Chapter 2

1. **Define Assembling. Explain the merits and demerits of Assembly Language Programming.**

- Assembling is the process of converting the assembly language to the low level i.e converting the mnemonics to their machine code. In other words translating mnemonics entered by ASCII keyboard into the corresponding binary machine codes of the microprocessor.
- Assembling is done with a assembler. A Processor understands only machine language instructions, which are strings of 1's and 0's. So program in assembly language must be converted to low level. Each microprocessor has its own assembler because mnemonics and machine codes are to the microprocessor used.

Merits:

i. The symbolic programming of Assembly Language is easier to understand and saves a lot of time and effort of the programmer.
ii. It is easier to correct errors and modify program instructions.
iii. Provides the better control on the hardware.
iv. Assembly Language has the same efficiency of execution as the machine level language.
v. The Program are memory efficient.

Demerits:

i. To write programs in the assembly language the internal structure of the microprocessor must be known.
ii. Assembly language varies by microprocessor type. A program written for the microprocessor may not work for other microprocessor.
iii. Programming in assembly language is more difficult and time consuming then high level language.
iv. Difficult to remember large number of mnemonics.
v. Long programs written in such languages cannot be executed on small sized computers.

2. **Explain the 8085 Programming model.**

- The 8085 programming model includes six registers, one accumulator, and one flag register, Figure. In addition, it has two 16-bit registers: the stack pointer and the program counter. They are described briefly as follows.

Registers

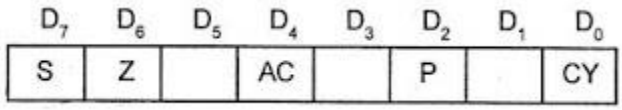
The 8085 has six general-purpose registers to store 8-bit data; these are identified as B,C,D,E,H, and L as shown in the figure. They can be combined as register pairs - BC, DE, and HL - to perform some 16-bit operations. The programmer can use these registers to store or copy data into the registers by using data copy instructions.

Accumulator

The accumulator is an 8-bit register that is a part of arithmetic/logic unit (ALU). This register is used to store 8-bit data and to perform arithmetic and logical operations. The result of an operation is stored in the accumulator. The accumulator is also identified as register A.

Flags

The ALU includes five flip-flops, which are set or reset after an operation according to data conditions of the result in the accumulator and other registers. They are called Zero(Z), Carry (CY), Sign (S), Parity (P), and Auxiliary Carry (AC) flags; their bit positions in the flag register are shown in the Figure below. The most commonly used flags are Zero, Carry, and Sign. The microprocessor uses these flags to test data conditions.



8085 has five flag registers:-

- ***Sign Flag (S):*** Sets or Resets based on the result stored in the accumulator. If the result stored is positive, the flag resets else if the result stored is negative the flag is set.
- ***Zero Flag (Z):*** Sets or Resets based on the result stored in the accumulator. If the result stored is zero the flag is set else it is reset.
- ***Auxiliary Carry Flag (AC):*** This flag is set if there is a carry from low nibble (lowest nibble (upper 4 bits) or a borrow from high nibble to low nibble, in the low order 8-bit portion of an addition or subtraction operation.
- ***Parity Flag (P):*** This flag is set if there is even parity else it resets.
- ***Carry Flag (CY):*** This flag is set if there is a carry bit else it resets..

These flags have critical importance in the decision-making process of the microprocessor. The conditions (set or reset) of the flags are tested through the software instructions. For example, the instruction JC (Jump on Carry) is implemented to change the sequence of a program when CY flag is set. The thorough understanding of flag is essential in writing assembly language programs.

Program Counter (PC)

This 16-bit register deals with sequencing the execution of instructions. This register is a memory pointer. Memory locations have 16-bit addresses, and that is why this is a 16-bit register.

The microprocessor uses this register to sequence the execution of the instructions. The function of the program counter is to point to the memory address from which the next byte is to be fetched. When a byte (machine code) is being fetched, the program counter is incremented by one to point to the next memory location

Stack Pointer (SP)

The stack pointer is also a 16-bit register used as a memory pointer. It points to a memory location in R/W memory, called the stack. The beginning of the stack is defined by loading 16-bit address in the stack pointer.

This programming model will be used in subsequent tutorials to examine how these registers are affected after the execution of an instruction.

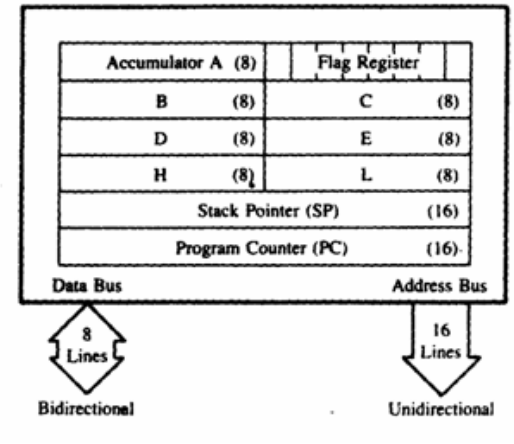


Figure 1 : Programming Model For 8085 microprocessor

3. Classify the 8085 Instruction Set with example.

The 8085 Instruction Classification can be categorized into five different groups based on the nature of function of the instructions.

i. Data transfer operations
ii. Arithmetic operations
iii. Logical operations
iv. Branch operations and
v. Stack, Input/output and Machine control operations

Data transfer operations

The data transfer instructions load given data into register, copy data from register to register, copy data from register to memory location, and vice versa. In other words we can say that data transfer instructions copy data from source to destination. Source can be data or contents of register or contents of memory location whereas destination can be register or memory location. These instructions do not affect the flag register of the process. Some of the instruction are :

MOV Rd, Rs(Move Data; Move the content of the one register to another
MVI Rd, data. (Move immediate data to register).
LXI Rp,16 bit data . (Load register pair immediate).
LDA address. (Load Accumulator direct).
STA address. (Store accumulator direct).
LHLD address. (Load H-L pair direct
SHLD address. (Store H-L pair direct)
LDAX rp. (LOAD accumulator indirect)
STAX rp. (Store accumulator indirect)
XCHG. (Exchange the contents of H-L with D-E pair)
IN port-address. (Input to accumulator from I/O port)
OUT port-address (Output from accumulator to I/O port)
PUSH Rp (Push the content of register pair to stack)

POP Rp (Pop the content of register pair, which was saved, from the stack))
XTHL (Exchange stack-top with H-L)
SPHL (Move the contents of H-L pair to stack pointer)

Arithmetic Instruction

The arithmetic instructions provided by 8085 perform addition, subtraction, increment and decrement operations.

ADD R. (Add register to accumulator)
ADD M. (Add memory to accumulator)
ADC R. (Add register with carry to accumulator).
ADC M. (Add memory with carry to accumulator)
ADI data (Add immediate data to accumulator)
ACI data (Add with carry immediate data to accumulator).
DAD Rp. (Add register paid to H-L pair).
SUB R. (Subtract register from accumulator).
SUB M. (Subtract memory from accumulator).
SBB R. (Subtract register from accumulator with borrow).
SBB M. (Subtract memory from accumulator with borrow).
SUI data. (Subtract immediate data from accumulator)
SBI data. (Subtract immediate data from accumulator with borrow).
INR r (Increment register content)
INR M. (Increment memory content)
DCR r. (Decrement register content).
DCR M. (Decrement memory content).
INX rp. (Increment register pair)
DCX rp (Decrement register pair)
DAA (Decimal adjust accumulator)

Logical operations

The logical instructions provided by 8085 perform logical, rotate, compare and complement operations.

ANA R. (AND register with accumulator)
ANI data. (AND immediate data with accumulator)
ORA R (OR register with accumulator)
ORI data. (OR immediate data with accumulator)
XRA r. (EXCLUSIVE â€“ OR register with accumulator)
XRI data. (EXCLUSIVE-OR immediate data with accumulator)
CMA. (Complement the accumulator)
CMC. (Complement the carry status)
STC. (Set carry status)
CMP R. (Compare register with accumulator)
CPI data. (Compare immediate data with accumulator)
RLC (Rotate accumulator left)
RRC. (Rotate accumulator right)
RAL. (Rotate accumulator left through carry)
RAR. (Rotate accumulator right through carry

Branch operations

These instructions allow the 8085 to change the sequence of the program, either unconditionally or under certain test conditions. These instructions include branch instructions, subroutine call and return instructions and restart instructions.

1. JMP addr (label). (Unconditional jump).
2. Conditional Jump addr (label):.
    a. **JZ** addr (label). (Jump if the result is zero)
    b. **JNZ** addr (label) (Jump if the result is not zero)
    c. **JC** addr (label). (Jump if there is a carry)
    d. **JNC** addr (label). (Jump if there is no carry)
    e. **JP** addr (label). (Jump if the result is plus)
    f. **JM** addr (label). (Jump if the result is minus)
    g. **JPE** addr (label) (Jump if even parity)
    h. **JPO** addr (label) (Jump if odd parity)
3. CALL addr (label) (Unconditional CALL)
4. RET (Return from subroutine)
5. RST n (Restart)

Stack, Input/output and Machine control operations

These instructions control the stack operations, input/output operations and machine operations. The stack instructions allow the transfer of data from register pair to stack memory and from stack memory to the register, pair. The input/output instruction allows the transfer of 8-bit data to input/output port. On the other hand machine instructions control the Machine operations such as interrupt, halt, or do nothing.

a. NOP (No Operation)
b. HLT (Halt)
c. EI (Enable Interrupts)
d. DI (Disable Interrupts)
e. SIM (Set Interrupt Masks)
f. RIM (Read Interrupt Masks)

3. Write short notes on: Instruction word size, Data format

Instruction word size

- Microprocessor recognizes and operates in binary number. However each microprocessor has its own binary words, meanings and language. The words are formed by combining a number of bits for a given machine. The word is defined as the number of bits the microprocessor recognizes and processes at a time. The word length ranges from four bits for small, microprocessor based system to 64 bits for high speed large computers. Another term commonly used to express word length is byte. A byte is defined as a group of eight bits. For example a 16-bit microprocessor has a word length equal of two bytes .The term nibble stands for a group of four bits .A byte has two nibbles called as lower and higher.

Instruction Word Size (8085):

According to the word length of the Intel 8085, there are three types of instructions:

i. 1-Byte instruction
ii. 2-Byte instruction
iii. 3-Byte instruction

One-Byte instruction: Also known as 8-bits instruction. In one-byte instruction, there is only the opcode. The Operand is specified in the Opcode itself.
Example: MOV A, C. Machine CODE: 78 The operand and the opcode are specified in the 8 bits or 1 byte.

Two-byte instructions: In 2-byte instruction the one byte is used to specify the Opcode (Operation code) and the second byte is used for an operand which can be DATA or an 8-bit address.
Example: MVI B,05 (Move the data 05H to register B). Machine CODE = 06, 05.
The 8bits (1 byte) will be occupied by the Opcode (MVI B) and other 8bits are used by the data (05). So in total it requires the 16bits.

Three-Byte instruction: In 3-byte instruction the first byte is used by an Opcode while the 2nd and 3rd bytes are either 16-bit data or 16-bit address. In total, it is consist of 32 bits.
Example: LXI H, 8500 (load 8500H to H-L registers pair). Machine CODE = 21, 00, 85.

Data Format:

- The 8085 is an 8-bit microprocessor, and it processes only binary numbers. However the real world operates in decimal number and language of alphabets and characters. In 8 bit processor systems, commonly used codes and data formats are ASCII , BCD , signed integer and unsigned integers.

ASCII Code:

This is 7 bit alphanumerical code that represent decimal number , English alphabets and nonprintable characters such as carriage return . Extended ASCII is an 8 bit code.The additional number represent the graphical characters.

BCD Code:

Binary coded decimal (**BCD**) is a system of writing numerals that assigns a four-digit binary code to each digit 0 through 9 in a decimal (base-10) numeral. The four-bit**BCD** code for any particular single base-10 digit is its representation in binary notation, as follows: 0 = 0000.An 8-bit register in the 8085 can accommodate two BCD number.

Signed Integer

Signed integers are numbers with a "+" or "-"sign. If n bits are used to represent a signed binary integer number, then out of n bits,1 bit will be used to represent a sign of the number and rest (n - 1)bits will be utilized to represent magnitude part of the number itself.In 8 bit processor the most significant digit , D7 is used for sign , 0 for positive sign and 1 represent negative sign . The remaining seven bit D6-D0 represents magnitude of number.

Unsigned Integer:

An integer without a sign can be represented by all the 8 bit in a microprocessor register. Therefore the largest number that can be processed at one time is FFH. However, this doesn’t imply that the 8085 microprocessor is limited to handling only 8-bit number. Number larger than 8 bit is processed by dividing them in group of 8 bit.