Prelim 1
Computer Science 3410, Cornell University
Spring 2018, Prof. Weatherspoon 15 March 2018

Solutions

- Write your NetID at the top of each physical page of the exam.
- **Complete the above steps or face TA wrath.**
- Please turn off and stow away all electronic devices. You may not use them for any reason for the entirety of the exam. Do not bring them with you if you leave the room temporarily.
- This is a closed book and notes examination. You may use the 3-sided reference provided. You may use the fourth side as scratch paper, but it will be recycled, and *not graded*.
- To receive partial credit you must show your work. If you believe a question is open to interpretation, then please ask us about it! Please state any assumptions you make.
- There are 6 problems and 13 pages. Make sure you have the whole exam.
- You have **120 minutes** to complete 125 points. Use your time accordingly.

Problem	Topic	Points	Score
1	Write your name and NetID	1	
2	Potpourri Short Answers	24	
3	Numbers and Arithmetic	24	
4	Logic Equations and Karnaugh Maps	20	
5	Finite State Machines	28	
6	Pipeline Data Hazards	28	
Total		125	

Your Name _____

Your NetID _____

1. Write your name and NetID [1 pt]

2. Potpourri Short Answers [24 pts] (parts a–f)

(a) [4 pts] **Shifting.** If you take a number and shift it, which of the following shift operations might pad the empty spots with 1s? (List all letters for which this might be true.)

A. Logical Shift Left
B. Arithmetic Shift Right
C. Arithmetic Shift Left
D. Logical Shift Right

Correct Answer: **B**

-1 for each additional wrong answer

-4 for no right answers

(b) [4 pts] **SRAM (caches) vs DRAM (memory).** List all the words which correctly complete this sentence:

```
SRAM is _____________ than DRAM.
```

A. denser
B. cheaper (per bit)
C. larger (total storage space)
D. faster

Correct Answer: **D**

-1 for each additional wrong answer

-4 for no right answers

(c) [4 pts] **True or False.** You can look at two assembly instructions and know know whether they will have a data hazard when they are being executed even before you look at the design of the processor they will run on.

Correct Answer: **False**

all or nothing

(d) [4 pts] **Transistors.** Which of the following statements is **false**?

A. P- and N-type transistors are both used in CMOS designs.
B. As transistors get smaller, the frequency of your processor will keep getting faster.
C. As transistors get smaller, you can fit more and more of them on a single chip.
D. Pure silicon is a semi-conductor.

Correct Answer: **B**

-1 for each additional wrong answer

(e) [4 pts] **Memory and Endiannes.** If the word `0x09876543` is stored at memory address `100` on a **big-endian** machine, what value is loaded into `$v0` by the following load byte (LB) instruction?

Note that the format for instruction `LB` is `LB rt, offset(base register)` where `R[rt] = memory[base + signExtendImm(offset)]`: The 16-bit signed offset is added to the contents of base register to form the effective address. The contents of the 8-bit byte at the memory location specified by the effective address (base plus offset) are fetched, sign-extended, and placed in register `R[rt]`.

```
LB $v0, 101($zero)
```

0xFFFFFF87
+4 0xFFFFFF87
+3 0xFF87
+2 0x87
+1: 0x09 or 0x65 or 0x43 or 0x87FFFFFF or 0x87000000 or 87
+0 otherwise

(f) [4 pts] **Calling Convention.** For each of the variables `prev`, `curr`, and `dist`, would it be better to use a caller-save or callee-save register? Why?

```
int convergence(int x)
{
    int prev;
    int curr = x;

    do {
        prev = curr;
        curr = calc_next(prev);
        int dist = calc_dist(prev, curr);
    }  while (dist != 0);

    return curr;
}
```

`prev` and `curr` should be *callee-save*: their values are needed several times before and after function calls, so they should be saved and restored only if the called functions need those registers.

`dist` should be *caller-save*: its value is used only between function calls, so we donâĂŹt want the sub-calls saving and restoring its value.

+2 points for giving the right answer, `prev` and `curr` are *callee-save*, `dist` shall be *caller-save*.
-1 for each wrong answer, up to max -2
+2 points for the explanation. one point for each variable.
-1 for each wrong answer, up to max -2

3. Numbers and Arithmetic [24 pts] (parts a–c)

(a) [16 pts] **Two's complement.** Fill in the missing digits. Negatives should be in 16-bit two's complement. (For octal numbers, sign extend to 18-bits).

`0xFF60` = decimal -160
binary `0000 0000 1000 1000` = decimal 136
decimal `-120` = binary 1111 1111 1000 1000
decimal `-120` = octal 777610
+4 for each right answer
-2 per sign error or sign extension error

(b) [4 pts] **Sign and Magnitude.** Consider the following 5-bit number: `10001`

If we interpret this number as a 5-bit sign magnitude number, what is its value in decimal (base 10) ?

10001: 1st bit means negative, remaining 1 bits = + 1 = -1_{10}
+4 correct answer
+1 for incorrect value, but correct sign

(c) [4 pts] What are the disadvantages of using a sign magnitude to represent numbers? (List all that apply.)

A. Sign magnitude is harder for people to read than two's complement.
B. Sign magnitude has two encodings for the number 0.
C. Sign magnitude addition does not work correctly on a standard adder.
D. Sign magnitude requires a ternary representation which would require three voltage levels instead of the two required by a binary system.
E. Sign magnitude cannot represent negative numbers.

Correct Answer: **B & C**

+2 for each correct answer
-1.5 for each incorrect answer

NetID: _____

4. Logic Equations and Karnaugh Maps [20 pts] (parts a–b) Fibonacci numbers have the following number sequence: $0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, \ldots$

   (a) [10 pts] Fill in the 3-bit truth table to output a 1 for each Fibonacci number **and** write the formula for this truth table in sum-of-products form.

A	B	C	`Fib`
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

   `Fibonacci` $= \overline{A}\overline{B}\overline{C} + \overline{A}\overline{B}C + \overline{A}B\overline{C} + \overline{A}BC + A\overline{B}C$

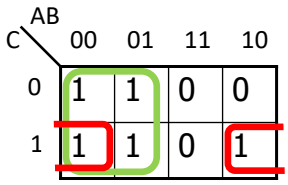
   +5 points for the table

   -1 for each incorrect entry, up to a max of -5

   +5 points for Boolean equation

   -1 for each mistake, up to a max of -5

(b) [10 pts] Fill in the corresponding Karnaugh map and determine the minimal logic equation for this function, **and** draw the corresponding circuit.



`Fibonacci` $= \overline{A} + \overline{B}C$

-3 Did not group bottom right corner, was inconsistent in rest of problem.
-3 Other incorrect K-map grouping/entries; otherwise inconsistent
-2 correct K-map entry, but incorrect equation
-2 Did not include circuit
-0 Consistent with Wrong Answer A

NetID: _____

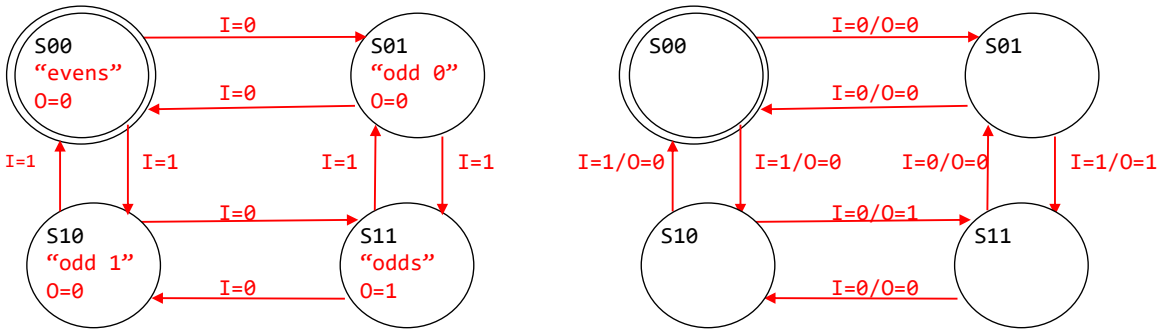
5. Finite State Machines [28 pts] (parts a–d)

(a) [8 pts]

You are an intern at a massive computer firm. Your first task is to design an "odd counter" circuit that receives a single bit input every cycle and outputs a single bit every cycle. It outputs a **1** *if and only if it has seen an odd number of ones* **AND** *an odd number of zeros*. It starts in a state where it has seen an even number of ones and an even number of zeros (remember, zero is an even number). As an example,

the input	I:	1 1 0 1 1 1 0 0 0 1 0 1 1 0 0
will produce the output	O:	0 0 0 1 0 1 0 1 0 0 0 1 0 0 0

Using a **Moore FSM**, complete the FSM diagram below. The names of the states are arbitrary, but use S00 as the start state.



(a) Moore Soln

(b) Mealy Soln

Figure 1: Moore and Mealy Machine Solutions

-4 if drew a Mealy machine instead of a Moore machine, or visa versa
-2 if not an FSM (e.g. no transition from a state)
-2 per incorrect edge
-4 for any output at all
-variable amount for other inconsistencies and errors

(b) [8 pts] Pick a minimal binary encoding for the states of your machine, and write down a truth table representing the output and next state functions.

Fill in the truth table. The previous state is encoded in (S_1,S_0), the next state is encoded in (N_1,N_0), the input is encoded as I, and output is encoded as O. Make sure to indicate the value of the output on your state.

Moore FSM

Current State		Next State				Output
		I=0		I=1		
S_1	S_0	N_1	N_0	N_1	N_0	O
0	0 (even)	0	1 (odd 0's)	1	0 (odd 1's)	0
0	1 (odd 0's)	0	0 (even)	1	1 (odd)	0
1	0 (odd 1's)	1	1 (odd)	0	0 (even)	0
1	1 (odd)	1	0 (odd 1's)	0	1 (odd 0's)	1

Mealy FSM

	S_1	S_1	I	O	N_1	N_0	
(even)	0	0	0	0	0	1	(odd 0's)
(even)	0	0	1	0	1	0	(odd 1's)
(odd 0's)	0	1	0	0	0	0	(even)
(odd 0's)	0	1	1	1	1	1	(odd)
(odd 1's)	1	0	0	1	1	1	(odd 0's)
(odd 1's)	1	0	1	0	0	0	(odd 1's)
(odd)	1	1	0	0	1	0	(even)
(odd)	1	1	1	0	0	1	(odd)

Other possible solutions since table will change based choice for state encoding
-1 per incorrect next state or incorrect output
-4 if Meally Machine truth table for a Moore Machine, or visa versa

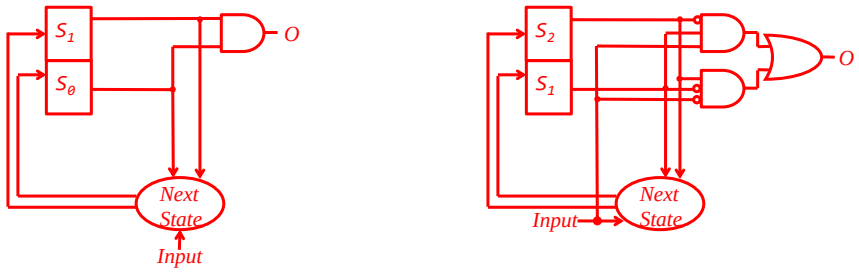
(c) [4 pts] Give Boolean equations for the **output**. You do not need to give equations for the next state, or optimize your answer in any way.

Moore FSM
$O = S_1 S_0$

Mealy FSM
$O = \overline{S_1} S_0 I + S_1 \overline{S_0 I}$

Full credit received if part (c) consistent with table in part (b)
-2 for each incorrect minterm

(d) [8 pts] Draw a circuit diagram for the **entire** FSM (i.e. input, logic circuit for the output function, next state function, and state): Use registers for the state, simple gates for the output function (AND, OR, NOT, etc.), and just represent the next state function with a circle (you will need to show the inputs into and outputs out of the circle representing the next state logic).



(a) Moore Soln (b) Mealy Soln

Figure 2: Moore and Mealy Machine Solutions

-1 small wiring issue
-3 significant wiring issue
-6 very large wiring issue

-2 per incorrect component

-variable for other inconsistencies or errors

6. Pipeline Data Hazards [28 pts] (parts a–e)

Prepare the same code for 3 different MIPS processors by injecting just enough *nop* instructions between the original lines of code to preserve correctness.

(a) [5 pts] **Processor 1:** has a single-cycle data path.

```
sll r6, r6, 4
_____ nops
sw r6 0(r3)
_____ nops
addiu r6, r6, 10
```

0 nops in line 2, 0 nops in line 4

+2.5 for each correct answer

(b) [5 pts] **Processor 2:** is a 5-stage, pipelined processor with **no forwarding logic**. The register file is written to during the first half of the cycle and read from during the second half.

```
sll r6, r6, 4
_____ nops
sw r6 0(r3)
_____ nops
addiu r6, r6, 10
```

2 nops in line 2, 0 nops in line 4

+2.5 for each correct answer

(c) [5 pts] **Processor 3:** is a 5-stage, pipelined processor with **full forwarding logic**. The register file is written to during the first half of the cycle and read from during the second half.

```
sll r6, r6, 4
_____ nops
sw r6 0(r3)
_____ nops
addiu r6, r6, 10
```

0 nops in line 2, 0 nop in line 4

+2.5 for each correct answer

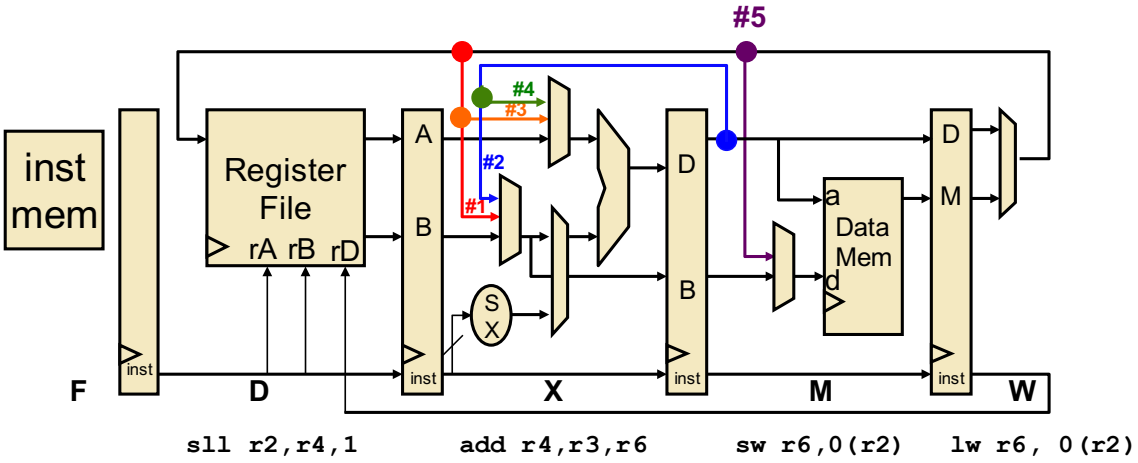
(d) [5 pts] **Processor 4:** is the same as Processor 3 above. Identify all the data hazards in the original code (reprinted below). State how the hazards are resolved (i.e. state the stages involved in forwarding and/or any stalls that may occur).

```
1   sll r6, r6, 4
2   _____ nops
3   sw r6 0(r3)          Ex/M → Ex resolves hazard for r6 between lines 1 and 3
4   _____ nops
5   addiu r6, r6, 10     M/WB → Ex resolves hazard for r6 between lines 1 and 5
```

+1.25 for identifying the hazards between the correct lines
+1.25 for identifying how the hazard is resolved
-1 Unnecessary stall
-3 Identified a non-existing structural hazard
-2 Identified incorrect load-use hazard. There is no data dependency nor load-use hazard between the `SW` and `ADDIU` instructions!
-2 Incorrectly forwarded from M/WB to Mem

(e) [8 pts] **Forwarding.**

In the pipeline below, the register file is written in the 1st half of the cycle and read in the 2nd half of the cycle. Bypasses #1-#5 are shaded. In the D/X latch, the 1st operand to an ALU instruction is stored in the A slot of the latch. The 2nd operand is stored in the B slot of the latch.



For each input specified below, state where the instruction gets its data. 1-5 refer to the bypasses shown in the above figure. RF stands for register file.

r6 of the sw instruction	**#5**
r3 of the add instruction	**RF**
r6 of the add instruction	**#1**
r4 of the sll instruction	**#4**

+2 per answer

[This page intentionally left empty]