ECE 6913 Final Exam Instructions

Computing Systems Architecture

Fall 2021 NYU ECE

Time: December 16th 2021, 11:00 AM – 1:30 PM.

Maximum time: 150 minutes : **11:00 AM – 1:30 PM** [+ 10 minutes to assemble PDF and upload]

Open Book, Open Notes,

Calculators allowed.

The Final will be visible as an assignment on NYU Brightspace at 11:00 AM on December 16th 2021

You Must show your work in steps – to get any credit

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

You may not communicate with anyone during the Final except with Course Staff

By University rules, students are required to login to the Zoom session and turn on their video and audio. Your Final cannot be graded if you are not logged into the zoom session with your camera and Mic turned on

This Test has 5 problems. Please attempt all of them. Please show **all work**. Please write **legibly**

1. Please be sure to have 10-15 sheets of white or ruled Paper, a Pen/Pencil & Eraser
2. Please 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. **Please stop working on your Quiz at 1:30 PM** – you have 10 minutes to scan/take pictures of each sheet and upload them as completed PDF assignment to NYU Classes **by 1:40 PM** – you may use any of several smartphone apps to integrate your scans/pictures of sheets into a PDF file.
4. Please take pictures of each sheet and **upload** the PDF of all sheets after checking you have all sheets in the right order **by 1:45 PM latest.**
5. You may use iPAD to write down your solutions directly rather than on paper

Portal **will close at 1:45 PM** not allowing upload of your quiz after 1:45 PM

1. Compute the effective CPI for RISC-V using *Figure P1* below and *Table P1* below. Average the instruction frequencies of `perlbench` and `sjeng` (shown in red box in Fig P1) to obtain the instruction mix

Instruction	Clock cycles
All ALU operations	1.0
Loads	3.5
Stores	2.8
Branches	
Taken	4.0
Not taken	2.0
Jumps	2.4

Table PI: CPI for Instruction classes

Program	Loads	Stores	Branches	Jumps	ALU operations
astar	28%	6%	18%	2%	46%
bzip	20%	7%	11%	1%	54%
gcc	17%	23%	20%	4%	36%
gobmk	21%	12%	14%	2%	50%
h264ref	33%	14%	5%	2%	45%
hmmer	28%	9%	17%	0%	46%
libquantum	16%	6%	29%	0%	48%
mcf	35%	11%	24%	1%	29%
omnetpp	23%	15%	17%	7%	31%
perlbench	25%	14%	15%	7%	39%
sjeng	19%	7%	15%	3%	56%
xalancbmk	30%	8%	27%	3%	31%

Figure P1: RISC-V dynamic instruction mix for the SPECint2006 programs.

2. Assume that we make an enhancement to a computer that improves some mode of execution by a factor of 10. Enhanced mode is used 50% of the time, measured as a percentage of the execution time when the enhanced mode is in use. Recall that Amdahl's law depends on the fraction of the original, unenhanced execution time that could make use of enhanced mode. Thus, we cannot directly use this 50% measurement to compute speedup with Amdahl's law

(i) What is the speedup we have obtained from fast mode?

(ii) What percentage of the original execution time has been converted to fast mode?

3. Your company has just bought a new Intel Core i5 dual core processor, and you have been tasked with optimizing your software for this processor. You will run two applications on this dual core, but the resource requirements are not equal. The first application requires 80% of the resources, and the other only 20% of the resources. Assume that when you parallelize a portion of the program, the speedup for that portion is 2

(i) Given that 40% of the first application is parallelizable, how much speedup would you achieve with that application if run in isolation?

(ii) Given that 99% of the second application is parallelizable, how much speedup would this application observe if run in isolation?

(iii) Given that 40% of the first application is parallelizable, how much overall system speedup would you observe if you parallelized it?

4. A. This question considers the basic, RISC-V, 5-stage pipeline. (In this problem, you may assume that there is full forwarding.)

(i) Explain how pipelining can improve the performance of a given instruction mix

(ii) Show how these instructions will flow through the pipeline:

	1	2	3	4	5	6	7	8	9	10	11
lw x10, 0(x11)											
add x9, x11, x11											
sub x8, x10, x9											
lw x7, 0(x8)											
sw x7, 4(x8)											

(iii) Where might the sw instruction get its data from? Be very specific. (i.e. “from the lw instruction” is not a good answer!)

B. This question considers the basic, RISC-V, 5-stage pipeline. (In this problem, you may assume that there is full forwarding.) Show how these instructions will flow through the pipeline below. For the instruction mix above, on what instruction results does the last add instruction depend on? predict that the beq instruction is not taken

	1	2	3	4	5	6	7	8	9	10	11
beq x1, x2, X											
lw x10, 0(x11)											
sub x14, x10, x10											
X: add x4, x1, x2											
lw x1, 0(x4)											
sub x1, x1, x1											
add x1, x1, x1											

5. (i) A cache may be organized such that:

- In one case, there are more data elements per block and fewer blocks
- In another case, there are fewer elements per block but more blocks However, in both cases – i.e., larger blocks but fewer of them OR shorter blocks, but more of them – the cache's total capacity (amount of data storage) remains the same

What are the pros and cons of each organization? Support your answer with a short example assuming that the cache is direct mapped

(ii) Assume:

- A processor has a direct mapped cache
- Data words are 8 bits long (i.e., 1 byte)
- Data addresses are to the word
- A physical address is 20 bits long
- The tag is 11 bits
- Each block holds 16 bytes of data

How many blocks are in this cache?

(iii) Consider a 16-way set-associative cache:

- Data words are 64 bits long

- Words are addressed to the half-word

- The cache holds 2 Mbytes of data

- Each block holds 16 data words

- Physical addresses are 64 bits long

How many bits of tag, index, and offset are needed to support references to this cache?