CSE331L_1 – Introduction to assembly language.

Introduction:

In this session, you will be introduced to assembly language programming and to the emu8086 emulator software. emu8086 will be used as both an editor and as an assembler for all your assembly language programming.

steps required to run an assembly program:

1. Write the necessary assembly source code.
2. Save the assembly source code.
3. Compile/ Assemble source code to create machine code.
4. Emulate / Run the machine code.

First, familiarize yourself with the software before you begin to write any code. Follow in-class instructions regarding the layout of emu8086.

Micro controllers vs micro processors

- A microprocessors is a CPU on a single chip.
- If a microporocessors, Its associated support circuitry, memory and peripheral I/O components are implemented on a single chip. It is a microcontroller.

Features of 8086

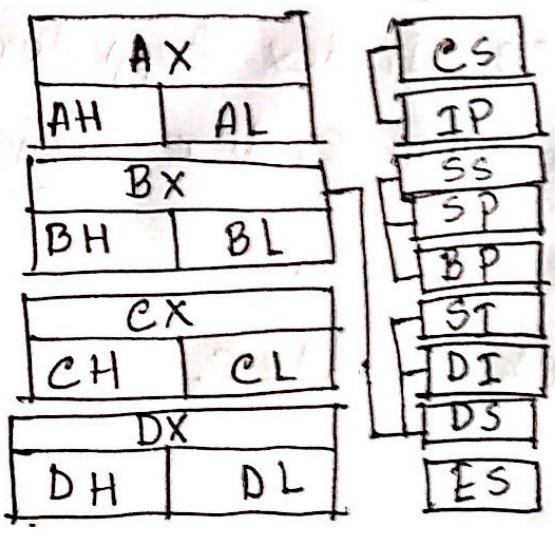
- 8086 is a 16 bit processors. Its ALU, registers work with 16 bit binary word.
- 8086 has a 16 bit data bus. It can read or write data to a memory/port either 16 bits or 8 bits at a time.
- 8086 has a 20 bits address bus which means, it can address up to 2^{20} = 1 MB memory location.

Register – Registor – Registrar.

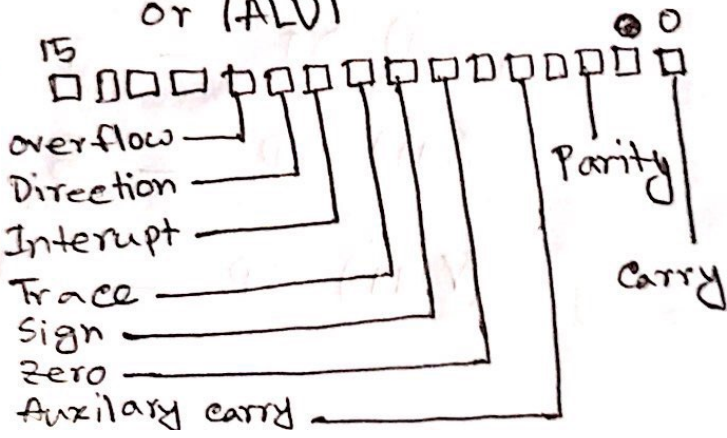
- Both ALU and FPU have a very small amount of super-fast private memory placed right next to them for their exclusive use. These are called registers.
- ALU and FPU stores intermediate and final results from their calculations in these registers.
- Processed data goes back to the data cache then to the main memory from these registers.

Inside the CPU: Get to know the various registers-

Central processing unit (CPU)



Arithmetic and logical Unit or (ALU)



Registers are basically CPU's own internal memory. They are used among other purpose to store temporary data while performing calculations. Let's look at each one in details.

General purpose Registers (GPR):

The 8086 has 8 general purpose registers. Each registers has it's own name:

(i) AX - The ACCUMULATOR REGISTER (Divided into AH and AL).

(ii) BX - The BASE REGISTER (Divide into BH and BL).

(iii) CX - The COUNT REGISTER (Divide into CH and CL).

(iv) DX - The DATA REGISTER (Divide into DH and DL).

(v) SI - Source Index Register.

(vi) DI - Destination Index Register.

(vii) BP - Base Pointer.

(viii) SP - Stack Pointer.

Despite the name of a register, It's a programmer who determines the uses of each general-purpose registers. The main purpose of a registers to keep a number or (variable). The size of these registers is 16 bits.

4- General Purpose registers (AX, BX, CX, DX). are made of 2 8 bits seperate registers. for example if AX is 0011000000111001b the AH = 00110000b and AL is 00111001b. Therefore, when you modify any of the 8 bits registers 16-bits registers are also updated, and vice-versa. The same is for others 3 registers. "H" is for high and "L" is for low part.

Since registers are located inside the CPU, they are much faster than a memory. Accessing a memory location requires the use of a system bus, so

it takes much longer. Accessing the data in a registers usually takes no time. Therefore, you should try to keep variables in registers. Register sets are very small and most registers has special purposes which limit their uses as variables, but they are still an excellent place to store temporary data of calculations.

Segment registers:

CS - points at the segment containing the current program.

DS - generally points at the segments where variables are defined.

ES - extra segment registers, it's upto a coder to define its usage.

SS - point at the segment containing the stack.

Although it is possible to store any data in segment registers. This is never

a good idea. The segment registers have a very special purpose - pointing at accessible block of memory. This will be discuss further in upcoming classes.

Special Purpose registers:

- IP - Instruction pointers. Points to the next location of instruction in the memory.
- Flag Register - Determines the current State of the microprocessors. Modified automatically by the CPU after some mathematical operations, determines certains types of results and determines how transfer control to a program.

Write your first assembly code

In order to write programs in assembly language, you will need to familiarize yourself with most, if not all, of the instructions in the 8086 - instruction set.

This class will introduce two instructions and will serve as the basis of your first assembly program.

The following table shows the instruction name, the syntax of its use, and its descriptions. The operands heading refers to the type of operands that can be used with the instructions along with their proper order:

- REG: Any valid register.
- Memory: Reffering to memory location in RAM.
- Immediate: Using direct values.

Instructions	Operands	Deseriptions
MOV	REG, memory memory, REG REG, REG memory, immediate REG, immediate	Copy operand2 to operand 1 • The MOV instruction cannot set the value of the CS and IP registers. • Copy value of one segment register to another

Instructions	Operands	Descriptions
		segment register (should copy to general ~~purpose~~ register first). • Copy an intermediate value to segment register. (should copy to general register first. Algorithm: operand 1 = Operand
ADD	REG, memory memory, REG REG, REG memory, immediate REG, immediate	Adds two numbers. Algorithm: Operand 1 = Operand 1 + Operand 2.