CSE 490/590 Computer Architecture
Quiz 2 Solutions

DIRECTIONS

- Time limit: 45 minutes (12pm - 12:45pm)
- There are 20 points plus 5 bonus points.
- This is a closed-book, no calculator, closed-notes exam.
- Each problem starts on a new page.
- Please use a pen, not a pencil. If you use a pencil, it won't be considered for regrading.
- Each problem also explains its grading criteria. "Atomic" means that you get either all the points or no point, i.e., there are no partial points.

Name:	
UBITName:	

Problem #	Score
1	
2	
3	
4	

1. List two limitations regarding VLIW. Explain what each limitation is. Note that we discussed five limitations in class.
(**Grading**: total 4 points; 2 points for each)

Answer:

Any two of the following five limitations.

(a) Object-code compatibility: necessary to recompile all code for every machine, even for two machines in same generation.

(b) Object code size: instruction padding wastes instruction memory/cache loop unrolling/software pipelining replicates code.

(c) Scheduling variable latency memory operations: caches and/or memory bank conflicts impose statically unpredictable variability.

(d) Knowing branch probabilities: profiling requires an significant extra step in build process.

(e) Scheduling for statically unpredictable branches: optimal schedule varies with branch path.

2. Explain what it means to support precise interrupts.
(**Grading**: total 2 points)

Answer:

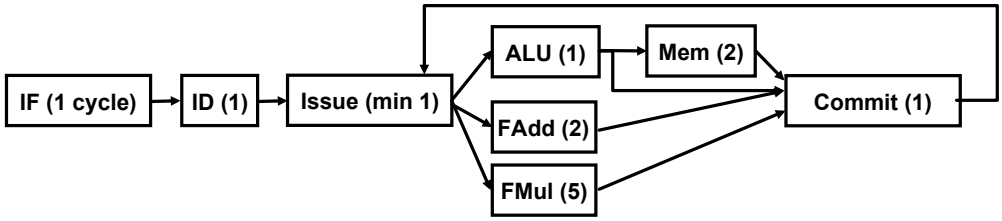
It must appear as if an interrupt is taken between two instructions (say I_i and I_{i+1}).

(a) The effect of all instructions up to and including I_i is totally complete

(b) No effect of any instruction after Ii has taken place

The interrupt handler either aborts the program or restarts it at I_{i+1}.

3. Suppose that you have the following out-of-order datapath.



Here is the list of assumptions on this processor:

(a) This processor has an *unlimited* reorder buffer. The unlimited reorder buffer is used to perform out-of-order execution, and since it is unlimited, *tags are never reused*, i.e., a new instruction always occupies a new tag in the reorder buffer.

(b) At the commit stage, the processor writes data back to a register in-order. The processor commits only one instruction per cycle, provided that an instruction is available for commit. Written register values can be read from the immediate next cycle. *When an instruction gets committed, it is deleted from the ROB*.

(c) As shown in the diagram above, it has 1-cycle ALU (for integer operations), 2-cycle MEM (for memory accesses), 2-cycle FAdd (for floating-point additions), 5-cycle FMul (for floating-point multiplications), and in-order fetch and commit. Each of IF, ID, and Commit takes 1 cycle.

(d) Adding an instruction to the ROB takes one cycle, i.e., the issue stage takes at least 1 cycle.

(e) The result from any functional unit becomes immediately available without any latency to the waiting instructions in the ROB. A waiting instruction can proceed from the immediate next cycle if its functional unit is available. For example, suppose *inst1* is using ALU and *inst2* is waiting for the result of *inst1* in the ROB. As soon as the ALU finishes *inst1* (say, at cycle 10), its result becomes immediately available to *inst2*, and *inst2* can be dispatched from the ROB and starts using its functional unit at cycle 11.

(f) The datapath does not support any branch or jump instructions.

Fill in *Src1* and *Src2* of the ROB (next page) for each of the following instructions with appropriate values, register names, or tags *at the time of adding each instruction to the ROB.*

Instruction Number	Instruction
I_1	LD F1, 0 (R1)
I_2	LD F2, 0 (R2)
I_3	FADD F3, F1, F2
I_4	LD F4, 0 (R3)
I_5	LD F5, 0 (R4)
I_6	FMUL F1, F3, F3
I_7	FADD F6, F4, F5
I_8	FMUL F5, F1, F6

Also fill in the start cycle and the end cycle for each instruction. The start cycle for an instruction indicates the time when the instruction gets added to the ROB. For example, the start cycle for the

very first instruction is always cycle 3 because IF and ID take 2 cycles in total. The end cycle for an instruction indicates the time when the instruction finishes its functional unit execution (or its memory access). For example, if the very first instruction is an FAdd operation, then the end cycle will be cycle 5.

(**Grading**: total 15 points; 3 points for each correct instruction)

Answer:

Tag	Op	Src1	Src2	Start Cycle	End Cycle
T1	LD	R1	0	3	6
T2	LD	R2	0	4	7
T3	FADD	T1	T2	5	9
T4	LD	R3	0	6	9
T5	LD	R4	0	7	10
T6	FMUL	T3	T3	8	14
T7	FADD	T4	T5	9	12
T8	FMUL	T6	T7	10	19

4. Gollum is a capable programmer who works for Google. He wrote a program that takes one web page as the input and outputs "yes" if the web page contains "my precious" in its text. He wants to run this program to process one million web pages. He can choose between two machines. These two machines are identical except that one machine uses an out-of-order superscalar CPU and the other uses a 2-way SMT CPU.

   The OoO superscalar machine can only run one thread and the 2-way SMT machine can run two threads simultaneously. However, due to the added complexity, the single thread performance for the 2-way SMT machine is worse than that of the OoO superscalar machine. On average, the superscalar machine is 1.2 times faster than the SMT machine when running one thread. However, with two threads, the SMT machine is 1.5 times faster than the superscalar machine.

   (a) Which machine should Gollum use in order to process the one million web pages he has as quickly as possible?
   (**Grading**: total 1 point)
   Answer: SMT

   (b) Explain your reasoning.
   (**Grading**: total 3 points)
   Answer: Suppose that P instructions/cycle is the performance of the SMT when it runs 1 thread (the unit could be different, but instructions/cycle is a good one to use here). Then the superscalar's performance is $1.2 \times P$.

   If we run 2 threads, either in sequence or at the same time, at best the superscalar's performance will be the same, so the best case performance is still $1.2 \times P$ (often times worse though). However, the SMT can run 2 threads at the same time and will improve the performance when it does that. The performance will be $1.5 \times 1.2 \times P$.

   With one million web pages to process, Gollum has plenty of threads to run. Thus, it's better if Gollum picks the SMT and runs two thread at a time to process one million web pages.