Introduction to Computer Engineering (E114)

Harris Spring 2000

Final Exam

This is an open book, open notes final exam. Along side each question, the number of points is written in brackets. The entire exam is worth 100 points. Plan your time accordingly. The exam is divided into two components. One is a written component that you are responsible for timing yourself on. You may spend a maximum of 2.5 hours on the written component. The second is a lab component, attached on the last page of this exam booklet. The lab component not timed so if you have major tool problems you will not be penalized. If the tools seem to be acting up on you, you are free to ask Professor Harris for help. You may not discuss the lab component with anybody else.

All work and answers should be written directly on this examination booklet, except the lab component printouts, which should be stapled to the exam. Use the backs of pages if necessary. Write neatly; illegible answers will be marked wrong.

Show your work for partial credit. Show your name for any credit.

Remember that you may receive the solutions before others have taken the exam. You are on the honor code not to share information about the exam with students who have not taken it.

Senior exams are due in Prof. Harris' mailbox by 5 p.m. on Thursday May 4th. All other exams are due by 5 p.m. on Thursday May 11th.

Name: _____

Do Not Write Below This Point

2) _____ /10

3) _____ /10

4) _____ /10

5) _____ /8

6-7) _____ /18

8) _____ /9

9-10) _____ /20 Lab Component

Total: _____ /85

MIPS Assembly Language Programming

[10] Translate the following program into MIPS assembly language. Pay particular attention to properly saving and restoring registers. Clearly comment your code. You may use the MIPS `mul` instruction.

```
int series(int n, int c)
{
     int a;

     if (n < 2) a = 1;
     else a = series(n-1, c) + series(n-2, c);
     return a*c;
}
```

Data Hazards

Figure 6.46 of the book shows a pipelined processor with a forwarding unit and hazard detection unit to handle data hazards.

a) [2] Give an example of a two-line assembly language program that causes a data hazard that is resolved by bypassing instructions with the Forwarding Unit.

(two-line program)

b) [2] Give an example of a two-line assembly language program that causes a data hazard that is resolved by stalling the pipeline with the Hazard Detection Unit.

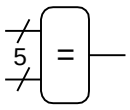
(two-line program)

c) [3] List all of the inputs and outputs of the Hazard Detection Unit.

Signal	Bits	Input/Output

(inputs and outputs)

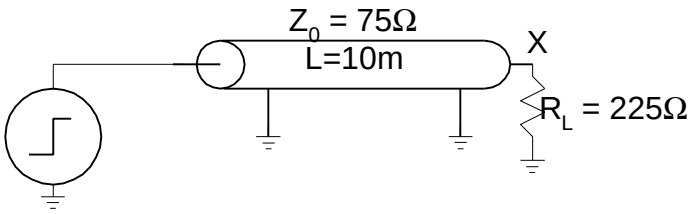
d) [3] Sketch a gate-level schematic of the Hazard Detection Unit. You may assume you have a five-bit equality comparator block available that takes two five-bit inputs and produces a single output that is true if the two inputs are equal.



(schematic)

Transmission Lines

The voltage X at the receiving end of the transmission line will overshoot its value when a step input is applied because the line is terminated with too high of a resistance.



a) [2] How long does it take for a signal to travel from one end to the other of the transmission line? Assume light travels at $2.5*10^8$ m/s in this cable.

Time: _____

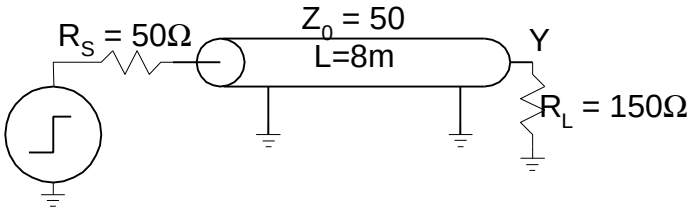
b) [3] What is the peak overshoot at X as a percentage of the input voltage?

Peak Overshoot: _____

c) [3] How long will it take for X to settle to within 10% of the steady-state value?

Settling Time: _____

d) [2] What is the steady-state voltage at the output Y of the transmission line shown below after reflections have died away if the input is a 4-volt step?



Steady-State Voltage: _____

Caches

You are building an instruction cache for a MIPS processor. It has a total size of 2^A bytes. It is 2^B-way set associative ($B \geq 1$), with a block size of 2^C bytes ($C \geq 3$). Addresses are 32 bits long, with bit 0 defined as the least significant bit. Because instruction addresses always fall on word boundaries, bits [1:0] of the address are always 00.

a) [2] In terms of these parameters, which bits of the address are used to select a word within a block?

Block Select Bits: _____

b) [2] In terms of these parameters, which bits of the address are used to select the set within the cache?

Index Bits: _____

c) [2] In terms of these parameters, how many bits is each tag?

Tag Size: _____

d) [2] In terms of these parameters, how many tag bits are in the entire cache?

Total Tag Bits: _____

Performance

You are microarchitecting a MIPS processor and are evaluating the performance of single-cycle, multicycle, and pipelined processors like those described in the text.

Your target application has a mix of 50% R-type instructions, 25% loads, 10% stores, 10% branches, and 5% jumps.

Your circuit designers tell you that the system has the following delays:

- ALU: 900 ps
- Memory: 1000 ps
- Register File: 400 ps

Other delays such as multiplexers and flip-flops may be neglected in this analysis.

a) [2] What is the cycle time of the single-cycle processor?

T_c: _____

b) [2] What is the cycle time of the multi-cycle processor?

T_c: _____

c) [2] What is the cycle time of the pipelined processor?

T_c: _____

Suppose the memory system is perfect, i.e., it has a 100% hit rate. How many millions of instructions per second are executed by:

d) [2] The single-cycle processor?

MIPS: _____

e) [2] The multi-cycle processor?

MIPS: _____

f) [2] The pipelined processor?

MIPS: _____

Now suppose the data cache is still perfect but the instruction cache has a 5% miss rate. On a cache miss, the processor stalls for 60 ns to access main memory, then resumes normal operation. Taking into account cache misses, how many millions of instructions per second are executed by:

g) [2] The single-cycle processor?

MIPS: _____

h) [2] The multi-cycle processor?

MIPS: _____

i) [2] The pipelined processor?

MIPS: _____

Short Questions

[3] Is the miss rate of a two-way set associative cache [always / usually / occasionally / never] better than that of a direct-mapped cache of the same capacity and block size? Explain.

Always / Usually / Occasionally / Never

(circle one)

(Explanation)

[3] Your fancy new web server is crashing once a month due to metastability problems in a synchronizer. Suppose you modify your synchronizer to wait twice as long after sampling an input before using the signal. Would you expect your server to crash [more than once a month / once a month / between once a month and once every three months / less than once every three months]? Explain.

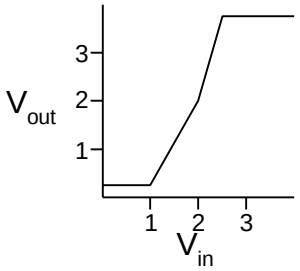
more than once a month / once a month

between once a month and once every three months / less than once every three months

(circle one)

(Explanation)

[3] What is the logic function of the circuit with the transfer function shown below? What are the input and output high and low logic levels?



Logic Function: _____

V_{il}: _____

V_{ih}: _____

V_{ol}: _____

V_{oh}: _____

Lab Question

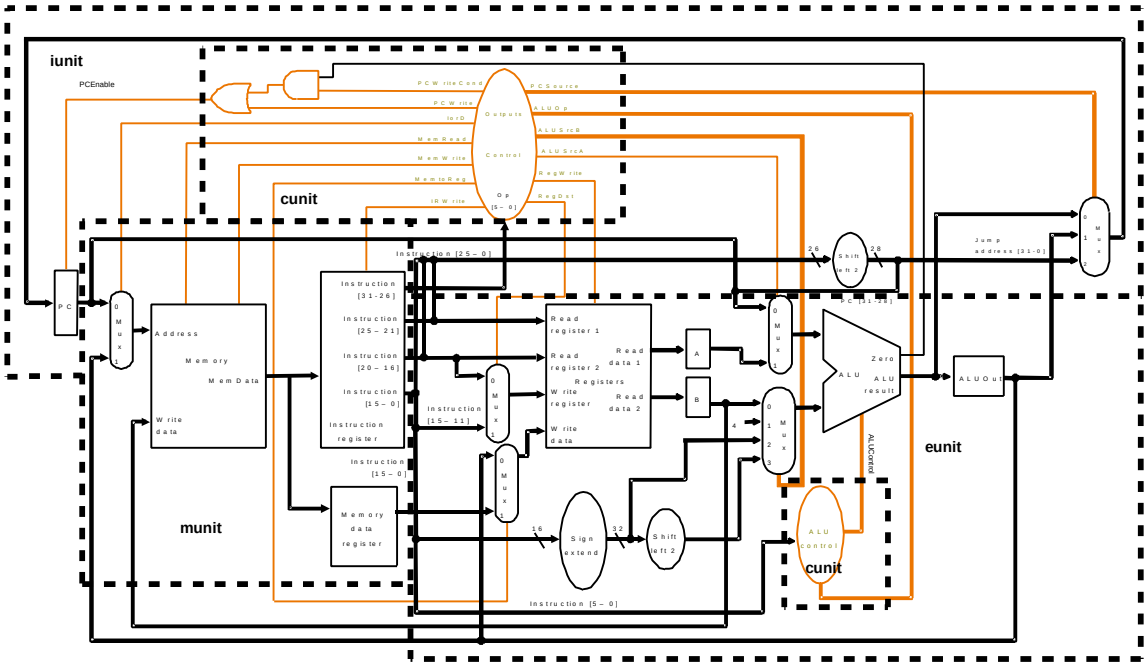
This question does not count toward your time spent on the final exam.

Find the project named final in the E114\Labs\final directory on Engserve1. Open it and copy it to your directory under the name final_xx. This project is an electronic version of my solutions to the multicycle processor from Lab 11. You are encouraged to start this question early so that if the tools act up, you can ask Prof. Harris for help. The instruction memory contains the following program:

```
          addi $t0, $0, 0F     # 2008000F
          jal later            # 0C000003
loop:     j loop               # 08000002
later:    addi $t0, $t0, 0E    # 2108000E
          jr $ra               # 6BE00000
```

Modify the multicycle processor to support the `jal` and `jr` instructions. In the MIPS instruction set, `jr` is an R-type instruction, which would complicate your implementation. To make your work easier, assume instead that `jr` has an opcode of 1A. The jump designation register is held in bits [25:21] of the instruction and the remainder of the bits in the instruction are 0.

a) [3] Mark up the datapath below with your changes. Do not change the number of bits in existing control signals. Use as few new control signals as possible. New control signals should have the value of 0 for instructions other than `jal` and `jr`.



(modified datapath)

b) [3] Add states to the microcode ROM to support your changes. Add any necessary new control signals; these signals should be 0 in states 0-B. Write the ROM contents in hexadecimal.

State	new control signals			ALUOp[1:0]	ALUSrcA	ALUSrcB[1:0]	RegWrite	RegDst	MemtoReg	IorD	MemRead	MemWrite	IRWrite	PCSource[1:0]	PCWrite	PCWriteCond	Seq[1:0]	ROM Contents
0				00	0	01	0	x	x	0	1	0	1	00	1	0	00	02148
1				00	0	11	0	x	x	x	0	0	0	xx	0	0	10	06002
2				00	1	10	0	x	x	x	0	0	0	xx	0	0	11	0C003
3				xx	x	xx	0	x	x	1	1	0	0	xx	0	0	00	00300
4				xx	x	xx	1	0	1	x	0	0	0	xx	0	0	01	01401
5				xx	x	xx	0	x	x	1	0	1	0	xx	0	0	01	00281
6				10	1	00	0	x	x	x	0	0	0	xx	0	0	00	28000
7				xx	x	xx	1	1	0	x	0	0	0	xx	0	0	01	01801
8				01	1	00	0	x	x	x	0	0	0	01	0	1	01	18015
9				xx	x	xx	0	x	x	x	0	0	0	10	1	0	01	00029
A				00	1	10	0	x	x	x	0	0	0	xx	0	0	00	0C000
B				xx	x	xx	1	0	0	x	0	0	0	xx	0	0	01	01001
insert new states here																		

(modified microcode ROM)

c) [8] Modify your final_xx project to implement your changes. Print out copies of all the schematics and .mem files you have modified. Circle your modifications.

(attach schematics and .mem files)

d) [6] Simulate your design with a 10 ns clock period with reset high for the first 10 ns. Print out waveforms of your simulation for 170 ns showing Clk, Reset, PC, Instruction, ALUResult, and CUnit/State values.

(attach waveforms)