CS/ECE 252: INTRODUCTION TO COMPUTER ENGINEERING
COMPUTER SCIENCES DEPARTMENT
UNIVERSITY OF WISCONSIN-MADISON

Prof. Mark D. Hill & Prof. Parmesh Ramanathan
TAs Kasturi Bidarkar, Ryan Johnson, Jie Liu, & Ramachandran Syamkumar

Midterm Examination 4
In Class (50 minutes)
Friday, December 11, 2009
Weight: 15%

CLOSED BOOK, NOTE, CALCULATOR, PHONE, & COMPUTER.

The exam in two-sided and has **THIRTEEN** pages, including one blank page and a copy of the ***Standard ASCII Table***, some ***Trap Service Routines*** description and the ***LC-3 Instruction Set handout*** on the final page (please feel free to detach this final page).

Plan your time carefully, since some problems are longer than others.

NAME: ____

SECTION: ____

ID#: ____

Problem Number	Maximum Points	Points Awarded
1	6	
2	8	
3	2	
4	7	
5	4	
6	3	
Total	30	

Problem 1: Assembly Language

A) (2 Point) Circle which (if any) of the following pseudo-ops can be used multiple times in a single assembly file:

.ORIG

.FILL

.BLKW

.STRINGZ

.END

B) (1 Point) How many memory locations are used by the following assembly directive:

.STRINGZ “Football”

9

C) (3 Points) Briefly explain the three assembly errors in the following program:

```
.ORIG x3000
INIT AND R1, R1, #0
     ADD R1, R1, #32
ADD ADD R1, R1, #-1
     BRnz ADD
     HALT
INIT .BLKW 1
.END
```

1) ADD used as label
2) INIT used twice
3) 32 requires more than the 5 bit immediate field

Problem 2: Two-Pass Assembly Process

An assembly language LC-3 program is given below:

```
;
; Program to multiply a number by the constant 6
;
.ORIG x3010
     LD R1, SIX
     LD R2, NUMBER
     AND R3, R3, #0 ; Clear R3. It will contain the product.

; The inner loop
AGAIN ADD R3, R3, R2
     ADD R1, R1, #-1  ; R1 keeps track of the iteration.
     BRp AGAIN
     HALT

NUMBER .BLKW 1
SIX .FILL x0006
.END
```

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

Symbol Name	Address
SIX	x3018
NUMBER	x3017
AGAIN	x3013

The binary version of the above assembly file created after the second pass of the assembler is given below (with 4 lines missing):

a. ________________

b. ________________

```
0010 0100 0000 0101 ;LD R2, NUMBER
0101 0110 1110 0000 ;AND R3, R3, #0
0001 0110 1100 0010 ;ADD R3, R3, R2
0001 0010 0111 1111 ;ADD R1, R1, #-1
0000 0011 1111 1101 ;BRp AGAIN
```

c. ________________

```
0000 0000 0000 0000 ;NUMBER .BLKW 1
```

d. ________________

B) (4 points) Circle the correct values to fill in for the missing lines:

a:

1) 0011 0000 0001 0000
2) 0010 0010 0000 1000
3) 0011 0000 0101 0000

b:

1) 0011 0000 0000 0000
2) 0010 0010 0000 0111
3) 0010 0010 0000 1000

c:

1) 1101 0000 0000 0110
2) 1100 0001 1100 0000
3) 1111 0000 0010 0101

d:

1) 0000 0000 0000 0110
2) 0000 0000 0000 0000
3) 1111 1111 1111 1111

Problem 3: Trap Routines

(2 points) How many TRAP service routines can be defined on the LC-3? Why?

256, because TRAP = 11110000 Trapvect8, 2^8 = 256

Problem 4: I/O

Suppose we modify the LC-3 ISA to use the opcode `1101` for I/O operations. The assembly instruction is written as: `IO <Command>`. We define the following commands:

Assembly	Binary	Function
`IO 0`	`1101 0000 0000 0000`	Check KBSR, set flags to N if ready, Z if not ready
`IO 1`	`1101 0000 0000 0001`	Load R0 with the contents of KBDR
`IO 2`	`1101 0000 0000 0010`	Check DSR, set flags to N if ready, Z if not ready
`IO 3`	`1101 0000 0000 0011`	Load DDR with the contents of R0

Remember:

KBSR - Bit[15] is 1 if keyboard has data to be read, else 0

KBDR - Contains keyboard data to be read

DSR - Bit[15] is 1 if monitor is ready to accept data for output

DDR - Register that accepts data for output to the monitor

A) (1 point) Circle the correct combination that describes this modified LC-3 system:

a. Memory mapped and interrupt driven
b. Special opcode for I/O and interrupt driven
c. Memory mapped and polling
d. Special opcode for I/O and polling

The following is an LC-3 assembly program which uses the modified ISA:

```
.ORIG x3000
      LD R0, ASCII
      LD R1, NEG
LOOP IO 2
      BRzp LOOP
      IO 3
      ADD R0, R0, #-1
      ADD R3, R0, R1
      BRp LOOP
      HALT
ASCII .FILL x0047
NEG   .FILL xFFBD
.END
```

B) (3 points) What is the output of the program on the LC-3 display?

GFED

C) (3 points) Briefly explain why is it important to call `IO 0` ***each time before calling*** `IO 1`***:***

Need to ensure a valid character exists in they keyboard data register or IO 1 may return corrupted data.

Problem 5: Subroutines

Consider the following LC-3 assembly program (assume the SAVERX labels are correctly defined elsewhere):

```
.ORIG x3000
      ST R0, SAVER0
      ST R1, SAVER1
      JSR SUB1
      LD R0, SAVER0
      LD R1, SAVER1
      HALT
SUB1 ST R7, SAVER7
      ST R3, SAVER3
      ST R2, SAVER2
      JSR SUB2
CHECK1 AND R1, R1, #0
      LD R7, SAVER7
      LD R3, SAVER3
      LD R2, SAVER2
      RET
SUB2 AND R0, R0, #0
CHECK2 RET
```

A) (1 point) What is the value of R7 at CHECK1?

x300A

B) (1 point) What is the value of R7 at CHECK2?

x300A

C) (1 point) Which registers are caller saved?

R0, R1

D) (1 point) Which registers are callee saved?

R2, R3, R7

Problem 6: Halting Problems

(1 point each) Mark statements as True or False.

T / F : Given enough time, it is possible to write a program that determines whether any possible program halts (does not loop forever) on a specific instance of the program's inputs.

T / F : Given enough time, it is possible to write a program that determines whether any possible program halts (does not loop forever) on all of the program's possible inputs.

T / F : Given enough time, it is possible to write a program to solve any problem that one can precisely specify.

Scratch Sheet (in case you need additional space for some of your answers)