Department of
Computer Science and Engineering

Title: Introduction to assembly language and EMU8086 instruction set

Microprocessors and microcontrollers

CSE 304

Green University of Bangladesh

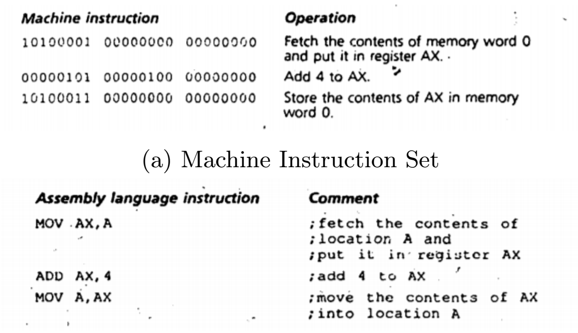
1 Objective(s)

- 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.



Machine instruction	**Operation**
10100001 00000000 00000000	Fetch the contents of memory word 0 and put it in register AX.
00000101 00000100 00000000	Add 4 to AX.
10100011 00000000 00000000	Store the contents of AX in memory word 0.

(a) Machine Instruction Set

Assembly language instruction	**Comment**
MOV AX,A	;fetch the contents of ;location A and ;put it in register AX
ADD AX,4	;add 4 to AX
MOV A,AX	;move the contents of AX ;into location A

(b) Assembly Instruction set

Figure 1: Instruction Set

2.1 Instruction Set of EMU8086

Instructions of 8086 microprocessor is based on the functions they perform. Here is given a Table 1 in the following page which shows various instructions that we will use in our next labs to solve different problems in assembly language.

2.2 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.

2.3 Data Transfer Instructions

Different data transfer instructions are shown in Table 3.

3 Example of MOV and ADD command in EMU8086

```

MOV AX,1234H
ADD BX,4H
```

Table 1: Instruction set of 8086

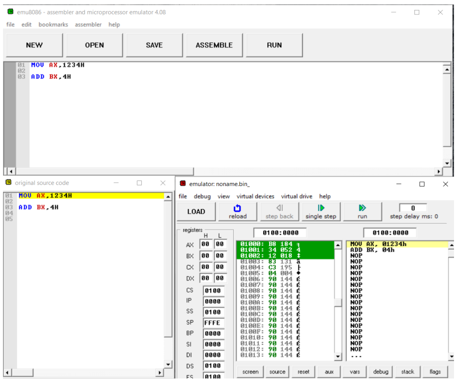
Instruction	**Description**
MOV	Moves data.
LEA	Loads offset address into specific register.
PUSH	Moves the content of a memory or register to the top of the stack.
POP	Pop out the top of the stack and puts the content in memory or register.
POPF	Pop content from the top of the stack and keeps them in the flag register.
AND	Performs bit by bit logical AND operation of two operands.
OR	Performs bit by bit logical OR operation of two operands.
XOR	Performs bit by bit logical XOR operation of two operands
NOT	Performs bit by bit logical NOT operation.
ADD/SUB/MUL/DIV	Performs addition, subtraction, multiplication and division respectively.
INC/DEC	Increment/decrement a register or memory by 1.
RCL/RCR/ROL/ROR	Rotate all the bits of an operand left/right using carry flag or without carry flag.
JAE/JNB/JNC	Jump if above, not below, equal or no carry i.e. when CF = 0
JA /JNBE	Jump if above, not below, or equal i.e. when CF and ZF = 0
JB/JNAE/JC	Jump if below, not above, equal or carry i.e. when CF = 0

Table 2: 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.

Table 3: 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



```
MOV AX,1234H

ADD BX,4H
```

```
     H  L
AX  00 00
BX  00 00
CX  00 00
DX  00 00
CS  0100
IP  0000
SS  0100
SP  FFFE
BP  0000
SI  0000
DI  0000
DS  0100
ES  0100
```

```
0100:0000

01000: B8 184 ╕
01001: 34 052 4
01002: 12 018 ↕
01003: 83 131 â
01004: C3 195 ├
01005: 04 004 ♦
01006: 90 144 É
01007: 90 144 É
01008: 90 144 É
01009: 90 144 É
0100A: 90 144 É
0100B: 90 144 É
0100C: 90 144 É
0100D: 90 144 É
0100E: 90 144 É
0100F: 90 144 É
01010: 90 144 É
01011: 90 144 É
01012: 90 144 É
01013: 90 144 É
```

```
0100:0000

MOV AX, 01234h
ADD BX, 04h
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
```

Figure 2: Output

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.

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.