Comp 150 Exam #3 Overview.

Resources During the Exam

The exam will be closed book, no calculators or computers. You may bring notes on **two** sides of 8.5x11 inch paper (either both sides of two sheets, or four sheets written on single sides). Write this as you study! I mostly want to test you on concepts, not memorized rote facts. Pip/Machine code table supplied.

Main topics that may be on the exam #3: web notes on gates/circuits/truth tables/number systems. Also know how to interpret the web-server programs.

1. Web server programs. You will not have to write code, but you will have to identify code components.
2. Conversions between different number systems (binary,decimal and hexadecimal).
3. Read/write Pip assembler and play computer. Understand the use of the accumulator and symbolic variables and labels for jumps. Follow and be able to write short computational sequences and if-else or while-loop logic with Pip assembler code
4. Be able to convert any way between Boolean expressions, sequential logic circuits, and truth tables.
5. Draw a logic circuit, given a Boolean expression or instruction.

Read the following before looking at either the problems or the solutions! (Same as exam 1)

1. Study first and then look at the sample problems. The sample problems cannot give complete coverage, and if you look at them first, you are likely to study just these points first, and will not get an idea how well you are prepared in general. Look at the list at the top of the page and start by filling in any holes.
2. Do not look at the answers until you have fully studied and tried the problems and gotten *help* getting over rough spots in the problems if you need it! Looking at the answers before this time makes the problems be just a few more displayed examples, rather than an opportunity to actively learn by doing and check out where you are. The *doing* is likely to help you be able to *do* again on a test.

New sample problems start on the next page.

Review Problems for Exam #3 (Solutions follow the problems.)

1. Write a sequence of PIP Assembler or machine code instructions that will copy the value of memory location 130 into memory location 131. (You do not need to write a whole program -- no HLT required.)

2. Convert the PIP machine code to assembler
00001100 00010010
00001111 00000000
00010011 00000100

3. Convert the PIP Assembler to Machine code
JMZ 12
MUL #5
NOT

4. Play computer with the silly program below, completing the log at the right, showing the machine state after each instruction. To save time, you may choose to show only those values that change at each line. To be consistent with the simulator display, I show columns for both the current and next IP addresses, but you only need to fill in the current IP address (left column) of the instruction just executed. The initial values are shown.

```
IP-->  ACCUM   X   Y
-----------------
--   0       0   0   0
 0   2      -5   0   0
```

```
Address    Assembler code
0          LOD #-5
2          STO X
4          MUL #-1
8          STO Y
8          CPL X
10         JMZ L1
12         LOD X
14         ADD Y
16         JMZ L2
18     L1: LOD X
20     L2: ADD X
22         JMP L3
24         SUB #1
26     L3: HLT
```

5. a. Convert the following code to Pip Assembler.
if X == 0:
  Y = 3
else:
  X = Y
Z = X + Y

b. For a challenge, what change would you need if the first line of pseudocode was: if X == Z:
Hint: What is a useful equivalent test?

6. Draw a circuit diagram that corresponds to the following Boolean expression: A(B + (CA)')

A -

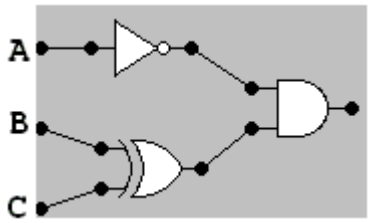
B -

C -

7. Complete the truth table below:

A	B	A'	A'B	A+B	A'B ⊕ (A+B)
0	0				
0	1				
1	0				
1	1				

8. Write a Boolean expression involving A, B, and C that corresponds to the following circuit:



9. Given the truth table below, write a Boolean expression in terms of A, B, and C for X.

A	B	C	X
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

10. Complete the truth table if X is true whenever B is different from both A and C

A	B	C	X
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

11. Do the following base conversions. Show work.
    a. Convert the decimal number 54 into binary.
    b. Convert the binary number 111100110110010010 into hexadecimal, without converting the entire base 2 representation to base 10 first.

12. Do the following base conversions. Show work.
    a. Convert the hexadecimal 2AF to decimal.
    b. Convert the decimal 844 to hexadecimal.

13. Review adder.html, and make sure that you know all what the different lines are referring to.

Answers on the next page

Final Exam Review Problem Answers

1. LOD 130
   STO 131

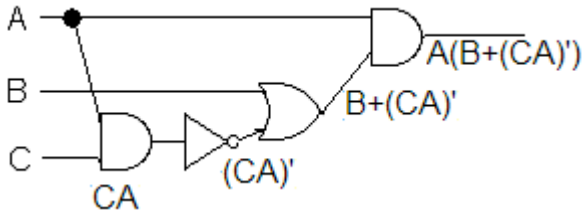
2. JMP 18 ; 0000 no #; JMP code 1100; 18 = 16+2 in binary 00010010
   HLT ; 0000 no #; HLT code is 1111; second byte just 0
   DIV #4 ; pound sign from 0001; DIV code 0011; 4 from binary 00000100

3. 00001101 00001100 ; no #; JMZ; 12 = 8+4
   00010010 00000101 ; # ; MUL; 5 = 4+1
   00001001 00000000 ; no #; NOT; just 0

4.

```
IP--> ACCUM   X Y
---------------
--   0      0   0 0
 0   2     -5   0 0 LOD #-5;acc=-5
 2   4     -5  -5 0 STO X ;X=acc=-5
 4   6      5  -5 0 MUL #-1 ;acc=-5*-1=5
 6   8      5  -5 5 STO Y ;Y=acc=5
 8  10      1  -5 5 CPL X ;-5<0 true acc=1
10  12      1  -5 5 JMZ L1; acc!=0;no jump
12  14     -5  -5 5 LOD X ; acc=X=-5
14  16      0  -5 5 ADD Y ;acc=acc+Y=-5+5=0
16  20      0  -5 5 JMZ L2 ;acc is 0; jump
20  22     -5  -5 5 ADD X ; acc=acc+X=0+-5
22  26     -5  -5 5 JMP L3 must jump
26  --     -5  -5 5 HLT
```

6. (Could use NAND instead of AND and NOT)



7.

A	B	A'	A'B	A+B	A'B ⊕ (A+B)
0	0	1	0	0	0
0	1	1	1	1	0
1	0	0	0	1	1
1	1	0	0	1	1

8. A'(B ⊕ C)
9. A'B'C' + A'BC + ABC

5a.

```
        LOD X   ; acc = X
        NOT     ; if acc != 0 (X != 0) acc now 0 (false)
        JMZ ELSE  ; jump if acc is 0 (X != 0)
        LOD #3
        STO Y
        JMP PAST
ELSE: LOD Y
        STO X
PAST: LOD X
        ADD Y
        STO Z
```

b. if X==Z
is the same as
if X-Z == 0,
so just insert the second line to calculate X-Z
(and I revised the comment on the next line for NOT):

```
        LOD X   ; acc = X
        SUB Z   ; inserted line!   acc = X - Z
        NOT     ; if acc != 0 (X-Z != 0; X != Z) acc now 0
        ...
```

10.

A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

11a. 110110: 54/2 = 27 R 0, 27/2 = 13 R 1, 13/2 = 6 R 1, 6/2 = 3 R 0, 3/2 = 1 R 1, 1/2 = 0 R 1 remainders backwards: 110110

b. 3CD92 11 1100 1101 1001 0010 group from the right!
3 C D 9 2

12a $2*16^2 + 10*16 + 15 = 512 + 160 + 15 = 687$

b. 844/16 = 52 R 12; 52/16 = 3 R 4; 3/16 = 0 R 3 Read remainders from right: 3 4 12; convert to hexadecimal digits: 34C.

(If you do not like arithmetic with 16's, you could do binary conversions in the middle: part a: convert to binary, then decimal. Part b: convert to binary; then hexadecimal, but that is longer to do.)