# CS/ECE 252: INTRODUCTION TO COMPUTER ENGINEERING

# UNIVERSITY OF WISCONSIN – MADISON

Instructor: Andy Phelps
TAs: Newsha Ardalani, Peter Ohmann, and Jai Menon

Midterm Examination 3
In Class (50 minutes)
Friday, April 8
Weight: 15%

**NO BOOK(S), NOTE(S), CALCULATOR(S) OF ANY SORT**

*This exam has 13 front-and-back pages. Plan your time carefully, since some problems are longer than others. You must turn in all pages.*

LAST NAME: ______________________________

FIRST NAME: ______________________________

SECTION: ______________________________

ID #: ______________________________

| **Question** | **Maximum Points** | **Points** |
|---|---|---|
| 1 | 4 | |
| 2 | 8 | |
| 3 | 6 | |
| 4 | 8 | |
| 5 | 4 | |
| 6 | 10 | |
| Total | 40 | |

## Problem 1 (4 points)

Suppose below to be the current snapshot of memory. Further,suppose, at the start, the PC is x4000.

```
x4000   |   0101 0010 0110 0000
x4001   |   0001 0110 0111 1111
x4002   |   0000 0110 0000 0011
```

a. Decode each instruction above.

x4000 _____________

x4001 _____________

x4002 _____________

After the execution of the instruction at x4002, what are the values of the following registers :

R1 _____________

R2 _____________

PC _____________

## Problem 2 (8 points)

Suppose below to be the current snapshot of memory. Further, suppose, at the start, that the PC is x5000.

```
x5000   |   1110 0010 0000 0011
x5001   |   1010 0100 0000 0010
x5002   |   0110 0110 0111 1111
x5003   |   0010 1001 1111 1101
x5004   |   0001 0011 1000 1011
```

a. Decode each instruction above.

X5000 _______________

x5001 _______________

x5002 _______________

x5003 _______________

x5004 _______________

b. After the program halts, what are the values (in hex) of:

R1 _____________

R2 _____________

R3 _____________

R4 _____________

Note: You ***MUST*** do part (a) to be given ***any*** credit for part (b).

## Problem 3 (6 points)

The following (incomplete) binary code snippet accepts an input value in register R1, increments it by 1 if the value is even and then halts. Odd values are left untouched. This can be represented in pseudocode as :

Note : A represents value in R1

```
if A is divisible by 2 then :
   A <-- A + 1
end if
halt
```

Complete the code to achieve functionality described above by filling in the blanks (two of the required instructions are already filled in for you). Also write down the corresponding decoded instructions.

Assume that the PC register contains 3001 initially.

```
x3001    __________________

x3002    0000 0010 0000 0001

x3003    __________________

x3004    1111 0000 0010 0101
```

Note that TRAP x25 is used to halt execution.

## Problem 4 (8 points)

The LC-3 ISA doesn't provide a subtract instruction though the required functionality can be implemented using instructions it does support. The code fragment listed below (Fig. 1) attempts to subtract two values stored at the memory addresses x3008 and x3009, leaving the result in register R3. Fig. 2 illustrates the relevant memory state at the time of execution.

Figure 1

| Address | Machine Code | Decoded Instruction |
|---|---|---|
| x3000 | 0010 0010 0000 1000 | |
| x3001 | 0010 0100 0000 1000 | |
| x3002 | 1001 0100 1011 1111 | |
| x3003 | 0001 0110 0100 0010 | |

Figure 2

| Address | Value |
|---|---|
| x3008 | xDEAD |
| x3009 | xBEEF |

Unfortunately, the code above is buggy and doesn't work as expected. Specifically, you will need to find and fix any errors in the instruction's machine code as well as any logical mistakes in the code.

*Note : It's easier to work with decoded instructions so an additional column is provided for you to write the decoded instructions in.*

Your solution should note any changes or additional instructions introduced as :

*<memory address> <assembly instruction> <machine code>*

**Hint** : *There are 3 errors in the given listing. Treat errors in different instructions as separate errors.*

## Problem 5 (4 points)

Part a.

Consider an LD instruction at x3020. What is the largest possible memory address this instruction can load from/reference? Conversely, which is the smallest possible address? Write the instruction and corresponding machine code which performs a load from these addresses.

Part b.

Now consider the same LD instruction but this time, assume you're writing code for a machine where the PC offset is ***not*** sign-extended. In other words, the offset field is **zero-extended** to 16 bits. In this case, what is the largest possible memory address this instruction can load from/reference? Conversely, which is the smallest possible address? Write the instruction and the corresponding machine code which performs a load from these addresses.

## Problem 6 (10 points)

The *execution trace* of a program usually records the state of various registers and execution contexts when the program is run for a given set of inputs. Such traces can be very useful tools when debugging code (and moreso when a debugger isn't available).

Figure 3

| Address | Machine Code | Decoded Instruction |
|---|---|---|
| x3000 | 0101 0010 0110 0000 | |
| x3001 | 0101 1001 0010 0000 | |
| x3002 | 0001 1001 0010 1010 | |
| x3003 | 0010 0100 1111 1100 | |
| x3004 | 0110 0110 1000 0000 | |
| x3005 | 0001 0100 1010 0001 | |
| x3006 | 0001 0010 0100 0011 | |
| x3007 | 0001 1001 0011 1111 | |
| x3008 | 0000 0011 1111 1011 | |
| x3009 | 1111 0000 0010 0101 | |

*Note : It's easier to work with decoded instructions so an additional column is provided for you to write the decoded instructions in.*

Figure 4

| Address | Value |
|---|---|
| x3100 | 0x3105 |
| x3101 | 0x0001 |
| x3102 | 0x0001 |
| x3103 | 0x0001 |
| x3104 | 0x0001 |
| x3105 | 0x0001 |
| x3106 | 0x0001 |
| x3107 | 0x0001 |
| x3108 | 0x0001 |
| x3109 | 0x0001 |
| x310A to x310E | 0x0000 |

Part a.

For the program listing shown in Fig. 3, record the state of registers R1, R2, R3 and R4 after the 2nd iteration of the loop (immediately after the branch has been taken for the second time). Use the table printed below for this purpose.

| Register | Value after 1st iteration | Value after 2nd iteration |
|---|---|---|
| R1 | | |
| R2 | | |
| R3 | | |
| R4 | | |

Part b.

The expected behavior of the program listed in Fig. 3 is that it should compute the sum of values stored in the address range x3100 to x3109, store the computed sum in R1 and halt. Taking this into account, would you say that the program listing in Fig. 3 works correctly?

- If it does, then what is the final value in R1?

- If not, how would you fix it (assuming that such a fix doesn't require changing more than one instruction, and doesn't add any more instructions)?

# Extra Scratch Paper

# Extra Scratch Paper

# Extra Scratch Paper