Lecture-19

MICROPROCESSOR INSTRUCTIONN SET:

Each subsystem in a microprocessor based system – the memory, the microprocessor and the input and output devices- can be thought of in terms of the registers it contains. Random access semiconductor memory is a collection of register. A microprocessor itself consists of general purpose and dedicated registers. Input and output devices contain registers that hold their data.

The function of a microprocessor system is implemented by a sequence of data transfer between registers in the memory, the μp and I/O devices and data transformation that occur primarily in registers within the microprocessor. Each register that can be manipulated under program control is addressable in same manner - allowing it to be singled out for use in a data transfer operation or transformation.

The kinds of individual transfers and transformations possible are specified by the microprocessor instruction sets. Each instruction in the set causes one or more data transfers and/or transformations. A sequence of instructions constitutes a program. The timing & control section decodes the program instruction in turn, and using timing signals derived from the system clock, controls what register transfers or transformations take place and when.

Each processor is designed to execute particular function set. Instruction set one fixed by design cannot be changed. Intel 8080 which came in to market in 1973 has 72 basic instructions and 244 variations. Intel 8085A μp which came in to the market in January 1977 is upward compatible with lintel 8080 μp. It has 74 basic

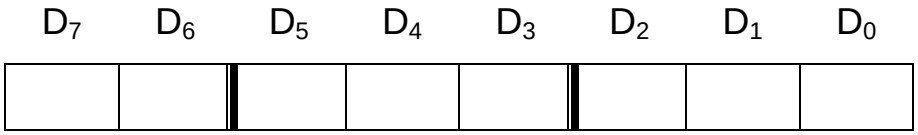
instructions and 246 variations. It includes all the 72 instruction of 8080 μp. The two other instructions in 8085A are RIM & SIM.

8085A is faster than 8080 μp. The program written for 8080 can be run without any modification on 8085A but the user has to be careful where the DELAY ROUTINES are involved because of difference in speed of operation.

INSTRUCTION FORMAT:

All instruction of 8085A μp are 1 to 3 bytes in length. The bit pattern of the first cycle is the op-code. The bit pattern is decoded in the instruction register and provides information used by the timing and control section to generate sequence of elementary operations-micro operations - that implement the instruction.

Figure shows the op-code of a instruction single–byte instruction available at memory location 'N':



The 8-bit of the op-code is divided into three portions. First group D_7, D_6, second group D_5 D_4 D_3 and third group D_2 D_1 D_0. D_2 D_1 D_0 when necessary contains the source code SSS as discussed in previous lecture. D_2 D_1 D_0 group contains the code of destination register DDD. If a register pair is involved, the code bits RP is placed in D_5 D_4. Whenever D_5 D_4 represents the register pair D_3 normal tells whether it is loading operation or storing operation. The first group D_7 D_6 gives the idea of the mnemonic of the operation code.

Whenever a 2-byte instruction is used the first byte at memory location N is the op-code of the instruction followed by either an 8-bit data or an 8-bit address at memory location N+1.

Whenever a 3-byte instruction is involved, the first byte at memory location N is the opcode followed by either a-16 bit address or a 16-bit data. The second memory location i.e., N+1 contains the lower order addresses or data and the third memory location N+2 contains the higher order address or data.

ADDRESSING MODES:

Most of the instruction execution requires two operands e.g. transfer of data between two registers of a microprocessor system. How the μp knows the positions of these operands? The method of identifying the operands position by the instruction format is known as the addressing mode. Whenever two operands are involved in an instruction, the first operand is assumed to be in any register of the μp itself. It is for the user to put that operand in the register involved. The second operand may be located in one of the following.

i. In any general purpose register of the microprocessor.
ii. In a particular external memory location.
iii. It can be immediately available in an instruction format.
iv. In an I/O device.

In Intel 8085A μp, the following addressing modes are used:

Register Addressing Mode:

When the operands for any instruction are available in internal general purpose registers, only the registers need be specified as the

address of the operands. Such instruction are said to use the register addressing mode. These instructions are one byte instructions. Within that byte i.e. op-code, the registers are specified.

For example, **MOV r_1, r_2**; **ADD r**; **XCHG**; **DAD rp** etc.

MOV r_1, r_2: This is an ALP statement and meaning of the instruction is move the content of register r_2 to register r_1. The opcode for the instruction is 01 DDD SSS. DDD specifies the code for the destination register r_2 and SSS specifies the code for source register r_1. The first two bits 01 specify the MOV operation e.g., **MOV B, A** the opcode of the instruction is $(01\ 000\ 111)_2 = 47_H$.

ADD r: This is an ALP statement. ADD is the mnemonic for addition and meaning of the instruction is add the content of the register to the content of the accumulator and store the result back in accumulator. Thus one of the operand is assumed to be in accumulator by implied addressing mode. The second operand is available in any general purpose register specified in the instruction. The op-code is 10 000 SSS.

E.g. **ADD H**. The opcode is $(10\ 000\ 100)_2 = 84_H$, the macro RTL implemented is (A) ← (A) + (r)

<u>Direct Addressing Mode:</u>

In this addressing mode, the instruction contains the address of the operand (external register) involved in the transfer. The 8085A provides16-bit memory address requiring that the address contained

in the instruction be 16-bit long as a second and third bytes of the instruction. Thus it is invariably a 3-byte instruction.

e.g. **LDA addr**. This is an ALP statement ‘addr’ in operand field is a symbolic name given to 16-bit address. The symbolic name can be chosen by the user with reference to context. LDA is the mnemonic for Load Accumulator Direct. The instruction format for this is as shown:

Opcode	N
$<B_2>$	N+1
$<B_3>$	N+2

where the memory location ‘N’ contains the opcode $(00\ 110\ 010)_2 = (3A)_H$ followed by a lower order 8-bits of address $<B_2>$ and higher order 8-bit of address $<B_3>$ at memory location N+1 & N+2 respectively. The meaning of instruction is load the accumulator from the memory location whose 16-bit address is available directly, in the instruction itself. The macro RTL implemented is,

$$(A) \longleftarrow M(B_3, B_2)$$

This is symbolic representation. Only one operand is involved in the instruction execution.

Register Indirect Addressing Mode:

In this case, the instruction specifies a register pair which contains the address of the memory where the data is located or into which the data in to be placed. Thus the address of the operand is given indirectly through a register pair. In other words, the operand is

in memory location or external register whose address is available in an internal general purpose register pair.

The (H, L) register pair is used as a pointer in many register indirect instructions of Intel 8085A. The register (H) holds the higher and register (L) holds the lower bytes of the effective address. e.g. **MOV r, M** transfers single byte from an external register (M) to any of the several internal working registers. External register (M) means the external register pointed by (H,L) register pair. The macro RTL implemented is,

$$(r) \longleftarrow M\ (H, L)$$

Before register indirect instructions are used in a program, a previous instruction must load register pair (H, L) with the appropriate address.

The register pair (B, C) & (D, E) are also used as memory pointer register in two 8085A instructions i.e. **LDAX rp** and **STAX rp**. The internal register involved for data transfer is always accumulator.

The meaning of LDAX rp is load the accumulator from the memory location whose address is available in rp (register pair either (B, C) or (D, E)). The macro RTL implemented is (A) $\longleftarrow$ M (rpH, rpL). The opcode for **LDAX B** is $(00\ 001\ 010)_2 = (0A)_H$.

Immediate Addressing Mode:

In this type of addressing mode, the operand is available directly in the instruction itself. If the operand data involved is of 8-bits then the instruction is of two bytes. The first byte is the opcode followed by 8-bit data byte. If 16-bit data is involved in the instruction then the first byte is opcode at memory location N followed by the

lower order data byte at memory location N+1 and higher order data byte at memory location N+2.

e.g. **MVI r, data** there is two byte instruction .the instruction format is

00 DDD 110	N
$<B_2>$	N+1

The meaning of the instruction is move the 8-bit data immediately available in the instruction as the 2nd byte to the destination register (r). DDD identifies the internal register the macro RTL implemented

i.e., (r) ⟵ $<B_2>$

Another example is LXI rp data,

00 RR0 001	N
$<B_2>$	N+1
$<B_3>$	N+2

The macro RTL implemented is

(RpL) ⟵ $<B_2>$

(RpH) ⟵ $<B_3>$

<u>Implied Addressing Mode</u>:

There are certain instructions that operate on one operand. Such instructions assume that the operand is in the ACC and, therefore, need not specify any address. Many instructions in the logical group like RLC, RRC, RAR, RAL and CMA fall in to this category. All these are one byte instruction. Those instructions that specify the address of one operand, use implied addressing for the other operand.