922 25300 Computer Organization and Assembly Languages **National Taiwan University**
Fall 2012 **Department of CSIE**

Homework 1

October 4, 2012 Due date: October 17, 2012

1. (12%) What is the binary representation of the following hexadecimal numbers? What are the decimal numbers they represent when interpreted as unsigned and signed integers.
   a. 5C
   b. B2

2. (8%) Prove that (a) {NOT, OR} and (b) {NOR} are universal.

3. (10%) Prove (a) $A + \overline{A}B = A + B$ (b) (A+B)(A+C)=A+BC both algebraically and using the truth table.

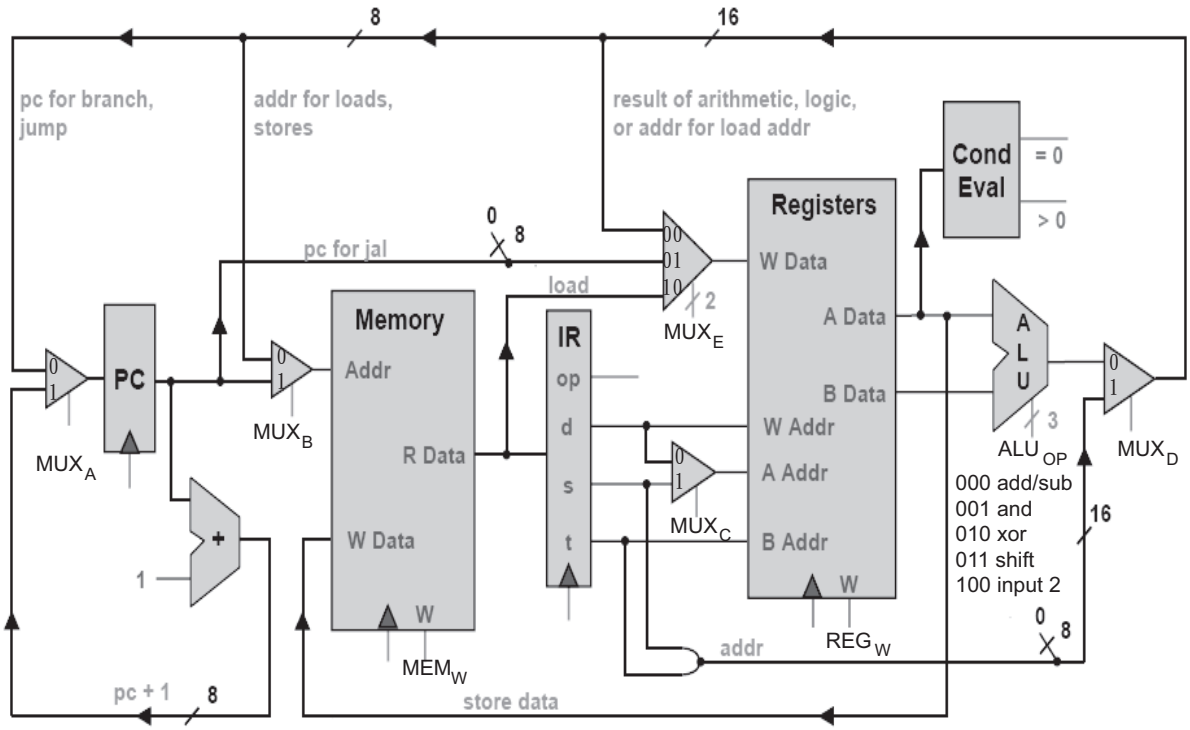
4. (10%) A leading-1 circuit has three inputs A_2, A_1, A_0 and two outputs Z_1, Z_0. The circuit returns the binary encoding of the position of the leading 1-bit. For example, if the input is 010, the output is 01 since the first 1 appears at A_1. For another example, for input 101, the output is 10 (because A_2 contains the leading 1-bit). If there is no 1 in the inputs (000), it returns 11. Implement the circuit with logic gates and components in the class.

5. (20%) Design a binary multiplier that multiplies two 3-bit unsigned integers, $X = X_2X_1X_0$ and $Y = Y_2Y_1Y_0$, and a 6-bit output $Z = Z_5Z_4 \ldots Z_0$ and $Z = X \times Y$, where X_0, Y_0 and Z_0 are LSBs. You may use the notation $X[n..m]$ to identify a portion of wires. For example, X[2..1] means the set of wires, X_2X_1.

6. (20%) Design an 8-bit increase-by-k circuit. The circuit is similar to the counter we mentioned in the class. However, it is different in two ways. First, it only counts up and the increment is a 4-bit constant k specified by the user rather than 1 as the counter in the class. Second, it checks for saturation. That is, if the number after increment is larger than 255, it keeps the number before increment. It has the following interface. For inputs, it has an 8-bit input D_{in}, a 4-bit k_{in} for the new increment and a 2-bit op for the opcode. The output is the 8-bit content of the counter. Its operations are listed in the following table. (Hint: A full-adder has an output for carry out.)

op[1]	op[0]	operation	semantics
0	0	**reset**	$C \leftarrow 0$
0	1	**load**	$C \leftarrow D_{in}$
1	0	**set k**	$k \leftarrow k_{in}$;
1	1	**inc**	if $(C + k < 256)$ then $C \leftarrow C + k$

7. (20%) Refer to the following figure for TOY architecture. Please specify the control signals, MUX_A, MUX_B, MUX_C, MUX_D, MUX_E, REG_W, MEM_W and ALU_{OP} during the execution stage for the instructions "shift left", "store", "load indirect" and "jump resister". For example, the answer for "jump and link" would be $MUX_A = 0, MUX_B = *, MUX_C = *, MUX_D = 1, MUX_E = 01, REG_W = 1, MEM_W = 0, ALU_{OP} = *$. Please note that the numbering of architecture could be different from the one in the class. Please refer to the following figure rather than slides in the class.


8
16
pc for branch, jump
addr for loads, stores
result of arithmetic, logic, or addr for load addr
Cond Eval
= 0
> 0
Registers
W Data
A Data
B Data
W Addr
A Addr
B Addr
W
0
8
pc for jal
load
00
01
10
2
MUX_E
Memory
Addr
R Data
W Data
W
IR
op
d
s
t
PC
MUX_A
MUX_B
MUX_C
MUX_D
ALU
ALU_OP
3
000 add/sub
001 and
010 xor
011 shift
100 input 2
16
+
1
pc + 1
8
MEM_W
REG_W
addr
store data