CS/ECE 252: INTRODUCTION TO COMPUTER ENGINEERING

UNIVERSITY OF WISCONSIN—MADISON

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

Midterm Examination 4
In Class (50 minutes)
Friday, May 6
Weight: 15%

This exam has 12 pages, including a blank page.
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, as well as a listing of the ASCII code.

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

LAST NAME: _____________________________________________________________

FIRST NAME: ___________________________________________________________

SECTION: ______________________________________________________________

ID# ________________________________________________________________
<table>
<thead>
<tr>
<th>Question</th>
<th>Maximum Points</th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>35</td>
<td></td>
</tr>
</tbody>
</table>
**1. Assembly Errors (4 Points)**

In the following template for an assembly program, circle any labels where the first pass of the assembler would report an error (if any).

```assembly
.ORIG x3000
Main ...
...
Loop ...
...
End ...
...
Add ...
...
Sum ...
...
Loop ...
...
And ...
...
Or ...
...
X10 ...
...
.END
```

1. Loop declared twice
2. And is an OPCODE, cannot be used as a label.
3. Add is an OPCODE, cannot be used as a label.
4. X10 is not a valid label.
2. Two-Pass Assembly Process (7 points)
An assembly language LC-3 program is given below:

```assembly
.ORIG x3000
AND R2, R2, #0
AND R3, R3, #0
LD R0, M0
LD R4, M1
LOOP BRz DONE
Jsr INCR3
ADD R2, R2, R0
Jsr DECR4
BR LOOP
DONE ST R2, ANSWER
Halt
INCR3 ADD R3, R3, #1
RET
DECR4 ADD R4, R4, #-1
RET
; Storage area for variables below:
ANSWER .BLKW #4
M0 .FILL x0007
M1 .FILL x0010
.END
```

a) Fill in the symbol table created by the assembler on the first pass of the above program. (4 points)

<table>
<thead>
<tr>
<th>Symbol Name</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOOP</td>
<td>3004</td>
</tr>
<tr>
<td>DONE</td>
<td>3009</td>
</tr>
<tr>
<td>INCR3</td>
<td>300B</td>
</tr>
<tr>
<td>DECR4</td>
<td>300D</td>
</tr>
<tr>
<td>ANSWER</td>
<td>300F</td>
</tr>
<tr>
<td>M0</td>
<td>3013</td>
</tr>
<tr>
<td>M1</td>
<td>3014</td>
</tr>
</tbody>
</table>
Once the symbol table is created, the assembler then creates a binary version (.obj) of the program. A portion is listed below. Three lines are missing the binary.

```
0101 0100 1010 0000 ;AND R2, R2, #0
0101 0110 1110 0000 ;AND R3, R3, #0
0010 0000 0001 0000 ;LD R0, M0
0010 0010 0001 0000 ;LD R4, M1
0000 0100 0000 0100 ;BRz DONE
  ____  ____  ____  ____ ;JSR INCR3
0001 0100 1000 0000 ;ADD R2, R2, R0
  ____  ____  ____  ____ ;JSR DECR4
0000 1111 1111 1011 ;BR LOOP
0011 0100 0000 0001 ;ST R2, RESULT
  ____  ____  ____  ____ ;HALT (TRAP x25)
0001 0110 1110 0001 ;ADD R3, R3, #1
1100 0001 1100 0000 ;RET
0001 1001 0011 1111;ADD R4, R4, #-1
1100 0001 1100 0000 ;RET
```

b) For each of the above missing lines, circle the correct binary value: (3 points)

**Missing Line #1**

1) 0100 1000 0000 0100
2) 0100 1000 0000 0101
3) 0100 1000 0000 1011

**Missing Line #2**

1) 0100 1000 0000 0100
2) 0100 1000 0000 0101
3) 0100 1000 0000 1101

**Missing Line #3**

1) 1111 0000 0010 0101
2) 1111 0000 0001 1001
3) 1111 0000 0000 1101
3. I/O in LC-3 (7 Points)

An LC-3 program is provided below:

```
.ORIG x3000
LD R0, ASCII
LD R1, NEG
AGAIN LDI R2, DSR
        BRzp AGAIN
STI R0, DDR
ADD R0, R0, #1
ADD R3, R0, R1
BRn AGAIN
HALT

ASCII .FILL x0043
NEG .FILL xFFB6
DSR .FILL xFE04 ; Address of DSR
DDR .FILL xFE06 ; Address of DDR
.END
```

a) What is the purpose of "STI R0, DDR"? (1 point)

This instruction writes the value of R0 into the Display data register. This is from where the Display picks up the character that it needs to display.

b) What is the purpose of reading the Display Status Register (DSR)? (2 points)

If we don’t check we could overwrite the previous value before it has been displayed by the monitor.

c) What is in DSR bit 14, but we are not using in this program? (1 point)

Interrupt enable bit

d) What does this program do? (3 points)

Prints “CDEFGHI”
4. Subroutines (7 Points)

In the code below, the Subroutine ONECHAR takes 1 character from the user (keyboard) and saves it into memory. The assembly code uses ONECHAR in a loop 6 times to input 6 characters and save them to memory. Finally, it prints the string to the screen.

```asm
:CODE TO INPUT AND PRINT 6 CHARACTERS

.ORIG x3000
AND R0, R0, #0 ; Initialize R0, our counter

LOOP
LEA R1, INPSTRING ; R1 now has base of INPSTRING
ADD R1, R1, R0 ; R1 now has base + offset = R0
ST R0, SAVREG1 ; SAVE R0
JSR ONECHAR ; Call Subroutine
LD R0, SAVREG1 ; Restore R0
ADD R0, R0, #1 ; Increment R0
LD R1, LENGTH ; Load R1 with minus length
ADD R1, R1, R0 ;
BRn LOOP ; loop till 6 characters are reached
LEA R0, INPSTRING ; Get ready to print
PUTS ; TRAP x22: Print string
HALT ; We're done

OVER
.Leave

ONECHAR
ST _R7_, SAVREG2 ; SAVE ?? upon entering the subroutine
GETC ; TRAP x20: Get a character from keyboard.
STR R0, R1, #0 ; Save keyboard input (R0 contains input)
LD _R7_, SAVREG2 ; Restore ?? before leaving
RET

LENGTH .FILL 0xFFFFA ; minus Length (-6)
KBSR .FILL 0xFE00
KBDR .FILL 0xFE02
SAVREG1 .FILL 0x0
SAVREG2 .FILL 0x0
INPSTRING .BLKW 7
.END
```

(a) Line 8 saves R0 before calling the subroutine ONECHAR. Briefly explain why this is necessary. (2 points)
R0 is going to be overwritten by GETC call in ONECHAR subroutine.

(b) What other register needs to be stored and restored inside the subroutine? Fill in lines 19 and 22 above. (2 points) R7 needs to be saved.

(c) In the program we save and restore registers two places, in lines 8 and 10 and again in lines 19 and 22. For lines 8 and 10, indicate whether it illustrates Caller-save or Callee-save. (1 point) Caller-save

(d) How many times will each character appear on the screen? Why? (2 point)
Once. GETC is used to read the character, so it doesn’t display. PUTS prints all the stored character at once.
5. General Questions (10 points)

Circle the best answer.

I. A new service routine is defined starting in memory location x3700. After loading a program that calls this subroutine, the user sets memory location x0067 to x3700. Which of the following can be used to call this subroutine?
   a. TRAP x3767
   b. TRAP x67
   c. TRAP x3700
   d. TRAP x0037

II. JSR <label> is equivalent to
   a. LEA R7, #0 [that is, 1110 111 000000000]
      BRnzp <label>
   b. LEA R7, #1
      BRnzp <label>
   c. LEA R7, <label>
      JMP R7
   d. All of the above are equivalent

III. JSRR R3 is equivalent to
   a. LEA R3, #0
      JMP R7
   b. LEA R7, #1
      JMP R3
   c. LEA R3, #0
      JMP R3
   d. LEA R3, #1
      JMP R7

IV. Into what phase of the control state machine is the logic to test for an interrupt signal usually added?
   a. just before Decode
   b. just after Decode
   c. just after Evaluate Address
   d. just before Execute
   e. just after Store Result

V. Which of the following can be used only once in the program?
   a. .END
   b. .HALT
   c. .FILL
   d. .BLKW
VI. Assembling the instruction ADD R1, R1, #45 causes which of the following errors?
   a. Immediate value is out of range
   b. ADD instruction takes only 3 register sources (2 sources + 1 destination)
   c. R1 is not initialized
   d. The instruction does not cause an error.

VII. How many memory locations are used by the following assembly directive?:
   MYFAVORITE .STRINGZ "Exam 4"
   a. 3
   b. 4
   c. 6
   d. 7
   e. 8

VIII. Which one of the following is correct about "Caller-save"? (circle the correct answer)
   a) It is better than Callee-save, since the caller knows exactly which registers
      the subroutine will overwrite.
   b) It is used primarily in ISAs that have very wide registers.
   c) It might end up saving registers that didn't really need to be saved.
   d) It's particularly useful when entering interrupt handlers, to save the user's registers.

IX. Comments are:
   a) Unimportant because they are only for people, not the assembler.
   b) Useful, but only if someone else takes over working on your program.
   c) So important that you should have a comment on every line, e.g.
      ADD R0,R0,R1 ; Adds R1 into R0
   d) Important for the author, as well as for reviewers, maintainers, and users of the software.

X. If you execute a RET instruction in memory location x4231, what is the possible range of values
   for the PC afterwards? [PC_{inc} indicates PC after it is incremented.]
   a. x0000 -- xFFFF
   b. (PC_{inc} - 2048) -- (PC_{inc} + 2047)
   c. (PC_{inc} - 256) -- (PC_{inc} + 255)
   d. (PC_{inc} - 16) -- (PC_{inc} + 15)
Scratch Paper
<table>
<thead>
<tr>
<th>Char</th>
<th>Dec</th>
<th>Oct</th>
<th>Hex</th>
<th>Char</th>
<th>Dec</th>
<th>Oct</th>
<th>Hex</th>
<th>Char</th>
<th>Dec</th>
<th>Oct</th>
<th>Hex</th>
</tr>
</thead>
<tbody>
<tr>
<td>(nul)</td>
<td>0</td>
<td>0000</td>
<td>0x00</td>
<td>(sp)</td>
<td>32</td>
<td>0040</td>
<td>0x20</td>
<td>@</td>
<td>64</td>
<td>0100</td>
<td>0x40</td>
</tr>
<tr>
<td>(soh)</td>
<td>1</td>
<td>0001</td>
<td>0x01</td>
<td>!</td>
<td>33</td>
<td>0041</td>
<td>0x21</td>
<td>A</td>
<td>65</td>
<td>0101</td>
<td>0x41</td>
</tr>
<tr>
<td>(stx)</td>
<td>2</td>
<td>0002</td>
<td>0x02</td>
<td>&quot;</td>
<td>34</td>
<td>0042</td>
<td>0x22</td>
<td>B</td>
<td>66</td>
<td>0102</td>
<td>0x42</td>
</tr>
<tr>
<td>(etx)</td>
<td>3</td>
<td>0003</td>
<td>0x03</td>
<td>#</td>
<td>35</td>
<td>0043</td>
<td>0x23</td>
<td>C</td>
<td>67</td>
<td>0103</td>
<td>0x43</td>
</tr>
<tr>
<td>(ecn)</td>
<td>4</td>
<td>0004</td>
<td>0x04</td>
<td>$</td>
<td>36</td>
<td>0044</td>
<td>0x24</td>
<td>D</td>
<td>68</td>
<td>0104</td>
<td>0x44</td>
</tr>
<tr>
<td>(ack)</td>
<td>6</td>
<td>0006</td>
<td>0x06</td>
<td>%</td>
<td>37</td>
<td>0045</td>
<td>0x25</td>
<td>E</td>
<td>69</td>
<td>0105</td>
<td>0x45</td>
</tr>
<tr>
<td>(bel)</td>
<td>7</td>
<td>0007</td>
<td>0x07</td>
<td>^</td>
<td>39</td>
<td>0047</td>
<td>0x27</td>
<td>G</td>
<td>71</td>
<td>0107</td>
<td>0x47</td>
</tr>
<tr>
<td>(bs)</td>
<td>8</td>
<td>0010</td>
<td>0x08</td>
<td>_</td>
<td>40</td>
<td>0050</td>
<td>0x30</td>
<td>H</td>
<td>72</td>
<td>0110</td>
<td>0x48</td>
</tr>
<tr>
<td>(ht)</td>
<td>9</td>
<td>0011</td>
<td>0x09</td>
<td></td>
<td></td>
<td>41</td>
<td>0051</td>
<td>0x31</td>
<td>I</td>
<td>73</td>
<td>0111</td>
</tr>
<tr>
<td>(nl)</td>
<td>10</td>
<td>0012</td>
<td>0x0a</td>
<td>*</td>
<td>42</td>
<td>0052</td>
<td>0x32</td>
<td>J</td>
<td>74</td>
<td>0112</td>
<td>0x4a</td>
</tr>
<tr>
<td>(vt)</td>
<td>11</td>
<td>0013</td>
<td>0x0b</td>
<td>+</td>
<td>43</td>
<td>0053</td>
<td>0x33</td>
<td>K</td>
<td>75</td>
<td>0113</td>
<td>0x4b</td>
</tr>
<tr>
<td>(np)</td>
<td>12</td>
<td>0014</td>
<td>0x0c</td>
<td>,</td>
<td>44</td>
<td>0054</td>
<td>0x34</td>
<td>L</td>
<td>76</td>
<td>0114</td>
<td>0x4c</td>
</tr>
<tr>
<td>(cr)</td>
<td>13</td>
<td>0015</td>
<td>0x0d</td>
<td>-</td>
<td>45</td>
<td>0055</td>
<td>0x35</td>
<td>M</td>
<td>77</td>
<td>0115</td>
<td>0x4d</td>
</tr>
<tr>
<td>(so)</td>
<td>14</td>
<td>0016</td>
<td>0x0e</td>
<td>.</td>
<td>46</td>
<td>0056</td>
<td>0x36</td>
<td>N</td>
<td>78</td>
<td>0116</td>
<td>0x4e</td>
</tr>
<tr>
<td>(si)</td>
<td>15</td>
<td>0017</td>
<td>0x0f</td>
<td>/</td>
<td>47</td>
<td>0057</td>
<td>0x37</td>
<td>O</td>
<td>79</td>
<td>0117</td>
<td>0x4f</td>
</tr>
<tr>
<td>(dle)</td>
<td>16</td>
<td>0020</td>
<td>0x10</td>
<td>0</td>
<td>48</td>
<td>0060</td>
<td>0x38</td>
<td>P</td>
<td>80</td>
<td>0120</td>
<td>0x50</td>
</tr>
<tr>
<td>(dc1)</td>
<td>17</td>
<td>0021</td>
<td>0x11</td>
<td>1</td>
<td>49</td>
<td>0061</td>
<td>0x39</td>
<td>Q</td>
<td>81</td>
<td>0121</td>
<td>0x51</td>
</tr>
<tr>
<td>(dc2)</td>
<td>18</td>
<td>0022</td>
<td>0x12</td>
<td>2</td>
<td>50</td>
<td>0062</td>
<td>0x3a</td>
<td>R</td>
<td>82</td>
<td>0122</td>
<td>0x52</td>
</tr>
<tr>
<td>(dc3)</td>
<td>19</td>
<td>0023</td>
<td>0x13</td>
<td>3</td>
<td>51</td>
<td>0063</td>
<td>0x3b</td>
<td>S</td>
<td>83</td>
<td>0123</td>
<td>0x53</td>
</tr>
<tr>
<td>(dc4)</td>
<td>20</td>
<td>0024</td>
<td>0x14</td>
<td>4</td>
<td>52</td>
<td>0064</td>
<td>0x3c</td>
<td>T</td>
<td>84</td>
<td>0124</td>
<td>0x54</td>
</tr>
<tr>
<td>(nak)</td>
<td>21</td>
<td>0025</td>
<td>0x15</td>
<td>5</td>
<td>53</td>
<td>0065</td>
<td>0x3d</td>
<td>U</td>
<td>85</td>
<td>0125</td>
<td>0x55</td>
</tr>
<tr>
<td>(syn)</td>
<td>22</td>
<td>0026</td>
<td>0x16</td>
<td>6</td>
<td>54</td>
<td>0066</td>
<td>0x3e</td>
<td>V</td>
<td>86</td>
<td>0126</td>
<td>0x56</td>
</tr>
<tr>
<td>(etb)</td>
<td>23</td>
<td>0027</td>
<td>0x17</td>
<td>7</td>
<td>55</td>
<td>0067</td>
<td>0x3f</td>
<td>W</td>
<td>87</td>
<td>0127</td>
<td>0x57</td>
</tr>
<tr>
<td>(can)</td>
<td>24</td>
<td>0030</td>
<td>0x18</td>
<td>8</td>
<td>56</td>
<td>0070</td>
<td>0x40</td>
<td>X</td>
<td>88</td>
<td>0130</td>
<td>0x58</td>
</tr>
<tr>
<td>(em)</td>
<td>25</td>
<td>0031</td>
<td>0x19</td>
<td>9</td>
<td>57</td>
<td>0071</td>
<td>0x41</td>
<td>Y</td>
<td>89</td>
<td>0131</td>
<td>0x59</td>
</tr>
<tr>
<td>(sub)</td>
<td>26</td>
<td>0032</td>
<td>0x1a</td>
<td>:</td>
<td>58</td>
<td>0072</td>
<td>0x42</td>
<td>Z</td>
<td>90</td>
<td>0132</td>
<td>0x5a</td>
</tr>
<tr>
<td>(esc)</td>
<td>27</td>
<td>0033</td>
<td>0x1b</td>
<td>;</td>
<td>59</td>
<td>0073</td>
<td>0x43</td>
<td></td>
<td></td>
<td>91</td>
<td>0133</td>
</tr>
<tr>
<td>(fs)</td>
<td>28</td>
<td>0034</td>
<td>0x1c</td>
<td>&lt;</td>
<td>60</td>
<td>0074</td>
<td>0x44</td>
<td></td>
<td></td>
<td>92</td>
<td>0134</td>
</tr>
<tr>
<td>(gs)</td>
<td>29</td>
<td>0035</td>
<td>0x1d</td>
<td>&gt;</td>
<td>61</td>
<td>0075</td>
<td>0x45</td>
<td></td>
<td></td>
<td>93</td>
<td>0135</td>
</tr>
<tr>
<td>(rs)</td>
<td>30</td>
<td>0036</td>
<td>0x1e</td>
<td>?</td>
<td>62</td>
<td>0076</td>
<td>0x46</td>
<td></td>
<td></td>
<td>94</td>
<td>0136</td>
</tr>
<tr>
<td>(us)</td>
<td>31</td>
<td>0037</td>
<td>0x1f</td>
<td></td>
<td></td>
<td>63</td>
<td>0077</td>
<td>0x47</td>
<td></td>
<td></td>
<td>95</td>
</tr>
</tbody>
</table>

**ASCII Table**