ECE 6913, Computing Systems Architecture

Spring 2020 NYU ECE

Please fill in your name: _____

Final, May 18th 2020

Maximum time: 80 minutes : **12:30 PM - 2:00 PM** [80 minutes + 10 minutes to assemble PDF and upload]

Open Book, Open Notes,

Calculators allowed.

Must show your work in steps – to get any credit

This is NOT a group project You may NOT discuss, share your Quiz solutions with anyone else.

You must stay logged in to Zoom throughout the Quiz

Instructor available online if you have questions on the Quiz, during the Quiz – enter question in Zoom chat box at any time during Quiz

This Test has 4 problems. Please attempt all of them. Please show **all work**. Please write **legibly**

1. Please be sure to have 5-10 sheets of white or ruled paper & a Pencil, Eraser
2. Write down your solutions on 8.5 x 11 sheets of white paper, single sided with your name printed in top right corner of each sheet and with **Page Number and Problem number identified clearly on each sheet**
3. **Stop working on your Quiz at 1:50 PM** – you have 10 minutes to scan/take pictures of each sheet and upload them as completed PDF assignment to NYU Classes – you may use any of several smartphone apps to integrate your scans/pictures of sheets into a PDF file
4. Take pictures of each sheet and **upload** the PDF of all sheets after checking you have all sheets in the right order **by 2:00 PM latest.**
5. You may use iPAD to write down your solutions directly rather than on paper
6. Portal **will close at 2:00 PM** not allowing upload of your quiz after 2:00 PM

1. Students M, O & P are building custom hardware and compilers for their project

The following observations are known. CPI = 1

• Student O has built his design and it has a known execution time for a new program of 600 seconds.

• Student M proposes a single cycle data path, and plans to use a compiler that will generate 300 Billion dynamic instructions per execution.

• Student P proposes a pipelined data path, with a clock period of 1.2 ns. His compiler will generate 400 Billion instructions. Of these 400 Billion instructions, 25% are loads, and 20% are branches. 40% of all loads cause a 2-cycle load-use stall. The penalty for a mis-predicted branch is 5 cycles. The processor has full bypassing, and does not suffer from any other stalls.

(i) How fast does M need to make his clock cycle in order to make the program run faster than O's?

(ii) How accurate does P's branch predictor need to be in order to make the program run faster than B's?

Answer:

$$x\frac{ns}{cycle} * 300B\frac{instructions}{execution} * 1\frac{cycle}{instruction} = 600\frac{seconds}{execution}$$

$$x = 2 \text{ ns}$$

His clock cycle must be shorter than 2 ns.

Answer:

$$1.2\frac{ns}{cycle} * 400B\frac{instructions}{execution} * (1 + (.25)(.40)(2) + (.20)(x)(5))\frac{cycles}{instruction} = 600\frac{seconds}{execution}$$

$$1.2\frac{ns}{cycle} * 400B\frac{instructions}{execution} * (1.2 + x)\frac{cycles}{instruction} = 600\frac{seconds}{execution}$$

$$x = .05$$

Her branch predictor must be greater than 95% accurate to make the program run faster.

2. In the conventional fully bypassed 5-stage pipeline discussed in class, we were able to make the assumption that the ALU is able to complete in one cycle because we assumed integer operations. For this problem, we will assume that the ALU takes two cycles to complete. In other words, *the ALU is pipelined itself, making the entire pipeline 6 cycles*. The ALU only *generates a result after 2 cycles* (i.e., there is no way to extract any meaningful result after the ALU's first cycle). Memory (instruction and data) still returns data after only one cycle. You may ignore branch and jump instructions in this problem.

As a first step, we will examine how this changes data hazards. For this question, we will examine only the ALU-ALU read after write (RAW) hazard where the two instructions are consecutive:

```
ADD x1, x2, x3          # x1 <= x2 + x3
ADD x5, x4, x1          # x5 <= x4 + x1
```

(i) Describe *how would you resolve this hazard with the minimum number of bubbles (if any) using a combination of data forwarding and stalls (if necessary).*

(ii) Then fill the following timing diagram to illustrate how the pipeline will behave, and show where data forwarding happens, if at all, by drawing an arrow between the two stages that participate in it.

	CC0	CC1	CC2	CC3	CC4	CC5	CC6	CC7	CC8
ADD x1, x2, x3									
ADD x5, x4, x1									

	CC0	CC1	CC2	CC3	CC4	CC5	CC6	CC7	CC8
ADD x1, x2, x3	IF	ID	ALU 1	ALU 2	MEM	WB			
NOP									
ADD x5, x4, x1			IF	ID	ALU 1	ALU 2	MEM	WB	

Data forwarding alone does not suffice here because by the time the first ALU instruction completes the second ALU (execute) step, the second ALU instruction is already in the first ALU step and thus it's too late.

Therefore, a NOP between the 2 instructions, as well as data forwarding between the second ALU step and the decode step.

3. Indicate if the following modifications (A,B,C) will cause each of the three metrics (three rightmost columns) to *increase*, *decrease*, or have *no effect*. Explain your reasoning

Assume the initial machine is pipelined. Also assume that any modification is done in a way that preserves correctness and maintains efficiency, but that the rest of the machine remains unchanged.

		Instructions/Program	CPI (Cycles/Instruction)	Circuit complexity
A	Replace the 2 operand ALU with a 3 operand one and add 3 operand register-register instructions to the ISA (for example, `ADD rs1,rs2,rs3,rd`)			
B	Use the same ALU for instructions and for incrementing the PC by 4			
C	Increase the number of user registers from 32 to 64			

		Instructions/Program	CPI (Cycles/Instruction)	Circuit complexity
A	Replace the 2 operand ALU with a 3 operand one and add 3 operand register-register instructions to the ISA (for example, `ADD rs1,rs2,rs3,rd`)	**Decrease** The new instructions will replace any two-instruction sequence that accomplished the same, such as `ADD rs1, rs2, rd` `ADD rs3, rd, rd`	**The same** The ALU will perform the three-way addition in one cycle. Also acceptable increase because more RAW	**Increase** Three-operand ALU is more complex than a two-operand one
B	Use the same ALU for instructions and for incrementing the PC by 4	**The same** No difference to instructions	**Increase** All instructions now use the same ALU ALU operations now have to stall	**Decrease** Now there is one less adder/ALU cycle
C	Increase the number of user registers from 32 to 64	**The same or decrease** If the more registers enable the Compiler to avoid loads and stores, Decrease. Otherwise the same.	**The same** Does not affect the actions of each instructions	**Increase** More registers complicate the register file

4. Assume that you have a computer with 1 clock cycle per instruction (CPI=1) when all accesses to memory are in cache. The only accesses to data come from load and store instructions. Those accesses account for 25 % of the total number of instructions. Miss penalty is 50 clock cycles and miss rate is 5 %. Determine the speedup obtained when there is no cache miss compared to the case when there are cache misses

Given:

- c_p: Processor cycles.
- c_w: Wait cycles.
- T: Clock period.
- IC: Amount of instructions o *Instruction Count.*
- CPI: Cycles per instruction.
- a_i: Amount of accesses produced by instructions.
- a_d: Amount of accesses produced by data.
- m: Miss rate.
- p_m: Miss penalty.

In case of not considering misses:

$$T_{cpu} = (c_p + c_w) \cdot T = (IC \cdot CPI + 0) \cdot T = IC \cdot 1 \cdot T = IC \cdot T$$

Including misses:

$$c_w = IC \cdot (a_i + a_d) \cdot m \cdot p_m = IC(1 + 1 \cdot 0{,}25) \cdot 0{,}05 \cdot 50 = IC \cdot 1{,}25 \cdot 0{,}05 \cdot 50 = 3{,}125$$

The CPU time will be:

$$T_{cpu} = (IC \cdot 1 + IC \cdot 3{,}125) \cdot T = 4{,}125 \cdot IC \cdot T$$

The speedup will be:

$$S = \frac{4{,}125 \cdot IC \cdot T}{IC \cdot T} = 4{,}125$$

5. Suppose that when Program A is run, the user CPU time is 3 seconds, the elapsed wallclock time is 4 seconds, and the system performance is 10 MFLOP/sec. Assume that there are no other processes taking any significant amount of time, and the computer is either doing calculations in the CPU, or doing I/O, but it can't do both at the same time. We now replace the processor with one that runs six times faster, but doesn't affect the I/O speed. What will the user CPU time, the wallclock time, and the MFLOP/sec performance be now?

$\text{CPU performance}_B/\text{CPU performance}_A = \text{CPU time}_A/\text{CPU time}_B$

$6 = 3/\text{CPU time}_B$

User CPU Time = .5 seconds

Since the I/O time is unaffected by the performance increase, it still takes 1 second to do I/O. Therefore it takes 1 + .5 = 1.5 seconds to run Program A on the faster CPU

Wallclock Time = 1.5 seconds

System Performance in MFLOPS = Number of Floating Point Operations * 10^6/Wallclock Time

Old System Performance (10) = #FLOP * $10^6/4$

#FLOP = 40 * 10^6

New System Performance = 40 * $10^6/1.5$

MFLOP/sec = 26.667