CS/ECE 252: INTRODUCTION TO COMPUTER ENGINEERING

UNIVERSITY OF WISCONSIN—MADISON

Instructor: Rahul Nayar
TAs: Annie Lin, Mohit Verma

Examination 4
In Class (50 minutes)
Wednesday, May 3rd, 2017
Weight: 17.5%

NO BOOK(S), NOTE(S), CALCULATORS OR ELECTRONIC DEVICES OF ANY SORT.
The exam has eleven pages. You must turn in the pages 1-9. Circle your final answers. Plan your time carefully since some problems are longer than others. Use the blank sides of the exam for scratch work.

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>5</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>4</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>32</td>
<td></td>
</tr>
</tbody>
</table>
1. The following LC-3 assembly code contains assembly syntax errors. Identify and fix at least 4 such errors. (4 points)

```
.ORIG x3000
    AND R5, R5, ZERO
    LD R5, STRING
.NEXT  ADD R5, R5, #32
        BRz NEXT
.MAIN  LD R4, MAIN
        SUB R4, R4, #1
        ST R4, STRING
.NEXT  HALT
.ZERO  .FILL #0
        .BLKW 4
        .BLKW 3
.STRING  .STRINGZ "ABC"
        .END
```
2. a) Fill in the symbol table for following LC-3 assembly code. You may not need to fill all rows. (5 points)

```
.ORIG x3000
AND R3, R3, #0
LD R4, VAL1
LOOP BRz DONE
  JSR INC1
  JSR INC2
DONE ST R3, ANS
OUT
HALT
INC1 ADD R3, R3, #1
  RET
INC2 ADD R4, R4, #-1
  RET ; Storage area for variables below:
ANS .BLKW #4
VAL0 .STRINGZ "CS"
VAL1 .STRINGZ "252"
.END
```

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>Value (in hex)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

b) Convert the instruction stored at memory location 0x3006 into binary.
3. An LC-3 assembly program is given below: (5 points)

```
.ORIG x3000
LD R0, DATA
ADD R0, R0, #10
PRINT1 OUT
  AND R0, R0, #0
  ADD R0, R7, R0
PRINT2 OUT
  HALT
DATA .FILL 0xFFFF
```

a. What is the output (in hex) after the OUT statement at the symbol PRINT1 finishes execution?

b. What is the output (in hex) after the OUT statement at the symbol PRINT2 finishes execution? Explain your answer.

c. Complete the following incomplete code snippet that uses the memory-mapped LC-3 registers KBSR and KBDR to take input from the keyboard instead of the GETC instruction. Your code should store the value entered from keyboard in register R1. (Assume KBSR is mapped to address xFE00 and KBDR is mapped to address xFE02.)

```
ECHO: ______,___, KBSR
      ______
      _____, R1, KBDR
```

KBSR .FILL xFE00
KBDR .FILL xFE02
4. Short answer questions (5 points)

a. Briefly describe what happens in the **linking** phase of an assembly program.

b. How are the Display Data Register (DDR) and Display Status Register (DSR) used when TRAP x21 (OUT) is called?

c. What is a **service routine** in LC-3? Give an example.

d. **Briefly** describe the difference between synchronous and asynchronous I/O.
5. The following LC-3 assembly code implements a subroutine. After taking input, it outputs a lowercase letter (a-z) if the input was uppercase (A-Z), and a “N” otherwise. (5 points)

SUBR GETC

ADD R5, R0, 0
NOT R5, R5
ADD R5, R5, 1

LD R1, LBOUND
LD R2, UBOUND
ADD R3, R5, R2
BRn PNOT
ADD R3, R5, R1
BRp PNOT

LD R3, DIFF
ADD R0, R0, R3
OUT
BRnzp FINISH

PNOT LD R0, N
OUT
FINISH RET

UBOUND .FILL x5A ; ASCII value for "Z"
LBOUND .FILL x41 ; ASCII value for "A"
N .FILL x4E ; ASCII value for "N"
DIFF .FILL x20

a. Is this code able to successfully return? Explain why or why not.

b. Add lines to the code above to make this subroutine callee-saved. You should not modify any existing lines. Clearly indicate which lines you have added and where.
6. Examine the code below. You may assume that at the start of the program, all registers are set to 0. (4 points)

```
.ORIG x3000
GETC
AND R3, R3, 0
ADD R3, R3, R0

LD R1, ONE
NOT R1, R1
ADD R1, R1, 1

PRINT LD R0, C
OUT
LD R0, A
OUT
LD R0, T
OUT

ADD R0, R3, R1
BRz FINISH

LD R0, S
OUT

FINISH HALT
C .FILL x63 ; ASCII 'c'
A .FILL x61 ; ASCII 'a'
T .FILL x74 ; ASCII 't'
S .FILL x73 ; ASCII 's'
ONE .FILL x31 ; ASCII '1'
```

a. If the input is the decimal number "1", what is output on the screen at the end of the program?

b. OUT only prints out one character at a time. Instead of printing individually, we decide to replace the code stored at memory locations x3006 to x300B with the following, much shorter block of code:

```
PRINT LD R0, C
PUTS
```

Will this output the same result as a)? Why or why not? You must explain your answer for credit.
7. Multiple choice questions. Circle **one** answer for each question. (4 points)

(i) Which of the following can **not** be used multiple times in a single assembly program?
   a) .FILL
   b) .ORIG
   c) .BLKW
   d) .STRINGZ

(ii) Assume that a LC-3 processor receives interrupts from 3 I/O devices (A, B and C) simultaneously. The priority levels for the interrupts are given below:
   A) PL4  B) PL2  C) PL6
   Assuming that no other interrupts come in, which of the above interrupts is serviced **last**?
   a) A
   b) B
   c) C
   d) Any selected at random

(iii) Our program begins at memory location x4000. We want to load the value x4020 into R3. Which LC-3 instruction can we use to accomplish this in a **single** line?
   a) LEA
   b) LD
   c) ST
   d) LDI

(iv) Which register is used to store **input data** after IN is called?
   a) R1
   b) R0
   c) R7
   d) R4
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.

### Instruction Set

<table>
<thead>
<tr>
<th>Opcode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00001</td>
<td>ADD DR, SR1, SR2; Addition</td>
</tr>
<tr>
<td>00001</td>
<td>DR[] SR1 + SR2 also setcc()</td>
</tr>
<tr>
<td>00001</td>
<td>ADD DR, SR1, imm5; Addition with Immediate</td>
</tr>
<tr>
<td>00001</td>
<td>DR[] SR1 + SEXT(imm5) also setcc()</td>
</tr>
<tr>
<td>0101</td>
<td>AND DR, SR1, SR2; Bit-wise AND</td>
</tr>
<tr>
<td>01001</td>
<td>DR[] SR1 AND SR2 also setcc()</td>
</tr>
<tr>
<td>01001</td>
<td>AND DR, SR1, imm5; Bit-wise AND with Immediate</td>
</tr>
<tr>
<td>01001</td>
<td>DR[] SR1 AND SEXT(imm5) also setcc()</td>
</tr>
<tr>
<td>00000</td>
<td>BRx, label (where x=n,z,p,np,nz,nzp); Branch</td>
</tr>
<tr>
<td>11000</td>
<td>mem[mem[PC' + SEXT(PCoffset9)]] also setcc()</td>
</tr>
<tr>
<td>01001</td>
<td>DR[] R7 also setcc()</td>
</tr>
<tr>
<td>00000</td>
<td>Mem[mem[PC' + SEXT(PCoffset9)]] also setcc()</td>
</tr>
<tr>
<td>01001</td>
<td>DR[] PC' + SEXT(PCoffset9) also setcc()</td>
</tr>
<tr>
<td>00000</td>
<td>Mem[mem[PC' + SEXT(PCoffset9)]] also setcc()</td>
</tr>
<tr>
<td>01001</td>
<td>DR[] PC' + SEXT(PCoffset9) also setcc()</td>
</tr>
<tr>
<td>01001</td>
<td>DR[] NOT(SR) also setcc()</td>
</tr>
<tr>
<td>01001</td>
<td>SR1 + SEXT(imm5) also setcc()</td>
</tr>
<tr>
<td>01001</td>
<td>SR1 + SR2 also setcc()</td>
</tr>
<tr>
<td>00000</td>
<td>See textbook (2nd Ed. page 537).</td>
</tr>
<tr>
<td>00000</td>
<td>ST SR, label; Store PC-Relative</td>
</tr>
<tr>
<td>00000</td>
<td>mem[PC' + SEXT(PCoffset9)][] SR</td>
</tr>
<tr>
<td>01001</td>
<td>STI, SR, label; Store Indirect</td>
</tr>
<tr>
<td>01001</td>
<td>mem[mem[PC' + SEXT(PCoffset9)]][] SR</td>
</tr>
<tr>
<td>01001</td>
<td>STR SR, BaseR, offset6; Store Base+Offset</td>
</tr>
<tr>
<td>01001</td>
<td>mem[BaseR + SEXT(offset6)][] SR</td>
</tr>
<tr>
<td>1111</td>
<td>TRAP; System Call</td>
</tr>
<tr>
<td>00000</td>
<td>R7[] PC', PC[] mem[ZEXT(trapvect8)]</td>
</tr>
<tr>
<td>1101</td>
<td>Initiate illegal opcode exception</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Opcode</th>
<th>Operand</th>
</tr>
</thead>
<tbody>
<tr>
<td>.ORIG</td>
<td>address</td>
</tr>
<tr>
<td>.END</td>
<td></td>
</tr>
<tr>
<td>.BLKW</td>
<td>n</td>
</tr>
<tr>
<td>.FILL</td>
<td>n</td>
</tr>
<tr>
<td>.STRINGZ</td>
<td>n-character string</td>
</tr>
</tbody>
</table>

### Trap Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Equivalent</th>
</tr>
</thead>
<tbody>
<tr>
<td>HALT</td>
<td>TRAP x25</td>
</tr>
<tr>
<td>IN</td>
<td>TRAP x23</td>
</tr>
<tr>
<td>OUT</td>
<td>TRAP x21</td>
</tr>
<tr>
<td>GETC</td>
<td>TRAP x20</td>
</tr>
<tr>
<td>PUTC</td>
<td>TRAP x22</td>
</tr>
</tbody>
</table>