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

EE 306, Fall, 2008
Yale Patt, Instructor
TAs: Jeffrey Allan, Arvind Chandrababu, Eiman Ebrahimi, Aravind Jakkani, Khubaib, Allison Korczynski, Pratyusha Nidamaluri, Zrinka Puljiz, Che-Chun Su, Christopher Wiley.
Final Exam, December 10, 2008

Name: Solution

Problem 1 (20 points):_____

Problem 2 (15 points):_____

Problem 3 (10 points):_____

Problem 4 (15 points):_____

Problem 5 (15 points):_____

Problem 6 (20 points):_____

Problem 7 (20 points):_____

Total (115 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 written legibly on each sheet of the exam.

I will not cheat on this exam.

_____
Signature

GOOD LUCK!

Name: Solution

Problem 1 (20 points)

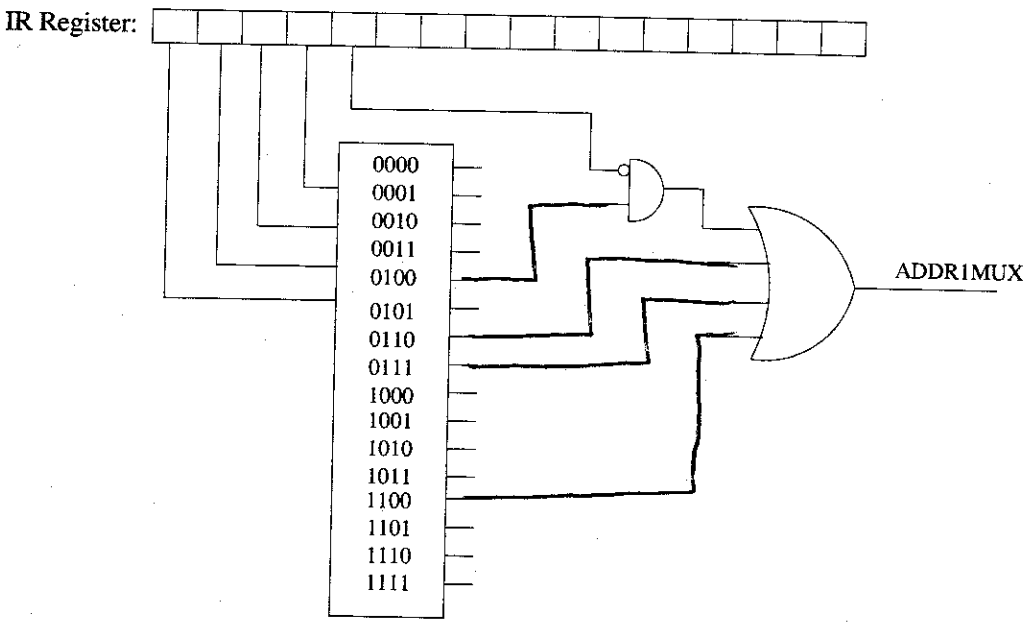
Part a (5 points): A main program puts a value in R0, and then calls SUB. What does the subroutine SUB do? Please answer in ten words or fewer.

```
SUB     ST R1, SaveR1
        ADD R1,R0,R0
        ADD R1,R1,R1
        ADD R1,R1,R1
        ADD R0,R0,R0
        ADD R0,R0,R1
        LD R1, SaveR1
        RET
SaveR1  .FILL x0
```

Put answer here:

R0 ← 10 x R0

Part b (5 points): The MUX select line labeled ADDR1MUX in the LC-3 Data Path is controlled by the output of the OR-gate shown below. Complete the wiring of the inputs to the OR-gate and the second input to the AND gate, to achieve the correct signal for ADDR1MUX.



Name: Solution

Part c (5 points): All instructions load the MDR during the fetch phase of the instruction cycle to fetch the instruction from memory on its way to the IR. After decode has completed, some instructions load the MDR again, using the source 0 input to the mux labeled A on the data path. Other instructions load the MDR, using the source 1 input to mux A. Only one of the 15 LC-3 instructions loads the MDR after decode, using both source 0 and source 1 at different times during the processing of that instruction. What is the opcode of that instruction?

STI

Part d (5 points): You all know that with 5 bits, we can represent 32 different values. If the data type is unsigned integer, those values are the integers from 0 to +31. "Integer" is a special case of "fixed point number" where the binary point is "fixed" to be implicitly to the right of the least significant bit. For example, 11001 really means

$$11001.$$

which is equal to 25 (decimal).

We can design a computer where we get to choose where the fixed point data type places the binary point. There are six choices. If $b_4b_3b_2b_1b_0$ represents a binary string, those six choices are:

$0.b_4b_3b_2b_1b_0$ $\quad$ $b_4.b_3b_2b_1b_0$ $\quad$ $b_4b_3.b_2b_1b_0$ $\quad$ (circled) $b_4b_3b_2.b_1b_0$ $\quad$ $b_4b_3b_2b_1.b_0$ $\quad$ $b_4b_3b_2b_1b_0.$

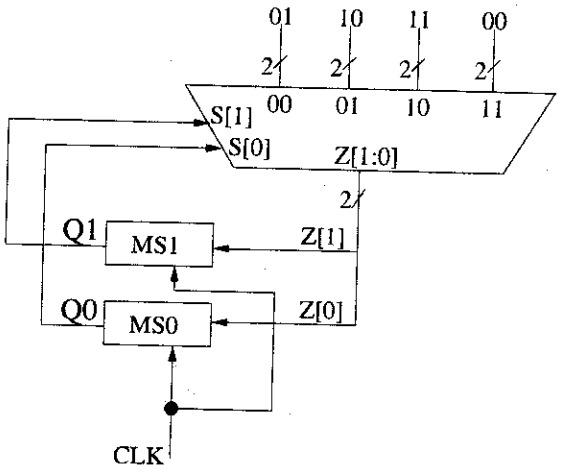
Suppose we wanted the difference (subtraction) between adjacent values to be 1/4. Where would we place the binary point (Circle the correct representation above). What is the largest value that can be represented using this representation. Express as a decimal value.

7.75

Name: Solution

Problem 2 (15 points):

Part a (7 points): In the circuit below, the two master-slave flip flops (MS0 and MS1) each initially contain the value 0. That is, Q1=0, Q0=0. Please fill in the values of Q1 and Q0 after the end of each clock cycle in the table below.



After the end of clock cycle	Q1	Q0
1	0	1
2	1	0
3	1	1
4	0	0
5	0	1
6	1	0
7	1	1

Name: Solution

Part b (8 points):
The following program starts execution at x3000 and stops after executing the HALT instruction.

```
        .ORIG x3000
        LD R0, Addr1
        LEA R1, Addr1
        LDI R2, Addr2
        ADD R1,R1,#-4
        STR R1,R0,#-3
        STI R2,Addr4
        HALT
Addr1   .FILL x3008
Addr2   .FILL x3009
Addr3   .FILL x000A
Addr4   .FILL x300A
        .END
```

Show the contents of R0 through R2 immediately after the ADD instruction in location x3003 executes.

Register	Value
R0	x3008
R1	x3003
R2	x000A

Show the contents of memory locations Addr1 through Addr4 after the LC-3 halts.

Memory Address Label	Value
Addr1	x3008
Addr2	x3009
Addr3	x3008
Addr4	x300A

Name: Solution

Problem 3 (10 points)
The TRAP instruction requires seven states to carry out its instruction cycle, where each state takes one clock cycle. The seven states are shown as rows of the table below. Several control signals determine the processing in the data path during each of the seven clock cycles. Each control signal is shown as a column in the table. Note that the column labeled DR[2:0] is actually a three bit control signal identifying the destination register.

We have provided the control signals for the first four clock cycles of processing the TRAP instruction, which corresponds to the FETCH and DECODE phases of the instruction cycle.
Your job: complete the table by filling in the values of the control signals for clock cycles 5, 6, and 7. Note that if it does not matter whether a particular control signal is 1 or 0, we represent that fact with an X.

Assume a memory access takes one clock cycle.

clock cycle	LD.MAR	LD.MDR	LD.IR	LD.PC	LD.REG	DR[2:0]	MARMUX	PCMUX[1:0]
1	1	0	0	1	0	XXX	X	00
2	0	1	0	0	0	XXX	X	XX
3	0	0	1	0	0	XXX	X	XX
4	0	0	0	0	0	XXX	X	XX
5	1	0	0	0	0	XXX	0	XX
6	0	1	0	0	1	111	X	XX
7	0	0	0	1	0	XXX	X	10

Name: Solution

Problem 4 (15 points):
As you know, the LC-3 ADD instruction adds 16-bit 2's complement integers. If we wanted to add 32-bit 2's complement integers, we could do that with the program shown below. Note that the program requires calling subroutine X which stores into R0 the carry that results from adding R1 and R2.

Your job: Fill in the missing pieces of both the program and the subroutine X, as identified by the empty boxes. Each empty box corresponds to **one** instruction or the operands of **one** instruction.
Note that a 32-bit operand requires two 16-bit memory locations. A 32-bit operand Y has Y[15:0] stored in address A, and Y[31:16] stored in address A+1.

```
.ORIG   x3000
        LEA R3, NUM1
        LEA R4, NUM2
        LEA R5, RESULT
        LDR R1, R3, #0
        LDR  R2, R4, #0
        ADD R0, R1, R2
        STR R0, R5, #0
        JSR X
        LDR R1, R3, #1
        LDR R2, R4, #1
        ADD R1, R1, R0
        ADD R0, R1, R2
        STR R0, R5, #1
        TRAP x25

X       ST  R4, SAVER4
        AND  R0, R0, #0
        AND R4, R1, R2
        BRn LABEL
        ADD R1, R1, #0
        BRn ADDING
        ADD R2, R2, #0
        BRn ADDING
        BRnzp EXIT
ADDING  ADD  R4, R1, R2
        BRn EXIT
LABEL   ADD  R0, R0, #1
EXIT    LD  R4, SAVER4
        RET

NUM1    .BLKW 2
NUM2    .BLKW 2
RESULT  .BLKW 2
SAVER4  .BLKW 1

.END
```

Name: Solution

Problem 5 (15 points):
An Aggie was asked to write a program that will do the following operation ten times: Wait for keyboard input until someone types a character. Then, write that character to the screen. His solution is shown below.

Note that his main program calls the subroutine INPUT ten times. The subroutine INPUT waits for keyboard input. After obtaining keyboard input, it calls the subroutine OUTPUT to output that character to the screen.

There are no assembly time errors in the program. However, there are three serious logical errors (the program does not work as intended). Describe them in the three boxes below. Please, no more than twenty words in each box.

```
.ORIG x3000

        LD R1, SAVER1
LOOP    ADD R1, R1, #-1
        JSR INPUT
        BRp LOOP
        HALT

INPUT   ST R7, SAVER7
        ST R1, SAVER1
BWAIT   LDI R1, KBSR
        BRzp BWAIT
        LDI R1, KBDR
        JSR OUTPUT
        LD R1, SAVER1
        LD R7, SAVER7
        RET

OUTPUT  ST R7, SAVER7
        ADD R7, R1, #0
        STI R7, DDR
        LD R7, SAVER7
        RET

SAVER1  .FILL x000A
SAVER7  .FILL x0000
KBSR    .FILL xFE00
KBDR    .FILL xFE02
DDR     .FILL xFE06

.END
```

"BRp LOOP" does not use condition codes set by ADD instruction.

ST instruction at OUTPUT clobbers return linkage for INPUT subroutine.

Ready bit of DSR is not checked before STI R7, DDR.

Name: Solution

Problem 6 (20 points):
In this problem, you may assume memory initially contains the following values at the following addresses:

Address	Value
x0180	x1000
x1000	xA802
x1001	x7180
x1002	x8000
x1003	xFE02
x3000	xE000
x3001	xEDFE
x3002	x0FFF
x3003	x3000

Your job here is to determine the values in the PC, R0, and R6 at the end of each of six successive instruction cycles. A single instruction executes in one instruction cycle. The values in PC, R0, R6, and xFE00 are initially as shown in the table below (i.e., before the start of the first instruction cycle). The contents of xFE00 are also shown at the end of each instruction cycle. Please fill in the missing values in the table.

Instruction Cycle	PC	R0	R6	xFE00
0	x3000	xFEED	x3004	x4000
1	x3001	x3001	x3004	x4000
2	x3002	x3001	x3000	xC000
3	x1001	x3001	x2FFE	x4000
4	x1002	x3001	x2FFE	x4000
5	x3001	x3001	x3000	x4000
6	x3002	x3001	x3000	x4000

Name: Solution

Problem 7 (20 points):
An LC-3 program starts execution at x3000. During the execution of the program, a snapshot of all 8 registers were taken at six different times as shown below: before the program executes, after execution of instruction 1, after execution of instruction 2, after execution of instruction 3, after execution of instruction 4, after execution of instruction 5, and after execution of instruction 6.

Registers	Initial Value	After 1st Instruction	After 2nd Instruction	After 3rd Instruction	After 4th Instruction	After 5th Instruction	After 6th Instruction
R0	x4006	x4050	x4050	x4050	x4050	x4050	x4050
R1	x5009	x5009	x5009	x5009	x5009	x5009	x5009
R2	x4008	x4008	x4008	x4008	x4008	x4008	xC055
R3	x4002	x4002	x8005	x8005	x8005	x8005	x8005
R4	x4003	x4003	x4003	x4003	x4003	x4003	x4003
R5	x400D	x400D	x400D	x400D	x400D	x400D	x400D
R6	x400C	x400C	x400C	x400C	x400C	x400C	x400C
R7	x6001	x6001	x6001	x6001	x400E	x400E	x400E

Also, during the execution of the program, the PC trace, the MAR trace, and the MDR trace were also recorded as shown below. Note that a PC trace records the addresses of the instructions executed in sequence by the program.

PC Trace
x3000
x3001
x3002
x400D
x5009
x400E

MAR Trace	MDR Trace
x3000	xA009
x3001	x3025
x3025	x4050
x3001	x1703
x3002	xC140
x400D	x4040
x5009	xC1C0
x400E	x1403

Your job: Fill in the missing entries in the 3 tables above.