University of Maryland Baltimore County
Department of Computer Science and Electrical Engineering

CMSC 611-101
Advanced Computer Architecture

Midterm Exam

Wednesday 10/28/09

Time allowed: 75 minutes

Student's Name: Sandipan Dey

Grade: $\frac{93}{100}$ +2 $\frac{95}{100}$

-5

(20 Points)

Question 1: *[15 minutes]*

Consider adding a new index addressing mode to MIPS. The addressing mode adds two registers and an 11-bit signed offset to get the effective address. Our compiler will be changed so that the following two code sequences will be replaced as indicated:

```
(1) ADD  R1, R1, R2
    LW   Rd, 100(R1)      }   LW   Rd, 100 (R1, R2)

(2) ADD  R1, R1, R2
    SW   Rd, 100(R1)      }   SW   Rd, 100 (R1, R2)
```

Use the instruction frequencies shown in the following table.

Instruction	load	store	add	sub	mul	compare	load imm	cond branch	jump	call	return	shift	and	or	other
Frequency	22.8%	14.3%	14.6%	0.5%	0.1%	12.4%	6.8%	11.5%	1.3%	1.1%	1.5%	6.2%	1.6%	4.2%	1.1%

A) Assume that the addressing mode can be used for 10% of the displacement loads and stores (accounting for both the frequency of this type of address calculation and the shorter offset). What is the ratio of instruction count on the enhanced MIPS compared to the original MIPS?

22.8
14.6
37.4

B) If the new addressing mode lengthens the clock cycle by 5%, which machine will be faster and by how much?

Answer:

A) Let's assume $IC_{original\ MIPS} = x = IC_{new}$

$\therefore \quad IC_{enhanced\ MIPS} =$

-5

$= 10\% \times \left(37.4\% \times \frac{x}{2} + 63.6\% \times x\right) + 90\% \times x.$

$= \left(\frac{0.0374}{2} + 0.636 + 0.9\right) x$

$\Rightarrow \frac{IC_{new}}{IC_{old}} = \frac{0.0187 + 1.536}{2} = \frac{1.6547}{2} = 0.827$

B) New Addressing mode lengthens clock cycle by 5%.

$C_{new} = 1.05\, C_{old}$

$\frac{Performance_{new}}{P_{old}} = \frac{Ex_{old}}{Ex_{new}} = \frac{IC_{old} \times CPI_{old} \times C_{old}}{IC_{new} \times CPI_{new} \times C_{new}}$

CPU time

Answer:

$$= \frac{IC_{old}}{IC_{new}} \times \frac{C_{old}}{C_{new}} \times \frac{CPI_{old}}{CPI_{new}}.$$

$$\cancel{= \frac{IC_{old}}{IC_{new}} \times \frac{1}{1.05}}$$

ok

$$\frac{C_{old}}{C_{new}} = \frac{1}{1.05}$$

$$= \frac{1}{0.827} \times \frac{1}{1.05}$$

$$\cancel{\frac{1.05}{0.827} = 1.27}$$

$$\frac{IC_{new}}{IC_{old}} = \frac{IC_{old} - 10\% \cdot (22.8 + 14.31)\, IC_{old}}{IC_{old}} = 0.96$$

CPI not affected

$$\underbrace{37.4\%}_{LD+SD}\Big(\overbrace{10\% \times \frac{x}{2}}^{halved} + \overbrace{90\% \times x}^{no\ improvement}\Big) + \underbrace{63.6\% \times x}_{NOT\ LD+SD}$$

$$= \underbrace{37.4\% \times 10\% \times \frac{x}{2}} + \underbrace{(37.4\% \times 90\% + 63.6\%)\, x}$$

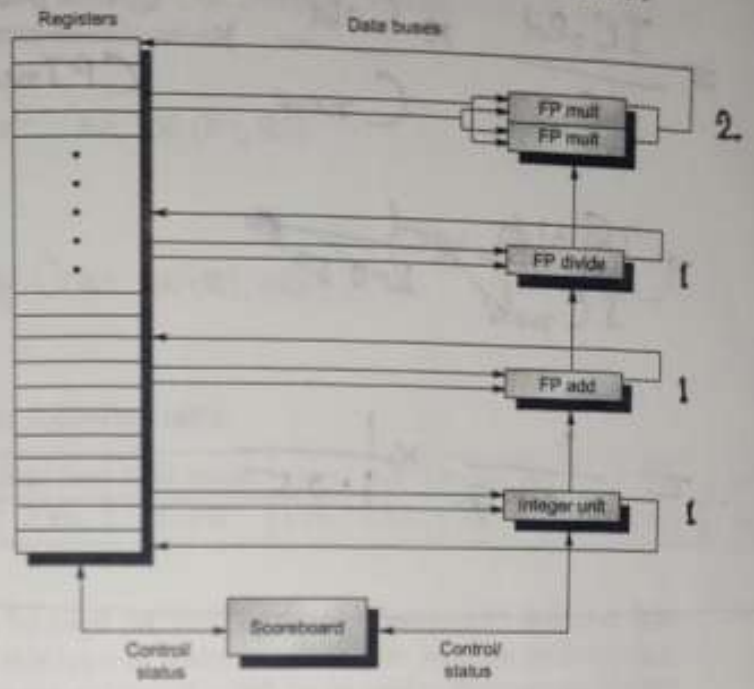
$$= 100\% \times x - 37.4 \times 10\% \times \frac{x}{2}$$

$$= IC_{old} - 10\% \cdot (22.8 + 14.31)\, \frac{IC_{old}}{2}$$

Question 2: *[30 minutes]* *(40 Points)*

Consider executing the following code on MIPS processor that uses a scoreboard with the FP functional units shown below.

1 L.D F6, 34(R2)
2 L.D F2, 45(R3)
3 MUL.D F0, F2, F4
4 DSUB.D F8, F6, F2
5 DIV.D F10, F0, F6
6 DADD.D F6, F8, F2



Assume the execution times indicated in the following table:

Operation	Cycles in EX stage
Load	1
Add	2
Multiply	10
Divide	40

A) How many cycles would the above code take to execute? (You need to have a table showing the cycle number at which each instruction enters the various stages).

B) Assume that we have added forwarding logic to the scoreboard. Repeat part (A) to capture the effect of this added feature.

Answer:

Possible RAW hazards (data dependencies): {2,3 1,4 1,5, 2,4 4,6 2,6}
WAR hazard (antidependency): (5,6)
WAW : (1,6)

Instruction Status

				Issue	Read Operand	Execution Completion	Write Result
LD	F_6	34	R_2	1	2	3	4
LD	F2	45	R3	5	6	7	8
MULD	F0	F2	F4	6	9	19	20
DSUB.D	F8	F6	F2	7	9	11	12
DIV.D	F10	F0	F6	8	21	61	62
DADD.D	F6	F8	F2	13	14	16	22 (WAR hazard)

1-4 1st L.D issue/read/exec/write, no other instrs will be issued since a structural hazard (integer unit)
5 2nd L.D issued
6 MULT.D. issued, L.D. continues
7 MULT.D can't read operand, data-dependent on 2nd load, which writes result at cycle 8, SUB issued
8 ~~MULT.D reads operands~~ DIV D issued (separate unit, no structural hazard) SUB.D/MULT.D can't read operands
9-12 DADD.D can't be issued (structural hazard → 1 FP Add unit)

Answer:

[redacted]

ith rwarding	Instruction Status	Issue	Read Operand	Execution Comp	Write
L.D.	F6 34 R2	1	2	3	4
L.D.	F2 45 R3	5	6	7	8
MULT.D	F0 F2 F4	6	8	18	19
DSUB.D	F8 F6 F2	7	8	10	11
DIV.D	F10 F0 F6	8	19	59	60
DADD.D	F6 F8 F2	12	13	15	20

(40 Points)

Question 3: *[30 minutes]*

The following is the MIPS code for the so-called SAXPY loop, which is the central operation in Gaussian elimination. The loop implements the vector operation Y = a . X + Y:

```
foo:  L.D     F2, 0(R1)      ; Load X(i)
      MULT.D  F4, F2, F0     ; multiply a by X(i)
      L.D     F6, 0(R2)      ; load Y(i)
      DADD.D  F6, F4, F6     ; add a * X(i) + Y(i)
      S.D     0(R2), F6      ; store Y(i)
      ADDIU   R1, R1, #8     ; increment X index
      ADDIU   R2, R2, #8     ; increment Y index
      SLTIU   R3, R1, done   ; test if done
      BEQZ    R3, foo        ; loop if not done
```

Assume that the contemporary 5 stages pipeline with integer ALU operations completed in one cycle. Control hazards require stalling the pipeline with branch condition evaluated and new address calculated in the ID stage. Assume that the results are fully bypassed. There are; one FP adder capable of performing addition and subtraction, and one FP multiplier unit for multiplication and division. Both the FP adder and multiplier are fully pipelined and take 4 clock cycles to perform their designated FP operation.

A) Show the stall cycles for each instruction and what clock cycle each instruction begins execution (i.e., enters its first EX cycle) on the first iteration of the loop.

B) Unroll the SAXPY loop to make four copies of the body and schedule it. When unwinding, you should reorder the code to maximize performance. You should show where stall cycles are needed (unavoidable despite your optimization).

Answer:

A)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
L.D F2, 0(R1)	IF	ID	EX	MEM	WB											
MULT.D F4, F2, F0		IF	ID	Stall	M1	M2	M3	M4	MEM	WB						
L.D F6, 0(R2)			IF	Stall	ID	EX	MEM	WB								
DADD.D F6, F4, F6					IF	ID	Stall	Stall	A1	A2	A3	A4	MEM	WB		
S.D 0(R2), F6						IF	Stall	Stall	ID	EX	Stall	Stall	MEM	WB		
ADDIU R1, R1, #8								IF	ID	Stall	Stall	EX	MEM	WB		
ADDIU R2, R2, #8									IF	Stall	Stall	ID	EX	MEM	WB	
SLTIU R3, R1, done																
BEQZ R3, foo																

M ≡ MEM S ≡ Stall ↘ Forwarding

M1 ≡ Mult Ex W ≡ WB

A1 ≡ Add Ex

Answer:

A)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
L.D	IF	ID	Ex	M	W													
MULT.D		IF	ID	S	M1	M2	M3	M4	M	W								
L.D			IF	S	ID	Ex	M	W	A1	A2	A3	A4	M	W				
DADD.D					IF	ID	S	S	ID	Ex	S	S	S	M	W			
S.D						IF	S	S	IF	ID	S	S	S	S	Ex	M	W	
ADDIU										IF	S	S	S	S	ID	Ex	M	W
ADDIU															IF	ID	Ex	M W
SLTIU																IF	S	ID M
BEQZ																		

~~foo: L.D.~~ ~~foo: L.D. F2, 0(R1)~~

~~MULT.D~~ ~~L.D~~ F2 – F6 | ~~L.D. F6, 0(R2)~~

F8 – F12 | ~~L.D. F8, 8(R1)~~

F14 – F18 | ~~L.D. F~~

F20 – F24

```
foo:  L.D.   F2, 0(R1)
      L.D.   F8, 8(R1)
      L.D.   F14, 16(R1)
      L.D.   F20, 24(R1)
      MULT.D F4, F2, F0
      MULT.D F10, F8, F0
      MULT.D F16, F14, F0
      MULT.D F22, F20, F0
      L.D.   F6, 0(R2)
      L.D.   F12, 8(R2)
      L.D.   F18, 16(R2)
      L.D.   F24, 24(R2)
      DADD.D F6, F4, F6
      DADD.D F12, F10, F12
      DADD.D F18, F16, F18
      DADD.D F24, F22, F24
      S.D.   0(R2), F6
      S.D.   8(R2), F12
      S.D.   16(R2), F18
      ~~S.D. 24(R2), F24~~
      ADDIU  R1, R1, #32
      ~~ADDIU R2, R2, #32~~
      SLTIU  R3, R1, done
      ~~BEQZ R3, foo~~
      ADDIU  R2, R2, #32
      BEQZ   R3, foo
      S.D.   -8(R2), F24
```

Assuming # times the loop executes is multiple of 4.

Stall cycles are not needed