Guidelines for B.Sc. (APS Paper)
CSPT 303 –Computer System Architecture
For the Semester III 21st July 2011 to 19th Dec 2011.
Dated 13th July 2011

Theory	4 periods per Week + 1 period per week tutorial
Practical	**4 periods per week /per batch of 20 students**
Theory Examination –	3 Hours, (75 Marks + 25 Marks Internal Assessment as per university rules)
Practical Examination	4 hours, 50 marks (breakup given below)

Reference Book for Theory

Computer System Architecture by M.Morris Mano, Third edition, Prentice Hall of India
Same book Published by Pearson Education.

I : Introduction: Logic gates, Boolean algebra, combinational circuits, circuit simplification, flip-flops and sequential circuits, decoders, multiplexers, registers, shift registers, counters, memory units.

Chapter	Topic & Sections	Hours
1.	Digital Logic Circuits (complete)	6
2.	Digital Components (complete)	6

II & III : Data Representation and Computer Arithmetic (of integers) : Number systems, complements, fixed and floating-point representation, character representation. Addition, subtraction, magnitude comparison, multiplication, and division algorithms for integers.

Chapter	Topic & Sections	Hours
3.	Data representation (Upto 3.4)	6

IV: Central Processing Unit : Register organization, register transfer and micro-operations-register transfer language, arithmetic and logical micro-operations, stack organization, micro programmed control.

Chapter	Topic & Sections	Hours
4.	Register transfer and micro-operations (4.1 to 4.5)	4

.
V : Basic Computer Organization and Design : Computer registers, bus system, instruction set, timing and control, instruction cycle, memory reference, input-output and interrupt, design of basic computer

Chapter	Topic & Sections	Hours
5.	Basic Computer Organization and Design (Upto 5.9)	15

VI : Programming the Basic Computer : Instruction formats, addressing modes, instruction codes, machine language, assembly language, input output programming.

Chapter	Topic & Sections	Hours

8.	Programming the Basic Computer(8.4 & 8.5, 8.8 – CISC Characteristics only i.e. Pages 282 – 284 PHI, 300 – 302 Pearson)	5

VII : Input – Output Organization : Peripheral devices, I/O interface, asynchronous data transfer, priority interrupt, direct memory access, I/O processor, serial communication.

Chapter	Topic & Sections	Hours
11.	Input – Output Organization (11.1, 11.2, 11.3 (intro only) pg 391 PHI/ 409 Pearson, 11.5 (till Daisy Ch) pg 409 PHI 426 Pearson 11.6 (intro of DMA Transfer) pg 418 PHI/ 436 Pearson	8

Practical Examination

4 hours, 50 marks
Breakup of 50 marks:
20 marks for lab record
10 marks for viva
20 marks for practical exam (Further breakup of these marks and 'what to submit-softcopy/hardcopy', will be discussed in the next meeting)

Book to be followed for Practical / Lab Assignments

Introduction to Computing Systems From bis & gates to C & beyond

By Yale N.Patt & Sanjay J.Patel

Publisher: Tata McGraw-Hill

Second Edition (Indian Reprint-2011)

Price of the book will be Rs.350/- after discount.

This book has introduced a hypothetical computer called LC3 (Little Computer -3) with all characteristics of 8086, Motorola 68000, Pentium IV but it is not a real computer. Authors have developed a simulator for LC-3. The lab exercises can be done using the

simulator. Simulator is easy to install and run. There is tutorial on how to use the simulator for doing machine language exercises as well as assembly language exercises. For understanding the architecture of LC3 (memory, number of registers, instruction set etc.) Chapters from 4 to 10 must be understood and taught in practical classes.

Five machine language assignments are listed here. Another seven assembly language assignments will be sent to all teachers soon.

For grading each exercise, please give marks as follows:

80% for correct values; 15% for commenting; 5% for technique

MP1: Data Analysis

Introduction: At the machine level, programs are sequences of bits that are interpreted by the computer to perform some computation. In this Machine Problem you will construct a LC-3 program at the machine level to determine some characteristics of a value stored in memory.

Specifications:

1. Your assignment is to, first, determine if the value store in memory location x3030 is even or odd. You are to store x0001 in memory location x3031 if it is odd, x0000 if it is even.

2. Next, your program will count the number of 1s in the value at memory location x3030 and store that count in memory location x3032.

3. Your program can use the LC-3 register file and memory address space. You cannot overwrite the original value in memory location x3030.

4. Your code must be well-commented. Follow the commenting style of the code examples provided in class and in the textbook.

5. Your program must be written in LC-3 Machine Language and originate at x3000.

Testing: You should test your program thoroughly before handing in your final version. Developing a good testing methodology is essential for being a good programmer. For this

assignment, you should run your program several times for different values in memory location x3030 (you can do this using the simulator) and checking the output by hand.

Tools: You will need to use the LC - 3 simulator in order to execute and test the program you write for this MP. You might also need to use editor in order to enter your machine language program.

MP2: Machine Problem 2

There are two parts to this assignment. In each part you will be asked to write a different program. You will therefore submit two programs, one for Part 1 and another one for Part 2. Both programs should be written in LC-3 machine language. Read the instructions carefully and make sure you follow them.

Part 1 - A Program that shifts a bit pattern to the left by a certain amount (a number between 0 and 16, including 0 and 16)

Problem: In this part, you are asked to write a program in LC-3 machine language to shift a bit pattern some number of bits to the left and store the result in memory. The number of bits the bit pattern should be shifted is called the shift amount. Shift amount is a non-negative number between 0 and 16, inclusive (that is 0 and 16 are valid shift amounts).

Your program should assume that the initial bit pattern to be shifted is in memory location **x3100** and the shift amount is stored in memory location **x3101**. Using those values, your program should perform the left shift and store the result in memory location **x3102**. Your program should start at memory location **x3000**.

Example: If the memory location x3100 contains the bit pattern 1101000100001011 and memory location x3101 contains the value 0000000000000101 (decimal 5), then your program needs to shift 1101000100001011 5 bits to the left and store the bit pattern 0010000101100000 in memory location x3102. Note that when you shift a bit pattern *n* bits to the left, you fill the lowest *n* bits of the bit pattern with 0's.

Hint: What happens when you add a number to itself?

Simulator Hint: You can test your program by setting the values of memory locations x3100 and x3101 before you run your program on the LC-3 simulator. On UNIX machines (Sun, Linux) you can do this by using the "Set Values" option on the menubar

and selecting the "Set Register or Memory" option. On Windows machines, you can click on "Simulate" in menu bar and select "Set Value". Instead, you can just press F4 and the "Set Value" dialog box will pop up.

Part 2 - A Program that rotates a bit pattern to the left by a certain amount (a number between 0 and 16, including 0 and 16)

Problem: Now that you have done the left shift, we'll ask you to do something more exciting: rotating a bit pattern. Your task in this part is to write a program in LC-3 machine language to rotate a bit pattern some number of bits to the left and store the result in memory. The rotate amount (number of bits you rotate the bit pattern to the left) is a non-negative integer between 0 and 16, inclusive. Your program should assume that the initial bit pattern is in memory location **x3100** and the rotate amount is stored in memory location **x3101**. Using those values, your program should perform the left rotation and store the result in memory location **x3102**. Your program should start at memory location **x3000**.

Example: If the memory location x3100 contains the bit pattern 1101000100001011 and memory location x3101 contains the value 0000000000000101 (decimal 5), then your program needs to rotate 1101000100001011 5 bits to the left and store the bit pattern 0010000101111010 in memory location x3102. Note that when you rotate a bit pattern *n* bits to the left, it is just like a left shift except that top n bits before the shift end up as the bottom *n* bits.

Notes and Suggestions:
- The first line of your programs must specify the memory address of the first instruction of your program. LC-3 simulator will place your program starting at that address. For this assignment, you should place your program starting at **x3000** (the first line of your program should contain the bit pattern 0011000000000000).
- If you are using a Windows machine, use the LC3Edit program to type in your programs.

MP3: Machine Problem 3

1. Background

Integer division is the kind we all learned in third grade, where we get a quotient and a remainder (i.e. 11/4 = 2 , with a remainder of 3). Computers perform similar division on integers, but the remainder is discarded.
This is called integer division (IDIV) (i.e. 11 IDIV 4 = 2).

X modulo Y is defined as the remainder of X IDIV Y.
This is commonly seen as X % Y (i.e. 11 % 4 = 3).

2. Assignment

Your will implement two functions, modulo and integer divide.
Y = A / B
Z = A % B

In high level statement above, Y and Z are destinations in memory, and A and B are sources in memory. Recall that the LC-3 is a Load/Store machine. Before operating on the values stored in A and B, we must first load them into registers.
After computing the integer quotient and the modulus, we must store these results into memory.
The source operand A is stored in location 0x3100 in memory.
The source operand B is stored in location 0x3101.
The result Y should be stored in location 0x3102.
The result Z should be stored in location 0x3103.
Your program should start at location 0x3000.
You should make the following assumptions:
A > 0
B > 0

3. Initial Values

You should write your program assuming that there are already values stored in memory at locations 0x3100 and 0x3101. When using the LC-3 simulator to test your program, you should manually load test values into these memory locations before running your program.

4. Format

Your program must be a text file of binary characters. Each line contains a 16 character string of ones and zeros followed by a carriage return.
The first line will give the starting address of the sequence of memory locations containing the program. Each line may have a comment, written as a string of ASCII codes, or not, as you wish. Comments are useful for helping you understand what the instruction does six months from now. The comment is separated from the 16 0's and 1's by a semi-colon. That is, the semi-colon denotes the end of what is actually stored in the memory location. Everything beyond that is discarded before the 16 bits are stored in the memory location.

Example: Suppose you were being asked to submit the program that was the
subject of our multiply routine. The first two lines might be:
0011000000000000 ; The program will be stored, starting with location x3000.
0101011011100000 ; Clear R3, which will be used to keep the intermediate results.
It is not necessary to have a comment next to every instruction.

MP4: Machine Problem 4

Reversing a bit pattern in memory

Problem Statement

You are asked to write a program in LC-3 machine language that takes a word stored at memory location **x3100** and reverse all the bits. You will store your result in memory location **x3300**.

Example: If the bit pattern 1010100101001011 is stored at memory location **x3100**, your program should write the bit pattern 1101001010010101 into memory location **x3300**.

Hint: As you know that, ADDing a bit pattern to itself effectively shifts the bit pattern to the left by one bit position. If I have a number X and I AND it with the bit pattern 0000000000000001, what do I get? What do I get when I AND the same number with 0000000000000010?

Notes and Suggestions

- The first line of your programs must specify the memory address of the first instruction of your program. LC-3 simulator will place your program starting at that address. For this assignment, you should place your program starting at **x3000** (the first line of your program should contain the bit pattern
0011000000000000).
- A good strategy is to spend time thinking about your program before sitting at the computer. You'll minimize the time spent programming and probably save yourself frustration when it comes time to debug your program. Our experience tells us that programs rarely work on the first shot. Be prepared to debug your program!
- Before running your program, you should deposit a value into memory location **x3100** using the "Set Register or Memory" or "Set Values" menu in the simulator.
If your program functions correctly, the number at the location **x3300** will be the reverse of the bit pattern stored at location **x3100**. You should verify that your program works correctly by testing it with several different values.
- If you are working on a Windows machine, use the LC3Edit program to type in your programs.

MP5: Machine Problem 5

1. Background:

There are many different ways of interpreting bits in memory. We are already familiar with the idea of a single LC-3 memory location containing either an LC-3 instruction or a 16-bit 2's complement value. Consider a program that deals with small values in the range -128 to +127. We would only need 8 bits to represent each number, so to conserve memory we could pack two such numbers in a single LC-3 location, hence the name "packed," as it is often called.

2. Assignment

Write a program to calculate the sum of all the 8-bit 2's complement values in a packed array of N elements. With N elements in the packed array, we require N/2 memory locations.
For this assignment, we will assume N is even.
(You might on your own think of what modifications you would have to make if N were odd.)

Your SUM will be stored in one word of memory as a 16-bit 2's complement number.
For each element in the array, bits [15:8] contain the first 8-bit value, and bits [7:0] contain the second value.
Your program should begin at x3000.
The size of the packed array, N, is stored in location x3101. We will restrict the value of N to be less than or equal to 600 (decimal). The packed array itself is stored beginning at location x3102. Your program should store the result SUM in location x3100.

3. Initial Values
Your program should assume that there are already values stored in memory at locations x3101 and x3102 through x3102 + (N/2 - 1). When using the LC-2 simulator to test your program, you should manually load test values into these memory locations before running your program.

Example: Consider the following array, where x3101 contains N, the number of elements in the array:
x3101 | x0006
x3102 | x7F03
x3103 | xFE7A
x3104 | x1B40
Your program should result in x0155 stored in location x3100.

4. Format
Your program must be a text file of binary characters (i.e., a .bin file). Each line contains a 16-character string of ones and zeros. You are encouraged to add comments when useful and appropriate.

Following Lab assignments will be distributed in next 2 to 3 weeks.

Assembly Language Programming Assignment 1

Assembly Language Programming Assignment 2

Assembly Language Programming Assignment 3

Assembly Language Programming Assignment 4

Assembly Language Programming Assignment 5

Assembly Language Programming Assignment 6

Assembly Language Programming Assignment 7