Department of Electrical and Computer Engineering
The University of Texas at Austin

EE 306, Fall 2009
Yale Patt, Instructor
Aater Suleman, Chang Joo Lee, Ameya Chaudhari, Antonius Keddis, Arvind Chandrababu, Bhargavi Narayanasetty, Eshar Ben-dor, Faruk Guvenilir, Marc Kellermann, RJ Harden, TAs
Exam 2, November 18, 2009

Name: Solution

Problem 1 (15 points):_____

Problem 2 (15 points):_____

Problem 3 (20 points):_____

Problem 4 (10 points):_____

Problem 5 (20 points):_____

Problem 6 (20 points):_____

Total (100 points):_____

Note: Please be sure that your answers to all questions (and all supporting work that is required) are contained in the space provided.

Note: Please be sure your name is recorded on each sheet of the exam.

I will not cheat on this exam.

_____
Signature

GOOD LUCK!

Name:_____

Problem 1. (15 points):
Part a. (5 points): Two lines of assembly code contain the following:

```
.STRINGZ "000"
.BLKW    4
```

Does the assembler generate the identical lines of machine code for both pseudo-ops? If yes, what is the machine code. If no, explain the difference (in no more than 20 words).

No. The .STRINGZ assembles into x30, x30, x30, x0
The four words in .BLKW are not specified

Part b. (5 points): In the data path, there is a MUX (PCMUX) whose output is used to load the PC. There are three inputs to that MUX. For each input, list the LC-3 instructions that use that input at some point in their instruction cycles.

Left input	Middle input	Right input
TRAP RTI (JMP)	JMP, JSR, JSRR BR, RET	All

Part c. (5 points): In two successive clock cycles, the PC is loaded. How is that possible? Explain in no more than 20 words.

The instruction which just finished is a control instruction and the next instruction was just fetched.

Name:_____

Problem 2. (15 points):
Part a. (10 points): A student wrote the following assembly language program:

```
          .ORIG   x4000
        0 LEA     R2,ONE
        1 AND     R5,R5,#0
TWO     2 ADD     R2,R2,#2
        3 STR     R5,R2,#0
        4 BRzp    TWO
        5 TRAP    x25
ONE     6 .BLKW   #50
THREE     .STRINGZ "Input a character:"
FOUR      .FILL   xC000
          .END
```

PC initially contains x4000.
Does the program ever halt? Circle one: (YES) NO
If yes, explain what makes it halt. If no, explain why not.

When the pointer in R2 reaches x8000.

What does this program do (in no more than 15 words)?

Store 0x0000 in every other memory location from x4008-x8000

Part b. (5 points): In assembling the program of part a, the assembler creates a symbol table during the assembly process. Show the symbol table in the box below.

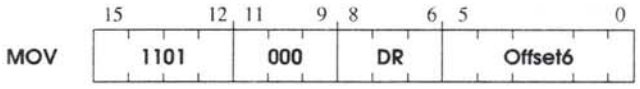
Symbol	Address
TWO	x4002
ONE	x4006
THREE	x4038
FOUR	x404B

x4006
x32
x4038

x13
x4038
x13
x404B

Name:_____

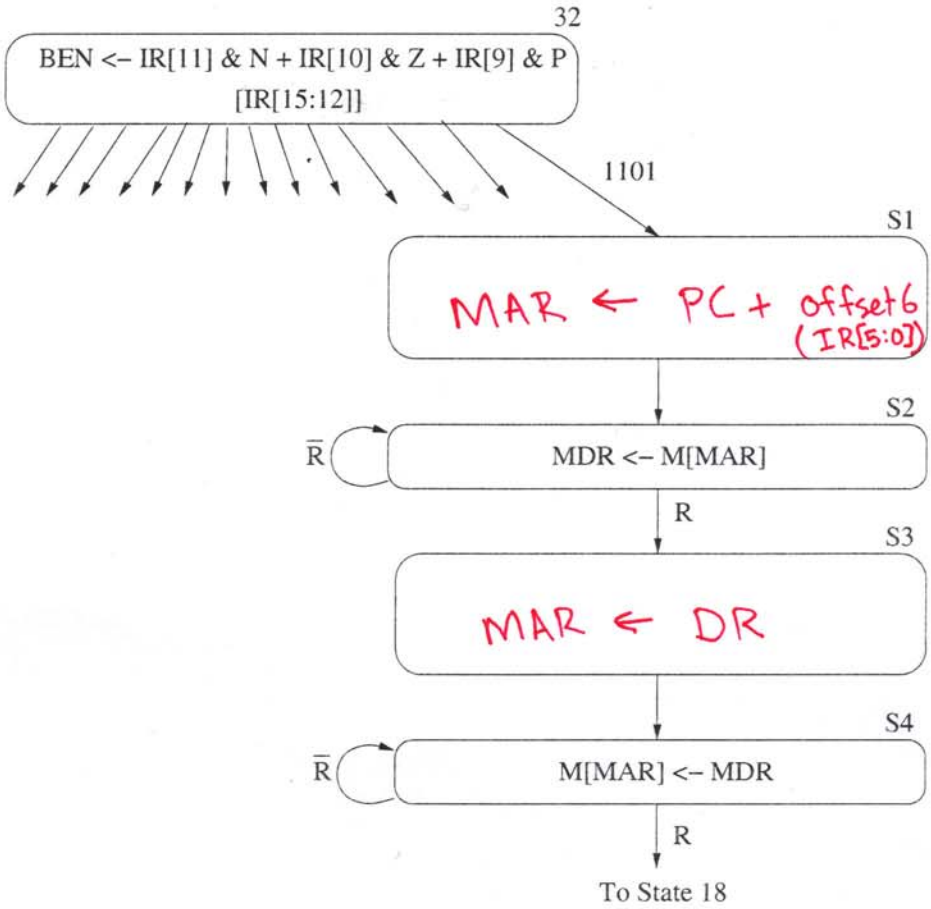
Problem 3. (20 points): As you know, in order to store into memory location A a value that is contained in memory location B, we have to first load the value into a register, and after that store it into A. Someone suggested we could do this with a single new instruction without changing the datapath in any way. Using the reserved opcode 1101, the new instruction has the following format:



The address of the source is computed by adding the incremented PC to SEXT(IR[5:0]). The address of the destination is in the register specified by IR[8:6].

Shown below are four states of the state machine required to do this job, labeled S1, S2, S3, and S4. A partial table of control signals is shown on the next page.

Part a. (10 points): Identify what processing is necessary in states S1 and S3.



Name:_____

Part b. (10 points): Complete the table, identify the value of each control signal during each of the four states S1, S2, S3, and S4. Use the convention specified below.

Note: For a particular state, if the value of a control signal does not matter, fill it with an X.

State	ADDR1 MUX	ADDR2 MUX	MAR MUX	ALUK	LD. MAR	LD. MDR	Gate MARMUX	Gate ALU	Gate MDR	MIO.EN	R/W
S1	1	10	1	X	1	0	1	0	0	0	0
S2	X	X	X	X	0	1	0	0	0	1	0
S3	0	11	1	X	1	0	1	0	0	0	0
S4	X	X	X	X	0	0	0	0	0	1	1

```
ADDR1MUX        0: SR1OUT
                1: PC

ADDR2MUX        00: IR[10:0]
                01: IR[8:0]
                10: IR[5:0]
                11: 0

MARMUX          0: ZEXT(IR[7:0])
                1: From adder

ALUK            00: ADD
                01: AND
                10: NOT
                11: Pass input A

LD.MAR          0: load not enabled
                1: load enabled

LD.MDR          0: load not enabled
                1: load enabled

GateMARMUX      0: do not pass signal
                1: pass signal

GateALU         0: do not pass signal
                1: pass signal

GateMDR         0: do not pass signal
                1: pass signal

MIO.EN          0: memory not accessed
                1: memory accessed

R/W             0: Read
                1: Write
```

Name:_____

Problem 4. (10 points): The following program is supposed to take a string of lower-case letters entered from the keyboard, convert the characters to upper-case, and then display the resulting string on the screen.

```
          .ORIG  x3000
          LEA    R0, PROMPT
          TRAP   x22            ; print prompt

          LEA    R1, INPUT
AGAIN     TRAP   x20            ; input a character
          TRAP   x21            ; echo the character
          ADD    R2, R0, #-10   ; subtract line feed
          BRz    DONE
          JSR    CONVERT        ; call convert
          STR    R0, R1, #0     ; save letter
          ADD    R1, R1, #1     ;
          BRnzp  AGAIN

DONE      AND    R0, R0, #0
          STR    R0, R1, #0     ; save NULL
          LEA    R0, INPUT
          TRAP   x22            ; print output string
          HALT

CONVERT   ST     R2, SaveR2
          LD     R2, MASK
          AND    R0, R0, R2     ; convert to uppercase
          LD     R2, SaveR2
          RET
SaveR2    .BLKW 1

PROMPT    .STRINGZ "Type a string, followed by Enter:"
INPUT     .BLKW 8
MASK      .FILL x00DF    ; NOT of 0x20
          .END
```

The program was executed twice with the following results displayed on the screen:

First time:

```
Type the string, followed by Enter: abcd
ABCD
```

Second time:

```
Type the string, followed by Enter: introduction
INTRODUCT@DD
```

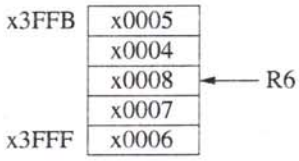
Explain why "ion" was converted into "@DD" rather than "ION". Please be specific.

The letter 'T' overwrote the MASK. Letters after 'T' are ANDed with x54, instead of xDF. For example, T(x54) AND i(x69) give @(x40). Similarly, T(x54) AND o(x6F) is D(x44)

Name:_____

Problem 5. (20 points): There are times when one wants to implement a stack in memory, but can not provide enough memory to be sure there will always be plenty of space to push values on the stack. Furthermore, there are times (beyond EE 306) when it is okay to lose some of the oldest values pushed on the stack. We can save that discussion for the last class if you like.

In such situations, a reasonable technique is to specify a circular stack as shown below. In this case, the stack occupies five locations x3FFB to x3FFF. Initially, the stack is empty, with R6 = x4000. The figure shows the result of successively pushing the values 1, 2, 3, 4, 5, 6, 7, 8 on the stack.



That is, the 1 was written into x3FFF, the 2 was written into x3FFE, etc. When the time came to push the 6, the stack was full, so R6 was set to x3FFF, and the 6 was written into x3FFF, clobbering the 1 which was originally pushed.

If we now pop five elements off the stack, we get 8, 7, 6, 5,and 4, AND we have an empty stack, even though R6 contains x3FFD. Why? Because 3,2, and 1 have been lost. That is, even though we have pushed 8 values, there can be at most only five values actually available on the stack for popping. We keep track of the number of actual values on the stack in R5.

Note that R5 and R6 are known to the calling routine, so a test for underflow can be made by the calling program using R5. Further, the calling program puts the value to be pushed in R0 before calling PUSH.

Your job: Complete the assembly language code shown below to implement the PUSH routine of the circular stack by filling in each of the four boxes with a missing instruction. We will save the POP routine for the final exam.

```
PUSH    ST R1, SAVER
        LD R1, NEGFULL
        ADD R1, R6, R1
        [ BRp SKIP          ]
        LD R6, BASE
SKIP    ADD R6, R6, #-1
        LD R1, MINUS5
        ADD R1, R5, R1
        BRz END
        [ ADD R5,R5,#1      ]
END     [ STR R0,R6,#0      ]
        [ LD R1,SAVER       ]
        RET
NEGFULL .FILL xC005     ; x-3FFB
MINUS5  .FILL xFFFB     ; #-5
BASE    .FILL x4000
SAVER   .BLKW #1
```

Name:_____

Problem 6. (20 points): A program running on the LC-3 has reached a breakpoint. The LC-3 registers contain the following values:

PC	R0	R1	R2	R3	R4	R5	R6	R7
xF018	xFF11	xB1CD	xE000	x613C	x5151	xAC22	x1234	xF019

The computer operator immediately presses the run button, resuming execution. The table below shows a memory trace of the first seven memory accesses after execution resumes.

Hint: What state is the computer in when a breakpoint occurs? Therefore, what is the first memory access when execution is resumed?

	MAR	MDR	
1st:	x F018	x4080	; Fetched JSRR R2, PC←R2 R7←xF019
2nd:	x E000	xE002	; LEA R0, #2; R0 ← xE003, PC←PC+1
3rd:	x E001	xBFFE	; STI R7, #-2;
4th:	x E000	x E002	; get address
5th:	x E002	x F019	; store.
6th:	x E002	x F019	; TRAP x19
7th:	x 0019	x1A00	; get trap routine's address.

Your job:

Part a. (14 points): Fill in the missing values in the above table.

Part b. (6 points): What are the values in PC, R0, and R7 during the cycle in which the 8th memory access occurs.

PC: x 1A01 R0: x E003 R7: x E003