Surname	
Name	
POLIMI Personal code	
Signature	

SOLUTION

Politecnico di Milano, 8 May 2023

Course on Advanced Computer Architectures

Prof. C. Silvano

EX 1	**(4 points)**	
EX 2	**(4 points)**	
EX 3	**(3 points)**	
EX 4	**(2 points)**	
QUIZ 5	**(1 point)**	
QUIZ 6	**(1 point)**	
TOTAL	**(15 points)**	
EX 7	**+ 3 extra points**	

EXERCISE 1: VLIW (4 points)

Let's consider the following FOR loop:

```
FOR:LD $F2, A($R1)
    LD $F4, B($R1)
    LD $F6, C($R1)
    FADD $F2,$F2,$F2
    SD $F2,B($R1)
    FADD $F4,$F4,$F4
    SD $F4,C($R1)
    FADD $F6,$F6,$F6
    SD $F6,D($R1)
    ADDUI $R1,$R1,4
    BNE $R1,$R2, FOR
```

Given a **4-issue VLIW** machine with **fully pipelined functional units:**

- **2 Memory Units with 2 cycle latency**
- **1 Integer ALU with 1 cycle latency to next Int/FP & 2 cycle latency to next Branch**
- **1 FP ALU with 3 cycle latency**

The branch is completed with 1 cycle delay slot (branch solved in ID stage). **No branch prediction.** In the Register File, it is possible to read and write at the same address at the same clock cycle. *Considering one iteration of the loop, complete the following table by using the **list-based scheduling*** (do NOT introduce any software pipelining, loop unrolling and modifications to loop indexes) on the *4-issue VLIW machine including the BRANCH DELAY SLOT.* Please do not write in NOPs.

	Memory Unit 1	Memory Unit 2	Integer Unit	Floating Point Unit
C1	`LD F2, A(R1)`	`LD F4, B(R1)`		
C2				
C3				
C4				
C5				
C6				
C7				
C8				
C9				
C10				
C11				
C12				
C13				
C14				
C15				

1. How long is the critical path? _____

2. What performance did you achieve in CPIas?

   _____

3. What performance did you achieve in FP ops per cycles?

   _____

4. How much is the code efficiency?

   _____

Feedback

	Memory Unit 1	Memory Unit 2	Integer Unit	Floating Point Unit
C1	`LD F2,A(R1)`	`LD F4,B(R1)`		
C2	`LD F6,B(R1)`			
C3				`FADD F2, F2, F2`
C4				`FADD F4, F4, F4`
C5				`FADD F6, F6, F6`
C6	`SD $F2,B(R1)`			
C7	`SD $F4,C($R1)`			
C8	`SD $F6,D($R1)`		`ADD R1,R1,4`	
C9				
C10			`BNE $R1,$R2, FOR`	
C11			`Br. delay slot`	
C12				
C13				
C14				
C15				

1. How long is the critical path?

   11 cycles

2. What performance did you achieve in CPIas?

   CPIas = (# cycles) / IC = 11 / 11 = 1

3. What performance did you achieve in FP ops per cycles?

   (# FP ops) / cycles = 3 / 11 = 0.27

4. How much is the code efficiency?

   Code_eff = IC/ (# cycles * # issues) = 11 / (11 * 4) = 0.25

EXERCISE 2 – TOMASULO (4 points)

Let's consider the following loop code to be executed on a CPU with dynamic scheduling based on **TOMASULO algorithm** with all cache HITS, a single Common Data Bus and:

- *2 RESERVATION STATIONS **(RS1, RS2)** for 2 LOAD/STORE UNITS **(LDU1, LDU2)** with latency **4***
- *2 RESERVATION STATION **(RS3, RS4)** for 2 ALU/BR FUs **(ALU1, ALU2)** with latency **2***
- Static Branch Prediction **BTFNT (BACKWARD TAKEN FORWARD NOT TAKEN)** with Branch Target Buffer

Please complete the following table:

INSTRUCTION	ISSUE	START EXEC	WRITE RESULT	Hazards Type	RSi	UNIT
`FOR1:beq $t6,$t7, END`	**1**	**2**	**4**	None	RS3	ALU1
`lw $t2,VECTA($t6)`						
`lw $t3,VECTB($t6)`						
`addi $t2,$t2,k`						
`sw $t2,VECTA($t6)`						
`add $t4,$t2,$t3`						
`sw $t4,VECTC($t6)`						
`addi $t6,$t6,4`						
`j FOR1`						

*Calculate the **CPI** and the **IPC:***

CPI = _____

IPC= _____

Feedback:

INSTRUCTION	ISSUE	START EXEC	WRITE RESULT	Hazards Type	RSi	UNIT
`FOR1:beq $t6,$t7, END`	**1**	**2**	**4**	None	RS3	ALU1
`lw $t2,VECTA($t6)`	2	3	7	(Control solved by BP-NT)	RS1	LDU1
`lw $t3,VECTB($t6)`	3	4	8	None	RS2	LDU2
`addi $t2,$t2,k`	4	8	10*	**RAW $t2**	RS4	ALU2
`sw $t2,VECTA($t6)`	8	11	15	**RAW $t2 + STRUCT RS1**	RS1	LDU1
`add $t4,$t2,$t3`	9	11	13	**RAW $t2** (RAW $t3 ok)	RS3	ALU1
`sw $t4,VECTC($t6)`	10	14	18	**RAW $t4**	RS2	LDU2
`addi $t6,$t6,4`	11	12	14	(WAR $t6 OK)	RS4	ALU2
`j FOR1`	14	15	17	**STRUCT RS3**	RS3	ALU1

(*) The result of ALU2 is forwarded through the CDB to RS1, RS3 and the RF.

***CPI** = # clock cycles / IC = 18 / 9 = 2*

***IPC** = 1/CPI = 1/ 2 = 0.5*

EXERCISE 3 – CACHE MEMORIES (3 points)

Let's consider 32-block main memory with a **2-way set associative** 8-block cache based on a ***write allocate*** with ***write-back*** protocol.

The addresses are expressed as:
Memory Address: $[0, 1, 2, \ldots\ldots\ldots 31]_{10}$
Cache Address: [a, b, c, d, e, f, g, h]
and Cache Tags are expressed as binary numbers.

At cold start the cache is empty, then there is the following sequence of memory accesses.
Please complete the following table:

	Type of memory access	Memory Address	HIT/MISS Type	Cache Tag	Cache Address	Set	Write in memory
1	Read	$[24]_{10}$	Cold-start Miss	$[110]_2$	a	$[0]_{10}$	No
2	Write	$[24]_{10}$	Hit	$[110]_2$	a	$[0]_{10}$	No
3	Read	$[14]_{10}$					
4	Read	$[04]_{10}$					
5	Write	$[14]_{10}$					
6	Write	$[24]_{10}$					
7	Read	$[12]_{10}$					
8	Read	$[21]_{10}$					
9	Write	$[18]_{10}$					
10	Read	$[02]_{10}$					

Feedback

The 2-way set-associative cache has 8-blocks [a, b, c, d, e, f, g, h] organized in 4 sets $[0, 1, 2, 3]_{10}$ where each set contains 2 blocks as follows:

Set_0: [a ,b]; Set_1: [c, d]; Set_2: [e, f]; Set_3: [g, h]

Being a 2-way set associative cache, the block replacement policy in each set uses the LRU algorithm.

Memory Address Mapping

S0	S1	S2	S3
0	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

	Type of memory access	Memory Address	HIT/MISS Type	Cache Tag	Cache Address	Set	Write in memory
1	Read	$[24]_{10}$	Cold-start Miss	$[110]_2$	a	$[0]_{10}$	No
2	Write	$[24]_{10}$	Hit	$[110]_2$	a	$[0]_{10}$	No
3	Read	$[14]_{10}$	Cold-start Miss	$[011]_2$	e	$[2]_{10}$	No
4	Read	$[04]_{10}$	Cold-start Miss	$[001]_2$	b	$[0]_{10}$	No
5	Write	$[14]_{10}$	Hit	$[011]_2$	e	$[2]_{10}$	No
6	Write	$[24]_{10}$	Hit	$[110]_2$	a	$[0]_{10}$	No
7	Read	$[12]_{10}$	Conflict Miss	$[011]_2$	b	$[0]_{10}$	No
8	Read	$[21]_{10}$	Cold-start Miss	$[101]_2$	c	$[1]_{10}$	No
9	Write	$[18]_{10}$	Cold-start Miss	$[100]_2$	f	$[2]_{10}$	No
10	Read	$[02]_{10}$	Conflict Miss	$[000]_2$	e	$[2]_{10}$	Yes Wr. in $M[14]_{10}$

EXERCISE 4 – SPECULATIVE TOMASULO WITH ROB (2 points)

Let's consider the following LOOP to be executed by a **Speculative Tomasulo** architecture with:

- 2 Load buffers (Load1, Load2);
- 2 FP ALU reservation stations (FP1 and FP2);
- 2 Integer reservation stations (INT1, INT2);
- 8-entry ROB (ROB0, ROB1, ….. , ROB7).

```
LOOP: LD $F0, 0 ($R1)
      FADD $F4, $F0, $F2
      FADD $F4, $F4, $F2
      SD $F4, AA ($R1)
      SUBI $R1, $R1, 8
      BNEZ $R1, LOOP   // branch prediction taken
EXIT:
```

Let's consider the ROB table by assuming that load of the first iteration is executing a cache miss. Let's continue to issue the next instructions until there are available resources or the ROB becomes full.
Please complete the ROB and the Rename Table

ROB

ROB#	Instruction	Dest.	Ready	Spec.	
ROB0	LD $F0, 0 ($R1) (1^ iteration exec. cache miss)	$F0	No	No	**HEAD**
ROB1					
ROB2					
ROB3					
ROB4					
ROB5					
ROB6					
ROB7					

Rename Table:

$F0	
$F2	
$F4	

Feedback

ROB

ROB#	Instruction	Dest.	Ready	Spec.	
ROB0	LD $F0, 0 ($R1) (1^ iteration exec. cache miss)	$F0	No	No	**HEAD**
ROB1	FADD $F4, $F0, $F2 (1^ iteration issued)	$F4	No	No	
ROB2	FADD $F4, $F4, $F2 (1^ iteration issued)	$F4	No	No	
ROB3	SD $F4, AA ($R1). (1^ iteration issued)	MEM	No	No	
ROB4	SUBI $R1, $R1, 8 (1^ iteration issued)	$R1	No	No	
ROB5	BNEZ $R1, LOOP (1^ branch predicted as taken)	--	No	No	
ROB6	LD $F0, 0 ($R1) (2^ iteration issued speculatively)	$F0	No	Yes	
ROB7					**TAIL**

Rename Table:

$F0	~~ROB0~~ ROB6
$F2	
$F4	~~ROB1~~ ROB2

We cannot issue instructions until the ROB becomes full because the 2 FP ALU reservation stations are busy. Therefore, the ROB7 is still empty because we cannot issue the FADD (speculatively). Notice that NO Store Buffers are needed in the Speculative Tomasulo architecture with ROB.

In the Rename Table, $F0 points to ROB6 because the WAW $F0 is solved, while $F4 points to ROB2 because the WAW $F4 is solved.

See also L07: Reorder Buffer & Speculation

EXERCISE 5 – DYNAMIC BRANCH PREDICTION (1 point)

Let's consider the following loop where $R1 is set to 0 and $R2 is set to 40:

```
LOOP:  LD $F0, 0($R1)
       FADD $F4, $F0, $F2
       SD $F4, 0($R1)
       ADDI $R1, $R1, 8
       BNE $R1, $R2, LOOP
```

Let's assume to have a 1-bit BHT as dynamic branch predictor initialized as TAKEN.
How much is the **misprediction rate** of this loop?
(SINGLE ANSWER)
1 point

Answer 1: 99%
Answer 2: 20%
Answer 3: 100%
Answer 4: 2 %
Answer 5: 25 %
Feedback
Answer 2: 20% (TRUE)
The LOOP is executed 5 times. Being the predictor initialized as TAKEN, there is 1 misprediction at the last iteration of the loop: 1 misprediction out of 5 predictions (20% misprediction rate and 80% success rate).

EXERCISE 6 – STATIC BRANCH PREDICTION (1 point)

Let's consider the following loop where $R1 is set to 0 and $R2 is set to 80:

```
LOOP:  LD $F0, 0($R1)
       FADD $F4, $F0, $F2
       SD $F4, 0($R1)
       ADDI $R1, $R1, 8
       BNE $R1, $R2, LOOP
```

Let's consider the BACKWARD TAKEN FORWARD NOT TAKEN static branch prediction.
How much is the **misprediction rate** of this loop?
(SINGLE ANSWER)
1 point

Answer 1: 99%
Answer 2: 10%
Answer 3: 100%
Answer 4: 1 %
Answer 5: 20 %

Feedback
Answer 2: 10% (TRUE)
The LOOP is executed 10 times. For this backward branch, the static prediction is TAKEN. Therefore, we have 1 misprediction only at the last iteration of the loop: 1 misprediction out of 10 predictions (10% misprediction rate and 90% success rate).

EXERCISE 7 – BRANCH PREDICTION (3 EXTRA points) -- OPTIONAL

Let's consider the following code where $R0 has been initialized to 0:

```
INIT:   ADDI $R1, $R0, 0
        ADDI $R2, $R0, 80
        ADDI $R3, $R0, 0
        ADDI $R4, $R0, 40

LOOP1: LD $F0, 0($R1)
        FADD $F4, $F0, $F2
        SD $F4, 0($R1)
        ADDI $R3, $R0, 0
LOOP2: LD $F6, 0($R3)
        FADD $F8, $F6, $F2
        SD $F8, 0($R3)
        ADDI $R3, $R3, 8
        BNE $R3, $R4, LOOP2

        ADDI $R1, $R1, 8
        BNE $R1, $R2, LOOP1
```

Answer to the following questions:

Question 1: How many iterations of LOOP1 and LOOP2 are executed?
(1 point)

Question 2: Let's assume to have a 1 entry 1-bit BHT as dynamic branch predictor (where the two branch instructions collide) initialized as Taken.
How many mispredictions are we going to observe? Please explain.
(1 point)

Question 3: Let's assume to have a 1 entry 2-bit BHT as dynamic branch predictor (where the two branch instructions collide) initialized Strongly Taken.
How many mispredictions are we going to observe? Please explain.
(1 point)

Question 1: How many iterations of LOOP1 and LOOP2 are executed?
(1 point)

Feedback
The outer loop LOOP1 is executed 10 times.
The inner loop LOOP2 is executed 5 times for each iteration of LOOP1
=> Globally LOOP2 is executed 50 times.

Question 2: Let's assume to have a 1 entry 1-bit BHT as dynamic branch predictor (where the two branch instructions collide) initialized as Taken.
How many mispredictions are we going to observe? Please explain
(1 point)

Feedback
Being the predictor initialized as TAKEN, we have a misprediction only at the last iteration (exit) of the inner LOOP2 and the prediction bit is turned to NT. So, for LOOP 2, we have 1 misprediction out of 5 predictions (misprediction rate 20% for LOOP2). Exiting from the inner LOOP2 with the NT prediction generates a misprediction on the BNE-LOOP1 for 9 iterations (except for the last iteration).

When re-entering in LOOP1, the prediction bit was turned to TAKEN when re-entering in the inner LOOP2 as before.

Counting the number of mispredictions, we have 1 misprediction for the BNE-LOOP2 only when exiting from LOOP2 for 10 iterations of the outer LOOP1 therefore 10 mispredictions. For the outer LOOP1, we have 9 mispredictions for BNE-LOOP1 (except for the last iteration). **Globally there are (10 + 9) = 19 mispredictions.**

Given **19 mispredictions** while the branch predictor has been used 60 times (50 times for BNE-LOOP2 and 10 times for BNE-LOOP1) => there are **19 mispredictions out of 60 predictions => 31.67% misprediction rate.**

Question 3: Let's assume to have a 1 entry 2-bit BHT as dynamic branch predictor (where the two branch instructions collide) initialized as Strongly Taken.
How many mispredictions are we going to observe? Please explain.
(1 point)
Feedback
Being the predictor initialized as Strongly Taken, the predictor fails and changes to the Weakly Taken state only at the last iteration (exit) of the inner LOOP2, but the prediction bit is still Taken So, for LOOP 2, we have 1 misprediction out of 5 predictions (misprediction rate 20% for LOOP2). Exiting from the inner LOOP2 as Weakly Taken, the prediction as TAKEN is correct for the BNE-LOOP1 for 9 iterations (except for the last iteration).

When re-entering in LOOP 1, the prediction state was turned to Strongly Taken when re-entering in the inner LOOP2 as before.

Counting the number of mispredictions, we have 1 misprediction for the BNE-LOOP2 only when exiting from LOOP2 for 10 iterations of the outer LOOP1 therefore 10 mispredictions. For the outer LOOP1, we have only 1 misprediction for BNE-LOOP1 at the last iteration. **Globally, there are (10 + 1) = 11 mispredictions.**

Given **11 mispredictions** while the branch predictor has been used 60 times (50 times for BNE-LOOP2 and 10 times for BNE-LOOP1) => there are **11 mispredictions out of 60 predictions => 18.33% misprediction rate.** As expected, the behavior of the 2-bit BHT is better than the previous 1-bit BHT.

Nome file: ACA_midterm_exam_2023_05_08_with_sol.docx
Directory: /Users/Cristina/Library/Containers/com.microsoft.Word/Data/Library/Preferences/AutoRecovery
Modello: /Users/Cristina/Library/Group Containers/UBF8T346G9.Office/User Content.localized/Templates.localized/Normal.dotm
Titolo:
Oggetto: Prof. C. Silvano
Autore: silvano
Parole chiave:
Commenti:
Data creazione: 19/09/16 18:51:00
Numero revisione: 128
Data ultimo salvataggio: 01/06/23 22:37:00
Autore ultimo salvataggio: Cristina Silvano
Tempo totale modifica 1.671 minuti
Data ultima stampa: 01/06/23 22:38:00
Come da ultima stampa completa
Numero pagine: 14
Numero parole: 2.333
Numero caratteri: 12.918 (circa)