McGill University
ECSE 425
COMPUTER ORGANIZATION AND ARCHITECTURE
Fall 2011
Midterm Examination

SOLUTION

10:35-11:25, October 12th, 2011

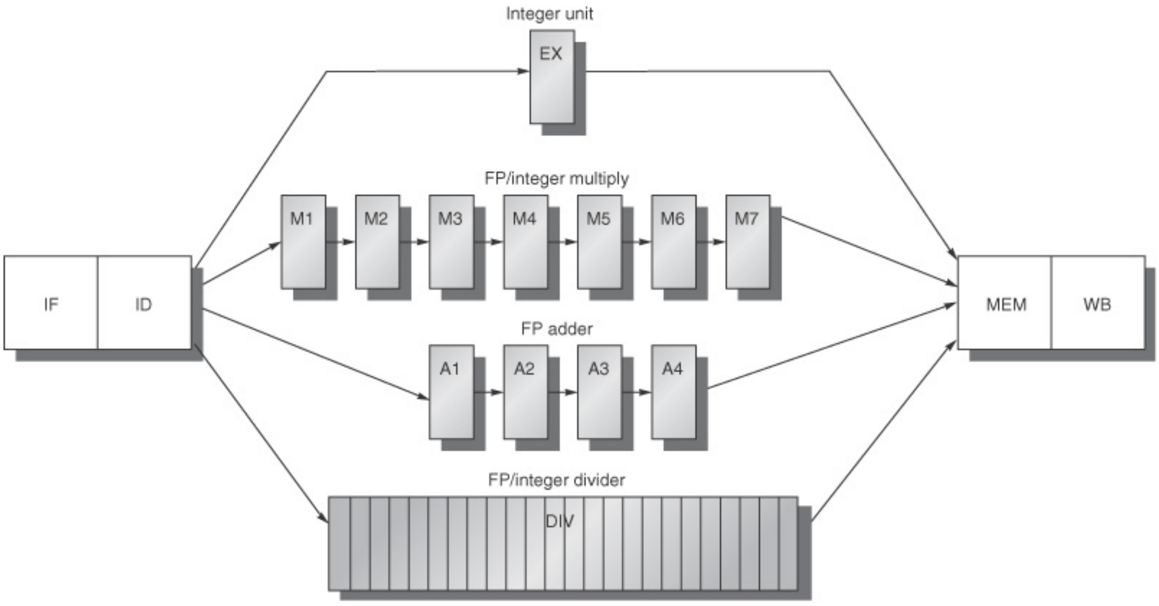
Duration: 50 minutes

- **Write your name and student number in the space below. Do the same on the top of each page of this exam.**
- **The exam is 8 pages long. Please check that you have all 8 pages.**
- **There are four questions for a total of 100 points. Not all parts of all questions are worth the same number of points; read the whole exam first and spend your time wisely!**
- **This is a closed-book exam. You may use one double-sided sheet of notes; please turn this sheet in with your exam.**
- **Calculators are permitted, but no cell phones or laptops are allowed.**
- **Clearly state any assumptions you make.**
- **Write your answers in the space provided. Show your work to receive partial credit, and clearly indicate your final answer.**

Question 1: MIPS Floating Point Pipeline (30 pts) Consider the following loop that computes the dot product of two vectors.

```
Loop:  L.D      F2, 0(R1)
       L.D      F4, 0(R2)
       MULT.D   F2, F2, F4
       ADD.D    F0, F0, F2
       DADDUI   R1, R1, #8
       DADDUI   R2, R2, #8
       SUBUI    R4, R3, R1
       BNEZ     R4, Loop
```

This code will run on the standard MIPS floating point pipeline, illustrated below.



(a) (20 pts) Assume:

- Full hazard detection and forwarding logic;
- The register file supports one write and two reads per clock cycle;
- Branches are resolved in ID;
- Branches are handled by flushing the pipeline;
- Memory accesses take one clock cycle;
- Structural hazards are resolved by giving priority to older instructions.

Fill in the chart on the next page to show the timing of one loop iteration.

Name:

ID:

Instr.	**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**
LD.D	F	D	E	M	W																				
LD.D		F	D	E	M	W																			
MULT.D			F	D	**stl**	M1	M2	M3	M4	M5	M6	M7	M	W											
ADD.D				F	**stl**	D	**stl**	**stl**	**stl**	**stl**	**stl**	**stl**	A1	A2	A3	A4	M	W							
DADDUI						F	**stl**	**stl**	**stl**	**stl**	**stl**	**stl**	D	E	M	W									
DADDUI													F	D	E	M	W								
SUBUI														F	D	E	M	**stl**	W						
BNEZ															F	D	**stl**	**stl**	E	M	W				
BNEZ+4																F	**stl**	**stl**	flsh						
LD.D																			F	D	E	M	W		

(b) (3 pts) If initially R3-R1 = 160, how many cycles are required to execute the loop given the assumptions in part (a)?

- Each iteration decrements R3 by 8. 160/8 = 20; there are 20 iterations.
- We observer in part (a) that the second iteration begins executing at cycle 19; each iteration requires 18 cycles. 20 * 18 = 360 cycles in total
- Since the branch results in a flush, each iteration has the same behavior (no difference in timing results from misprediction, since we aren't predicting).

(c) (3 pts) What about the MIPS FP pipeline makes maintaining precise exceptions challenging? Give a brief example.

Because the different FP functional units have different latencies, instructions can complete out of order. As a result, a later instruction may finish, and irrevocably modify system state, before it is known that an earlier instruction will cause an exception. When the exception occurs, the effective state at the end of that earlier instruction cannot be restored; this is known as an imprecise exception.

Consider the following instruction sequence:

I1: DIVD F0, F2, F4
I2: ADDD F6, F6, F8

ADDD will issue and complete prior to DIVD. Since ADDD destroys an operand (F6), if DIVD causes an exception, the state as of the beginning of I1 cannot be restored.

(d) (2 pts) Give two reasons why maintaining precise exceptions is important.

- It is required for virtual memory: without precise exceptions, a processor can't resume after an exception is handled, since the state of the processor may be incorrect.
- The IEEE FP standard requires it.
- It is helpful for debugging: without precise exceptions, diagnosing the cause of an exception during debugging is challenging, since later instructions may have modified the value of registers used by earlier instructions.

(e) (2 pts) *Briefly* describe one way of extending the MIPS FP pipeline to support precise exceptions.

- Save operands before they are used (history file).
- Hold results before they are committed (future file, similar to re-order buffer).
- Prevent FP operations from overlapping.
- Only allow FP operations to overlap when it can be shown that currently executing FP operations will not cause an exception.

Question 2: Loop Unrolling and Code Scheduling (30 pts) Consider the code in Question 1 once more.

```
Loop:  L.D      F2, 0(R1)   I1
       L.D      F4, 0(R2)   I2
       MULT.D   F2, F2, F4  I3
       ADD.D    F0, F0, F2  I4
       DADDUI   R1, R1, #8  I5
       DADDUI   R2, R2, #8  I6
       SUBUI    R4, R3, R1  I7
       BNEZ     R4, Loop    I8
```

(a) (5 pts) Indicate which pairs of instructions have RAW, WAR, and WAW hazards in the above code segment (specify the pair of instructions and hazard type).

Identifying hazards is different from identifying stalls. In this problem, we find all dependencies, independent of whether or not they cause a stall in practice.

RAW: (I1→I3), (I2→I3), (I3→I4), (I7→I8), (I5→I7)
WAR:
WAW: (I3→I1)

(b) (5 pts) Perform register renaming on the above code segment to eliminate as many of the hazards in part (a) as possible without violating true dependencies.

Loop:	L.D	F2, 0(R1)
	L.D	F4, 0(r2)
	MULT.D	**F6**, F2, F4
	ADD.D	F0, F0, **F6**
	DADDUI	R1, R1, #8
	DADDUI	R2, R2, #8
	SUBUI	R4, R3, R1
	BNEZ	R4, Loop

Name: ID:

Question 3: Amdahl's Law, Cost, and Yield (20 pts)

Three enhancements with the following speedups are proposed for a new architecture: S_1=2, S_2=8, S_3=32. The enhancements are mutually exclusive, and may be applied 40%, 20% and 10% of the time, respectively.

(a) (10 pts) What is the best speedup that can be achieved using any pair of enhancements?

$\text{Speedup}(S_x, S_y) = ((1 - F_x - F_y) + 1/S_x + 1/S_y)^{-1}$

$\text{Speedup}(S_1, S_2) = ((1 - 0.4 - 0.2) + 0.4/2 + 0.2/8)^{-1} = 1.6$
$\text{Speedup}(S_1, S_3) = ((1 - 0.4 - 0.1) + 0.4/2 + 0.1/32)^{-1} = 1.42$
$\text{Speedup}(S_2, S_3) = ((1 - 0.2 - 0.1) + 0.2/8 + 0.1/32)^{-1} = 1.37$

Where speedup is concerned, S_1 and S_2 is the best combination.

(b) (10 pts) Unenhanced, the architecture requires 200 mm^2. Enhancements 1, 2 and 3 increase the area of the processor by 10, 25, and 100 mm^2 respectively. When multiple metrics are important, we can combine them to determine the best possible trade-off. Determine which pair of enhancements strikes the best trade-off in performance improvement and yield by maximizing the objective function:

$$\text{Speedup} \times \text{Yield}.$$

Assume that wafer yield is 1, defect density is 0.04/cm^2, and $\alpha = 4$.

The first step is to convert the areas of each processor we're evaluating to cm^2 so that the units match the units of our given defect density (/cm^2).

$\text{Area}(S_1, S_2) = 200\text{ mm}^2 + 10\text{ mm}^2 + 25\text{ mm}^2 = 235\text{ mm}^2 = 2.35\text{ cm}^2$
$\text{Area}(S_1, S_3) = 200\text{ mm}^2 + 10\text{ mm}^2 + 100\text{ mm}^2 = 310\text{ mm}^2 = 3.1\text{ cm}^2$
$\text{Area}(S_2, S_3) = 200\text{ mm}^2 + 25\text{ mm}^2 + 100\text{ mm}^2 = 325\text{ mm}^2 = 3.25\text{ cm}^2$

Now we can calculate the yield of each combination of enhancements.

$\text{Yield}(S_1, S_2) = (1 + 0.25 * 0.04 * 2.35)^{-4} = 0.91$
$\text{Yield}(S_1, S_3) = (1 + 0.25 * 0.04 * 3.10)^{-4} = 0.89$
$\text{Yield}(S_2, S_3) = (1 + 0.25 * 0.04 * 3.25)^{-4} = 0.88$

And finally, we can calculate Speedup × Yield (SXY) for each pair.

$\text{SXY}(S_1, S_2) = 1.6 * 0.91 = 1.72$
$\text{SXY}(S_1, S_3) = 1.42 * 0.89 = 1.26$
$\text{SXY}(S_2, S_3) = 1.37 * 0.88 = 1.21$

Where speedup and yield are concerned, S_1 and S_2 are the best combination.

Question 4: Branch Prediction (20 pts) Consider a simple program with one branch. Given the different predictors and the following sequence of branch outcomes, complete the tables below.

(a) (10 pts) What is the branch prediction accuracy of a 1-bit predictor? Assume all predictor state is initialized to "not taken," or N.

Prediction	Outcome	Update
N	T	**T**
T	T	**T**
T	T	**T**
T	N	**N**
N	T	**T**
T	N	**N**
N	T	**T**
T	T	**T**
T	T	**T**
T	N	**N**
N	T	**T**
T	N	**N**

4/12 correct, or 33% accuracy.

(b) (10 pts) What is the branch prediction accuracy of a (1, 1) correlating predictor? The branch history register (BHR) is used to store the direction of the last branch. Assume all predictor state is initialized to "not taken," or N.

Pred.: last branch N	Pred.: last branch T	BHR	Pred.	Outcome	Update: last branch N	Update: last branch T
N	N	N	**N**	T	**T**	**N**
T	**N**	**T**	**N**	T	**T**	**T**
T	**T**	**T**	**T**	T	**T**	**T**
T	**T**	**T**	**T**	N	**T**	**N**
T	**N**	**N**	**T**	T	**T**	**N**
T	**N**	**T**	**N**	N	**T**	**N**
T	**N**	**N**	**T**	T	**T**	**N**
T	**N**	**T**	**N**	T	**T**	**T**
T	**T**	**T**	**T**	T	**T**	**T**
T	**T**	**T**	**T**	N	**T**	**N**
T	**N**	**N**	**T**	T	**T**	**N**
T	**N**	**T**	**N**	N	**T**	**N**

7/12 correct, or 58% accuracy.