CS/ECE 252: INTRODUCTION TO COMPUTER ENGINEERING

UNIVERSITY OF WISCONSIN—MADISON

Professor Guri Sohi
TAs: Newsha Ardalani and Rebecca Lam

Examination 4
In Class (50 minutes)
Wednesday, Dec 14, 2011
Weight: 17.5%

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

This exam has 12 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 9.

LAST NAME: ____________________________________________
FIRST NAME: ____________________________________________
SECTION: ________________________________________________
ID# ______________________________________________________
<table>
<thead>
<tr>
<th>Question</th>
<th>Maximum Point</th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>5</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>
Q1. Syntax Errors in LC-3 Assembly Code (3 points)

a. Circle any illegal labels in an assembly language programs: (**1 point**)

- ADD
- END
- .FILL
- BLKW
- OR
- NAND

b. The following program has multiple syntax errors. One of them, along with an explanation of the error, is indicated in the table below. In the two blank rows of the table, identify and explain two more syntax errors. (**2 points**)

```
.ORIG x3000
LDI R1, COUNT
AND R1, R1, M1
LOOP LEA R0, x2FF
ADD R0, R1, R2
BRz LOOP
NOT R1, R1, R1
HALT
M1 .FILL x4000
COUNT .FILL #100
.END
```

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>AND R1, R1, M1</td>
<td>Can’t use a label as operand</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Q2. Two-Pass Assembly Process (5 points)**

An assembly language LC-3 program is given below:

```
.ORIG    x3000
L1      LEA    R1, L1
        AND    R2, R2, x0
        ADD    R2, R2, x3
        LD     R3, P1
L2      LDR    R0, R1, xC
        TRAP   x21 ; OUT (Write char)
        ADD    R3, R3, #-1
        BRz    GLUE
        ADD    R1, R1, R2
        BRnzp  L2
GLUE    HALT
P1      .FILL  x7
        .STRINGZ “GWHoei1Tdchboymreee”
 .END
```

a. Fill in the symbol table created by the first pass of the assembler on the above program. (2 points)

<table>
<thead>
<tr>
<th>Symbol Name</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1</td>
<td></td>
</tr>
<tr>
<td>L2</td>
<td></td>
</tr>
<tr>
<td>GLUE</td>
<td></td>
</tr>
<tr>
<td>P1</td>
<td></td>
</tr>
</tbody>
</table>

b. After the program is assembled and loaded, what binary pattern is stored in memory location x3005? (1 point)

c. What is the output of this program? (2 points)
Q3. Logical Error (4 points)

We want the following code to shift the value at memory location M1 to the left by the number of bits stored at memory location M2, but there is one error in this code.

```
.ORIG x3000
LD    R1, M1
LD    R2, M2
LOOP  BRz  DONE
    ADD  R2, R2, #-1
    ADD  R1, R1, R1
    BRnzp LOO
DONE  HALT

M1    .FIL    x000C
M2    .FIL    x0004
.END
```

(a) How many times does the instruction labeled LOOP get executed? Explain. (2 points)

(b) What is wrong with this program? Explain. (2 points)
Q4. Trap Handling (3 points)

The figure shown below represents the flow control from a user program to an OS service routine and back when a trap instruction is called. The flow control goes from A within a user program, to B the operating system service routine, back to the user program C. Fill out the three empty boxes below corresponding with question marks. Boxes 1 and 2 should be filled with addresses and box 3 should be filled with an instruction.

Write your answers in \texttt{hexadecimal}.


d| 1 | 2 | 3
---|---|---|---
  |   |   |   

Q5. Traps and subroutines (5 points)

An LC-3 programmer wrote the code below to read 10 single digit decimal number from the keyboard, compute their average, and display the ceiling of the resultant average on the monitor.

Fill in the blanks below with assembly code to complete the program.

```
.ORIG x3000

AND R2, R2, #0 ; R2 keeps track of the sum
LD R6, CHtoD ; Char->Digit template
LD R5, DtoCH ; Digit->Char template
LD R7, COUNT ; Initialize to 10
ST _, SAVEDREG ; Save ?? upon call of trap
AGAIN TRAP x23 ; Get char
LD ___, ___, R6 ; Restore ?? before continuing
ADD R0, R0, R6 ; Convert to number
ADD R2, R2, R0 ; Add the new number to the sum
ADD R7, R7, -1 ; Decr counter
ST __, SAVEDREG ;
BRp AGAIN ; More digit?
LD R1, COUNT ;

ADD R0, R0, R5 ; Convert to char
TRAP x21 ; Output char
HALT

; DIV subroutine
; Args: R2,R1 RET: R0=R2/R1
DIV AND R0, R0, #0 ; Initialize to 0
   NOT R1, R1 ;
   ADD R1, R1, #1 ; Negate R1
LOOPDIV ADD R0, R0, #1

   BRP LOOPDIV
RET

DtoCH .FILL x0030
CHtoD .FILL xFFD0
COUNT .FILL #10

SAVEDREG .BLKW 1

.END
```
Q6. Short Answer Questions (5 points)

Answer the following short answer questions in one or two sentences.

1. What problem could occur if the display hardware does not check the DSR before writing to the DDR? (1 point)

2. What is the difference between asynchronous and synchronous I/O? (1 point)

3. What is the difference between memory mapped I/O and special I/O instructions? (1 point)

4. Give one potential benefit and one potential drawback of RFID implants (2 points).
Q7. General Questions (5 points, 1 point each)

Circle the best answer for the following questions about LC-3:

1. Which of the following can be used only once per file?
   a. .STRINGZ
   b. .BLKW
   c. .ORIG
   d. .FILL

2. Which of the following is true about “callee-save”?
   a. Used by calling routine to save and restore registers that will be used in the routine
   b. Save R7 before calling TRAP
   c. Save R0 before calling TRAP x23
   d. Used by called routine to save registers used by the routine

3. Suppose JSR label is stored at memory location x3000. After the instruction is executed, which of the following is true if label=x3050 and R7=x4000 before execution?
   a. R7 = x3050
   b. R7 = x3001
   c. R7 = x3000
   d. R7 = x4000

4. Which bit in the KBSR is the interrupt enable bit?
   a. 15
   b. 14
   c. 13
   d. 12

5. Which of the following is not true about interrupt driven I/O?
   a. The device controls the interaction by sending a special signal to the processor when it is ready
   b. It is more efficient than polling
   c. It has built in priority levels for different device requests
   d. The processor must routinely check the status register for the device until new data arrives or the device is ready
<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>&quot;</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>'</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>,</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>.</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>_</td>
<td>5F</td>
<td>del</td>
<td>7F</td>
</tr>
</tbody>
</table>
LC-3 Instruction Set (Entered by Mark D. Hill on 03/14/2007; last update 03/15/2007)


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 | SR2 | DR SR1 + SR2 also setcc()
++-------------------------------+ ADD DR, SR1, imm5 ; Addition with Immediate
| 0 0 0 | DR | SR1 | l | imm5 |
++-------------------------------+ DR SR1 + SEXT(imm5) also setcc()
++-------------------------------+ AND DR, SR1, SR2 ; Bit-wise AND
| 0 1 0 1 | DR | SR1 | 0 | 0 | SR2 | DR SR1 AND SR2 also setcc()
++-------------------------------+ AND DR, SR1, imm5 ; Bit-wise AND with Immediate
| 0 1 0 1 | DR | SR1 | l | imm5 |
++-------------------------------+ DR SR1 AND SEXT(imm5) also setcc()
++-------------------------------+ BRx.label (where x=n,z,p,zp,sp,ns,zp);
| 0 0 0 | 0 | n | x | p | PCoffset9 | Branch
| 0 0 0 | 0 | n | x | p |
++-------------------------------+ if (GO is true) then PCPC' = SEXT(PCoffset9)
++-------------------------------+ JMP BaseR ; Jump
| 1 1 0 0 0 | 0 | 0 | 0 | BaseR | 0 0 0 0 0 0 0 |
++-------------------------------+ PC BaseR
++-------------------------------+ JSR label ; Jump to Subroutine
| 0 1 0 0 | 1 | | PCoffset11 |
| 0 1 0 | 0 | 0 | 0 | 0 | BaseR | 0 0 0 0 0 0 0 |
++-------------------------------+ temp PC', PC BaseR, R7 temp
| 0 1 0 0 0 0 0 | BaseR | 0 0 0 0 0 0 0 |
++-------------------------------+ LD DR, label ; Load PC-Relative
| 0 0 1 0 | DR | PCoffset9 |
| 0 0 1 0 | DR |
++-------------------------------+ DR mem[PC' + SEXT(PCoffset9)] also setcc()
++-------------------------------+ LDI DR, label ; Load Indirect
| 0 1 1 | DR | PCoffset9 |
++-------------------------------+ DR mem[mem[PC' + SEXT(PCoffset9)]] also setcc()
++-------------------------------+ LDR DR, BaseR, offset6 ; Load BaseOffset
| 0 1 1 | DR | BaseR | offset6 |
++-------------------------------+ DR mem[BaseR + SEXT(offset6)] also setcc()
++-------------------------------+ LEA, DR, label ; Load Effective Address
| 1 1 1 | DR |
| 1 1 1 | DR |
++-------------------------------+ DR PC' + SEXT(PCoffset9) also setcc()
++-------------------------------+ NOT DR, SR ; Bit-wise Complement
| 1 0 0 | DR | SR | 1 | 1 | 1 | 1 | 1 |
++-------------------------------+ DR NOT(SR) also setcc()
++-------------------------------+ RET ; Return from Subroutine
| 1 1 0 0 0 | 0 | 0 | 0 | 0 | 0 |
++-------------------------------+ PC R7
++-------------------------------+ RTI ; Return from Interrupt
| 1 0 0 0 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
++-------------------------------+ See textbook (2nd Ed. page 537).
++-------------------------------+ ST SR, label ; Store PC-Relative
| 0 1 1 | SR |
| 0 1 1 | SR |
++-------------------------------+ 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 BaseOffset
| 0 1 1 1 | SR | BaseR | offset6 |
++-------------------------------+ mem[mem[BaseR + SEXT(offset6)]] SR
++-------------------------------+ TRAP ; System Call
| 1 1 | SR |
| 1 1 | SR |
++-------------------------------+ mem[BaseR + SEXT(offset6)] SR
++-------------------------------+ t
++-------------------------------+ 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 |