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

EE 306, Fall, 2004
Yale Patt, Instructor
TAs: Siddharth Balwani, Linda Bigelow, Tommy Buell, Jeremy Carrillo, Aamir Hasan, Danny Lynch, Rustam Miftakhutdinov, Veynu Narasiman, Vishal Parikh, Basit Sheikh
Final Exam, December 10, 2004

Name: Solution

Problem 1 (20 points): 20

Problem 2 (15 points): 15

Problem 3 (15 points): 15

Problem 4 (15 points): 15

Problem 5 (15 points): 15

Problem 6 (15 points): 15

Problem 7 (15 points): 15

Problem 8 (15 points): 15

Total (125 points): 125

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.

Solution
Signature

GOOD LUCK!

Name: Solution

Problem 1 (20 points):

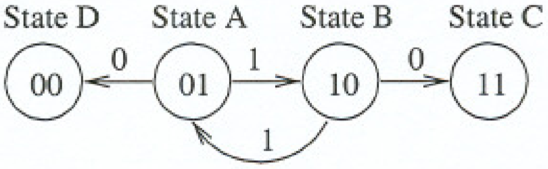
Part a (5 points): The following program is assembled. Complete the symbol table entry for THIS.

Note: Descriptions for pseudo-ops such as .ORIG, .END, .FILL, .BLKW, and .STRINGZ can be found in the auxiliary packet.

```
        .ORIG   x3000
        LD  R0,START
        TRAP    x21
        HALT
START   .STRINGZ "Let's go!"
THIS    .STRINGZ "and THAT"
        .END
```

Symbol	Address
START	x3003
THIS	x300D

Part b (5 points): The finite state machine shown below has one external input and two external outputs. The output for each state is shown inside the state. The machine is initially in state A and receives an input string of n 1's followed by a single 0. What does a final output of 00 signify about n?



Answer: n is even.

Part c (5 points): During the execution of the following program, how many times does the instruction labeled AA get executed?

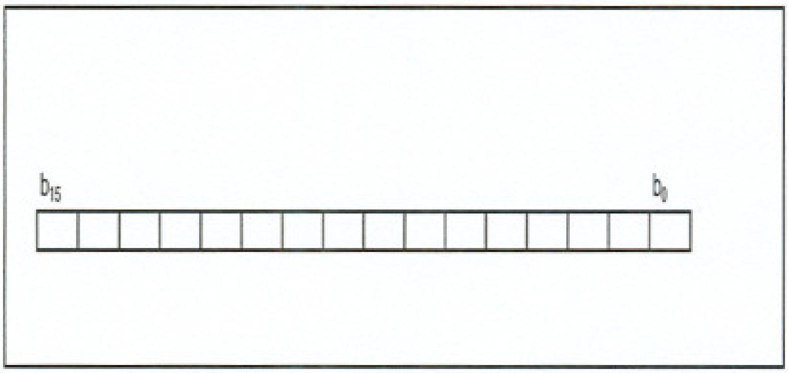
```
        .ORIG x3000
        AND R1, R1, #0
        ADD R3, R1, #2
AA      ADD R3, R3, R3
        BRnp   AA
        HALT
        .END
```

Answer: 15

Name: Solution

Problem 1 (continued):

Part d (5 points): What does the Assembler produce when presented with the assembly language instruction LDR R3, R2, #–35. (An instruction layout is provided for your use, if you need it.)

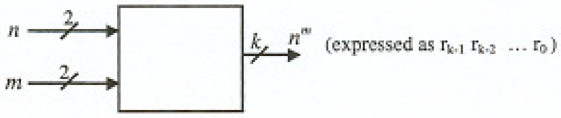


It produces an error message saying that #-35 cannot be represented in 5 bits.

Name: Solution

Problem 2 (15 points):

In this problem you are asked to design a logic circuit that computes n^m, for 2-bit unsigned integers n, m.



Part a : What is the maximum value of n? 3 of m? 3

Part b : What is the maximum possible value for n^m? 27

Part c : In the block diagram above, what is the value of k? 5

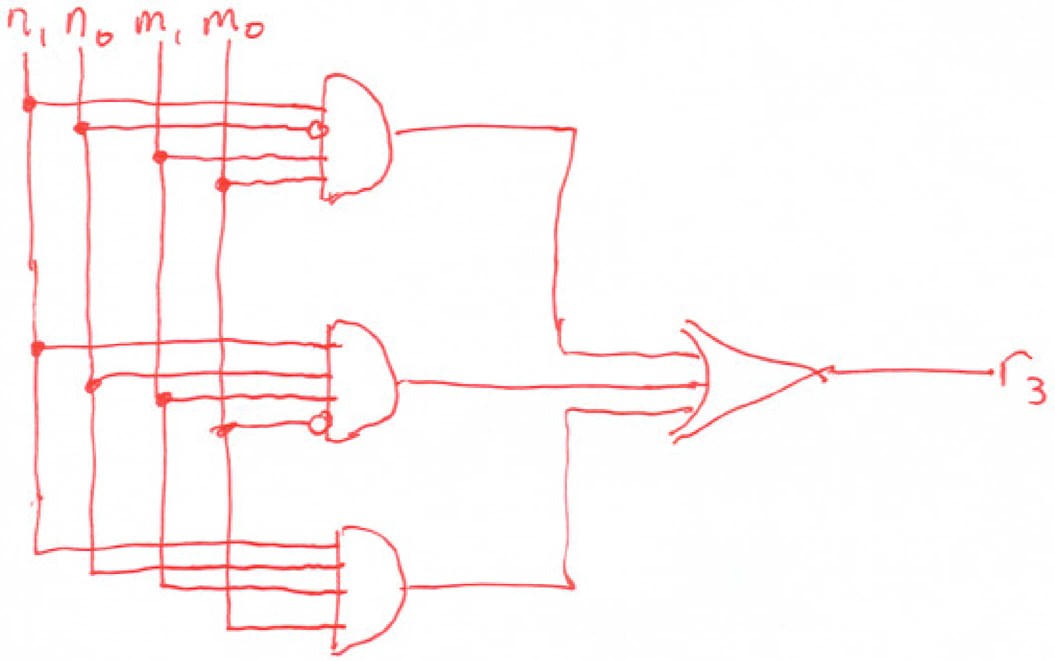
Part d : Construct the truth table for the logic function that computes n^m. (Assume $0^0 = 1$)

n_1	n_0	m_1	m_0	r_4	r_3	r_2	r_1	r_0
0	0	0	0	0	0	0	0	1
0	0	0	1	0	0	0	0	0
0	0	1	0	0	0	0	0	0
0	0	1	1	0	0	0	0	0
0	1	0	0	0	0	0	0	1
0	1	0	1	0	0	0	0	1
0	1	1	0	0	0	0	0	1
0	1	1	1	0	0	0	0	1
1	0	0	0	0	0	0	0	1
1	0	0	1	0	0	0	1	0
1	0	1	0	0	0	1	0	0
1	0	1	1	0	1	0	0	0
1	1	0	0	0	0	0	0	1
1	1	0	1	0	0	0	1	1
1	1	1	0	0	1	0	0	1
1	1	1	1	1	1	0	1	1

Name: Solution

Problem 2 (continued) :

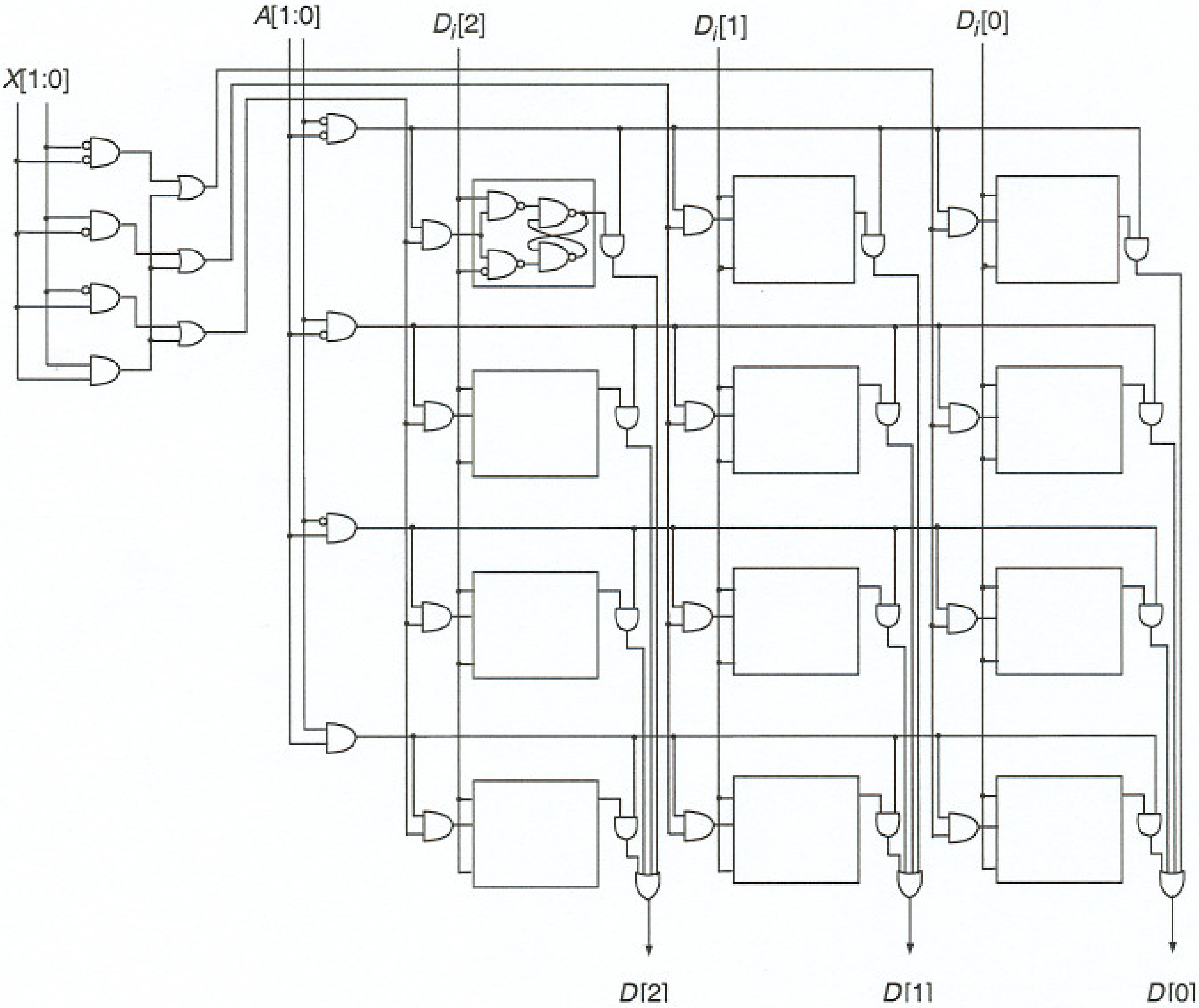
Part e : Construct the logic circuit for the output r_3.



Name: Solution

Problem 3 (15 points):

Figure 3.22 in the textbook shows a 2^2 by 3 memory. In the figure below, we have replaced the one-bit WE signal with a 2-bit signal we call X plus the additional logic that is necessary to perform the functions required by the four values of X. This figure is also included in your auxiliary packet.



Name: Solution

Problem 3 continued :

Part a : In 15 words or fewer each, what function is performed for each of the four values of X.

X=00

Enables writing to bit 0 of the value stored at A.

X=01

Enables writing to bit 1 of the value stored at A

X=10

Enables writing to bit 2 of the value stored at A.

X=11

Enables writing to all bits of the value stored at A.

Part b : What important situation can no longer occur if we replace WE by the 2-bit signal X? (In 15 words or fewer, please.)

Answer:

You can no longer do a pure read from memory.

Name: Solution

Problem 4 (15 points):

What does the following program do? Explain in the box provided below.

```
          .ORIG x3000
          AND R1, R1, #0
          AND R2, R2, #0
          LDI R0, IN_ADDR
 L0       BRz END
          BRp L1
          ADD R1, R1, #1
 L1       ADD R0, R0, R0
          BRnzp L0
 END      ADD R1, R1, #-8
          BRnz L2
          ADD R2, R2, #1
 L2       ST R2, OUTPUT
          HALT
IN_ADDR   .FILL x4000
OUTPUT    .BLKW x1
          .END
```

Answer:

If the value at location x4000 has a majority (> 8) of its bits set, a 1 will be stored at OUTPUT, otherwise a 0 will be stored at OUTPUT.

Name: Solution

Problem 5 (15 points):

Shown below is a snapshot of the 8 registers and the PC of the LC-3 at two instances of time: (1) before the instruction at location x4000 is fetched , and (2) after the instruction at location x4001 has completed. Note that one piece of data is missing.

	Before Instruction at x4000	After Instruction at x4001
PC	x4000	x4010
R0	x0000	x0000
R1	x1000	x1000
R2	x2000	x2000
R3	x3000	x3000
R4	x4000	x020E
R5	x5000	x5000
R6	x6000	x6000
R7	x7000	x7000

Part a : Complete the following two entries:

	15															0
Instruction at x4000:	0	1	1	0	1	0	0	1	0	0	0	0	0	0	0	1

	15															0
Instruction at x4001:	0	0	0	0	0	0	1	0	0	0	0	0	1	1	1	0

Part b : Supply the missing data element in the table (shown above).

Note: More than one correct set of answers is possible for this problem. Any correct set of answers will do.

Note: There are 3 other solutions since the N and Z bits in the branch instruction can be either 0 or 1.

Name: Solution

Problem 6 (15 points):

It has been suggested that we use the unimplemented LC-3 opcode to add push and pop to the LC-3 ISA, as follows:

PUSH:

1110	SR	000 0 00000

POP:

1110	DR	000 1 00000

The assembly language syntax is:

PUSH SR
POP DR

Pushing and popping works as discussed in the textbook, using R6 as the stack pointer. The table shown below shows the contents of several memory locations before and after the following assembly language program is executed.

	Before	After
x3FFB	0	8
x3FFC	0	16
x3FFD	0	4
x3FFE	0	4
x3FFF	0	3

```
          LD R6, STACK; *           0010 110 000010000
          AND R0, R0 #0; *          0 101 000 000 100000
          PUSH R0
          POP R1
          ADD R1, R1, #3
          PUSH R1
          ADD R2, R1, #1
          PUSH R2 ; *           1110 010 000 0 00000
          PUSH R2
          ADD R3, R2, R2 ; *          0001 011 010 000 010
          PUSH R3
          PUSH R3
          POP R2
          POP R1
          ADD R3, R2, R1
          PUSH R3
          HALT
STACK     .FILL x4000
          .END
```

Note that three of the instructions are missing from the assembly language program, and that four instructions have a * in the comment field.

Your job:

Part a : Fill in the three missing instructions in the program, one per box, in the boxes provided.

Part b : For each of the four instructions with the * comment, provide the corresponding machine language instruction in the box to the right of it.

Part c : What is the contents of R6 after this program halts? x3FFC

Name: Solution

Problem 7 (15 points):

What does the following program do? Explain in not more than 15 words in the box provided below.

```
         .ORIG x3000
         AND R1, R1, #0
         ADD R2, R1, #15
         ADD R3, R1, #1
         LDI R0, IN_ADDR
L0       AND R4, R0, R3
         BRz L1
         ADD R1, R1, #1
L1       ADD R1, R1, R1
         ADD R3, R3, R3
         ADD R2, R2, #-1
         BRp L0
         AND R4, R0, R3
         BRz L2
         ADD R1, R1, #1
L2       ST R1, OUTPUT
         HALT
IN_ADDR .FILL x4000
OUTPUT  .BLKW x1
         .END
```

Answer: Reverses the bit string of the value at x4000, and stores the result at OUTPUT.

If you are not sure what this program does, answer the following questions for partial credit. If you know what it does, you do not need to answer these questions.

Part a (1 points): Where in memory is the input to the program:

Answer: x4000

Part b (1 points): Where in memory is the output of the program:

Answer: x3011

Part c (2 points): At the end of the program what are the contents of the following registers (for any input):

R2: 0 R3: x8000

Part d (1 points): What input will cause the output to be 0:

Answer: 0

Part e (2 points): What is the output for the following binary inputs:

0000 0000 0000 0101: 1010 0000 0000 0000 1100 0000 0000 0000: 0000 0000 0000 0011

Name: Solution

Problem 8 (15 points):

What is the output of the following program? Show your work. **Be very careful.**

```
            .ORIG x3000
            AND R1, R1, #0
            LDI R0, START
            ST R1, #0
            LEA  R0, Message1
            Trap x22
            HALT
START      .FILL x3002
Message1   .STRINGZ "I HOPE"
           .BLKW x01F0
Message2   .STRINGZ "I PASSED"
           .END
```

Answer:

PASSED