ECE 6913, Computing Systems Architecture

Spring 2021 NYU ECE

Please fill in your name: __________________________________

Quiz 2, April 16th 2021

Maximum time: 80 minutes : **11:00 AM** - **1:00 PM** [includes time to assemble PDF and upload]

Open Book, Open Notes,

Calculators allowed.

Must show your work in steps – to get any credit

You may NOT discuss, share your Quiz solutions with anyone else.

You must stay logged in to Zoom throughout the Quiz, with your camera on – you may leave after you upload your 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 5 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 12:50 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 1:00 PM latest.**
5. You may use iPAD to write down your solutions directly rather than on paper

Portal **will close at 1:00 PM** not allowing upload of your quiz after this time

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.

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

Tcycle * 300B instructions * CPI = Tcycle *300B * 1 < 600 secs

Tcycle < 600/300B = 2ns

2. 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: **cp:** Processor cycles. **cw:** Wait cycles. **T:** Clock period. **IC:** Number of instructions. **CPI:** Cycles per instruction. **a_i** : Amount of accesses produced by instructions. **a_d:** Amount of accesses produced by data. **mr:** Miss rate. **mp:** Miss penalty.

In case of no misses:

Tcpu = (cp + cw)·T = (IC·CPI + 0)· T = IC·1·T = IC·T

Including misses:

cw = IC·(a_i+a_d)·mr·mp

= IC(1 + 1·0.25)·0.05·50 = IC·1.25·0.05·50 = 3.125

So,

Tcpu = (IC·1 + IC·3.125)·T = 4.125·IC·T

Speedup = S = 4.125·IC·T/[IC·T] = 4.125

3. When Program X is run, the user CPU time is 3 seconds, the total wall clock 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 wall clock time, and the MFLOP/sec performance be now?

CPU performance$_B$ /CPU performance$_A$ = CPU time$_A$ /CPU time$_B$

6 = 3/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

Wall clock Time = 1.5 seconds

System Performance in MFLOPS =

Number of Floating-Point Operations * 10^6 / Wall clock Time

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

#FLOP = 40 * 10^6

New System Performance = 40 * 10^6 /1.5

= 26.67 MFLOP/sec

4. For the 5-stage pipeline, with forwarding hardware discussed in class, our assumption was that the ALU is able to complete in one cycle because we assumed integer operations. Here, we 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 one cycle. You may ignore branch and jump instructions in this problem.

Let's look at 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	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.

5. Your favorite Computer is described with the following features:
• 95% of all memory accesses are found in the cache.
• Each cache block is two words, and the whole block is read on any miss.
• The processor sends references to its cache at the rate of 10^9 words per second.
• 25% of those references are writes.
• Assume that the memory system can support 10^9 words per second, reads or writes.
• The bus reads or writes a single word at a time (the memory system cannot read or write two words at once).
• Assume at any one time, 30% of the blocks in the cache have been modified.
• The cache uses write allocate on a write miss.

You are considering adding a peripheral to the system, and *you want to know how much of the memory system bandwidth is already used*.

(a) Calculate the percentage of memory system bandwidth used assuming the cache is Write Back.
(b) Calculate the percentage of memory system bandwidth used assuming the cache is Write Through.
Be sure to state your assumptions.

We know:

* Miss rate = 0.05

* Block size = 2 words (8 bytes)

* Frequency of memory operations from processor = 109

* Frequency of writes from processor = 0.25 * 109

* Bus can only transfer one word at a time to/from processor/memory

* On average 30% of blocks in the cache have been modified (must be written back in the case of the write back cache)

* Cache is write allocate

So:

Fraction of read hits = 0.75 * 0.95 = 0.7125

Fraction of read misses = 0.75 * 0.05 = 0.0375

Fraction of write hits = 0.25 * 0.95 = 0.2375

Fraction of write misses = 0.25 * 0.05 = 0.0125

(a) Write through cache

- On a read hit there is no memory access
- On a read miss memory must send two words to the cache
- On a write hit the cache must send a word to memory
- On a write miss memory must send two words to the cache, and then the cache must send a word to memory

Thus:

Average words transferred = 0.7125 * 0 + 0.0375 * 2 + 0.2375 * 1 + 0.0125 * 3 = 0.35

Average bandwidth used = 0.35 * 10^9

Fraction of bandwidth used =

[0.35 x 10^9] / 10^9

= 0.35 (1)

(b) Write back cache

On a read hit there is no memory access

On a read miss:

1. If replaced line is modified then cache must send two words to memory, and then

memory must send two words to the cache

2. If replaced line is clean then memory must send two words to the cache

On a write hit there is no memory access

On a write miss:

1. If replaced line is modified then cache must send two words to memory, and then memory must send two words to the cache

2. If replaced line is clean then memory must send two words to the cache

Thus:

Average words transferred = 0.7125 * 0 + 0.0375 * (0.7 * 2 + 0.3 * 4) + 0.2375 * 0 + 0.0125 *

(0.7 * 2 + 0.3 * 4) = 0.13

Average bandwidth used = 0.13 * 10^9

Fraction of bandwidth used =

0.13 x $10^9/10^9$

= 0.13 (2)

Comparing 1 and 2 we notice that the write through cache uses more than twice the cache-memory bandwidth of the write back cache.