NYU Tandon School of Engineering

Fall 2022, ECE 6913 Section A

Quiz 2

1. [10 points] Convert the following high-level language script into RISCV code. Assume the signed integer variables `g` and `h` are in registers `x5` and `x6` respectively.

(i)

```
if (g > h)
     g = g + 1;
else
     h = h - 1;
```

(ii)

```
if (g <= h)
 g = 0;
else
 h = 0;
```

2. [10 points] A fast, energy efficient computer core minimizes (1) *the number of Instructions in the ISA*, (2) the *number of instructions in a Program* from that ISA and (3) *number of cycles per instruction* required to execute that Program. Compilers can actually minimize (2) by *using the least number of registers* leading us to minimize the likelihood of a 'spill' to memory if a program needs more registers than it has available.

Identify the best algorithm to swap 2 registers without using a third register and write a RISC-V program *for swapping the contents of two registers*, x5 and x6. You may not use any other registers

3. This problem explores energy efficiency and its relationship with performance. The parts of this problem assume the following energy consumption for activity in Instruction memory, Registers, and Data memory. You can assume that the other components of the datapath consume a negligible amount of energy. ("Register Read" and "Register Write" refer to the register file only.)

I-Mem	1 Register Read	Register Write	D-Mem Read	D-Mem Write
140pJ	70pJ	60pJ	140pJ	120pJ

Assume that components in the datapath have the following latencies. You can assume that the other components of the datapath have negligible latencies.

I-Mem	Control	Register Read or Write	ALU	D-Mem Read or Write
200 ps	150 ps	90 ps	90 ps	250 ps

3.1 [5 points] How much energy is spent to execute an **addi** instruction in a single-cycle design and in the five-stage pipelined design

3.2 [5 points] How much energy is spent to execute a **ld** instruction in a single-cycle design

3.3 [5 points] How much energy is spent to execute a **beq** instruction in a single-cycle design

4. 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
```

4.1 [10 points] 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).*

4.2 [10 points] 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									

5. [10 points] How many cycles are required to run the following program on the multicycle RISC-V processor? What is the CPI of this program?

```
addi s0, zero, 5 # result = 5
L1:
 bge zero, s0, Done # if result <= 0, exit loop
 addi s0, s0, -1 # result = result - 1
 j L1
Done:
```

6. [10 points] A pipelined RISCV processor is running this sequence of instructions shown below. Identify the registers being written and being read in the fifth cycle? This RISCV processor has a Hazard Unit. Assume a memory that returns data within a cycle.

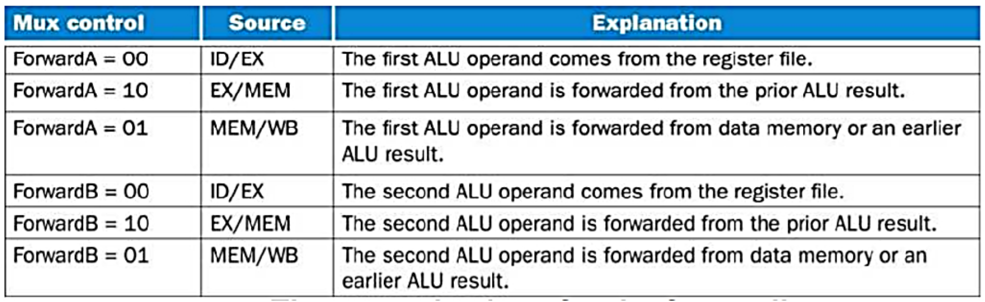
```
xor s1, s2, s3        # s1 = s2 ^ s3
addi s0, s3, -4       # s0 = s3 - 4
lw s3, 16(s7)         # s3 = memory[s7+16]
sw s4, 20(s1)         # memory[s1+20] = s4
or t2, s0, s1         # t2 = s0 | s1
```

7. Consider the following RISC V Instruction sequence executing in a 5-stage pipeline:

```
or   x13, x12, x11
ld   x10, 0(x13)
ld   x11, 8(x13)
add  x12, x10, x11
subi x13, x12, 16
```

7.1 [10 points] Identify all of the data hazards and their resolution with NOPs assuming no forwarding or hazard detection hardware is being used

7.2 [10 points] If there is forwarding, for the first seven cycles during the execution of this code, *specify which signals are asserted in each cycle by hazard detection and forwarding units* in Figure below.



Mux control	Source	Explanation
ForwardA = 00	ID/EX	The first ALU operand comes from the register file.
ForwardA = 10	EX/MEM	The first ALU operand is forwarded from the prior ALU result.
ForwardA = 01	MEM/WB	The first ALU operand is forwarded from data memory or an earlier ALU result.
ForwardB = 00	ID/EX	The second ALU operand comes from the register file.
ForwardB = 10	EX/MEM	The second ALU operand is forwarded from the prior ALU result.
ForwardB = 01	MEM/WB	The second ALU operand is forwarded from data memory or an earlier ALU result.

	Clock Cycle → Instruction ↓	1	2	3	4	5	6	7	8	9	10
1	`or`										
2	`ld`										
3	.										
.	.										
.	.										

8. [5points] 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			