode == (LW) \\ \& R\_MEM.opcode == (SW)$$

Equation de détection de stall (IF,RF et ALU1 sont à bloquer) :

$$R\_MEM.nRdst == R\_ALU1.nRadr \& R\_MEM.opcode == (OPR|OPM|LW) \\ \& R\_ALU1.opcode == (OPM|LW|SW) \\ |R\_ALU2.nRdst == R\_ALU1.nRadr \& R\_ALU2.opcode == (OPR|OPM) \\ \& R\_ALU1.opcode == (OPM|LW|SW) \\ |R\_MEM.nRdst == R\_ALU1.nRsrc1 \& R\_MEM.opcode == (OPR|OPM) \\ \& R\_ALU1.opcode == (SW)$$

Le stall nécessite de mettre l'inverse du résultat de détection sur les WE des registres de l'étages IF, de RF et de ALU1. Il faut aussi mettre un opcode nop dans R_MEM.

Toutes ces équations sont calculées au plus tard (i.e. dans le cycle où elles sont utiles). Il est possible de les anticiper dans les cycles précédents pour moins affecter la fréquence d'horloge.

Question 5 - Pour finir, donnez la liste des aléas liés au contrôle (i.e. le nombre d'instructions suivant un branchement, qui peuvent poser un problème).

On découvre que l'instruction est un branchement dans RF. Dans ALU1, on calcule l'adresse de destination du saut éventuel. Dans ALU2, on sait si on saute ou non. Si on n'applique aucune solution, cela signifie qu'il y aura 4 instructions dans le pipe : soit un delay slot de 4 instructions.

Il peut introduire à ce stade le prédicteur de branchement. Très simplement, on peut faire l'hypothèse de branchements non pris. Coût 0 si la prédiction est bonne, coût 4 instructions à flusher sinon.

S'il reste du temps, on peut faire remarquer qu'il faudrait faire en sorte que la décision de saut soit beaucoup plus proche du fetch.