Spring 2011 Name: _____

Computer Organization Test 2

Question 1. (12 points) Select the best answer to the following true-or-false questions:	**Circle the correct answer**	
a. A jump instruction changes the flow of execution by changing the PC.	True	False
b. Registers are storage locations within the CPU itself.	True	False
c. The main memory (RAM) is faster to access than registers.	True	False
d. Only the AC register in MARIE can be used to hold arbitrary data.	True	False
e. One assembly language instruction generally translates to one machine language instruction.	True	False
f. One high-level language (e.g., Ada, C++, Java, etc.) instruction generally translates to many machine language instruction.	True	False

Question 2. (18 points) Translate the following high-level language code segment to MARIE assembly language. Use the variable labels indicated in the code.

```
SUM = 0
INPUT  COUNT
FOR I = 1 TO COUNT DO
    INPUT  X
    SUM = SUM + X
END FOR
```

Question 3.

a) (5 points) For the below MARIE program, complete the symbol table.

Symbol	**Address of Symbol (in hex.)**
END_WHILE	
SUM	
WHILE	
X	

b) (10 points) Translate the given MARIE assembly language into machine language.

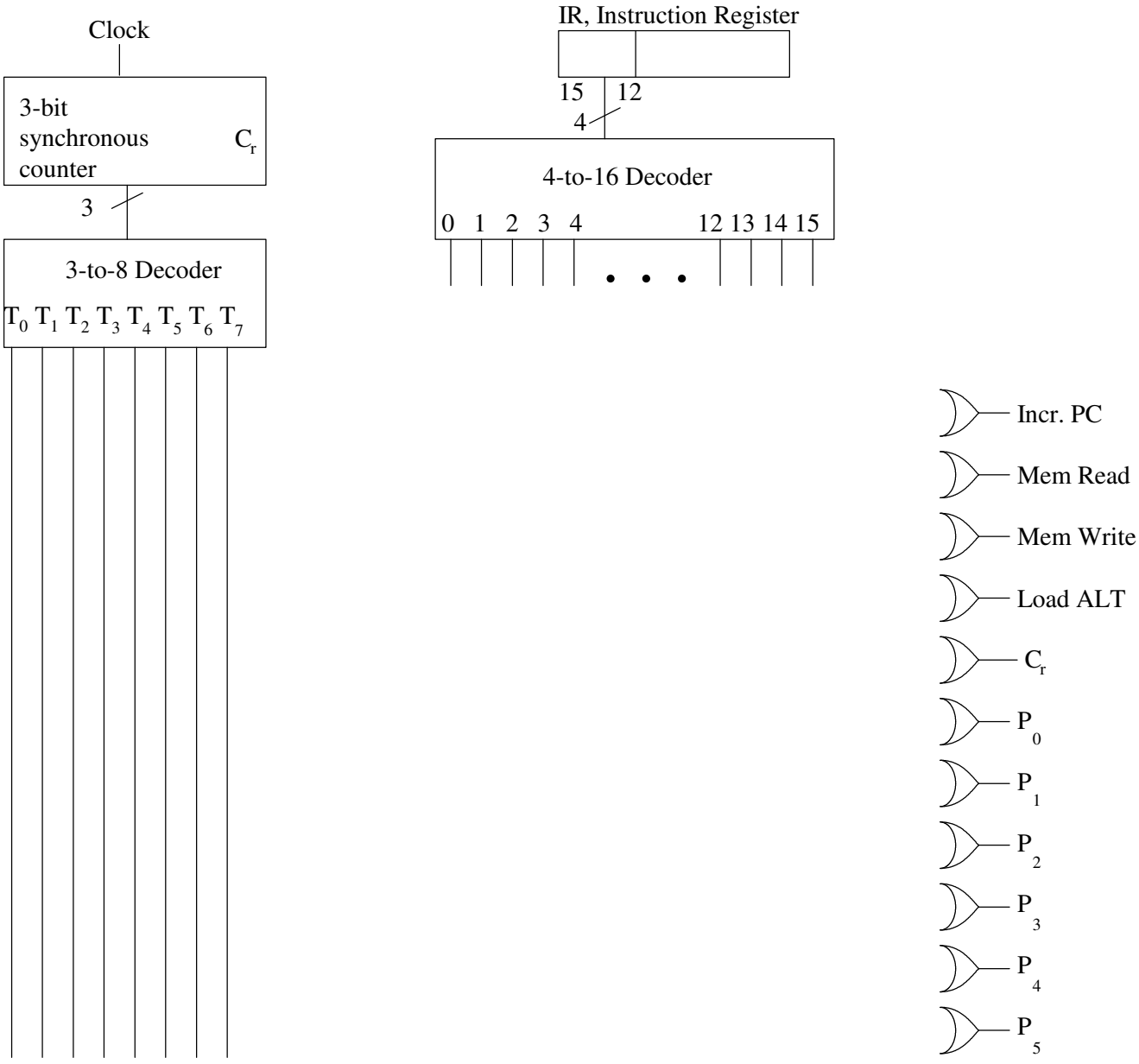
Address	**Label**	**Assembly Language**	**Machine Language** (in hex)
100_{16}	WHILE,	LOAD X	
101_{16}		SKIPCOND 000	
102_{16}		JUMP END_WHILE	
103_{16}		ADD SUM	
104_{16}		STORE SUM	
105_{16}		INPUT	
106_{16}		STORE X	
107_{16}		JUMP WHILE	
108_{16}	END_WHILE,	HALT	
109_{16}	X,	DEC 10	
$10A_{16}$	SUM,	DEC 0	

c) (10 points) Translate the above MARIE assembly language into high-level language "pseudo" code.

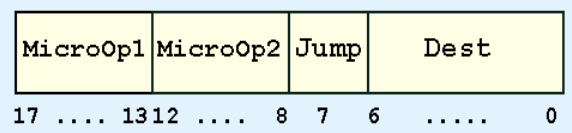
Question 4. (10 points) Which control signals should contain a "1" for each steps in the STORE X instruction?

Step	RTN	Step #	P_5	P_4	P_3	P_2	P_1	P_0	C_r	Incr PC	Mem Read	Mem Write	Load ALT
Fetch	MAR ← PC	T_0											
	MBR ← M[MAR]	T_1											
	IR ← MBR	T_2											
Decode IR[15-12]	PC ← PC + 1	T_3											
Get operand	MAR ← IR[11-0]	T_4											
Execute	MBR ← AC	T_5											
	M[MAR] ← MBR	T_6											
		T_7											

Question 5. (10 points) Draw the partial combinational logic of the hardwired control unit to handle the STORE X (opcode 2_{16}) instruction.



Question 6. Recall that the microprogrammed version of MARIE executes a fixed microprogam to perform the fetch-decode-execute cycle. The instruction format for the microinstructions look like:



`MicroOp1` encodes the type of register transfer notation (RTN) to perform (e.g., AC ← 0 is 00010_2)

`MicroOp2` is used only when decoding the instruction. It contains the binary codes for each instruction to allow comparison to the IR opcode. (Since the MARIE opcodes are only 4-bits long, only bits 12 - 9 are used and bit 8 is unused.

`Jump` is a single bit indicating that the value in the **Dest** field is a valid micro-address and should be placed in the microsequencer; if **`Jump`** is "FALSE" (0), then increment to the next microinstruction.

Table 4.8. Microoperation Codes and Corresponding MARIE RTN (p. 221)

MicroOp Code	Microoperation	MicroOp Code	Microoperation
00000	NOP	01100	MBR ← M[MAR]
00001	AC ← 0	01101	OutREG ← AC
00010	AC ← AC - MBR	01110	PC ← IR[11-0]
00011	AC ← AC + MBR	01111	PC ← MBR
00100	AC ← InREG	10000	PC ← PC + 1
00101	IR ← M[MAR]	10001	If AC = 00
00110	M[MAR] ← MBR	10010	If AC > 0
00111	MAR ← IR[11-0]	10011	If AC < 0
01000	MAR ← MBR	10100	If IR[11-10] = 00
01001	MAR ← PC	10101	If IR[11-10] = 01
01010	MAR ← X	10110	If IR[11-10] = 10
01011	MBR ← AC	10111	If IR[15-12] = MicroOp2[4-1]

We need to augment this table to include a few omitted microoperations and because we modified Figure 4.9 to remove the Memory from direct connection to the datapath. The following additional microoperations are used.

MicroOp Code	Microoperation
00101*	IR ← MBR
11000	AC ← MBR

* This microop code is being reused.

a) (10 points) Explain why a microprogrammed control unit is slower than a hardwired control unit?

b) (15 points) Extend the partial microprogram below to include microoperations to decode and implement the execution of the instructions: STORE X and CLEAR. **(Fill in only the bolded boxes)**

Revised Figure 4.21 Partial Microprogram

Part of Cycle	RTN (of MicroOp1)	μAddr	MicroOp1	MicroOp2	Jump	Dest
Fetch	MAR ← PC	0	01001	0000	0	0
	MBR ← M[MAR]	1	01100	0000	0	0
	IR ← MBR	2	00101	0000	0	0
	PC ← PC + 1	3	10000	0000	0	0
Decode	If ADD, Jump	4	10111	00110	1	
	If LOAD, Jump	5	10111	00010	1	
("Jump	If STORE, Jump	6	10111	00100	1	
Table")	If SKIPCOND, Jump	7	10111	10000	1	
	If SUBT, Jump	8	10111	01000	1	
	If JUMP, Jump	9	10111	10010	1	
	If ADDI, Jump	10	10111	10110	1	
	If CLEAR, Jump	11	10111	10100	1	
	If JNS, Jump	12	10111	00000	1	
	If JUMPI, Jump	13	10111	11000	1	
	If INPUT, Jump	14	10111	01010	1	
	If OUTPUT, Jump	15	10111	01100	1	
	If HALT, Jump	16	10111	01110	1	
Execute STORE X	MAR ← IR[11-0]	17				
	MBR ← AC	18				
	M[MAR] ← MBR	19				
Execute CLEAR	AC ← 0	20				
		21				
		22				
		23				
		24				