

**CS/ECE 252: INTRODUCTION TO COMPUTER ENGINEERING  
UNIVERSITY OF WISCONSIN – MADISON**

Prof. Mark D. Hill and Prof. Parmesh Ramanathan

TAs Kasturi Bidarkar, Ryan Johnson, Jie Liu, & Ramachandran Syamkumar

Midterm Examination 3

In Class (50 minutes)

Friday, November 13, 2009

Weight: 15%

**CLOSED BOOK, NOTE, CALCULATOR, PHONE, & COMPUTER.**

The exam has **three** two-sided pages and a copy of the *LC-3 Instruction Set Handout* on the final page (Please feel free to detach this final page, but insert it into your exam when you turn it in).

Plan your time carefully, since some problems are longer than others.

NAME: \_\_\_\_\_

SECTION: \_\_\_\_\_

ID#: \_\_\_\_\_

| <b>Problem Number</b> | <b>Maximum Points</b> | <b>Actual Points</b> |
|-----------------------|-----------------------|----------------------|
| 1                     | 6                     |                      |
| 2                     | 4                     |                      |
| 3                     | 4                     |                      |
| 4                     | 4                     |                      |
| 5                     | 8                     |                      |
| 6                     | 4                     |                      |

**Problem 1 (6 points)**

a) Register R1 contains xA010 and register R2 contains x5FF0. What will be the values of the condition codes N, Z, and P after executing the instruction AND R3, R1, R2? Also State contents of R3.

Contents of R3: 16'h0010

N= 0

Z= 0

P= 1

b) How many times does the LC-3 make a Read Request and Write Request to the memory during the processing of a STI instruction ?. (Do not include Instruction Fetch)

Number of Write Requests = 1

Number of Read requests = 1

**Problem 2 (4 points)**

a) The following table shows a part of LC-3's memory. If the instruction after the conditional branch is fetched from location x3100, what are the initial contents of register R1. Write your answer in hexadecimal format.

| Address | Data                                  |
|---------|---------------------------------------|
| x3100   | 1110 010 000 000111 ; R2 <- x3108     |
| x3101   | 1001 001 001 111111 ; R1 <- NOT(R1)   |
| x3102   | 0001 001 001 1 00001 ; R1 <- R1 + 1   |
| x3103   | 0001 011 001 0 00 010 ; R3 <- R1 + R2 |
| x3104   | 0000 010 111111011 ; BRz x3100        |

Contents of R1 = x3108 (in hexadecimal format, please)

b) In any program, what is the range of values that the immediate field of an AND instruction can take?

$-2^4$  to  $(2^4 - 1)$

**Problem 3 (4 points)**

a) Identify an LC-3 instruction which **does not change the contents of the registers, memory, and the condition codes**. Fill in the machine code for the identified in the space below.

|    |  |  |  |  |  |  |  |  |  |  |  |  |  |   |  |
|----|--|--|--|--|--|--|--|--|--|--|--|--|--|---|--|
|    |  |  |  |  |  |  |  |  |  |  |  |  |  |   |  |
| 15 |  |  |  |  |  |  |  |  |  |  |  |  |  | 0 |  |

**Branch condition**

b) Identify an LC-3 instruction which **does not change the contents of the registers, memory**. Fill in the machine code for the identified in the space below.

|    |  |  |  |  |  |  |  |  |  |  |  |  |  |   |  |
|----|--|--|--|--|--|--|--|--|--|--|--|--|--|---|--|
|    |  |  |  |  |  |  |  |  |  |  |  |  |  |   |  |
| 15 |  |  |  |  |  |  |  |  |  |  |  |  |  | 0 |  |

Multiple Answers possible: ADDI /ANDI /BR/JMP

**Problem 4 (4 points)**

The program below checks if bit 2 (the 3rd bit) of R5 is a one. If bit 2 is one, the program adds the contents of R3 and R4 and stores the result in R1. Otherwise, it does nothing. Insert the missing LC-3 machine language instructions. **No credit will be given if you do not comment your instruction.**

| Address | ISA Instruction                               |
|---------|---|
| x3000   | 0101 0100 1010 0000 ;Clear R2                 |
| x3001   | 0001 0100 1010 0100 ;R2 <- R2+4               |
| x3002   | <b>0101 xxx0 1000 0101 DR &lt;- R2 AND R5</b> |
| x3003   | 0000 0100 0000 0001 ;BRz x3005                |
| x3004   | <b>0001 0010 1100 0100 R1 &lt;- R3 +R4</b>    |
| x3005   | 1111 0000 0010 0101 ;HALT                     |

**Problem 5 (8 points)**

We are about to execute the following program:

| Address | ISA Instruction                         |
|---------|---|
| x3000   | 0010 0000 0000 0110 ; LD R0, x0006      |
| x3001   | 0110 0010 1101 1101 ; LDR R1, R3, x001D |
| x3002   | 1010 0100 1111 0000 ; LDI R2, x00F0     |
| x3003   | 1110 0110 0000 1010 ; LEA R3, x000A     |
| x3004   | 1111 0000 0010 0101 ; HALT              |

The state of the machine before the program starts is given below:

| Address | Memory Contents | Register | Initial contents |
|---------|-----------------|----------|------------------|
| x158E   | x0012           | R0       | x200E            |
| x200E   | x3258           | R1       | x200E            |
| x2257   | x0000           | R2       | x3001            |
| x2FFF   | x4567           | R3       | x3001            |
| x3002   | xA4F0           |          |                  |
| x3006   | xABEB           |          |                  |
| x3007   | xAAAA           |          |                  |
| x300E   | x92FE           |          |                  |
| x301D   | x2121           |          |                  |
| x301E   | x2020           |          |                  |
| x30F3   | x3111           |          |                  |
| x3111   | x1332           |          |                  |
| xABCD   | x1580           |          |                  |
| xABDB   | x0001           |          |                  |

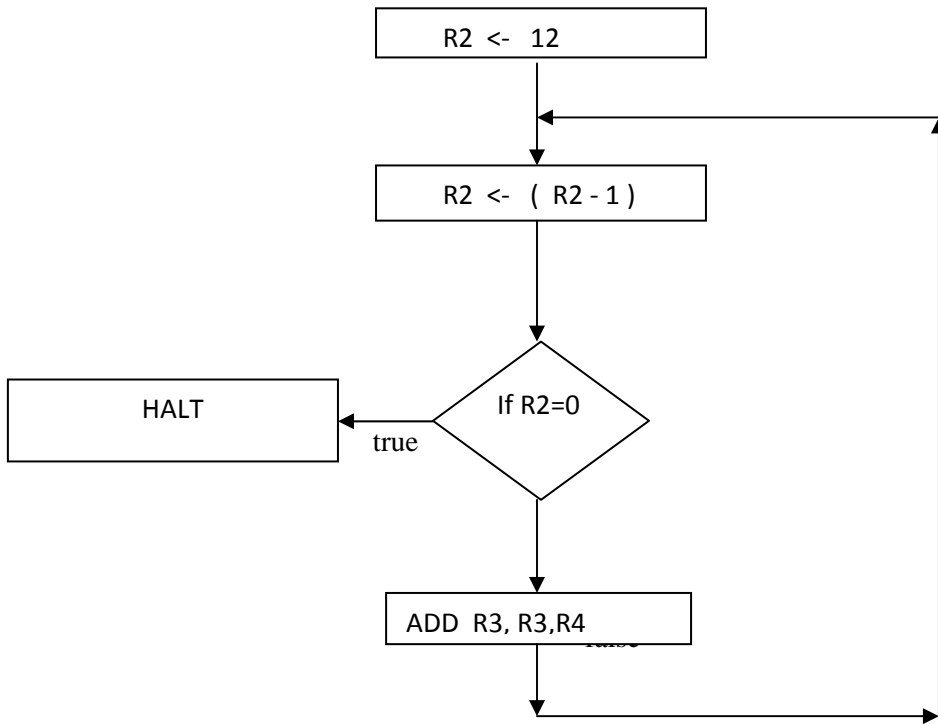
The contents of Registers R0, R1, R2, and R3 after the program completes execution are as follows. **Please write your answer in hexadecimal format.**

Contents of R0 = xAAAA      Contents of R1 = x2020

Contents of R2 = x1332      Contents of R3 = x300E

**Problem 6(4 points)**

A flowchart is drawn below and the memory contents are shown in a table on the next page. Please enter the values of the condition codes and offsets in accordance with the flowchart.



|       |                        |                         |                              |
|-------|------------------------|-------------------------|------------------------------|
| x3100 | <b>R2 &lt;- 12</b>     |                         |                              |
| x3101 | <b>R2 &lt;- R2 - 1</b> |                         |                              |
|       | <b>0000</b>            | <b>a)</b><br><b>010</b> | <b>b)</b><br><b>00000010</b> |
| x3102 | <b>R3 &lt;- R3 +R4</b> |                         |                              |
| x3103 |                        |                         |                              |
| x3104 | <b>0000</b>            | <b>c)</b><br><b>111</b> | <b>d)</b><br><b>11111100</b> |
| x3105 | <b>HALT</b>            |                         |                              |

# LC-3 Instruction Set (Entered by Mark D. Hill on 03/14/2007; last update 03/15/2007)

PC': incremented PC. setcc(): set condition codes N, Z, and P. mem[A]:memory contents at address A.  
 SEXT(immediate): sign-extend immediate to 16 bits. ZEXT(immediate): zero-extend immediate to 16 bits.  
 Page 2 has an ASCII character table.

| 15    | 14 | 13 | 12 | 11 | 10 | 9  | 8 | 7          | 6 | 5       | 4         | 3         | 2 | 1 | 0   |  |   |   |   |  |  |
|-------|----|----|----|----|----|----|---|------------|---|---------|-----------|-----------|---|---|-----|--|---|---|---|--|--|
| ----- |    |    |    |    |    |    |   |            |   |         |           |           |   |   |     | ADD DR, SR1, SR2 ; Addition                          |   |   |   |  |  |
|       | 0  | 0  | 0  | 1  |    | DR |   | SR1        |   | 0       |           | 0         | 0 |   | SR2 |  |   |   |   |  |  |
| ----- |    |    |    |    |    |    |   |            |   |         |           |           |   |   |     | DR ← SR1 + SR2 also setcc()                          |   |   |   |  |  |
| ----- |    |    |    |    |    |    |   |            |   |         |           |           |   |   |     | ADD DR, SR1, imm5 ; Addition with Immediate          |   |   |   |  |  |
|       | 0  | 0  | 0  | 1  |    | DR |   | SR1        |   | 1       |           | imm5      |   |   |     |  |   |   |   |  |  |
| ----- |    |    |    |    |    |    |   |            |   |         |           |           |   |   |     | DR ← SR1 + SEXT(imm5) also setcc()                   |   |   |   |  |  |
| ----- |    |    |    |    |    |    |   |            |   |         |           |           |   |   |     | AND DR, SR1, SR2 ; Bit-wise AND                      |   |   |   |  |  |
|       | 0  | 1  | 0  | 1  |    | DR |   | SR1        |   | 0       |           | 0         | 0 |   | SR2 |  |   |   |   |  |  |
| ----- |    |    |    |    |    |    |   |            |   |         |           |           |   |   |     | DR ← SR1 AND SR2 also setcc()                        |   |   |   |  |  |
| ----- |    |    |    |    |    |    |   |            |   |         |           |           |   |   |     | AND DR, SR1, imm5 ; Bit-wise AND with Immediate      |   |   |   |  |  |
|       | 0  | 1  | 0  | 1  |    | DR |   | SR1        |   | 1       |           | imm5      |   |   |     |  |   |   |   |  |  |
| ----- |    |    |    |    |    |    |   |            |   |         |           |           |   |   |     | DR ← SR1 AND SEXT(imm5) also setcc()                 |   |   |   |  |  |
| ----- |    |    |    |    |    |    |   |            |   |         |           |           |   |   |     | BRx, label (where x = {n,z,p,zp,np,nz,nzp}) ; Branch |   |   |   |  |  |
|       | 0  | 0  | 0  | 0  |    | n  |   | z          |   | p       |           | PCoffset9 |   |   |     |  |   |   |   |  |  |
| ----- |    |    |    |    |    |    |   |            |   |         |           |           |   |   |     | GO ← ((n and N) OR (z AND Z) OR (p AND P))           |   |   |   |  |  |
| ----- |    |    |    |    |    |    |   |            |   |         |           |           |   |   |     | if (GO is true) then PC ← PC' + SEXT(PCoffset9)      |   |   |   |  |  |
| ----- |    |    |    |    |    |    |   |            |   |         |           |           |   |   |     | JMP BaseR ; Jump                                     |   |   |   |  |  |
|       | 1  | 1  | 0  | 0  |    | 0  | 0 | 0          |   | BaseR   |           | 0         | 0 | 0 | 0   | 0  |   |   |   |  |  |
| ----- |    |    |    |    |    |    |   |            |   |         |           |           |   |   |     | PC ← BaseR   |   |   |   |  |  |
| ----- |    |    |    |    |    |    |   |            |   |         |           |           |   |   |     | JSR label ; Jump to Subroutine                       |   |   |   |  |  |
|       | 0  | 1  | 0  | 0  |    | 1  |   | PCoffset11 |   |         |           |           |   |   |     |  |   |   |   |  |  |
| ----- |    |    |    |    |    |    |   |            |   |         |           |           |   |   |     | R7 ← PC', PC ← PC' + SEXT(PCoffset11)                |   |   |   |  |  |
| ----- |    |    |    |    |    |    |   |            |   |         |           |           |   |   |     | JSRR BaseR ; Jump to Subroutine in Register          |   |   |   |  |  |
|       | 0  | 1  | 0  | 0  |    | 0  |   | 0          | 0 |         | BaseR     |           | 0 | 0 | 0   | 0  | 0 | 0 |   |  |  |
| ----- |    |    |    |    |    |    |   |            |   |         |           |           |   |   |     | temp ← PC', PC ← BaseR, R7 ← temp                    |   |   |   |  |  |
| ----- |    |    |    |    |    |    |   |            |   |         |           |           |   |   |     | LD DR, label ; Load PC-Relative                      |   |   |   |  |  |
|       | 0  | 0  | 1  | 0  |    | DR |   | PCoffset9  |   |         |           |           |   |   |     |  |   |   |   |  |  |
| ----- |    |    |    |    |    |    |   |            |   |         |           |           |   |   |     | DR ← mem[PC' + SEXT(PCoffset9)] also setcc()         |   |   |   |  |  |
| ----- |    |    |    |    |    |    |   |            |   |         |           |           |   |   |     | LDI DR, label ; Load Indirect                        |   |   |   |  |  |
|       | 1  | 0  | 1  | 0  |    | DR |   | PCoffset9  |   |         |           |           |   |   |     |  |   |   |   |  |  |
| ----- |    |    |    |    |    |    |   |            |   |         |           |           |   |   |     | DR ← mem[mem[PC' + SEXT(PCoffset9)]] also setcc()    |   |   |   |  |  |
| ----- |    |    |    |    |    |    |   |            |   |         |           |           |   |   |     | LDR DR, BaseR, offset6 ; Load Base+Offset            |   |   |   |  |  |
|       | 0  | 1  | 1  | 0  |    | DR |   | BaseR      |   | offset6 |           |           |   |   |     |  |   |   |   |  |  |
| ----- |    |    |    |    |    |    |   |            |   |         |           |           |   |   |     | DR ← mem[BaseR + SEXT(offset6)] also setcc()         |   |   |   |  |  |
| ----- |    |    |    |    |    |    |   |            |   |         |           |           |   |   |     | LEA, DR, label ; Load Effective Address              |   |   |   |  |  |
|       | 1  | 1  | 1  | 0  |    | DR |   | PCoffset9  |   |         |           |           |   |   |     |  |   |   |   |  |  |
| ----- |    |    |    |    |    |    |   |            |   |         |           |           |   |   |     | DR ← PC' + SEXT(PCoffset9) also setcc()              |   |   |   |  |  |
| ----- |    |    |    |    |    |    |   |            |   |         |           |           |   |   |     | NOT DR, SR ; Bit-wise Complement                     |   |   |   |  |  |
|       | 1  | 0  | 0  | 1  |    | DR |   | SR         |   | 1       |           | 1         | 1 | 1 | 1   | 1  |   |   |   |  |  |
| ----- |    |    |    |    |    |    |   |            |   |         |           |           |   |   |     | DR ← NOT(SR) also setcc()                            |   |   |   |  |  |
| ----- |    |    |    |    |    |    |   |            |   |         |           |           |   |   |     | RET ; Return from Subroutine                         |   |   |   |  |  |
|       | 1  | 1  | 0  | 0  |    | 0  | 0 | 0          |   | 1       | 1         | 1         |   | 0 | 0   | 0  | 0 | 0 | 0 |  |  |
| ----- |    |    |    |    |    |    |   |            |   |         |           |           |   |   |     | PC ← R7  |   |   |   |  |  |
| ----- |    |    |    |    |    |    |   |            |   |         |           |           |   |   |     | RTI ; Return from Interrupt                          |   |   |   |  |  |
|       | 1  | 0  | 0  | 0  |    | 0  | 0 | 0          | 0 | 0       | 0         | 0         | 0 | 0 | 0   | 0  | 0 | 0 | 0 |  |  |
| ----- |    |    |    |    |    |    |   |            |   |         |           |           |   |   |     | See textbook (2 <sup>nd</sup> Ed. page 537).         |   |   |   |  |  |
| ----- |    |    |    |    |    |    |   |            |   |         |           |           |   |   |     | ST SR, label ; Store PC-Relative                     |   |   |   |  |  |
|       | 0  | 0  | 1  | 1  |    | SR |   | PCoffset9  |   |         |           |           |   |   |     |  |   |   |   |  |  |
| ----- |    |    |    |    |    |    |   |            |   |         |           |           |   |   |     | mem[PC' + SEXT(PCoffset9)] ← SR                      |   |   |   |  |  |
| ----- |    |    |    |    |    |    |   |            |   |         |           |           |   |   |     | STI, SR, label ; Store Indirect                      |   |   |   |  |  |
|       | 1  | 0  | 1  | 1  |    | SR |   | PCoffset9  |   |         |           |           |   |   |     |  |   |   |   |  |  |
| ----- |    |    |    |    |    |    |   |            |   |         |           |           |   |   |     | mem[mem[PC' + SEXT(PCoffset9)]] ← SR                 |   |   |   |  |  |
| ----- |    |    |    |    |    |    |   |            |   |         |           |           |   |   |     | STR SR, BaseR, offset6 ; Store Base+Offset           |   |   |   |  |  |
|       | 0  | 1  | 1  | 1  |    | SR |   | BaseR      |   | offset6 |           |           |   |   |     |  |   |   |   |  |  |
| ----- |    |    |    |    |    |    |   |            |   |         |           |           |   |   |     | mem[BaseR + SEXT(offset6)] ← SR                      |   |   |   |  |  |
| ----- |    |    |    |    |    |    |   |            |   |         |           |           |   |   |     | TRAP ; System Call                                   |   |   |   |  |  |
|       | 1  | 1  | 1  | 1  |    | 0  | 0 | 0          | 0 |         | trapvect8 |           |   |   |     |  |   |   |   |  |  |
| ----- |    |    |    |    |    |    |   |            |   |         |           |           |   |   |     | R7 ← PC', PC ← mem[ZEXT(trapvect8)]                  |   |   |   |  |  |
| ----- |    |    |    |    |    |    |   |            |   |         |           |           |   |   |     | ; Unused Opcode                                      |   |   |   |  |  |
|       | 1  | 1  | 0  | 1  |    |    |   |            |   |         |           |           |   |   |     |  |   |   |   |  |  |
| ----- |    |    |    |    |    |    |   |            |   |         |           |           |   |   |     | Initiate illegal opcode exception                    |   |   |   |  |  |

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0