

**CS/ECE 252: INTRODUCTION TO COMPUTER ENGINEERING**

**UNIVERSITY OF WISCONSIN—MADISON**

Instructor: Rahul Nayar  
TAs: Mohit Verma, Annie Lin

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#: \_\_\_\_\_

<b>Problem</b>	<b>Maximum Points</b>	<b>Points Earned</b>
1	4	
2	5	
3	5	
4	5	
5	5	
6	4	
7	4	
Total	32	

1. The following LC-3 assembly code contains assembly syntax errors. Identify and fix at least 4 errors. (4 points)

```
.ORIG x3000
    ADD R3, R3, ONE
    LD R5, STRING
LOOP  ADD R1, R1, #32
      BRz LOOP
MAIN  LD R4, MAIN
      SUB R3, R3, #1
      ST R3, STRING
LOOP  HALT
ONE   .FILL #1
      .BLKW 4
      .BLKW 3
STRING .STRINGZ "ABC"
      .END
```

```
.ORIG x3000
    ADD R3, R3, ONE
    LD R5, STRING
LOOP ADD R1, R1, #32
      BRz LOOP

MAIN LD R4, MAIN
      SUB R3, R3, #1
      ST R3, STRING
LOOP HALT
ONE  .FILL #1
      .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 EXIT
        JSR INC
        JSR DEC
EXIT    ST R3, ANS
        PUTS
        HALT
INC     ADD R3, R3, #1
        RET
DEC     ADD R4, R4, #-1
        RET ; Storage area for variables below:
ANS     .BLKW #5
VAL0    .STRINGZ "252"
VAL1    .STRINGZ "CS"
        .END

```

SYMBOL	Value (in hex)
LOOP	0x3002
EXIT	0x3005
INC	0x3008
DEC	0x300A
ANS	0x300C
VAL0	0x3011
VAL1	0x3015

b) Convert the instruction stored at memory location 0x3006 into binary.

0xF022 (TRAP x22)

3. An LC-3 assembly code is given below: (5 points)

```
        .ORIG x3000
        AND R0, R0, #0
        LD R0, DATA
        ADD R0, R0, #11
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?

0x000A

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

0x3004

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 R3.

(Assume KBSR is mapped to address xFE00 and KBDR is mapped to address xFE02.)

```
ECHO:  _____, ____, KBSR
        _____
        _____, R3, KBDR
```

```
KBSR .FILL xFE00
KBDR .FILL xFE02
```

ECHO: LDI, R0, KBSR  
      BRnp ECHO  
      LDI, R2, 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.

Linking is resolving symbols between independent object files.

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

When the monitor is ready to display another character, DSR[15] is set to 1. DSR[7:0] is displayed and DSR[15] is set to 1. Any other data written to DDR is ignored while DSR[15] is zero.

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

A service routine is a function that performs a specific operation (optionally as part of the operating system). In LC-3, traps are service routines. Any trap will work here as an example.

d. **Briefly** describe the difference between synchronous and asynchronous I/O.

Synchronous I/O events occur at fixed, predictable rates. CPU reads every X seconds. Asynchronous I/O is unpredictable. Can use flag to achieve I/O. Example is keyboard input.

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 of "Z"
LBOUND .FILL x41 ;ASCII value of "A"
N      .FILL x4E ;ASCII value of "N"
DIFF   .FILL x20
```

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

No. R7 is overwritten by GETC and OUT.

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.**

Before GETC:  
ST R0, SAVE0  
ST R1, SAVE1  
ST R2, SAVE2

```
ST R3, SAVE3
ST R5, SAVE5
ST R7, SAVE7
```

Before FINISH:

```
LD R0, SAVE0
LD R1, SAVE1
LD R2, SAVE2
LD R3, SAVE3
LD R5, SAVE5
LD R7, SAVE7
```

After DIFF

```
SAVE0 .FILL x0
SAVE1 .FILL x0
SAVE2 .FILL x0
SAVE3 .FILL x0
SAVE5 .FILL x0
SAVE7 .FILL x0
```



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 "3", what is output on the screen at the end of the program?

cats

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.**

No. It'll output a bunch of garbage (whatever is stored at memory location x63 until it hits a null terminator). This is because PUTS uses the register value as an address, not as a value.

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) .BLKW
- b) .FILL
- c) **.ORIG**
- 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) PL0 B) PL7 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) LD
- b) **LEA**
- c) LDI
- d) ST

(iv) Which register is used to store **input data** after IN is called?

- a) R7
- b) R4
- c) R1
- d) **R0**



## Assembler Directives

<i>Opcode</i>	<i>Operand</i>
<b>.ORIG</b>	<b>address</b>
<b>.END</b>	
<b>.BLKW</b>	<b>n</b>
<b>.FILL</b>	<b>n</b>
<b>.STRINGZ</b>	<b>n-character string</b>

## Trap Codes

<i>Code</i>	<i>Equivalent</i>
<b>HALT</b>	<b>TRAP x25</b>
<b>IN</b>	<b>TRAP x23</b>
<b>OUT</b>	<b>TRAP x21</b>
<b>GETC</b>	<b>TRAP x20</b>
<b>PUTS</b>	<b>TRAP x22</b>