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

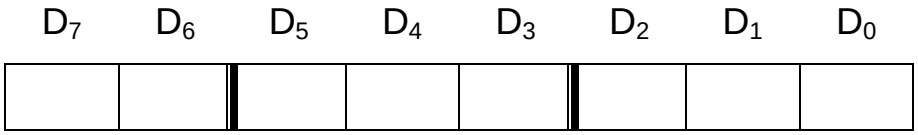
instructions and 246 variations. It includes all the 72 instruction of 8080 μp. The two other instructions in 8085A are RIM & SIM.

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.

Lecture-20

INSTRUCTION SET

As discussed in previous lecture, Intel 8085A μp has 74 basic instructions and 246 variations. These instructions are used for data transformation and data manipulations in the takes place in the μp based system. The instructions of 8085A μp are 1 to 3 bytes in length. The first byte of the instruction is always the opcode of the instruction. The 2nd and 3rd bytes, if available, are the data bytes or the address bytes. The instructions normally involve either one operand or two operands. The position of the operand(s) is(are) identified by the addressing mode. The addressing modes are Register addressing mode, Direct addressing mode, Register indirect addressing mode, Immediate addressing mode and implied addressing mode. Depending on the type of actions taken by the processor, these instructions can be put in different categories.

Classification of Instruction Set:

The 74 instructions available in the 8085A can be divided into five groups depending on their function:

1. **Data Transfer group**: This group comprises instructions that move data between internal registers of μp, between internal register and external memory location and I/O transfer.
2. **Arithmetic group**: Instructions meant for arithmetic operations that add, subtract, increment or decrement data in a register are put in this group.
3. **Logic group**: Instructions that carry out logic operation, such as AND, OR, EX-OR, compare data in the accumulator with another

internal or external register, complement and rotate data in the accumulator are considered in this group.

4. **Branch group**: This group consists of instructions that change the execution sequence of a program, such as conditional and unconditional jumps, subroutine call and return instructions.
5. **Stack Machine Control group:** The instructions used for maintaining the stack and internal control flags are put in this group.

DATA TRANSFER GROUP:

It consists of 15 basic instructions and 86 variations. The basic operation involved is DATA transfer between two register of a microprocessor system. One of the register is always located in the μp itself; the other may be located in one of the following

1) An I/o device
2) Memory
3) The microprocessor

Registers located within the microprocessor are referred to as internal registers. The accessible internal registers are A, B, C, D, E, H, L, SP, PC. The registers located in memory ROM, RWM or I/O devices are referred to as external registers. Therefore, this group includes transfer of data from internal register to another internal register, internal register to memory, memory to internal register, accumulator (A) to output device, or from input device to accumulator.

The register from which data is transferred is the source register and the register to which data is transferred in the destination register. A transfer involves copying the contents of the source register into the destination register; the contents of the source

register are not altered. Each data transfer instruction identifies the source register and the destination register. Identification of one or both of these register may be implied by the instruction in memories or may be explicit. Internal registers are frequently implied; whereas the external registers are usually identified by an explicit address that is part of the instruction. The operation code format for the instructions of this group is shown in fig.5.1.

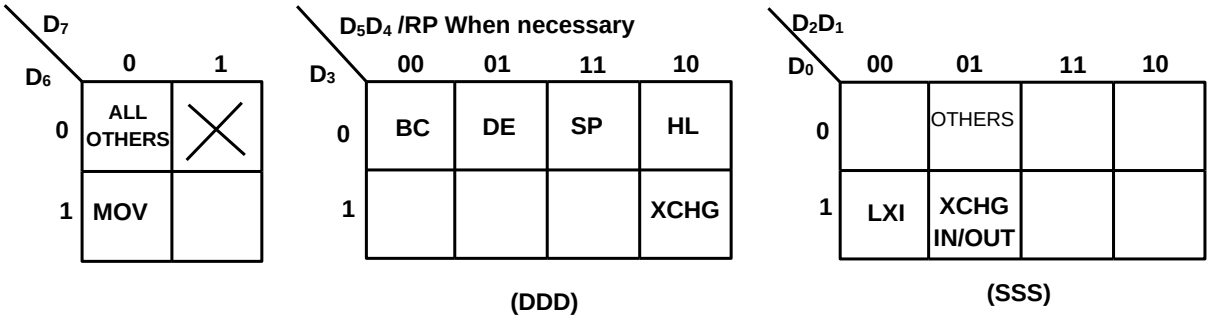


Fig.5.1 Operation Code Format of Data Transfer Group Instructions

1. MOV r_1, r_2: This is an ALP statement. MOV is the mnemonic for move operation. r_2 is the source register and r_1 is the destination register in the operand fields. The meaning of the instruction is "Move the contents of the register r_2 into r_1". The content of register r_2 is not destroyed. Content of r_1 is destroyed and new value form r_2 takes it place. The macro RTL implemented is

$$(r_1) \longleftarrow (r_2)$$

This is a single byte instruction at memory location N. The opcode of the instruction will be

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
0	1	D	D	D	S	S	S	N

The instruction has 49 variations (7x7), seven combinations for source registers (SSS) and seven combinations for destination registers (DDD) other than 110. It has register addressing mode.

How the execution of this instruction takes place? During the T_1 state of the first machine cycle (OFMC), the contents of the program counter are placed on the address bus (A_{15} ...A_8) and address/data bus (AD_7- AD_0). Since address may be low or high, it is customary to use the double sided waveforms, the high byte of the program counter (PCH) goes to the address bus and the low byte (PCL) to the address/data bus.

The ALE signal initially goes high, then midway through the T_1 state, ALE goes low. It is this falling edge that latches the address in to the external latch. Also IO/$\overline{M}$ goes low near the beginning of the T_1 state. This enables the peripheral chips for a memory operation rather than an I/O operation.

During the T_2 state, the program counter is incremented. The address disappears from the address/data bus at the beginning of the T_2 state. This is necessary because an instruction fetch is in progress and the address/data bus is required. The dashed line an AD_7-AD_0 wave from means that the data on the bus is invalid. Towards the end of the T_2 state, the opcode of the instruction appears on the address/data bus. The precise time when opcode appears depends on the memory access time; the length of the buses and other factors.

At the beginning of the T_2 state $\overline{RD}$ goes low and stays low until the middle of the T_3 state. During the T_3 state, the opcode available on the AD bus is copied into the instruction register (IR). During the T_4

state, the instruction is decoded. The μp now knows that the instruction is MOV r_1, r_2. It is represented as MOV r_1, r_2=1 and is executed. After the fetching operation of this instruction, instruction pointer goes back to T_1 state & PC points to next address.

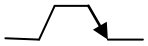
In fact the execution does not take place instantaneously. When the instruction is decoded, first the contents of register r_2 are copied into the temporary register and then contents of the temporary register are copied into register r_1. This operation takes place in the next two states T_1 and T_2. This completes the execution of the MOV instruction. Since in these two extra states during MC-2 only the internal bus is used, address bus and address/data bus are not used, therefore, there buses can be used for some other purpose to save execution time.

One way to save processing time is by starting the next instruction OFMC during the second machine cycle MC-2 of the move instruction. This is called Fetch Execute Overlap (FEO). During T_1 of this machine cycle, the contents of the program counter are outputted to address bus & address/data bus. During T_2 state the next instruction opcode is transferred to address/data bus form memory externally and simultaneously MOV r_1, r_2 instruction is completed internally. Hence, during the MC-2 of the move instruction cycle, MC–1 of the next cycle is operative.

Thus the instruction cycle for MOV r_1,r_2 takes only 4 clock periods (& not six). After the execution PC points to next address. The micro RTL flow for the instruction execution is shown below:

Machine Cycle- 1:

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

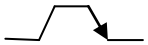
T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (IR) ⟵ AD_7-AD_0

T_4: μp decodes the opcode. MOV r_1, r_2= 1.

Machine Cycle- 2 or (Machine Cycle-1 of next instruction):

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

FEO [(Z) ⟵ (r_2)]

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

FEO [(r_1) ⟵ (Z)]

The data flow and the timing waveforms during the execution of this instruction are shown in fig.5.2a and fig.5.2b respectively:

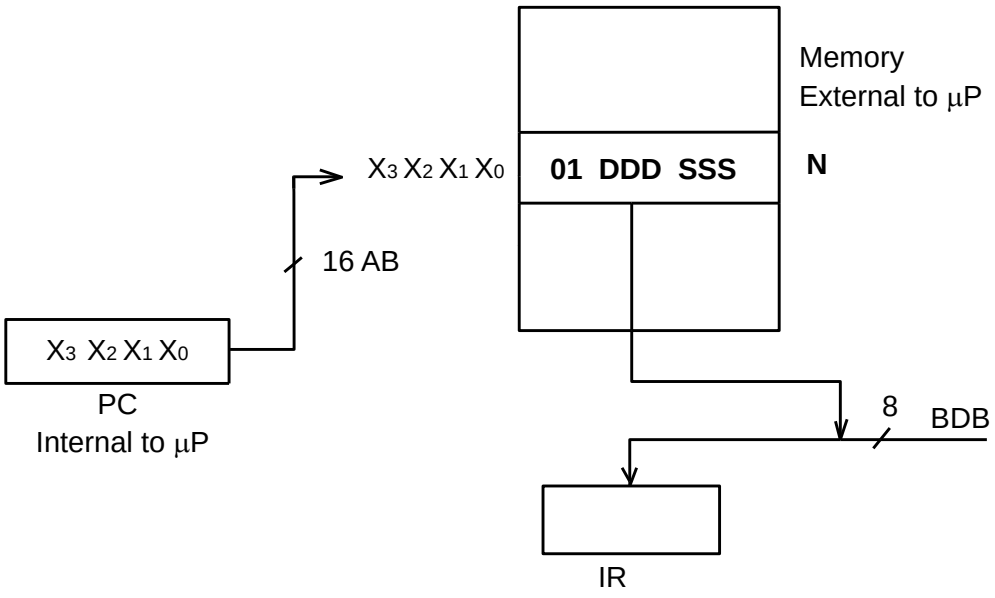


Fig.5.2a Data Flow During the Execution of MOV r_1,r_2 Instruction

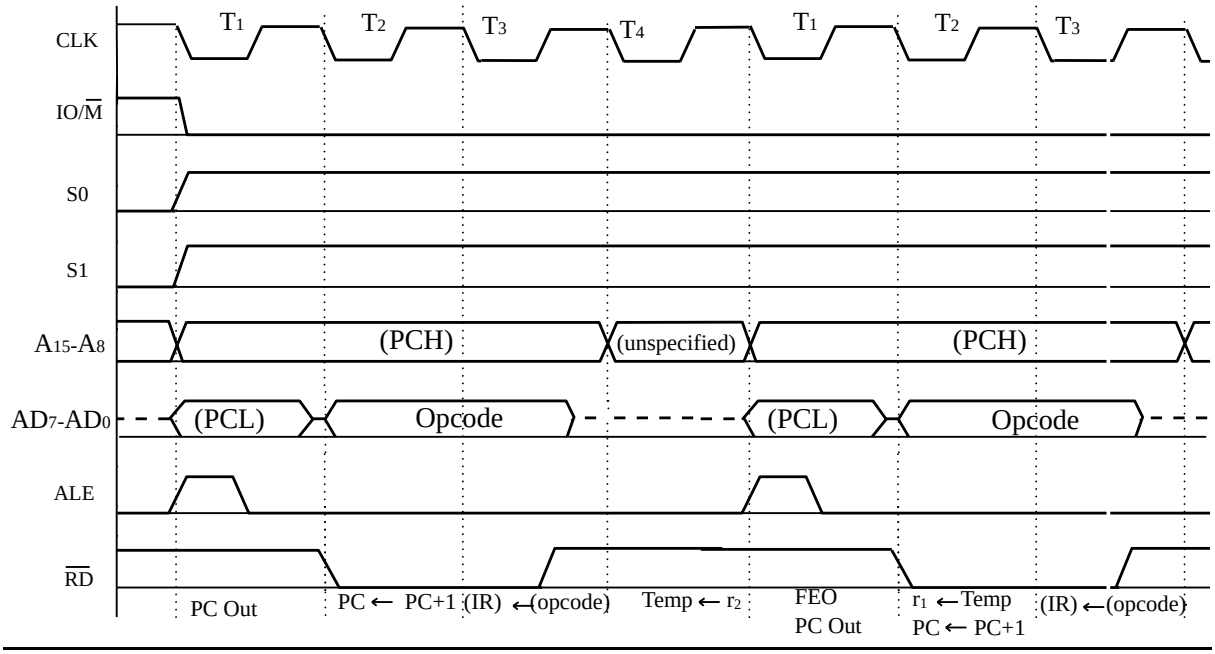


Fig.5.2b Timing Waveforms During the Execution of MOV r_1,r_2 Instruction

2. MOV r, M: This is an ALP statement. The meaning of this instruction is "Move the content of the memory location whose address is available into (H,L) pair into the internal general purpose register r". The macro RTL implemented is

(r) ⟵ M(H,L)

M(H,L) is the source register, (r) is the destination register. It is a single byte instruction at memory location N. The operation code is,

0	1	D	D	D	1	1	0	N

It has seven variations. DDD cannot be 110 because direct memory to memory data transfer is not allowed in this processor. If memory to memory data transfer is required, it has to be routed through one of the internal register. The micro RTL flow is given below:

Machine Cycle- 1:

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = ⎍

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (IR) ⟵ AD_7-AD_0

T_4: μp decodes the opcode. MOV r, M= 1.

Machine Cycle- 2:

MRMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=0

T_1: A_{15}-A_8 ⟵ (H), AD_7-AD_0 ⟵ (L), ALE = ⎍

T_2: $\overline{RD}$ = 0, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (r) ⟵ AD_7-AD_0

This Operation requires only two machine cycles, OFMC & MRMC, and total 7 states. It requires 3.5µsec using 2 MHz clock.

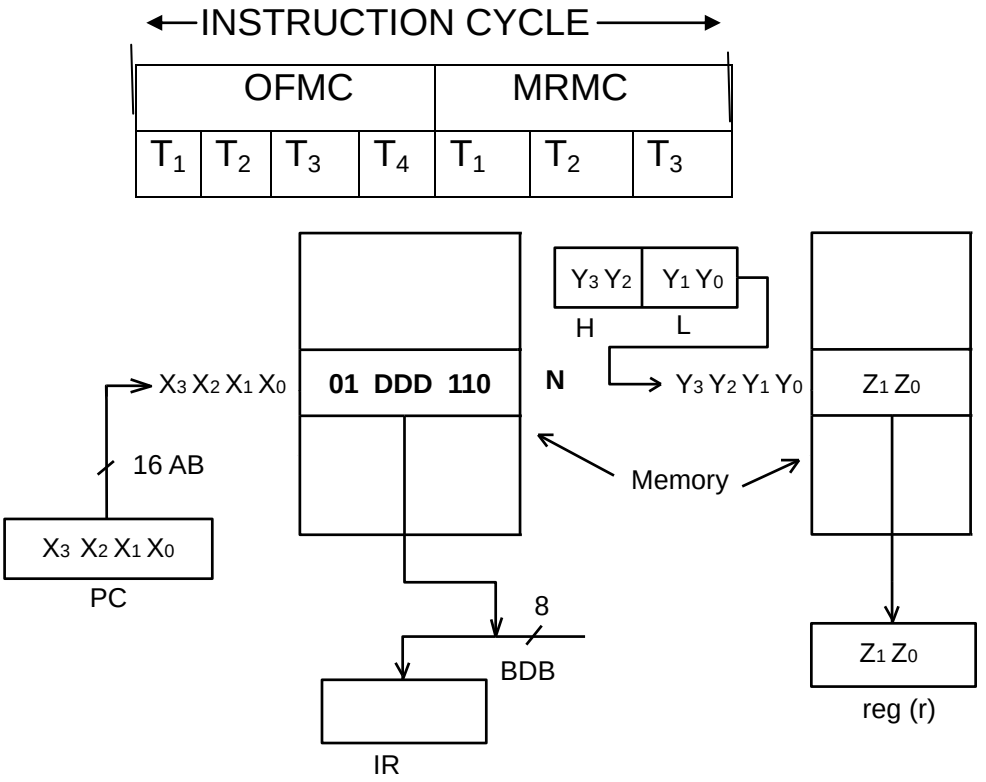


Fig.5.3 Data Flow during the Execution of MOV r, M Instruction

The addressing mode is register indirect addressing mode. Fig.5.3 illustrates the operation cycle.

3. MOV M, r: This is an ALP statement. The meaning of the instruction is "Move the content of the internal general purpose register r into the memory location whose address is available in (H,L) register pair". The macro RTL implemented is

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

It is a single byte instruction. The operation code is

0	1	1	1	0	S	S	S	N

It has seven variations. SSS cannot be 110 because direct memory to memory data transfer is not allowed. The micro RTL flow is given below:

Machine Cycle- 1:

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = ⎍

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (IR) ⟵ AD_7-AD_0

T_4: μp decodes the opcode. MOV M, r = 1.

Machine Cycle- 2:

MWRMC: Status signals IO/$\overline{M}$=0, S_1=0, S_0=1

T_1: A_{15}-A_8 ⟵ (H), AD_7-AD_0 ⟵ (L), ALE = ⎍

T_2: $\overline{WR}$ = 0, AD_7-AD_0 ⟵ (r)

T_3: $\overline{WR}$ = 1, ↑, M(AB) ⟵ AD_7-AD_0

The operation requires only two machine cycles- OFMC & MWRMC and total 7 states. It requires 3.5 µsec using 2 MHZ internal clock. The addressing mode is register indirect addressing mode. Fig.5.4 illustrates the operation cycle.

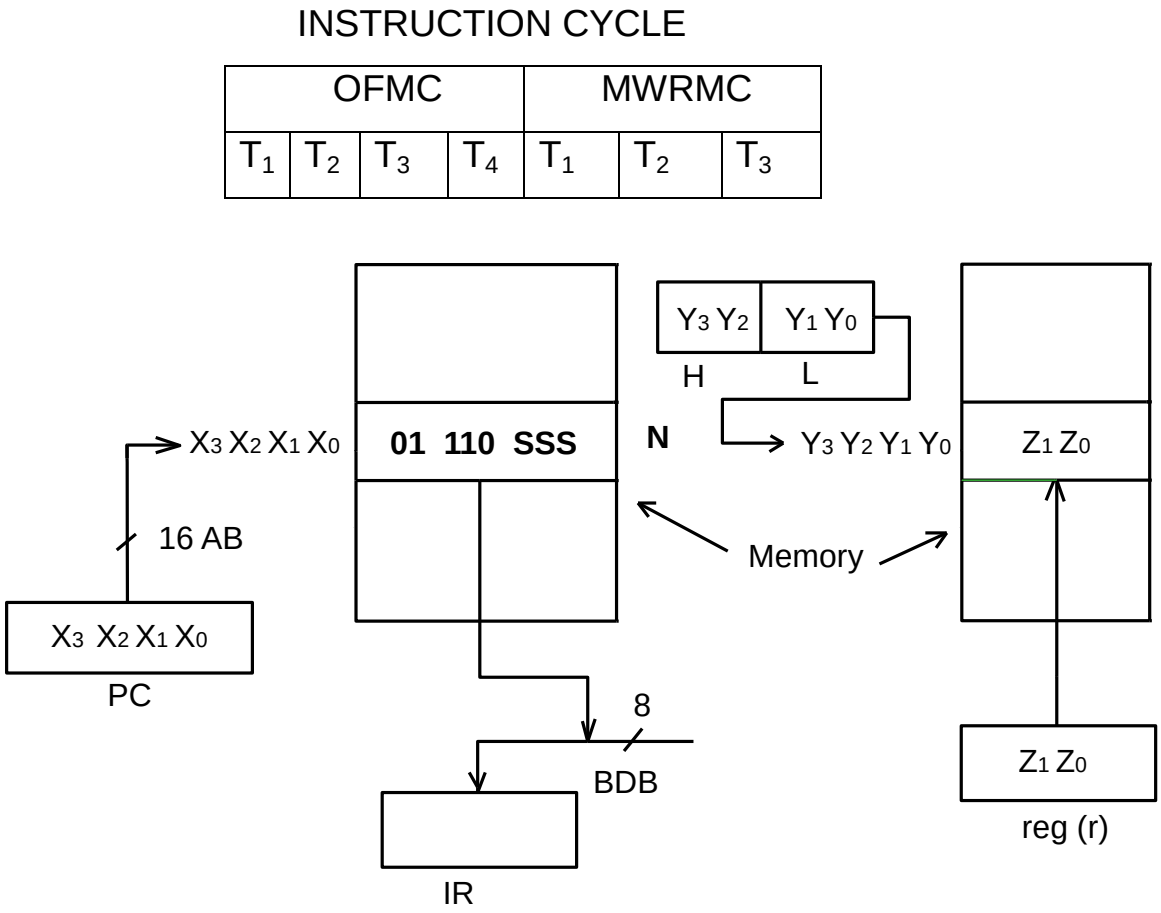


Fig.5.4 Data Flow during the Execution of MOV M, r Instruction

Lecture-21:

Instruction Set Contd.

4. MVI r, DATA: This is an ALP statement. DATA is the symbolic name given to 8-bit data which is immediately available as second byte of the instruction. Therefore, the source of data is the 2nd byte of the instruction itself. Therefore, addressing mode is immediate addressing mode. MVI is the mnemonic for move immediate. 'r' is the destination register. It is a 2 byte instruction at the memory location N & N+1. The opcode of the instruction is,

0	0	D	D	D	1	1	0	N
<B_2>								N+1

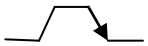
The meaning of the instruction is "Move 8-bit data immediately available as a 2nd byte of the instruction itself into the destination register r". The destination register is specified by 3-bit code DDD. It has 7 variations. The macro RTL implemented is

(r) ⟵ <B_2>

The micro RTL flow is

Machine Cycle- 1:

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (IR) ⟵ AD_7-AD_0

T_4: μp decodes the opcode. MVI r, data = 1.

Machine Cycle- 2:

MRMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=0

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = ⎍

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (r) ⟵ AD_7-AD_0

It requires only two machine cycles - OFMC & MRMC and 7 states. It requires 3.5µsec using 2 MHz internal clock. The operation is shown in fig.5.5

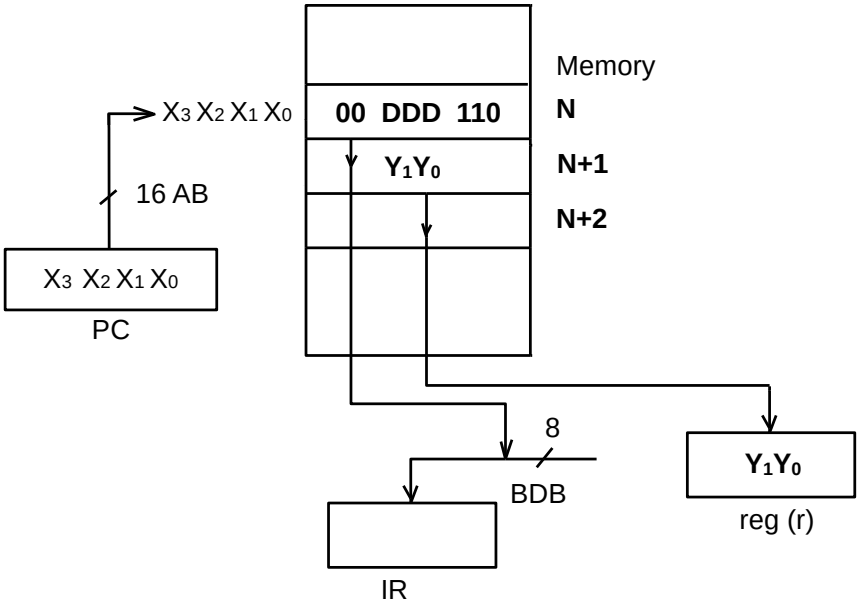


Fig.5.5 Data Flow during the Execution of MVI r, data Instruction

5. MVI M, DATA: This is an ALP statement. DATA is the symbolic name given to the 2nd byte of the instruction. MVI is the mnemonic for move immediate. M in the operand field stands for memory pointer. It is a 2 byte instruction at the memory location N & N+1. First byte is opcode byte and 2nd byte is 8 bit data. The opcode of the instruction is,

0	0	1	1	0	1	1	0	N
<B_2>								N+1

The meaning of the instruction is “Move 8-bit data available immediately as a 2nd byte of the instruction to the memory location whose address is available in memory pointed by (H, L) register pair”. The macro RTL implement is

$$M(H,L) \longleftarrow \langle B_2 \rangle$$

There is no variation in this instruction. The operation is illustrated in fig.5.6

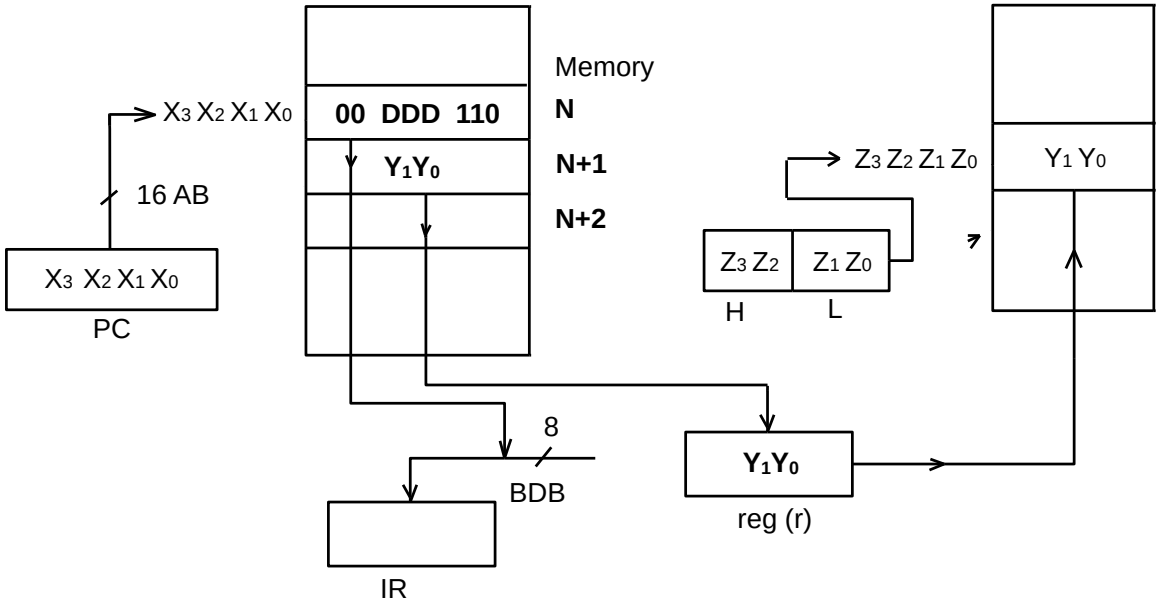


Fig.5.6 Data Flow during the Execution of MVI M, data Instruction

The micro RTL flow for the instruction is given below

Machine Cycle- 1:

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = ⎍

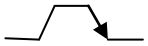
T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (IR) ⟵ AD_7-AD_0

T_4: μp decodes the opcode. MVI M, data = 1.

Machine Cycle- 2:

MRMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=0

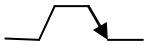
T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (Z) ⟵ AD_7-AD_0

Machine Cycle- 3:

MWRMC: Status signals IO/$\overline{M}$=0, S_1=0, S_0=1

T_1: A_{15}-A_8 ⟵ (H), AD_7-AD_0 ⟵ (L), ALE = 

T_2: $\overline{WR}$ = 0, AD_7-AD_0 ⟵ (Z)

T_3: $\overline{WR}$ = 1, ↑, M(AB) ⟵ AD_7-AD_0

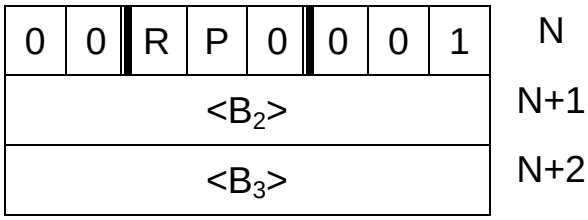
Therefore, the instruction requires three machine cycles OFMC, MRMC, MWRMC and total 10 states. It needs 5µsec time to execute this instruction using 2 MHz internal clock. The instruction uses immediate addressing mode.

6. LXI rp, DDATA: This is an ALP statement. DDATA stands for double-data, a symbolic name given to 16-bit data available immediately as the 2nd & 3rd bytes of the instruction. 'rp' stands for LOAD IMMEDIATE register pair. The alphabet 'X' in the mnemonic tells that a register pair is involved in the instruction. It is true for all instructions. The meaning of the instruction is "Load the 16 bit data immediate available as the 2nd & 3rd bytes of the instruction into register pair rp". The macro RTL implemented is,

(rpL) ⟵ <B_2>

(rpH) ⟵ <B_3>

The instruction format is



Only in the instruction D_3 bit is zero for loading otherwise D_3 bit is '1' for loading. There are 4 variations of this instruction. The bit code RP in the opcode at memory location N identifies the register pair.

RP = 00 LXI B, DDATA

RP = 01 LXI D, DDATA

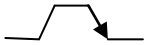
RP = 10 LXI H, DDATA

RP = 11 LXI SP, DDATA

The micro RTL flow implemented is given below:

Machine Cycle- 1:

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

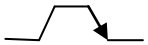
T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (IR) ⟵ AD_7-AD_0

T_4: μp decodes the opcode. LXI rp, d-data = 1.

Machine Cycle- 2:

MRMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=0

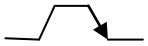
T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (rpL) ⟵ AD_7-AD_0

Machine Cycle- 3:

MRMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=0

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (rpH) ⟵ AD_7-AD_0

It requires three machine cycles- OFMC & two MRMC and total number of 10 states. It needs 5µsec using 2 MHz internal clock. It makes use of immediate addressing mode.

7. LDA ADDR: This is an ALP statement. ADDR is the symbolic name given to the 16 bit address directly available in the instruction. LDA is the mnemonic for LOAD ACCUMULATOR DIRECT. The meaning of the instruction is "Load the content of the memory location whose address is directly available in the instruction as 2nd and 3rd bytes into the accumulator". This is a 3-byte instruction. The instruction format is

0	0	1	1	1	0	1	0	N
<B_2>								N+1
<B_3>								N+2

The macro RTL implemented is

(A) ⟵ M(B_3, B_2)

This operation of this instruction is shown in fig.5.7.

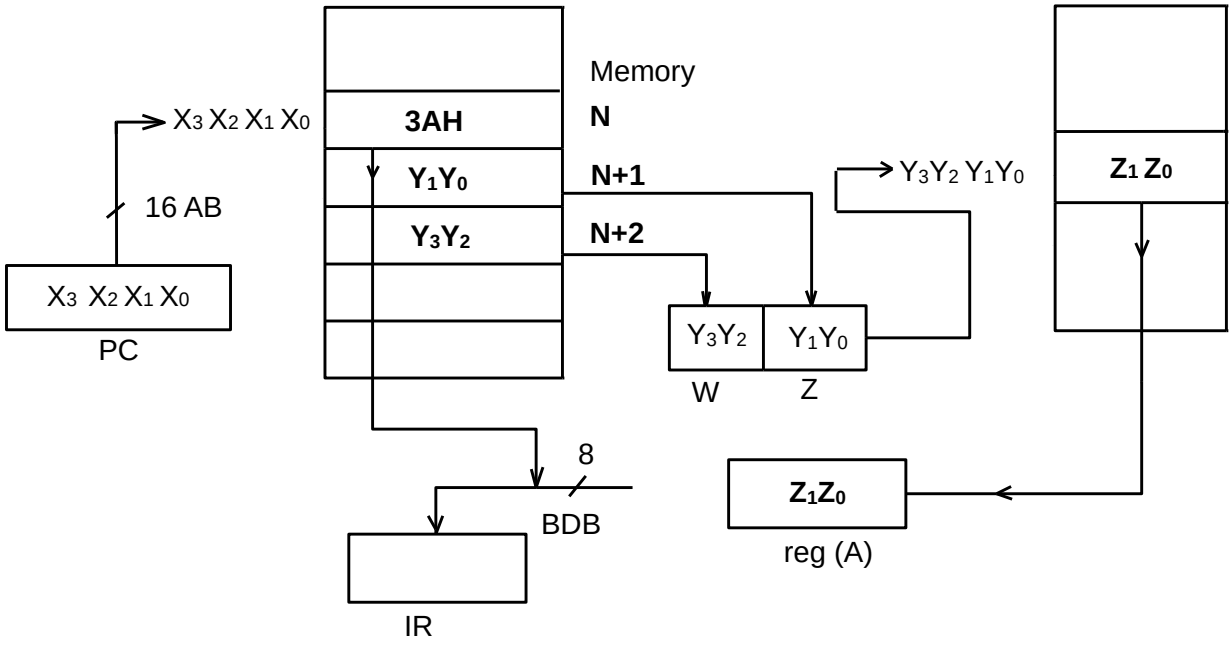


Fig.5.7 Data Flow during the Execution of LDA Addr Instruction

The micro RTL flow implemented is given below:

Machine Cycle- 1:

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ← (PCH), AD_7-AD_0 ← (PCL), ALE = ⎍

T_2: $\overline{RD}$ = 0, (PC) ← (PC) +1, AD_7-AD_0 ← M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (IR) ← AD_7-AD_0

T_4: μp decodes the opcode. LDA addr = 1.

Machine Cycle- 2:

MRMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=0

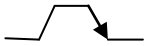
T_1: A_{15}-A_8 ← (PCH), AD_7-AD_0 ← (PCL), ALE = ⎍

T_2: $\overline{RD}$ = 0, (PC) ← (PC) +1, AD_7-AD_0 ← M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (Z) ← AD_7-AD_0

Machine Cycle- 3:

MRMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=0

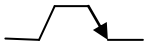
T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (W) ⟵ AD_7-AD_0

Machine Cycle- 4:

MRMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=0

T_1: A_{15}-A_8 ⟵ (W), AD_7-AD_0 ⟵ (Z), ALE = 

T_2: $\overline{RD}$ = 0, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (A) ⟵ AD_7-AD_0

It requires 4 machine cycles- OFMC & three MRMC and total number of 13 states. It needs 6.5µsec using 2 MHz internal clock. It makes use of direct addressing mode.

8. STA ADDR: This is an ALP statement. STA is the mnemonic for the STORE ACCUMULATOR DIRECT. The meaning of the instruction is “Store the content of the accumulator into the memory location whose address is directly available in the instruction itself as a 2nd & 3rd byte of instruction”. This is a 3-byte instruction. The instruction format is

0	0	1	1	0	0	1	0	N
<B_2>								N+1
<B_3>								N+2

The macro RTL implemented is

$M(B_3, B_2)$ ⟵ (A)

This instruction has no variation. The addressing mode is direct addressing mode. This operation of this instruction is shown in fig.5.8.

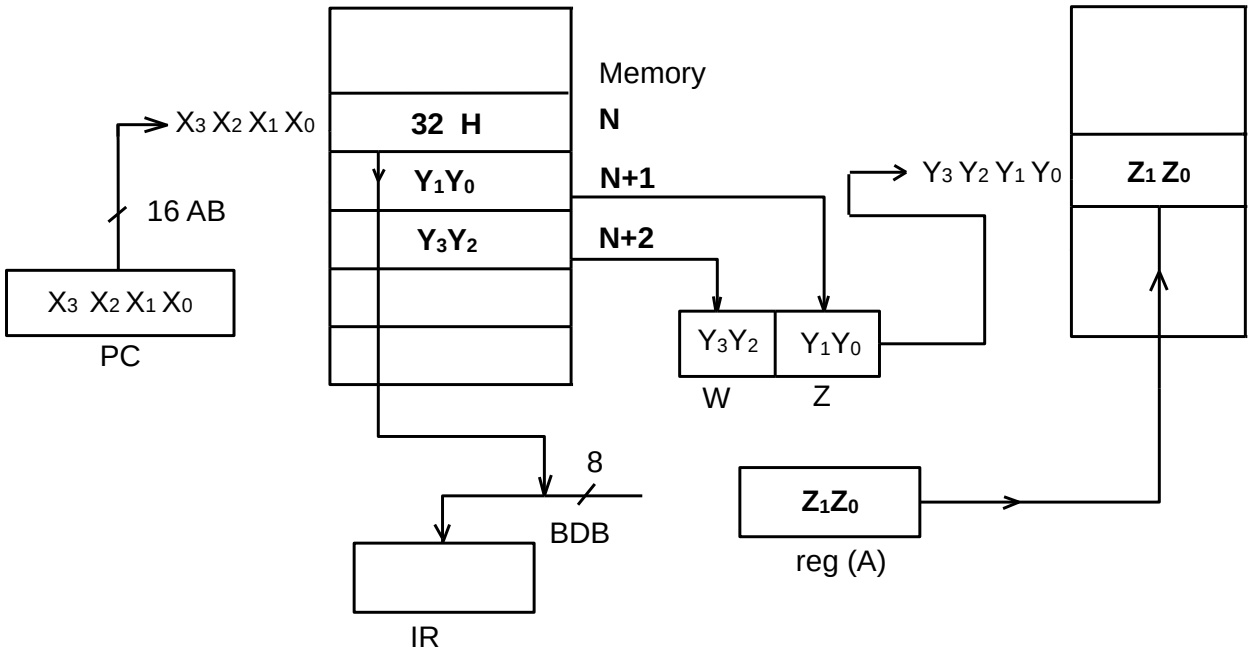


Fig.5.8 Data Flow during the Execution of STA Addr Instruction

The micro RTL flow is given below:

Machine Cycle- 1:

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = ⎍

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (IR) ⟵ AD_7-AD_0

T_4: μp decodes the opcode. STA addr = 1.

Machine Cycle- 2:

MRMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=0

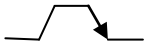
T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = ⎍

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (Z) ⟵ AD_7-AD_0

Machine Cycle- 3:

MRMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=0

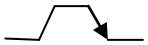
T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (W) ⟵ AD_7-AD_0

Machine Cycle- 4:

MWRMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (W), AD_7-AD_0 ⟵ (Z), ALE = 

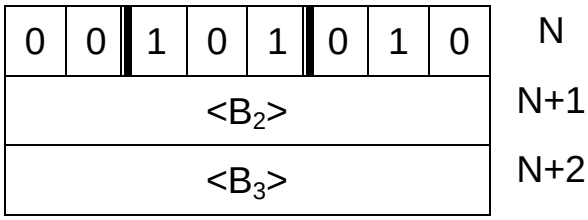
T_2: $\overline{RD}$ = 0, AD_7-AD_0 ⟵ (A)

T_3: $\overline{RD}$ = 1, ↑ , M(AB) ⟵ AD_7-AD_0

It requires 4 machine cycles- OFMC, two MRMC & one MWRMC and total number of 13 states. It needs 6.5µsec using 2 MHz internal clock. It makes use of direct addressing mode.

Lecture-22:

9. LHLD ADDR: This is an ALP statement. LHLD is the mnemonic for LOAD (H,L) REGISTER PAIR DIRECT. The meaning of the instruction is "Load the content of the memory location whose address is directly available as 2nd & 3rd byte of the instruction to register L and the content of the memory location at next higher address to the register H". This is a 3-byte instruction. The instruction format is



The macro RTL implemented is

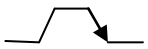
$$(L) \longleftarrow M(B_3, B_2)$$

$$(H) \longleftarrow M((B_3, B_2) + 1)$$

This instruction has no variation. The addressing mode is direct addressing mode. The micro RTL flow is:

Machine Cycle- 1:

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

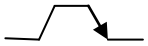
T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (IR) ⟵ AD_7-AD_0

T_4: μp decodes the opcode. LHLD addr = 1.

Machine Cycle- 2:

MRMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=0

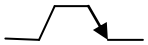
T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (Z) ⟵ AD_7-AD_0

Machine Cycle- 3:

MRMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=0

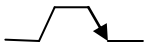
T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (W) ⟵ AD_7-AD_0

Machine Cycle- 4:

MRMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=0

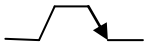
T_1: A_{15}-A_8 ⟵ (W), AD_7-AD_0 ⟵ (Z), ALE = 

T_2: $\overline{RD}$ = 0, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (L) ⟵ AD_7-AD_0

Machine Cycle- 5:

MRMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=0

T_1: A_{15}-A_8 ⟵ (W), AD_7-AD_0 ⟵ (Z+1), ALE = 

T_2: $\overline{RD}$ = 0, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (H) ⟵ AD_7-AD_0

The instruction requires 5 machine cycles- OFMC & fours MRMC and total number of 16 states. It needs 8.0μsec using 2 MHz internal

clock. The data flow during the execution of instruction is shown in fig.5.9.

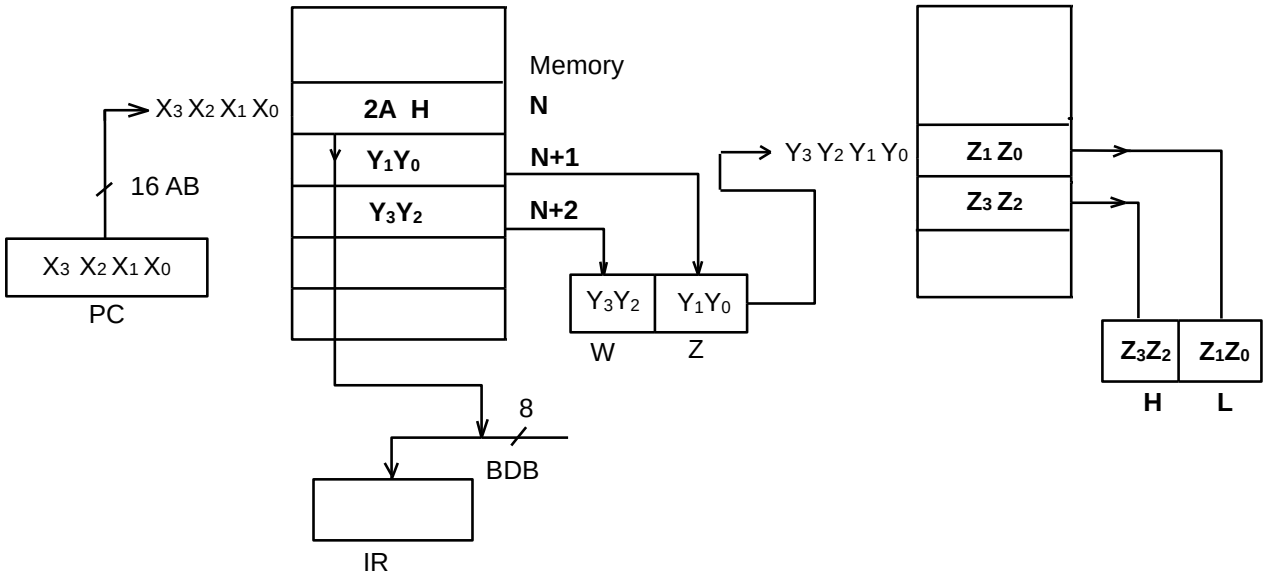


Fig.5.9 Data Flow during the Execution of LHLD Addr Instruction

10. SHLD ADDR: This is an ALP statement. SHLD is the mnemonic for STORE (H, L) REGISTER PAIR DIRECT. The meaning of the instruction is the "Store the content of (L) register into the memory location whose address is available as 2nd & 3rd byte of the instruction itself and store the content of (H) register into the memory location whose address is next higher address". The instruction format is

0	0	1	0	0	0	1	0	N
<B_2>								N+1
<B_3>								N+2

The macro RTL implemented is

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

$M((B_3, B_2) + 1) \longleftarrow (H)$

This instruction has no variation. The addressing mode is direct addressing mode. The data flow during the execution of instruction is shown in fig.5.10.

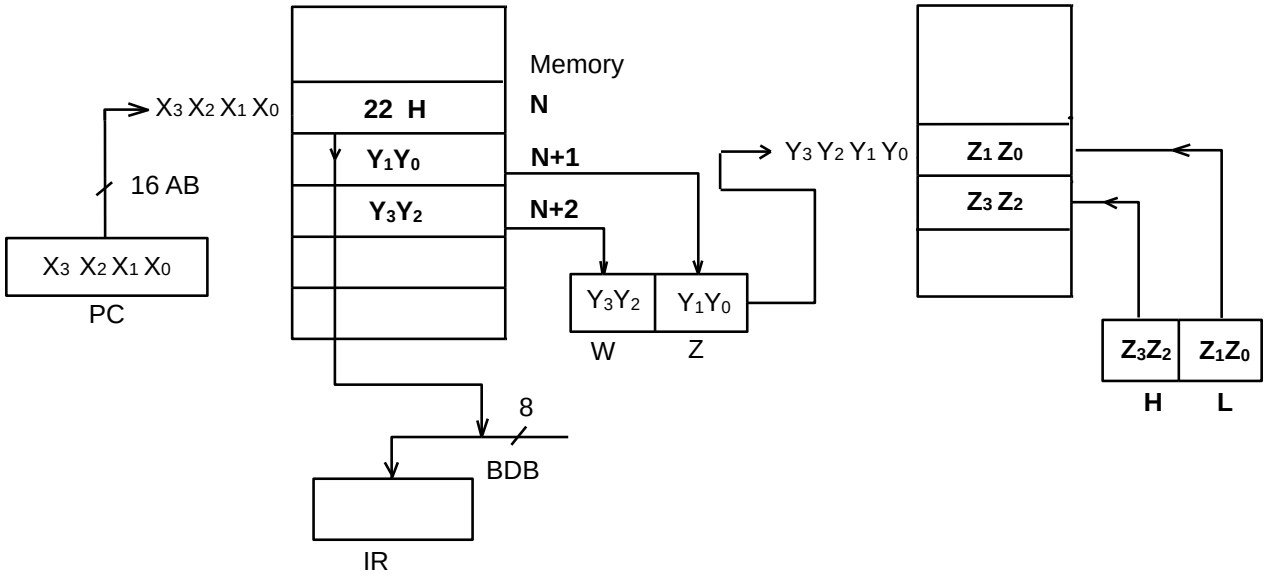


Fig.5.10 Data Flow during the Execution of SHLD Addr Instruction

The micro RTL flow is:

Machine Cycle- 1:

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = ⎍

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (IR) ⟵ AD_7-AD_0

T_4: μp decodes the opcode. SHLD addr = 1.

Machine Cycle- 2:

MRMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=0

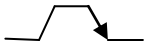
T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = ⎍

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (Z) ⟵ AD_7-AD_0

Machine Cycle- 3:

MRMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=0

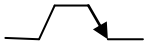
T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (W) ⟵ AD_7-AD_0

Machine Cycle- 4:

MWRMC: Status signals IO/$\overline{M}$=0, S_1=0, S_0=1

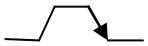
T_1: A_{15}-A_8 ⟵ (W), AD_7-AD_0 ⟵ (Z), ALE = 

T_2: $\overline{WR}$ = 0, AD_7-AD_0 ⟵ (L)

T_3: $\overline{WR}$ = 1, ↑ , M(AB) ⟵ AD_7-AD_0

Machine Cycle- 5:

MWRMC: Status signals IO/$\overline{M}$=0, S_1=0, S_0=0

T_1: A_{15}-A_8 ⟵ (W), AD_7-AD_0 ⟵ (Z+1), ALE = 

T_2: $\overline{WR}$ = 0, AD_7-AD_0 ⟵ (H)

T_3: $\overline{WR}$ = 1, ↑ , M(AB) ⟵ AD_7-AD_0

The instruction requires 5 machine cycles- OFMC, two MRMC & two MWRMC and total number of 16 states. It needs 8.0μsec using 2 MHz internal clock.

11. LDAX rp: This is an ALP statement. LDAX is the mnemonic for LOAD ACCUMULATOR INDIRECTLY. As discussed earlier, the alphabet ‘X’ in the mnemonic tells that a register pair is involved in the instruction. ‘rp’ stands for register pair. The meaning of the

instruction is “Load the accumulator from the memory location whose address is available in a register pair specified in the instruction”. Only one operand in involved in this instruction & the operand is available in the memory location whose address is in a register pair. The macro RTL implemented is

(A) ⟵ M(rpH, rpL)

This is a single byte instruction. The instruction format is

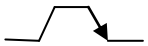
0	0	R	P	1	0	1	0

N

There are two variations of this instruction RP = 00 for (B, C) register pair and RP= 01 for (D, E) register pair. Note that RP = 10 and 11 are not allowed in this instruction. The micro RTL is

Machine Cycle- 1:

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

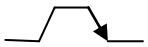
T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (IR) ⟵ AD_7-AD_0

T_4: μp decodes the opcode. LDAX rp = 1.

Machine Cycle- 2:

MRMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=0

T_1: A_{15}-A_8 ⟵ (rpH), AD_7-AD_0 ⟵ (rpL), ALE = 

T_2: $\overline{RD}$ = 0, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (A) ⟵ AD_7-AD_0

This instruction requires 2 machine cycles OFMC & MRMC and 7 states. It needs 3.5 µsec using 2MHz clock. The addressing mode is register indirect addressing mode.

12. STAX rp: This is an ALP statement. STAX is the mnemonic for store accumulator indirectly using register indirect addressing mode. The meaning of the instruction is “Store the content of accumulator in the memory location whose address is available in register pair specified in the instruction”. The macro RTL implemented is

$$M(rpH, rpL) \longleftarrow (A)$$

This is a single byte instruction. The instruction format is

0	0	R	P	0	0	1	0	N

This instruction also has only two variations. RP = 00 and 01 are allowed for register pair (B, C) and (D, E) respectively. The micro RTL flow is

Machine Cycle- 1:

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = ⎍

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (IR) ⟵ AD_7-AD_0

T_4: μp decodes the opcode. LDAX rp = 1.

Machine Cycle- 2:

MWRMC: Status signals IO/$\overline{M}$=0, S_1=0, S_0=1

T_1: A_{15}-A_8 ⟵ (rpH), AD_7-AD_0 ⟵ (rpL), ALE = ⎍

T_2: $\overline{WR}$ = 0, AD_7-AD_0 ⟵ (A)

T_3: $\overline{WR}$ = 1, ↑ , M(AB) ⟵ AD_7-AD_0

This instruction requires 2 machine cycles OFMC & MWRMC and 7 states. It needs 3.5 µsec using 2MHz clock. The addressing mode is register indirect addressing mode.

Lecture-23

IN PORT & OUT PORT:

Data is transferred from the microprocessor to an output device in an output transfer and from an input device to the microprocessor in an input transfer.

Output devices contain one or more register each of which is addressable and strobed by the microprocessor to latch the data on the data bus at the appropriate time during an output transfer. Input devices contain a register that holds the data to be transferred to the microprocessor and an addressable three-state buffer. The buffer is enabled by a strobe from the microprocessor at the appropriate time to place data on data bus.

The 8085 A CPU outputs a signal IO/$\overline{M}$ which is '1' when an I/O port is being accessed and '0' when memory is being accessed. This can be used to distinguish I/O device & memory read & write operations. If IO/$\overline{M}$ =1 is used to select any I/O port then it is known as I/O mapped I/O structure or isolated I/O or standard I/O. If IO/$\overline{M}$ =0 is used to select any I/O port then decoding circuit will not be able to distinguish I/O device & memory. An I/O port will then be treated as just another memory location. This is referred to as memory mapped I/O. In this case, same of the 64 k addressable locations are used to interface the I/O device & not memory.

13. IN PORT: This is an ALP statement. PORT is the symbolic name given to 8-bit address of the input device available as a 2^{nd} byte of the

instruction. IN is the mnemonic for INPUT. The meaning of the instruction is “Input an 8-bit data from the input device whose address is available as a 2^{nd} byte of the instruction and load it into accumulator”. The IN PORT is a 2- byte instruction, the first byte is the opcode and the second byte is the 8-bit input device address. The instruction format is

1	1	0	1	1	0	1	1	N
$<B_2>$								N+1

The macro RTL implemented is,

(A) ⟵ I/O (PORT)

or (A) ⟵ I/O ($<B_2>$)

The necessary interfering circuitry is shown in fig.5.11 for inputting an 8-bit data from an input device whose port address is B_2.

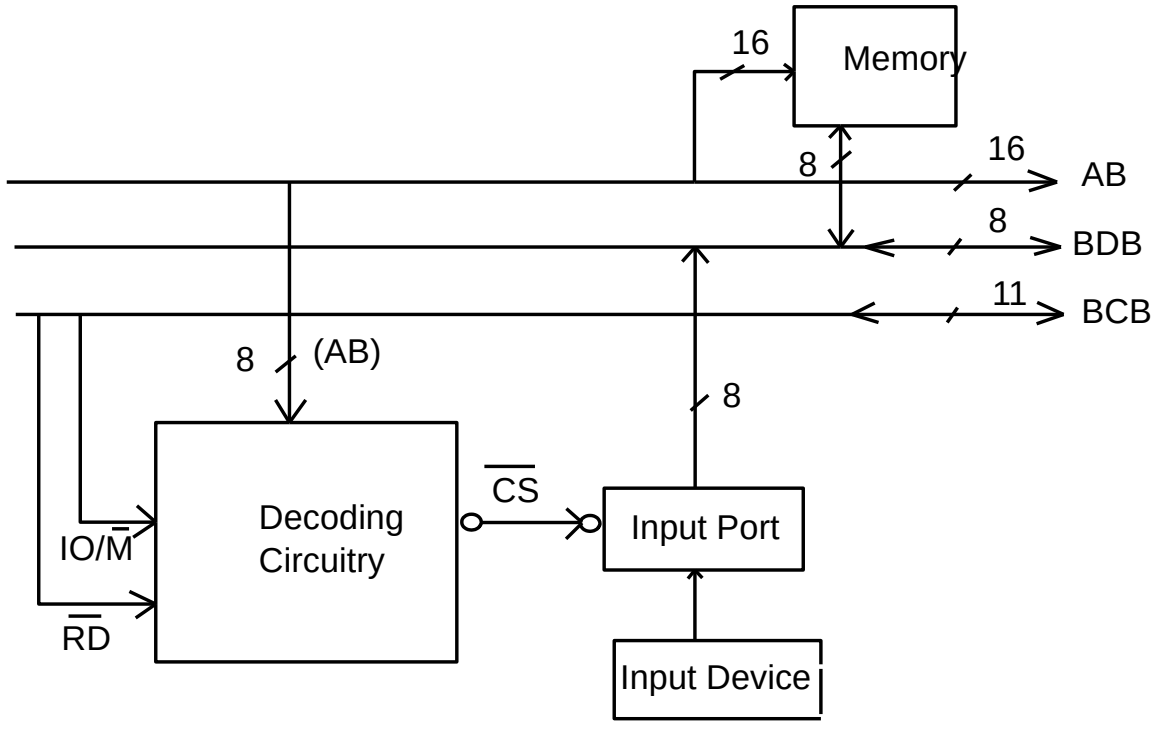


Fig.5.11 Schematic Diagram of Interfacing Circuitry for Input Device

The data generated by an input device is stored temporarily in some register which can be used by the microprocessor when necessary. In order to place the contents of the input register on the microprocessor system bus, the outputs are connected to the data bus through a three, state buffer or input port. During execution of the IN PORT instruction, μp places an 8-bit device address at the address bus, where it is duplicated on higher & lower order bus. An external decoder decodes the device address, the IO/$\overline{M}$, & the $\overline{RD}$ pulse in order to generate an input device select pulse. The device select pulse enable the 3-state buffer of the addressed input port, thus placing the input port's data on the data bus.

Whenever an input device is interface as shown in figure, the data is inputted using IN PORT instruction. It is known as I/O mapped I/O structure or isolated I/O structure. The address of an input device, being 8-bit, may vary from 00_H to FF_H. Thus, a total of 256 input devices can be connected directly, through isolate I/O structure, where the source register is identified by an explicit 8-bit address, the corresponding I/O structure is shown by the map.

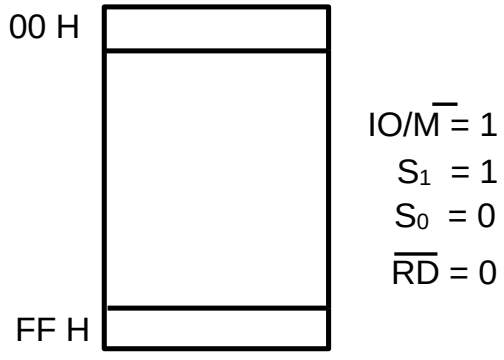
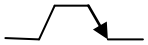


Fig.5.11 Isolated I/O Structure for Input Ports

The micro RTL flow for executing this instruction is as follows:

Machine Cycle- 1:

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

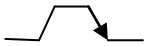
T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (IR) ⟵ AD_7-AD_0

T_4: μp decodes the opcode. IN PORT = 1.

Machine Cycle- 2:

MRMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=0

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

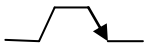
T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (Z) ⟵ AD_7-AD_0, (W) ⟵ AD_7-AD_0

Thus at the end of 2nd machine cycle W, Z registers together contain (Y_1Y_0, Y_1Y_0), where Y_1Y_0 is the address of the input device available in the instruction as 2nd byte.

Machine Cycle- 3:

IORMC: Status signals IO/$\overline{M}$=1, S_1=1, S_0=0

T_1: A_{15}-A_8 ⟵ (W), AD_7-AD_0 ⟵ (Z), ALE = 

T_2: $\overline{RD}$ = 0, AD_7-AD_0 ⟵ IO(AB)

T_3: $\overline{RD}$ = 1, ↑ , (A) ⟵ AD_7-AD_0

This instruction requires 3 machine cycles – OFMC, MRMC & IORMC and 10 states. It needs 5.0 µsec using 2MHz clock. The addressing mode is direct addressing mode.

14. OUT PORT: This is an ALP statement. Figure gives the interfacing circuitry for outputting an 8-bit data to an output device using OUT PORT instruction.

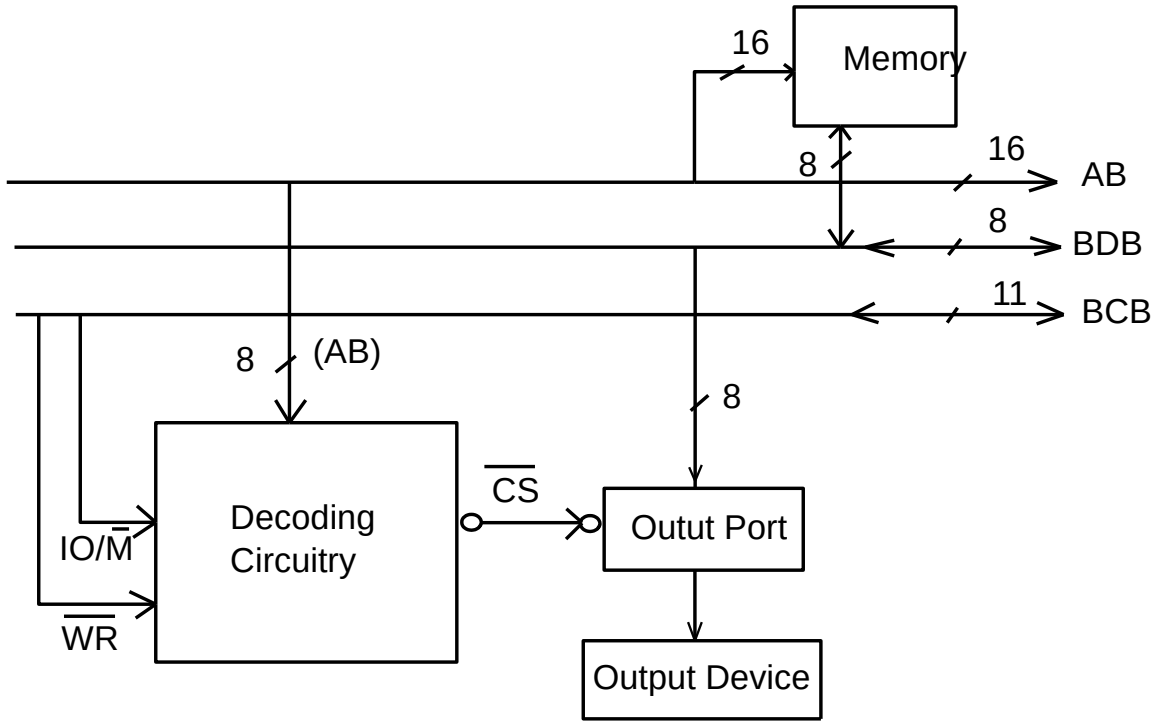


Fig.5.13 Schematic Diagram of Interfacing Circuitry for Output Device

This is a 2 byte instruction. PORT is the symbolic name given to the 8-bit address of the output device available as the 2nd byte of instruction. The instruction format is

1	1	0	1	0	0	1	1	N
<B_2>								N+1

The meaning of the instruction is “Output the content of accumulator to an output device whose address is available in the instruction as the 2nd byte of the instruction”. The macro RTL implemented is,

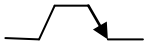
I/O (PORT) ⟵ (A)

or I/O (<B_2>) ⟵ (A)

It is direct addressing mode and has no variations. The micro RTL flow for executing this instruction is as follows:

Machine Cycle- 1:

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

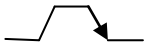
T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (IR) ⟵ AD_7-AD_0

T_4: μp decodes the opcode. OUT PORT = 1.

Machine Cycle- 2:

MRMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=0

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

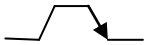
T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (Z) ⟵ AD_7-AD_0, (W) ⟵ AD_7-AD_0

In this case also the 8-bit address of the output device is duplicated in W & Z registers.

Machine Cycle- 3:

IOWRMC: Status signals IO/$\overline{M}$=1, S_1=0, S_0=1

T_1: A_{15}-A_8 ⟵ (W), AD_7-AD_0 ⟵ (Z), ALE = 

T_2: $\overline{WR}$ = 0, AD_7-AD_0 ⟵ (A)

T_3: $\overline{WR}$ = 1, ↑ , IO(AB) ⟵ AD_7-AD_0

This instruction requires 3 machine cycles – OFMC, MRMC & IOWRMC and a total of 10 states. It needs 5.0 µsec using 2MHz clock. If the output device is interfaced for outputting the data using this instruction then the structure is known as isolated I/O structure

I/O structure or I/O map I/O structure. The address of an output device may vary from 00_H to FF_H. Thus, a total of 256 output devices can be connected directly, through isolate I/O structure, where the source register is identified by an explicit 8-bit address. The corresponding I/O structure is shown in fig.5.14 by the map.

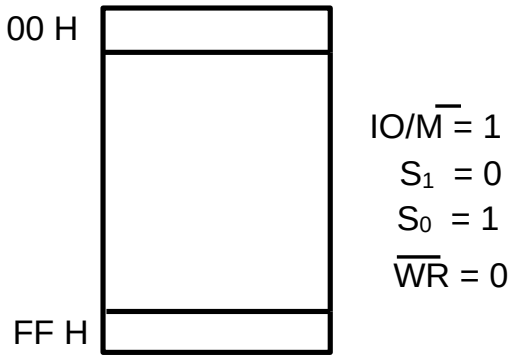


Fig.5.14 Isolated I/O Structure for Output Ports

15. XCHG: This is an ALP statement. This is a single byte instruction. The meaning of the instruction is “Exchanged the contents of (H, L) register pair with the contents of (D,E) pair”. The macro RTL implemented is

(H) ⟷ (D)

(L) ⟷ (E)

Or (H,L) ⟷ (D,E)

The single byte operation code is

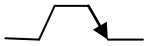
1	1	1	0	1	0	1	1

N

The micro RTL flow is given below:

Machine Cycle- 1:

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

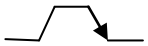
T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (IR) ⟵ AD_7-AD_0

T_4: μp decodes the opcode. XCHG = 1.

Machine Cycle- 2 or (Machine Cycle-1 of next instruction):

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

FEO [(H) ⟷ (D)]

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

FEO [(L) ⟷ (E)]

Thus, this instruction requires only 1 machine cycle – OFMC and a total of 4 states. It needs 2.0 µsec using 2MHz clock.

Lecture-24

ARITHMETIC GROUP:

The instructions of this group perform the arithmetic operations on the operands. Normally two operands are necessary for any arithmetic operation. One of the operand is always assumed to be available in accumulator. The other operand can be made available in one of the three locations:

(a) In an internal general purpose register (r).

(b) In a memory location pointed by M-pointer i.e., (H, L) pair.

(c) Immediately in the instruction itself as a 2nd byte.

All the flags of Flag register are affected as per standard rule. There are 20 basic instructions in this group. The format of the operation code is shown in fig5.15.

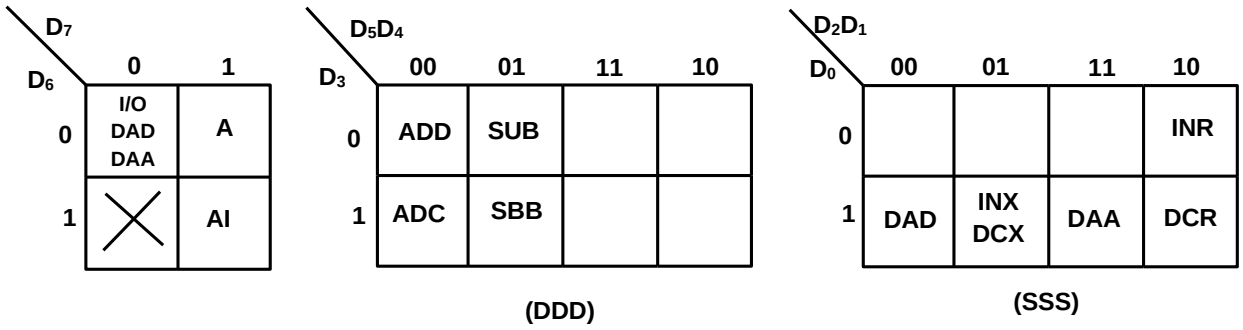


Fig.5.15 Format of Arithmetic Group Instructions

1. ADD r: This is a single byte instruction. The meaning of the instruction is “Add the content of register (r) to the content of accumulator and store the result back into the accumulator”. The macro RTL implemented is

$$(A) \longleftarrow (A) + (r)$$

The opcode of the instruction is,

1	0	0	0	0	S	S	S

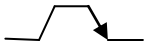
N

It has 7 variations for 7 internal general purpose registers.

The micro RTL flow is given below:

Machine Cycle- 1:

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

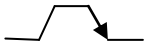
T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (IR) ⟵ AD_7-AD_0

T_4: μp decodes the opcode. ADD r = 1.

Machine Cycle- 2 or (Machine Cycle-1 of next instruction):

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

FEO [(Temp) ⟵ (r)]

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

FEO [(A) ⟵ (Temp) + (A)]

It requires single machine cycle OFMC and 4 states. The addressing mode is register addressing mode.

2. ADD M: This is a single byte instruction. The meaning of the instruction is “Add the content of memory location whose address is in (H,L) pair to the content of accumulator and store the result back into the accumulator”. The macro RTL implemented is

(A) ⟵ (A) + M(H,L)

The opcode of the instruction is,

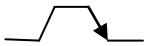
1	0	0	0	0	1	1	0

N

It has no variations. Register indirect addressing mode is used in this instruction. The micro RTL flow is given below:

Machine Cycle- 1:

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ← (PCH), AD_7-AD_0 ← (PCL), ALE = 

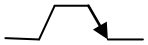
T_2: $\overline{RD}$ = 0, (PC) ← (PC) +1, AD_7-AD_0 ← M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (IR) ← AD_7-AD_0

T_4: μp decodes the opcode. ADD M = 1.

Machine Cycle- 2:

MRMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=0

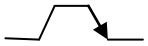
T_1: A_{15}-A_8 ← (H), AD_7-AD_0 ← (L), ALE = 

T_2: $\overline{RD}$ = 0, AD_7-AD_0 ← M(AB)

T_3: $\overline{RD}$ = 1, ↑, (Temp) ← AD_7-AD_0

Machine Cycle- 3 or (Machine Cycle-1 of next instruction):

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ← (PCH), AD_7-AD_0 ← (PCL), ALE = 

T_2: $\overline{RD}$ = 0, (PC) ← (PC) +1, AD_7-AD_0 ← M(AB)

FEO [(A) ← (Temp) + (A)]

It requires two machine cycles OFMC & MRMC and 7 states.

3. ADI DATA: It is a two byte instruction. The meaning of the instruction is "Add the content available as the second byte of the instruction to the content of accumulation and store the result back into the accumulator".

The opcode of the instruction is,

1	1	0	0	0	1	1	0	N
$<B_2>$								N+1

It has no variations. The macro RTL implemented is

$$(A) \longleftarrow (A) + <B_2>$$

It has immediate addressing mode. The micro RTL flow is given below:

Machine Cycle- 1:

OFMC: Status signals $IO/\overline{M}=0$, $S_1=1$, $S_0=1$

T_1: A_{15}-A_8 ← (PCH), AD_7-AD_0 ← (PCL), ALE = ⎍

T_2: $\overline{RD}$ = 0, (PC) ← (PC) +1, AD_7-AD_0 ← M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (IR) ← AD_7-AD_0

T_4: μp decodes the opcode. ADI data = 1.

Machine Cycle- 2:

MRMC: Status signals $IO/\overline{M}=0$, $S_1=1$, $S_0=0$

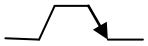
T_1: A_{15}-A_8 ← (PCH), AD_7-AD_0 ← (PCL), ALE = ⎍

T_2: $\overline{RD}$ = 0, (PC) ← (PC) +1, AD_7-AD_0 ← M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (Temp) ← AD_7-AD_0

Machine Cycle- 3 or (Machine Cycle-1 of next instruction):

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

FEO [(A) ⟵ (Temp) + (A)]

It requires two machine cycles OFMC & MRMC and 7 states.

4. ADC r: It is a single byte instruction. The meaning of the instruction is "Add the content of register (r) to the content of accumulator with carry and store the result back into accumulator". The macro RTL implemented is

(A) ⟵ (A) + (r) + (CY)

The opcode of the instruction is,

1	0	0	0	1	S	S	S

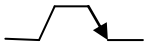
N

It has 7 variations for 7 internal general purpose registers.

The micro RTL flow is given below:

Machine Cycle- 1:

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

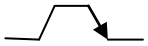
T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (IR) ⟵ AD_7-AD_0

T_4: μp decodes the opcode. ADC r = 1.

Machine Cycle- 2 or (Machine Cycle-1 of next instruction):

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

FEO [(Temp) ⟵ (r)]

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

FEO [(A) ⟵ (Temp) + (A) + (CY)]

It requires single machine cycle OFMC and 4 states. The addressing mode is register addressing mode.

5. ADC M: It is a single byte instruction. The meaning of the instruction is "Add the contents of memory location whose address is available in (H, L) pair to the contents of accumulator with carry and store the result back into accumulator". The macro RTL implemented is

(A) ⟵ (A) + M(H,L) + (CY)

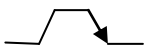
The opcode of the instruction is,

1	0	0	0	1	1	1	0	N

It has no variations. The addressing mode is register indirect addressing mode. The micro RTL flow is given below:

Machine Cycle- 1:

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

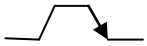
T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (IR) ⟵ AD_7-AD_0

T_4: μp decodes the opcode. ADC M = 1.

Machine Cycle- 2:

MRMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=0

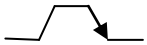
T_1: A_{15}-A_8 ⟵ (H), AD_7-AD_0 ⟵ (L), ALE = 

T_2: $\overline{RD}$ = 0, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑, (Temp) ⟵ AD_7-AD_0

Machine Cycle- 3 or (Machine Cycle-1 of next instruction):

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

FEO [(A) ⟵ (Temp) + (A) + (CY)]

It requires two machine cycles OFMC & MRMC and 7 states.

6. ACI DATA: It is a two byte instruction. The meaning of the instruction is "Add the content available as the second byte of instruction itself to the content to accumulator with carry and store the result back into the accumulator". The opcode of the instruction is

1	1	0	0	1	1	1	0	N
<B_2>								N+1

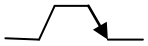
The macro RTL implemented is

(A) ⟵ (A) + <B_2> + (CY)

It has no variations. It has immediate addressing mode. The micro RTL flow is

Machine Cycle- 1:

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

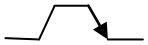
T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (IR) ⟵ AD_7-AD_0

T_4: μp decodes the opcode. ACI data = 1.

Machine Cycle- 2:

MRMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=0

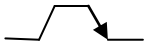
T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (Temp) ⟵ AD_7-AD_0

Machine Cycle- 3 or (Machine Cycle-1 of next instruction):

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

FEO [(A) ⟵ (Temp) + (A) + (CY)]

It requires two machine cycles OFMC & MRMC and 7 states.

Note: In all these above instruction all the flags are affected as per rule.

Lecture-25

7. SUB r: This is a single byte instruction. The meaning of the instruction is "Subtract the content of register (r) from the content of accumulator and store the result back into the accumulator". The macro RTL implemented is

$$(A) \longleftarrow (A) - (r)$$

The opcode of the instruction is,

1	0	0	1	0	S	S	S	N

It has 7 variations for 7 internal general purpose registers.

The micro RTL flow is given below:

Machine Cycle- 1:

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = ⎍

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (IR) ⟵ AD_7-AD_0

T_4: μp decodes the opcode. SUB r = 1.

Machine Cycle- 2 or (Machine Cycle-1 of next instruction):

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = ⎍

FEO [(Temp) ⟵ (r)]

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

FEO [(A) ⟵ (A) - (Temp)]

It requires single machine cycle OFMC and 4 states. The addressing mode is register addressing mode.

8. SUB M: This is a single byte instruction. The meaning of the instruction is “Subtract the content of memory location whose address is in (H,L) pair from the content of accumulator and store the result back into the accumulator”. The macro RTL implemented is

(A) ⟵ (A) - M(H,L)

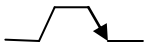
The opcode of the instruction is,

1	0	0	1	0	1	1	0	N

It has no variations. Register indirect addressing mode is used in this instruction. The micro RTL flow is given below:

Machine Cycle- 1:

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

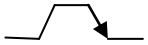
T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1,  , (IR) ⟵ AD_7-AD_0

T_4: μp decodes the opcode. SUB M = 1.

Machine Cycle- 2:

MRMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=0

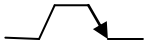
T_1: A_{15}-A_8 ⟵ (H), AD_7-AD_0 ⟵ (L), ALE = 

T_2: $\overline{RD}$ = 0, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1,  , (Temp) ⟵ AD_7-AD_0

Machine Cycle- 3 or (Machine Cycle-1 of next instruction):

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

FEO [(A) ⟵ (A) - (Temp)]

It requires two machine cycles OFMC & MRMC and 7 states.

9. SUI DATA: It is a two byte instruction. The meaning of the instruction is "Subtract the content available as the second byte of the instruction from the content of accumulation and store the result back into the accumulator".

The opcode of the instruction is,

1	1	0	1	0	1	1	0	N
<B_2>								N+1

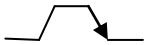
It has no variations. The macro RTL implemented is

(A) ⟵ (A) - <B_2>

It has immediate addressing mode. The micro RTL flow is given below:

Machine Cycle- 1:

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

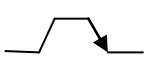
T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (IR) ⟵ AD_7-AD_0

T_4: μp decodes the opcode. SUI data = 1.

Machine Cycle- 2:

MRMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=0

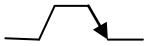
T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (Temp) ⟵ AD_7-AD_0

Machine Cycle- 3 or (Machine Cycle-1 of next instruction):

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

FEO [(A) ⟵ (A) - (Temp)]

It requires two machine cycles OFMC & MRMC and 7 states.

10. SBB r: It is a single byte instruction. The meaning of the instruction is "Subtract the content of register (r) from the content of accumulator with borrow and store the result back into accumulator". The macro RTL implemented is

(A) ⟵ (A) - (r) - (CY)

The opcode of the instruction is,

1	0	0	1	1	S	S	S

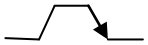
N

It has 7 variations for 7 internal general purpose registers.

The micro RTL flow is given below:

Machine Cycle- 1:

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

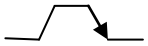
T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (IR) ⟵ AD_7-AD_0

T_4: μp decodes the opcode. SBB r = 1.

Machine Cycle- 2 or (Machine Cycle-1 of next instruction):

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

FEO [(Temp) ⟵ (r)]

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

FEO [(A) ⟵ (A) - (Temp) - (CY)]

It requires single machine cycle OFMC and 4 states. The addressing mode is register addressing mode.

11. SBB M: It is a single byte instruction. The meaning of the instruction is "Subtract the content of memory location whose address is available in (H, L) pair from the content of accumulator with borrow and store the result back into accumulator". The macro RTL implemented is

(A) ⟵ (A) - M(H,L) - (CY)

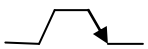
The opcode of the instruction is,

1	0	0	1	1	1	1	0	N

It has no variations. The addressing mode is register indirect addressing mode. The micro RTL flow is given below:

Machine Cycle- 1:

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

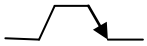
T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (IR) ⟵ AD_7-AD_0

T_4: μp decodes the opcode. SBB M = 1.

Machine Cycle- 2:

MRMC: Status signals $IO/\overline{M}=0$, $S_1=1$, $S_0=0$

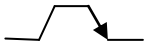
T_1: A_{15}-A_8 ⟵ (H), AD_7-AD_0 ⟵ (L), ALE = 

T_2: $\overline{RD}$ = 0, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑, (Temp) ⟵ AD_7-AD_0

Machine Cycle- 3 or (Machine Cycle-1 of next instruction):

OFMC: Status signals $IO/\overline{M}=0$, $S_1=1$, $S_0=1$

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

FEO [(A) ⟵ (A) - (Temp) - (CY)]

It requires two machine cycles OFMC & MRMC and 7 states.

12. SBI DATA: It is a two byte instruction. The meaning of the instruction is "Subtract the content available as the second byte of instruction itself from the content to accumulator with borrow and store the result back into the accumulator". The opcode of the instruction is

1	1	0	1	1	1	1	0	N
<B_2>								N+1

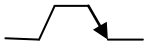
The macro RTL implemented is

(A) ⟵ (A) - <B_2> - (CY)

It has no variations. It has immediate addressing mode. The micro RTL flow is

Machine Cycle- 1:

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

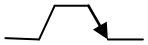
T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (IR) ⟵ AD_7-AD_0

T_4: μp decodes the opcode. SBI data = 1.

Machine Cycle- 2:

MRMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=0

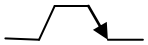
T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (Temp) ⟵ AD_7-AD_0

Machine Cycle- 3 or (Machine Cycle-1 of next instruction):

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

FEO [(A) ⟵ (A) - (Temp) - (CY)]

It requires two machine cycles OFMC & MRMC and 7 states.

Note: In all these above instruction all the flags are affected as per rule.

13. INR r: It is a single byte instruction. The meaning of the instruction is “Increment the content of register (r) by 1 and store the result back to the register (r)”. The macro RTL implemented is

(r) ⟵ (r) + 1

The opcode of the instruction is

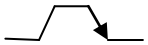
0	0	D	D	D	1	0	0

N

It has 7 variations in this all the flags are affected except carry flag. The micro RTL flow implemented is given below:

Machine Cycle- 1:

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

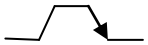
T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (IR) ⟵ AD_7-AD_0

T_4: μp decodes the opcode. INR r = 1.

Machine Cycle- 2 or (Machine Cycle-1 of next instruction):

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

FEO [(Temp) ⟵ (r)]

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

FEO [(r) ⟵ (Temp) + 1]

It requires single machine cycle OFMC and 4 states. The addressing mode is register addressing mode. All flags are affected except CY flag.

14. INR M: It is a single byte instruction. The meaning of the instruction is “Increment the content of memory location by 1 whose address is available in (H, L) pair and stores the result back in the same location”. The macro RTL implemented is

M(H,L) ⟵ M(H,L) + 1

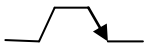
The opcode of the instruction is

0	0	1	1	0	1	0	0	N

It has no variations. All the flags are affected except carry flag. The micro RTL flow implemented is given below:

Machine Cycle- 1:

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

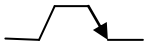
T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (IR) ⟵ AD_7-AD_0

T_4: μp decodes the opcode. INR M = 1.

Machine Cycle- 2:

MRMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=0

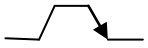
T_1: A_{15}-A_8 ⟵ (H), AD_7-AD_0 ⟵ (L), ALE = 

T_2: $\overline{RD}$ = 0, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑, (Temp) ⟵ AD_7-AD_0

Machine Cycle- 3

MWRMC: Status signals IO/$\overline{M}$=0, S_1=0, S_0=1

T_1: A_{15}-A_8 ⟵ (H), AD_7-AD_0 ⟵ (L), ALE = 

T_2: $\overline{WR}$ = 0, AD_7-AD_0 ⟵ (Temp) + 1

T_3: $\overline{WR}$ = 1, ↑ , M(AB) ⟵ AD_7-AD_0

It requires three machine cycles OFMC, MRMC & MWRMC, and 10 states. The addressing mode is register addressing mode. All flags are affected except CY flag.

Lecture-26

15. DCR r: It is a single byte instruction. The meaning of the instruction is "Decrement the content of register (r) by 1 and store the result back to the register (r)". The macro RTL implemented is

$$(r) \longleftarrow (r) + 1$$

The opcode of the instruction is

0	0	D	D	D	1	0	1	N

It has 7 variations in this all the flags are affected except carry flag. The micro RTL flow implemented is given below:

Machine Cycle- 1:

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = ⎍

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (IR) ⟵ AD_7-AD_0

T_4: μp decodes the opcode. DCR r = 1.

Machine Cycle- 2 or (Machine Cycle-1 of next instruction):

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = ⎍

FEO [(Temp) ⟵ (r)]

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

FEO [(r) ⟵ (Temp) - 1]

It requires single machine cycle OFMC and 4 states. The addressing mode is register addressing mode. All flags are affected except the CY flag.

16. DCR M: It is a single byte instruction. The meaning of the instruction is “Decrement the content of memory location by 1 whose address is available in (H, L) pair and stores the result back in the same location”. The macro RTL implemented is

$$M(H,L) \longleftarrow M(H,L) + 1$$

The opcode of the instruction is

0	0	1	1	0	1	0	1	N

It has no variations. All the flags are affected except carry flag. The micro RTL flow implemented is given below:

Machine Cycle- 1:

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = ⎍

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (IR) ⟵ AD_7-AD_0

T_4: μp decodes the opcode. DCR M = 1.

Machine Cycle- 2:

MRMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=0

T_1: A_{15}-A_8 ⟵ (H), AD_7-AD_0 ⟵ (L), ALE = ⎍

T_2: $\overline{RD}$ = 0, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑, (Temp) ⟵ AD_7-AD_0

Machine Cycle- 3

MWRMC: Status signals IO/$\overline{M}$=0, S_1=0, S_0=1

T_1: A_{15}-A_8 ⟵ (H), AD_7-AD_0 ⟵ (L), ALE = ⎍

T_2: $\overline{WR}$ = 0, AD_7-AD_0 ⟵ (Temp) + 1

T_3: $\overline{WR}$ = 1, ↑ , M(AB) ⟵ AD_7-AD_0

It requires three machine cycles OFMC, MRMC & MWRMC, and 10 states. The addressing mode is register addressing mode. All flags are affected except CY.

17. INX rp: It is a single byte instruction. The meaning of the instruction is "Increment the content of register pair (rp) by 1 and store it in same register pair. The macro RTL implemented is

(rp) ⟵ (rp) + 1

or (rpH, rpL) ⟵ (rpH, rpL) + 1

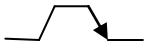
The opcode of the instruction is

0	0	R	P	0	0	1	1	N

It has 4 variations for register pairs (B,C), (D,E), (H,L) and 16-bit register (SP). No flag is affected in this instruction. It is register addressing mode. The micro RTL flow implemented is given below:

Machine Cycle- 1:

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (IR) ⟵ AD_7-AD_0

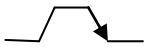
T_4: μp decodes the opcode. INX rp = 1.

T_5: (Temp) ⟵ (rpL), (Z) ⟵ (F)

T_6: (rpL) ⟵ (Temp) + 1, CY Flag is affected.

Machine Cycle- 2 or (Machine Cycle-1 of next instruction):

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

FEO [(Temp) ⟵ (rpH)]

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

FEO [(rpH) ⟵ (Temp) + (CY), (F) ⟵ (Z)]

Therefore, the instruction requires only one machine cycle OFMC and of 6 states.

18. DCX rp: It is a single byte instruction. The meaning of the instruction is "Decrement the content of register pair (rp) by 1 and store it in same register pair. The macro RTL implemented is

(rp) ⟵ (rp) - 1

or (rpH, rpL) ⟵ (rpH, rpL) - 1

The opcode of the instruction is

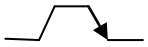
0	0	R	P	1	0	1	1

N

It has 4 variations for register pairs (B,C), (D,E), (H,L) and 16-bit register (SP). No flag is affected in this instruction. It is register addressing mode. The micro RTL flow implemented is given below:

Machine Cycle- 1:

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (IR) ⟵ AD_7-AD_0

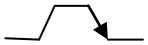
T_4: μp decodes the opcode. INX rp = 1.

T_5: (Temp) ⟵ (rpL), (Z) ⟵ (F)

T_6: (rpL) ⟵ (Temp) - 1, CY Flag is affected.

Machine Cycle- 2 or (Machine Cycle-1 of next instruction):

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

FEO [(Temp) ⟵ (rpH)]

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

FEO [(rpH) ⟵ (Temp) - (CY), (F) ⟵ (Z)]

Therefore, the instruction requires only one machine cycle OFMC and of 6 states.

19. DAD rp: It is a single byte instruction. The meaning of the instruction is double precision addition, i.e., "Add the contents of (H,L) register pair with the contents of register pair (rp) and store the result back into (H,L) register pair". The macro RTL implemented is

(H, L) ⟵ (H, L) + (rp)

or (H, L) ⟵ (H, L) + (rpH, rpL)

The opcode of the instruction is

0	0	R	P	1	0	0	1

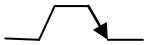
N

It has 4 variations for register pairs (B,C), (D,E), (H,L) and 16-bit register (SP). Only CY flag is affected in this instruction. It is register addressing mode. Since the instruction is a single byte instruction, therefore, only one memory reference operation is required. To add two 16-bit numbers, two 8-bit operations are required one by one which need two machine cycles. These operations cannot be completed during FEO (where only two extra states are available)

and, therefore, two Bus Idle Machine Cycles (BIMC) are required. The micro RTL flow implemented is given below:

Machine Cycle- 1:

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (IR) ⟵ AD_7-AD_0

T_4: μp decodes the opcode. DAD rp = 1.

Machine Cycle- 2:

BIMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=0

T_1: (W) ⟵ (A), (Z) ⟵ (F)

T_2: (A) ⟵ (rpL), (Temp) ⟵ (L)

T_3: (L) ⟵ (L) + (Temp), CY flag is affected as per result

Machine Cycle- 3:

BIMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=0

T_1: (A) ⟵ (rpH), (Temp) ⟵ (H)

T_2: (H) ⟵ (A) + (Temp) + (CY)

T_3: (A) ⟵ (W), (F) ⟵ (Z), All flags are restored except CY.

Therefore, the instruction requires three machine cycles OFMC & teo BIMC and a total of 10 states.

DAD H is one important instruction. The macro RTL implemented is

(H, L) ⟵ (H, L) + (H, L)

The meaning of this multiplying the 16-bit contents of (H,L) by 2. It actually shifts all the 16 bits of (H, L) register pair to left by one bit position.

20. DAA: It is a single byte instruction. The meaning of the instruction is decimal adjust accumulator. It is specifically used for BCD addition operation. It adjusts the content of accumulator to two digit BCD. It performs the following operations on 8-bit data in the accumulator.

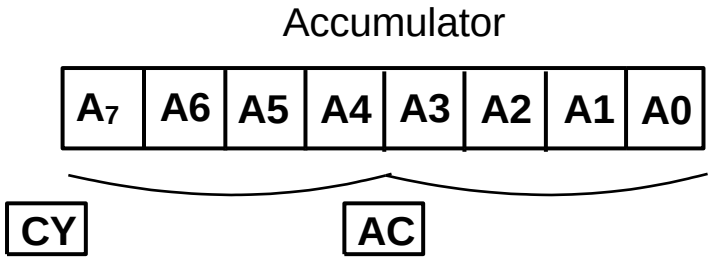


Fig.5.16 CY Flag, AC Flag and Both Nibbles checked for DAA Instruction

This instruction is used just after the addition operation. Whenever DAA is executed the following checks and corrections are made:

A. If the lower order 4-bits $A_3A_2A_1A_0$ is an illegal BCD code or if AC is SET then 06_H is added to accumulator.

B. Thereafter, if the higher order 4-bits $A_7A_6A_5A_4$ is an illegal BCD code or if CY is SET then 60_H is added to accumulator else no action.

The opcode of the instruction is

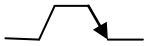
0	0	1	0	0	1	1	1

N

It has no variations. The micro RTL flow is

Machine Cycle- 1:

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

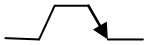
T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1,  , (IR) ⟵ AD_7-AD_0

T_4: μp decodes the opcode. DAA = 1.

Machine Cycle- 2 or (Machine Cycle-1 of next instruction):

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

FEO [Checks the lower nibble of (A) and adjusts it to valid BCD]

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

FEO[Checks the upper nibble of (A) and adjusts it to valid BCD]

Therefore, the instruction requires only one machine cycle OFMC and a total of 4 states.

Lecture-27

LOGICAL GROUP:

Fig.5.17 gives the operation code format for logical group. It consists of 19 basic instructions. There are three basic logic operation AND, OR & XOR. Logical operation takes place bit by bit. If two operands are involved in the instruction, the first operand is assumed to be in the accumulator. The second 8-bit operand is available either in the internal general purpose register or in the memory location pointed to by the M-pointer or as the 2nd byte of the instruction itself. The result of the logical operation is stored back in the accumulator.

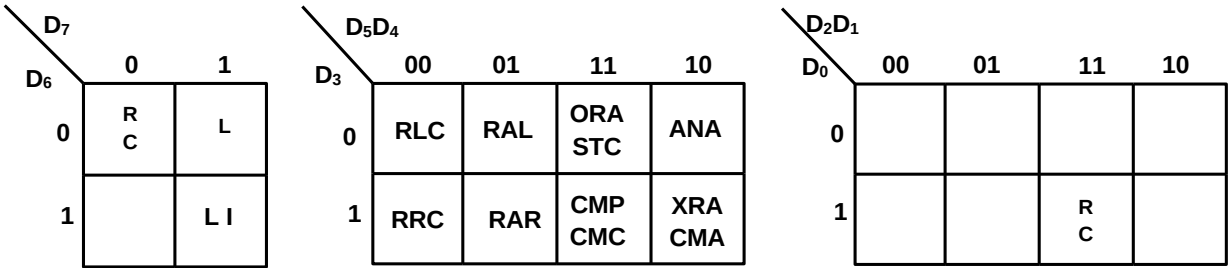


Fig.5.17 Format of Logical Group Instructions

1. ANA r: This is a single byte instruction. The meaning of the instruction is "AND bit by bit the content of register (r) to the content of accumulator and store the result back in the accumulator". The macro RTL implemented is

$$(A_i) \longleftarrow (A_i) \cap (r_i) \quad i = 0 \text{ to } 7$$

$$(A) \longleftarrow (A) \cap (r)$$

The opcode of the instruction is

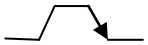
1	0	1	0	0	S	S	S

N

It has 7 variations. In this AND operation all flags are affected as per standard rule. However, CY flag is cleared and AC flag is set. The micro RTL flow implemented is given below:

Machine Cycle- 1:

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

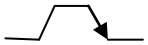
T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (IR) ⟵ AD_7-AD_0

T_4: μp decodes the opcode. ANA r = 1.

Machine Cycle- 2 or (Machine Cycle-1 of next instruction):

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

FEO [(Temp) ⟵ (r)]

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

FEO [(A) ⟵ (A) ∩ (Temp)]

It requires single machine cycle OFMC of 4 states. It has register addressing mode.

2. ANA M: This is a single byte instruction. The meaning of the instruction is “AND bit by bit the content of memory location pointed by (H, L) register pair to the content of accumulator and store the result back in the accumulator”. The macro RTL implemented is

(A) ⟵ (A) ∩ M(H, L)

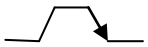
The opcode of the instruction is

1	0	1	0	0	1	1	0	N

It has no variations. In this instruction also, all flags are affected as per standard rule. However, CY flag is cleared and AC flag is set. The micro RTL flow implemented is given below:

Machine Cycle- 1:

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

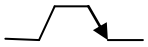
T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (IR) ⟵ AD_7-AD_0

T_4: μp decodes the opcode. ANA M = 1.

Machine Cycle- 2:

MRMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=0

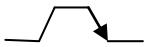
T_1: A_{15}-A_8 ⟵ (H), AD_7-AD_0 ⟵ (L), ALE = 

T_2: $\overline{RD}$ = 0, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑, (Temp) ⟵ AD_7-AD_0

Machine Cycle- 3 or (Machine Cycle-1 of next instruction):

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

FEO [(A) ⟵ (A) ∩ (Temp)]

It requires two machine cycles OFMC & MRMC and a total of 4 states. It makes use of register indirect addressing mode.

3. ANI DATA: It is a two byte instruction. The meaning of the instruction is "AND bit by bit the content available as a second byte of instruction to the content of the accumulator and store the result back in the accumulator". The macro RTL implemented is

$$(A) \longleftarrow (A) \cap \langle B_2 \rangle$$

The opcode of the instruction is

1	1	1	0	0	1	1	0	N
$\langle B_2 \rangle$								N+1

It has no variations. In this AND immediate instruction all flags are affected as per standard rule. However, CY flag is cleared and AC flag is set. The micro RTL flow implemented is given below:

Machine Cycle- 1:

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = ⎍

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (IR) ⟵ AD_7-AD_0

T_4: μp decodes the opcode. ANI data = 1.

Machine Cycle- 2:

MRMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=0

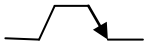
T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = ⎍

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑, (Temp) ⟵ AD_7-AD_0

Machine Cycle- 3 or (Machine Cycle-1 of next instruction):

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

FEO [(A) ⟵ (A) ∩ (Temp)]

It requires two machine cycles OFMC & MRMC and a total of 7 states. It makes use of register indirect addressing mode.

4. XRA r: It is a single byte instruction. The meaning of the instruction is "Exclusively ORed bit by bit the content of register (r) with the content of accumulator and store the result back into accumulator". The macro RTL implemented is,

(A) ⟵ (A) ⊕ (r)

The operation code is

1	0	1	0	1	S	S	S

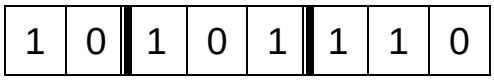
N

It has seven variations and the addressing mode is register addressing mode. The micro RTL flow is identical to ANA r instruction. Only one Machine cycle is required i.e. OFMC of 4 states.

5. XRA M: (Exclusive OR memory) This is a single byte instruction. The meaning of the instruction is "Exclusively ORed bit by bit the content of memory location pointed by (H, L) register pair to the content of accumulator and store the result back in the accumulator". The macro RTL implemented is

(A) ⟵ (A) ⊕ M(H, L)

The opcode of the instruction is

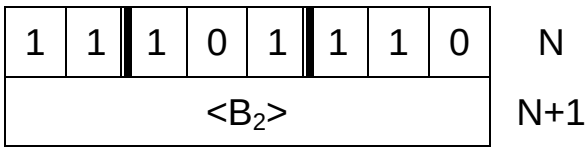


It has no variations. The addressing mode is register indirect addressing mode. The micro RTL flow implemented is identical to ANA M. It requires two machine cycles OFMC & MRMC and a total of 7 states.

6. XRI data:(Exclusive OR immediate) It is a two byte instruction. The meaning of the instruction is “Exclusively ORed bit by bit the content of second byte of the instruction with the content of accumulator and store the result back into accumulator”. The macro RTL implemented is

$$(A) \longleftarrow (A) \oplus \langle B_2 \rangle$$

The operation code is,



It has no variation. The addressing mode is immediate addressing mode. The micro RTL flow is identical to ANI data. Therefore, needs two machine cycles are required OFMC-4 &MRMC-3 and a total of 7 states.

Note:

A. In all exclusive–OR instructions, all the flags are affected as per standard rule expect AC & CY flags are cleared.
B. There is no explicit instruction to clear the accumulator to ‘zero’. However, it is implicit is the statement XRA A. The macro RTL is

$$(A) \longleftarrow (A) \oplus (A)$$

The result will be $(0000\ 0000)_2$ in the accumulator along with clearing AC & CY flags.

7. ORA r: It is a single byte instruction. The meaning of the instruction is "ORed bit by bit the content of register (r) with the content of accumulator and store the result back into accumulator". The macro RTL implemented is,

$$(A) \longleftarrow (A) \cup (r)$$

The operation code is

1	0	1	1	0	S	S	S	N

It has seven variations and the addressing mode is register addressing mode. The micro RTL flow is identical to ANA r instruction. Only one Machine cycle is required i.e. OFMC of 4 states. All the flags are affected except AC& CY flags are cleared.

8. ORA M: (OR memory) This is a single byte instruction. The meaning of the instruction is "ORed bit by bit the content of memory location pointed by (H, L) register pair to the content of accumulator and store the result back in the accumulator". The macro RTL implemented is

$$(A) \longleftarrow (A) \cup M(H, L)$$

The opcode of the instruction is

1	0	1	1	0	1	1	0	N

It has no variations. The addressing mode is register indirect addressing mode. The micro RTL flow implemented is identical to ANA M. It requires two machine cycles OFMC & MRMC and a total of 7 states. All the flags are affected except AC& CY flags are cleared.

9. ORI data:(OR immediate) It is a two byte instruction. The meaning of the instruction is “ORed bit by bit the content of second byte of the instruction with the content of accumulator and store the result back into accumulator”. The macro RTL implemented is

$$(A) \longleftarrow (A) \cup \langle B_2 \rangle$$

The operation code is,

1	1	1	1	0	1	1	0	N
$\langle B_2 \rangle$								N+1

It has no variation. The addressing mode is immediate addressing mode. The micro RTL flow is identical to ANI data. Therefore, needs two machine cycles are required OFMC-4 &MRMC-3 and a total of 7 states. All the flags are affected except AC& CY flags are cleared.

Note: ANI data, XRI data & ORI data are used for manipulating any bit of the accumulator without affecting the other bits. For example,

(i) Set Bit 5 of accumulator without affecting the other bits.

	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
	0	0	1	0	0	0	0	0
OR operation	A_7	A_6	1	A_4	A_3	A_2	A_1	A_0

ORI 20_H set bit 5 of (Acc) without affecting the other bits. This is known as Masking.

(ii) Force bit 3 & bit 2 to 0, 0 without affecting the other bits.

	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
	1	1	1	1	0	0	1	1
AND operation	A_7	A_6	A_5	A_4	0	0	A_1	A_0

ANI $F3_H$ resets bit 3 and bit 2 of (Acc) without affecting the other bits.

(iii) Complement bit-7, bit-5 & bit-3 of (Acc) without affecting the other bits.

	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
	1	0	1	0	1	0	0	0
XOR operation	$\overline{A_7}$	A_6	$\overline{A_5}$	A_4	$\overline{A_3}$	A_2	A_1	A_0

XRI $A8_H$ complements bit7, bit-5 and bit-3 of (Acc) without affecting the other bits.

(iv) Test a particular bit (say bit-6) of accumulator to find whether it is ‘0’ or ‘1’.

	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
	0	1	0	0	0	0	0	0
AND operation	0	A_6	0	0	0	0	0	0

ANI 40H instruction is used to check A_6 bit. It makes the result in accumulator 00H or 40H depending on A_6 bit. If A_6 bit is 0, content of accumulator will be zero. If A_6 bit is 1, content of accumulator will be non-zero.

Lecture-28

10. CMP r: (Compare register with ACC) It is a single byte instruction. The meaning of the instruction "Compare the content of accumulator with the content of register (r)." In this instruction, the content of register (r) is subtracted from the content of accumulator (A) but the result of the subtraction operation is not stored back into the Acc. Thus the contents of Acc & register (r) are not destroyed but as result of this operation the flags are affected as per standard rule as given below:

If (A) > (r) CY = 0, Z = 0

(A) = (r) CY = 0, Z = 1

(A) < (r) CY = 1, Z = 0

The operation code of the instruction is,

1	0	1	1	1	S	S	S

N

It has seven variations. The addressing mode is register addressing mode. The micro RTL flow implemented is given below:

Machine Cycle- 1:

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = ⎍

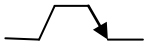
T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (IR) ⟵ AD_7-AD_0

T_4: μp decodes the opcode. CMP r = 1.

Machine Cycle- 2 or (Machine Cycle-1 of next instruction):

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

FEO [(Temp) ⟵ (r)]

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

FEO [(A) - (Temp) operation takes place and flags are affected]

The instruction needs one machine cycle OFMC of 4 states.

11. CMP M: (Compare memory with accumulator) It is a single byte instruction. The meaning of the instruction is "Compare the content of accumulator with the content of memory location whose address is available in (H, L) register pair". In this instruction, the content of memory location pointed to by (H, L) register pair is subtracted from the content of accumulator (A) but the result of the subtraction operation is not stored back into the Acc. Thus the contents of Acc & memory are not destroyed but as result of this operation the flags are affected as per standard rule as given below:

If (A) > M(H, L) CY = 0, Z = 0

(A) = M(H, L) CY = 0, Z = 1

(A) < M(H, L) CY = 1, Z = 0

The operation code is,

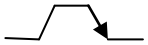
The operation code of the instruction is,

1	0	1	1	1	1	1	0	N

It has no variations. The addressing mode is register indirect addressing mode. The micro RTL flow implemented is given below:

Machine Cycle- 1:

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

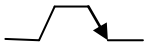
T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (IR) ⟵ AD_7-AD_0

T_4: μp decodes the opcode. CMP M = 1.

Machine Cycle- 2:

MRMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=0

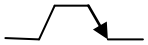
T_1: A_{15}-A_8 ⟵ (H), AD_7-AD_0 ⟵ (L), ALE = 

T_2: $\overline{RD}$ = 0, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑, (Temp) ⟵ AD_7-AD_0

Machine Cycle- 3 or (Machine Cycle-1 of next instruction):

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

FEO [(A) - (Temp) operation takes place and flags are affected]

The instruction requires two machine cycles OFMC & MRMC and a total of 7 states.

12. CPI data: (Compare accumulator with immediate data) This is an ALP statement. The meaning of the instruction is “Compare the content of accumulator with the 8-bit data available in the instruction

as immediate data byte". In this instruction, the 8-bit data available in the instruction as 2nd byte is subtracted from the content of accumulator (A) but the result of the subtraction operation is not stored back into the Acc. Thus the contents of Acc is not destroyed but as result of this operation the flags are affected as per standard rule as given below:

If (A) > $<B_2>$ CY = 0, Z = 0

(A) = $<B_2>$ CY = 0, Z = 1

(A) < $<B_2>$ CY = 1, Z = 0

The operation code of the instruction is,

1	0	1	1	0	1	1	0	N
$<B_2>$								N+1

It has no variations. The addressing mode is immediate addressing mode. The micro RTL flow implemented is given below:

Machine Cycle- 1:

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = ⎍

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (IR) ⟵ AD_7-AD_0

T_4: μp decodes the opcode. CPI data = 1.

Machine Cycle- 2:

MRMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=0

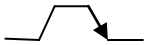
T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = ⎍

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (Temp) ⟵ AD_7-AD_0

Machine Cycle- 3 or (Machine Cycle-1 of next instruction):

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

FEO [(A) - (Temp) operation takes place and flags are affected]

The instruction requires two machine cycles OFMC & MRMC and a total of 7 states.

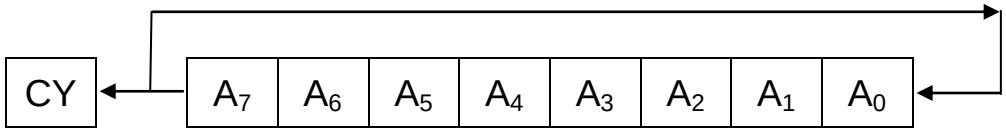
13. RLC: This is an ALP statement. The meaning of the instruction is "Rotate left the content of the accumulator by one bit without carry flag. The lower order bit & the CY flag are both set to the value shifted out the high order bit position". In this instruction it is assumed that the operand is available in accumulator. The macro RTL implemented is,

(A_{i+1}) ⟵ (A_i) i = 0 to 6

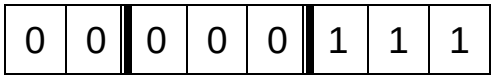
(A_0) ⟵ (A_7)

(CY) ⟵ (A_7)

Or,



The operation code of the instruction is,

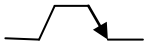


N

It has no variations. The addressing mode is implied addressing mode. The micro RTL flow implemented is given below:

Machine Cycle- 1:

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

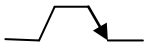
T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (IR) ⟵ AD_7-AD_0

T_4: μp decodes the opcode. RLC = 1.

Machine Cycle- 3 or (Machine Cycle-1 of next instruction):

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

FEO [RLC operation takes place and flags are affected]

The instruction requires one machine cycle OFMC and a total of 4 states. This instruction is used to check accumulator bits one by one. Only the CY flag is affected in this operation.

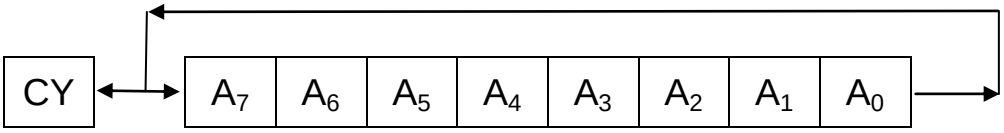
14. RRC: This is an ALP statement. The meaning of the instruction is "Rotate right the content of the accumulator by one bit without carry flag. The higher order bit & the CY flag are both set to the value shifted out the lower order bit position". In this instruction also it is assumed that the operand is available in accumulator. The macro RTL implemented is,

(A_i) ⟵ (A_{i+1}) i = 0 to 6

(A_7) ⟵ (A_0)

(CY) ⟵ (A_0)

Or,



The operation code of the instruction is,

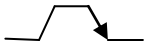
0	0	0	0	1	1	1	1

N

It has no variations. The addressing mode is implied addressing mode. The micro RTL flow implemented is given below:

Machine Cycle- 1:

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

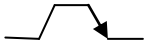
T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (IR) ⟵ AD_7-AD_0

T_4: μp decodes the opcode. RRC = 1.

Machine Cycle- 3 or (Machine Cycle-1 of next instruction):

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

FEO [RRC operation takes place and flags are affected]

The instruction requires one machine cycle OFMC and a total of 4 states. This instruction is used to check accumulator bits one by one. Only the CY flag is affected in this operation.

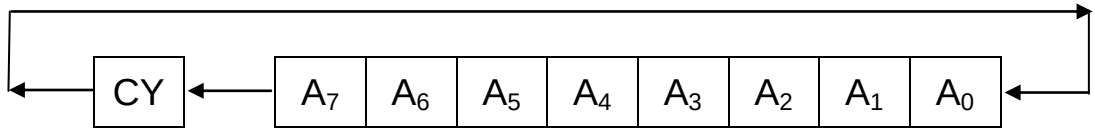
15. RAL: (Rotate left through carry) This is an ALP statement. The meaning of the instruction is "Rotate left the content of the accumulator by one bit through the carry flag. The lower order bit set equal to the CY flag and CY flag is set to the value shifted out of the A_7 bit.". In this instruction it is assumed that the operand is available in accumulator. The macro RTL implemented is,

$(A_{i+1}) \longleftarrow (A_i)$ i = 0 to 6

$(CY) \longleftarrow (A_7)$

$(A_0) \longleftarrow (CY)$

Or,



The operation code of the instruction is,

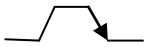
0	0	0	1	0	1	1	1

N

It has no variations. The addressing mode is implied addressing mode. The micro RTL flow implemented is given below:

Machine Cycle- 1:

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-$A_8 \longleftarrow$ (PCH), AD_7-$AD_0 \longleftarrow$ (PCL), ALE = 

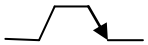
T_2: $\overline{RD}$ = 0, (PC) $\longleftarrow$ (PC) +1, AD_7-$AD_0 \longleftarrow$ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (IR) $\longleftarrow$ AD_7-AD_0

T_4: μp decodes the opcode. RAL = 1.

Machine Cycle- 2 or (Machine Cycle-1 of next instruction):

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

FEO [RAL operation takes place and flags are affected]

The instruction requires one machine cycle OFMC and a total of 4 states. This instruction is used to check accumulator bits one by one. Only the CY flag is affected in this operation.

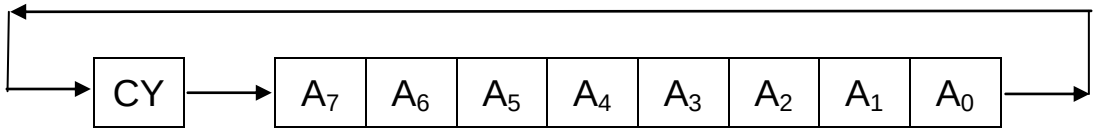
16. RAR: (Rotate accumulator right through carry) This is an ALP statement. The meaning of the instruction is "Rotate right the content of the accumulator by one bit through carry flag. The higher order bit is set equal to the CY flag and the CY flag is set to the value shifted out the lower order bit position". The operand is assumed to be available in accumulator. The macro RTL implemented is,

(A_i) ⟵ (A_{i+1}) i = 0 to 6

(CY) ⟵ (A_0)

(A_7) ⟵ (CY)

Or,



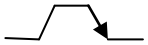
The operation code of the instruction is,

0	0	0	1	1	1	1	1	N

It has no variations. The addressing mode is implied addressing mode. The micro RTL flow implemented is given below:

Machine Cycle- 1:

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

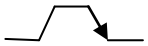
T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (IR) ⟵ AD_7-AD_0

T_4: μp decodes the opcode. RAR = 1.

Machine Cycle- 2 or (Machine Cycle-1 of next instruction):

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

FEO [RAR operation takes place and flags are affected]

The instruction requires one machine cycle OFMC and a total of 4 states. This instruction is used to check accumulator bits one by one. Only the CY flag is affected in this operation.

Lecture-29

17. CMA: (Complement accumulator) This is an ALP statement. The meaning of the instruction is "Complement the content of accumulator bit by bit and store the result back into the accumulator". The macro RTL implemented is,

$$(A) \longleftarrow (\bar{A})$$

No flag is affected in this instruction. It is a single byte instruction. The operation code is,

0	0	1	0	1	1	1	1	N

It has no variations. The addressing mode is implied addressing mode. The micro RTL flow implemented is given below:

Machine Cycle- 1:

OFMC: Status signals IO/$\bar{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = ⎍

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (IR) ⟵ AD_7-AD_0

T_4: μp decodes the opcode. CMA = 1.

Machine Cycle- 2 or (Machine Cycle-1 of next instruction):

OFMC: Status signals IO/$\bar{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = ⎍

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

FEO [Complement accumulator operation takes place]

The instruction requires one machine cycle OFMC and a total of 4 states. This instruction is used to check accumulator bits one by one. Only the CY flag is affected in this operation.

18. CMC: (Complement carry flag) This is an ALP statement. The meaning of the instruction is “Complement CY flag and store it back into the same CY flag position”. The macro RTL implemented is,

$(CY) \longleftarrow (\overline{CY})$

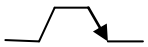
The operation code is,

0	0	1	1	1	1	1	1	N

It has no variations. The addressing mode is implied addressing mode. The micro RTL flow implemented is given below:

Machine Cycle- 1:

OFMC: Status signals $IO/\overline{M}=0$, $S_1=1$, $S_0=1$

T_1: A_{15}-$A_8 \longleftarrow$ (PCH), AD_7-$AD_0 \longleftarrow$ (PCL), ALE = 

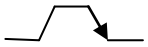
T_2: $\overline{RD} = 0$, (PC) $\longleftarrow$ (PC) +1, AD_7-$AD_0 \longleftarrow$ M(AB)

T_3: $\overline{RD} = 1$,  , (IR) $\longleftarrow$ AD_7-AD_0

T_4: μp decodes the opcode. CMC = 1.

Machine Cycle- 2 or (Machine Cycle-1 of next instruction):

OFMC: Status signals $IO/\overline{M}=0$, $S_1=1$, $S_0=1$

T_1: A_{15}-$A_8 \longleftarrow$ (PCH), AD_7-$AD_0 \longleftarrow$ (PCL), ALE = 

T_2: $\overline{RD} = 0$, (PC) $\longleftarrow$ (PC) +1, AD_7-$AD_0 \longleftarrow$ M(AB)

FEO [Complement CY operation takes place]

The instruction requires one machine cycle OFMC and a total of 4 states. This instruction may be used after STC to clear carry without affecting any other flag and register. Only the CY flag is affected in this operation.

19. STC: (Set carry flag) This is an ALP statement. The meaning of the instruction is “Set the CY flag to ‘1’”. The macro RTL implemented is,

(CY) ⟵ 1

The operation code is,

0	0	1	1	0	1	1	1	N

It has no variations. The addressing mode is implied addressing mode. The micro RTL flow implemented is given below:

Machine Cycle- 1:

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = ⎍

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (IR) ⟵ AD_7-AD_0

T_4: μp decodes the opcode. STC = 1.

Machine Cycle- 2 or (Machine Cycle-1 of next instruction):

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = ⎍

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

FEO [CY flag is set to 1]

The instruction requires one machine cycle OFMC and a total of 4 states. This instruction may be used to set CY without affecting any other flag and register. Only the CY flag is affected in this operation.

BRANCH GROUP:

This group of instructions is used to alter the normal sequential program flow and force the program to proceed from a different point. Condition flags are not affected by any instruction in this group only program counter (PC) is affected; sometimes stack is also affected. Branch instructions can be of two types: conditional & unconditional. Unconditional branch instructions simply cause the program to branch to the indicated instruction whenever these instructions are encountered, i.e., (PC) is loaded with a new address. Conditional branch instructions examine the status of one of the four processor flags (Z, CY, P, S) to determine if the specified branch instruction is to be executed. If the condition tested is TRUE, it causes a branching to occur otherwise not. AC is not used for specifying condition. The 8 conditions that are tested are given below:

Condition		Flag Tested	Identification Code (CCC)
NZ	Not zero	Z=0	000
Z	Zero	Z=1	001
NC	Not carry	CY=0	010
C	Carry	Cy=1	011
PO	Parity odd	P=0	100
PE	Parity even	P=1	101
P	Plus	S=0	110
M	Minus	S=1	111

If Z = 0 is not true, then NZ is true. If Z = 0 is true, and then NZ is not true.

The identification condition codes (CCC) are used in conditional branch instructions. In such conditional branch instructions, CCC code is as $D_5\,D_4\,D_3$ of the operation code

There are 8 basic operations in this group the operation code format is shown in fig.518.

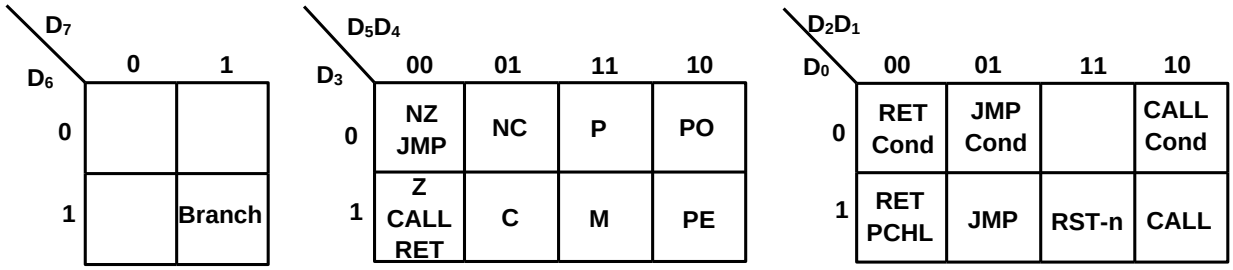


Fig.5.18 Format of Branch Group Instructions

1. JMP ADDR: This is an ALP statement. ADDR is the symbolic name given to the 16- bit address data available as the 2nd and 3rd bytes of the instruction. JMP is the mnemonic for jump. The meaning of the instruction is “Load the PC with the 16- bit data available in the instruction itself as the 2nd and 3rd bytes of instruction so that the next instruction is fetched from this address in the succeeding instruction cycle. This is a 3 –byte instruction. The operation code format is

1	1	0	0	0	0	1	1	N
<B_2>								N+1
<B_3>								N+2

The macro RTL implemented is

(PCH) ⟵ < B_3 >

(PCL) ⟵ < B_2 >

Memory conditions before & after the execution of JMP ADDR instruction are shown in fig.5.19

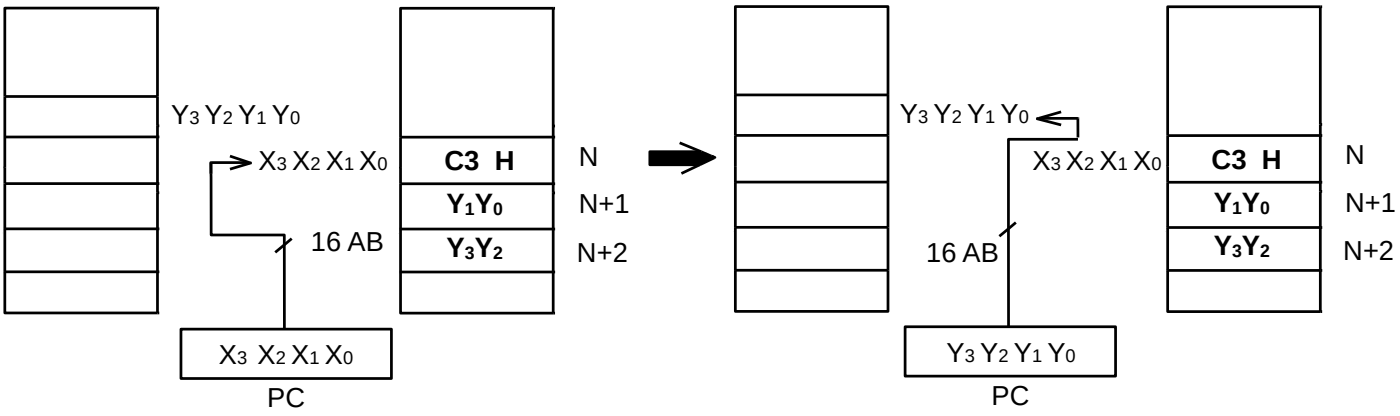


Fig.5.19 Memory Conditions Before and After JMP ADDR Instruction

The macro RTL flow is as shown below:

Machine Cycle- 1:

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ← (PCH), AD_7-AD_0 ← (PCL), ALE = ⎍

T_2: $\overline{RD}$ = 0, (PC) ← (PC) +1, AD_7-AD_0 ← M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (IR) ← AD_7-AD_0

T_4: μp decodes the opcode. JMP addr = 1.

Machine Cycle- 2:

MRMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=0

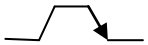
T_1: A_{15}-A_8 ← (PCH), AD_7-AD_0 ← (PCL), ALE = ⎍

T_2: $\overline{RD}$ = 0, (PC) ← (PC) +1, AD_7-AD_0 ← M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (Z) ← AD_7-AD_0

Machine Cycle- 3:

MRMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=0

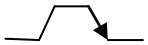
T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (W) ⟵ AD_7-AD_0

Machine Cycle- 4 or (Machine Cycle-1 of next instruction):

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (W), AD_7-AD_0 ⟵ (Z), ALE = 

T_2: $\overline{RD}$ = 0, FEO [(PC) ⟵ (W, Z) +1], AD_7-AD_0 ⟵ M(AB)

Thus, it requires three machine cycle OFMC, MRMC & MRMC and total 10 states. Using a 2 MH_3 internal clock it requires 5μ sec. There is no variation of this instruction. The addressing mode used in this instruction is immediate addressing mode because the 16-bit address data is immediately available in the instruction itself to be loaded into the PC.

2. Jcond ADDR: This is also an ALP statement. This is a 3-byte conditional jump instruction. There are 8 conditions that can be checked. Accordingly, there are 8-variations for this instruction depending upon the conditions to be tested. They are JNZ, JZ, JNC, JC, JPO, JPE, JP, and JM. The operation code format is,

1	1	C	C	C	0	1	0	N
<B_2>								N+1
<B_3>								N+2

The macro RTL implemented is

If condition CCC is TRUE

(PCH) ⟵ < B_3 >

(PCL) ⟵ < B_2 >

Else

Nothing.

The meaning of the instruction is illustrated in the flow chart

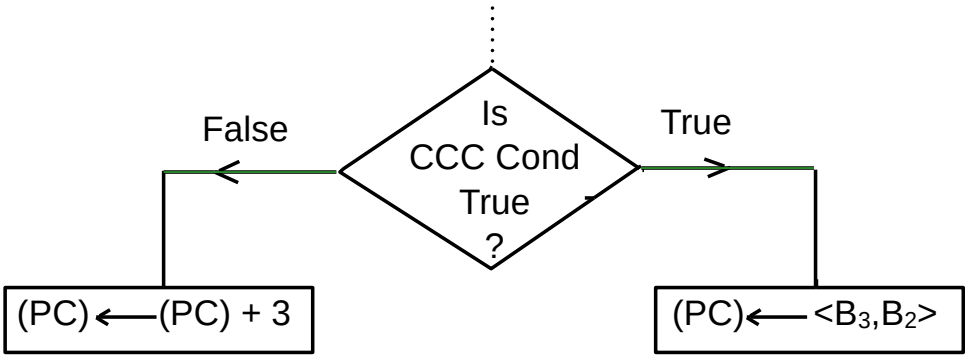


Fig.5.20 Flow Chart for Conditional Jump Instruction

where (PC) = $(X_3X_2X_1X_0)_H$ just at the start of instruction exist.

Fig.5.21 shows the condition existing just before the start of instruction cycle Jcond ADDR.

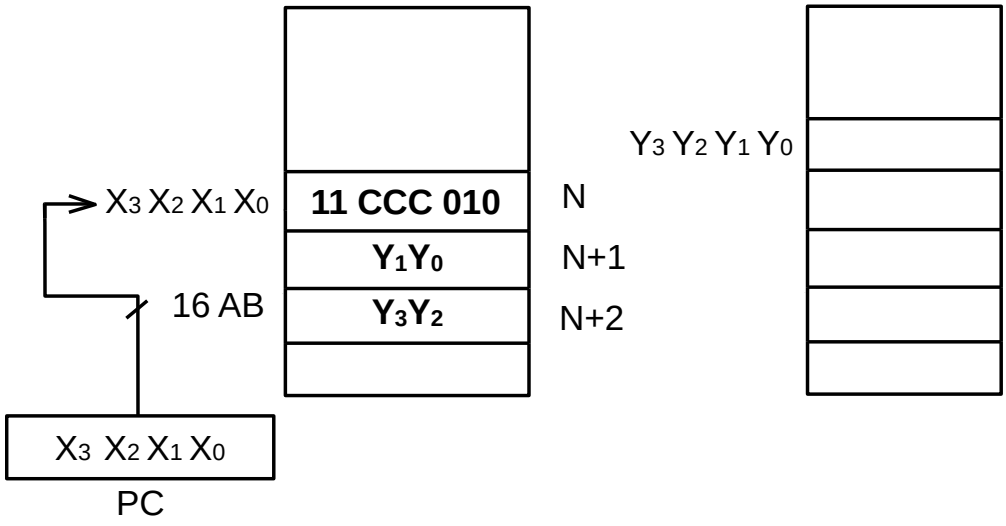


Fig.5.21 Memory Conditions Before Execution of Jcond Instruction

The I6-bit address in the (PC) just at the end of the instruction depends upon the condition to be tested. (PC) will be loaded with B_3B_2 if the given condition is TRUE, otherwise (PC) will go to $(PC)_i+3$ where $(PC)_i$ shall be the address $X_3X_2X_1X_0$ which points to the operation code $(11\ CCC\ 010)_2$. The micro RTL flow given below:

Machine Cycle- 1:

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = ⎍

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (IR) ⟵ AD_7-AD_0

T_4: μp decodes the opcode. Jcond addr = 1.

Machine Cycle- 2:

MRMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=0

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = ⎍

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (Z) ⟵ AD_7-AD_0.

If condition is TRUE go to MC-3 else PC ⟵ (PC) +1.

Machine Cycle- 3:

MRMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=0

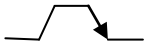
T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = ⎍

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (W) ⟵ AD_7-AD_0

Machine Cycle- 4 or (Machine Cycle-1 of next instruction):

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

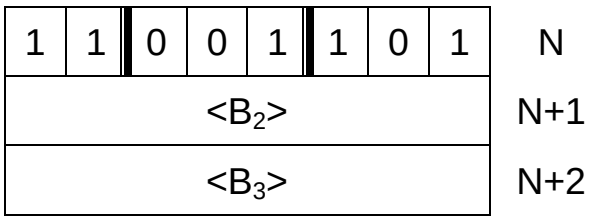
T_1: A_{15}-A_8 ← (W), AD_7-AD_0 ← (Z), ALE = 

T_2: $\overline{RD}$ = 0, FEO [(PC) ← (W, Z) +1], AD_7-AD_0 ← M(AB)

Therefore, if the condition is TRUE, it takes three machine cycles OFMC, MRMC & MRMC to get the instruction executed and a total of 10 states. If the condition is not TRUE then it takes two machine cycles OFMC, MRMC and a total of 7 states. The addressing mode used in this instruction is immediate addressing mode because the 16-bit address data is immediately available in the instruction itself to be loaded into the PC. No flag is affected.

Lecture-30

3. CALL ADDR: (Unconditional subroutine call) This is an ALP statement. ADDR is the symbolic name given to the 16-bit address available as the 2nd and 3rd byte of the instruction. This is a 3-byte instruction. The operation code format is,



In a program, the instruction looks as shown in fig.5.22

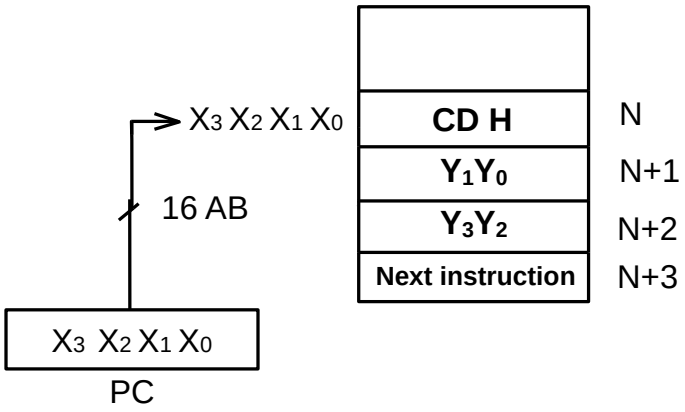


Fig.5.22 CALL ADDR Instruction Stored in Memory

The op-code CD_H, being at memory location N, the hexa address being $(X_3X_2X_1X_0)_H$, then N+3 = $(X_3X_2X_1X_0 + 3)_H$ is known as the return address. This instruction is used for unconditional subroutine CALL. The starting address of the subroutine is $Y_3Y_2Y_1Y_0$, available immediately in the instruction itself as 2nd and 3rd bytes of the instruction. The stack is made use of to store the return address before jumping to the subroutine. The meaning of the instruction is "Save the return address on the top of the stack and thereafter, load the (PC) with the 16-bit address immediately available in the instruction as 2nd and 3rd bytes". The macro RTL implemented is,

$$M[(SP)-1] \leftarrow (PCH)$$
$$M[(SP)-2] \leftarrow (PCL)$$
$$(SP) \leftarrow (SP) - 2$$
$$(PC) \leftarrow (B_3, B_2)$$

where (PCH,PCL) is the return address. The higher order 8-bits of the return address are moved to the memory location whose address is one less than the content of (SP). The lower order 8-bits of the return address are moved to the memory location whose address is two less than the content of (SP). The content of the register (SP) is decremented by 2 in this process so that it always point to the top of the stack. Control is transferred to the instruction whose address is specified in 2nd and 3rd bytes of the instruction. The addressing mode used is immediate addressing mode because the 16-bit address of the subroutine where the control is transferred is immediately available in the instruction. This instruction also includes register indirect addressing mode because the return address has to be saved on the top of the stack pointed to by 16-bit register (SP). Memory conditions before & after the execution of CALL ADDR instruction are shown below.

Condition before execution is,

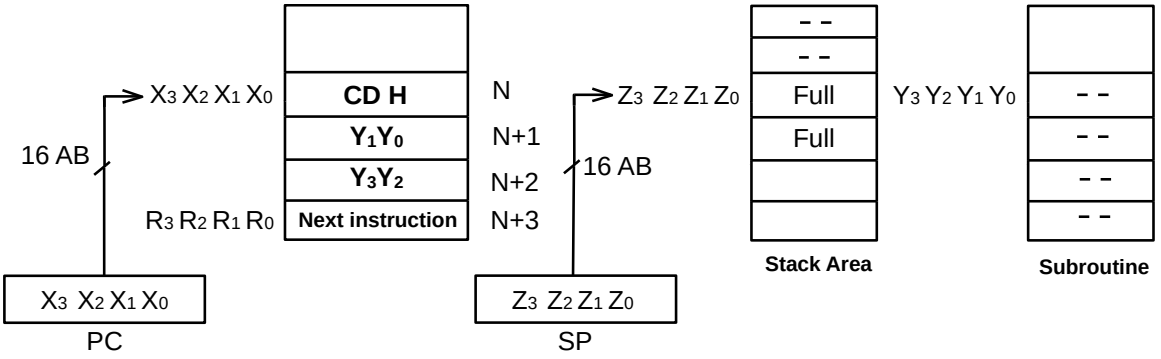


Fig.5.23 Condition of Memory, SP and PC Before Execution of CALL ADDR

After the execution

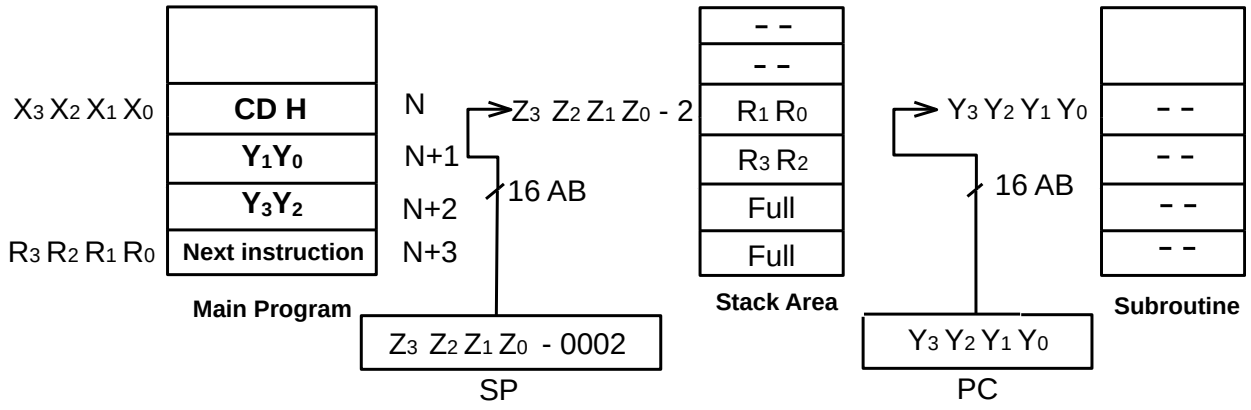


Fig.5.24 Condition of Memory, SP and PC After Execution of CALL ADDR

The macro RTL implemented is

(PCH) ⟵ < B_3 >

(PCL) ⟵ < B_2 >

The macro RTL flow is as shown below:

Machine Cycle- 1:

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = ⎍

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (IR) ⟵ AD_7-AD_0

T_4: μp decodes the opcode. JMP addr = 1.

T_5:

T_6: (SP) ⟵ (SP) -1

Machine Cycle- 2:

MRMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=0

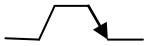
T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = ⎍

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (Z) ⟵ AD_7-AD_0

Machine Cycle- 3:

MRMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=0

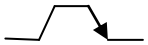
T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (W) ⟵ AD_7-AD_0

Machine Cycle- 4:

MWRMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

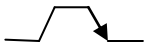
T_1: A_{15}-A_8 ⟵ (SPH), AD_7-AD_0 ⟵ (SPL), ALE = 

T_2: $\overline{WR}$ = 0, AD_7-AD_0 ⟵ (PCH)

T_3: $\overline{WR}$ = 1, ↑ , M(AB) ⟵ AD_7-AD_0, (SP) ⟵ (SP) -1

Machine Cycle- 5:

MWRMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

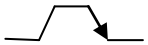
T_1: A_{15}-A_8 ⟵ (SPH), AD_7-AD_0 ⟵ (SPL), ALE = 

T_2: $\overline{WR}$ = 0, AD_7-AD_0 ⟵ (PCL)

T_3: $\overline{WR}$ = 1, ↑ , M(AB) ⟵ AD_7-AD_0,

Machine Cycle- 6 or (Machine Cycle-1 of next instruction):

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (W), AD_7-AD_0 ⟵ (Z), ALE = 

T_2: $\overline{RD}$ = 0, FEO [(PC) ⟵ (W, Z) +1], AD_7-AD_0 ⟵ M(AB)

Thus, it requires five machine cycles OFMC, two MRMC & two MWRMC and total 18 states, the longest instruction cycle in 8085A processor. Using a 2 MH_3 internal clock it requires 9μ sec. There is no variation of this instruction.

4. <u>Ccond ADDR</u> (Unconditional subroutine call): This is an ALP statement. This is also 3-byte instruction. 2nd and 3rd bytes give the address of the subroutine. When this instruction is executed, the μp jump to the subroutine if the condition tested is TRUE. If the condition is not TRUE then μp goes to execute the next instruction. The operation code format is,

1	1	C	C	C	1	0	0	N
<B_2>								N+1
<B_3>								N+2

where CCC represents the 8 possible conditions. This instruction has 8 variations for CCC- $(000)_2$ to $(111)_2$. The corresponding instructions are CNZ, CZ, CNC, CC, CPO, CPE, CP and CM.

The macro RTL implemented is,

If CCC= TRUE

M [(SP) -1] ⟵ (PCH)

M [(SP) - 2] ⟵ (PCL)

(SP) ⟵ (SP) – 2

(PC) ⟵ (B_3, B_2)

Else

Nothing.

where (PCH,PCL) is the return address. If the condition tested is TRUE, then the higher order 8-bits of the return address are moved to the memory location whose address is one less than the content of (SP). The lower order 8-bits of the return address are moved to the

memory location whose address is two less than the content of (SP). The content of the register (SP) is decremented by 2 in this process so that it always point to the top of the stack. Control is transferred to the instruction whose address is specified in 2nd and 3rd bytes of the instruction. If the condition is not found TRUE, the control goes to the next instruction in sequence.

The addressing mode used is immediate addressing mode because the 16-bit address of the subroutine where the control is transferred is immediately available in the instruction. This instruction also includes register indirect addressing mode because the return address has to be saved on the top of the stack pointed to by 16-bit register (SP). The macro RTL implementation flowchart is shown below:

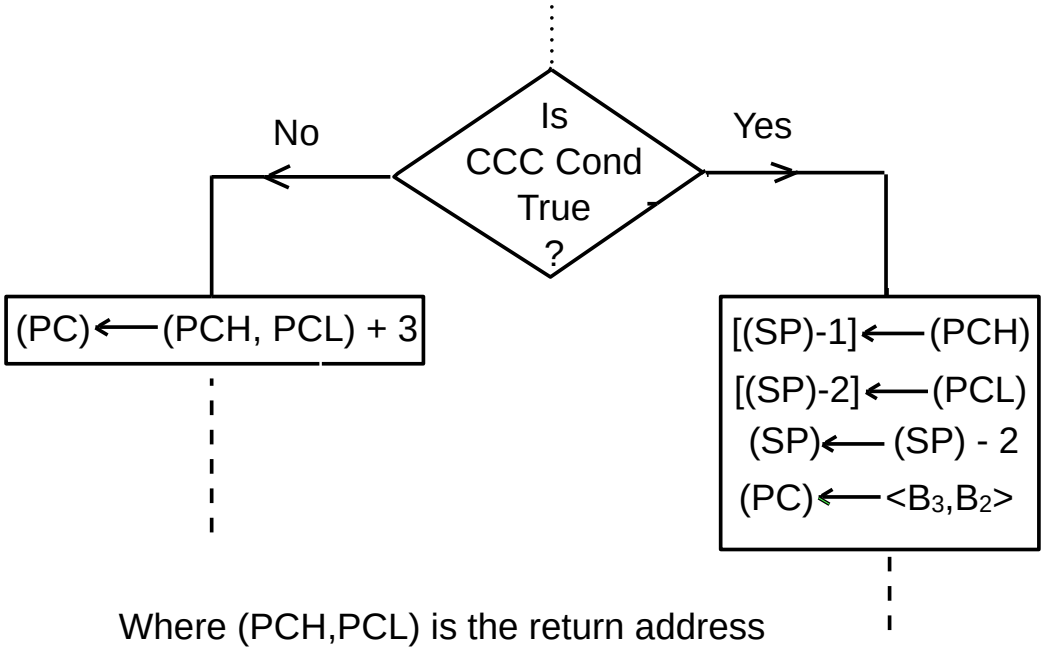
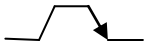


Fig.5.25 Flowchart for Conditional CALL ADDR Instruction

The macro RTL flow is as shown below:

Machine Cycle- 1:

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ← (PCH), AD_7-AD_0 ← (PCL), ALE = 

T_2: $\overline{RD}$ = 0, (PC) ← (PC) +1, AD_7-AD_0 ← M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (IR) ← AD_7-AD_0

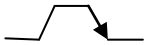
T_4: μp decodes the opcode. Jcond addr = 1.

T_5: } If condition CCC is TRUE then

T_6: } (SP) ← (SP) -1

Machine Cycle- 2:

MRMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=0

T_1: A_{15}-A_8 ← (PCH), AD_7-AD_0 ← (PCL), ALE = 

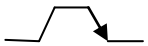
T_2: $\overline{RD}$ = 0, (PC) ← (PC) +1, AD_7-AD_0 ← M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (Z) ← AD_7-AD_0

If condition is TRUE go to MC-3 else PC ← (PC) +1.

Machine Cycle- 3:

MRMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=0

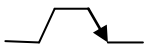
T_1: A_{15}-A_8 ← (PCH), AD_7-AD_0 ← (PCL), ALE = 

T_2: $\overline{RD}$ = 0, (PC) ← (PC) +1, AD_7-AD_0 ← M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (W) ← AD_7-AD_0

Machine Cycle- 4:

MWRMC: Status signals IO/$\overline{M}$=0, S_1=0, S_0=1

T_1: A_{15}-A_8 ← (SPH), AD_7-AD_0 ← (SPL), ALE = 

T_2: $\overline{WR}$ = 0, AD_7-AD_0 ⟵ (PCH)

T_3: $\overline{WR}$ = 1, ↑ , M(AB) ⟵ AD_7-AD_0, (SP) ⟵ (SP) -1

Machine Cycle- 5:

MWRMC: Status signals IO/$\overline{M}$=0, S_1=0, S_0=1

T_1: A_{15}-A_8 ⟵ (SPH), AD_7-AD_0 ⟵ (SPL), ALE = ⎍

T_2: $\overline{WR}$ = 0, AD_7-AD_0 ⟵ (PCL)

T_3: $\overline{WR}$ = 1, ↑ , M(AB) ⟵ AD_7-AD_0,

Machine Cycle- 6 or (Machine Cycle-1 of next instruction):

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (W), AD_7-AD_0 ⟵ (Z), ALE = ⎍

T_2: $\overline{RD}$ = 0, FEO [(PC) ⟵ (W, Z) +1], AD_7-AD_0 ⟵ M(AB)

Thus, if condition is TRUE, it requires five machine cycles OFMC, two MRMC & two MWRMC and total 18 states otherwise only two machine cycles OFMC & one MRMC and total of 9 states are required. Using a 2 MH_3 internal clock it requires either 9μsec or 4.5μsec.

5. RET (Unconditional Return): This is an ALP statement and stands for RETURN. The meaning of the instruction is “Return to the main program from the subroutine unconditionally”. Obviously RET instruction should be a part of the subroutine program. This is a single byte instruction. The operation code format is,

1	1	0	0	1	0	0	1	N

The macro RTL implemented is,

(PCL) ⟵ M [(SP)]

(PCH) ⟵ M [(SP) + 1]

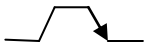
(SP) ⟵ (SP)+2

When this instruction is executed the 16-bit address data available at the top of the stack is loaded into the (PC) and stack pointer is readjusted so that it again points to the top of the stack. The content of the memory location whose address is specified in register (SP) is moved to the lower order 8-bits of program counter (PCH). The content of the memory location whose address is one more than the content of register (SP) is moved to the higher order 8-bits of program counter (PCH). In this process the stack goes down and the content of the register (SP) is incremented by 2. It is for the user to ensure that the proper RETURN ADDR is available correctly on the top of the stack before asking the processor to execute the RET instruction

The micro RTL flow is,

Machine Cycle- 1:

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

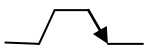
T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (IR) ⟵ AD_7-AD_0

T_4: μp decodes the opcode. RET = 1.

Machine Cycle- 2:

MRMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=0

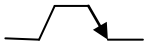
T_1: A_{15}-A_8 ⟵ (SPH), AD_7-AD_0 ⟵ (SPL), ALE = 

T_2: $\overline{RD}$ = 0, (SP) ← (SP) +1, AD_7-AD_0 ← M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (PCH) ← AD_7-AD_0

Machine Cycle- 3:

MRMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=0

T_1: A_{15}-A_8 ← (SPH), AD_7-AD_0 ← (SPL), ALE = 

T_2: $\overline{RD}$ = 0, (SP) ← (SP) +1, AD_7-AD_0 ← M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (PCH) ← AD_7-AD_0

Thus it requires 3 machine cycles and 10 states. For $2MH_3$ internal clock the time required is 5μsec. The addressing mode is register indirect addressing mode as the return address is to be read from top of the stack pointed to by (SP).

Lecture-31

6. Rcond: This is a conditional return statement. It is also a part of the subroutine. Whenever this instruction is executed, μp checks the conditional flags. If the condition is found true then the μp returns to the main program by loading the (PC) with the return address stored at the top of the stack. If the condition is found not true then the next instruction of the subroutine will be executed. This is a single byte instruction. The operation code format is

1	1	C	C	C	0	0	0

N

There are 8 variations in this statement depending upon CCC. They are RNZ, RZ, RNC, RC, RPO, RPE, RP and RM. The addressing mode is register indirect because the return address is available in the memory location pointed by SP. The macro RTL implemented is,

If CCC= TRUE

(PCL) ⟵ M [(SP)]

(PCH) ⟵ M [(SP) +1]

(SP) ⟵ (SP) + 2

Else

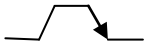
Nothing.

where (PCH,PCL) is the return address. If the condition tested is TRUE, then the 8-bit data (lower order 8-bit return address) from the top of the stack pointed by (SP) is moved to (PCL). The 8-bit data (higher order 8-bit return address) from next higher memory location whose address is one more than the content of (SP) is moved to (PCH). The content of the register (SP) is incremented by 2 in this

process so that it always point to the top of the stack. Control is transferred back to the main program. If the condition is not found TRUE, the control goes to the next instruction of subroutine in sequence. The macro RTL implementation flowchart is shown below:

Machine Cycle- 1:

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ← (PCH), AD_7-AD_0 ← (PCL), ALE = 

T_2: $\overline{RD}$ = 0, (PC) ← (PC) +1, AD_7-AD_0 ← M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (IR) ← AD_7-AD_0

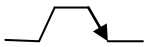
T_4: μp decodes the opcode. Rcond = 1.

T_5: } If condition CCC is TRUE then

T_6: } go for MC-2 else nothing.

Machine Cycle- 2:

MRMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=0

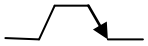
T_1: A_{15}-A_8 ← (SPH), AD_7-AD_0 ← (SPL), ALE = 

T_2: $\overline{RD}$ = 0, (SP) ← (SP) +1, AD_7-AD_0 ← M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (PCL) ← AD_7-AD_0

Machine Cycle- 3:

MRMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=0

T_1: A_{15}-A_8 ← (SPH), AD_7-AD_0 ← (SPL), ALE = 

T_2: $\overline{RD}$ = 0, (SP) ← (SP) +1, AD_7-AD_0 ← M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (PCH) ← AD_7-AD_0

Thus, if condition is TRUE it requires 3 machine cycles and 12 states otherwise one machine cycle OFMC and 6 states. For $2MH_3$ internal clock the time required is 6μsec or 3 μsec.

7. RST n: This is a single byte unconditional subroutine call instruction. The address of the subroutine is fixed depending upon the decimal number 'n' in the instruction. The decimal number 'n' can change from 0 to 7. The corresponding binary number $(NNN)_2$ is available as $D_5D_4D_3$ bits of the operation code.

N N N	'n'	RST-n
0 0 0	0	RST-0
0 0 1	1	RST -1
0 1 0	2	RST-2
0 1 1	3	RST-3
1 0 0	4	RST-4
1 0 1	5	RST-5
1 1 0	6	RST-6
1 1 1	7	RST-7

The operation code format is

1	1	N	N	N	1	1	1

N

There are 8 variations of this instructions depending upon NNN as specified in the table above. The subroutine starting address is calculated as follows:

Multiply the decimal number 'n' by 8. Convert it to the corresponding 2 digit hexa decimal number $(Y_1Y_0)_H$ (= 8 x n). Append

$(00)_H$ for the two most significant hexa digits $(Y_3Y_2)_H$ so that a 16-bit address $(Y_3Y_2Y_1Y_0)_H \longrightarrow (00\ Y_1Y_0)_H$ is obtained. E.g. the subroutine starting address for RST-5 is $(8 \times 5)_D = 40_D = 28_H \longrightarrow (0028)_H$.

The macro RTL implemented is,

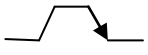
$$M[(SP) - 1] \longleftarrow (PCH)$$
$$M[(SP) - 2] \longleftarrow (PCL)$$
$$(SP) \longleftarrow (SP) - 2$$
$$(PC) \longleftarrow (8 \times n)$$

where (PCH,PCL) is the return address. The higher order 8-bits of the return address are moved to the memory location whose address is one less than the content of (SP). The lower order 8-bits of the return address are moved to the memory location whose address is two less than the content of (SP). The content of the register (SP) is decremented by 2 in this process so that it always point to the top of the stack. Control is transferred to the instruction whose address is calculated according to number 'n' specified in the instrcution. The addressing mode used is register indirect addressing mode because the return address has to be saved on the top of the stack pointed to by 16-bit register (SP).

The macro RTL flow is as shown below:

Machine Cycle- 1:

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-$A_8 \longleftarrow$ (PCH), AD_7-$AD_0 \longleftarrow$ (PCL), ALE = 

T_2: $\overline{RD}$ = 0, (PC) $\longleftarrow$ (PC) +1, AD_7-$AD_0 \longleftarrow$ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (IR) ⟵ AD_7-AD_0

T_4: μp decodes the opcode. RST-n = 1.

T_5: } Address (8 x n) is calculated and stored in (W,Z) register pair.

T_6: } (SP) ⟵ (SP) -1

Machine Cycle- 2:

MWRMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (SPH), AD_7-AD_0 ⟵ (SPL), ALE = ⎍

T_2: $\overline{WR}$ = 0, AD_7-AD_0 ⟵ (PCH)

T_3: $\overline{WR}$ = 1, ↑ , M(AB) ⟵ AD_7-AD_0, (SP) ⟵ (SP) -1

Machine Cycle- 3:

MWRMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (SPH), AD_7-AD_0 ⟵ (SPL), ALE = ⎍

T_2: $\overline{WR}$ = 0, AD_7-AD_0 ⟵ (PCL)

T_3: $\overline{WR}$ = 1, ↑ , M(AB) ⟵ AD_7-AD_0,

Machine Cycle- 4 or (Machine Cycle-1 of next instruction):

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (W), AD_7-AD_0 ⟵ (Z), ALE = ⎍

T_2: $\overline{RD}$ = 0, FEO [(PC) ⟵ (W, Z) +1], AD_7-AD_0 ⟵ M(AB)

Thus, it requires three machine cycles OFMC, & two MWRMC and total 12 states. Using a 2 MH_3 internal clock it requires 6μsec.

One important point while writing a subroutine program starting from the fixed location as calculated.

Consider for example RST 5 & RST 6. The corresponding starting addresses of the subroutine are 0028_H for RST 5 & 0030_H for RST 6. There is RST 5.5 interrupt control signal input at pin no 9 of the μp. This also corresponding to an interrupt service routine starting from $002C_H$. These addresses are shown in fig.

0028	C3	RST -5
0029	Y_1Y_0	$Y_3Y_2Y_1Y_0$ RST- 5 subroutine address
$002A_H$	Y_3Y_2	
$002B_H$	00	No operation
$002C_H$	C3	RST 5.5
$002D_H$	Z_1Z_0	$Z_3Z_2Z_1Z_0$ RST 5 subroutine address
$002E_H$	Z_3Z_2	
$002F_H$	00	No operation
0030_H	C3	RST-6
0031_H	X_1X_0	$X_3X_2X_1X_0$ RST- 6 subroutine address
0032_H	X_3Y_2	
0033_H	00	

The figure shows that there are only four bytes, possible for each subroutine link and it is difficult to write subroutine here. Therefore, an unconditional jump instruction if 3-byts is written here. First byte will be $C3_H$ (jump Instruction) and 2^{nd} & 3^{rd} bytes will be the address of some memory locations which is the starting address of subroutine. This is known as subroutine link address 4^{th} byte is no operation.

8. PCHL: This is a single byte instruction. The operation code is

1	1	1	0	1	0	0	1

N

There is no variation of this instruction. The macro RTL implemented is,

(PCL) ⟵ (L)
(PCH) ⟵ (H)
Or (PC) ⟵ (H,L)

The meaning of the instruction is “Move the content of register (H) to the higher byte of program counter (PCH) and move the content of register (L) to lower byte of program counter (PCL)”. In this way the control is transferred to the memory location whose address is available in (H, L) register pair. This is called register indirect jump instruction. This is the only instruction available in 8085A which allows the user to obtain a jump address from a register pair. This instruction uses register addressing mode. Other jump instructions use immediate and register indirect addressing mode.

This instruction is very useful in implementing select structure of the software program. The macro RTL flow is as shown below:

Machine Cycle- 1:

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = ⎍

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (IR) ⟵ AD_7-AD_0

T_4: μp decodes the opcode. PCHL = 1.

T_5: (PCL) ⟵ (L)

T_6: (PCH) ⟵ (H)

Thus 6 states are required no flags affected.

Lecture-32

STACK GROUP:

This group of instructions manipulates the stack. Unless otherwise specified, condition flags are not affected by any instruction of this group. It is the user responsibility to define stack area and to initialize the stack pointer (SP) with the bottom address before these instructions are used.

1. PUSH rp: This is a single byte instruction. The meaning of the instruction is "Push or save the contents of register pair (rpH, rpL) on top of the stack". The macro RTL implemented is,

$$M[(SP) - 1] \longleftarrow (rpH)$$
$$M[(SP) - 2] \longleftarrow (rpL)$$
$$(SP) \longleftarrow (SP) - 2$$

The content of the higher order register of register pair (rp) is moved to the memory location whose address is one less than the content of register (SP). The content of the lower order register of register pair (rp) is moved to the memory location whose address is two less than the content of register SP. In this process, the content of the (SP) register is decremented by 2. The operation code format is,

1	1	R	P	0	1	0	1

N

This instruction has 3 variations for register pair (B, C), (D, E) & (H, L) for three combinations of RP as $(00)_2$, $(01)_2$ and $(10)_2$ as below:

R P	Register Pair
0 0	(B, C)
0 1	(D, E)
1 0	(D, E)

SP is not allowed in this instruction. The addressing mode is register indirect addressing mode. The macro RTL flow is,

Machine Cycle- 1:

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE =

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (IR) ⟵ AD_7-AD_0

T_4: μp decodes the opcode. PUSH rp = 1.

T_5:

T_6: (SP) ⟵ (SP) -1

Machine Cycle- 2:

MWRMC: Status signals IO/$\overline{M}$=0, S_1=0, S_0=1

T_1: A_{15}-A_8 ⟵ (SPH), AD_7-AD_0 ⟵ (SPL), ALE =

T_2: $\overline{WR}$ = 0, AD_7-AD_0 ⟵ (PCH)

T_3: $\overline{WR}$ = 1, ↑ , M(AB) ⟵ AD_7-AD_0, (SP) ⟵ (SP) -1

Machine Cycle- 3:

MWRMC: Status signals IO/$\overline{M}$=0, S_1=0, S_0=1

T_1: A_{15}-A_8 ⟵ (SPH), AD_7-AD_0 ⟵ (SPL), ALE =

T_2: $\overline{WR}$ = 0, AD_7-AD_0 ⟵ (PCL)

T_3: $\overline{WR}$ = 1, ↑ , M(AB) ⟵ AD_7-AD_0,

Thus, it requires three machine cycles OFMC, two MWRMC and a total 12 states. Using a 2 MH_3 internal clock it requires either 6μsec.

2. PUSH PSW (Push Processor Status Word): This is also a single byte instruction. The meaning of the instruction is “Save or push the processor status word, comprising of (A) and (F) register, on the top of the stack”. The accumulator & flag register together form a 16- bit word known as the processor status word (PSW), ACC occupies the higher order 8-bits and flag register occupies the lower order 8-bits in PSW. The macro RTL implemented is,

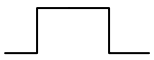
$$M[(SP) - 1] \longleftarrow (A)$$
$$M[(SP) - 2] \longleftarrow (F)$$
$$(SP) \longleftarrow (SP) - 2$$

The content of the accumulator (A) is moved to the memory location whose address is one less than the content of register (SP). The content of flag register (F) is moved to the memory location whose address is two less than the content of register SP. In this process, the content of the (SP) register is decremented by 2. The operation code format is,

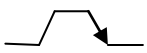
1	1	1	1	0	1	0	1	N

This instruction has no variations. The addressing mode is register indirect addressing mode. The macro RTL flow is,

Machine Cycle- 1:



OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

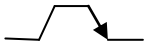
T_3: $\overline{RD}$ = 1, ↑ , (IR) ⟵ AD_7-AD_0

T_4: μp decodes the opcode. PUSH PSW = 1.

T_5:
T_6: (SP) ⟵ (SP) -1

Machine Cycle- 2:

MWRMC: Status signals IO/$\overline{M}$=0, S_1=0, S_0=1

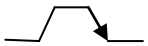
T_1: A_{15}-A_8 ⟵ (SPH), AD_7-AD_0 ⟵ (SPL), ALE = 

T_2: $\overline{WR}$ = 0, AD_7-AD_0 ⟵ (A)

T_3: $\overline{WR}$ = 1,  , M(AB) ⟵ AD_7-AD_0, (SP) ⟵ (SP) -1

Machine Cycle- 3:

MWRMC: Status signals IO/$\overline{M}$=0, S_1=0, S_0=1

T_1: A_{15}-A_8 ⟵ (SPH), AD_7-AD_0 ⟵ (SPL), ALE = 

T_2: $\overline{WR}$ = 0, AD_7-AD_0 ⟵ (F)

T_3: $\overline{WR}$ = 1,  , M(AB) ⟵ AD_7-AD_0,

Thus, it requires three machine cycles OFMC, two MWRMC and a total 12 states. Using a 2 MH_3 internal clock it requires either 6μsec.

3. POP rp: It is a single byte instruction. The meaning of the instruction is “Pop or load the content form the top of the stack into register pair (rp)”. The macro RTL implemented is,

(rpH) ⟵ M [(SP)]
(rpL) ⟵ M [(SP) +1]
(SP) ⟵ (SP) + 2

The content of the memory location pointed by the content of stack pointer (SP) is moved to higher order register of register pair (rp) and

the content of the next memory location whose address is one more than the content of (SP) is moved to lower order register of register pair (rp). In this process, the content of the (SP) register is incremented by 2. The operation code format is,

1	1	R	P	0	0	0	1

N

This instruction has 3 variations for register pair (B, C), (D, E) & (H, L) for three combinations of RP as $(00)_2$, $(01)_2$ and $(10)_2$ as below:

R P	Register Pair
0 0	(B, C)
0 1	(D, E)
1 0	(D, E)

SP is not allowed in this instruction. The addressing mode is register indirect addressing mode. The macro RTL flow is,

Machine Cycle- 1:

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE =

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (IR) ⟵ AD_7-AD_0

T_4: μp decodes the opcode. POP rp = 1.

Machine Cycle- 2:

MRMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=0

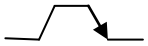
T_1: A_{15}-A_8 ⟵ (SPH), AD_7-AD_0 ⟵ (SPL), ALE =

T_2: $\overline{RD}$ = 0, (SP) ⟵ (SP) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (rpL) ⟵ AD_7-AD_0

Machine Cycle- 3:

MRMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=0

T_1: A_{15}-A_8 ⟵ (SPH), AD_7-AD_0 ⟵ (SPL), ALE = 

T_2: $\overline{RD}$ = 0, (SP) ⟵ (SP) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (rpH) ⟵ AD_7-AD_0

Thus, it requires three machine cycles OFMC, two MRMC and a total 10 states. Using a 2 MH_3 internal clock it requires either 5μsec.

4. POP PSW: (Pop Processor Status Word) It is a single byte instruction. The meaning of the instruction is "Pop or load the content form the top of the stack into processor status word comprising of (A) and (F) register". The accumulator & flag register together form a 16-bit word known as the processor status word (PSW), ACC occupies the higher order 8-bits and flag register occupies the lower order 8-bits in PSW. The macro RTL implemented is,

(A) ⟵ M [(SP)]

(F) ⟵ M [(SP) +1]

(SP) ⟵ (SP) + 2

The operation code format is,

1	1	1	1	0	0	0	1

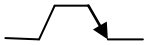
N

It has no variations. The content of the memory location pointed by the content of stack pointer (SP) is moved to accumulator (A) and the content of the next memory location whose address is one more than the content of (SP) is moved to flag register (F). In this process, the

content of the (SP) register is incremented by 2. The addressing mode is register indirect addressing mode. The macro RTL flow is,

Machine Cycle- 1:

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

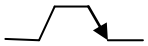
T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (IR) ⟵ AD_7-AD_0

T_4: μp decodes the opcode. POP PSW = 1.

Machine Cycle- 2:

MRMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=0

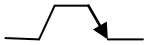
T_1: A_{15}-A_8 ⟵ (SPH), AD_7-AD_0 ⟵ (SPL), ALE = 

T_2: $\overline{RD}$ = 0, (SP) ⟵ (SP) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (F) ⟵ AD_7-AD_0

Machine Cycle- 3:

MRMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=0

T_1: A_{15}-A_8 ⟵ (SPH), AD_7-AD_0 ⟵ (SPL), ALE = 

T_2: $\overline{RD}$ = 0, (SP) ⟵ (SP) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (A) ⟵ AD_7-AD_0

Thus, it requires three machine cycles OFMC, two MRMC and a total 10 states. Using a 2 MH_3 internal clock it requires either 5μsec. Flags affected are Z, S, AC, P, and CY.

5. XTHL: It is a single byte instruction. The meaning of the instruction is “Exchange the contents from the top of the stack with the contents of (H, L) register pair”. The macro RTL implemented is,

M (SP) ⟷ (L)

M (SP+1) ⟷ (H)

The content of register (L) is exchanged with the content of the memory location whose address is specified by the content of (SP). The content of register (H) is exchanged with content the memory location whose address is one more than the content of (SP). The operation code format is,

1	1	1	0	0	0	1	1	N

It has no variations. The addressing mode is register indirect addressing mode none flags affected. The micro RTL flow is,

Machine Cycle- 1:

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ← (PCH), AD_7-AD_0 ← (PCL), ALE = ⎍

T_2: $\overline{RD}$ = 0, (PC) ← (PC) +1, AD_7-AD_0 ← M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (IR) ← AD_7-AD_0

T_4: μp decodes the opcode. XTHL = 1.

Machine Cycle- 2:

MRMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=0

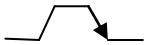
T_1: A_{15}-A_8 ← (SPH), AD_7-AD_0 ← (SPL), ALE = ⎍

T_2: $\overline{RD}$ = 0, (SP) ← (SP) +1, AD_7-AD_0 ← M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (Z) ← AD_7-AD_0

Machine Cycle- 3:

MRMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=0

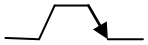
T_1: A_{15}-A_8 ⟵ (SPH), AD_7-AD_0 ⟵ (SPL), ALE = 

T_2: $\overline{RD}$ = 0, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (W) ⟵ AD_7-AD_0

Machine Cycle- 4:

MWRMC: Status signals IO/$\overline{M}$=0, S_1=0, S_0=1

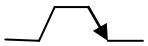
T_1: A_{15}-A_8 ⟵ (SPH), AD_7-AD_0 ⟵ (SPL), ALE = 

T_2: $\overline{WR}$ = 0, AD_7-AD_0 ⟵ (H)

T_3: $\overline{WR}$ = 1, ↑ , M(AB) ⟵ AD_7-AD_0, (SP) ⟵ (SP) -1,
(H) ⟵ (W)

Machine Cycle- 3:

MWRMC: Status signals IO/$\overline{M}$=0, S_1=0, S_0=1

T_1: A_{15}-A_8 ⟵ (SPH), AD_7-AD_0 ⟵ (SPL), ALE = 

T_2: $\overline{WR}$ = 0, AD_7-AD_0 ⟵ (L)

T_3: $\overline{WR}$ = 1, ↑ , M(AB) ⟵ AD_7-AD_0,
(L) ⟵ (Z)

Thus, it requires five machine cycles OFMC, two MRMC & two MWRMC and a total 16 states. Using a 2 MH_3 internal clock it requires either 8μsec. No flag is affected. The common use of XTHL instruction is to modify the return address from subroutine depending on the result of an execution.

6. SPHL: It is a single byte instruction. The meaning of the instruction is "Move the contents of register pair (H, L) to 16-bit register stack pointer (SP). The macro RTL implements is,

(SPH) ⟵ (L)

(SPH) ⟵ (H)

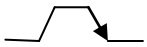
The operation code format is,

1	1	1	1	1	0	0	1	N

It has no variations. The addressing mode is register addressing mode. None of the flags are affected. The micro RTL flow is,

Machine Cycle- 1:

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1,  , (IR) ⟵ AD_7-AD_0

T_4: μp decodes the opcode. SPHL = 1.

T_5: (SPL) ⟵ (L),

T_6: (SPH) ⟵ (H)

It requires one machine cycle OFMC and 6 states.

Lecture-33

MACHINE CONTROL INSTRUCTIONS:

1. EI (Enable interrupts): The interrupt system is disabled just after RESET operation. There is an internal INTE F/F (Interrupt enable flip-flop) which is reset in RESET operation. It is a single byte instruction. The operation code format is

1	1	1	1	1	0	1	1	N

The interrupt system is enabled using EI instruction. When this instruction is executed, then INTE F/F is set so that all the interrupts are enabled and 8085 A will recognize external interrupt request except those that are masked. The INTE F/F is set following the execution of the next instruction. It has no variations and none of the flags are affected. The macro RTL flow is,

Machine Cycle- 1:

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = ⎍

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (IR) ⟵ AD_7-AD_0

T_4: μp decodes the opcode. EI = 1.

Machine Cycle- 2 or (Machine Cycle-1 of next instruction):

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = ⎍

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1], AD_7-AD_0 ⟵ M(AB)

FEO[INTE F/F ⟵ 1]

Thus, it requires one machine cycle OFMC & a total of 4 states.

Note: Interrupts are not recognized during the execution of EI instruction. Placing an EI instruction code on the bus in response to an $\overline{INTA}$ pulse during an interrupt acknowledge machine cycle is prohibited.

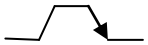
2. DI (disable interrupts): It is also a single byte instruction. The interrupt system is disabled immediately following the execution of the DI instruction, i.e, INTE F/F is reset. The operation code format is

1	1	1	1	0	0	1	1	N

When this instruction is executed, then INTE F/F is reset so that all the interrupts are disabled except TRAP and 8085 A will not recognize any external interrupt request. The INTE F/F is reset following the execution of the next instruction. It has no variations and none of the flags are affected. The macro RTL flow is,

Machine Cycle- 1:

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

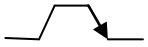
T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (IR) ⟵ AD_7-AD_0

T_4: μp decodes the opcode. DI = 1.

Machine Cycle- 2 or (Machine Cycle-1 of next instruction):

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1], AD_7-AD_0 ⟵ M(AB)

FEO[INTE F/F ⟵ 0]

Thus, it requires one machine cycle OFMC & a total of 4 states.

Note: Interrupts are not recognized during the execution of DI instruction. Placing an DI instruction code on the bus in response to an $\overline{INTA}$ pulse during an interrupt acknowledge machine cycle is prohibited.

3. NOP (No operation): It is a single byte instruction. The meaning of the instruction is "No operation is performed". The registers and flags are unaffected. The operation code format is

0	0	0	0	0	0	0	0	N

It has no variation and none of the flag is affected. The micro RTL flow is

Machine Cycle- 1:

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = ⎍

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (IR) ⟵ AD_7-AD_0

T_4: μp decodes the opcode. NOP = 1.

Thus one machine cycle OFMC of 4 states is required. This instruction has three main uses:

1) It is frequently used in delay loops to introduce a delay of 4 states.
2) It is also used to interface slower peripheral devices with 8085A.
3) It is also used to remove or introduce any instruction in the program.

4. HLT: (Halt) It is a single byte instruction. When this instruction is executed the processor is stopped, i.e, it stops fetching and executing instructions from the program memory. The address bus, data bus and control bus are tri-stated. The operation code format is

0	1	1	1	0	1	1	0	N

The macro RTL implemented is

HALT F/F ⟵ 1

It has no variation. The registers & flags are unaffected. When this instruction is fetched & decoded during T_4 state of OFMC cycle, then an internal F/F/ known as HALT F/F is set and in the immediate next T_1 state of next OFMC, this HALT F/F is checked to see whether it is SET or not. If it is found SET, μp goes to HALT state otherwise it goes to T_2 state.

The micro RTL flow implemented is

Machine Cycle- 1:

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = ⎍

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (IR) ⟵ AD_7-AD_0

T_4: μp decodes the opcode. HLT = 1. HALT F/F ⟵ 1.

Machine Cycle- 2:

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = ⎍

If HALT F/F is found SET, processor enters in HALT state else it goes to T_2 state.

Thus, this instruction requires 5 states.

RIM & SIM: These instructions are dual purpose instructions and used both for interrupts mask control and serial communication. The details will be discussed later on. In brief, the interrupt mask control and serial communication is discussed below:

The 8085A has three interrupt inputs RST 5.5, RST 6.5 and RST 7.5 which are referred to as vectored interrupts. A proper input on these 8085A interrupt inputs will cause the program being executed by the CPU to branch to known locations:

I.e. $002C_H$ for RST 5.5

0034_H for RST 6.5

$003C_H$ for RST 7.5

These locations should have the first instruction of the corresponding interrupt service subroutine i.e. the jump instruction. The RST 5.5 & RST 6.5 inputs are level sensitive interrupts where as RST 7.5 is edge sensitive interrupt. The RST5.5 & RST 6.5 input will have to be held high till the 8085A acknowledge the interrupt. A low to high transition of RST7.5 sets an internal flip flop so this input can then be lowered without losing the interrupt request.

The 8085A also has an INTR interrupt & one non-maskable vectored interrupt called TRAP. A high at the TRAP input after a low to high edge causes the program to branch to 0024H.

The 8085A also provides on chip facility for serial I/O via the SID& SOD lines. The SID (serial input data) line can be used to input serial data to the 8085A. While the SOD (serial output data) line

outputs serial data from the 8085A. Inputting & outputting serial data on these pins are also achieved using the RIM & SIM instruction.

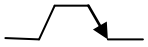
5. SIM (Set interrupt mask): This is a single byte instruction. It operates on the contents of (A) before SIM is executed. The operation code format is

0	0	1	1	0	0	0	0	N

It has no variation. The registers & flags are unaffected. The micro RTL flow implemented is

Machine Cycle- 1:

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

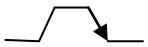
T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (IR) ⟵ AD_7-AD_0

T_4: μp decodes the opcode. SIM = 1.

Machine Cycle- 2:

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

FEO [Interrupt mask ⟵ (A)]

Thus, this instruction requires one machine cycle OFMC and of 4 states.

The SIM instruction uses the contents of the accumulation to perform the following functions:

(i) Program the interrupts mask for the RST 5.5, RST 6.5, & RST 7.5 hardware interrupts. The interrupts can be masked or unmasked by controlling A_0, A_1, & A_2 bits of accumulator before using SIM. If any bit is'1', then the corresponding interrupt is masked i.e, these interrupts will not be recognized. If any of these bits is '0', the corresponding interrupt is unmasked, i.e, if the corresponding interrupt occurs, it will be acknowledged. These bits will affect the interrupts only if MSE (mask set enable) bit (A_3) is also 1. If A_3 is '0', when the SIM instruction is executed, the interrupt mask register is not changed.

(ii) RST 7.5 is edge sensitive interrupt (LOW ⟶ HIGH). A pulse at the RST 7.5 always sets an internal R7.5 F/F even if the jump to the service routine is inhibited by masking. If interrupts are disabled at the time the RST 7.5 pulse occurs, this input will still be recognized later since the R7.5 F/F has already been SET by the pulse. This F/F can be cleared (or Reset) by keeping bit A_4=1 when SIM is executed. This may be required if we do not want to service an earlier RST 7.5 interrupted.

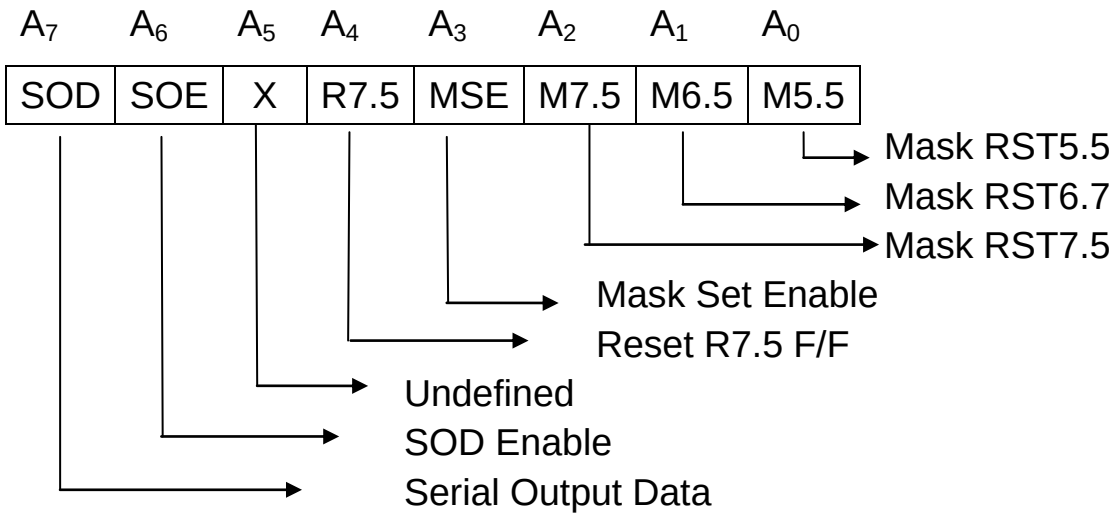
The R7.5 F/F is also cleared by a $\overline{\text{RESET IN}}$ signal input or when the CPU acknowledges a RST 7.5 interrupt. All three mask bits (A_2, A_1, A_0) are also set by $\overline{\text{RESET IN}}$ i.e, all vectored interrupts (except TRAP) are masked and, therefore, disabled.

(iii) Load the SOD (serial output data) latch.

The SOD output of the 8085A shows the status of a 1-bit output port. Execution of the SIM instruction sets the output port content to

that of A_7 provided the serial output enable A_6 is also 1. If SOE = A_6=0, the contents of the output port is unaffected SOD is reset by the $\overline{\text{RESET IN}}$ input.

Using SOE & MSE properly one can use the SIM instruction is different ways the format of accumulator for SIM instruction is



This instruction requires one machine cycle of four states. No flags are affected.

E.g. (1) Send a '1' to the SOD output line. The following sequence of instruction shall do the word.

MVI A, $C0_H$ $(1100\ 0000)_2$

SIM

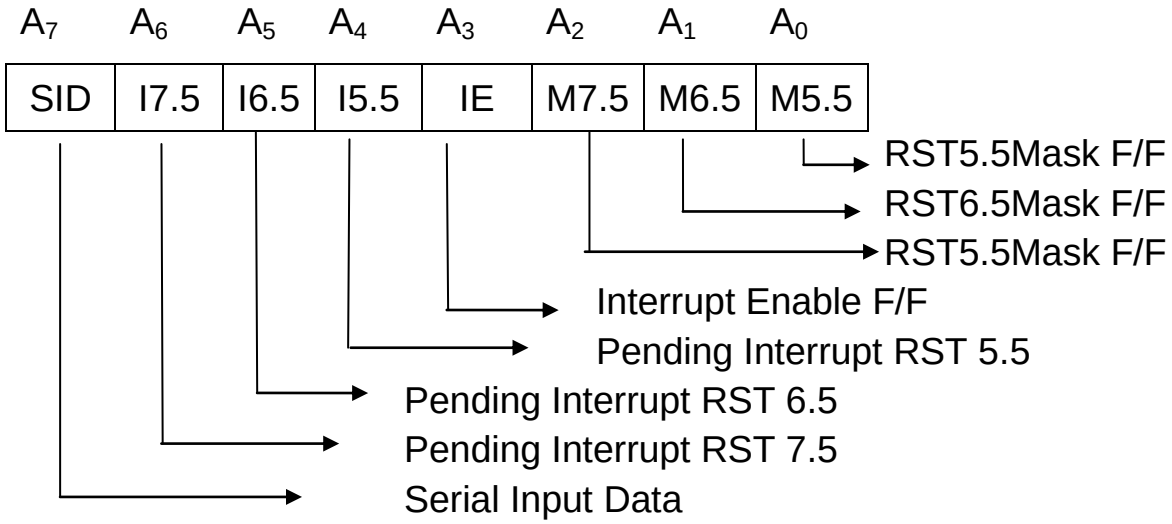
(2) The following sequence of instruction shall unmask RST 6.5 interrupt control signal input & mask all other interrupts after power on.

MVI A, $0D_H$ $(0000\ 1101)_2$

SIM

EI

6. RIM: (Read interrupt mask): Whenever this instruction is executed the status of the mask F/Fs, INTE F/F the SID input & of pending interrupts is read into accumulation as follows.



The RIM instruction loads data into accumulation relating to interrupts & the serial input. After RIM is executed the content of A are as follows:

(i) Current interrupt enables status for the RST 5.5, 6.5, 7.5 hardware interrupts in A_0 A_1 & A_2 bits (1= if mask disabled or 0 if they are enabled).

(ii) Current interrupt enable flag status in bit A_3. (=1 interrupts enabled) except immediately following a TRAP interrupt. Following a TRAP interrupt, IE = A3 reports the status of interrupts (enabled/$\overline{\text{disabled}}$) prior to the TRAP interrupt. This is useful to retrieve current interrupt status following TRAP. This is important since TRAP is a non mask able interrupt which can happen at any time.

(iii) Hardware pending interrupts (Interrupt received but not serviced). on the RST 7.5, RST 6.5 & RST 5.5 lines. A '1' at A_6, A_5, A_4, respectively indicate that RST 7.5, 6.5 & 5.5 interrupts are pending.

(iv) Transfer the bit present at 8085A's SID input to A_7.

Apart from the letting the CPU get the SID input, the RIM instruction is primarily used to monitor interrupt status. e.g.

a) Monitor whether or not an interrupt is pending without actually servicing it.
b) Check using IE properly if CPU is currently, servicing an interrupt.
c) By properly using RIM & SIM one can design any other priorrity structure.

The operation code format is

0	0	1	0	0	0	0	0	N

It has no variation. The registers & flags are unaffected. The micro RTL flow implemented is

Machine Cycle- 1:

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = ⎍

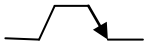
T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (IR) ⟵ AD_7-AD_0

T_4: μp decodes the opcode. RIM = 1.

Machine Cycle- 2:

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

FEO [(A) ⟵ Interrupt mask]

Thus, this instruction requires one machine cycle OFMC and of 4 states.

Lecture-34

UNSPECIFIED OP CODES OF THE 8085A

If one examines the instruction set of 8085A, one finds that out of the 256 possible opcode with 8-bits, Intel had announced only 246 opcodes or 246 instructions. There are no instructions in the instruction set corresponding to the 10 missing opcodes. Users have since reported that Intel 8085A does have instructions corresponding to these missing opcodes. These instructions reduce the program length as well as execution time. The flags specified by Intel are S, Z, AC, P, CY located in the flag register as

S	Z	X	AC	X	P	X	CY

Users have reported that there are two more flag bits, at bit-1 and bit-5 positions, having same useful meaning.

S	Z	X_5	AC	X	P	V	CY

where,

V = Overflow bit. This bit is set to 1 if overflow occurs in 2's complement addition/ subtraction/ DCX/ INX for 8 and 16-arithmetic operation.

X_5 = This bit has been named for its position in the flag register. It does not resemble any normal flag but this bit is affected as per the following expression.

$$X_5 = S_1.S_2 + S_1.R + S_2.R$$

where, S_1= sign of operand 1

S_2= sign of operand 2

R = sign of result.

For subtraction & comparison, replace S_1 by $\overline{S}_2$ where operand 2 is the subtrahend i.e., Result = Operand 1- Operand 2.

The only use for this bit is found as an unsigned overflow indication resulting from a data change of $FFFF_H$ to 0000_H on executing the instruction INX and as an unsigned underflow indicator from a data change of 0000_H to $FFFF_H$ on executing DCX.

The ten new instructions, include seven opcode that involve the processing of register pairs, two that involve jump operation with X_5 flag bit and one that performs a conditional restart on the overflow flag bit V. The various instructions are discussed below:

1. DSUB: (Double byte subtraction): This is a single byte instruction. The meaning of the instruction is that the contents of register pair (B,C) are subtracted foam the contents of register pair (H,L) and result is stored back in register pair (H,L). The macro RTL implements is

$$(H, L) \longleftarrow (H, L) - (B, C)$$

The opcode of the instruction is $(0B)_H$. All seven condition flag are affected as per the result. It requires 3 machine cycles OFMC & two BIMC and 10 states similar to DAD rp instruction. The addressing mode is register addressing mode.

In the absence of this instruction, the operation can be performed by writing a sequence of specified instructions as below:

```
MOV     A,  L
SUB     C
MOV     L,  A
MOV     A,  H
SBB     B
MOV     H,  A
```

The length of the program as well as execution time both increases.

2. ARHL: (Arithmetic Right-shift HL register pair) This is a single byte instruction. The meaning of the instruction is that the contents of register pair (H, L) are shifted right arithmetically by one bit. The upper most bit is duplicated and the lower bit is shifted into the carry bit. The result is placed back into the (H, L) register pair. The macro RTL implemented is

$(H_7) \leftarrow (H_7)$, $(H_{n-1}) \leftarrow (H_n)$,

$(L_7) \leftarrow (H_0)$, $(L_{n-1}) \leftarrow (L_n)$, $(CY) \leftarrow (L_0)$

The opcode of the instruction is $(10)_H$. Only (CY) flag is affected. It requires 2 machine cycles OFMC and one BIMC and 7 states. The addressing mode is register addressing mode.

In the absence of this instruction, the operation can be performed by writing a sequence of specified instructions as below:

```
MOV     A,  H
RAL
MOV     A,  H
RAR
MOV     H,  A
MOV     A,  L
RAR
MOV     L,  A
```

The length of the program as well as execution time both increases.

3. RDEL: (Rotate (D,E) register pair left through carry) This is a single byte instruction. The operation code is 18_H. The meaning of the instruction is “Rotate the contents of register pair (D, E) left by one bit through the carry flag”. The lower order bit is set equal to the CY flag

and the CY flag is set to the value shifted out of the higher order bit. The result is placed book into the DE register pair. The macro RTL implemented is

$(CY) \longleftarrow (D_7), (D_{n+1}) \longleftarrow (D_n),$

$(D_0) \longleftarrow (E_7), (E_{n+1}) \longleftarrow (E_n), (E_0) \longleftarrow (CY)$

Only the CY flag and the V flag bits are affected. It requires three machine cycles OFMC & 2 BIMC and a total of 10 states. The addressing mode is register addressing mode.

4. LDHI: (Load (D, E) register pair with (H, L) plus 8-bit immediate data). This is a 2 byte instruction. The operation code format is

28	N
$<B_2>$	N+1

The macro RTL implemented is

$(D, E) \longleftarrow (H, L) + <B_2>$

The meaning of the instruction is the contents of register pair (H, L) are added to the 8-bit immediate data byte and the result is stored in register pair (D, E). No condition flags are affected.

It requires three machine cycles OFMC, one MRMC & one BIMC and a total of 10 states. The addressing mode is immediate & register addressing mode the second byte is called offset.

5. LDSI: (Load (D, E) register pair with (SP) plus 8-bit immediate data byte). This is two byte instruction. The opcode format is

38	N
$<B_2>$	N+1

The macro RTL implemented is

$$(D,E) \longleftarrow (SP) + <B_2>$$

The meaning of the instruction is “Add the contents of register pair (SP) to the 8-bit data byte immediately available as a2ndbyte of the instruction & store the result in register pair (D,E)”. No condition flag are affected.

It requires 3 machine cycles OFMC,1 MRMC and 1 BIMC and a total of 10 states. The addressing mode is immediate register addressing mode.

6. RST V (Restart on overflow). This is a single byte instruction. The opcode is $(CS)_H$. the meaning is the overflow flag V is set, the actions specified above are performed, otherwise control continues sequentially. The macro RTL implemented is shown below:

If overflow flag V=1

$M[(SP) – 1] \longleftarrow (PCH)$

$M[(SP) – 2] \longleftarrow (PCL)$

$(SP) \longleftarrow (SP) – 2$

$(PC) \longleftarrow 0040_H$

Otherwise

$(PC) \longleftarrow (PC) + 1$

where (PC) on right hand side is having the address of the opcode.

It requires 1 or 3 machine cycles and 6 or 12 states. The addressing mode is register indirect addressing mode. None of the flags are affected.

7. SHLX: (Store (H,L) register pair indirect through (D,E) register pair). This is a single byte instruction. The opcode is $(D9)_H$. The meaning of the instruction is "Move the content of register (L) to the memory location whose address is in (D,E) register pair and move the content of register (H) to the memory location whose address is one more than the (D,E) register pair. The macro RTL implemented is

$$M[(D,E)] \longleftarrow (L)$$
$$M[(DE) + 1] \longleftarrow (H)$$

It requires 3 machine cycles OFMC & two MWRMC and a total of 10 states. The addressing mode is register indirect addressing mode. None of the flags are affected.

8. JNX5: (Jump on not X5) This is a 3-byte conditional jump instruction. The operation code format is

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

This instruction tests the X_5 bit. If the X_5 bit is '0', the control is transferred to the instruction whose address is specified in 2nd byte & 3rd byte of the instruction, otherwise control continues sequentially. The macro RTL implemented is shown below:

If X_5 flag = 0

$$(PCL) \longleftarrow <B_2>$$
$$(PCH) \longleftarrow <B_3>$$

otherwise

$$(PC) \longleftarrow (PC) + 3$$

where (PC) on right hand side is having the address of the opcode.

It requires 2 or 3 machine cycle of 7 or 10 states depending on whether the condition is TRUE or not. The addressing mode is immediate mode. No flag is affected.

9. JX5: (Jump on X5) This is also a 3-byte conditional jump instruction. The operation code format is

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

This instruction tests the X_5 bit. If the X_5 bit is '1', the control is transferred to the instruction whose address is specified in 2nd byte & 3rd byte of the instruction, otherwise control continues sequentially. The macro RTL implemented is shown below:

If X_5 flag = 1

(PCL) ⟵ $<B_2>$

(PCH) ⟵ $<B_3>$

otherwise

(PC) ⟵ (PC) + 3

where (PC) on right hand side is having the address of the opcode.

It requires 2 or 3 machine cycle of 7 or 10 states depending on whether the condition is TRUE or not. The addressing mode is immediate mode. No flag is affected.

10. LHLX: (Load (H,L) register pair indirect through (D,E) register pair) This is a single byte instruction. The operation code is ED_H. The meaning of the instruction is "Load (L) register with the content of the memory location whose address is in (D,E) register pair and load (H)

register with the content of the memory location whose address is one more the contents of (D,E) register". The macro RTL implemented is

$$(L) \leftarrow M[(D, E)]$$

$$(H) \leftarrow M[(D, E) + 1]$$

The instruction requires 3 machine cycles OFMC & two MRMC and 10 states. The addressing mode is register indirect addressing mode. None of the flags are affected in this instruction.