CS/ECE 252: INTRODUCTION TO COMPUTER ENGINEERING

UNIVERSITY OF WISCONSIN—MADISON

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

Midterm 3
In Class (50 minutes)
Note: This was originally given as an optional supplemental test after Midterm 3; for future reference it is more appropriately regarded simply as Midterm 3.

NO: BOOK(S), NOTE(S), CALCULATORS OF ANY SORT.

This exam has 10 pages, including a blank page. Plan your time carefully, since some problems are longer than others. You must turn in all pages.

For your convenience, a listing of the **LC-3 instruction set** is given at the end of this booklet.

LAST NAME: _____

FIRST NAME:_____

SECTION: _____

ID # _____

Question	Maximum Points	Points
1	6	
2	6	
3	5	
4	7	
5	7	
6	6	
Total	37	

Problem 1. (6 points)

This program adds up ten numbers, leaving the sum in R1. The numbers are found in memory starting at location x3100. There is **one** error in this program. **Circle** the bad instruction, and write the corrected instruction (in binary) below.

```
x3000 0101 001 001 1 00000  Clear R1
x3001 1110 010 011111110    R2 ← x3100
x3002 0101 011 011 1 00000  Clear R3
x3003 0001 011 011 1 01010  R3 ← 10
x3004 0000 100 000000100    Branch if done
x3005 0110 100 010 000000   R4 ← M[R2]
x3006 0001 001 001 000 100  R1 ← R1 + R4
x3007 0001 011 011 1 11111  Decrement loop count
x3008 0000 111 111111011    Branch Loop
x3009 1111 0000 0010 0101   HALT
```

Corrected Instruction: _____

As we said in class, there are really *two* errors in this program. Extra credit if you fix both of them; a maximum of 3 extra points if you accurately and completely fix both.

Error #1: The loop iterates 11 times, not 10. This can be fixed in one of three ways:

a. Change the first branch to BRz or BRnz, instead of BRn.
   x3004 0000 110 000000100
b. Put a 9 in R3 instead of a 10:
   x3003 0001 011 011 1 01001
c. (Not quite as neat) Change the second branch to only branch back on positive
   x3008 0000 001 111111011

Error #2: R2, the data pointer, never gets incremented. Insert the following instruction after either x3005 or x3006 (though not after x3007):

0001 010 010 1 00001

Problem 2. (6 Points)

Some of the following instructions are "no-operation"; that is, they do not change the value of any of the eight general-purpose registers or the PC. (For this problem, don't worry about the condition codes.)

Check the box next to each no-operation instruction.

☐ `0001 000 000 100000 Add` Yes, adds zero

☐ `0001 000 000 000000 Add` No, doubles R0

☐ `1110 000 000000000 LEA` No, copies PC into R0

☐ `0101 000 000 100000 And` No, clears R0

☐ `0101 000 000 000000 And` Yes, ANDs R0 with itself which doesn't change it

☐ `0000 000 000000000 Branch` Yes, never branches

☐ `0000 111 000000000 Branch` Yes, branches but doesn't add anything to PC

☐ `0110 000 000 000000 LDR` No, does a load into R0

Problem 3. (5 points)

a. Suppose you wish to test whether or not bit 2 is set in register R0. (That is, you wish to set the condition codes so that you can do a conditional branch.)
Write the **instruction or instructions** below, in binary, for the test. (You do not need to include the branch itself). Assume the data is already in R0.

b. Suppose you need to do the same thing, except test for bit 8. You do **not** need to write the binary, but state how many instructions are needed and describe what they are.
Bit 8 is beyond what you can set in the immediate field. You need to load a register with a word that just has bit 8 set (x0100), then do the AND. There are several ways to put x0100 into a register. The easiest is to load it:

```
    LD R1,BitEight
    AND R1,R1,R0
    . . .
    etc
BitEight .FILL x0100
```

If you want to be creative, you could create x0100 by putting x0008 into a register and then shifting it left (adding to itself) five times; this could be done in a loop or just with five ADDs. (There is no point into doing this in real life.)

Another approach is to shift R0 left seven times (add to itself 7 times). Then you don't even need the AND; just branch on the 'n' bit.

Problem 4. (7 points)

Most computers have an XOR instruction, but the LC-3, alas, does not. But recall that XOR (Exclusive OR) means "one or the other but not both"; we can write a program to do that operation as ((A or B) and not (A and B)). It is started below; finish it. Include **both** binary and comments. Assume the operands are in R0 and R1, and leave the result in R2.

```
1001 100 000 111111      R4 ← NOT R0

1001 101 001 111111      R5 ← NOT R1

0101 110 100 000 101     R6 ← R4 AND R5
```

```
1001 110 110 111111      R6 <- NOT R6   (which is R0 OR R1)
_____________________  ___________________________________

0101 100 000 000 001     R4 <- R0 AND R1
_____________________  ___________________________________

1001 100 100 111111      R4 <- NOT R4   = (NOT (R0 AND R1))
_____________________  ___________________________________

0101 010 100 000 110     R2 <- R4 AND R6   = (R0 OR R1) AND NOT (R0 AND R1)
_____________________  ___________________________________
```

Problem 5. (7 points)

What ends up in the registers after these instructions are executed?

```
x3000 0010 000 000001111    LD   R0 <- M[x3010]

x3001 1110 001 000001111    LEA  R1 <- x3011

x3002 1010 010 000001111    LDI  R2 <- M[M[x3012]]

x3003 0110 011 011 000010   LDR  R3 <- M[R1+2] = M[x3013]

x3004 0110 100 010 000010   LDR  R4 <- M[R0+2] = M[x3016]

x3005 1111 0000 0010 0101   HALT

. . .
. . . These are data, not instructions:

x3010 0011 0000 0001 0100   x3014

x3011 0011 0000 0001 1111   x301f

x3012 0011 0000 0001 0101   x3015

x3013 0000 0000 0010 0100   x0026

x3014 0011 0000 0001 0110   x3016

x3015 0000 0000 1001 0001   x0091

x3016 0011 0000 0010 0000   x3020
```

R0 _____ x3014

R1 _____ x3011

R2 _____ x0091

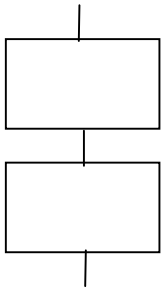
R3 _____ x0026

R4 _____ x3020

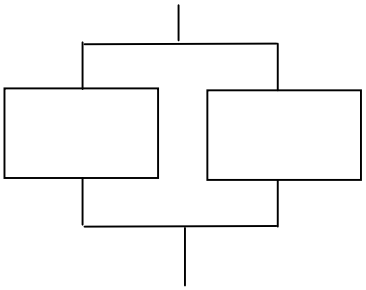
A number of people added wrong for the first three instructions. They converted 1111 to a decimal 15, then proceeded to add HEX 15 to the PC. This resulted in the wrong answer for all five registers. If it was clear to me that you understood the instructions and did everything right except this one thing, I gave back some points. If you got a zero and feel you fall into that category, come see the instructor or a TA.

Problem 6. (6 points)
Which of these constructs are the ones we studied for *systematic decomposition*? List the letters corresponding to each one that applies: A, D, F (Don't forget that all such decompositions have exactly one path in at the top and one out at the bottom, so they can replace a rectangular box.

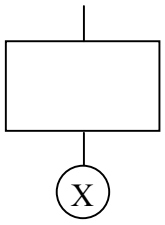
a. Sequential



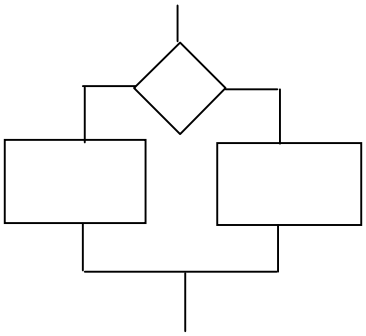
b. Parallel



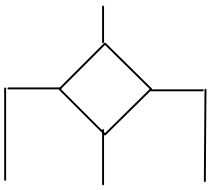
c. Halting



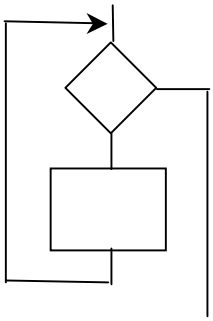
d. Conditional



e. Multi-way Conditional



f. Iterative



Scratch paper