CpE 313
Microprocessor System Design
Practice Exam 2

Instructions

Read each individual problem appearing on this exam carefully and do only what is specifically stated.

This exam is designed to be completed by a well-prepared student in approximately **75 minutes**; most students are expected to finish it within the allotted time. On your initial pass through this exam, skip any problems that appear to be overly difficult.

You can choose to turn in this exam after 24 hours or before. However, you will get 100% of the points you score if you turn in your exam within 75 minutes. You will get 80% of the points you score if you turn in within 24 hours after the exam.

IMPORTANT: Write your name at the TOP of EACH page. Also, be sure to read and sign the Academic Honesty Statement that follows:
"In signing this statement, I hereby certify that the work on this exam is my own and that I have not copied the work of any other student while completing this exam. I understand that, if I fail to honor this agreement, I will receive a score of ZERO for this exam and will be subject to possible disciplinary action."

Printed Name: ..... **Signature:** .....

Date: .....

DO NOT BEGIN UNTIL INSTRUCTED TO DO SO.

Precise Exceptions

Problem 1.

(a) List the different interrupts that can be raised in the 5-stage pipeline we studied in this course. Separate the interrupts according to the stage in which they can be raised. Indicate any interrupts that can be raised in any stage.
Page fault, misaligned access, and memory protection violation in stages IF and MEM. Illegal opcode in ID. Arithmetic exception in EXE. Stage independent interrupts can be due to power failure, user debugging, and from I/O devices.

(b) A particular pipelined processor services an interrupt as soon as the interrupting event occurs. Is this processor capable of precise interrupt handling? Why or why not? Explain with an example.
No. For example, in the instruction sequence

```
ld R2, 0(R4)
add R3, R5, R7
```

assume the ld instruction raises a page fault in the MEM stage and the add instruction raises a page fault for in the IF stage. This particular processor will service the add page fault before the ld page fault. This will be out-of-order interrupt processing, and hence not precise.

(c) Suggest how to modify the processor in **Problem 1(b)** so as to implement precise interrupts?
Assuming a 5-stage integer pipeline, if the interrupts are posted as soon as they occur (instead of being serviced as soon as they occur), and then serviced when the offending instruction reaches the WB stage, all interrupts will be precise.

Dynamically Scheduled Pipelines

Problem 2.

(a) What is dynamic scheduling? How is it different from static scheduling?
Scheduling or instruction scheduling, in general, is the re-ordering of instructions to avoid control, data, or structural hazards. Dynamic scheduling is the scheduling done by hardware, e.g., when Tomasulo's algorithm stalls a dependent instruction but issues a non-dependent one. Static scheduling is the scheduling done by the compiler. This happens before the processor receives the instruction stream.

(b) Illustrate with a simple example (consisting of no more than three instructions) the particular drawback of the simple 5-stage pipeline that is fixed by dynamic scheduling.
In a simple pipeline, if one instruction is stalled in ID stage, then all of the later instructions are stalled as well, whether they are dependent on the executing instructions or not. Dynamic scheduling allows later independent instructions to bypass a stalled instruction.

Thornton's Scoreboard

Problem 3.

Consider the instruction sequence given below. These instructions are to be executed by dynamically scheduled single-issue pipeline that is managed by Thornton's scoreboard.

```
mul F2, F1, F4
xor F2, F5, F7
sub F8, F2, F3
```

Assume that instruction `mul F2, F1, F4` is already being executed during clock cycle x, and will finish execution at the end of clock cycle $x + 5$.

(a) Will `xor` be fetched in clock cycle $x + 1$? If yes, will `xor` be issued in clock cycle $x + 2$? *Explain* your answers.
The instruction **`xor`** *will be fetched in CC* $x + 1$ *but will not finish the issue stage in CC* $x + 2$ *because at that time there is still an active instruction, i.e.,* **`mul`** *that has a WAW hazard with* **`xor`**. *Scoreboard avoids WAW hazards by stalling WAW a dependent instruction in the issue stage.*

(b) Will `sub` be fetched in clock cycle $x + 2$? If yes, will `xor` be issued in clock cycle $x + 3$? *Explain* your answers.
Because **`xor`** *is stalled in issue stage,* **`sub`** *will be not fetched at all.*

Tomasulo's Algorithm

Problem 4. List three ways in which a dynamically scheduled single-issue pipeline managed by Tomasulo's algorithm (TA) differs from a dynamically scheduled single-issue pipeline managed by Thornton's scoreboard (TS).
Item 1: TA employs "early reads," i.e., all available register operands are read at issue stage, and are buffered along with the instruction in "reservation stations." TS, however, employs "late reads," i.e., all operands are read from register file when all operands are ready. It only buffers the instruction.

Item 2: TA renames registers to avoid WAR hazards. Destination registers are renamed at "read pending operands" stage. The waiting consumer instructions replace a pending operands register specifier with a "tag" indicating the producer instruction's RS. In contrast, TS stalls the pipeline to avoid WAR.

Item 3: TA renames registers to avoid WAW hazards. TS, however, avoids a WAW hazard by stalling the WAW-dependent instruction in the issue stage. This feature stalls all instructions behind the WAW-dependent instruction even if they have no dependencies.

Tomasulo's Algorithm

Problem 5. For the code sequence below, show the register file and reservation station states for the first 16 clock cycles. Please assume that:
1) floating point add/sub take 2 cycles of execution
2) floating point mult takes 10 cycles of execution
3) there are three FP adder units and two mult units
4) WR takes one clock cycle.

You will need to fill in the work sheets on the next four pages. If you leave an entry blank, I will assume that its value is the same as that shown in the work sheet for the previous stage.

Omit the clock cycles for which the reservation station and register file states do not change. Do not feel obligated to use all work sheets. There are more work sheets here than you need.

The initial states of reservation stations and register file are shown in the work sheet below.

Instruction		j	k
ADDD	R5	R4	R2
SUBD	R3	R1	R2
MULTD	R3	R3	R5

finishing times

IS	RPO	EXE	WR

RS	RS Tag	Busy	Op	Vj	Vk	Qj	Qk
Add1	1						
Add2	2						
Mult1	3						
Mult2	4						

Integer Reg #	Q	V
1	0	100
2	0	0
3	0	31
4	0	1000
5	0	2000
6	0	3000
7	0	49

Figure 1: Initial State

Instruction	j	k
ADDD R5	R4	R2
SUBD R3	R1	R2
MULTD R3	R3	R5

finishing times

IS	RPO	EXE	WR

RS	RS Tag	Busy	Op	Vj	Vk	Qj	Qk
Add1	1						
Add2	2						
Mult1	3						
Mult2	4						

Integer Reg #	Q	V
1		
2		
3		
4		
5		
6		
7		

Figure 2: Clock cycle .....

Instruction	j	k
ADDD R5	R4	R2
SUBD R3	R1	R2
MULTD R3	R3	R5

finishing times

IS	RPO	EXE	WR

RS	RS Tag	Busy	Op	Vj	Vk	Qj	Qk
Add1	1						
Add2	2						
Mult1	3						
Mult2	4						

Integer Reg #	Q	V
1		
2		
3		
4		
5		
6		
7		

Figure 3: Clock cycle .....

Instruction	j	k
ADDD R5	R4	R2
SUBD R3	R1	R2
MULTD R3	R3	R5

finishing times

IS	RPO	EXE	WR

RS	RS Tag	Busy	Op	Vj	Vk	Qj	Qk
Add1	1						
Add2	2						
Mult1	3						
Mult2	4						

Integer Reg #	Q	V
1		
2		
3		
4		
5		
6		
7		

Figure 4: Clock cycle .....

Instruction	j	k
ADDD R5	R4	R2
SUBD R3	R1	R2
MULTD R3	R3	R5

finishing times

IS	RPO	EXE	WR

RS	RS Tag	Busy	Op	Vj	Vk	Qj	Qk
Add1	1						
Add2	2						
Mult1	3						
Mult2	4						

Integer Reg #	Q	V
1		
2		
3		
4		
5		
6		
7		

Figure 5: Clock cycle .....

Instruction	j	k
ADDD R5	R4	R2
SUBD R3	R1	R2
MULTD R3	R3	R5

finishing times

IS	RPO	EXE	WR

RS	RS Tag	Busy	Op	Vj	Vk	Qj	Qk
Add1	1						
Add2	2						
Mult1	3						
Mult2	4						

Integer Reg #	Q	V
1		
2		
3		
4		
5		
6		
7		

Figure 6: Clock cycle .....

Instruction	j	k
ADDD R5	R4	R2
SUBD R3	R1	R2
MULTD R3	R3	R5

finishing times

IS	RPO	EXE	WR

RS	RS Tag	Busy	Op	Vj	Vk	Qj	Qk
Add1	1						
Add2	2						
Mult1	3						
Mult2	4						

Integer Reg #	Q	V
1		
2		
3		
4		
5		
6		
7		

Figure 7: Clock cycle .....

Instruction	j	k
ADDD R5	R4	R2
SUBD R3	R1	R2
MULTD R3	R3	R5

finishing times

IS	RPO	EXE	WR

RS	RS Tag	Busy	Op	Vj	Vk	Qj	Qk
Add1	1						
Add2	2						
Mult1	3						
Mult2	4						

Integer Reg #	Q	V
1		
2		
3		
4		
5		
6		
7		

Figure 8: Clock cycle .....

Instruction	j	k
ADDD R5	R4	R2
SUBD R3	R1	R2
MULTD R3	R3	R5

finishing times

IS	RPO	EXE	WR

RS	RS Tag	Busy	Op	Vj	Vk	Qj	Qk
Add1	1						
Add2	2						
Mult1	3						
Mult2	4						

Integer Reg #	Q	V
1		
2		
3		
4		
5		
6		
7		

Figure 9: Clock cycle .....

Re-Order Buffer

Problem 6. A dynamically scheduled single-issue pipeline managed by Tomasulo's algorithm may cause imprecise exceptions even if all exceptions are handled in the write-results stage.

(a) Write down a sequence of two instructions to illustrate the above issue.
Consider
`mul F0, F2, F4`
`add F1, F3, F5.`
Assume that `mul` *takes 10 CC to complete, and that* `add` *takes 5 CC to complete. Further assume that the* `mul` *raises an interrupt in CC 6 and add raises one in CC 3. If we handle all interrupts in the WB stage, we will end up handling the* `add` *interrupt before the* `mul` *interrupt. This handling would be out-of-order and therefore imprecise.*

(b) A re-order buffer adds a *commit* stage to a dynamically scheduled single-issue pipeline managed by Tomasulo's algorithm. Explain how the commit stage allows precise interrupts.
A re-order buffer ensures that the interrupt from a later instruction is not serviced until all interrupts from all prior instructions have been serviced. When instructions are issued, they are assigned a ROB entry at the tail of the queue. In every clock cycle, the instruction at the head of the ROB is examined to see if it has completed and has not raised an exception. If yes, the result of the instruction is written to the register file. If the instruction has raised an interrupt, then the entire ROB is flushed, interrupt serviced, and the offending instruction restarted.

Wide-Issue Microprocessors

Problem 7.

(a) How does an N-issue microprocessor differ from a single-issue microprocessor?
A single-issue microprocessor issues ONE instruction every clock cycle at best. So CPI can be 1 at best (CPI could be zero if the dependencies require that the instruction be stalled).
An N-issue microprocessor can issue N instructions in each cycle. At best, CPI can be as small as $1/N$.

(b) VLIW and superscalar are two types of wide-issue processors. What distinguishes a VLIW processor from a superscalar processor? Explain with the help of a diagram. *A superscalar processor is a dynamic issue N-issue processor. That is, once N instructions have been fetched, the decision to issue an instruction is made by hardware based on the dependencies of that instruction with those prior in fetch order, and those in execution.*
A VLIW processor is a static issue N-issue processor. That is, the compiler makes the decisions about which instructions to issue in which clock cycle. Once these decisions are made, the compiler packages the instructions in N-instruction long instruction packets, called VLIWs, and sends them to the processor.

Branch Prediction

Problem 8.

(a) With respect to UltraSPARC-III instruction fetch pipeline stages, indicate clearly the stalls that branch prediction can avoid.
The UltraSPARC-III instruction fetch pipeline has 7 stages. The branch target address is known at the end of the fourth stage, B. The branch direction is known at the end of the seventh stage, R. If a branch is not predicted, the instruction after the branch must be fetched when both branch direction and branch target are known, which happens after R. This is a penalty of 6 cycles. If a branch is predicted, the predicted instruction can be fetched after the stage B. This is a penalty of 3 cycles. So the branch prediction can avoid stalls that correspond to stages between B and R.