

# Compte rendu Projet 2 - SE201

Sevin Adrien, Gaillard Vincent, Bias Steven

October 7, 2016

## 3 - Pipelining

### 3.1

En observant les instructions, on remarque un data hazard lors de l'exécution de l'instruction 3. En effet, le store (instruction 3) nécessite la valeur du registre 29, mais celle-ci n'est pas encore à jour. L'instruction précédente qui écrit dans ce registre n'est pas encore terminée.

Au cycle 17, la valeur de r29 n'est pas à jour mais elle devrait être chargée par le store qui est en ID. Au cycle 18, l'instruction 3 passe directement à l'exécution sans avoir attendu de pouvoir lire la valeur du registre 29. Il y a donc bien forwarding.

cycles and pipeline																				
Instructions/Cycles	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
nop	WB	WB	WB	WB	WB	WB	WB	WB	WB	WB	WB	WB	WB	WB	WB	WB	WB	WB	WB	WB
addui r29,r29,-32						IF	IF	IF	IF	IF	IF	ID	EX	EX	EX	EX	EX	EX	MEM	MEM
sw 28(r29),r31												IF	ID	ID	ID	ID	ID	ID	EX	EX
or r30,r29,r0													IF	IF	IF	IF	IF	IF	ID	ID
lhi 29,3840																			IF	IF
jal 0x1054																				
nop																				

### 3.2

Premier exemple :

slti r2,r2,2																				
beqz r2,0xfffff50	WB	WB	WB	WB																
nop	MEM	MEM	MEM	MEM	WB															
sw 28(r30),r0	EX	EX	EX	EX	MEM	WB														
beqz r0,0x78	ID	ID	ID	ID	EX	MEM	WB													
nop	IF	IF	IF	IF	ID	EX	MEM	WB												
lw r2,28(r30)						IF	ID	EX	MEM	WB										
lw r2,24(r30)							IF	ID	EX	MEM	WB	WB								
addui r3,r2,-1								IF	ID	ID	EX	MEM								
lw r2,28(r30)									IF	IF	ID	EX								
slt r2,r2,r3												IF	ID							

Dans le cas d'une condition (beqz, bnez) à cause d'une "misprediction" le processeur va "flush" les instructions qui avaient été commencé. Celles-ci ne se finiront jamais.

Deuxième exemple :

lw r30, 20(r29)	EX	EX	EX	EX	EX	MEM	WB	WB	WB	WB	WB	WB	
addui r29, r29, 24	ID	ID	ID	ID	ID	EX	MEM	MEM	MEM	MEM	MEM	MEM	WB
jr r31	IF	IF	IF	IF	IF	ID	EX	EX	EX	EX	EX	EX	MEM
nop						IF	ID	ID	ID	ID	ID	ID	EX
addui r29, r29, -72							IF	IF	IF	IF	IF	IF	ID
lw r2, 28(r30)													IF
addui r2, r2, 1													
sw 28(r30), r2													

Dans le cas d'un jump (jr) les instructions qui suivent vont être "flushed" et ne seront pas finies.

### 3.3

Notons que chaque jump et condition est suivie d'une instruction nop. Cette instruction n'est pas "flushed".

Cette technique permet de ne jamais exécuter trop longtemps une instruction qui sera stoppée ensuite.

Concrètement, une instruction "flushed" n'atteint jamais l'état ID.

## 4 - Branch Prediction

### 4.1

1-bit saturation counting branch predictor : misprediction rate = 19.14%

2-bit saturation counting branch predictor : misprediction rate = 18.09%

Donc le branch predictor sur 2 bits est meilleur que celui sur 1 bit.

On peut aussi voir la meilleure prédiction dans le nombre de "fetches" qui ont dû être faits. 2499 contre 2503 pour le *1-bit prediction*.

Cela est due au fait que le *2-bit prediction* permet de supprimer le bruit en quelque sorte. Effectivement, si dans un programme il y a beaucoup de boucles (ce qui est le cas ici), même si de temps en temps les boucles seront finies et il y aura une misprediction, on peut supposer que la condition est toujours vérifiée.

Dans ce cas, la méthode *1-bit prediction* entraîne deux mispredictions, contre une seule pour celle de *2-bit prediction*.

### 4.2

Selon les branches, on remarque quelques différences entre les *branch predictors*.

Les seules branches où il y a une différences sont :

1-bit saturation counting branch predictor :

bpc: 0x00001104 [4] a:56 t/nt: 10/46 mp/cp: 9/47 mp-ratio: 0,16

bpc: 0x00001158 [88] a:27 t/nt: 21/6 mp/cp: 12/15 mp-ratio: 0,44

2-bit saturation counting branch predictor :

```
Number of unique jumps: 10  
bpc: 0x0000104c [76] tgts: [0x0000113c] a:66 t/nt: 66/0 mp/cp: 5/61 mp-ratio: 0,08  
bpc: 0x00001104 [4] tgts: [0x0000113c] a:46 t/nt: 10/36 mp/cp: 10/36 mp-ratio: 0,22  
bpc: 0x00001158 [88] tgts: [0x000010d4] a:27 t/nt: 21/6 mp/cp: 7/20 mp-ratio: 0,26  
bpc: 0x00001134 [52] tgts: [0x00001000] a:16 t/nt: 16/0 mp/cp: 1/15 mp-ratio: 0,06  
bpc: 0x000010cc [76] tgts: [0x00001148] a:11 t/nt: 11/0 mp/cp: 6/5 mp-ratio: 0,55
```

On remarque donc que pour la branche [4] le branch predictor sur 1 bit est légèrement meilleur, mais sur la branche [88], le branch predictor sur 2 bit est bien meilleur.

On peut en conclure qu'il est difficile de classer les prédicteurs par performance car elles dépendent du code exécuté.

### 4.3



slt r2,r2,7	IF	IF	IF	ID	ID	EX	MEM	WB											
bnez r2,0xffffffffbc				IF	IF	ID	EX	MEM	WB	WB	WB	WB	WB	WB	WB	WB	WB	WB	WB
nop						IF	ID	EX	MEM	MEM	MEM	MEM	MEM	MEM	MEM	MEM	MEM	MEM	MEM
lhi 2,1							IF	ID	EX	EX	EX	EX	EX	EX	EX	EX	EX	EX	EX
addui r2,r0,7								IF	ID	ID	ID	ID	ID	ID	ID	ID	ID	ID	ID

slt r2,r2,r3	ID	ID	ID	ID	EX	MEM	WB	WB	WB	WB	WB	WB	WB	WB	WB	WB	WB	WB	WB
beqz r2,0x34	IF	IF	IF	IF	ID	EX	MEM	MEM	MEM	MEM	MEM	MEM	MEM	MEM	MEM	MEM	MEM	MEM	MEM
nop					IF	ID	EX	EX	EX	EX	EX	EX	EX	EX	EX	EX	EX	EX	EX
lw r2,28(r30)						IF	ID	ID	ID	ID	ID	ID	ID	ID	ID	ID	ID	ID	ID
addui r3,r30,32							IF	IF	IF	IF	IF	IF	IF	IF	IF	IF	IF	IF	IF
lw r2,28(r30)															IF	ID	ID	ID	ID
slli r2,r2,2															IF	IF	IF	IF	IF
addui r4,r3,r2																			

## 5 - Data Caches

### 5.1

Nombre de ligne dans le cache : 4 Taille d'un ligne dans le cache : 4 Taille totale du cache : 16 octets Accesses : 488 hits : 364 misses : 124 hit rate : 74,59loaded words : 124

### 5.2

Nombre de ligne dans le cache : 8 Taille d'un ligne dans le cache : 4 Taille totale du cache : 32 octets Accesses : 613 hits : 557 misses : 56 hit rate : 90,86Nombre de ligne dans le cache : 16 Taille d'un ligne dans le cache : 4 Taille totale du cache : 64 octets Accesses : 688 hits : 673 misses : 15 hit rate : 97,82Nombre de ligne dans le cache : 32 Taille d'un ligne dans le cache : 4 Taille totale du cache : 128 octets Accesses : 728 hits : 721 misses : 7 hit rate : 99,04Nombre de ligne dans le cache : 64 Taille d'un ligne dans le cache : 4 Taille totale du cache : 256 octets Accesses : 728 hits : 721 misses : 7 hit rate : 99,04La taille du cache est égale à la taille d'une ligne multipliée par le nombre de ligne. Plus le nombre de ligne augmente, donc la taille du cache augmente, plus le hit rate augmente. Les configurations avec 32 lignes et 64 lignes offrent le meilleurs hit rate. Mais on peut dire que la meilleure configuration est celle avec 32 lignes car la taille du cache est plus petite que celle avec 64 lignes.