Department of Electrical and Computer Engineering
The University of Texas at Austin

EE 382N.19, Spring 2022
Y. N. Patt, Instructor
Chester Cai, TA
Written Midterm Question Packet
March 23, 2022

There are 12 problems on this exam. Your job is to do #1 and #2 and any six of the remaining 10 (i.e., #3 through #12). You have been given a separate solution sheet, to provide your answers. You need to turn in only your solution sheet. Please make sure your name is on every sheet of paper on your solution sheet, and that you have clearly identified (by problem number) which six problems you have chosen to answer.

Question 1

In an out-of-order machine, being able to determine the relative order of two instructions can be very helpful. For example, when there is a branch misprediction, the processor needs to determine the relative ordering of the instructions to determine which ones should be flushed.

Assume that each instruction is assigned a ROB ID at rename which indicates its position in the ROB. The ROB is implemented as a circular buffer with a head and tail pointer both starting at 0. The head pointer points to the oldest instruction in the ROB, and the tail pointer points to the next available entry in the ROB. We increment the head pointer everytime we retire an instruction, and increment the tail pointer everytime we insert an instruction into the ROB. The ROB can hold up to 32 entries.

Your job: Finish the following verilog module that determines the relative ordering of the two instructions in the ROB. You are given the ROB ID of the two instructions and the head pointer. The output should be a 1 if inst0 is older than inst1 and 0 if inst0 is younger than inst1. You may assume inst0 and inst1 are different instructions, and are valid in the ROB.

```
module  your_module (
   input[4:0] inst0_rob_id,
   input[4:0] inst1_rob_id,
   input[4:0] head_ptr,

   output[0:0] result
)
```

You are allowed to use the following modules:

```
module comp_5b(out, in0, in1);        //out = 1 if in0 > in1, 0 if otherwise
module  mux2_1(out, in0, in1, sel);   //1-bit 2 to 1 mux
```

Any basic logic gate (inverter, or, and, xor) you need

Question 2

The contents of memory at starting location 0x3000 are shown below.

0x3000: 05 04 03 02

0x3004: 01 66 93 0F

0x3008: B0 FF 65 83

0x300C: 4C C3 FC F4

Note: Location 0x3000 contains 05, and location x300F contains F4.

Your job: Decode the x86 instructions specified by these bytes. Show the exact address calculation for any memory operand. Use M_32[ ] to indicate a 32 bit memory location. Immediates, displacements, and literals should begin with the $ symbol. Shown below is some example assembly. Please use this as a reference for notation.

```
OR M_32[(ES<<16) + EDX], $0x87654321
MOV AX, $0xAB87
DAA
```

Question 3

A problem can be solved in 8 units of time by an algorithm that consists of 24 operations, each requiring a single unit of time, if we have 4 processing elements available to handle the 24 operations. Since it would take 24 units of time if we only had only one processing element, we could say our speed up with 4 processing units is 24/8, or 3. Why is "a speed up of 3" an incorrect conclusion to make from the above information?

Question 4

The wish branch recognized that the decision to predicate or branch predict depended on the prediction accuracy of the branch predictor, which is only known at run-time. Yet we can immediately say at compile time that certain branches should not be predicated. Why?

Question 5

We say that the atomic unit of processing in the Block-Structured ISA is a basic block. What does that mean? What major benefit does that buy you in constructing a microprocessor to implement a Block-structured ISA?

Question 6

Scott McFarling improved the prediction accuracy of my GAg branch predictor with his invention of gshare. How did he get better prediction accuracy from gshare?

Question 7

Since DIVIDE instructions take far too much time to execute, Seymour Cray did not have a DIVIDE unit in the Cray I. Since scientific code has lots of divide instructions, how did he implement the divide instruction?

Question 8

With SMT, predication, and SIMD, GPUs can have tens of thousands of threads in progress every clock cycle. Unfortunately that benefit can be killed by branch divergence or memory coalescing. Explain.

Question 9

You are in charge of producing the next processor. You come up with a great idea that new technology allows you to improve performance substantially by implementing that idea in the ISA or in the microarchitecture. Which do you choose? Why?

Question 10

I have made the comment in class on several occasions that I do not believe the von Neumann model of processing is dead, contrary to what many people in computer architecture claim. On what do I base my reasoning?

Question 11

Out-of-order execution, which we showed in our 1985 first HPS papers, was a way to exploit instruction level parallelism. We coined our model "Restricted Data Flow." But it was not *real* data flow. What is the distinction?

Question 12

Each clock cycle, a CPU can fetch until a taken branch or until the end of a cache line. If there is often a taken branch instruction in the first few bytes of the cache line, the fetch unit would perform poorly. If we know the direction of branches in the program are highly predictable, what microarchitecture enhancement discussed in class would solve this problem? How would it solve the problem?