

Readings:

1. Mano sections 4-1, 4-2 and 4-3 (on register transfer language and tri-state output gates as bus drivers.).
2. PH sections 4.1 to 4.4 **prerequisites from Assembly Language course!**
3. PH section 4.5 (omit carry lookahead on first reading.)
4. Mano rest of chapter 4.
5. Mano 5-1, 5-2 **do not read 5-3 yet because it's hard to understand without knowing the Mano machine's machine language!**
6. Mano 6-1, 6-2; then 5-3.

Practice Problem A: Binary arithmetic and character code review. (Some overlap with Hw 1.) Make sure that you can do the following problems from Mano: 3-1, 3-2, 3-3, 3-4, 3-5, 3-9, 3-10, 3-13, 3-15, 3-17. In necessary, study material from PH 4.1, 4.2, 4.3 and/or Mano 3-1, 3-2, 3-3.

- a. Mano p. 90 3-17, except that instead of using  $(70_{\text{ten}})$  and  $(80_{\text{ten}})$  use  $(7X_{\text{ten}})$  and  $(8Y_{\text{ten}})$  where  $X$  is the last (low order, one's) decimal digit from your student ID and  $Y$  is the next-to-last (ten's) digit from your student ID.
- b. What is the maximum value for the sum of two integers that will not produce overflow when 8 bit signed 2-s complement binary representation is used? What is the minimum (i.e., most negative) value for the sum that will not produce overflow with this representation? Write each answer three ways: 8 bit signed 2-s complement binary, two digit hexadecimal representation of the 8 bit binary representation bit strings, and signed decimal.
- c. Explain why overflow never occurs when two integers with the same sign are subtracted, and when two integers with opposite signs are added.

Problem 1: This problem is to practice creating ROBDDs. Review the quotations on the Web lecture outline page and study the referenced parts of the linked papers.

- a. Create a reduced ordered binary decision diagram for the following Boolean function  $F$  of 2 variables  $X$  and  $Y$  using the total order in which  $X$  is before  $Y$  (so the root of your diagram should be labelled  $X$ ):

$X$	$Y$	$F(X, Y)$
0	0	1
0	1	0
1	0	1
1	1	1

Observe a significance of this function: When  $X$  and  $Y$  are taken to represent unsigned integers (i.e., numbers of arithmetic) using one bit binary encoding,  $F(X, Y) = \mathbf{True}$  precisely when  $X \geq Y$ . It would be smart to verify this for yourself to prepare for what comes next..

- b. Create a ROBDD for the Boolean function  $F_2(X_1, X_0, Y_1, Y_0)$  for which  $F_2(X_1, X_0, Y_1, Y_0) = \mathbf{True}$  precisely when  $(X_1X_0)_{\text{TWO}} \geq (Y_1Y_0)_{\text{TWO}}$ . In English, this means the **number** (0, 1, 2 or 3) represented by  $X_1X_0$  using unsigned binary is greater than or equal to the corresponding number interpretation of  $Y_1Y_0$ . Use the total ordering  $X_1 < Y_1 < X_0 < Y_0$ . (Little or no credit will be given for answers don't have enough clarity and explanatory labels for the TA to quickly verify the correctness or find clerical errors.)

- c. Think of a general plan for ROBDDs that test  $(X_N X_{N-1} \dots X_2 X_1 X_0)_{\text{TWO}} \geq (Y_N Y_{N-1} \dots Y_2 Y_1 Y_0)_{\text{TWO}}$  using the total ordering  $X_N < Y_N < X_{N-1} < Y_{N-1} < \dots < X_0 < Y_0$ . Think of and explain why a plan that uses the different ordering  $X_N < X_{N-1} < \dots < X_1 < X_0 < Y_N < Y_{N-1} < \dots < Y_1 < Y_0$  must have **many** more vertices. (This problem is relatively difficult and it is an example of the kind of research question a computer scientist would ask when ROBDDs were first invented.) Notice the “bad” ordering tests all the bits in array  $X$  first before examining any of the bits of  $Y$ .

Practice Problem B: Mano page 119 problems (a) 4-2 and (b) 4-5.

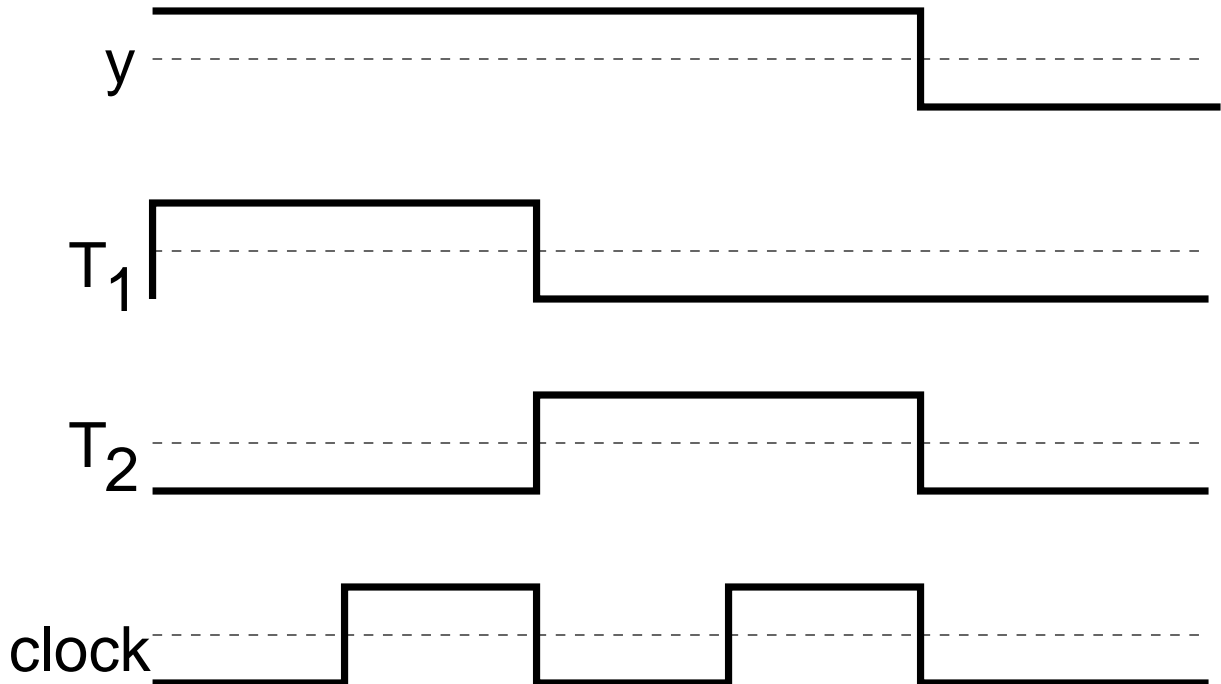
Problem 2:

- a. Mano page 118 problem 4-1.  
 b. Repeat (a) for the RTL specification

$$yT_1 : R2 \leftarrow R1$$

$$yT_2 : R1 \leftarrow R2$$

- c. Suppose that for each system specified by (a) and (b):
- The initial values in  $R1$  and  $R2$  were respectively  $39_{\text{ten}}$  and  $52_{\text{ten}}$ .
  - The waveforms generated by the control circuit are



- The flip-flops are negative edge triggered.

Explain why for system (a) the final values are  $R1 = 52$  and  $R2 = 39$ , while for system (b) the final values are  $R1 = 39$  and  $R2 = 39$ .

See other side...

Problem 3: Copy the “Binary Program” of Table 6-2 onto your homework paper with two blank lines between each line you write. Next to each 16 bit binary instruction code write its hexadecimal representation.

Now explain in the following way what the Mano basic computer does when it executes this program. Assume that when the computer starts,  $AC = 0$ ,  $E = 0$ ,  $PC = 0$  and the given program is in the memory. Underneath the first line, explain what changes when the first line is executed and state what the changed value is. Also write the values of  $AC$ ,  $E$  and  $PC$  after execution of the first line is complete. (Express these in the form  $AC = ??$ ,  $E = ??$ , and  $PC = ??$ .)

If the current instruction changes memory contents, first explain in English and then cross out the old memory word data (so the grader can still see it) and write the new data next to it.

If the current instruction halts the computer, write the  $AC$ ,  $E$  and  $PC$  values as before, write “Computer Halted” and DO NOT go on to the next instruction. (Assume that the  $PC$  was incremented by the HALT instruction.)

When you are done with one instruction, go on to the next, etc.

In order to do this problem, you **must** refer to Figure 5-5 and either Table 5-2 or Table 6-1.

Problem 4: Simulate the Mano basic computer on the machine language program below as you did before, with the following changes.

**First**, copy the program into hexadecimal representation onto the left side of your answer paper. Label each word with its address. Write “Memory Contents/Static Program.” above the program you copied.

**Second**, write “Dynamic Program and Computer Actions” above the space to the right of the Static Program copy.

**Finally**, use the space on the right to copy each instruction before executing it. Solutions that are NOT CLEAR about, say, which instruction execution does what actions will receive little credit.

This program has a loop. Therefore, instructions fetched from some memory locations will be executed more than once. Under the instruction, explain as before what it did. Mark any changes to memory contents on the **Left** side. You may use hexadecimal to write instructions, other memory contents and the  $AC$  contents. You may add the binary equivalent if it will help you simulate the calculations the Mano computer would do.

Location	Instruction code			
0	0010	0000	0000	1010
1	0001	0000	0000	1011
10	0011	0000	0000	1011
11	0110	0000	0000	0000
100	0110	0000	0000	0001
101	0110	0000	0000	0010
110	0110	0000	0000	1001
111	0100	0000	0000	0000
1000	0111	0000	0000	0001
1001	1111	1111	1111	1101
1010	0000	0000	0000	0001
1011	0000	0000	0000	0010
1100	1111	1111	1111	1110
1101	0000	0000	0000	0000

Beware: This is a “self-modifying” program. Serious programmers stopped writing code like this decades ago.

Problem 5: Copy all 16 “Basic Identities...” from Mano’s page 8, with with their numbers and with identity (14) corrected, onto a single vertical column. To the right of each identity, write its **dual**. Then look for each dual among the original identities and write its number or draw an arrow to it.

The dual of a Boolean algebra formula or identity is formed by interchanging **AND** with **OR**, and constants **0** with **1** (leaving **NOT**s unchanged.) The key result about duality is that the dual of an identity is an identity. (Ask yourself: What is an identity?)