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

EE 306, Fall 2013
Yale Patt, Instructor
Ben Lin, Mochamad Asri, Ameya Chaudhari, Nikhil Garg, Lauren Guckert,
Jack Koenig, Saijel Mokashi, Sruti Nuthalapathi, Sparsh Singhai, Jiajun Wang
Exam 2, November 13, 2013

Name: Key

Problem 1 (20 points):______

Problem 2 (20 points):______

Problem 3 (20 points):______

Problem 4 (20 points):______

Problem 5 (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: Key

Problem 1. (20 points):
Part a. (5 points):

A student is debugging his program. His program does not have access to memory locations x0000 to x2FFF. Why that is the case we will discuss before the end of the semester. The term is "privileged memory" but not something for you to worry about today.

He sets a breakpoint at x3050, and then starts executing the program. When the program stops, he examines the contents of several memory locations and registers, then hits single step. The simulator executes one instruction and then stops. He again examines the contents of the memory locations and registers. They are as follows:

	Before	After
PC	x3050	x3051
R0	x2F5F	xFFFF
R1	x4200	x4200
R2	x0123	x0123
R3	x2323	x2323
R4	x0010	x0010
R5	x0000	x0000
R6	x1000	x1000
R7	x0522	x0522
M[x3050]	**x6???**	**x6???**
M[x4200]	x5555	x5555
M[x4201]	xFFFF	xFFFF

Complete the contents of location x3050

x3050	0	1	1	0	0	0	0	0	0	1	0	0	0	0	0	1

Part b. (5 points):
A student is writing a program and needs to subtract the contents of R1 from the contents of R2, and put the result in R3. Instead of writing:

```
NOT   R3,R1
ADD   R3,R3,#1
ADD   R3,R3,R2
```

he writes:

```
NOT   R3,R1
.FILL x16E1
ADD   R3,R3,R2
```

He assembles the program and attempts to execute it. Does the subtract execute correctly? Why or why not?

Circle one: [✓ YES]/NO. Explain in not more than fifteen words.
.FILL x16E1 is assembled and equal to the assembled instruction ADD R3, R3, #1

Name: Key

Part c. (5 points):
An assembly language program contains the following subroutine, which the LC-3 assembler stores in memory, starting at location x3070.

Construct the symbol table entries for the subroutine. **Hint: I am asking you to ONLY construct the symbol table entries for this subroutine, and nothing more.**

```
INPUT   ST R7 SAVER7
        LD  R1, MINUS
        LEA R5, BUFFER
        LD R0, LF
        TRAP x21
        LEA R0, PROMPT
        TRAP x22
        LD  R0, LF
        TRAP x21
AGAIN   TRAP x20
        STR R0, R5, #0
        ADD R0,R0,R1
        BRz NEXT
        ADD R5, R5, #1
        BRnzp AGAIN
NEXT    LD R7, SAVER7
        RET
SAVER7  .BLKW 1
MINUS   .FILL xFFDD
BUFFER  .BLKW x21
PROMPT  .STRINGZ "Type a word, then type #"
LF      .FILL x0A
```

Label	Address
INPUT	x3070
AGAIN	x3079
NEXT	x307F
SAVER7	x3081
MINUS	x3082
BUFFER	x3083
PROMPT	x30A4
LF	x30BD

Name: Key

Part d. (5 points):
An Aggie (always an Aggie!) modified the service routine that the operating system executes as a result of a user program executing the TRAP x20 instruction. The modification consists of inserting three lines of code into the trap service routine:

1. Adding the following instruction to the beginning of the service routine:

```
        LD   R2,MASK
```

2. Inserting the instruction AND R0,R1,R2 in the place shown in the original service routine:

```
AGAIN   LDI  R1,KBSR
        AND  R0,R1,R2
        BRzp AGAIN
        LDI  R0, KBDR
```

3. Inserting the following pseudo-op immediately after RET:

```
MASK    .FILL x7FFF
```

The complete TRAP service routine after adding the three changes:

```
        ST  R1, SaveR1
        ST  R2, SaveR2
;
        LD  R2,MASK
;
AGAIN   LDI  R1,KBSR
        AND  R0,R1,R2
        BRzp AGAIN
        LDI  R0, KBDR
;
        LD  R1, SaveR1
        LD  R2, SaveR2
;
        RET
MASK     .FILL x7FFF
KBSR    .FILL xFE00
KBDR    .FILL xFE02
SaveR1  .BLKW 1
SaveR2  .BLKW 1
```

Your job: Answer the question: Will the trap service routine still work, and explain why or why not in fifteen words or fewer.

No. The inserted AND instruction prevents the ready bit from ever being detected, creating an infinite loop.

Name: Key

Problem 2. (20 points):
The following program pushes elements onto a stack with JSR PUSH and pops elements off of the stack with JSR POP.

```
                      .ORIG X3000
                      LEA  R6, STACK_BASE

X                     TRAP  x20           ;GETC
                      TRAP  x21           ;OUT
                      ADD  R1, R0, x-0A   ;x0A is ASCII code for line feed,
                                          ;x-0A is the negative of x0A
                      BRz  Y
                      JSR  PUSH
                      BRnzp  X

Y                     LEA  R2, STACK_BASE
                      NOT  R2, R2
                      ADD  R2, R2, #1
                      ADD  R3, R2, R6
                      BRz  DONE
                      JSR  POP
                      TRAP  x21           ;OUT
                      BRnzp  Y

DONE                  TRAP  x25           ;HALT
STACK                 .BLKW  5
STACK_BASE            .FILL  x0FFF

PUSH                  ADD  R6, R6, #-1
                      STR  R0, R6, #0
                      RET

POP                   LDR  R0, R6, #0
                      ADD  R6, R6, #1
                      RET

                      .END
```

What will appear on the screen if a user, sitting at a keyboard, typed the three keys a, b, c, followed by the <Enter> key?

abc
cba

What will happen if a user, sitting at a keyboard, typed the eight keys a, b, c, d, e, f, g, h, followed by the <Enter> key? (Please, no more than fifteen words in the box.)

the stack will overflow and overwrite the [illegible] TRAP x25 instruction, causing an infinite loop

Name: Key

Problem 3. (20 points):

An aggressive young engineer decides to build and sell the LC-3, but is told that if he wants to succeed, he really needs a SUBTRACT instruction. Given the unused opcode 1101, he decides to specify the SUBTRACT instruction as follows:

15 12	11 9	8 6	5 3	2 0
1101	DR	SR1	000	SR2

The instruction is defined as: DR ← SR2 - SR1, and the condition codes are set. **Assume DR, SR1, and SR2 are all different registers.**

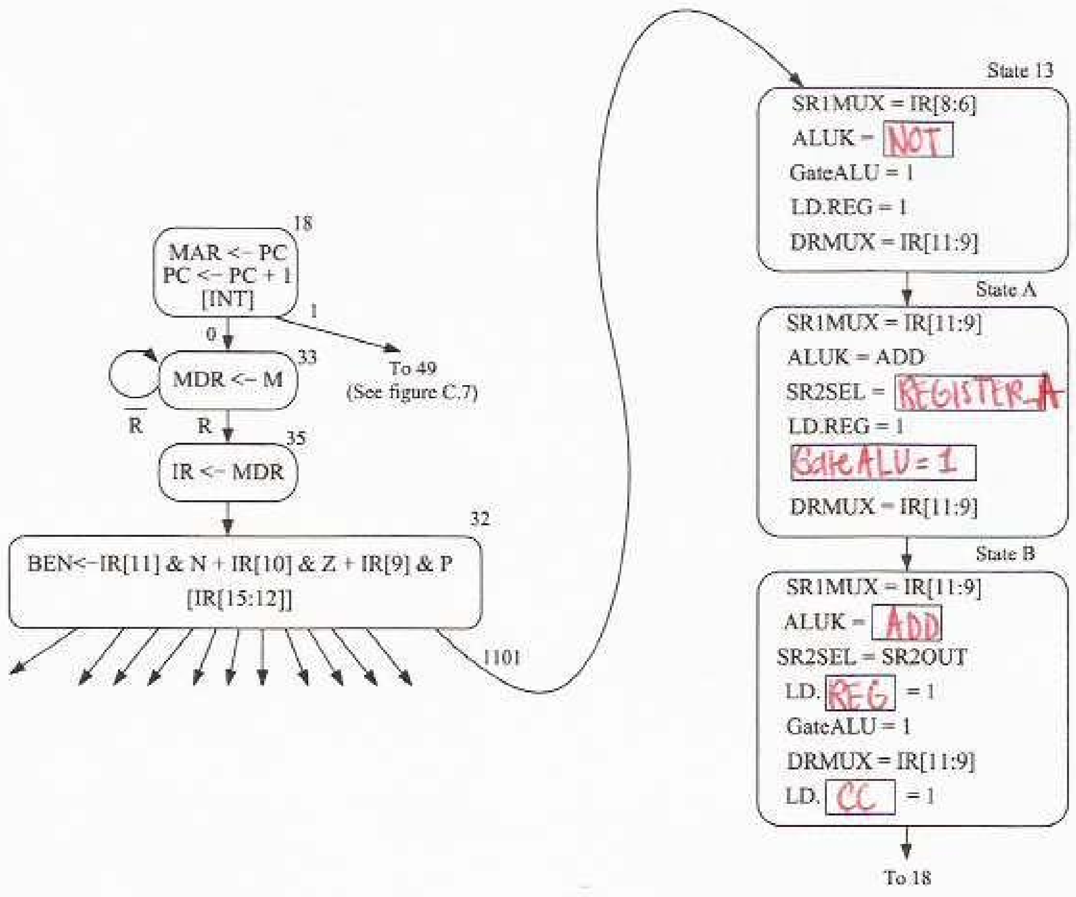
To accomplish this, the engineer needs to add three states to the state machine and a mux and register A to the data path. The modified state machine is shown below and the modified data path is shown on the next page. The mux is controlled by a new control signal SR2SEL which selects one of its two sources.

SR2SEL/1: SR2OUT, REGISTER_A

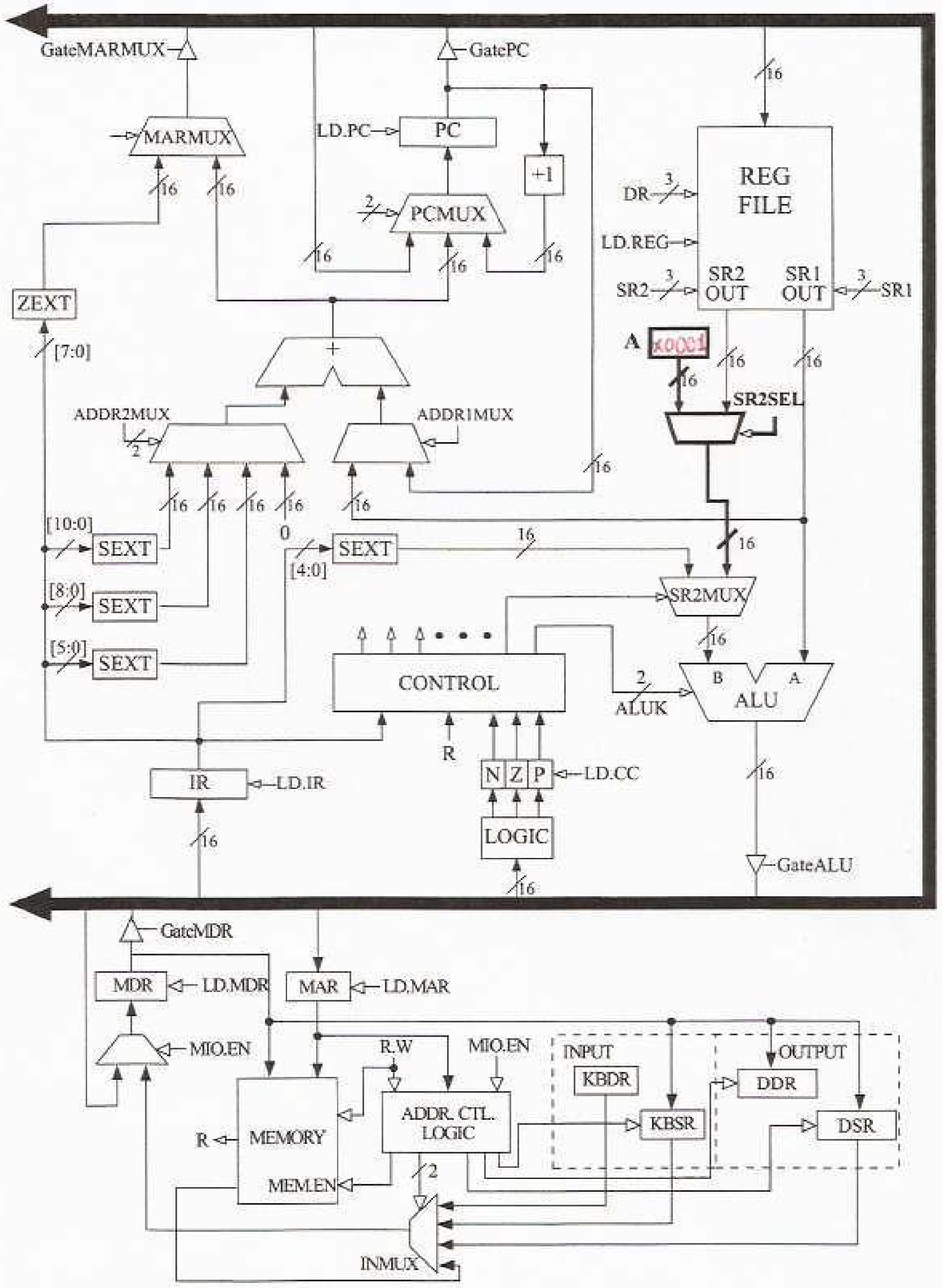
Your job:

For the state machine shown below, fill in the empty boxes with the control signals that are needed in order to implement the SUBTRACT instruction.

For the data path, fill in the value in register A.



Key



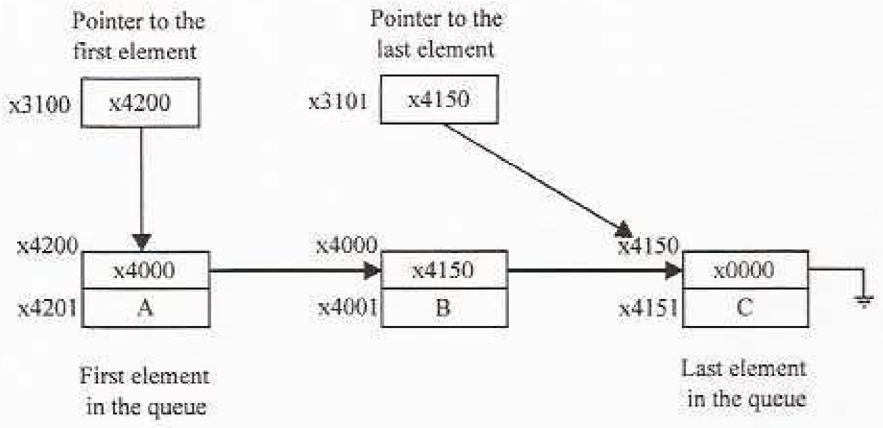
GateMARMUX
GatePC
MARMUX
LD.PC
PC
+1
PCMUX
REG FILE
DR
LD.REG
SR2
SR2 OUT
SR1 OUT
SR1
A
x0001
SR2SEL
ZEXT
[7:0]
ADDR2MUX
ADDR1MUX
[10:0]
SEXT
[8:0]
[5:0]
[4:0]
0
SR2MUX
CONTROL
ALUK
ALU
R
N Z P
LD.CC
LOGIC
IR
LD.IR
GateALU
GateMDR
MDR
LD.MDR
MAR
LD.MAR
MIO.EN
R.W
MEMORY
MEM.EN
ADDR. CTL. LOGIC
INPUT
KBDR
KBSR
OUTPUT
DDR
DSR
INMUX
16

Name: Key

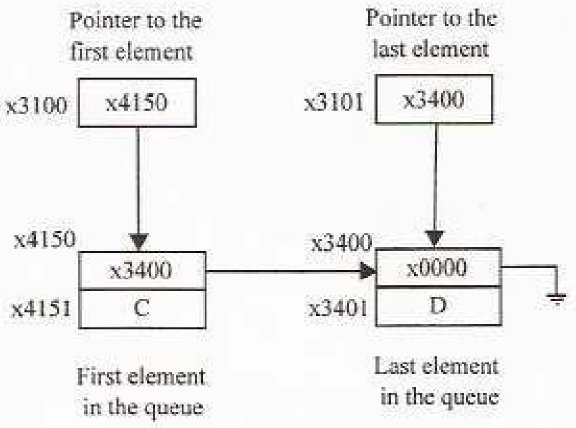
Problem 4. (20 points):
In class we talked about stacks, which are LIFO (Last In, First Out) mechanisms that allow us to push (insert) and pop (remove) elements from the top. Another data structure is a queue. It operates as a FIFO (First In, First Out) mechanism. Elements are removed from the front and inserted in the back, much like the way a queue works in our daily lives.

Our queue is implemented as a linked list. Each element consists of two words: a pointer to the next element that entered the queue and a value. We also need two pointers, one to the front of the queue which we use to remove elements, and one to the last element of the queue which we use to add another element. The last element points to NULL (x0000).

The figure below shows a queue with three elements, the first is A, the second is B, the last is C. M[x3100] contains the address of the front of the queue. M[x3101] contains the address of the last element of the queue.


Pointer to the first element
Pointer to the last element
x3100
x4200
x3101
x4150
x4200
x4000
x4201
A
x4000
x4150
x4001
B
x4150
x0000
x4151
C
First element in the queue
Last element in the queue

If we add an element D and we remove the elements A and B, the queue looks like this:


Pointer to the first element
Pointer to the last element
x3100
x4150
x3101
x3400
x4150
x3400
x4151
C
x3400
x0000
x3401
D
First element in the queue
Last element in the queue

Name: Key

Your job: Complete the subroutines to dequeue (remove) the front element of the queue and enqueue (insert) a new element to the back of the queue. After the DEQUEUE subroutine is executed, R3 should contain the address of the element that was just dequeued; before the ENQUEUE subroutine is executed, R3 will contain the address of the new element to be enqueued. You do NOT have to worry about the case when the queue is (or becomes) empty; that is, you can assume the queue will always have at least one element before and after any operation.

```
DEQUEUE    ST   R0, A
           LDI R3, B
LDR RØ,R3,#0
STI  RØ,B
           LD   R0, A
           RET
A          .BLKW  1
B          .FILL  x3100
```

```
ENQUEUE    ST   R0, C
           LDI R0, D
STR R3, R0,#0
STI R3,D
           LD   R0, C
           RET
C          .BLKW  1
D          .FILL  x3101
```

Name: Key

Problem 5. (20 points):

During the execution of an LC-3 program, an instruction in the program starts executing at clock cycle T and requires 15 cycles to complete.

The table below lists **ALL** five clock cycles during the processing of this instruction which require use of the bus. The table shows for each of those clock cycles: which clock cycle, the state of the state machine, the value on the bus, and the important control signals that are active during that clock cycle.

Note: In class on Monday, I gave an example where it took 18 clock cycles for memory to read or write. Part (d) of this problem asks you how many clock cycles it takes for memory to read or write in this example.

Cycle	State	Bus	Important Control Signals For This Cycle
T	18	x3010	LD.MAR = 1, LD.PC =1, PCMux = PC + 1, GatePC = 1
T + 4	35	xA202	GateMDR = 1, LD.IR = 1
T + 6	10	x3013	ADDR1MUX = PC, ADDR2MUX = PC offset9, MARMUX = ADDER, GateMARMUX = 1, LD.MAR = 1
T + 10	26	x4567	GateMDR = 1, LD.MAR = 1
T + 14	27	x0000	LD.REG = 1, LD.CC = 1, GateMDR = 1, DR = 001

a. Fill in the missing entries in the table.

b. What is the instruction being processed? LDI R1, #2

c. Where in memory is that instruction? x3010

d. How many clock cycles does it take memory to read or write? 3

e. There is enough information above for you to know the contents of three memory locations. What are they and what are their contents?

Memory address	Contents
x3010	xA202
x3013	x4567
x4567	x0000