Capital University of Science and Technology

Department of Computer Science

CS2523 - Computer Organization and Assembly Language

Course Title:	Computer Organization and Assembly Language (CS2523)
Pre-requisite(s):	Introduction to Programming (CS1133)
Credit Hours:	3
Instructor(s):	
Text Book(s):	Assembly Programming using intel 8088 by Belal Hashmi
Web Reference:	• https://jbwyatt.com/253/emu/asm_tutorial_01.html

Course Introduction:

The main objective of this course is to introduce the organization of computer systems and usage of assembly language for optimization and control. Emphasis should be given to expose the low-level logic employed for problem solving while using assembly language as a tool. The course will be delivered with the consideration of the one of the most famous microprocessors of Intel. i.e., iAPX88 and our assembly language programs will be implemented with this architecture. A very important aspect in assembly will be looking at the working of a hidden but very important aspect of a program which is stack and the clear understanding of this topic will help to understand many topics in the Operating System Course. At the end of the course the students should be capable of writing moderately complex assembly language subroutines and interfacing them to any high-level language.

Course Objectives:

The main objective of this course is to introduce the organization of computer systems and usage of assembly language for optimization and control. Emphasis is given to expose the low-level logic employed for problem solving while using assembly language as a tool. At the end of the course the students should be able to Identify the major components of computer architecture, and explain their purposes and interactions. Simulate the internal representation of data, and show how data is stored and accessed in memory. Explain the relationships between hardware architecture and its instruction set, and simulate micro-programs. Explain the Instruction Execution Cycle.

Explain the differences and relationships among high-level, assembly, and machine languages. Write well-modularized computer programs in an assembly language, implementing decision, repetition, and procedure structures. Write moderately complex assembly language subroutines and interfacing them to any high-level language. Use a debugger, and explain register contents. Simulate the system stack as it is used for procedure calls and parameter passing. Explain how editors, assemblers, linkers, and operating systems enable computer programming. Explainvarious mechanisms for implementing parallelism in hardware/software.

Course Learning Outcomes (CLOs):

At the end of this course, the students should be able to:

CLO1: Define concepts in the design of microprocessor as state machine and designing its data path and its controller. [C1- Remembering]
CLO2: Describe how the basic units of the Intel 8088 architecture work together to represent Integer Numbers, Floating Numbers and register representation inside the microprocessor. [C2- Understanding]
CLO3: Implement assembly programs of intermediate complexity using the intel 8088

architecture. The student should also be able to convert intermediate complexity program in high level language into assembly code. [C3- Applying]

CLOs – PLOs Mapping:

	CLO:1	**CLO:2**	**CLO:3**
PLO:1 (Academic Education)			
PLO:2 (Knowledge for Solving Computing Problems)	√	√	
PLO:3 (Problem Analysis)			
PLO:4 (Design/Development of Solutions)			√
PLO:5 (Modern Tool Usage)			

Course Contents:

Week	Contents
1	Introduction to computer organization & architecture, general introduction of the course Assembly and Machine language, compiler and assembler, why learn assembly, comparison of assembly and high-level language.
2	Programmer's view of a computer system, (App. Program, assembly language, operating system, instruction set architecture, microarchitecture, digital logic) Data representation, Binary numbers, converting binary to decimal, hexadecimal integers, hexadecimal to binary, decimal to hexadecimal, Integer storage sizes
3	Binary addition, Hexadecimal addition, Signed Integers (sign-magnitude, Biased representation) Signed Integers (1's complement, 2's complement), Exercises, Dis advantages of signed magnitude.
4	Excess representation, floating point representation, Summary of number representation 2's complement of hexadecimal ranges of signed integers, carry and overflow, character storage, Printable ASCII Codes, control characters.
5	Introduction to the IAPX88 architecture, registers (General Purpose, Pointer register, Segment register) The 88 flag register and interruption of flags (ZF, CF, SF, OF, AF, PF, IF), Instruction groups (Data Movement, arithmetic and logic, control, special instructions)
6	First assembly program, tools debugger, linker, assembler, how to assemble and run the assembly program Segmented memory model in IAPX88, using the debugger to explain the segmented memory model.
7	Discussion on the Command file, List file, relative address and physical address. Offset, segment, physical address calculation, paragraph boundaries, overlapping segments
8	Exercise questions from the notes Data declaration, difference between direct and in direct addressing, assembly programs showing different ways to represent data in the memory. Register addressing, memory addressing register to memory, memory to register, register to register, register to constant.
Mid-Term Exam	

9	Segments, default segments in direct and indirect modes, [base + offset + index] method to calculate the effective address and using it with the associated segment to calculate the physical address. Conditional Jumps, Flags and their role in decision making, Adding logic in assembly programs.
10	Short Jump, Far Jump, Near Jump, Unconditional jumps Sorting Algorithm i.e., Bubble sort using flags and conditional jumps.
11	Exercises from the 3rd chap to solve conditional jumps and unconditional jumps related problems. Bit manipulation, multiplication algorithm, shift left, shift right, rotate left , rotate right, rotate through carry left & right
12	Multiplication assembly program and limitation Processor as a state machine
13	Machine Instruction Cycle, Register Transfer Level Activity of machine instruction cycle, timing diagram of machine instruction cycle Data Path architecture, data path architecture of the machine instruction cycle
14	Data path control architecture hardwired; firmware approach Data path control Continued
15	Master Slave JK Flip Flop system firm ware approach Master Slave JK Fkip Flop System hardwired approach
16	Operating system support

Grading Policy:

S.No	**Grading**	**% of Total Marks**
1	Assignments	20
2	Quizzes	20
4	Mid-term Exam	20
5	Final Exam	40
	Total	**100**