

CS/ECE 252: INTRODUCTION TO COMPUTER ENGINEERING

UNIVERSITY OF WISCONSIN—MADISON

Prof. Karthikeyan Sankaralingam, Pradip Vallathol

TAs: Deepika Muthukumar, Sujith Surendran, Murali Sivalingam, Lisa Ossian

Midterm Examination 4

In Class (50 minutes)

Wednesday, May 09, 2014

Weight: 17.5%

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

The exam has **nine** pages. You **must turn in the pages 1-7**. Use the blank sides of the exam for scratch work.

Circle your final answers. Plan your time carefully since some problems are longer than others.

Note:

- The Instruction set is provided on Page 9
- TRAP codes and Assembler directives are provided on Page 10

LAST NAME: _____

FIRST NAME: _____

ID# _____

Problem	Maximum Points	Points Earned
1		
2		
3		
4		
5		
Total	30	

Problem 1: Short answer type questions (7 Points)

a) (1 Point) Mention what problem could come up if DSR is not checked before writing into DDR.

You could overwrite the previous data even before it is displayed.

b) (2 Points) What is the difference between polling and interrupt based I/O ? Briefly explain a scenario where you would prefer interrupt based I/O over polling based I/O?

In Polling based I/O, the program polls checks continuously for the status of the data. In an interrupt based I/O, an interrupt will be generated to signal the program of the completion of work.

If the I/O device takes a lot of time to execute the command, then polling consumes a lot of cycles. In these cases, interrupt driven I/O is preferred

c) (2 Points) Briefly mention what happens during linking and loading phases of a program.

During the Linking phase, the symbols between different object files which are linked together gets resolved.

During the loading phase, the executable image is copied onto the memory

d) (2 Points) What will be the value in R2 if you execute the following program (ie, when you reach the HALT instruction) ?

```
.ORIG x3000
AND R0, R0, #0
ADD R0, R0, #6
STI R0, DATA1
LD R2, DATA2
HALT
```

```
DATA1 .FILL x3006
DATA2 .FILL x3
```

Answer: 6

Problem 2: Assembly Errors (2 Points)

Identify 2 assembly errors in the following program:

```
                .ORIG x3001
                AND R5, R5, ZERO
                LD R5, FOOBAR
NEXT           ADD R5, R5, #1
                BRz NEXT
                LDR R4, R2, #0
                ST R4, FOOBAR
NEXT           HALT

ZERO          .FILL      #0
FOOBAR       .STRINGZ  "YAY!! LAST EXAM"
                .END
```

- a. double declaration of NEXT
- b. AND with ZERO (address)

Problem 3: Traps and Subroutines (6 Points)

Suppose we want to write a new TRAP subroutine, **TRAP x01**. This subroutine takes two inputs from the caller of the subroutine through registers R2 and R1. R2 has the memory address of the first character of a string and R1 has the number of characters to be printed. The subroutine then prints R1 number of characters from the starting of the string (whose address is located in R2). Fill in the missing blanks to complete this subroutine code.

Assume that we are implementing callee save subroutine. Save only those registers that are needed.

Assume the trap vector table (also known as system control block) is as shown below:

<u>Address</u>	<u>Value</u>
x0000	x5000
x0001	x6000
x0002	x7000

```
.ORIG   x6000  

    ST   R0  ,  SAVEREG_LOCATION1
    ST   R1  ,  SAVEREG_LOCATION2
    ST   R2  ,  SAVEREG_LOCATION3
    ST   R7  ,  SAVEREG_LOCATION4

LOOP  LDR R0, R2, #0           ; Load the character to be printed
      OUT                               ; Print the extracted character
      ADD R2, R2, #1           ; Point R2 to the next character
      ADD R1, R1, #-1          ; Set the condition flags if this is the last character
      BRp LOOP                 ; If this is not the last character, branch to LOOP

      LD   R0  ,  SAVEREG_LOCATION1
      LD   R1  ,  SAVEREG_LOCATION2
      LD   R2  ,  SAVEREG_LOCATION3
      LD   R7  ,  SAVEREG_LOCATION4

      RET

SAVEREG_LOCATION1 .BLKW 1
SAVEREG_LOCATION2 .BLKW 1
SAVEREG_LOCATION3 .BLKW 1
SAVEREG_LOCATION4 .BLKW 1
```

Problem 4: I/O (4 Points)

For the assembly program, assume all the registers (R0 - R7) are initialized to the value of zero.

```

        .ORIG x3000
        LD  R1, NEG
        LD  R2, SET
ILOOP   LDI  R3, KBSR
        BRzp ILOOP
        LDI  R4, KBDR
        STI  R2, KBSR
OLOOP   LDI  R2, DSR
        BRzp OLOOP
        ADD R0, R4, #0
        STI  R0, DDR
        ADD R5, R5, 1
        ADD R6, R5, R1
        BRnp OLOOP
        HALT

NEG     .FILL xFFFB
SET     .FILL x4000
DSR     .FILL xFE04 ; Address of DSR
DDR     .FILL xFE06 ; Address of DDR
KBSR    .FILL xFE00 ; Address of KBSR
KBDR    .FILL xFE02 ; Address of KBDR
        .END
```

(a) (2 Points) What does the above LC-3 program do?

- Enables Keyboard interrupt
- Displays the ASCII character that the user entered 5 times back to the user.

(b) (2 Points) How is the operation of the keyboard affected by the instruction **STI R2, KBSR**?

- No impact to the program logic. It just enables the Keyboard interrupt . If we are polling for next character, there can be a error window where the same old data is used even if new character is entered.

Problem 5: Two Stage Assembly Process (10 Points)

Consider the following assembly program.

```
        .ORIG x3000
        LD R0, DATA
        LEA R1, ZERO
        STR R0, R1, #-3
        LEA R5, STR1

LOOP    LDR R0, R5, 0
        BRz END
        OUT
        ADD R5, R5, #1
        BR LOOP

END     HALT

ZERO    .FILL    #0
STR1    .STRINGZ "Wierd_Question"
ARRAY   .BLKW   x5
DATA    .FILL   x1B62
```

(a) (3 Points) In the first pass, the assembler creates symbol table. Fill in the symbol table created by the assembler for this program (in Problem 2(a))

Label	Address
LOOP	3004
END	3009
ZERO	300A
STR1	300B
ARRAY	301A
DATA	301F

(b) (2 Points) In the second pass, the assembler creates a binary (.obj) version of the program, using the entries from the symbol table. Write the binary code generated for the first two instructions (LEA and LD) in the table below.

Assembly code	Binary Code
LD R0, DATA	0010 000 000011110
LEA R1, ZERO	1110 001 000001000

(c) (3 Points) Complete the missing comments for this program (For loop, consider only first iteration):

```

.ORIG x3000
LD R0, DATA      ; Value loaded into R0 is x1B62
LEA R1, ZERO      ; Loads address of ZERO into R1
STR R0,R1,#-3     ; Stores value x1B62 into address x3007
LEA R5, STR1      ; Loads address of STR1 into R5

LOOP   LDR R0, R5, 0 ; Value at R0 in the first iteration is x57
      BRz END       ; Branch to END if ZERO flag is set
      OUT           ; Print the character at R0
      ADD R5, R5, #1 ;
      BR LOOP       ; Branch to LOOP

END    HALT         ; HALT

ZERO   .FILL      #0
STR1   .STRINGZ  "Wierd_Question"
ARRAY  .BLKW     x5
DATA   .FILL     x1B62

```

(d) (3 Points) What will be printed on the console if this program is run on PennSim?

WedQeto

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. SEXT(immediate): zero-extend immediate to 16 bits.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
0	0	0	0	1	DR		SR1	0	0	0	SR2		ADD DR, SR1, SR2 ; Addition				
DR ← SR1 + SR2 also setcc()																	
0	0	0	1	DR		SR1	1		imm5							ADD DR, SR1, imm5 ; Addition with Immediate	
DR ← SR1 + SEXT(imm5) also setcc()																	
0	1	0	1	DR		SR1	0	0	0	SR2		AND DR, SR1, SR2 ; Bit-wise AND					
DR ← SR1 AND SR2 also setcc()																	
0	1	0	1	DR		SR1	1		imm5							AND DR, SR1, imm5 ; Bit-wise AND with Immediate	
DR ← SR1 AND SEXT(imm5) also setcc()																	
0	0	0	0	n	z	p	PCoffset9						BRx, label (where x={n,s,p,sp,np,ns,ns,p}); Branch				
GO ← ((n AND N) OR (z AND Z) OR (p AND P))																	
if(GO is true) then PC ← PC' + SEXT(PCoffset9)																	
1	1	0	0	0	0	0	0	BaseR	0	0	0	0	0	0	0		
JMP BaseR ; Jump																	
PC ← BaseR																	
0	1	0	0	1		PCoffset11						JSR label ; Jump to Subroutine					
R7 ← PC', PC ← PC' + SEXT(PCoffset11)																	
0	1	0	0	0	0	0	0	BaseR	0	0	0	0	0	0	0		
JSRR BaseR ; Jump to Subroutine in Register																	
temp ← PC', PC ← BaseR, R7 ← temp																	
0	0	1	0	DR		PCoffset9						LD DR, label ; Load PC-Relative					
DR ← mem[PC' + SEXT(PCoffset9)] also setcc()																	
1	0	1	0	DR		PCoffset9						LDI DR, label ; Load Indirect					
DR ← mem[mem[PC' + SEXT(PCoffset9)]] also setcc()																	
0	1	1	0	DR		BaseR		offset6							LDR DR, BaseR, offset6 ; Load Base+Offset		
DR ← mem[BaseR + SEXT(offset6)] also setcc()																	
1	1	1	0	DR		PCoffset9						LEA, DR, label ; Load Effective Address					
DR ← PC' + SEXT(PCoffset9) also setcc()																	
1	0	0	1	DR		SR	1	1	1	1	1	1	1	1	1		
NOT DR, SR ; Bit-wise Complement																	
DR ← NOT(SR) also setcc()																	
1	1	0	0	0	0	0	0	1	1	1	0	0	0	0	0		
RET ; Return from Subroutine																	
PC ← R7																	
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
RTI ; Return from Interrupt																	
See textbook (2 nd Ed. page 537).																	
0	0	1	1	SR		PCoffset9						ST SR, label ; Store PC-Relative					
mem[PC' + SEXT(PCoffset9)] ← SR																	
1	0	1	1	SR		PCoffset9						STI, SR, label ; Store Indirect					
mem[mem[PC' + SEXT(PCoffset9)]] ← SR																	
0	1	1	1	SR		BaseR		offset6							STR SR, BaseR, offset6 ; Store Base+Offset		
mem[BaseR + SEXT(offset6)] ← SR																	
1	1	1	1	0	0	0	0	trapvect8						TRAP ; System Call			
R7 ← PC', PC ← mem[SEXT(trapvect8)]																	
1	1	0	1												; Unused Opcode		
Initiate illegal opcode exception																	

TRAP CODES

Code	Equivalent	Description
HALT	TRAP x25	Halt execution and print message to console.
IN	TRAP x23	Print prompt on console, read (and echo) one character from keybd. Character stored in R0[7:0].
OUT	TRAP x21	Write one character (in R0[7:0]) to console.
GETC	TRAP x20	Read one character from keyboard. Character stored in R0[7:0].
PUTS	TRAP x22	Write null-terminated string to console. Address of string is in R0.

ASSEMBLER DIRECTIVES

Opcode	Operand	Meaning
.ORIG	address	starting address of program
.END		end of program
.BLKW	n	allocate n words of storage
.FILL	n	allocate one word, initialize with value n
.STRINGZ	n-character string	allocate n+1 locations, initialize w/characters and null terminator