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

Prof. Gurindar Sohi
TAs: Rebecca Lam, Mona Jalal, Preeti Agarwal, Pradip Vallathol

Midterm Examination 4
In Class (50 minutes)
Wednesday, May 8, 2013
Weight: 17.5%

NO: BOOK(S), NOTE(S), OR CALCULATORS OF ANY SORT.
The exam has 12 pages. Circle your final answers. Plan your time carefully since some problems are longer than others. You must turn in the pages 1-9. The LC-3 instruction set is provided to you on the last page.

LAST NAME: _____________________________________________________________
FIRST NAME: _____________________________________________________________
ID# ________________________________________________________________
<table>
<thead>
<tr>
<th>Problem</th>
<th>Maximum Points</th>
<th>Points Earned</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>30</td>
<td></td>
</tr>
</tbody>
</table>
Problem 1: Multiple Choice Questions (4 Points)

For the following questions, select the best answer. Choose only one answer per question.

i. The TRAP instructions in LC-3 are similar to which of the following instructions in terms of the number of memory accesses that are made to the fetch and execute the instruction?
   a. LD
   b. LDR
   c. LEA
   d. LDI

ii. Which of the following is not true about polling?
   a. The CPU keeps monitoring status register.
   b. CPU cannot perform other tasks during polling.
   c. Polling requires changes to the Fetch and Decode logic of the CPU.
   d. Polling wastes a lot of CPU time.

iii. Which of the following is not true about comments in an LC-3 program?
   a. Anything after the semicolon is a comment.
   b. They can be used multiple times in a program.
   c. It is used by the assembler to understand the program.
   d. Can be used to separate pieces of the program.

iv. JSRR R5 is equivalent to
   a. LEA R5, #1
      JMP R7
   b. LEA R5, #1
      JMP R5
   c. LEA R7, #1
      JMP R5
   d. LEA R7, #1
      JMP R7
   e. All of the above are equivalent
Problem 2: Assembly Process (5 Points)
Answer the questions below for the following program:

```
.ORIG x4000
LD R2, LOW_A
NOT R2, R2
ADD R2, R2, #1
LEA R0, STRG
; Comment 1
L1 LDR R1, R0, #0
BRz DONE
ADD R3, R1, R2
BRnp SKIP

LD R1, UPP_A
STR R1, R0, #0
SKIP ADD R0, R0, #1
BRnzp L1
DONE LEA R0, STRG
PUTS ; Display the string at the address in R0
HALT
LOW_A .FILL x61 ; ASCII Character 'a'
STRG .STRINGZ "Salt and Pepper"
UPP_A .FILL x41 ; ASCII Character 'A'
.END
```

(a) Fill out the following symbol table: (3 Points)

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1</td>
<td></td>
</tr>
<tr>
<td>SKIP</td>
<td></td>
</tr>
<tr>
<td>DONE</td>
<td></td>
</tr>
<tr>
<td>LOW_A</td>
<td></td>
</tr>
<tr>
<td>STRG</td>
<td></td>
</tr>
<tr>
<td>UPP_A</td>
<td></td>
</tr>
</tbody>
</table>

(b) What is the output of this program? (2 Points)
Problem 3: Assembly Errors

Identify the assembly errors in the following assembly program.

```
.ORIG x3000

ADD R1, R2, #21
; OR R2, R3, R4

LOOP AND R3, R3, #0
ADD R4, R4, R4
ADD R3, R3, #1
BRzp NEXT

STRG .STRINGZ "Error"

HALT STR R4, R4, #16
TRAP x25

.END
```

(a)

(b)

(c)
Problem 4: TRAPS

Suppose the following LC-3 subroutine implements a new service routine called GETS. The subroutine will store the input string starting at the address in R0 and then return to normal execution. It performs this operation by repeatedly taking input characters from the keyboard and storing it in the location specified by R0 until it sees the 'n' character.

Note: The most significant bit of the KBSR is 1 if keyboard has received a new character.

a. Fill in the blanks. There should be only one instruction per line. (4 Points)

```
.ORIG x0650
ST R0, R0_TMP
ST R1, R1_TMP
ST R2, R2_TMP
L1 LDI R1, KBSR
(a) ______________ ; Check KBSR
(b) ___ R2, KBDR ; Load value in the KBDR into R2
LD R1, NEGCHAR
ADD R1, R1, R2
BRz DONE ; Check for 'n'
STR R2, R0, #0
ADD R0, R0, #1
BRnzp L1
DONE (c) ______________ ; Store NULL CHAR
STR R2, R0, #0
LD R2, R2_TMP
LD R1, R1_TMP
LD R0, R0_TMP
(d) ______________
```

```
KBSR .FILL xFE00 ; Address of KBSR
KBDR .FILL xFE02 ; Address of KBDR
NEGCHAR .FILL xFFF6 ; Negative value of character 'n'
R0_TMP .FILL 0
R1_TMP .FILL 0
R2_TMP .FILL 0
.END
```

b. Assume the above assembly code is a service routine that can be called using TRAP x55. What is the address of the corresponding System Control Block entry and what are its contents? Give your answer in hex. (1 Point)

<table>
<thead>
<tr>
<th>Address of trap vector table entry</th>
<th>Contents at this memory location</th>
</tr>
</thead>
</table>
Problem 5: Subroutines (5 Points)

a. There is a problem with the below assembly code segment for a subroutine called PUTCH. What is it, and how can you fix the error? (2 Points)

```assembly
.ORIG x3010

PUTCH

ST R0, TMP_R0
ADD R0, R4, 0
OUT ; TRAP x21 which displays the
     ; character in R0
LD R0, TMP_R0
RET

TMP_R0 .FILL 0
.END
```

b. Is the above subroutine PUTCH a callee-save or caller-save subroutine? Explain. (1 Point)

c. Given the following initial values of registers, what are the values of the registers after the execution of an instruction at address x3030: JSR PUTCH; and before the execution of the first instruction of the subroutine. (2 Points)

<table>
<thead>
<tr>
<th>Register</th>
<th>Initial</th>
<th>Final</th>
</tr>
</thead>
<tbody>
<tr>
<td>R0</td>
<td>0x3010</td>
<td></td>
</tr>
<tr>
<td>R4</td>
<td>0x3030</td>
<td></td>
</tr>
<tr>
<td>R7</td>
<td>0x3010</td>
<td></td>
</tr>
<tr>
<td>PC</td>
<td>0x3030</td>
<td></td>
</tr>
</tbody>
</table>
Problem 6: I/O (3 Points)
Let us monitor the contents of the KBSR (Keyboard Status Register), KBDR (Keyboard Data Register), DSR (Display Status Register) and DDR (Display Data Register) during the execution of TRAP x23 (IN) in LC-3. The leftmost bit of the block is the MSB and the rightmost bit is the LSB of the registers. Note: TRAP x23 (IN) prints prompt to console, read and echo a character from the keyboard.

Below fill in the contents of the different registers at the different steps b, c, and d during the execution of the trap handler for TRAP x23.

a. Initial State:

**KBDR**

| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**KBSR**

| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

b. The user types in character “S” on the keyboard, but the character is not read.

**KBDR**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

**KBSR**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

c. The character “S” is read from the keyboard and no new character is typed.

**KBSR**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

d. The display is ready but the character is not yet written to the Display Data Register.

**DSR**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
Problem 7: General Questions  

(5 Points)

Answer the following short answer questions using 1-2 sentences.

a. What is the difference between Memory Mapped I/O and Special I/O instructions?  
(2 Points)

b. Why are two passes required during the assembly process?  
(1 Point)

c. What is the difference between a subroutine call and a branch instruction?  
(1 Point)

d. What do labels represent in an LC-3 assembly program?  
(1 Point)
Scratch page. You do not need to turn this page in.
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. SEXT(immediate): zero-extend immediate to 16 bits.

---

ADD DR, SR1, SR2 ; Addition

ADD DR, SR1, imm5 ; Addition with Immediate

AND DR, SR1, SR2 ; Bit-wise AND

AND DR, SR1, imm5 ; Bit-wise AND with Immediate

DR ← SR1 AND SEXT(imm5) also setcc()

---

PCoffset9

---

DR ← SR1 + SEXT(imm5) also setcc()

---

AND DR, SR1, SR2 ; Bit-wise AND

---

PCoffset9

---

DR ← SR1 AND SEXT(imm5) also setcc()

---

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 label ; Jump to Subroutine

---

PCoffset11

---

R7 ← PC', PC ← PC' + SEXT(PCoffset11)

---

JSRR BaseR ; Jump to Subroutine in Register

---

BaseR | 0 0 0 0 0 0

---

LD PC', PC ← BaseR, R7 ← temp

---

LD DR, label ; Load PC-relative

---

PCoffset9

---

DR ← mem[PC' + SEXT(PCoffset9)] also setcc()

---

LDI DR, label ; Load Indirect

---

PCoffset9

---

DR ← mem[mem[PC' + SEXT(PCoffset9)]] also setcc()

---

LDR DR, BaseR, offset6 ; Load Indirect Offset

---

PCoffset9

---

DR ← mem[BaseR + SEXT(offset6)] also setcc()

---

LEA DR, label ; Load Effective Address

---

PCoffset9

---

DR ← PC' + SEXT(PCoffset9) also setcc()

---

NOT DR, SR ; Bit-wise Complement

---

DR ← NOT(SR) also setcc()

---

RET ; Return from Subroutine

---

PCoffset9

---

R7

---

RTI ; Return from Interrupt

---

See textbook (2nd Ed. page 537).

---

ST SR, label ; Store PC-relative

---

SR | 0 0 0 0 0 0

---

mem[PC' + SEXT(PCoffset9)] ← SR

---

STI SR, label ; Store Indirect

---

SR | 0 0 0 0 0 0

---

mem[mem[PC' + SEXT(PCoffset9)]] ← SR

---

STR SR, BaseR, offset6 ; Store Indirect Offset

---

SR | 0 0 0 0 0 0

---

mem[BaseR + SEXT(offset6)] ← SR

---

TRAP ; System Call

---

mem[trapvector]

---

R7 ← PC', PC ← mem[SEXT(trapvector)]

---

Unused Opcode

---

Initiate illegal opcode exception
## ASCII Table

<table>
<thead>
<tr>
<th>Character</th>
<th>Hex</th>
<th>Character</th>
<th>Hex</th>
<th>Character</th>
<th>Hex</th>
<th>Character</th>
<th>Hex</th>
</tr>
</thead>
<tbody>
<tr>
<td>nul</td>
<td>00</td>
<td>sp</td>
<td>20</td>
<td>@</td>
<td>40</td>
<td>`</td>
<td>60</td>
</tr>
<tr>
<td>soh</td>
<td>01</td>
<td>!</td>
<td>21</td>
<td>A</td>
<td>41</td>
<td>a</td>
<td>61</td>
</tr>
<tr>
<td>stx</td>
<td>02</td>
<td>“</td>
<td>22</td>
<td>B</td>
<td>42</td>
<td>b</td>
<td>62</td>
</tr>
<tr>
<td>etx</td>
<td>03</td>
<td>#</td>
<td>23</td>
<td>C</td>
<td>43</td>
<td>c</td>
<td>63</td>
</tr>
<tr>
<td>eot</td>
<td>04</td>
<td>$</td>
<td>24</td>
<td>D</td>
<td>44</td>
<td>d</td>
<td>64</td>
</tr>
<tr>
<td>enq</td>
<td>05</td>
<td>%</td>
<td>25</td>
<td>E</td>
<td>45</td>
<td>e</td>
<td>65</td>
</tr>
<tr>
<td>ack</td>
<td>06</td>
<td>&amp;</td>
<td>26</td>
<td>F</td>
<td>46</td>
<td>f</td>
<td>66</td>
</tr>
<tr>
<td>bel</td>
<td>07</td>
<td>‘ (Aposr.)</td>
<td>27</td>
<td>G</td>
<td>47</td>
<td>g</td>
<td>67</td>
</tr>
<tr>
<td>bs</td>
<td>08</td>
<td>(</td>
<td>28</td>
<td>H</td>
<td>48</td>
<td>h</td>
<td>68</td>
</tr>
<tr>
<td>ht</td>
<td>09</td>
<td>)</td>
<td>29</td>
<td>I</td>
<td>49</td>
<td>i</td>
<td>69</td>
</tr>
<tr>
<td>lf</td>
<td>0A</td>
<td>*</td>
<td>2A</td>
<td>J</td>
<td>4A</td>
<td>j</td>
<td>6A</td>
</tr>
<tr>
<td>vt</td>
<td>0B</td>
<td>+</td>
<td>2B</td>
<td>K</td>
<td>4B</td>
<td>k</td>
<td>6B</td>
</tr>
<tr>
<td>ff</td>
<td>0C</td>
<td>, (Comma)</td>
<td>2C</td>
<td>L</td>
<td>4C</td>
<td>l</td>
<td>6C</td>
</tr>
<tr>
<td>cr</td>
<td>0D</td>
<td>-</td>
<td>2D</td>
<td>M</td>
<td>4D</td>
<td>m</td>
<td>6D</td>
</tr>
<tr>
<td>so</td>
<td>0E</td>
<td>. (Period)</td>
<td>2E</td>
<td>N</td>
<td>4E</td>
<td>n</td>
<td>6E</td>
</tr>
<tr>
<td>si</td>
<td>0F</td>
<td>/</td>
<td>2F</td>
<td>O</td>
<td>4F</td>
<td>o</td>
<td>6F</td>
</tr>
<tr>
<td>dle</td>
<td>10</td>
<td>0</td>
<td>30</td>
<td>P</td>
<td>50</td>
<td>p</td>
<td>70</td>
</tr>
<tr>
<td>dc1</td>
<td>11</td>
<td>1</td>
<td>31</td>
<td>Q</td>
<td>51</td>
<td>q</td>
<td>71</td>
</tr>
<tr>
<td>dc2</td>
<td>12</td>
<td>2</td>
<td>32</td>
<td>R</td>
<td>52</td>
<td>r</td>
<td>72</td>
</tr>
<tr>
<td>dc3</td>
<td>13</td>
<td>3</td>
<td>33</td>
<td>S</td>
<td>53</td>
<td>s</td>
<td>73</td>
</tr>
<tr>
<td>dc4</td>
<td>14</td>
<td>4</td>
<td>34</td>
<td>T</td>
<td>54</td>
<td>t</td>
<td>74</td>
</tr>
<tr>
<td>nak</td>
<td>15</td>
<td>5</td>
<td>35</td>
<td>U</td>
<td>55</td>
<td>u</td>
<td>75</td>
</tr>
<tr>
<td>syn</td>
<td>16</td>
<td>6</td>
<td>36</td>
<td>V</td>
<td>56</td>
<td>v</td>
<td>76</td>
</tr>
<tr>
<td>etb</td>
<td>17</td>
<td>7</td>
<td>37</td>
<td>W</td>
<td>57</td>
<td>w</td>
<td>77</td>
</tr>
<tr>
<td>can</td>
<td>18</td>
<td>8</td>
<td>38</td>
<td>X</td>
<td>58</td>
<td>x</td>
<td>78</td>
</tr>
<tr>
<td>em</td>
<td>19</td>
<td>9</td>
<td>39</td>
<td>Y</td>
<td>59</td>
<td>y</td>
<td>79</td>
</tr>
<tr>
<td>sub</td>
<td>1A</td>
<td>:</td>
<td>3A</td>
<td>Z</td>
<td>5A</td>
<td>z</td>
<td>7A</td>
</tr>
<tr>
<td>esc</td>
<td>1B</td>
<td>;</td>
<td>3B</td>
<td>[</td>
<td>5B</td>
<td>{</td>
<td>7B</td>
</tr>
<tr>
<td>fs</td>
<td>1C</td>
<td>&lt;</td>
<td>3C</td>
<td>\</td>
<td>5C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>gs</td>
<td>1D</td>
<td>=</td>
<td>3D</td>
<td>]</td>
<td>5D</td>
<td>)</td>
<td>7D</td>
</tr>
<tr>
<td>rs</td>
<td>1E</td>
<td>&gt;</td>
<td>3E</td>
<td>^</td>
<td>5E</td>
<td>~</td>
<td>7E</td>
</tr>
<tr>
<td>us</td>
<td>1F</td>
<td>?</td>
<td>3F</td>
<td>_ (Undrsc)</td>
<td>5F</td>
<td>del</td>
<td>7F</td>
</tr>
</tbody>
</table>