Department of
Computer Science and Engineering

Title: Introduction to assembly language and EMU 8086 instruction set

Microprocessors and microcontrollers

CSE 304

Green University of Bangladesh

1 Objective(s)

- To have insight on assembly language.
- To understand EMU8086 instruction sets.

2 Problem analysis

Computers basically work binary digits of 0 and 1. Central processing units only understand commands in machine language which only consists bit string of 0 or 1. However, operating a computer using machine language is an arduous task for users. That's why assembly language provides a flexible way to code. In assembly language, symbolic names are used to represent operations, registers and memory locations. Though assembly language helps to make computer commands a little bit understandable for human users, but it must be converted into machine language for CPU processing. Assembler is used to convert assembly language segments to machine language.

EMU 8086 is microprocessor 8086 emulator. It has a built in 8086 assembler. The word "Emulate" means to imitate or copy something. EMU 8086 is designed to emulate hardware. It runs the program in a step by step mode showing registers, memory, stack, variable etc just like a microprocessor.

Assembly Language	Machine Code
SUB AX,BX	001010111000011
MOV CX,AX	100010111001000
MOV DX,0	10111010000000000000000

Figure 1: Instruction Set (Machine Instruction Set vs Assembly Instruction set)

2.1 Architecture of EMU8086

A low-level programming language is assembly language. In order to grasp anything, you must first learn about computer structure. The system bus connects the various components of a computer, according to the simple computer model in Figure 2. The **CPU** is the computer's heart; most computations take place within the CPU. **RAM** is a memory location where programs are loaded in order to be executed. In this manual we are focusing on the central processing unit of EMU8086, shown in Figure 3.

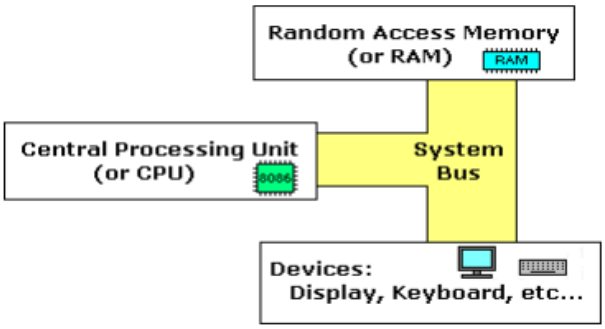

Figure 2: The Simple Computer Model

2.1.1 Central Processing Unit

In the 8086 Microprocessor, the registers are categorized into mainly four types:

- General Purpose Registers
- Segment Registers

- Pointers and Index Registers
- Flag or Status Register

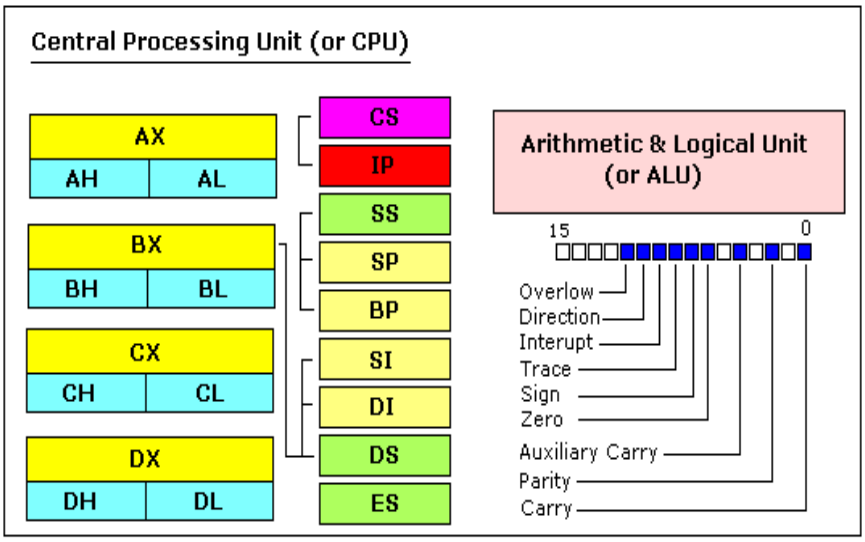

Figure 3: Central Processing Unit of EMU8086

2.1.2 General Purpose Registers

General purpose registers are used to store temporary data within the microprocessor. There are 8 general purpose registers in 8086 microprocessor. Descriptions of general purpose registers are given in Table 2.

Table 1: General Purpose Registers

Registers	Description
AX	This is accumulator register. It is a 16 bit register.
BX	It is known as base register. Used for store the value of offset.
CX	Also known as counter register. Used for loop and rotation.
DX	Known as data register. Used for manipulating input output port address.
SP	Stack Pointer.
BP	Base Pointer
SI	Source Index Register.
DI	Destination Index register.

2.1.3 Segment Registers

There are 4 segment registers in 8086 Microprocessor and each of them is of 16 bit. The code and instructions are stored inside these different segments.

- Code Segment (CS) Register: The user cannot modify the content of these registers. Only the microprocessor's compiler can do this.
- Data Segment (DS) Register: The user can modify the content of the data segment.
- Stack Segment (SS) Registers: The SS is used to store the information about the memory segment. The operations of the SS are mainly Push and Pop.
- Extra Segment (ES) Register: By default, the control of the compiler remains in the DS where the user can add and modify the instructions. If there is less space in that segment, then ES is used. ES is also used for copying purpose.

2.1.4 Pointers and Index Registers

The pointers will always store some address or memory location. In 8086 Microprocessor, they usually store the offset through which the actual address is calculated.

- Instruction Pointer (IP): The instruction pointer usually stores the address of the next instruction that is to be executed. Apart from this, it also acts as an offset for CS register.
- Base Pointer (BP): The Base pointer stores the base address of the memory. Also, it acts as an offset for Stack Segment (SS).
- Stack Pointer (SP): The Stack Pointer Points at the current top value of the Stack. Like the BP, it also acts as an offset to the Stack Segment (SS).
- The indexes are used with the extra segment and they usually are used for copying the contents of a particular block of memory to a new location.
- Source Index (SI): It stores the offset address of the source.
- Destination Index (DI): It stores the offset address of the Destination.

2.1.5 Flag or Status Register

The Flag or Status register is a 16-bit register which contains 9 flags, and the remaining 7 bits are idle in this register. These flags tell about the status of the processor after any arithmetic or logical operation. IF the flag value is 1, the flag is set, and if it is 0, it is said to be reset.

2.2 Instruction Set of EMU8086

Instructions of 8086 microprocessor is based on the functions they perform. Here is given a Figure 4 in the following page which shows various instructions that we will use in our next labs to solve different problems in assembly language.

	CMPSB				MOV		
AAA	CMPSW	JAE	JNBE	JPO	MOVSB	RCR	SCASB
AAD	CWD	JB	JNC	JS	MOVSW	REP	SCASW
AAM	DAA	JBE	JNE	JZ	MUL	REPE	SHL
AAS	DAS	JC	JNG	LAHF	NEG	REPNE	SHR
ADC	DEC	JCXZ	JNGE	LDS	NOP	REPNZ	STC
ADD	DIV	JE	JNL	LEA	NOT	REPZ	STD
AND	HLT	JG	JNLE	LES	OR	RET	STI
CALL	IDIV	JGE	JNO	LODSB	OUT	RETF	STOSB
CBW	IMUL	JL	JNP	LODSW	POP	ROL	STOSW
CLC	IN	JLE	JNS	LOOP	POPA	ROR	SUB
CLD	INC	JMP	JNZ	LOOPE	POPF	SAHF	TEST
CLI	INT	JNA	JO	LOOPNE	PUSH	SAL	XCHG
CMC	INTO	JNAE	JP	LOOPNZ	PUSHA	SAR	XLATB
CMP	IRET	JNB	JPE	LOOPZ	PUSHF	SBB	XOR
	JA				RCL		

Figure 4: Complete 8086 instruction set

2.3 Operand types

Normally instructions are applied on operands, so it is necessary to know which one can be used as operand are discussed as follows:
REG: AX, BX, CX, DX, AH, AL, BL, BH, CH, CL, DH, DL, DI, SI, BP, SP.
SREG: DS, ES, SS, and only as second operand: CS.
memory: [BX], [BX+SI+7], variable, etc..
immediate: 5, -24, 3Fh, 10001101b, etc...

2.4 Data Transfer Instructions

MOV is one of the frequently used instruction to perform data transfer operations, Different data transfer instructions using **MOV** are shown in Table 2.

Table 2: Data Transfer Instructions

Data Transfer Mode	Examples
Registers(Direct)	Move contents of BX register to AX register Example: MOV AX,BX
Direct	Move contents of the variable labeled COUNT to AX register. Example: MOV AX, COUNT
Immediate	Load CX register with the value 240d Example: MOV CX, 00F0H MOV CX, 240
Memory	Load CX register with the value at address 240. Example: MOV CX, [0F0H]
Registers(Indirect)	Move contents of AL register to memory location in BX. Example: MOV [BX], AL

3 Example of MOV and ADD command in EMU8086

```
1
2 MOV AX,1234H
3 ADD BX,4H
```

4 Input/Output

Here the output for the written two commands MOV AX,1234H and ADD BX,4H has been shown(Figure 2). This output analysis would help students to gain more insights on 8086 working principles.

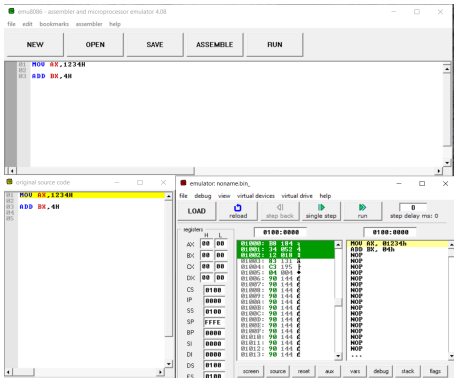

Figure 5: Output

5 Discussion & Conclusion

Based on the focused objective(s) to understand about the machine and assembly language,EMU 8086 and the additional lab exercise made me more confident towards the fulfilment of the objectives(s).

6 Lab Task (Please implement yourself and show the output to the instructor)

1. Install EMU8086 on your computer.
2. Run simple MOV and ADD commands. (Example: MOV AX,40H).

7 Lab Exercise (Submit as a report)

- Discuss about advantage and disadvantages of assembly language compared to high level languages.
- Put 100H to register BX, Then move the contents of this register to AX register.
- After that add 10H to the contents of AX register.

8 Policy

Copying from internet, classmate, seniors, or from any other source is strongly prohibited. 100% marks will be *deducted* if any such copying is detected.