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# _____

Question	Maximum Points	Points
1	4	
2	7	
3	7	
4	7	
5	10	
Total	35	

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).

```
            .ORIG x3000

Main        ...
            ...
Loop        ...
            ...
End         ...
            ...

Add         ...
            ...

Sum         ...
            ...
Loop        ...
            ...

And         ...
            ...

Or          ...
            ...

X10         ...
            ...
            .END
```

2. Two-Pass Assembly Process (7 points)

An assembly language LC-3 program is given below:

```
            .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**)

Symbol Name	**Address**

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)**

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

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

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

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.

```
1       ;CODE TO INPUT AND PRINT 6 CHARACTERS
2
3                       .ORIG   x3000
4                       AND     R0, R0, #0              ; Initialize R0, our counter
5       LOOP
6                       LEA     R1, INPSTRING           ; R1 now has base of INPSTRING
7                       ADD     R1, R1, R0              ; R1 now has base + offset = R0
8                       ST      R0, SAVEREG1            ; SAVE R0
9                       JSR     ONECHAR                 ; Call Subroutine
10                      LD      R0, SAVEREG1            ; Restore R0
11                      ADD     R0, R0, #1              ; Increment R0
12                      LD      R1, LENGTH              ; Load R1 with minus length
13                      ADD     R1, R1, R0              ;
14                      BRp     LOOP                    ; loop till 6 characters are reached
15                      LEA     R0, INPSTRING           ; Get ready to print
16                      PUTS                            ; TRAP x22: Print string
17                      HALT                            ; We're done
18
19      ONECHAR         ST      ____, SAVEREG2          ; SAVE ?? upon entering the subroutine
20                      GETC                            ; TRAP x20: Get a character from keyboard.
21                      STR     R0, R1, #0              ; Save keyboard input (R0 contains input)

22                      LD      ____, SAVEREG2          ; Restore ?? before leaving
23                      RET
24
25      LENGTH          .FILL   0xFFFA                  ; minus Length (-6)
26      KBSR            .FILL   0xFE00
27      KBDR            .FILL   0xFE02
28      SAVEREG1        .FILL   0x0
29      SAVEREG2        .FILL   0x0
30      INPSTRING       .BLKW   7
31                      .END
```

(a) Line 8 saves R0 before calling the subroutine ONECHAR. Briefly explain why this is necessary. **(2 points)**

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

(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)**

(d) How many times will each character appear on the screen? Why? **(2 points)**

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

ASCII Table

Char	Dec	Oct	Hex	Char	Dec	Oct	Hex	Char	Dec	Oct	Hex	Char	Dec	Oct	Hex
(nul)	0	0000	0x00	(sp)	32	0040	0x20	@	64	0100	0x40	`	96	0140	0x60
(soh)	1	0001	0x01	!	33	0041	0x21	A	65	0101	0x41	a	97	0141	0x61
(stx)	2	0002	0x02	"	34	0042	0x22	B	66	0102	0x42	b	98	0142	0x62
(etx)	3	0003	0x03	#	35	0043	0x23	C	67	0103	0x43	c	99	0143	0x63
(eot)	4	0004	0x04	$	36	0044	0x24	D	68	0104	0x44	d	100	0144	0x64
(enq)	5	0005	0x05	%	37	0045	0x25	E	69	0105	0x45	e	101	0145	0x65
(ack)	6	0006	0x06	&	38	0046	0x26	F	70	0106	0x46	f	102	0146	0x66
(bel)	7	0007	0x07	'	39	0047	0x27	G	71	0107	0x47	g	103	0147	0x67
(bs)	8	0010	0x08	(	40	0050	0x28	H	72	0110	0x48	h	104	0150	0x68
(ht)	9	0011	0x09	)	41	0051	0x29	I	73	0111	0x49	i	105	0151	0x69
(nl)	10	0012	0x0a	*	42	0052	0x2a	J	74	0112	0x4a	j	106	0152	0x6a
(vt)	11	0013	0x0b	+	43	0053	0x2b	K	75	0113	0x4b	k	107	0153	0x6b
(np)	12	0014	0x0c	,	44	0054	0x2c	L	76	0114	0x4c	l	108	0154	0x6c
(cr)	13	0015	0x0d	-	45	0055	0x2d	M	77	0115	0x4d	m	109	0155	0x6d
(so)	14	0016	0x0e	.	46	0056	0x2e	N	78	0116	0x4e	n	110	0156	0x6e
(si)	15	0017	0x0f	/	47	0057	0x2f	O	79	0117	0x4f	o	111	0157	0x6f
(dle)	16	0020	0x10	0	48	0060	0x30	P	80	0120	0x50	p	112	0160	0x70
(dc1)	17	0021	0x11	1	49	0061	0x31	Q	81	0121	0x51	q	113	0161	0x71
(dc2)	18	0022	0x12	2	50	0062	0x32	R	82	0122	0x52	r	114	0162	0x72
(dc3)	19	0023	0x13	3	51	0063	0x33	S	83	0123	0x53	s	115	0163	0x73
(dc4)	20	0024	0x14	4	52	0064	0x34	T	84	0124	0x54	t	116	0164	0x74
(nak)	21	0025	0x15	5	53	0065	0x35	U	85	0125	0x55	u	117	0165	0x75
(syn)	22	0026	0x16	6	54	0066	0x36	V	86	0126	0x56	v	118	0166	0x76
(etb)	23	0027	0x17	7	55	0067	0x37	W	87	0127	0x57	w	119	0167	0x77
(can)	24	0030	0x18	8	56	0070	0x38	X	88	0130	0x58	x	120	0170	0x78
(em)	25	0031	0x19	9	57	0071	0x39	Y	89	0131	0x59	y	121	0171	0x79
(sub)	26	0032	0x1a	:	58	0072	0x3a	Z	90	0132	0x5a	z	122	0172	0x7a
(esc)	27	0033	0x1b	;	59	0073	0x3b	[	91	0133	0x5b	{	123	0173	0x7b
(fs)	28	0034	0x1c	<	60	0074	0x3c	\	92	0134	0x5c	|	124	0174	0x7c
(gs)	29	0035	0x1d	=	61	0075	0x3d	]	93	0135	0x5d	}	125	0175	0x7d
(rs)	30	0036	0x1e	>	62	0076	0x3e	^	94	0136	0x5e	~	126	0176	0x7e
(us)	31	0037	0x1f	?	63	0077	0x3f	_	95	0137	0x5f	(del)	127	0177	0x7f