Examination 4
In Class (50 minutes)
Friday, May 11, 2012
Weight: 17.5%

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

This exam has 11 pages, including a blank page at the end. Plan your time carefully, since some problems are longer than others. You must turn in pages 1 to 11.

LAST NAME: __________________________________________________________

FIRST NAME: _________________________________________________________

SECTION: ____________________________________________________________

CAMPUS ID# _________________________________________________________

EMAIL ID ____________________________________________________________
<table>
<thead>
<tr>
<th>Question</th>
<th>Maximum Points</th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>30</strong></td>
<td></td>
</tr>
</tbody>
</table>
Problem 1 (6 points)

An LC-3 assembly language program is given below.

```asm
.ORIG x3000
LEA R0, STR1
LEA R1, STR2
LOOP LDR R2, R0, #0
BRz DIFF
LDR R3, R1, #0
BRz DIFF
ADD R0, R0, #1
ADD R1, R1, #1
BRnzp LOOP
DIFF LDR R2, R0, #0
LDR R3, R1, #0
NOT R3, R3
ADD R3, R3, #1
ADD R3, R2, R3
BRp OUT1
BRn OUT2
BRz OUT3
OUT1 LEA R0, STR1
BRnzp DONE
OUT2 LEA R0, STR2
BRnzp DONE
OUT3 LEA R0, STR3
DONE PUTC
HALT

STR1 .STRINGZ "There’s a lady who’s sure all that glitters is gold"
STR2 .STRINGZ "Unsealed on a porch a letter sat"
STR3 .STRINGZ "I close my eyes only for a moment and the moment’s gone"
```

Note: there are 52 characters in the first string (including null)

a. Fill in the symbol table created by the first pass of the assembler on the above program for the following labels / symbols (3 points)

<table>
<thead>
<tr>
<th>Label</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOOP</td>
<td></td>
</tr>
<tr>
<td>DIFF</td>
<td></td>
</tr>
<tr>
<td>OUT1</td>
<td></td>
</tr>
<tr>
<td>OUT2</td>
<td></td>
</tr>
<tr>
<td>STR1</td>
<td></td>
</tr>
<tr>
<td>STR2</td>
<td></td>
</tr>
</tbody>
</table>


b. What does the above program do? Give your answer in one or two sentences. (1 point)

c. What is the output of this program? (2 points)

Problem 2 (4 points) 1 point each
Answer the following short answer questions in one or two sentences.
  a. What are the two methods for accessing I/O devices?

  b. Define and contrast callee-save and caller-save.

  c. Give one example of a syntax error using the AND immediate instruction and explain the error.

  d. Is there a problem with using X1 as a label in an LC-3 assembly program? Why or why not?
Problem 3 (5 points)
The following LC-3 subroutine implements the PUTS service routine. This means it will output the string stored at the address stored in R0 and then return to normal execution.

```
.ORIG x0227
ST R1, R1_TMP
ST R2, R2_TMP
L1 LDI R2, DSR

LDR R1, R0, #0
BRz DONE

ADD R0, R0, #1
BRnzp L1
DONE LD R2, R2_TMP
LD R1, R1_TMP

DSR .FILL xFE04 ; Address of DSR
DDR .FILL xFE06 ; Address of DDR
R1_TMP .FILL 0
R2_TMP .FILL 0
```

a. Fill in the blanks. **There should be one instruction per line and no NOPs.** (3 points)

b. Assume the above assembly code is a service routine that is a part of the LC-3 OS. Suppose we can call it using TRAP x22. Where is the address of the trap vector, and what are its contents? (2 points)

**Give all answers in hex**

<table>
<thead>
<tr>
<th>Address of TRAP vector</th>
<th>Contents at this memory location</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Problem 4 (4 points)

The following program reads two inputs from memory locations A and B and compares their contents. If \( M[A] = M[B] \), the program terminates with the value 1 in memory location COMPARE_RESULT, otherwise it stores zero in COMPARE_RESULT and terminates. What is the problem with the following code? How can you resolve it? Answer in one or two sentences in the box provided.

```
.ORIG x3000
LD R0, A
LD R1, B
JSR COMPARE
HALT

COMPARE
ST R7, R7_TMP
JSR NEG ; compute -R0
ADD R0, R0, R1 ; compute R1-R0
BRZ STORE_ONE
STORE ZERO
LD R0, ZERO
ST R0, COMPARE_RESULT
BR END_COMP
STORE ONE
LD R0, ONE
ST R0, COMPARE_RESULT
END_COMP
LD R7, R7_TMP
RET

NEG
ST R7, R7_TMP
NOT R0, R0
ADD R0, R0, #1
LD R7, R7_TMP
RET

A .BLKW 1
B .BLKW 1
R7_TMP .BLKW 1
COMPARE_RESULT .BLKW 1

ZERO .FILL 0
ONE .FILL 1
.END
```

Answer:
Problem 5 (2 points)

A programmer decides to design a variant of the LC-3 that does not need a keyboard status register. Instead, he creates a readable/writable keyboard data and status register (KBDSR). With the KBDSR, a program requiring keyboard input waits until a nonzero value appears in the KBDSR and reads this value into R0. The nonzero value is the ASCII value of the last key pressed. Then the program writes a zero into the KBDSR indicating that it has read the data. Fill in the following blanks to implement this new scheme. (1 instruction per line)

\[
\begin{array}{ll}
\text{ZERO} & \text{LDI R0, A} \\
& \text{BRz ZERO} \\
\text{NONZERO} & \hline \\
& \text{BRnzp NEXT_TASK} \\
A & \text{.FILL xFE00 ; Address of KBDSR}
\end{array}
\]
Problem 6 (5 points)

Select **only one answer** choice for the following questions about LC-3:

1. Suppose \texttt{JSRR R5} is stored at memory location x4009 and R5 contains x3002. The execution of JSRR instruction will cause:
   a. R7 to be loaded with x4010 and the PC to be loaded with x3002
   b. R7 to be loaded with x3003 and the PC to be loaded with x4009
   c. R7 to be loaded with x400A and the PC to be loaded with x3002
   d. R7 to be loaded with x3002 and the PC to be loaded with x400A

2. Which of the following is **not true** about interrupt driven I/O?
   a. It is more efficient than polling
   b. The device controls the interaction by sending a special signal to the processor when it is ready
   c. The processor must routinely check the status register for the device until new data arrives or the device is ready
   d. It has built in priority levels for different device requests

3. At the end of which phase of instruction cycle does an interrupt-driven I/O processor test for an interrupt signal?
   a. Fetch
   b. Decode
   c. Execute
   d. Store result

4. In LC-3, in what range of addresses are the trap vectors stored?
   a. x0000 - x00FF
   b. xFDFF - xFFFF
   c. xFF00 - xFFFF
   d. xFE00 - xFFFF

5. Which one is the highest priority level?
   a. PL1
   b. PL7
   c. PL0
   d. They are all equal in priority
Problem 7 (4 points)
Suppose the LC-3 is modified such that the TRAP vector table exists in the range of addresses from 0x0000 to 0x0007. The following tables show the TRAP vector table and the relevant service routines.

TRAP vector table:

<table>
<thead>
<tr>
<th>Address</th>
<th>Memory Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000</td>
<td>0x000A</td>
</tr>
<tr>
<td>0x0001</td>
<td>0x000F</td>
</tr>
<tr>
<td>0x0002</td>
<td>0x000E</td>
</tr>
<tr>
<td>0x0003</td>
<td>0x000A</td>
</tr>
<tr>
<td>0x0004</td>
<td>0x000B</td>
</tr>
<tr>
<td>0x0005</td>
<td>0x000D</td>
</tr>
<tr>
<td>0x0006</td>
<td>0x000F</td>
</tr>
<tr>
<td>0x0007</td>
<td>0x000C</td>
</tr>
</tbody>
</table>

Service routines:

<table>
<thead>
<tr>
<th>Address</th>
<th>Label</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000A</td>
<td></td>
<td>LD R0, L2</td>
</tr>
<tr>
<td>0x000B</td>
<td></td>
<td>ADD R0, R0, #10</td>
</tr>
<tr>
<td>0x000C</td>
<td></td>
<td>RET</td>
</tr>
<tr>
<td>0x000D</td>
<td>L1</td>
<td>LD R0, L3</td>
</tr>
<tr>
<td>0x000E</td>
<td></td>
<td>BRn L1</td>
</tr>
<tr>
<td>0x000F</td>
<td></td>
<td>RET</td>
</tr>
<tr>
<td>0x0010</td>
<td>L2</td>
<td>.FILL x100A</td>
</tr>
<tr>
<td>0x0011</td>
<td>L3</td>
<td>.FILL x100B</td>
</tr>
</tbody>
</table>

Suppose we execute a TRAP x3 instruction at memory location xABCD. Show the MAR trace for the TRAP x3 call. **Give all the answers in HEX.**

**You may not need to use all the blank boxes.**

<table>
<thead>
<tr>
<th>MAR Trace</th>
</tr>
</thead>
<tbody>
<tr>
<td>xABCD</td>
</tr>
</tbody>
</table>
LC-3 Instruction Set

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. SEXT(immediate): zero-extend immediate to 16 bits.

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
| 0 0 0 1 | DR | SR1 | 0 | 0 | 0 | SR2 | | ADD DR, SR1, SR2 ; Addition | +|-----------------------------| DR SR1 + SR2 also setcc(); | |-----------------------------| ADD DR, SR1, imm5 ; Addition with Immediate | +|-----------------------------| DR SR1 + SEXT(imm5) also setcc(); | |-----------------------------| AND DR, SR1, SR2 ; Bit-wise AND | +|-----------------------------| DR SR1 AND SR2 also setcc(); | |-----------------------------| AND DR,SR1,imm5 ; Bit-wise AND with Immediate | +|-----------------------------| DR SR1 AND SEXT(imm5) also setcc(); | |-----------------------------| BRx,label (where x=(n,z,p,zp,zp,nz,pnz)); Branch | |-----------------------------| GO ((n and n) OR (z AND z) OR (p AND p)) | +|-----------------------------| if(GO is true) then PC=PC+SEXT(PCoffset9) | |-----------------------------| JMP BaseR ; Jump | +|-----------------------------| JSR BaseR ; Jump to Subroutine | |-----------------------------| JSRRA BaseR ; Jump to Subroutine in Register | +|-----------------------------| temp PC', PC BaseR, R7 temp | |-----------------------------| LD DR, label ; Load PC-relative | +|-----------------------------| DR mem[PC' + SEXT(PCoffset9)] also setcc(); | +|-----------------------------| LDI DR, label ; Load Indirect | +|-----------------------------| Dmem[mem[PC'+SEXT(PCoffset9)]] also setcc(); | |-----------------------------| LDR DR, BaseR, offset6 ; Load Base+Offset | +|-----------------------------| DR mem[BaseR + SEXT(offset6)] also setcc(); | +|-----------------------------| LEA DR, label ; Load Effective Address | +|-----------------------------| DR PC' + SEXT(PCoffset9) also setcc(); | |-----------------------------| NOT DR, SR ; Bit-wise Complement | +|-----------------------------| DR NOT(SR) also setcc(); | |-----------------------------| RET ; Return from Subroutine | +|-----------------------------| RTI ; Return from Interrupt | +|-----------------------------| See textbook (2nd Ed. page 537). | |-----------------------------| ST SR, label ; Store PC-relative | +|-----------------------------| mem[PC' + SEXT(PCoffset9)] SR | |-----------------------------| STI, SR, label ; Store Indirect | +|-----------------------------| mem[mem[PC'+SEXT(PCoffset9)]] SR | +|-----------------------------| STR SR, BaseR, offset6 ; Store Base+offset | +|-----------------------------| mem[BaseR + SEXT(offset6)] SR | |-----------------------------| TRAP ; System Call | +|-----------------------------| R7 PC', PC mem[SEXT(trapvent8)] | +|-----------------------------| ; Unused Opcode | |-----------------------------| Initiate illegal opcode exception |