Unit-01

Architecture of 8085 microprocessor

Introduction to microprocessor :-

→ microprocessor एक programable integreated device होते है जिसमे किसी computer के CPU की तरह computing और decision making capabilities होती है।

→ microprocessor Binary number 0 और 1 में communicate और operate करता है जिन्हे इन binary number को bits कहा जाता है।

→ microprocessor के पास binary pattern के form में एक fixed setup of instruction होता है जिसे machine language कहते है human binary language को नही समझ सकता है इसलिए इस binary instruction को एक special name दिया जाता है। जिसे mnemonics कहते है जो कि microprocessor के assembly language perform करता है।

Definition of microprocessor :-

microprocessor को processor, central processing unit (CPU) 'chip' के नाम से भी जाना जाता है। यह computer का मस्तिष्क है और computer मे होने वाला सारा का सारा काम उस पर निर्भर है।
दो number को add करना हो या music सुनना हो या email type करना हो छोटे से छोटे और बड़े बड़ा काम processor के माध्यम से ही पूरा होता है।

computer को दिया जाना वाला हर निर्देश microprocessor के पास पहुचता है और वह data processing करता है। Processing का अर्थ है निर्देशो (instructions) को execute करना और result show करना।
सिर्फ computer ही नही calculator से लेकर watch और television तक हर electronic device मे किसी न किसी प्रकार का microprocessor लगा होता है।

Q. 8085 के architecture के block diagram को उदाहरण सहित समझाइए।

यह diagram microprocessor 8085 के Internal architecture को show करता है। इसके architecture को five main fonctional block मे devide किया जा सकता है।

8085 microprocessor एक 8 bit microprocessor है जिसको NMOS technique का use करके 1977 मे intel ने विकसित किया था यह computer का CPU (central Processing unit) होता है।

8085 microprocessor का use washing machine, microwave oven तथा mobile phone आदि मे किया जाता है।

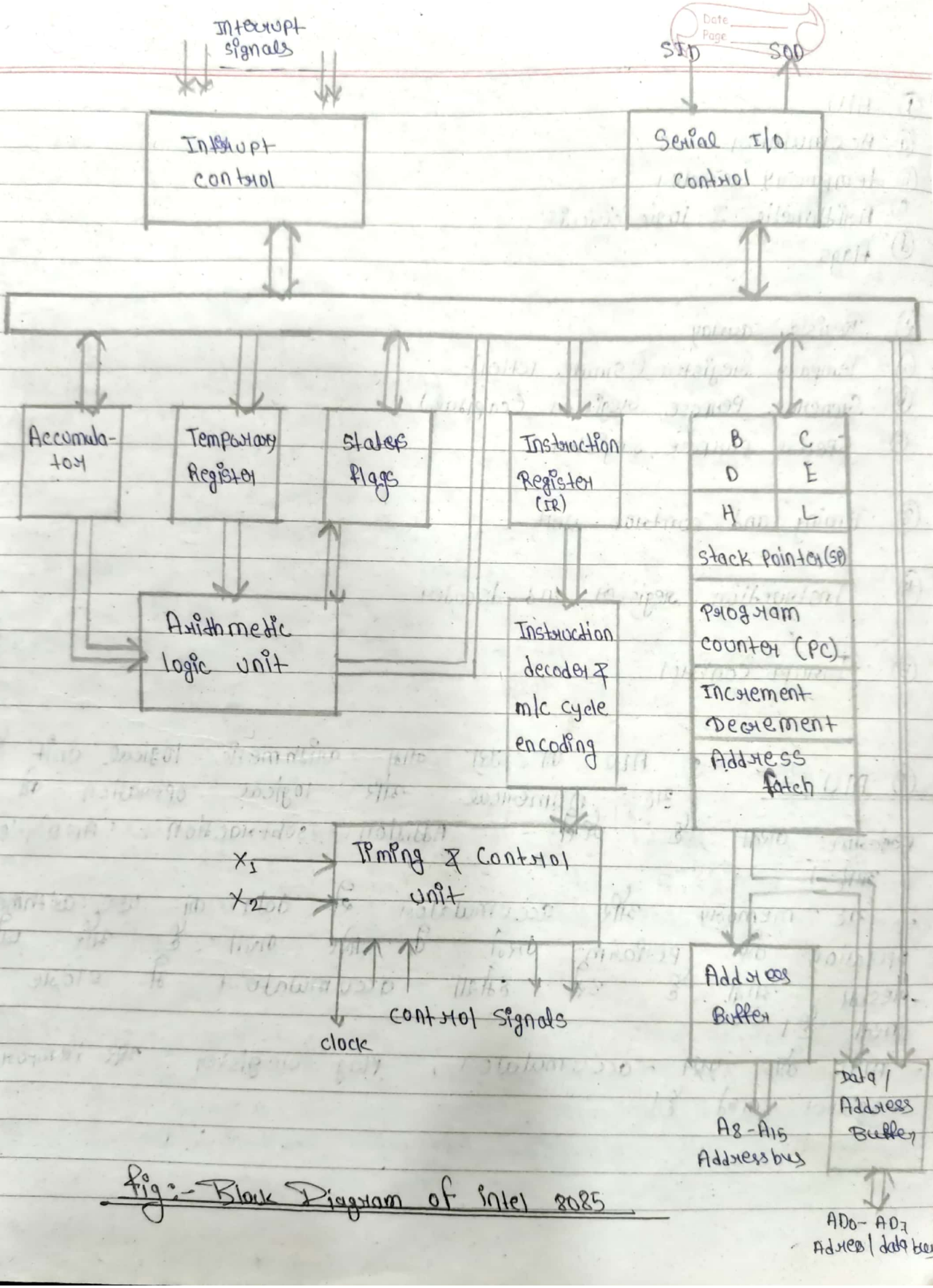


Fig:- Block Diagram of Intel 8085

① ALU
   ⓐ Accumulator
   ⓑ temporary register
   ⓒ Arithmetic & logic circuit
   ⓓ flags

② Register array
   ⓐ Temporary register (small letter)
   ⓑ General purpose register (capital)
   ⓒ Special purpose register.

③ Timing and control unit

④ Instruction register and decoder

⑤ Interrupt control

① ALU –

- ALU का पूरा नाम arithmetic logical unit है यह numerical और logical operation को perform करता है जैसे - Addition, subtraction, 'AND', 'OR' आदि।
- यह memory और accumulator से data का use arithmetic operation को perform करने के लिए करता है और जो result आता है उसे हमेशा accumulator में store करता है।
- ALU को ~~प्रय~~ - accumulator, flag register और temporary register कहते है।

Date
Page

(a) Accumulator - यह एक 8 bit programmable register होता है जिसमें arithmetic और logic operation का result store होता है microprocessor में इसे 'A' से denote किया जाता है। arithmetic और logical operation के दौरान किसी एक operand को hold करने के लिए किया जाता है। arithmetic और logic operation तब तक नही होता है (जब तक दो operand मे से एक operand accumulator मे न हो) Arithmetic logic operation perform करने के लिए दूसरे operand memory या general purpose register मे present हो सकता है। ALU operation के दौरान final result को accumulator मे store किया जाता है।

- accumulator एक 8 bit register होता है जो कि ALU का एक हिस्सा है इस register का use 8 bit data को store करने के लिए तथा arithmetic और logical operation को पूरा करने के लिए किया जाता है।

(b) Temporary Register - यह एक 8 bit non programmable (non accessiable by the user) होता है। जिसका use arithmetic और logic operation के दौरान data को hold करने के लिए किया जाता है। microprocessor 8085 मे दो temporary register w और z होते है।

(c) Arithmetic and logic circuit - यह unit actual arithmetic और logic operation perform करने के लिए ALU को data accumulator memory और general purpose register से प्राप्त होता है।

(d) Flags – 8085 के पास 8 bit flag register होता है इसमें केवल 5 active flags होते है। जो binary number '0' और '1' को store करते है। ALU द्वारा perform किये जाने वाले arithmetic और logic operation के आधार पर flags register perform किये गये operation के states को show करता है ज्यादातर case मे result को accumulator मे store किया जाता है।

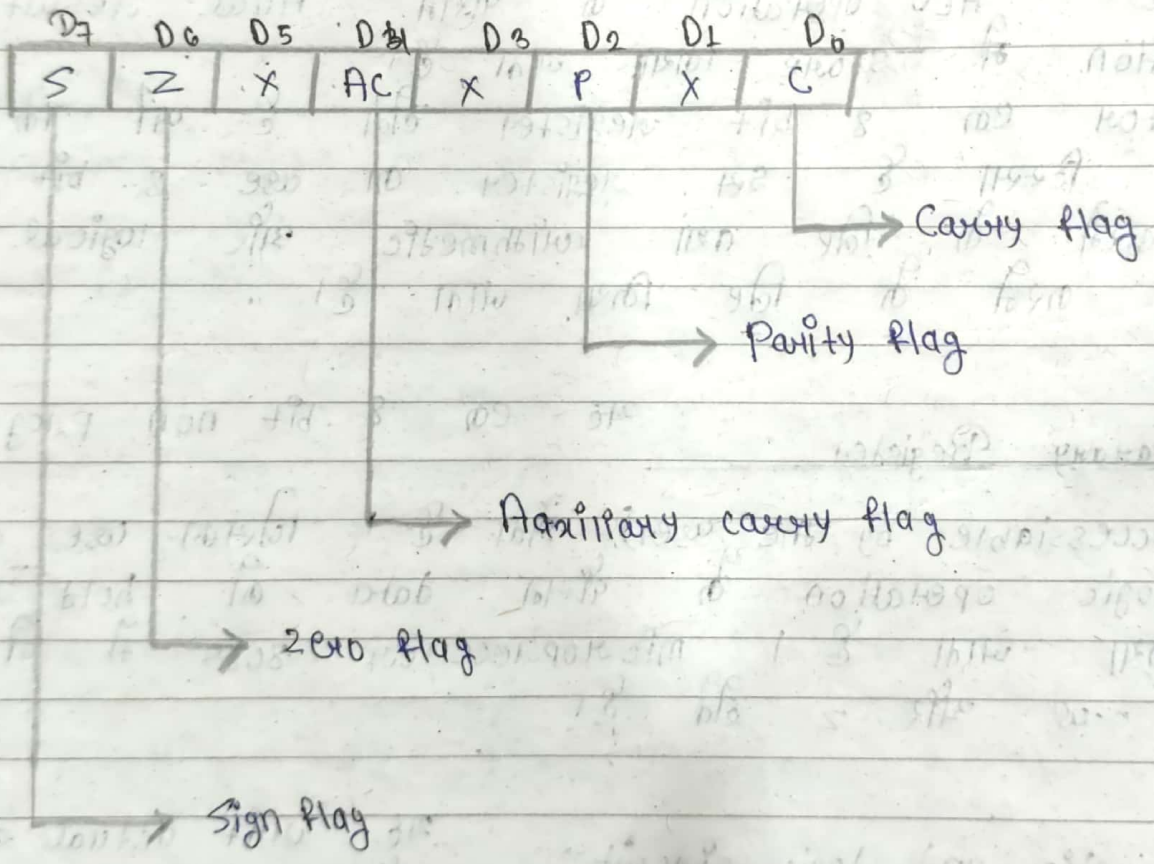


Fig:- 8085 flag register

The description & the condition of the flag are as follows:-

(a) Sign flag (S) - sign flag यह indicate करता है कि mathematical और logical operation का result negative है या positive है।

यदि परिणाम negative होता है तो यह flag set हो जायेगा (अर्थात् S=1) और यदि परिणाम positive होता है तो यह flag reset हो जायेगा (अर्थात् S=0).

(b) Zero flag (Z) - zero flag यह indicate करता है कि mathematical और logical operation का result zero है या नहीं।
यदि वर्तमान operation का result zero है तो flag set हो जायेगा (अर्थात् Z=1) अन्यथा flag reset (Z=0) हो जायेगा।
इस flag को accumulator और अन्य registers में result के द्वारा modify किया जा सकता है।

$$Z = \begin{cases} 1 & , \text{result} = 0 \\ 0 & , \text{result} \neq 0 \end{cases}$$

(c) Auxiliary Carry flag (AC) - एक operation में जब D_3 bit के द्वारा carry को generate किया जाता है और उसे D_4 को भेज दिया जाता है तब AC flag SET होगा अन्यथा RESET होगा।

इसका use केवल BCD operations के लिए ही किया जाता है और इसमें programmer इसके क्रम को jump instruction के द्वारा नहीं बदल सकते है।

(d) Parity Flag (P) - यह flag यह indicate करता है कि वर्तमान result even parity है या odd parity.
यदि even parity होता है तो P set हो जायेगा अन्यथा reset होगा।

$$P = \begin{cases} 1 & \text{If result contains even No. of 1's} \\ 0 & \text{If result contains odd No. of 1's} \end{cases}$$

(e) Carry Flag (C) :- यह flag यह indicate करता है कि जोड़ने या घटाने के दौरान carry या borrow उत्पन्न होता है या नहीं।
यदि carry या borrow उत्पन्न होता है तो यह flag set हो जायेगा अन्यथा reset.
Flag register का D0 bit carry को represent करता है।
~~यदि~~ किसी ALU carry flag का use addition और subtraction के लिए किया जाता है। यदि दो 8 bit number के addition के बाद प्राप्त sum 8 bit से बड़ा होता है तो carry produce होती है और carry flag set होता है।
इसी प्रकार यदि subtraction operation के दौरान borrow occur होता है तो carry flag set हो जाता है।

Note - PSW (Program status word)
16 bit का एक special register होता है यह दो 8 bit registers का combination होता है। Accumulate register इसके upper 8 bit register की तरह work करता है और flag register के dont care. flag को PSW register के लिए 0 (zero) assume किया जाता है।

(2) Register array :-

- (a) Temprary register — ex - W, Z
- General perpose register — ex - B, C, D, E, H, L
- Special Purpose register — ex - SP, PC

(a) Temprary Register - Temprary register W, Z का use data और address को temporary store करने के लिए किया जाता है। microprocessor program को execute करने के दौरान internet operation register का use करता है। इन register को Invisible reg. भी कहा जाता है। क्योंकि ये programming के समय user के लिए available नही है।

(b) General perpose Register - microprocessor 8085 मे 6, 8-bit general Purpose Reg. होते है। क्योंकि B, C, D, E, H & L है। 16 bit data को hold करने के लिए दो 8 bit register combination का use किया जाता है। दो 8-bit register के combination को register pair BC, DE & HL होते है।

(c) Special Purpose Register - microprocessor 8085 मे दो 16 bit के special purpose register होते है। जो कि program counter PC एवं stack pointer (SP) है।

Program Counter (PC) - यह एक 16 bit special Purpose register है जो execute होने

वाले next instruction के memory address को hold करता है।

Microprocessor किसी instruction को execute करने के पश्चात program counter के content को 1 increament कर देता है। जिससे कि program counter program के next instruction के memory address को point करता है।

Note :- जब microprocessor को reset किया जाता है तो program counter (PC) की value 0004 होती है।

Stack Pointer (SP) - Stack pointer एक 16 bit special purpose register होता है। जिसका use stack operation के लिए memory pointer की तरह किया जाता है। Stack pointer stack memory के top location को point करता है।

Stack memory RAM का ही part होता है। जिसे subroutine push और pop operation के दौरान use किया जाता है।

Note:- Stack pointer LIFO (Last In Fast Out) के principle पर work करता है। Stack botton से top की ओर grow करता है और जैसे जैसे stack grow करता है। Stack pointer का content decrese होते जाता है।

(3) Instruction register & Decoder :-

जब किसी instruction को memory से processor में fetch किया जाता है तब उसे instruction register में load किया जाता है। इस register का use केवल fetch किये जाने वाले instruction के opcode को store करने के

लिए होता है। यह programming के लिए available नही होता है।

Instruction register का output instruction decoder से connected होता है। यह decoder instruction register से प्राप्त opcode को decode करता है।

(4) Timing & Control Unit :-

Timing & control unit CPU का वह section होता है जो किसी instruction को execute करने के लिए necessary timing और control signals generate करता है।

किसी instruction के execution के लिए यह system clock से microprocessor के सभी operation को synchronize करता है। और microprocessor & peripheral के बीच communication के लिए जरूरी control signal generate करता है। यह unit computer system के brain की तरह भी कार्य करता है।

(5) Interrupt Control-

यह unit interrupt process को control करता है। main program को execute करने के दौरान जब भी interrupt signal occure होता है या interrupt request की जाती है तो microprocessor program control को main program से shift कर देता है। जिससे कि वह incoming interrupt request को process कर सके। interrupt request process होने के बाद program control को दोबारा main program मे shift कर देता है।

8085 microprocessor में दो type के interrupt occure होते है।

① Hardware interrupt

Rsin	Vector address
RST 4.5 →	0024H
RST 5.5 →	0020H
RST 6.5 →	0034H
RST 7.2 →	0030H
INTR →	non vector

② Software Interrupt

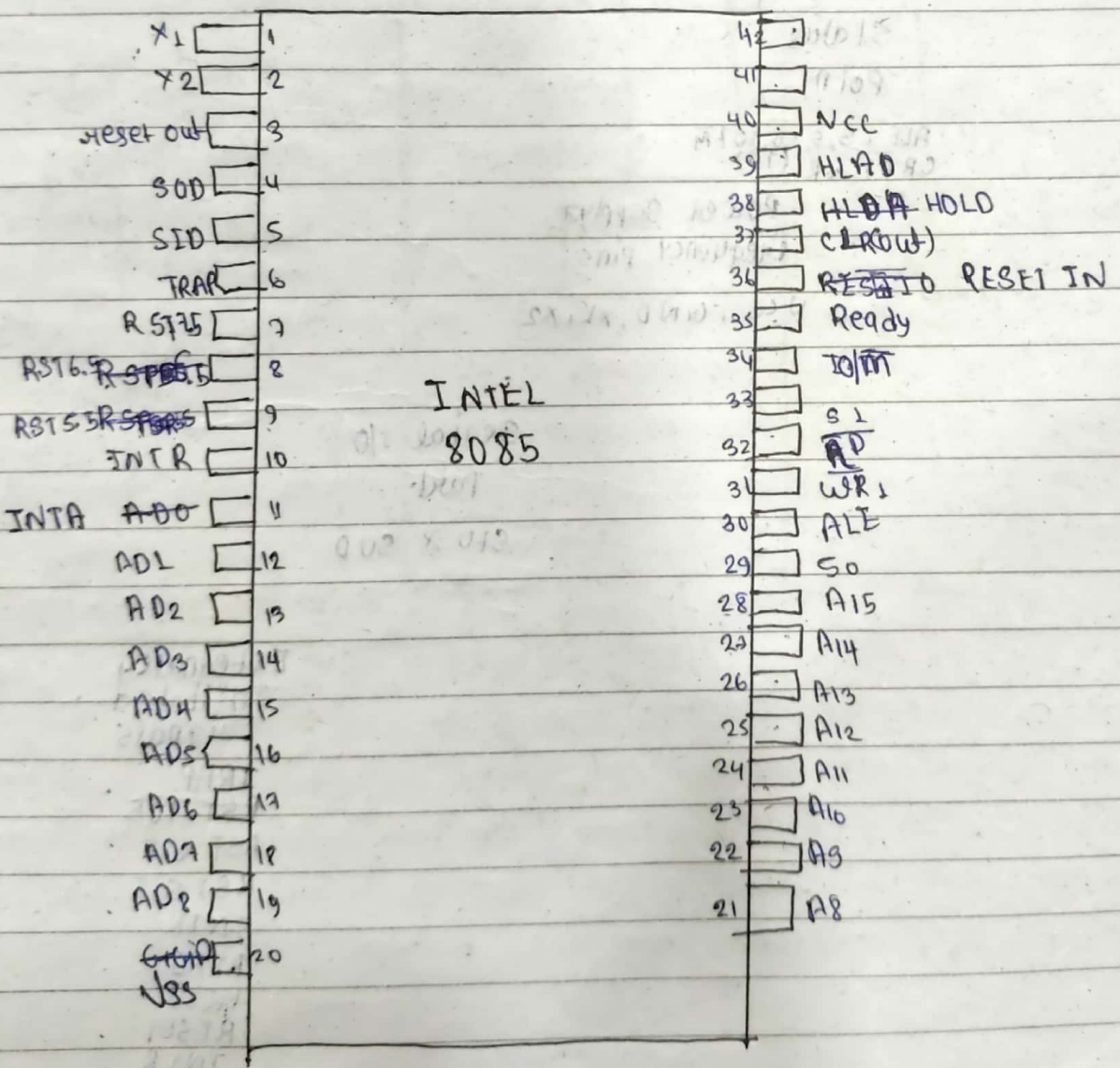
Rsin	vated
RST0 →	0000H
RST1 →	0008H
RST2 →	0010H
RST3 →	0018H
RST4 →	0020H
RST5 →	0028H
RST6 →	0030H

PIN Configuration of microprocessor 8085

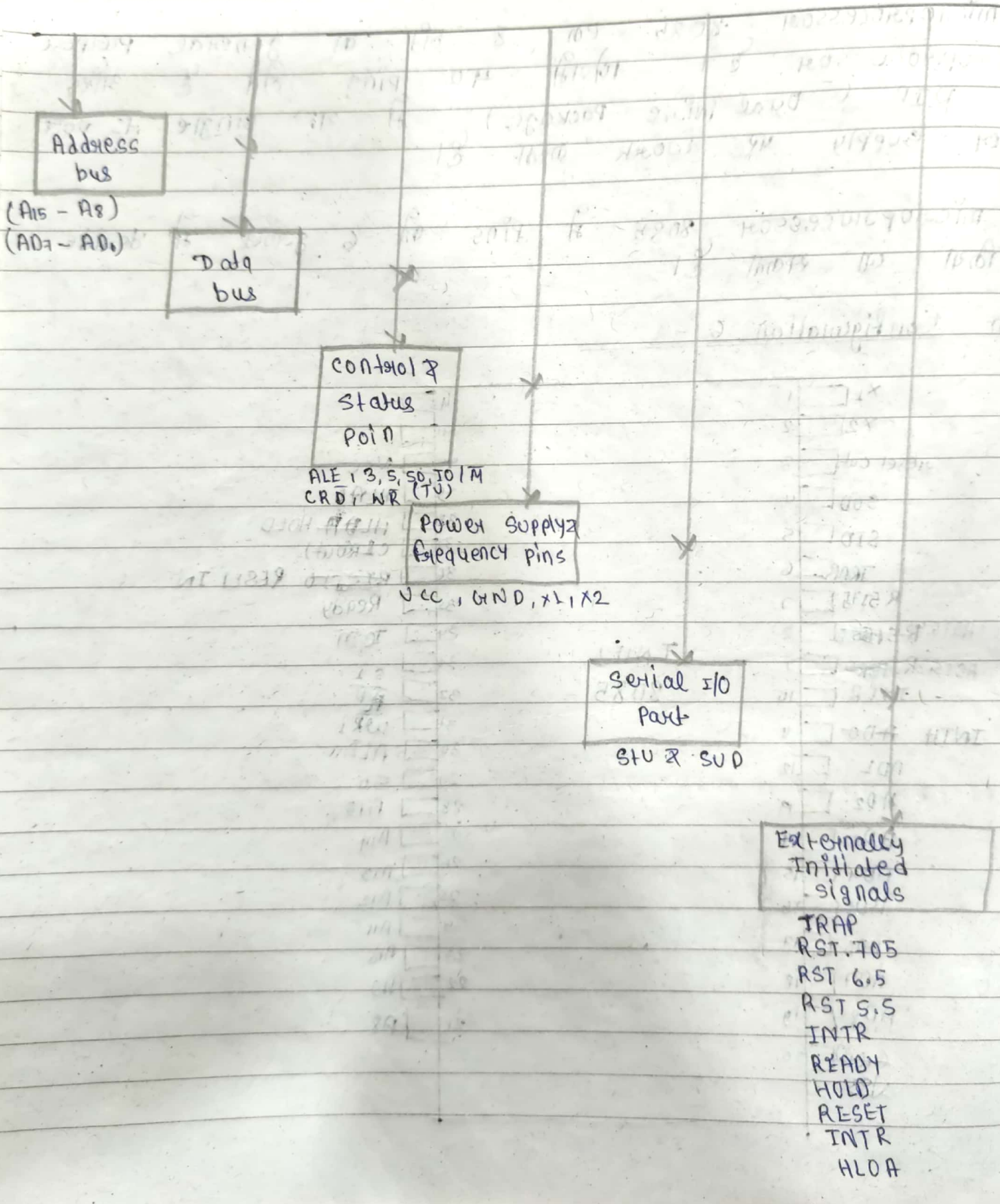
① microprocessor 8085 एक 8 bit का general purpose microprocessor है। जिसमें 40 pins होते है और यह DIP (Dual inline Package) में यह single 5 volt Power Supply पर work करता है।

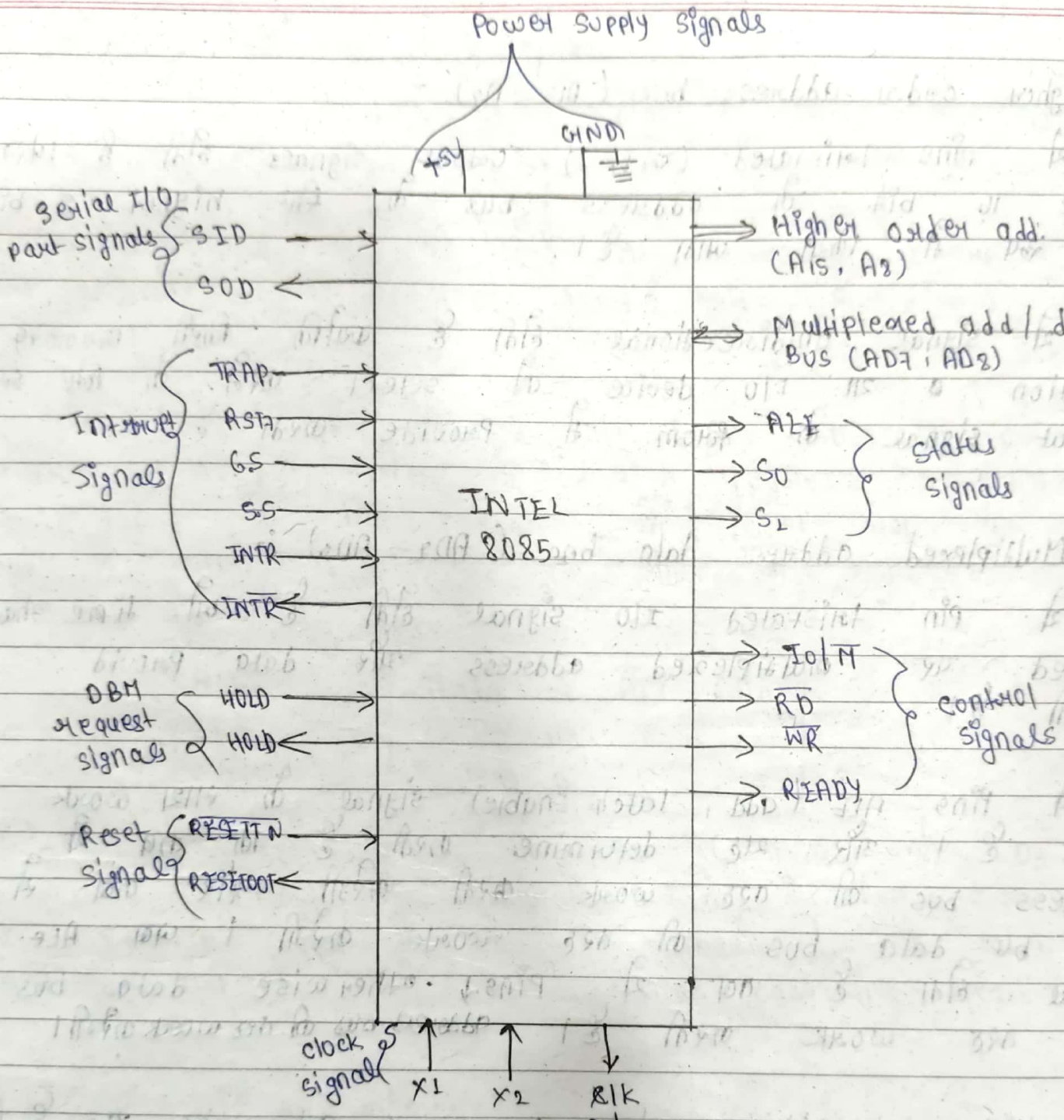
② microprocessor 8085 में pins को 6 group में devide किया जा सकता है।

Pin Configuration 6:-



Pin Configuration





① Higher order address bus ($A_{15} - A_8$) –

(i) ये pins tristated (0, 1, x) output signals होती है जिसका use 16 bit के address bus के लिए higher 8 bit के रूप में किया जाता है।

(ii) ये signal unidirectional होती है क्योंकि किसी memory location या I/O device को select करने के लिए इसे output signal के form में provide करता है।

② Multiplexed address data bus ($AD_7 - AD_0$) :-

(i) ये pin tristated I/O signal होती है जो time sharing based पर multiplexed address और data provide करती है।

(ii) ये pins ALE (add. latch Enable) signal के साथ work करती है। और यह determine करती है कि कब ये address bus की तरह work करेगी और कब से pins data bus की तरह work करेगी। जब ALE signal होता है तब ये pins otherwise data bus की तरह work करती है। ~~address bus की तरह work करेगी।~~

(iii) multiplexed address data bus का advantage यह है कि pins की requirement कम हो जाती है।
इसका disadvantage यह है कि microprocessor को latch के form में extra hardware की जरूरत पड़ जाती है। और microprocessor की speed कम हो जाती है।

(3) ALE - (address latch enable) :-

(i) यह एक output signal होता है जिसका use (AD_7-AD_0) lines को information देने के लिए किया जाता है।

(ii) microprocessor जब भी कोई नया instruction start करता है। तब signal एक positive going pulse के form मे generate होता है।

(iii) जब pulse high होती है तब (AD_7 - AD_0) का content address (A_7-A_0) होता है। ये address इस line पर एक finite के लिए present होती है। जब यह pulse low हो जाती है। तब इन lines मे data (D_7-D_0) present होता है।

ALE	State of AD_7 - AD_0
1	A_7 - A_0
0	D_7 - D_0

(iv) basically इस signal का use (A_7-A_0) और (D_7-D_0) को (AD_7-AD_0) से separate करने के लिए किया जाता है separate करने के लिए इन lines से एक latch conected होता है। जिसे ALE signal control करता है।

(4) IO/M (I/O and memory)

- यह pin यह बताती है कि I/O या memory मे से कौनसा operation perform हो रहा है।
- यदि IO/M = 1, तब I/O operation perform हो रहा है।
- यदि IO/M = 0, तब I/O memory operation perform हो रहा है।

(5) Status Signal (S_1-S_0) :- यह एक output signal होता है जिसका use microprocessor के द्वारा perform किये जाने वाले operation को information देने के लिए किया जाता है।

S_1	S_0	Operation
0	0	Halt
0	1	~~Read~~ write
1	0	~~write~~ Read
1	1	Opcode fetch

(6) $\overline{RD}$ (Read) :-

- $\overline{RD}$ का दूसरा नाम read है
- यह एक active low input signal होता है। इसका use memory या I/O device से data को read करने के लिए किया जाता है।
- यह एक control signal है जिसका use memory और input devices में read operation को पूरा करने के लिए किया जाता है।

(7) $\overline{WR}$:-

- इसका दूसरा नाम write है।
- यह एक active low output signal होता है जिसका use memory या I/O device में data को write करने के लिए किया जाता है।

• यह एक control signal है जिसका use memory और input devices में write operation को perform करने के लिए किया जाता है।

IO/M	S_1	S_0	Operation
0	0	0	memory write
1	1	0	I/O write
0	0	1	memory read
1	0	1	I/O Read
0	1	1	Opcode fatch memory
1	1	1	interupt knowloged
x	0	0	Hold

(8) Ready :-

(i) यह एक active high Interupt control signal है जिसे microprocessor यह detect करने के लिए use करता है कि peripheral ने data transfer completes कर लिया है या नही। जब ready pin high होता है (तब microprocessor next operation के लिए provide करता है)

(ii) जब ready pin low होता है microprocessor ready signal के high होने का wait करता है।

(iii) इस pin का main function यह है कि यह slower peripheral और faster microprocessor को sychonize करता है।

(9) TRAP (RST 4.5) :-

(i) यह एक active high level और edge triggered non maskable highest priority interrupt है।

(ii) जब trap pin active होता है। तब program control trap के vector location 0024H में 8 bit हो जाता है।

(iii) इसकी priority सबसे अधिक होती है।

(10) (RST 6.5 RST 7.5 , RST 5.5) :-

(i) यह active high edge triggered (RST 7.5) और level triggered (RST 6.5 , RST 5.5) maskable interrupt होते है, इनकी priority TRAP > RST 7.5 > RST 6.5 > RST 5.5 होती है।

(ii) जब (RST 7.5, RST 6.5 , RST 5.5) active होता है। तब microprocessor program control को इनके vector location 003CH , 0034H , 002CH respectively में shift कर देता है।

(11) INTR and $\overline{\text{INTA}}$:-

INTR एक active high level triggered general purpose interrupt signal है। जब microprocessor INTR signal receive करता है। तब microprocessor INTR request भेजने वाले device को as an acknowledgement $\overline{\text{INTA}}$ signal sent करता है जो यह indecate करता है कि

microprocessor ने INTR signal को receive कर लिया है।

(12) HOLD and HLDA :-

(i) HOLD एक active high input signal होता है। जिसे दूसरे controls या master microprocessor के address bus, data bus और control bus के use के लिए request signal भेजने के लिए use करते है।

(ii) जब microprocessor hold request signal receive करता है। तब यह current machine cycle को complete करता है और buses के control को give up कर देता है।

(iii) HLDA एक active high output signal है जो यह indicate करता है कि microprocessor ने HOLD signal को recive कर लिया है।

(13) RESET IN :-

(i) यह एक active low input signal है जब RESET IN=0 होता है। तब Program counter की value 0000 हो जाती है और microprocessor की buses tristated हो जाती है।

(ii) Reset होने के बाद microprocessor द्वारा instruction का execution 0004 से start करता है। इसलिए 0004 को microprocessor का reset address भी कहते है।

(14) RESET OUT :-

(i) यह एक active high output signal है जो यह indicate करता है कि microprocessor reset हो चुका है।

(ii) इस signal का use system reset की तरह microprocessor से connect device को reset करने के लिए किया जाता है।

(15) SID (Serial Input data) :-

(i) यह एक active high serial input port pin होता है। जिसका use software control के द्वारा one bit serial data को accept करने के लिये किया जाता है।

(ii) जब एक RIM (Read interrupt mask) instruction execute होता है। तब SID pin का data accumulator के bit D7 मे load हो जाता है।

(16) SOD (Serial output data) :-

यह एक active serial output port pin होता है। जिसका use software control के द्वारा one bit serial data को transfer करने के लिए किया जाता है।

- इस pin मे उपस्थित output को accumulator के 7 वें bit पर store किया जाता है bit को store करने के लिए SIM (set interrupt mask) instruction का use किया जाता है।

(17) X_1 and X_2 :-

- इन दोनो pins को crystal input pins भी कहते है इनका use internal clock generator की frequency को set करने के लिए किया जाता है. यदि हमे किसी system को 8 MHz पर operate करना है तो crystal के पास 6 MHz की frequency होनी चाहिए।
- X_1 और X_2 pins micro के internal clock generator circuit को drive करते है। इस frequency को internally दो से devide की तरह use किया जाता है।

Unit-02

Assembly Language Programming Of 8085

Instruction :⇒

(1) Instruction एक command या group of command होता है। जिसे binary pattern के form मे किसी specific data पर कोई task perform करने के लिये किया जाता है।

(2) Instruction के दो parts होते है पहले part को OPCODE (operational code) और दूसरे part को (Operand code) कहा जाता है।

(3) Opcode यह बताता है कि कौन सा operation perform किया जाना हो और operand यह बताता है कि किस data पर operation perform किया जाना है।

(4) किसी operand को एक Instruction मे कई प्रकार से show किया जा सकता है।
जैसे -

(a) 8 bit / 16 bit General Purpose Register.
(b) memory location.
(c) 8 bit port address.
(d) Implicit operand.

Classification of Instruction set ⇒

(1) Classification on the based of operation.
Introduction to Instruction cycle, machine cycles and T states ⇒
"a program is a group of Instruction written in a sequance."

② Instruction को opcode में convert करके memory में store किया जाता है। उसके बाद microprocessor opcode का execution start करता है।

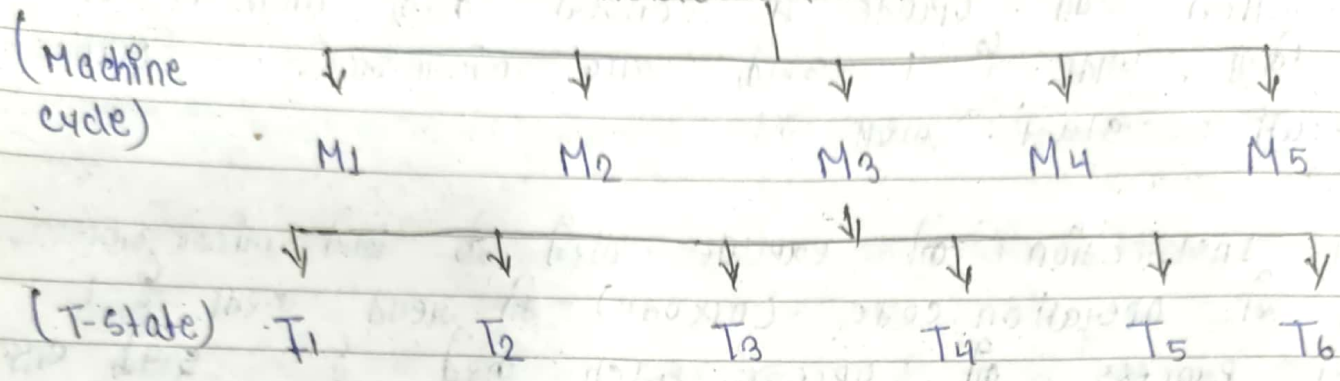
③ किसी Instruction को execute करने के लिये microprocessor memory से operation code (opcode) को read करता है। और इस process को opcode fetch करते है। इसके बाद opcode को Instruction decoder द्वारा decode किया जाता है और decoded opcode के according microprocessor operation perform करता है जिसे execution कहते है।

④ Perform किये जाने वाली सभी operation को microprocessor के द्वारा sequentially system clock के reference में perform किया जाता है। microprocessor किसी operation को clock pulse के term में एक specific time period perform करता है। System clock के एक clock cycles को T states कहा जाता है।

⑤ किसी operation को perform करने के लिए जितने T-states की requirement होती है। उसे machine cycles कहा जाता है।

⑥ किसी Instruction को fetch और execute करने के लिये जितने cycle की requirement होती है। उसे machine cycle कहा जाता है।

Instruction Cycle (IC)



Note :- किसी एक Instruction cycles में maximum 5 machine cycle होते है। एक machine cycle में 3 से 6 T-state होते है।

① machine cycle microprocessor द्वारा perform किए जाने वाले एक complete operation होते है।

Types of machine cycle :-

① OPCODE Fetch
② OPerand Fetch
③ memory Read
④ Memory Write
⑤ I/O Read
⑥ I/O Write
⑦ Interrupt acknowledgement
⑧ Idel machine.

Classification of instruction set

1) Data Transfer Instruction
2) Arithmetic Instruction
3) Logical Instruction
4) Control Instruction
5) Branching Instruction

① Data Transfer Instruction

यह group उन instruction का होता है जो data को source से destination तक copy करता है, बिना source की contents को बदले। विभिन्न type के data transfer possible है जो, निम्नलिखित data register और memory location के बीच किए जा सकते है

The data transfer group of instruction include following instruction.

MOV, MOVE का संक्षिप्त रूप है,

① MOV Rd, Rs :-

Description - यह instruction data को Rs source register से Rd destination register में copy करता है। Rs और Rd के example सामान्य purpose register है जैसे - A, B, C, D, E, H and L, source register की content में कोई change नहीं किया जाता है।

No. of byte - 1 byte, Operation - Rs → Rd
Addressing mode - Register addressing
Flags - No flag effected

Example - MOV A, B the contents of B reg. are copied to A reg.

A	20		F
B	10		C
D			E
H			L

Before Execution

A	10		F
B	10		C
D			E
H			L

After Execution

② MOV R,M or M,R -

MOV R,M

Description - This instruction copies data from memory M to register R.

यह specific memory M से data को register R में copy करता है। M, HL memory pointer को specify करता है। HL register pair की content को address के रूप में use किया जाता है और उस memory location की content को specific R यानी register में transfer किया जाता है। R के example सभी general purpose register वाले register है जैसे - A, B, C, D, E, H, L.

No. of bytes - 1 byte, Operation - M → R or (HL) → R

Flags - No. flag effected

Example - MOV B, M

मानाकि HL pair की content C000 H, B reg = 20H, at add C000 H 10 H store है और instruction MOV B, M is executed.

Addressing mode - Indirect Addressing

					Add.	Data
					BFFF	
A			F			
B	20		C			
D			E	C000		10
H	C0	00	L	C000		

Before execution

					Add.	Data
					BFFF	
A			F			
B	10		C			
D			E	C000		10
H	C0	00	L	C000		

After execution

MOV M, R :

Description - यह instruction R register से data को memory M में copy करता है। HL register pair का use memory pointer के रूप में किया जाता है। register pair के content को memory address के रूप में use किया जाता है और specific register के content को memory location में transfer किया जाता है।

No. of byte - 1 byte, Operation - R → m or R → (HL)
Flags - No. flag effected
Addressing mode - Indirect Addressing
Example - MOV M, C.

मानलि HL pair के content है - C200. C reg = 20H at address C200: 10 store है और instruction MOV M, C executed है।

					Add.	Data
A			F			
B		20	C			
D			E			
H	C2	00	L	C200		10
				C201		

Before Execution

					Add.	
A			F			
B		20	C			
D			E	C1FF		
H	C2	00	L	C200		20
				C201		

After Execution

(3) MVI R, data - Move immediate data to specified reg.

Description - 8 bit data को instruction के साथ specified किया जाता है। यह instruction immediate 8 bit data को register में ले जाता है। यह दो byte instruction है इसकि instruction का पहला byte OPCODE होगा और दूसरा 8 bit data होगा। R के example सभी general purpose register है - A, B, C, D, E, H & L.

No. of byte - 2 bytes, Operation - Data → R
Flags - No flag effected
Addressing mode - Immediate addressing
Example - MVI C, 05H The data, 05 will be loaded in reg. C

A			F	A			F
B		20	C	B		05	C
D			E	D			E
H			L	H			L

Before execution | After execution

(4) MVI M, data -

Description - यह instruction तत्काल data को memory में ले जाता है। HL register pair का use memory pointer के रूप में किया जाता है। इन register pair की content को memory address के रूप में किया जाता है और तत्काल data को उस memory location पर transfer किया जाता है।

No. of byte - 2 byte, Operation - Data → M or Data - (HL)
Flags - No flag effected
Addressing mode - immediate indirect addressing.

Example - MVI H, 20H : 20 data is transfered to H reg.
MVI L, 00H : 00 data is transfered to L reg.
MVI M, 10H : 10 data is transfered to memory.

When MVI M instruction is executed, the data 10 H will be stored in memory location addressed by HL pair ie 2000

				Add.	data
				FFFF	
A			F		
B			C		
D			E	2000	
H	20	00	L	2001	

Before Execution

				Add	data
				FFFF	
A			F		
B			C		
D			E	2000	10
H	20	00	L	2001	

After execution

(5) LXI Rp, data 16 Bit -

Description - रजिस्टर पेअर को 16 bit data से load करे -

यह instruction के साथ specific 16 bit data को R रजिस्टर पेअर में load करता है। instruction में रजिस्टर पेअर के लिए केवल high order रजिस्टर specify किया गया है यानी अगर HL पेअर को load किया जाना है तो instruction में केवल H रजिस्टर specify किया जाएगा। R. के ex. BC पेअर, DE पेअर, HL पेअर व stack pointer SP है।

No. of byte - 3 byte, Operation - 16 bit data → Rp
Flag - No flag effected
Addressing mode - Immediate Addressing
Example - (i) LXI H, 2000H : load HL pair with data 2000H. 20 will be loaded in H reg. and 00 H in L.

A			F
B			C
D			E
H	FF	FF	L

Before execution

A			F
B			C
D			E
H	00	00	L

After execution

(ii) LXI SP, C600 : Load stack pointer with data C600H

A			F
B			C
D			E
SP	0000		

Before execution

A			E
B			C
D			E
SP	0000		

After execution

⑥ LDA Address - Load accumulator direct from memory

Description - यह instruction memory location की content को accumulator में copy करता है। instruction के साथ memory location का address specify किया जाता है। यह एक 3 byte instruction है, इसलिए इसे LXI R, data instruction यानी OPCODE के समान memory में पहले byte lower order के address के रूप में store किया जाता है, दूसरे byte के रूप में और highest order address के तीसरे byte के रूप में।

No. of byte - 3 byte, Operation - (Address) → A
Flag - No flag effected
Addressing mode - Direct addressing
Example - LDA C100 A : Load accumulator with the contents of memory load C100.

				Add.	Data
				C0FF	
A	50		F		
B			C		
D			E		
H			L	C100	
				C101	

Before Execution

				Add.	Data
				C0FF	
A	20		F		
B			C		
D			E		
H			L	C100	20
				C101	

After Execution.

(7) STA Address - Store accumulator direct to memory

Description - यह instruction accumulator की content को memory location पर copy करता है। memory location का address instruction के साथ specified किया गया है। यह एक 3 byte instruction है इसलिए इसका storing format OFD के रूप में पहला byte, lower order address दूसरे byte के रूप में high order address तीसरे byte के रूप में।

No. of byte - 3 byte Operation - A → (Address)

Flags - No flag effected

Addressing mode - Direct addressing

Example - STA C200 H : Store accumulator contents of memory location C200.

				add.	Data
A	20		F		
B			C		
D			E		
H			L	C200	10
				C201	

Before Execution

				add.	Data
A	20		F		
B			C		
D			E		
H			L	C200	20
				C201	

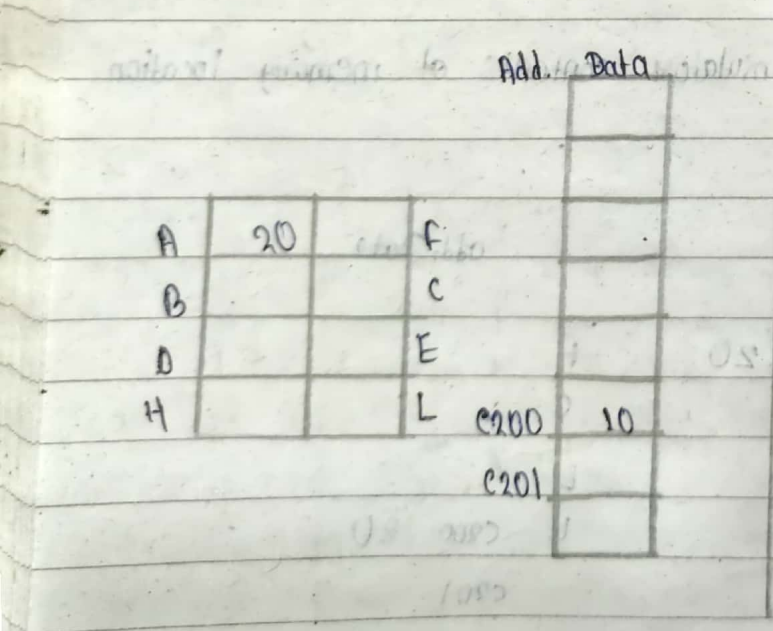
After Execution

(8) LHLD Address - Load HL pair direct from memory

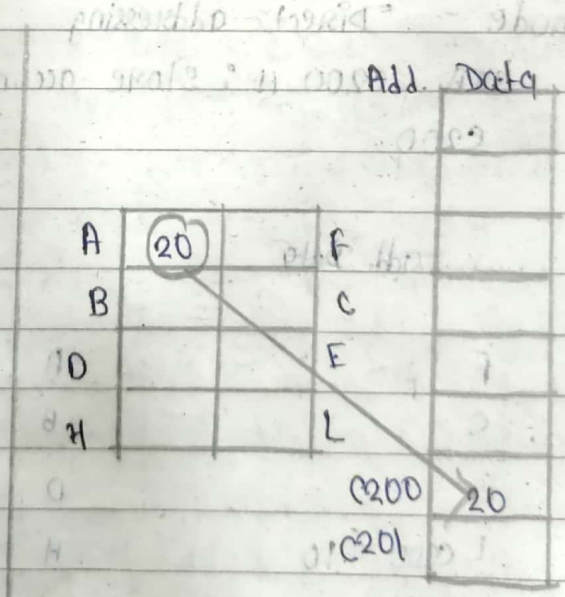
Description - यह instruction memory location की content को handle register में copy करता है। instruction के साथ memory location का address specify किया जाता है। memory location की content जिसका address instruction के साथ specify किया जाता है। उसे L register में और content को H register में transfer किया जाता है। इस instruction का use memory से H और L register को load करने के लिए किया जाता है। यह 3 byte instruction है, इसलिए storing format पहले byte के रूप में opcode, दूसरे byte के रूप में lower order address और तीसरे byte के रूप में high order address है।

No. of byte - 3 byte, Operation - Address → L. reg.
Flag - No. flag effected Address → H. reg.
Addressing mode - Direct Addressing

Ex - LHLD C200 ; Load HL pair from memory location C200 & C201. Suppose H = 05H, L = 04H at memory location C200 & C201 the data 20, 30 is stored respectively & instruction LHLD C200 is executed.



Before Execution



After Execution

⑨ SHLD Address – Store HL pair direct in memory

Description – यह instruction register H, L की content को memory location पर copy करता है। instruction के साथ memory location का address specify किया जाता है। L register की content उस memory location पर store की जाती है जिसका address specify किया गया है और H register की content (Add. +1) location पर। इस instruction का use H and L register को सीधे memory में store करने के लिए किया जाता है। Store format OPCODE को पहले byte के रूप में और high order को तीसरे byte के रूप में store किया जाता है।

No. of byte – 3 byte
Flags – No flag effected
Addressing mode – Direct addressing
Operation – (Address) ← L reg.
(Address) ← H reg.
Example – SHLD C300 ; Store HL pair to memory location C300 & C301.

Suppose H = 05H, L = 04H, at memory location C300 & C301 data 20 and 30 is stored respectively and instruction SHLD C301.

A			F	
B			C	
D			E	
H	05	04	L	

Add	Data
C2FF	
C300	20
C301	30

Before execution

A			F	
B			C	
D			E	
H	05	04	L	

Add	Data
C2FF	
C300	04
3301	05

After execution

(10) LDAX Rp – Store accumulator Indirected by using a memory Pointer.

Description – यह instruction memory location accumulator की content को copy करता है। memory location का address instruction के साथ specific R pair द्वारा दिया जाता है। RP का example – B (BC pair) ₹ D (DE Pair) है।

No. of byte – 1 byte , operation – (Rp) → A
Flag – No flag effected
Addressing Mode – indirect addressing
Example – LDAXB : load accumulator with the contents of memory location with add is given by BC register pair.
suppose A=00, B=20, C=02 at memory location, 2002 : 50 stored and instruction LDAXB is executed.

				Add.	Data
				..	
A	00		F		
B	20	02	C		
D			E	2001	
H			L	2002	50
				2003	

Before execution

				Add.	Data
				..	
A	50		F		
B	20	02	C		
D			E	2001	
H			L	2002	50
				2003	

After execution

(11) STAX Rp – Store accumulator indirected by using a memory pointer.

Description – यह instruction accumulator की content को memory location पर copy करता है। memory location का address instruction के साथ Rp pair ₹ instruction पर के साथ किया जाता है उदा. के लिए Rp B (BC Pair) ₹ D (BE Pair)

No. of byte – 1 byte , Operation - A → (RP)
Flags – No flag effected
Addressing mode - register indirect addressing.
Example - STAX D : store accumulator to the memory location, whose address is given by DE register pair.
Suppose A = 20 , D = 25 , E = 05 , at memory location, 2505 : 10 is stored and instruction STAX D is executed.

					Add.	Data
A	20		F			
B			C			
D	25	05	E	2504		
E			L	2505	10	
				2506		

Before execution

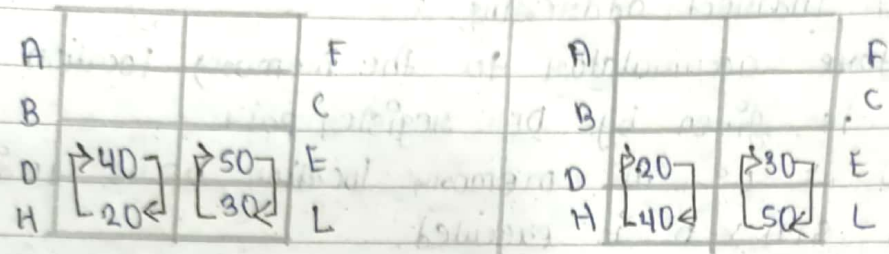
					Add.	Data
A	20		F			
B			C			
D	25	05	E	2504		
E			L	2505	20	
				2506		

After execution.

(12) XCHG - Exchange the contents of HL with DE pair.
Description - यह instruction H reg. की सामग्री को D reg. से तथा L reg. की content को E reg. से बदलता है।

No. of byte – 1 byte , Operation - H ⟷ D , L ⟷ E
Flag – No. flag effected
Addressing mode - register addressing
Example - XCHG

Suppose H = 20H , L = 30H , D = 40H , E = 50H and instruction on XCHG is executed.



Before execution After execution

(13) IN - 8 bit address -

Description - यह instruction किसी specific 8-bit address वाले I/O port से data को input करके accumulator में load करता है।

Operation - (A) ← (I/O port) (accumulator में I/O port का data load होता है)

No. of byte - 2 byte

Addressing mode - Direct addressing mode

Flag - No flag affected

(4) OUT 8-bit address

Description - यह instruction accumulator की content को किसी specific - 8 bit address वाले I/O port पर भेजता है।

Operation - (I/O port) ← (A)

No. of byte - 2 byte

Addressing mode - Direct addressing mode

Flags - No flag affected

Arithmetic Instruction

(1) ADD R -

Add register R content to accumulator

Description - यह instruction register R और accumulator के contents को add करता है और result को accumulator में store करता है। R के example सभी general purpose registers जैसे की A, B, C, D, E, H & L। Accumulator में result के अलावा सारे Flag भी modify होते हैं ताकि operation का result reflect हो सके।

Operation - A+R → A
No. of byte - 1 byte
Addressing mode - Register addressing
Flag - All flag affected
Example - ADD B : A+B → A
Suppose A=47 H, B= 51H and instruction ADD B is executed.

A	47		F
B	51		C
D			E
H			L

Before Execution

A	98		F
B	51		C
D			E
H			L

After Execution

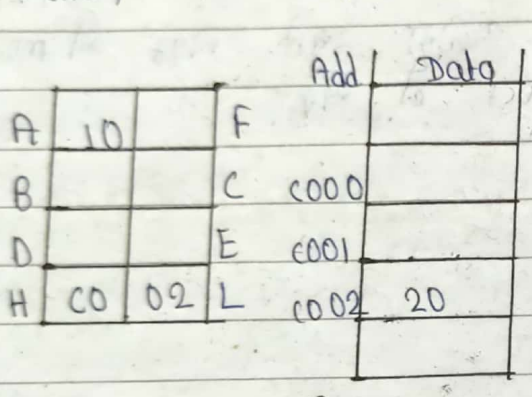
(2) ADD M -

memory location के content को accumulator में add करना

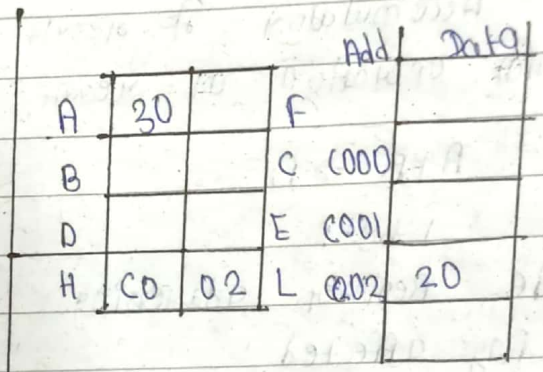
Description - ये instruction HL pair को memory pointer के रूप में use करता है। जो memory location HL pair द्वारा point की जा रही है, उसके contents को accumulator के साथ add किया जाता है और result accumulator में store होता है। सारे Flag भी accordingly modify है ताकि operation का result reflect हो सके।

Operation – A+M → A or A+(HL) → A
No. of byte – 1 byte
Addressing mode – register indirect addressing
Flag – All flag affected
Example – ADD M : A+(HL) → A

Suppose A = 10 , H = C0 , L = 02 , at memory location C002 : 20 data
A+(HL) → A : 10+20 → 30



Before execution



After execution

③ ADC R – register R और carry flag के contents को accumulator में store करना।

Description – यह instruction, Register R के contents, carry flag (CY) और accumulator के content को add करता है और उस result को accumulator में store करता है। Register R के ex. – A, B, C, D, E, H और L (ये सब general purpose register होते है)। accumulator में result store होने के साथ-साथ, सभी flag को भी update किया जाता है ताकि operation का result reflect हो सके।

Operation – A+R+CY → A
No. of byte – 1 byte
Addressing mode – Register ~~indirect~~ addressing
Flags – All flag affected
Example – ~~ADD M : A+(HL) → A~~ ADC B : A+B+CY → A

Suppose A = 10, H = C0, L = 02, at memory location C002 : 20 data is stored and ADD M instruction is executed

A + (HL) → A : 10 + 20 → 30

Suppose B = 20, A = 3F, CY = Set and ADC B instruction is executed

So CY = Reset, Z = Reset, P = Set, AC = Set, S = Reset

A	3F		F
B	20		C
D			E
H			L

Before Execution

A	60		F
B	20		C
D			E
H			L

After Execution

(4) ADC M :- Add memory location and carry flag contents to accumulator.

Description - यह instruction HL pair को memory pointer की तरह use करता है। memory location, जो HL pair द्वारा point की गई है, उसकी contents और carry flag को accumulator के साथ add किया जाता है। result को accumulator में store किया जाता है। इस साथ ही, सारे flag को modify किया जाता है ताकि operation का result reflect हो सके।

Operation - (HL) + CY + A → A or M + CY + A → A

No. of byte - 1 byte

Addressing mode - Indirect addressing

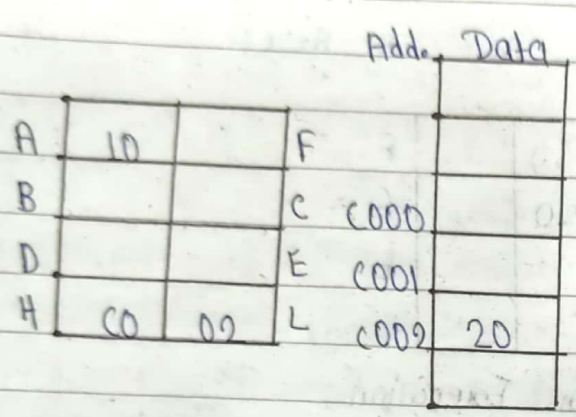
Flags - All flag affected

Example - ADC M : A + (HL) + CY → A

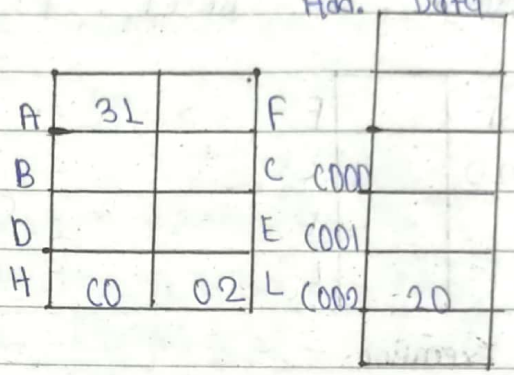
Suppose A = 10, H = C0, L = 02, CY = Set, at memory location C002 : 20 data is stored and ADC M instruction is executed.

```
A    0 0 0 1 0 0 0 0
M    0 0 1 0 0 0 0 0
CY                 1
     0 0 1 1 0 0 0 1
```

Result in A = 31



Before execution



After execution

(5) **ADI Data** – Add immediate data to accumulator

Description – ये instruction 8-bit का data, जो instruction के साथ specify किया गया है, उसे accumulator में add करता है और result accumulator में store होता है। सारे flags को भी modify किया जाता है ताकि operation का result reflect हो सके। इस instruction का storing format होता है - पहला byte होता है OPCODE, और दूसरा byte होता है operand (data)।

Operation – A + data → A
No. of byte – 2 byte
Addressing mode – Immediate addressing
Flag – All flag affected
Example – ADI B7 H : Add B7 H data to accumulator and store result in accumulator

Suppose A = 59 H and instruction ADI B7 is executed.

A = 0 1 0 1 1 0 0 1

Data = 1 0 1 1 0 1 1 1

1 0 0 0 1 0 0 0

The result in A = 10

A	59		F
B			C
D			E
H			L

Before execution

A	10		F
B			C
D			E
H			L

After execution

⑥ ACI Data - Add immediate data and carry flag to accumulator

Description - यह instruction immediate data, carry flag और accumulator को add करता है और result को accumulator में store करता है। सारे flag को भी modify किया जाता है ताकि operation का result reflect हो सके। इस instruction का storing format होता है: पहले byte OPCODE और दूसरा byte operand (data)।

Operation - A + data + CY → A

No. of byte - 2 byte

Addressing mode - Immediate addressing

Flags - All flag affected

Example - ACI 20H : add 20 H data and carry flags to accumulator and result is placed in accumulator.

Suppose A = C0, CY = Reset and instruction ACI 20 is executed.

A	C0		F
B			C
D			E
H			L

Before execution

A	E0		F
B			C
D			E
H			L

After execution

(7) SUB R - Subtract register from accumulator

ये instruction register के content को accumulator से subtract करता है और result accumulator में ही store होता है। R register के content change नही होते सारे flags को accordingly modify किया जाता है। R के ex - A, B, C, D, E, H and L है। subtraction perform होती है 2's complement method से ये 2's complement microprocessor खुद generate करता है, programmer को manually कुछ करने की जरूरत नही होती।

Operation — A - R → A
No. of byte — 1 byte
Addressing mode — Register Addressing
Example - SUB B: A-B → A

1. Suppose the content of A = 37 H and B = 40H and instruction SUB is executed

B = 0 1 0 0 0 0 0 0
2's complement of reg. = 1 1 0 0 0 0 0 0

A = 0 0 1 1 0 1 1 1
2's complement of reg. B = + 1 1 0 0 0 0 0 0
1 1 1 1 0 1 1 1

The flags status will be as follow
Z = reset, P = Reset, AC = Reset, S = set and CY set

A	37		F
B	40		C
D			E
H			L

A	F7		F
B	40		C
D			E
H			L

2. Suppose the content of A = 40 H and B = 37 H and instruction SUB B is executed

B = 0 0 1 1 0 1 1 1

2's complement of reg. B = 1 1 0 0 1 0 0 1

A = 0 1 0 0 0 0 0 0

2's complement of reg. B = 1 1 0 0 1 0 0 1

1 + 0 0 0 0 1 0 0 1

The status of flags will be as follow

Z = Reset , P = Set , AC = Reset , S = Reset and CY = Reset

A	40		F
B	37		C
D			E
H			L

A	09		F
B	37		C
D			E
H			L

(8) SUB M :- Subtract memory location content from accumulator

Description - यह instruction memory location के contents (जिसका address HL register के through दिया गया है) को accumulator से subtract करता है और result फिर से accumulator में store होता है। subtraction बिल्कुल वैसे ही perform होता है जैसे SUB R instruction में होता है। सारे flag modify होते है जबकि operation के result को reflect कर से।

Operation - A - (HL) → A

No. of byte - 1 byte

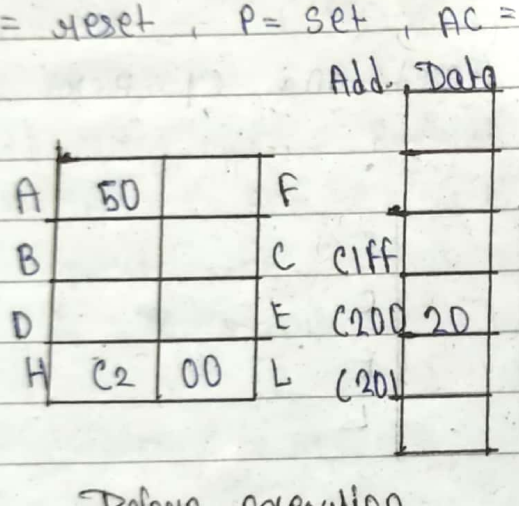
Addressing mode - Indirect addressing

Example - SUB M : A - (HL) → A

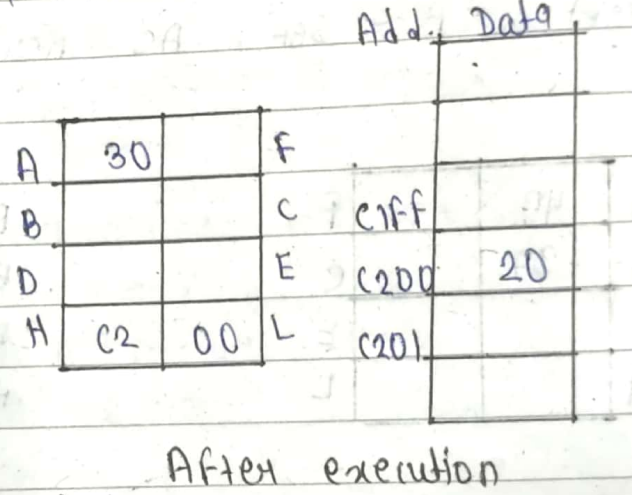
Suppose A = 50 H, H = C2, L = 00 at memory location C200 : 20 is stored and instruction SUB M is executed

Data	=	0 0 1 0	0 0 0 0	
2's complement	=	1 1 1 0	0 0 0 0	
A	=	0 1 0 1	0 0 0 0	
2's complement	=	+ 1 1 1 0	0 0 0 0	
		1	0 0 1 1	0 0 0 0

The status of flag will be as follow :-
Z = reset, P = set, AC = reset, S = reset and CY = reset



Before execution



After execution

(9) SBB A - Subtract register and borrow flag from accumulator.

Description - ये instruction accumulator से register और borrow flag (carry flag) को subtract करता है और result accumulator में store होता है। सभी flag को modify किया जाता है ताकि subtraction के result को reflect किया जा सके। subtraction instruction में carry flag को borrow flag कहा जाता है। R के ex. है सारे general purpose register जैसे A, B, C, D, E, H & L है।

Operation - A - R - Borrow flag → A

No. of byte - 1 byte

Addressing mode - Register Addressing
Flags - All flag affected
Example - SBB B : A - B - CY → A
Suppose A = 37H, B = 3F and carry i.e. Borrow flag is set and SBB instruction is executed

B = 0 0 1 1 1 1 1 1
Borrow = 1
0 1 0 0 0 0 0 0

2's complement = 1 1 0 0 0 0 0 0
+
A = 0 0 1 1 0 1 1 1
A = 1 1 1 1 0 1 1 1

The flag status will be as follow
Borrow = set, AC = Reset, Z = Reset, S = Set, P = Reset

A	37		F
B	3F		C
D			E
H			L

Before execution

A	F7		F
B	3F		C
D			E
H			L

After execution

(10) SBB M - Subtract memory and borrow flag from accumulator

Description - यह instruction memory location के content और borrow flag को accumulator से subtract करता है और result फिर से accumulator में store होता है। memory location का address HL register pair के through दिया जाता है। operation बिल्कुल SBB R instruction जैसे ही होते हैं।

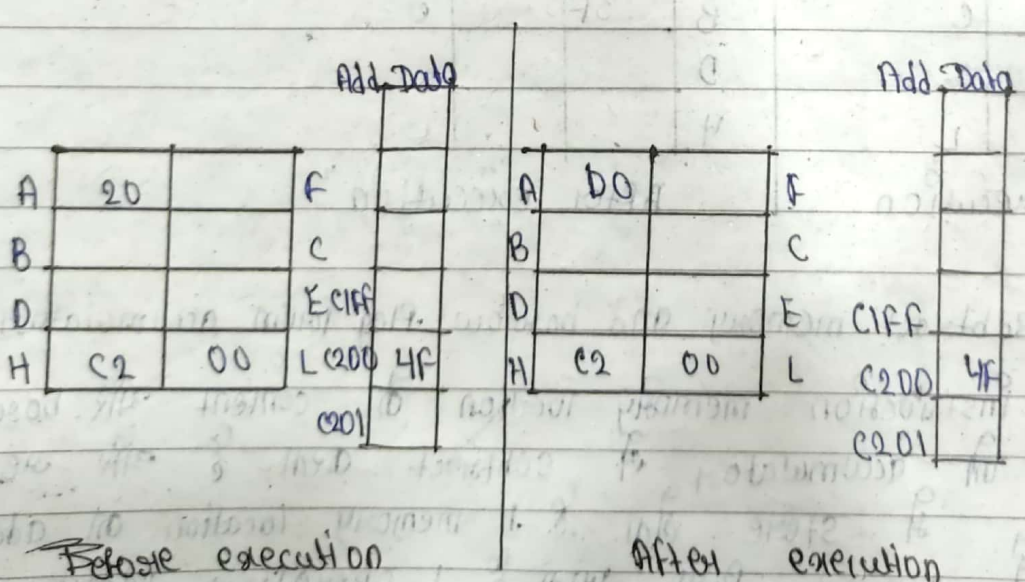
Operation – A - (HL) - Borrow ⇒ A OR A - M - Borrow → A
No. of byte – 1 byte
Addressing mode – Indirect Addressing
Flag – All flag affected
Example – SBB M : A - (HL) - Borrow → A

Suppose A = 20H, H = C2 H, L = 00H, CY = Set, at memory location C200 : 4F H is stored and instruction SBB M is executed
Data = 4F + 1 = 50

Data = 0 1 0 1 0 0 0 0
2's complement = 1 0 1 1 0 0 0 0

A = 0 0 1 0 0 0 0 0
+
2's complement = 1 0 1 1 0 0 0 0
1 1 0 1 0 0 0 0

The status of flag will be follows
Z = reset, P = reset, AC = reset, S = set, CY = set



(11) SUI Data - Subtract immediate data from accumulator

Description - यह instruction accumulator में से वही data minus करता है जो instruction के साथ दिया गया होता है, और result वापस accumulator में store होता है। subtraction 2's complement method का use करके किया जाता है, और इसका operation SUB R instruction जैसे ही होता है।

Operation — A → data → A
No. of byte — 2 byte
Addressing mode - Immediate addressing.
Flags - All flag affected
example - SUI 50 ; A - 50 → A
Suppose A = 20 and instruction SUI 50 is executed

A	20		F
B			C
D			E
H			L

Before execution

A	D0		F
B			C
D			E
H			L

After execution

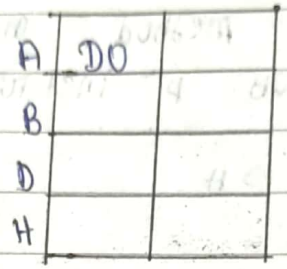
(12) SBI Data - Subtract immediate data and borrow flag from accumulator.

Description - यह instruction accumulator में से data और borrow flag दोनों को minus करता है और result accumulator में store करता है। instruction का operation SUBB R instruction जैसा होता है।

Operation — A - data - borrow → A.
No. of byte — 2 bytes
Addressing mode — Immediate addressing

Flags – All flag affected
Example – SBI 4F : A-4F-CY → A
Suppose A = 20, CY = Set and instruction SBI 4F is executed

· Before execution

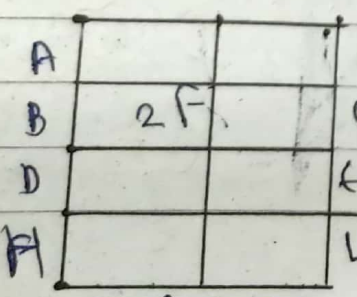


After execution

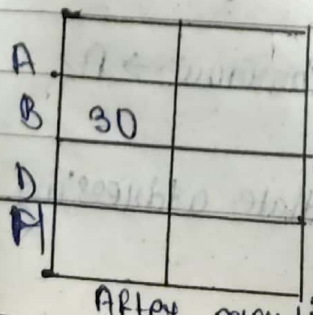
(13) INR R – Increment register contents by one

Description – यह instruction specified register के content को 1 से increment करता है। result उसी register में store होता है। R के ex. है general purpose register जैसे A, B, C, D, E, H व L है।
सिर्फ carry flag modify नही होता बाकी सब flags modify हो जाते है।

Operation – R+1 → R
No. of bytes – 1 byte
Addressing mode – register addressing
Flage – All flag affected
example – INR B : B+1 → B
suppose B = 2F, Flag reg. = 11 X0 X0 X1 and instruction INR B is executed.



Before execution

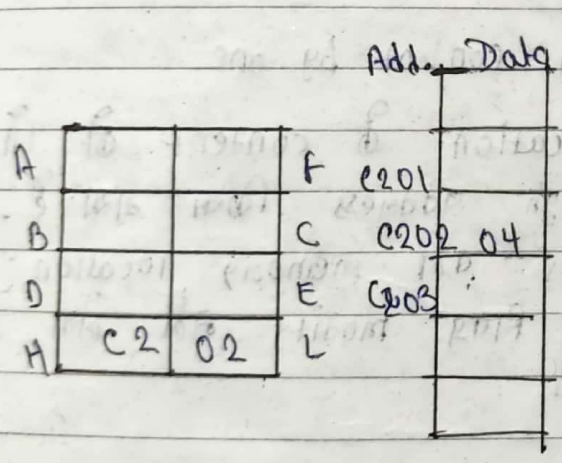


After

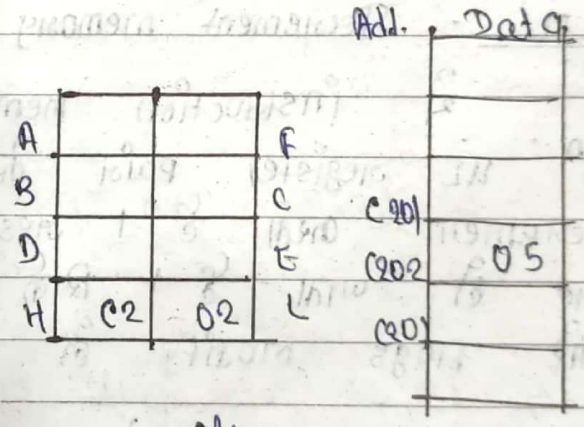
(14) INR M - Increment memory contents by one

Description - यह instruction memory location के content को, जो HL register pair से address होती है, 1 से increment करता है। result उसी memory location में वापस store हो जाता है। सिर्फ carry flag modify नहीं होता बाकि सब flags modify हो जाते है।

Operation - (HL) + 1 → (HL) or M + 1 → M.
No. of byte - 1 byte
Addressing mode - register Indirect addressing
Flag - All flag affected
Example - INR M : (HL) + 1 → (HL)
Suppose H = C2, L = 02, at memory location C202 : 04 is stored, flag reg = 00 x 1 x 0 x 1 and instruction INR M is executed



Before execution

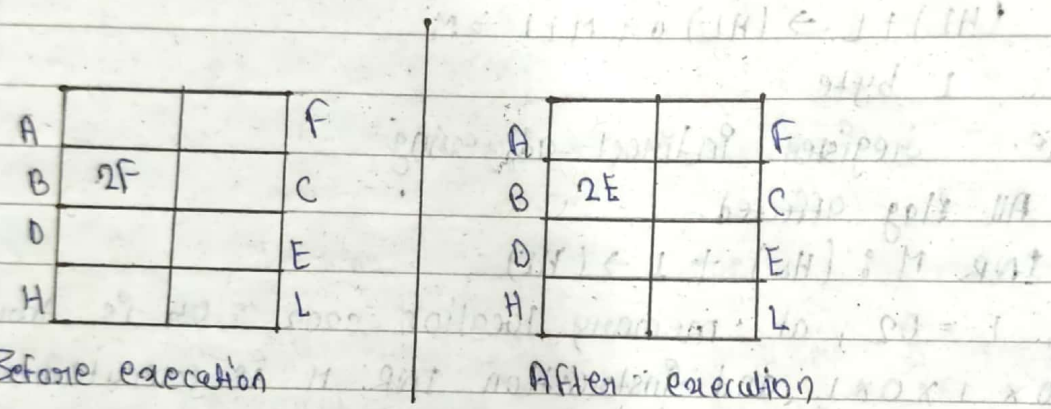


After execution.

(15) DCR R - Decrement register content by one

Description - यह instruction register के content को 1 से decrement करता है और result वही register में store होता है। R के example है general purpose registers जैसे A, B, C, D, E, H और L है। सिर्फ carry flag modify नहीं होता बाकी सब flags modify हो जाते है।

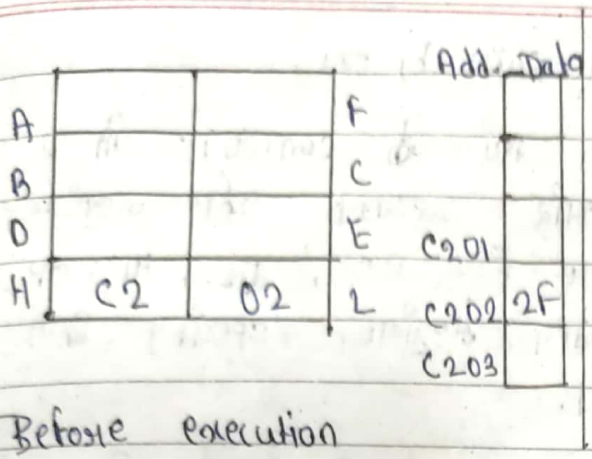
Operation - R-1 → R
No. of byte - 1 byte
Addressing Mode - Register Addressing
Flags - All Flag. affected
Example - DCR B :. B-1 → B
Suppose B = 2F, Flag reg = 01 X 0 X 0 X 1 and instruction DCR B is executed.



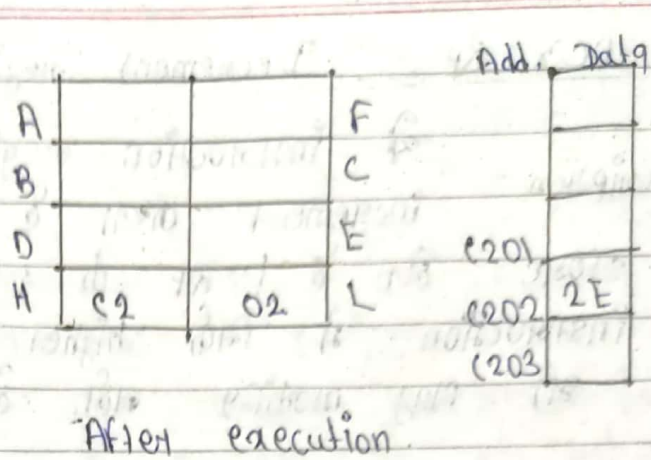
(16) DCR M -. Decrement memory location content by one.

Description - यह instruction memory location के content को, जो HL register pair के through address किया गया है, 1 से decrement करता है। result फिर वही memory location पर store हो जाता है। सिर्फ carry flag modify नहीं होता, बाकी सारे flags modify हो जाते है।

Operation - (HL) → (HL) - 1 → (HL) or M-1 → M
No. of byte - 1 byte
Addressing Mode - Indirect addressing
Flags - All flag affected
Example - DCR M : (HL) - 1 → (HL)
Suppose H = C2, L = 02 at memory location C202 : 2F data is stored Flag reg = 01 X 0 X 0 X 1 and instruction DCR M is executed.



Before execution



After execution

(17) **INX Rp** :- Increment register pair by one.

Description - यह instruction register pair के contents को 1 से decrement करता है और result फिर वही register pair मे store होता है। Rp के ex. है: SP, BC, DE, HL ; इस instruction मे सिर्फ high order register को specify किया जाता है कोई भी flag modify नही होता।

Operation - Rp + 1 → Rp

No. of byte - 1 byte

Addressing mode - register addressing

Flags - ~~All flag~~ affected No flags are modified

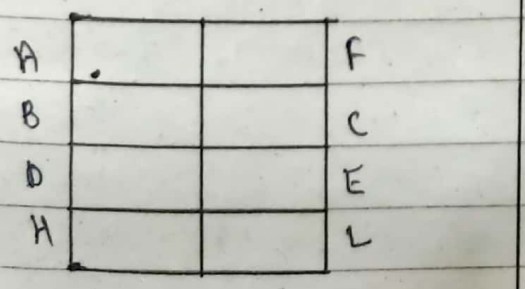
Example - INX SP : SP + 1 → SP

INX B : BC + 1 → BC

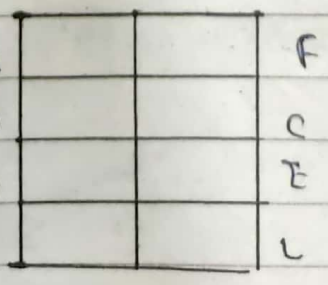
Suppose B = 02 ; C = ff , SP = C200 , flag reg. = 01 X 1 X 1 X 0 and instruction INX B and INX SP are executed.

BC + 1 → BC ; 02ff + 1 → 0300 , so , B = 03 H and C = 00 H

SP + 1 → SP ; C200 + 1 → C201 , so , SP = C201



Before execution



After execution

(18) DCX Rp - Decrement register pair by one.

Description - यह instruction register pair के contents को 1 से increment करता है और result वही register pair में store होता है। Rp के example है : BC, DE, HL, SP. यह instruction में सिर्फ higher order register specify किया जाता है कोई भी flag modify नही होता।

Operation - Rp - 1 → Rp
No. of byte - 1 byte
Addressing mode - Register addressing
Flags - NO flag are modified
Example - DCX D : DE - 1 → DE
DCX H : HE - L → HL

Suppose D = 02, E - FF, H = C2, L = 00, flags reg = 01, x1 x 1x0 and instruction DCX B and DCX H are executed.

DE - 1 → DE ; 02FF - 1 → 02FE
HL - 1 → HL ; C200 - 1 → C1FF

A			F
B			C
D	02	FF	E
H	C2	00	L

Before execution

A			F
B			C
D	02	FE	E
H	C1	FF	L

After execution

Logical Instruction

① ANA R

Description - ANDing bit by bit perform होता है i.e हर bit of accumulator का AND होता है D bit register के साथ example के लिए अगर R कोई general - purpose register है जैसे A, B, D, C, E, H या L. तो हर bit दो bit of register से लेके Dn bit तक accumulator के corresponding bit के साथ AND किया जाता है।

Operation - A AND R → A

No. of byte - 1 byte

Addressing mode - Register addressing

Flags - S, Z, P are modified to reflect the result of operation. Carry is reset and AC is set.

Example - ANA B

Suppose A = 56 and B = 82 and instruction ANA B is executed

```
A = 0 1 0 1 0 1 1 0
B = 1 0 0 0 0 0 1 0
A = 0 0 0 0 0 0 1 0
```

A	56		F
B	82		C
D			E
H			L

Before execution

A	02		F
B	82		C
D			E
H			L

After execution

(2) ANA M - Logical AND memory with accumulator

Description - memory location के content को accumulator के साथ AND किया जाता है और result accumulator में store होता है। memory location का address HL register pair से दिया जाता है। AND operation ANA R instruction के जैसे ही perform होता है।

Operation - A AND M → A OR A AND (HL) → A

No. of byte - 1 byte

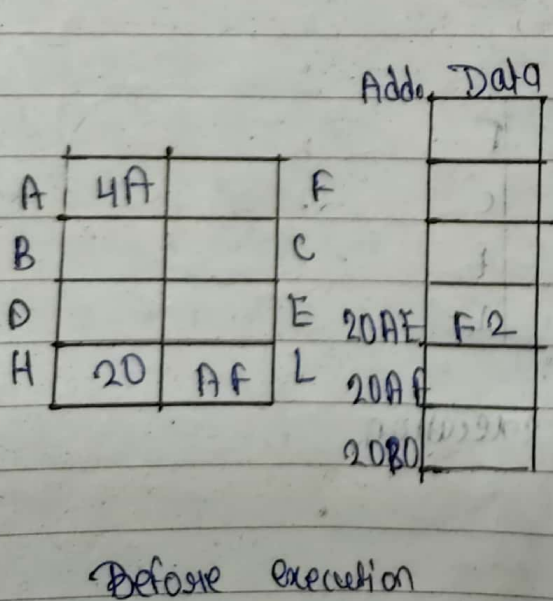
Addressing mode - indirect addressing

Flags - S, Z, P are modified to reflect the result of ANDing
Cy is reset and Ac is set.

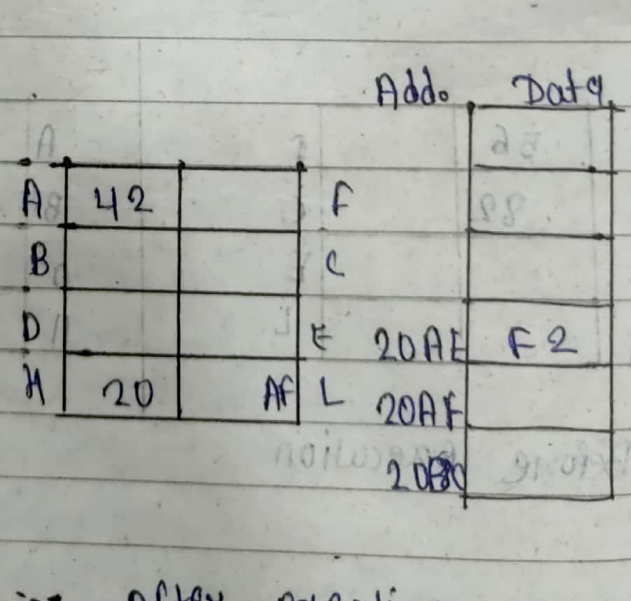
Example - ANA M

Suppose A = 4H, H=20, L=AF at memory location 20 AF : F2 is stored and instruction ANA is executed.

```
A =  0 1 0 0  1 0 1 0
M =  1 1 1 1  0 0 1 0
-------------------------
A =  0 1 0 0  0 0 1 0
```



Before execution



After execution

(3) ANI Data - Logically AND immediate data with accumulator

Description - Accumulator के content को 8-bit immediate data (जो instruction के साथ दिया होता है) के साथ AND किया जाता है और result accumulator में store होता है। AND operation बिल्कुल ANA R जैसे perform होता है।

Operation - A AND data → A

No. of byte - 2 byte

Addressing mode - immediate addressing

Flags - S, Z, P are modified to reflect the result of operation CY is reset and AC is set.

Example - ANI 0F

Suppose A = F7 and instruction ANI 0F is executed.

A = 1 1 1 1 0 1 1 1

Data = 0 0 0 0 1 1 1 1

A = 0 0 0 0 0 1 1 1

A	F7		F
B			C
D			E
H			L

Before execution

A	07		F
B			C
D			E
H			L

After execution

(4) ORA R - Logically OR register contents with accumulator

Description - Register के content और accumulator के content bit by bit OR होते हैं जैसे कि D0 of register OR D0 of accumulator, D1 with D1 और ऐसे ही D7 तक OR का result accumulator में ही store होता है। ये operation

general purpose register के साथ किया जा है जैसे A, B, C, D, E, and L है।

Operation – A OR R → A
No. of byte – 1 byte
Addressing mode – Register addressing
Flags – S, Z, P are modified to reflect the result of operation AC and CY are reset
example – ORA C

Suppose A = A2 H and C = B5 H and instruction ORA C is executed.

```
A =  1 0 1 0  0 0 1 0
B =  1 0 1 1  0 1 0 1
-------------------------
A =  1 0 1 1  0 1 1 1
```

So result in accumulator will be B7 and flag status ORA C is executed

CY = Reset, AC = Reset, S = Set, P = Set, Z = Reset.

A	A2		F
B		B5	C
D			E
H			L

Aft Before execution

A	B7		F
B		B5	C
D			E
H			L

After execution.

(15) **ORA M** – Logical OR memory with accumulator

Description – accumulator के content और memory location (जो HL register pair से point किया गया है) का content logically OR किया जाता है। जो result आता है वह accumulator में store होता है, ये operation वैसे ही perform होता है जैसे ORA R instruction में होता है।

Operation – A OR M → A or A OR (HL) → A

No. of byte – 1 byte

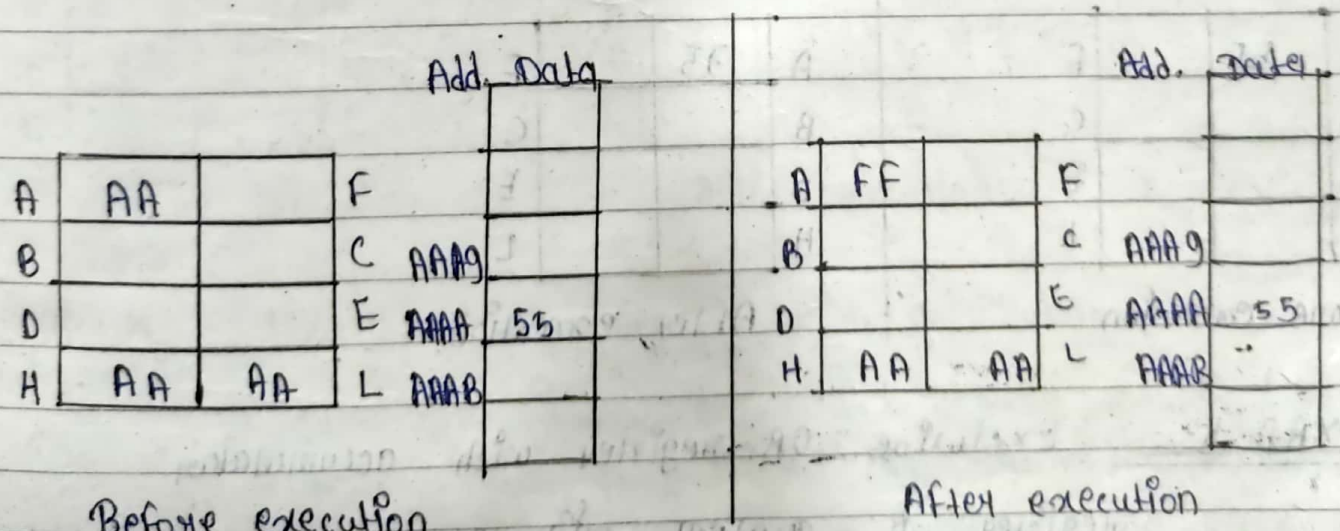
Addressing mode – Indirect addressing

Flags – S, Z, P are modified to reflect the result of operation AC = reset and CY = Reset

Example – ORA M

Suppose A = AA, H = AA, L = AA, at memory location AAAA; 55 data is stored and instruction ORA M is executed.

```
A =  1 0 1 0  1 0 1 0
M =  0 1 0 1  0 1 0 1
     ---------------
A =  1 1 1 1  1 1 1 1
```



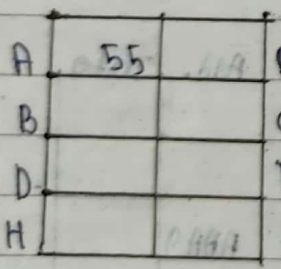
⑥ ORI Data – logically OR immediate data with accumulator

Description – accumulator के content को OR किया जाता है एक 8-bit immediate data के साथ, जो instruction के साथ ही दिया गया होता है। जो result मिलता है उसे accumulator में ही store किया जाता है, ये OR operation बिल्कुल वैसे ही होता है जैसे ORA R instruction में perform होता है।

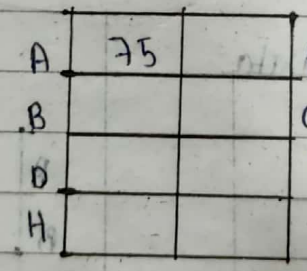
Operation – A OR data → A
No. of byte – 2 byte
Addressing mode – Immediate addressing
Flags – S, Z, P are modified to reflect the result of operation AC = Reset and CY = Reset

Example – ORI 20, suppose A = 55 and instruction ORI 20 is executed.

```
A    = 0 1 0 1 0 1 0 1
Data = 0 0 1 0 0 0 0 0
A    = 0 1 1 1 0 1 0 1
```



Before execution



After execution

⑦ XRA R – Exclusive-OR register with accumulator

Description – register के content और accumulator को bit-by-bit XOR किया जाता है जैसे D0 bit of register को D0 bit of accumulator के साथ XOR किया जाता है।

फिर D1 bit को D1 bit के साथ (XOR) किया जाता है ऐसे करते - करते D7 तक किया जाता है जो result आता है वह accumulator में ही store होते हैं इस instruction के ex जैसे A, B, C, D, E, H and L general purpose register है।

Operation — A EXOR R → A
No of byte — 1 byte
Addressing mode — Register addressing
Flags — S, Z, P are modified to reflect the result of operation. AC and CY are reset.
Example — XRA D

Suppose A = 77 H and D = 06 H and instruction XRA D is executed.

```
A =  0 1 1 1 0 1 1 1
D =  0 0 0 0 0 1 1 0
-------------------
A =  0 1 1 1 0 0 0 1
```

A	77		F
B			C
D	06		E
H			L

Before execution

A	71		F
B			C
D	06		E
H			L

After execution

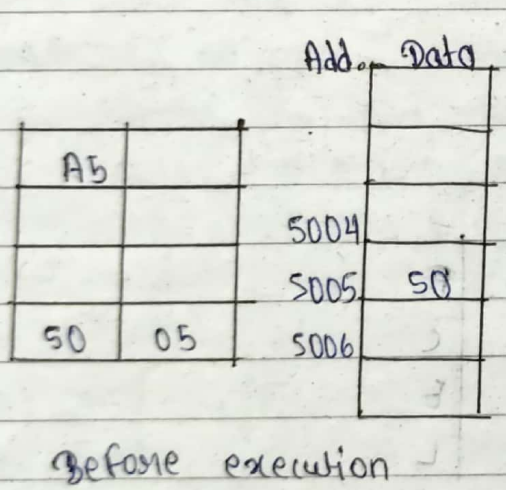
(8) XRA M - Logically EXOR memory with accumulator

Description - accumulator के content और memory location (जो HL register pair से point किया गया है) का content logically EXOR किया जाता है जो result मिलता है वह accumulator में store होता है। EXOR का operation बिल्कुल वैसे ही perform होता है, जैसे XRA R instruction में होता है।

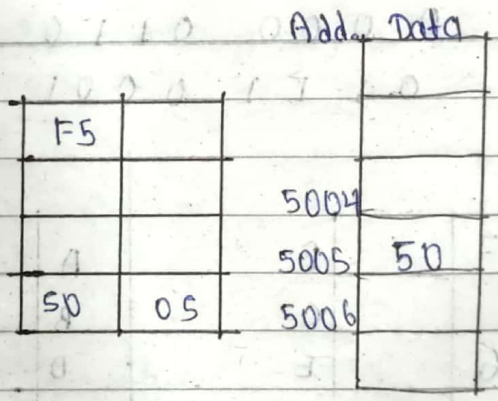
Operation - A EXOR M → A OR A EXOR (HL) → A
No. of byte - 1 byte
Addressing mode - Indirect addressing
Flags - S, Z, P are modified to reflect the result of operation AC = Reset and CY = Reset
Example - XRA M

Suppose A = A5 ; H = 50, L = 05 at memory location 5005 : 50 is stored and instruction XRA M is executed

```
A =  1 0 1 0  0 1 0 1
H =  0 1 0 1  0 0 0 0
-------------------------
A =  1 1 1 1  0 1 0 1
```



Before execution



After execution

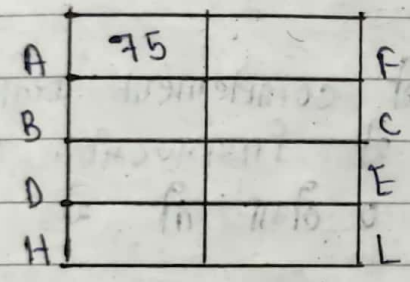
(9) XRI Data - Logically EXOR immediate data with accumulator.

Description - Accumulator के content को 8-bit immediate data के साथ logically EXOR (XOR) किया जाता है, जो instruction के साथ ही दिया गया होता है। जो result आता है, वो accumulator में ही store होता है। EXOR operation बिल्कुल वैसे ही होता है जैसे XRA R instruction में perform होता है।

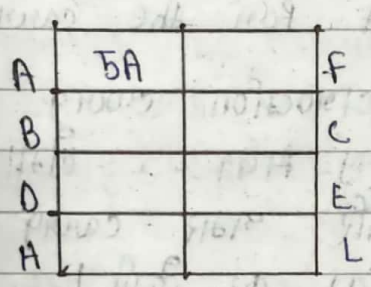
Operation - A EXOR data → A
No. of byte - 2 byte
Addressing mode - Immediate addressing
Flags - S, Z, P are modified to reflect the result of operation AC = Reset and CY = reset

Example - XRI data, suppose A=75 and instruction XRI 2F is executed.

A = 0 1 1 1 0 1 0 1
Data = 0 0 1 0 1 1 1 1
A = 0 1 0 1 1 0 1 0



Before execution



After execution

(10) CMA - Complement accumulator

Description - इसमे accumulator के हर bit पर NOT operation apply किया जाता है यानि जितनी भी 0s है वह 1s बन जाते है और जितनी भी 1s है वह 0s बन जाते है। जो result मिलता है वह फिर से accumulator मे ही store होता है।

Operation - $\overline{A} \rightarrow A$ (Bar placed above A is NOT operation)
No. of byte - 1 byte
Addressing mode - Implicit addressing
Flags - No flag affected

Example - Suppose A = AAH, flag reg. = 10X1X1X1 and CMA instruction is executed.

$A = 1010\ 1010$

$\overline{A} = 0101\ 0101$

A	AA		F
B			C
D			E
H			H

Before execution

A	55		F
B			C
D			E
H			H

After execution

(11) CMC - Complement the carry flag.

Description - ये instruction carry flag को complement करती है अगर carry flag = 1 होगा तो ये instruction उसे reset (0) कर देगी और अगर carry flag = 0 होगा तो ये instruction उसे set (1) कर देगी।

Operation - $\overline{CY} \rightarrow CY$

~~Addressing mode~~ -

No. of byte - 1 byte

Flags - Only carry flag is complemented. No other flag are modified.

(12) STC -

Description - यह instruction carry flag को set करती है इसमें पहले से carry flag क्या था उससे कोई फरक नहीं पड़ता बस carry flag को 1 कर देती है।

Operation - CY = set

No. of byte - 1 byte

Flag - only carry flag is set. No other flag are modified.

(12) CMP R - Compare register with accumulator

Description - ये register के content को accumulator के साथ compare करती है। compare करने का काम accumulator minus register (A-R) करके होता है, लेकिन ना तो accumulator का content change होता है ना ही register का comparision का result flag के through दिखाया जाता है।

- अगर A > R : तो carry flag (CY) reset होता है और zero flag भी Reset होता है।
- अगर A = R : तो zero flag set होता है।
- अगर A < R : तो carry flag set होता है।

इसके अलावा S (sign), P (Parity), AC (Auxiliary carry) flag भी modify होता है subtraction के result के basis पर।
Z (Zero) और CY (carry) flag से comparision का result indicate होता है। इस instruction में R कोई भी general purpose register हो सकता है जैसे A, B, C, D, E, H या L।

Operation — A compare R → Flag register
No. of byte — 1 byte
Addressing mode — Register addressing
Flags — S, Z, P are modified to reflect the result of subtraction and Z, CY are used to indicate the result of comparision

Example — CMP B
Suppose A = 20H and B = 10H and CMP B instruction is executed

```
A = 0010 0000
B = 0001 0000
    0001 0000
```

Flags status will be as follow
Carry = Reset, Zero = Reset and P = Reset, S = Reset, AC = Reset The A=20 and B=10 contents of A and B are not altered, only result is indicated by carry and zero flags.

A	20		F
B	10		C
D			E
H			L

Before execution

A	20		F
B	10		C
D			E
H			L

After execution

(14) CMP M - Compare memory with accumulator

Description - ये instruction accumulator और memory location के content को compare करती है। memory location का address HL register pair के through दिया जाता है। comparision का operation बिल्कुल वैसे ही होता है जैसे CMP R instruction में होता है यानी accumulator minus memory (A-M), लेकिन accumulator या memory का content change नहीं होता। compare का result flag के through दिखाया जाता है (Z, Cy, etc) बिल्कुल CMP R के जैसे।

Operation - A compare M → flag register

No. of byte - 1 byte

Addressing mode - Indirect addressing

Flags - Z and Cy are used to indicate result of comparision.
S, P, AC are modified to reflect the result of instruction.

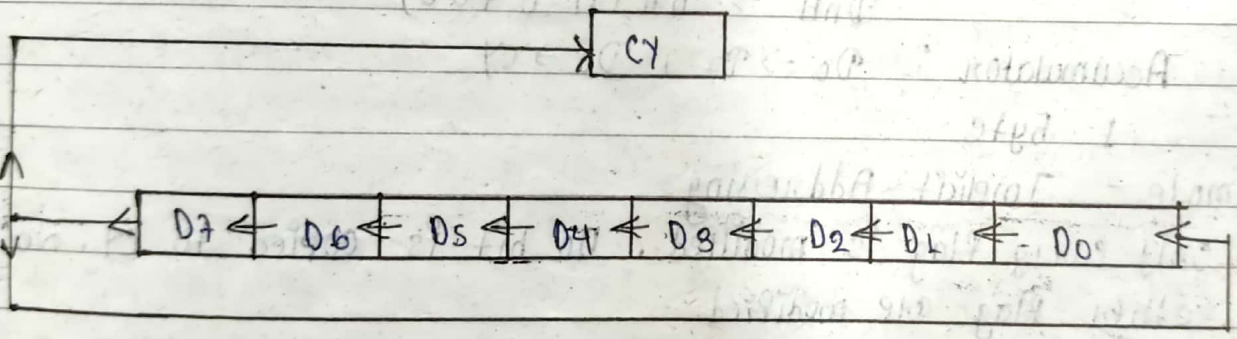
(15) CPI Data - Compare immediate data with accumulator

Description - ये instruction immediate data को accumulator के साथ compare करती है बिल्कुल CMP R instruction की तरह। compare का result Z (zero) और Cy (carry) flag के through दिखाया जाता है। accumulator का content change नहीं होता।

Operation – A compare data → Flag register
No. of byte – 2 byte
Addressing mode – Immediate addressing
Flags – Z and CY are used to indicate result of comparision S, P, AC are modified to reflect the result of subtraction.

(6) RLC – Rotate accumulator left

Description – ये instruction accumulator के content को 1 bit left rotate करती है। इसमें हर bit को left shift किया जाता है। D0 bit चली जाती D1 में D1 to D2, D2 to D3 ... D6 to D7, D7 bit transfer होती है D0 में और carry flag में भी। यानी D7 bit दोनो जगह जाती है - D0 position पर और carry flag में।



$D_n \rightarrow D_{n+1}$ $(n = 0 \text{ to } 6)$

Operation – Accumulator : $D_7 \rightarrow D_0$, $D_7 \rightarrow CY$
No. of byte – 1 byte
Addressing mode – Implicit Addressing
Flags – Only carry flag is modified. D7 bit is copied to carry flag No other flag are modified.
Example – Suppose A = If and instruction RLC is executed.

A	0001 1111	F

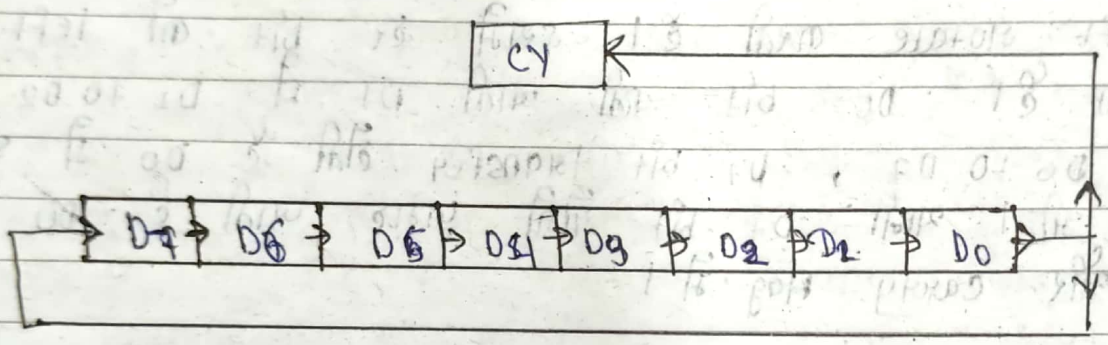
Before execution

A	0011 1110	F

After execution

(17) RRC – Rotate accumulator right

Description – यह instruction accumulator के content को 1 bit right rotate करती है। इसमें हर bit को right shift किया जाता है। D7 चला जाता है D6 में, D6 to D5, D5 to D4 .... D1 to D0, D0 bit transfer होती है D7 में और carry flag में भी।
यानि D0 bit दोनो जगह जाती है - D7 position पर और carry flag में



$D_{n+1} \rightarrow D_n \; (n = 0 \text{ to } 6)$

Operation – Accumulator : $D_0 \rightarrow D_7$, $D_0 \rightarrow CY$
No. of byte – 1 byte
Addressing mode – Implicit Addressing
Flags – Only carry flag is modified. D0 bit is copied to CY. No other flag are modified.
Example – Suppose A = 1C and instruction RRC is executed

A | 0001 1100 | F Before execution

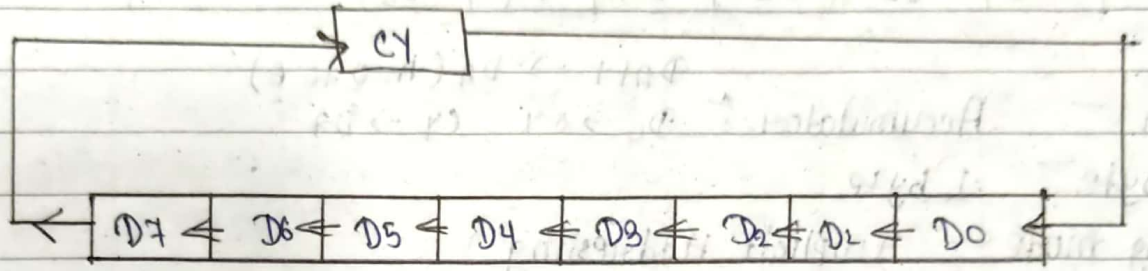
A | 0000 1110 | F After execution

(18) RAL :- Rotate accumulator left through carry

Description – यह instruction accumulator के contents को carry के through 1 bit left rotate करती है।

ये operation RLC जैसे ही होता है, लेकिन एक difference के साथ D0 to D1, D1 to D2, D2 to D3 .... D6 to D7, D7 bit copy होती है carry flag में और जो पहले से carry flag में था वह D0 में चला जाता है।

Operation



Operation - Accumulator : $D_n \rightarrow D_{n+1}$ $(n = 0 \text{ to } 6)$
$D_7 \rightarrow CY$, $CY \rightarrow D_0$

No. of byte - 1 byte

Addressing mode - Implicit addressing

Flags - Only carry flag is modified. D7 bit is copied to carry flag.
No other flags modified

Example - Suppose A = LE and CY = set and instruction RAL is executed.

A	0001 1110	F

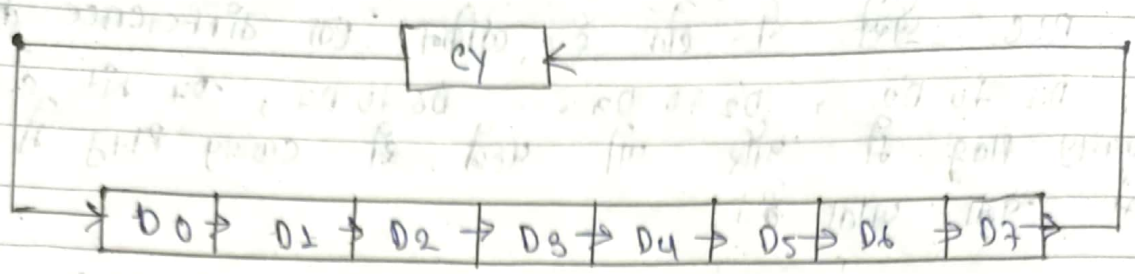
Before execution

A	0011 1101	F

After execution

(19) RAR :- Rotate accumulator right through carry.

Description - ये instruction accumulator के contents को carry के through 1 bit right rotate करती है। ये operation RRC जैसे होता है लेकिन थोड़ा different : D7 to D6, D6 to D5.... D1 to D0, D0 bit copy होती है carry flag में और जो पहले carry flag में था वह D7 में चला जाता है।



Operation – Accumulator : $D_{n+1} \rightarrow D_n (n=0 \text{ to } 6)$
$D_0 \rightarrow CY, CY \rightarrow D_7$

No. of byte – 1 byte

Addressing mode – Implicit Addressing

Flags – only carry flag is modified, D_0 bit is copied to carry flag. No others flag are modified.

Example – Suppose A = 0E and CY = set and instruction RAR is executed.

A	0000 1110	F

Before execution

A	1000 0111	F

After execution

Stack and Machine Control Instruction

यह एक microprocessor का read/write head होता है।

① **SPHL** – Load stack pointer with HL register pair contents.

Description – जब ये instruction execute होता है, तो HL register pair का content stack pointer register में transfer हो जाता है। H register का content जाता है stack pointer के higher order 8 bit में, और L register का content जाता है stack pointer के lower 8 bit में।

Operation – HL → SP

No. of byte – 1 byte

Addressing mode – register addressing
Flags – NO flags affected
Example – suppose HL pair = ABCD, stack pointer = 1234 and SPHL instruction is executed

A			F
B			C
D			E
H	AB	CD	L
SP	1234		

Before execution

A			F
B			C
D			E
H	AB	CD	L
SP	ABCD		

After execution

(6). PUSH Rp – Push the content of register pair on to stack

Description – जब इस instruction को execute किया जाता है तो given register pair का data stack में load हो जाता है। यह operation दो step में होता है।

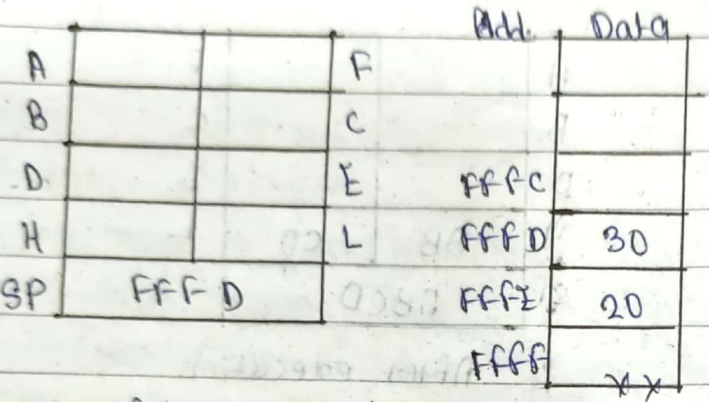
Step (1) – stack pointer को 1 से increment किया जाता है और register pair के higher data को stack में load किया जाता है।

A			F
B	20	30	C
D			E
H			L
FFFF			

add.	data
FFFC	
FFFD	
FFFE	
FFFF	

Before execution

Step ② - Stack pointer को दोबारा 1 से decrement किया जाता है और register pair में lower order register के data को stack में load किया जाता है।



After execution

Operation - lower order RP → SP-2

Higher order RP → SP-1

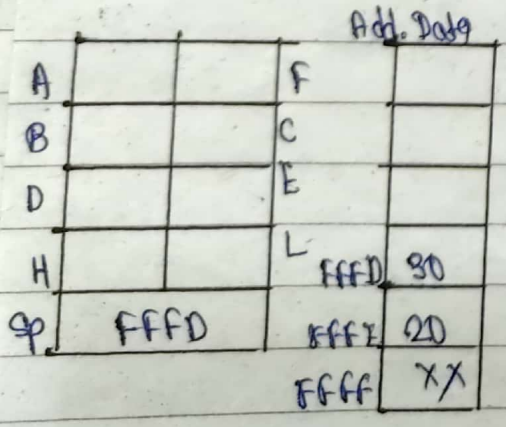
SP-2 → SP new

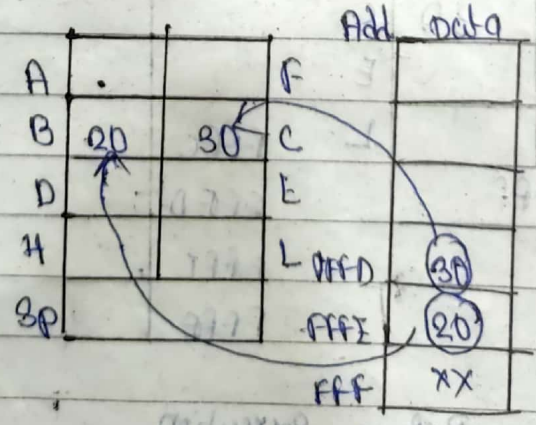
No. of byte - 1 byte

Addressing mode - Register addressing

Flag - no. flag affected

③ POP Rp - जब इस instruction को execute किया जाता है तो stack के data को register pair में load किया जाता है।





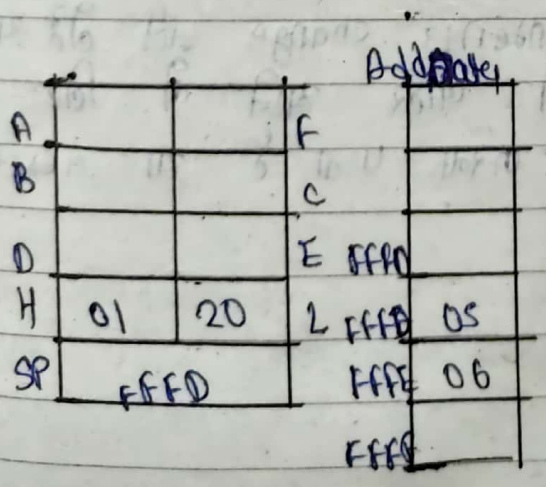
Operation ... lower order $Rp \rightarrow sp$
Higher order $Rp \rightarrow sp+1$
$sp \rightarrow sp+2$

No. of byte - 1 byte
addressing mode - Register addressing
Flag - No. flag affected

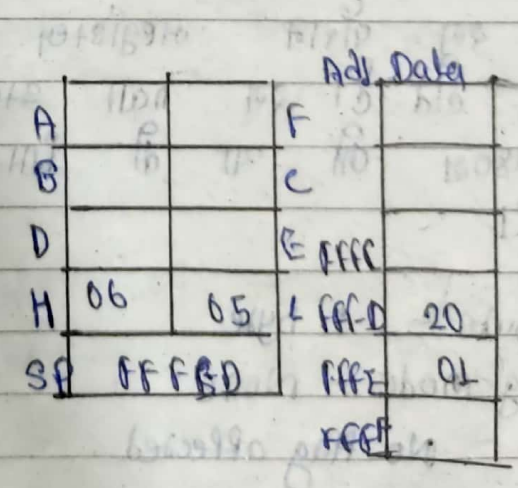
④ XTHL - Exchang HL with top of stack

Description - जब ये instruction execute होता है तो L register का content exchange होता है उस memory location के साथ जिसे stack pointer (sp) point कर रहा है। उसके बाद H register का content exchange होता है next stack location (sp+1) के साथ। इस पूरान stack pointer register का content change नही होता यानी वो unaltered रहता है।

Operation - $L \rightleftarrows (SP)$, $H \rightleftarrows (SP+1)$
No. of byte - 1 byte
Addressing mode - Register indirect addressing
Flage - no flag affected
Example - XTHL, suppose H=01, L=20, SP=FFFD and at memory location FFFD = 05 and FFFE = 06 is stored and XTHL is executed.



Before execution



After execution

⑤ NOP - No operation

Description - जब ये instruction fetch और decode होता है, तब कोई operation perform नही होता। यानी जब ये instruction execute होता है, तो microprocessor कोई काम नही करता और सीधे next instruction पर चला जाता है।
इस instruction को एक छोटे delay के लिए भी use किया जा सकता है कुछ T-states के delay के लिए।
अगर आपको अपने program से कोई instruction delete करनी है तो आप उसकी जगह पर NOP instruction use कर सकते हो।

Operation — PC + 1 → PC
No. of byte - 1 byte
Addressing mode - None
Flag - No flag affected.

⑥ HLT - जब इस instruction को execute किया जाता है। तब microprocessor current execution को complete करता है और कोई भी आगे की instruction execute नही करता।
microprocessor halt acknowledge machine cycle मे enter करता है और हर clock period मे wait states insert हो जाते है।
address और data bus को high impedance state मे डाल दिया जाता है। इस दौरान register के contents change नही होते यानी unmodified होते है। इस halt state से बाहर आने के लिए microprocessor को या तो interrupt करना पड़ता है या reset देना पड़ता है।

No. of byte — 1 byte
Addressing mode — None
Flags — No flag affected.

Addressing mode of 8085 -

किसी instruction मे data को जिस प्रकार से represent किया जाता है। उसे addressing mode कहते है 8085 mic processor मे 5 addressing mode होते है।

① **Immediate Addressing Mode -** इस addressing mode मे data को directly Instruction मे mentioned किया जाता है।

ex - MVI B, 65 H (MVI R, 8 bit data)
LXI D, 6500H (LXI Rp, 16 bit data)

Immediate addressing mode वाले instruction minimum 2 byte के होते है जिसमे पहला byte OPCODE का होता है।

② **Register Addressing mode -** इस addressing mode मे data एक general purpose register मे present होता है। register addressing mode वाले instruction generally 1 byte के होते है।

ex - ADD R → ADD B
MOV A, B

③ **Direct Addressing mode -** direct addressing mode मे data के address को instruction मे numerically mention किया जाता है।

ex - LDA C200
STA C201

direct addressing mode वाले instruction generally 3 byte के होते है जिसमे पहला byte OPCODE दूसरा तीसरा byte address का होता है।

④ Indirect addressing mode – Indirect addressing mode में instruction में data के address को mention नही किया जाता है। बल्कि instruction उस location को point करता है जहा data का address store होता है।

ex MOV A, M
LDAX Rp

⑤ Implicit / Implied / Inherent → इस addressing mode में किसी Operand की specification नही होती है। instruction का opcode ही data को specify करती है। ये instruction generally one byte के होते है।

ex – RAL, RLC, RAR, CMA.

Types Of Machine Cycle

① OPCODE FETCH cycles – microprocessor cycle का use memory से opcode को read करने के लिए करता है और इस process को opcode fetch कहा जाता है।

② opcode जिस memory location में store होता है उसका address Program counter (PC) provide करता है।

③ opcode fetch cycle किसी भी instruction cycle का पहला machine cycle होता है।

④ इस opcode को decode और execute करने के लिए instruction register और decoder unit में भेजा जाता है।

⑤ opcode fetch machine cycle में 4T-state या 6T-state.

कि आवश्यकता होती है

② Operand fetch cycle - दो या तीन byte वाले instruction में opcode fetch cycle के अलावा operand fetch cycle की भी आवश्यकता होती है।

② दो byte वाले instruction में एक operand fetch cycle और 3 byte वाले instruction में 2 operand fetch cycle की आवश्यकता होती है।

③ Instruction में mentioned operand का address program counter (PC) provide करता है।

2000H : LXI B, C200H

PC : opcode fetch 2000H opcode (LXI B)

PC : operand fetch { 2001H 00H
2002H C2H

④ लेकिन operand का content instruction register में न जाकर temporary register में जाता है।

⑤ operand cycle के लिए general 3T-states की आवश्यकता होती है।

③ Memory Read cycle - इस cycle का use memory से data को Read करने के लिए किया जाता है।

① data जिस address में present होता है उसके memory location को instruction के द्वारा point किया जाता है।

④ Memory Write cycle - इस cycle का use microprocessor के द्वारा memory में data को write करने के लिये किया जाता है।

① data जिस address में present होता है उसके memory location

को instruction के द्वारा point किया जाता है।

⑤ I/O Read cycle - इस cycle का use microprocessor के द्वारा I/O device से data को read करने के लिए किया जाता है।

① I/O device जिस port से connect होता है उसका address instruction के द्वारा दिया जाता है port (8 bit) address होता है।

⑥ I/O write cycle - इस cycle का use microprocessor के द्वारा I/O device में data को write करने के लिए किया जाता है।

① I/O device जिस port से connect होता है। उसका address instruction के द्वारा दिया जाता है।

Timing Diagram of opcode fetch cycle -

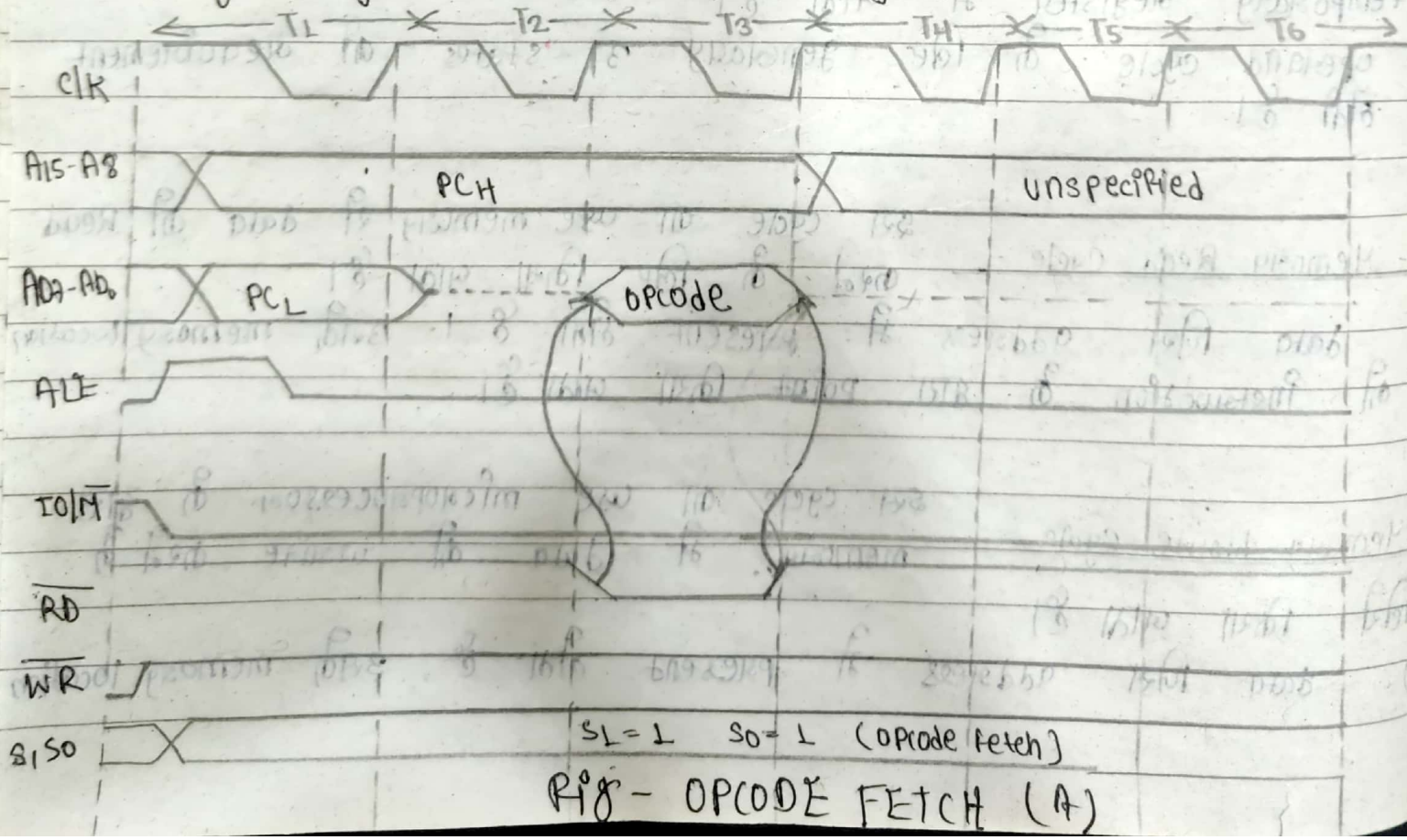


Fig - OPCODE FETCH (A)

Operation

Step ① → (State T_L) → T_L state में microprocessor IO/$\overline{M}$, S_1 और S_0 status signal generate करता है।

$IO/\overline{M} = 0$, $S_1 S_0 = 1$

जो यह indicate करता है। कि opcode fetch cycle. microprocessor 16 bit address को $A_{15} - A_8$ और $AD_7 - AD_0$ lines पर send करता है। program counter का higher order byte $A_{15} - A_8$ line में place किया जाता है और यह (T_3) T state तक इन line पर present रहता है।

② Program counter को lower order byte $AD_7 - AD_0$ line पर place किया जाता है। और यह (T_1) T state तक इन lines पर present रहता है।

③ (T_L) T state के दौरान ALE signal positive pulse provide करता है। जो यह indicate करता है कि $AD_7 - AD_0$ lines पर address present है।

Step (2) (State T_2) – Program counter का lower order byte $AD_7 - AD_0$ lines से disappear हो जाता है ताकि इन lines का use data line की तरह किया जा सके। microprocessor के द्वारा $\overline{RD}$ signal को low कर दिया जाता है। जिससे memory का read circuit enable हो जाता है।

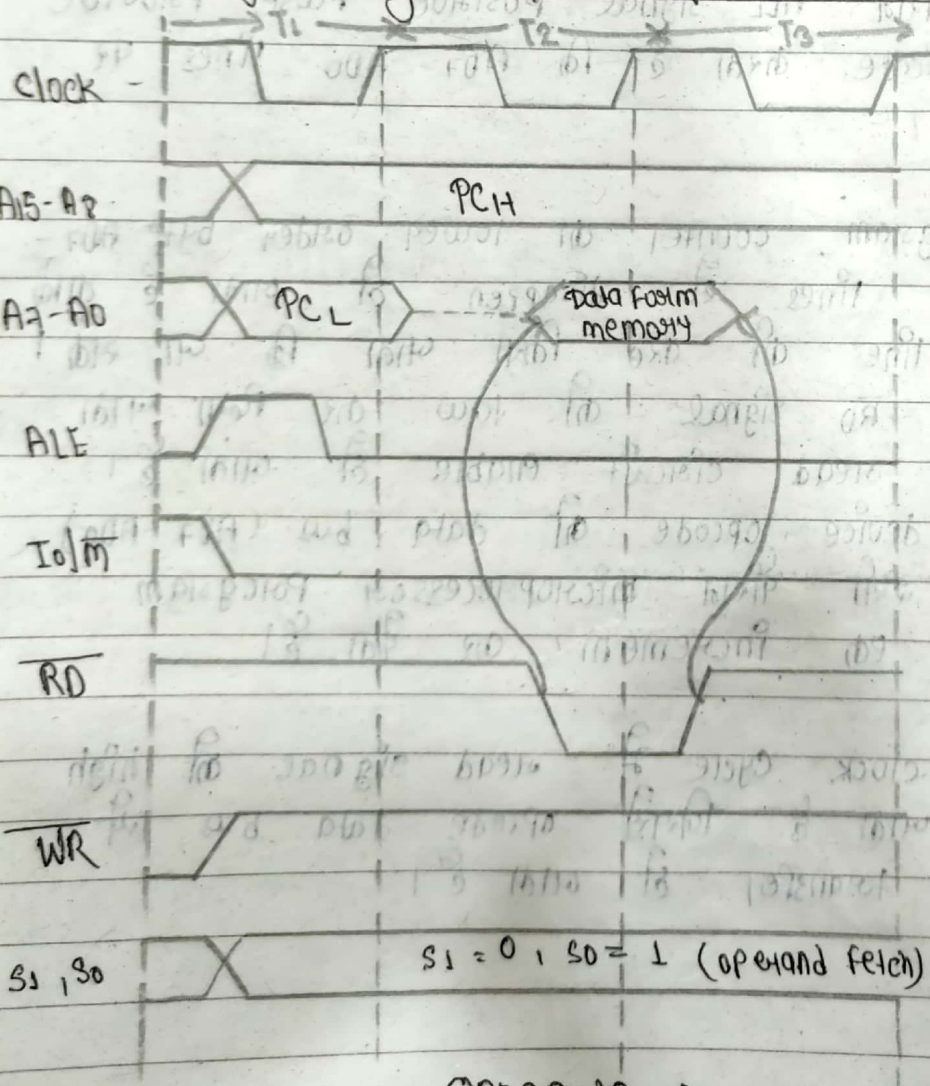
② इसके बाद memory device opcode को data bus ($AD_7 - AD_0$) पर place कर देता है इसी दौरान microprocessor program counter के content को एक increment कर देता है।

Step (3) (State T_3) – इस clock cycle में read signal को high किया जाता है जिससे opcode data bus से instruction register में transfer हो जाता है।

Step (4) (State T_4) – इस clock cycle में microprocessor internal operation perform करता है। microprocessor के द्वारा opcode को decode किया जाता है और decoding के according microprocessor execution start करता है।

Step (5) (State T_5, T_6) – जिन instruction के opcode fetch के लिए T_6 state opcode की अवश्यकता होती है। उनमे decoding को compile करने के लिए और internal operation perform करने के लिए T_5 और T_6 state की जरूरत होती है।

Timing Diagram of Operand Fetch –



OPERAND FETCH (B)

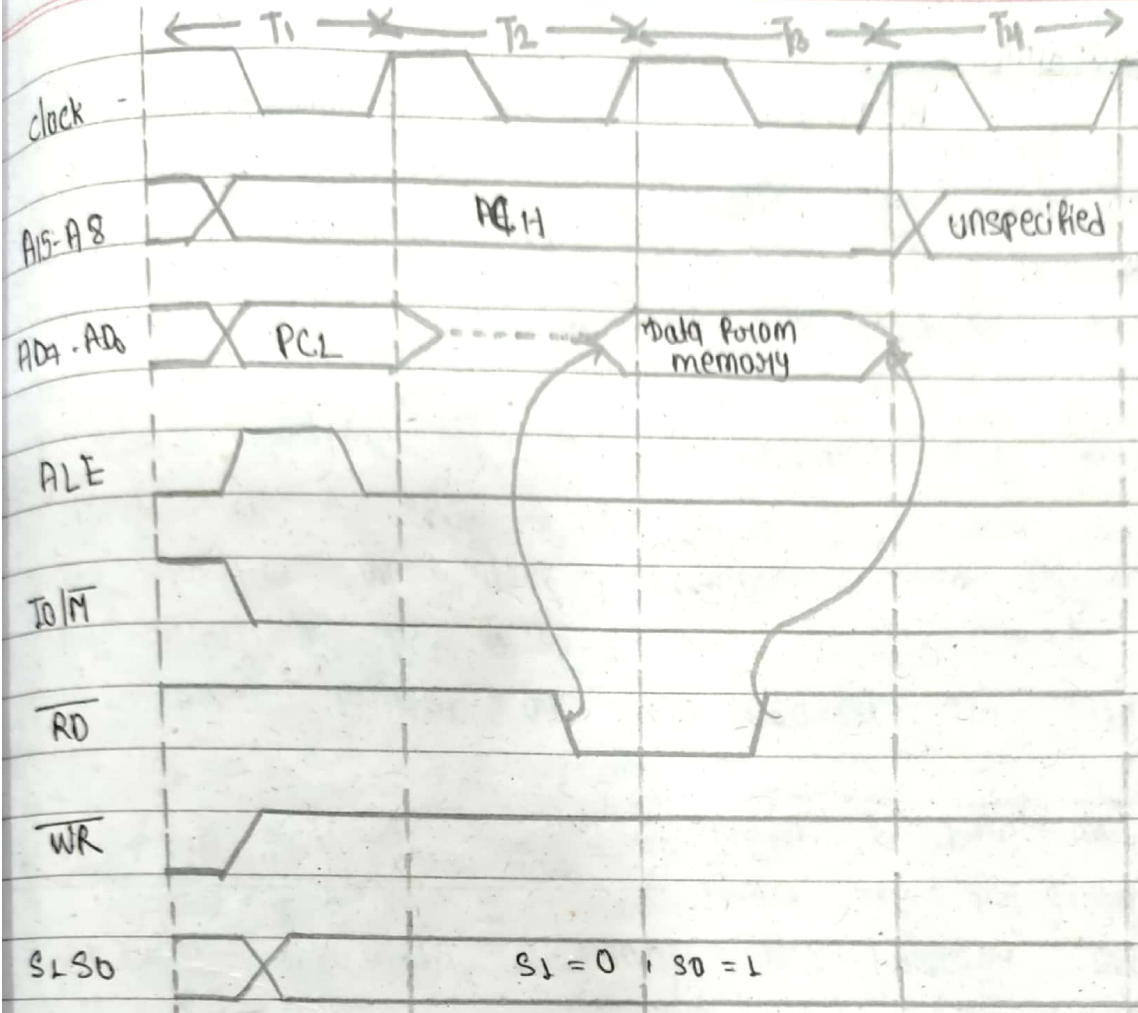
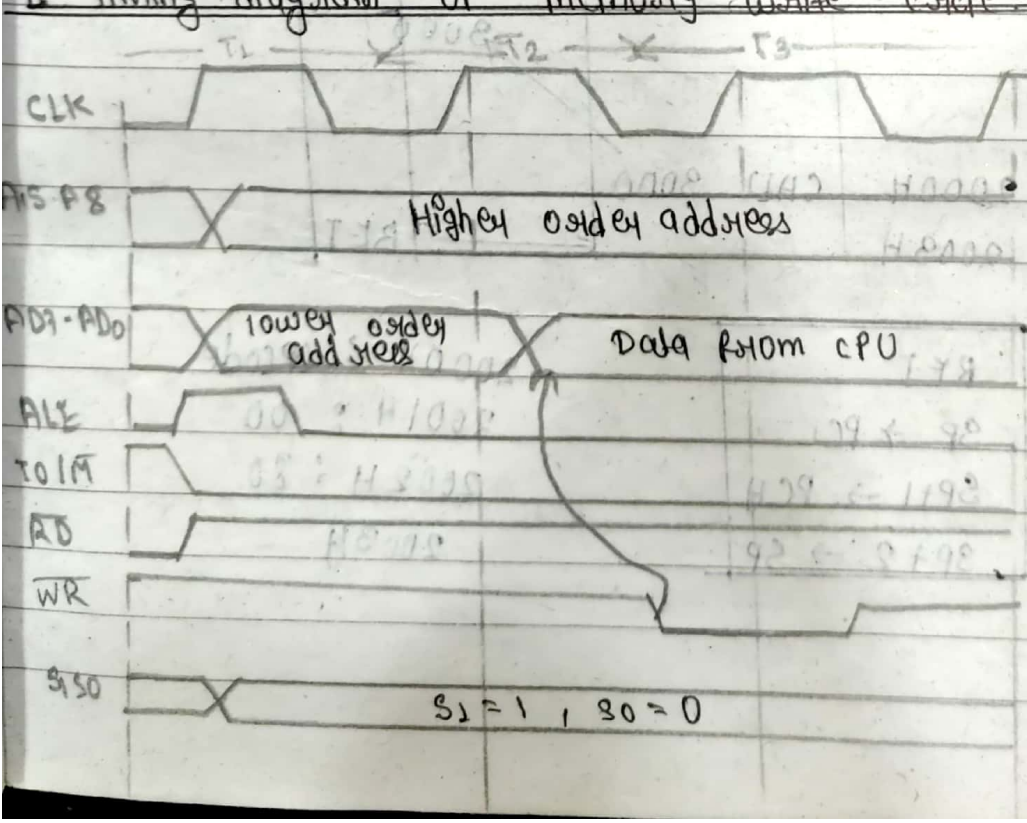


Fig - Memory Read

Timing diagram of memory write cycle



Branching Instruction

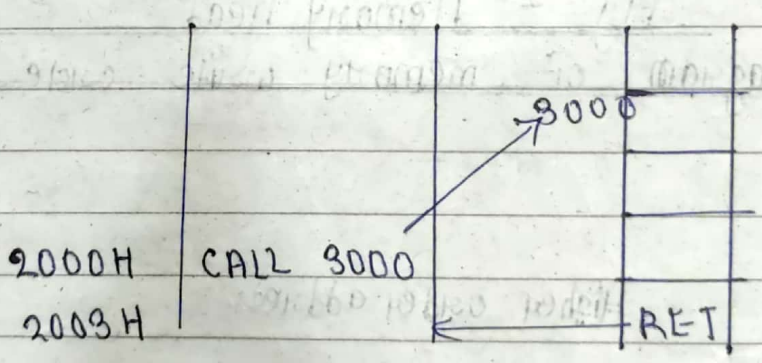
Jump CALL RET

- Jump
  - unconditional
    - Jump 16 bit address
  - conditional
    - JC address (16 bit)
    - JNC address
    - JZ address
    - JNZ address
    - JPE address
    - JPO address
    - JP address
    - JM address

CALL Instruction

Stack

9003	03 (PCL)
9004	20 (PCH)
9005	XX



Call
PCH → SP-1
PCL → SP-2
SP-2 → SP

RET
SP → PCL
SP+1 → PCH
SP+2 → SP

2000H : Opcode
2001H : 00
2002H : 80
2003H

* CALL Instruction ⇒ CALL एक 3 byte का instruction होता है। जिसका use microprocessor के द्वारा program control को subroutine में transfer करने के लिए किया जाता है।

(2) Program control को subroutine में transfer करने से पहले next instruction (CALL के बाद) के address को stack में store कर लिया जाता है।

(3) Stack pointer को एक decrement करके PC_H को उसमें load किया जाता है $PC_H \rightarrow SP - 1$
Stack pointer को दोबारा decrement करके PC_L को उसमें store किया जाता है $PC_L \rightarrow SP - 2$
SP - 2 stack pointer का new address बन जाता है $SP - 2 \rightarrow SP$ new

* RET (Return) Instruction = Return 1 byte का instruction होता है जिसका use program control को subroutine से main program में transfer करने के लिए किया जाता है।

(2) Program control main program में transfer करने से पहले returning address को stack से program counter में load किया जाता है।

(3) Stack pointer के content को PC_L में load किया जाता है और stack pointer को एक से increment कर दिया जाता है।
$SP \rightarrow PC_L$

(4) SP+1 के content को PC_H में load किया जाता है और program counter को एक से increment कर दिया जाता है।
$SP + 1 \rightarrow PC_H$

(5) SP+2 stack pointer का new address बन जाता है।
$SP + 2 \rightarrow SP$

Unconditional

JMP address - यह एक 3 byte का unconditional jump instruction है। जिसके द्वारा program control को instruction मे दिये गये address मे transfer कर दिया जाता है। इस instruction मे program control को transfer करने से पहले किसी भी condition को check नही किया जाता इसलिए इसे unconditional jump instruction कहते है।

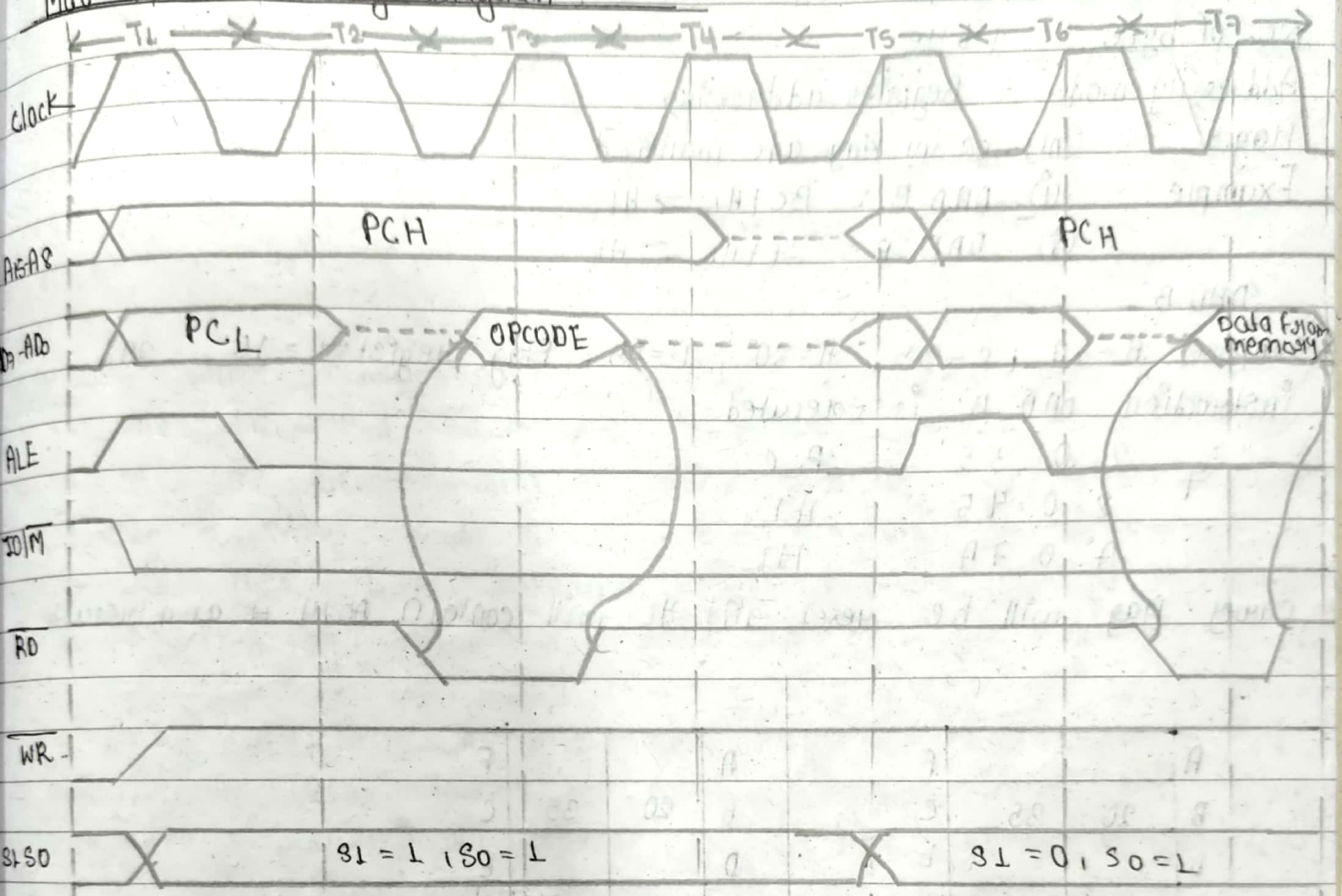
Ex - JMP control 8200H : C005

C005 : OPCODE
C006 : 00
C007 : C2

Conditional JMP Instruction ये instruction 3 byte के होते है। जिसमे program control को jump address पर transfer करने से पहले एक condition को check किया जाता है।

(2) यदि condition satisfy हो जाते है। तो program control jump add. पर transfer हो जाता है और यदि condition satisfy नही हो जाती है तो microprocessor next instruction को execute करना start करता है।

MOV R, Mem Timing Diagram



Data Transfer Group Instruction.

① **DAD Rp** - Add the specified register pair to HL pair

Description - ये instruction दिये गये register pair के contents को HL के content के साथ add करता है और result को और result को HL pair मे store करता है example के लिये Rp हो सकता है: SP, BC, DE या HL, इस operation के बाद सिर्फ carry flag update होता है बाकी flag पर कोई effect नही पड़ता है।

Operation – RP + HL → HL
No. of byte – 1 byte
Addressing mode – Register addressing
Flags – Only carry flag are modified
Example – (i) DAD B : BC+HL → HL
(ii) DAD SP : SP+HL → HL

DAD B

Suppose B=20, C=35, H=80, L=45, Flag register = 10 and instruction DAD B is executed

	2 0 3 5	B C
+	8 0 4 5	H L
	A 0 7 A	HL

carry flag will be reset and HL will contain A074 H as a result

A			F
B	20	35	C
D			E
H	80	45	L

Before execution

A			F
B	20	35	C
D			E
H	A0	7A	L

After execution.

(2) DAA – Decimal Adjust Accumulator

Description – Accumulator के content को binary value से equivalent 2-digit 4-bit BCD number में convert किया जाता है। ये एक ऐसी instruction है जो auxiliary carry flag का use करती है binary to BCD conversion के लिए।

DAA Instruction working – ये instruction following condition check करता है।

① अगर accumulator के low order 4 bits ($D_3 - D_0$) की value 9 से ज्यादा है या AC flag set है तो instruction accumulator के lower 4 bits में 6 add करती है।

② अगर high order 4 bits ($D_7 - D_4$) की value 9 से ज्यादा है या CY flag set है तो instruction accumulator के high order 4 bits में 6 add करती है।

Note – DAA instruction flag condition पे depend करती है अगर आप कोई ऐसी ~~accumulator~~ instruction run करते हो जो CY या AC flag को affect करती है और उसके बाद आप accumulator को BCD format में adjust करने के लिए DAA use करते हो तो result galat हो सकता है। instruction हमेशा previous instructions के AC और CY flag का use करके decision लेती है।

So, DAA instruction का use ADD, ADI जैसे instruction के साथ होता है जब BCD format में addition perform करनी होती है ADD instruction hexadecimal format में 2 BCD numbers का addition करता है और DAA उस result को BCD format में convert करता है।

Operation - A reg. in binary → A reg. in BCD
No. of byte - 1 byte
Flags - All flag affected
Example - 1. अगर आप दो BCD numbers 12 और 39 को add करना चाहते हैं और result भी BCD form में चाहिए तो नीचे दिये गये steps follow करने होंगे।

```
MVI M, 12
ADI 39
DAA
```

Immediate data 39 is added to A (=12)

```
  0 0 0 1   0 0 1 0   12
+ 0 0 1 1   1 0 0 1   39
  ---------------------
  0 1 0 0   1 0 1 1   4B
```

The answer is 4B in hexadecimal form. When DAA instruction executed it checks.

(a) अगर low order 4 bits (D_0-D_3) का value 9 से ज्यादा हो या AC flag set हो, तो इन दोनो मे से कोई भी condition satisfy होती है तो 6 को low order 4 bits मे add किया जाता है। इस example मे B>9 है और AC reset है इसलिए 6 को low order 4 bit मे add किया जायेगा।

(b) अगर high order 4 bits (D_4-D_7) का value 9 से ज्यादा हो या CY flag set हो, तो इन मे से कोई भी condition satisfy होने पर 6 को high order 4 bit मे add किया जाता है, इस example मे कोई भी condition satisfy नही हो रही है क्योकि 4≤3 है और CY = reset है, इसलिए 6 को high order 4 bits मे add नही किया जायेगा।

```
  0 1 0 0   1 0 1 1
+           0 1 1 0
  ---------------------
  0 1 0 1   0 0 0 1
```

result in accumulator = 51 in BCD form.

A	48		F
B			C
D			E
H			L

Before execution

A	51		F
B			C
D			E
H			L

After execution.

Unit-03

Introduction to Interrupt :-

① Interrupt एक data transfer process होते है। जिसमे कोई external device या peripheral processor को inform करता है कि वह communication के लिए ready है और उसकी attention request करता है।

② Interrupt process को external device initiate करता है और यह asynchronous होता है।

③ जिसे system clock के reference के बीना किसी भी time initiate किया जा सकता है लेकिन Interrupt request के response को microprocessor ही control करता है।

Interrupt request दो type के होते है :-

① maskable interrupt :- 8085 microprocessor मे 4 maskable interrupt होते है। और 1 non maskable Interrupt होता है।
microprocessor किसी maskable Interrupt को ignore या delay कर सकता है। जब वह कोई critical task perform कर रहा हो। लेकिन non maskable Interrupt request को immediately response करना पड़ता है।

Interrupt control logic Instruction :-

① EI (Enable Instruction) :- यह एक one byte instruction Interrupt enable Flip-flop को set करता है। जिससे कि interrupt process enable हो जाता है।

② DI (Disable Interrupt) :- यह one byte instruction DI

Instruction interrupt enable Flip-Flop को reset करता है जिससे कि Interrupt Flip-Flop disable हो जाता है।

Software Interrupt - Microprocessor 8085 में 8 s/w Interrupt होते हैं। RST 0, 1, 2, 3, 4, 5, 6, 7 इन्हें 1 byte call instruction भी कहा जाता है। जो program control को एक specific memory location पर transfer करते हैं जिसे interrupt service routine (ISR) कहा जाता है।

Instruction	Address of ISR
RST0	0000H
RST1	0001H
RST2	0010H
RST3	0018H
RST4	0020H
RST5	0028H
RST6	0030H
RST7	0038H

The execution of RST instruction is as follows -

(1) RST instruction call instruction की तरह ही execution है।

(2) RST N के corresponding ISR पर program control transfer होने से पहले stack में PC के content को store कर लिया जाता है।

(3) ISR का last instruction RET instruction होता है। जब MP return instruction को execute करता है तब stack का

data PC मे load हो जाता है और PC main program मे वापस आ जाता है।

Interrupt Structure of 8085 -

(1) Intel 8085 मे पाँच interrupt input होते है।
TRAP, RST 7.5, RST 6.5, RST 5.5, INTR
TRAP की priority highest होती है और उसके बाद RST 7.5, RST 6.5 व RST 5.5 की priority होती है। INTR की priority lowest होती है।

(2) जब भी interrupt का use किया जाता है तब उसे main program मे software के द्वारा EI instruction का use करके enable किया जाता है।
EI instruction interrupt enable flip-flop को set करके सभी interrupt को enable कर देता है।

(3) DI instruction का use interrupt को disable करने के लिए किया जाता है कुछ condition मे जब microprocessor कोई important task perform कर रहा हो, interrupt को disable करना जरूरी हो जाता है।
इन condition मे DI instruction का use किया जाता है DI instruction interrupt enable flip flop को reset कर देता है और non maskable TRAP को छोड़कर बाकी सभी interrupt को disable कर देता है।

(4) System RESET interrupt enable flip-flop को reset कर कर देता है। जिसमे सारे interrupt disable हो जाते है।

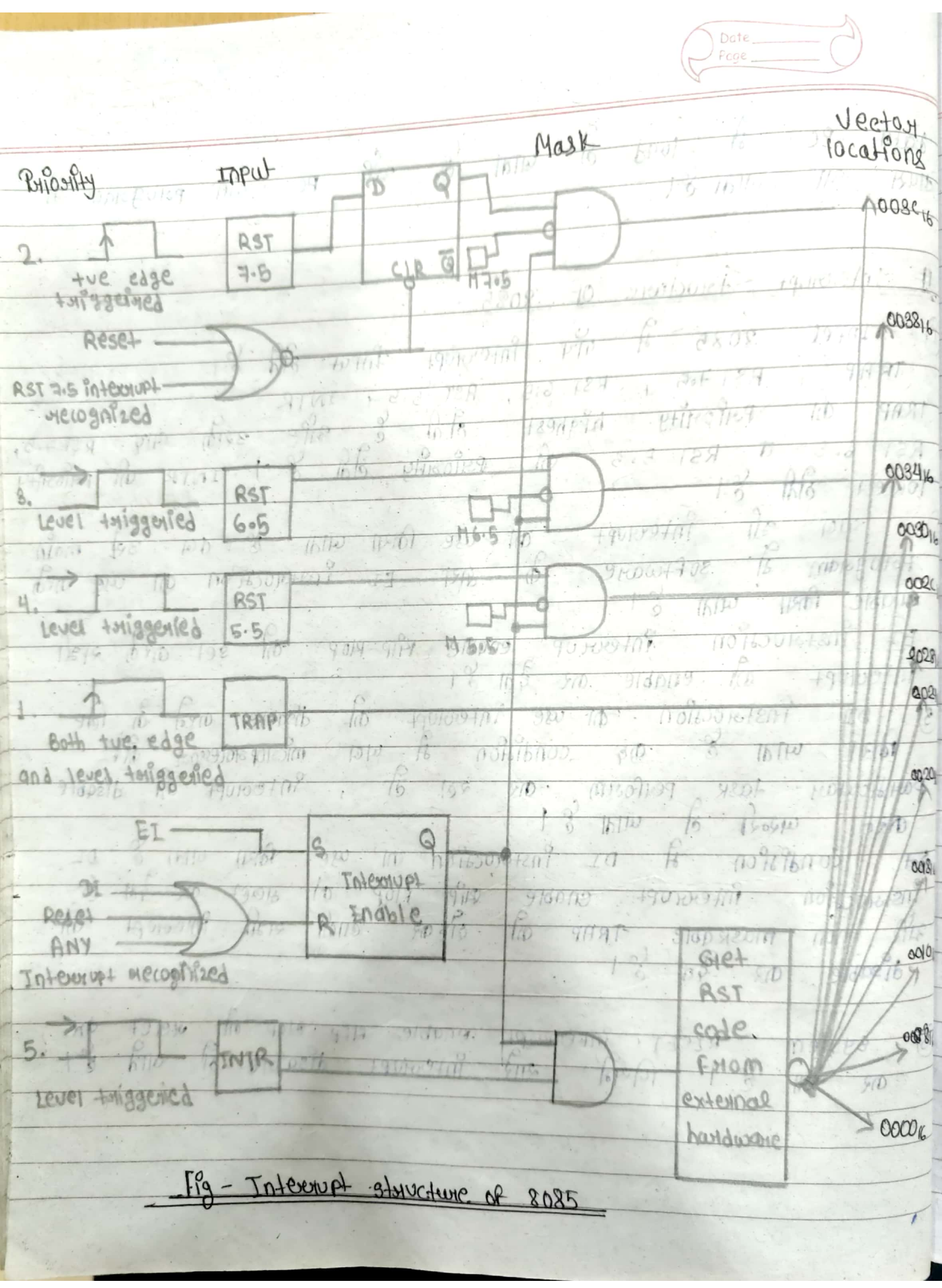


Fig - Interrupt structure of 8085

(5) जब कोई interrupt request आता है तब microprocessor current instruction को complete करता और counter के content को stack में store करता है।
यहाँ interrupt enable flip-flop को interrupt service subroutine (ISS) में जाने से पहले RESET कर देता है ताकि ISS के execution के दौरान कोई दूसरा interrupt request न आ सके।
Interrupt enable flip flop को तीन type से RESET किया जा सकता है -
(1) software के द्वारा DI instruction का use करने।
(2) System RESET
(3) Recognition of interrupt

(6) Interrupt service subroutine (ISS) से main program में वापस लौटने से पहले सभी interrupt को enable कर दिया जाता है ऐसा करने के लिये ISS के return (RET) instruction में पहले EI instruction का use करते है।

(7) कुछ condition में जब microprocessor current task perform करा रहा होता है कुछ interrupt को disable करना जरूरी हो जाता है। इन interrupt को masking का use करके disable किया जाता है ऐसा interrupt जिन्हे mask किया जा सकता है। उन्हे maskable interrupt कहते है।
masking software के द्वारा किया जाता है. TRAP non maskable होता है, जबकि RST 7.5 RST 6.5 RST 5.5 maskable interrupt होते है। RST 7.5, RST 6.5, RST 5.5 को mask करने के लिये SIM instruction को masking bits M7.5, M6.5 और M5.5 का use किया जाता है।

Concept of Stack -

① Program के execution के दौरान कभी - कभी register को content को save करना जरूरी होता है क्योंकि इन registers की assignment दूसरे operation के लिये होता है।

② Registers के content को external memory location में push operation के दौरान move किया जाता है। इसके बाद registers का use दूसरे operation के लिये किया जाता है operation complete होने के बाद memory में save किये गये content को pop operation के दौरान दोबारा registers में transfer किया जाता है।

③ इस purpose के लिए memory location को program के द्वारा पहले ही define किया जाता है। इस प्रकार से define किये गये memory location को stack कहा जाता है।

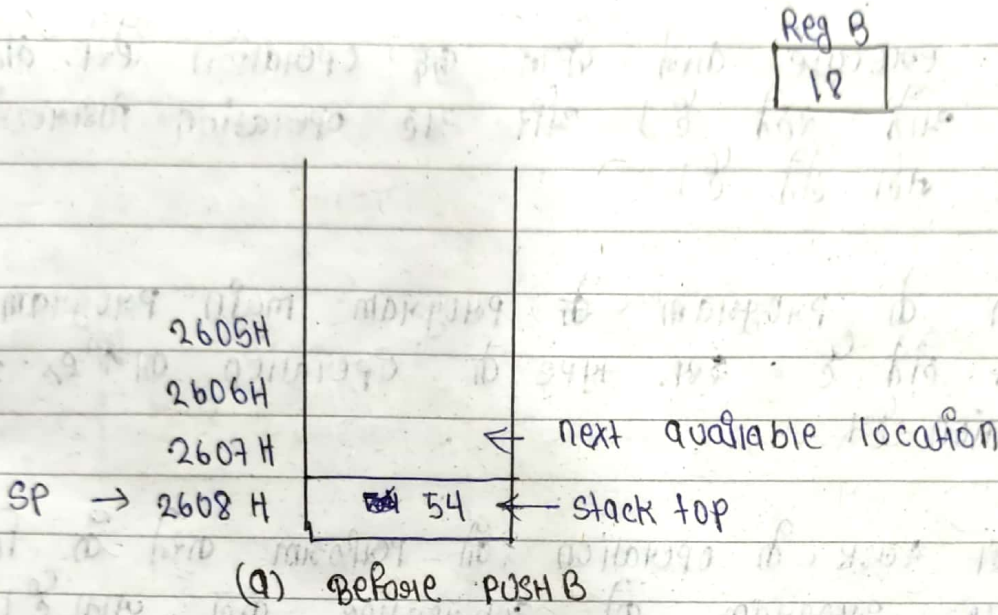
Definition - Stack read/write memory का ही part होता है। जिसका use microprocessor के द्वारा temporary data intermediate result को store करने के लिए किया जाता है।

Initialization - stack को initialize करने के लिये LXI SP या SPHL instruction का use किया जाता है। memory के किसी भी area का use stack के लिये किया जा सकता है। memory में stack location के लिए कोई restriction नहीं है।

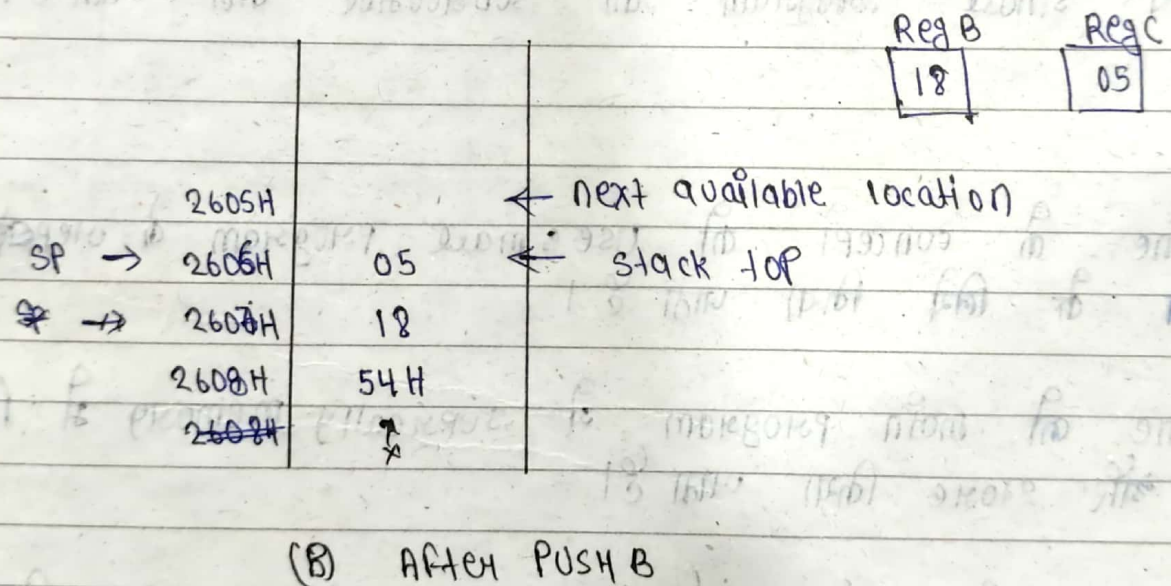
Principle - stack last in first out (LIFO) के principle पर work करता है।

Stack Top - stack memory के last occupied address को stack top कहा जाता है। stack pointer जो कि

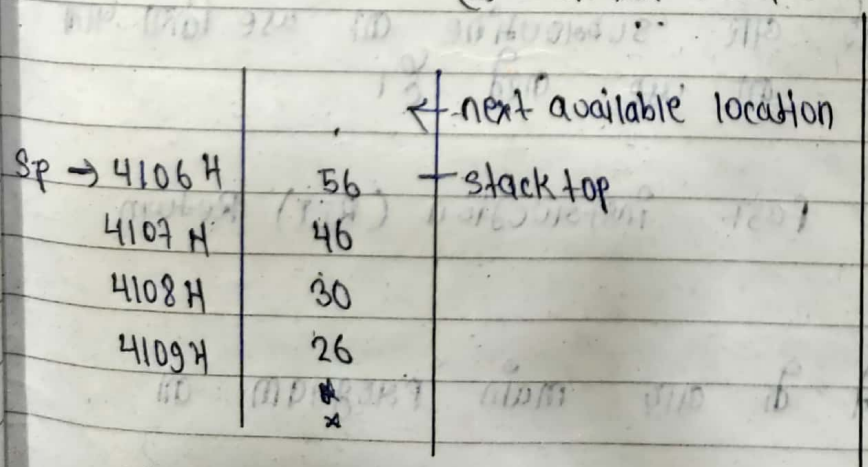
एक special 16 bit रजिस्टर होता है। stack top के address को hold करता है।



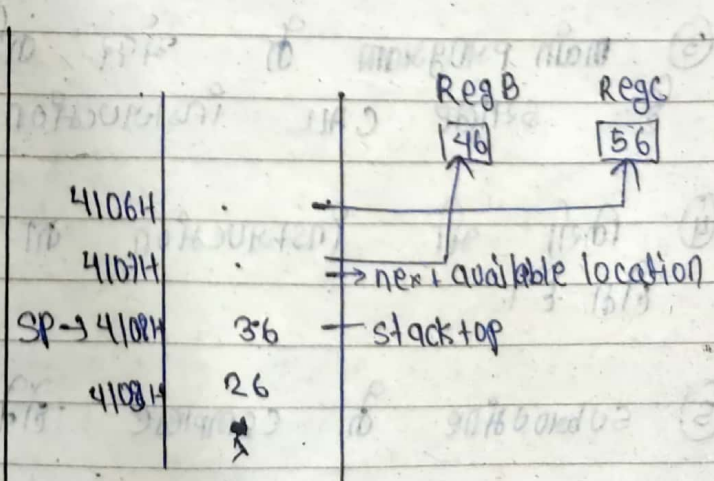
(A) Before PUSH B



(B) After PUSH B



(C) Before POP B



(D) After POP B

Concept Of Subroutine :-

Definition :-

① Program को execute करने दौरान कुछ operation ऐसे होते है जो बार-बार आते रहते है। और यह operation instruction के form मे available नही होते है।

② इन operation के ~~program के~~ program main program मे बार-बार repeat होते है. इस type के operation का ex. Intel 8085 multiplication है।

③ किसी particular task के operation को perform करने के लिए लिखे गये small program को subroutine कहा जाता है।

Use :-

① subroutine के concept को use small program के repetation से बचने के लिये किया जाता है।

② subroutine को main program से superiority memory मे लिखा जाता है और store किया जाता है।

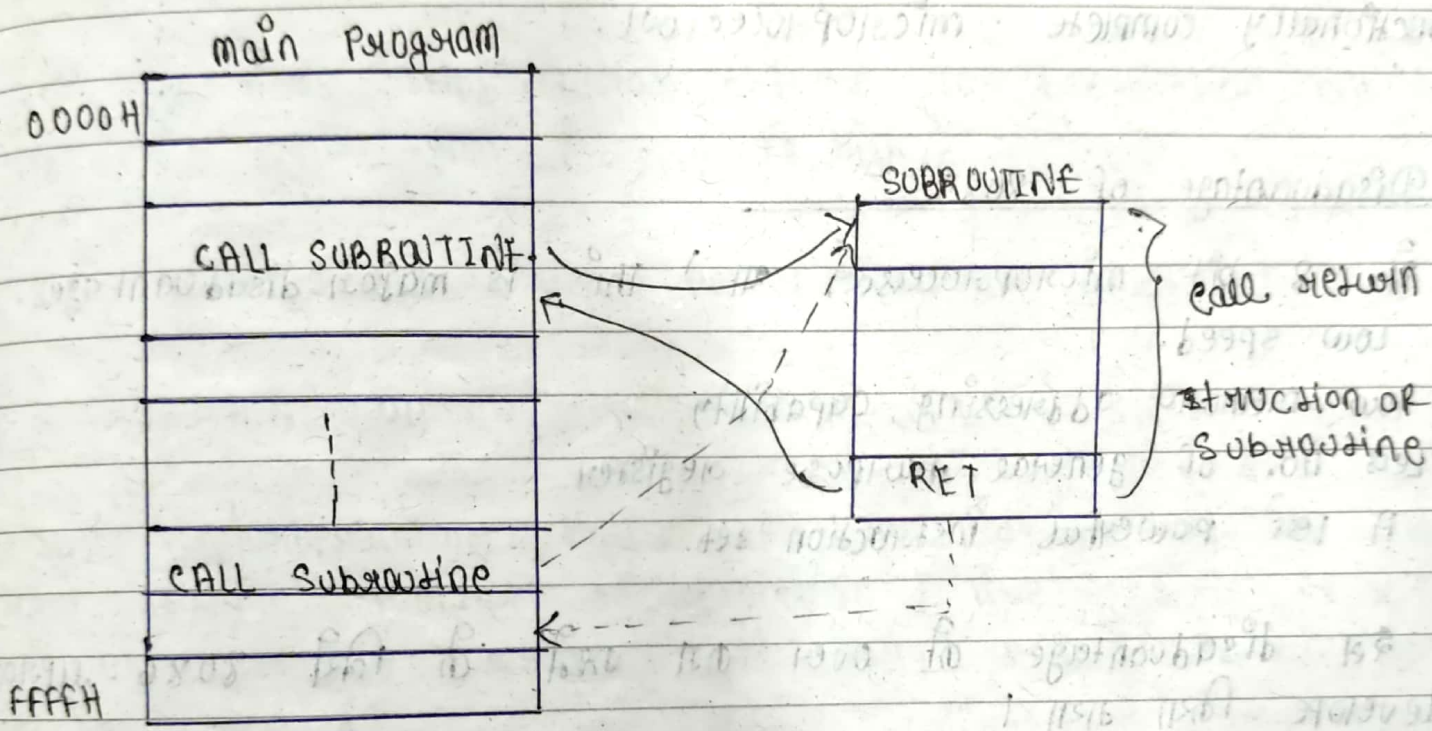
③ main program के अंदर कई बार subroutine का use किया जाता है इसलिए CALL instruction का use करते है।

④ किसी भी instruction का last instruction (RET) Return होता है।

⑤ subroutine के complete होने के बाद main program का

उस instruction से start होता है जो CALL instruction के बाद लिखा होता है।

किसी भी program में एक से ज्यादा subroutine हो सकते है। subroutine के concept से memory save होती है।



Introduction Microprocessor 8086 :-

- Intel द्वारा first 8 bit general purpose microprocessor 8080 in 1974 में invent किया गया था।
- इसके बाद microprocessor 8085 आया which contained some additional features over 8080. यह 1st processor 8085 functionally complete microprocessor.

Disadvantage of 8085 -

① ये 8 bit microprocessor था। This is major disadvantages.
② Low speed.
③ low memory addressing capability.
④ Less no. of general purpose register.
⑤ A less powerful instruction set.

⇒ इस disadvantage को over करने के लिये 8086 µprocessor develope किया गया।

Features of microprocessor 8086 -

⇒ Intel के 16-bit µprocessor family में 8086 को सबसे पहले 1978 में invent किया गया था। इस Family में 80186 व 80286 भी है।

① 8086 µprocessor में 16 bit data lines होते है।

② 8086 µprocessor में 20 Address lines होते है इसे 2^{20}
$n=20$, n is no. of Add. lines.

③ 8086 माइक्रोप्रोसेसर में 2^{20} = 1Mb memory access करने की capability होती है।

④ 8086 माइक्रोप्रोसेसर में different type में clock आवृत्ति 5, 8 व 10 MHz होती है।

⑤ ये सब single semiconductor chip व IC Fabrication Nchhanel HMOS से बनाया जाता है, H Ps high.

⑥ इसमें 40 pins होते है, low power consumes करता है।

⑦ ये 360 mA on 5V consiste करता है।

⇒ 8086 माइक्रोप्रोसेसर based CPU अब design नहीं किए जाते है। इनकी study advanced माइक्रोप्रोसेसर [live core is i5, व i7] के architecture व working को easily समझने के लिए इसका use किया जाता है।

Architecture & Block Diagram of माइक्रोप्रोसेसर 8086 -

ये functional unit होते है -

8086 Architecture

BIU (Bus Interface Unit)

- Segment Register
- Instruction Pointer
- 6-byte Instruction queue.

EU (Execution Unit)

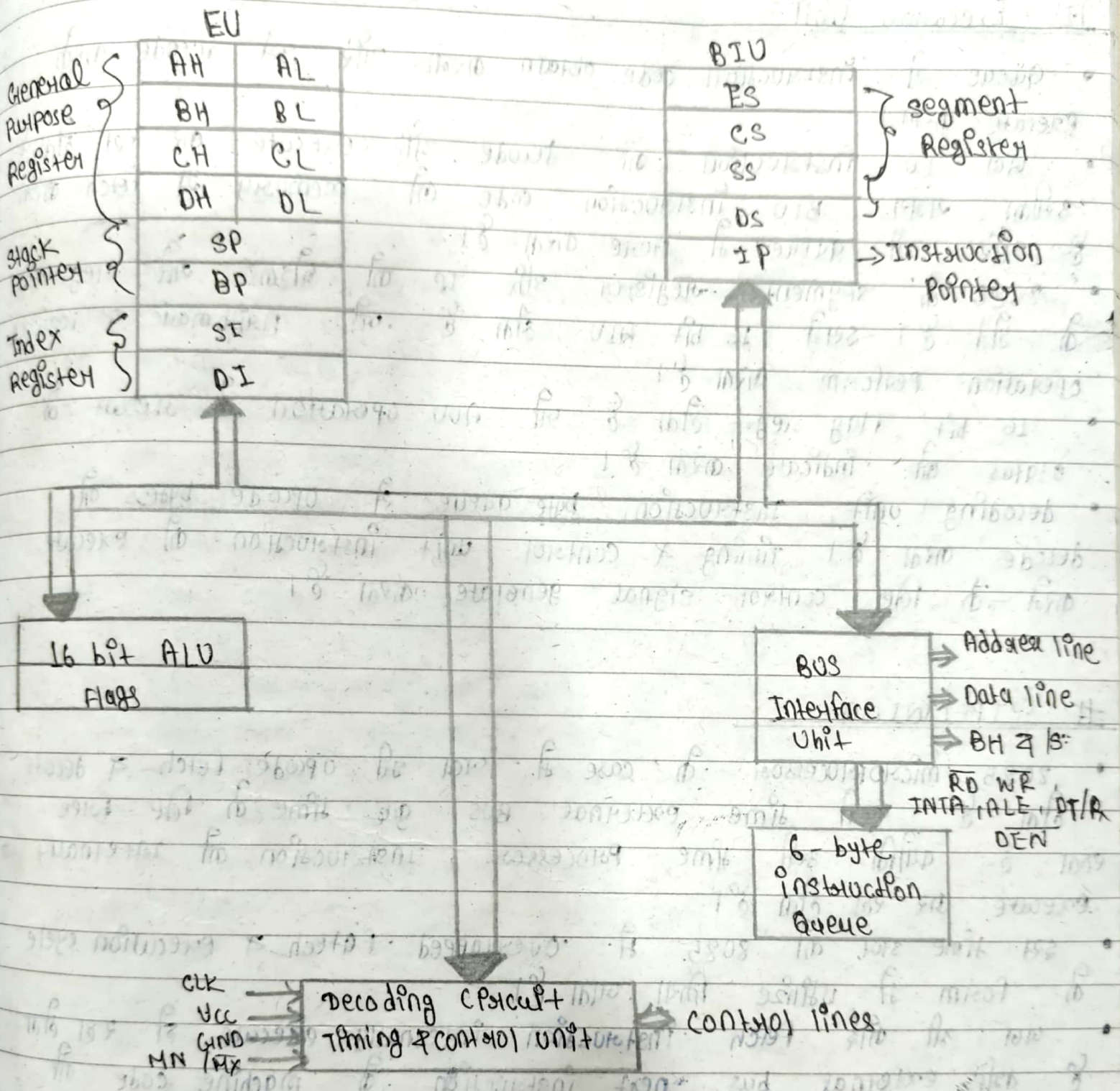
- General Purpose Register
- stack Pointer, Base Pointer, Index Reg.
- ALU flag reg & Instruction decoder.

⇒ BIU & EU, Independently एक साथ work करते है इससे processing speed fast हो जाती है।

⇒ EU, instruction को execute करता है जबकि BIU instruction को fetch करता है। Functional unit के इस type के overlapped operation pipelining कहते है।

[1] BIU [Bus Interface Unit]

- data & address को transfer करता है processor और memory I/O device के बीच।
- Address को calculate करना & send करना।
- Instruction को fetch करना और उन्हे queue मे store करना। [Queue is a group of Reg. जो FIFO (First in First out)] के principle पर work करता है।
- Memory & I/O device से data को read करना।
- Memory और I/O device पर data को write करना।
- EU से प्राप्त unrelocated operand address को relocate करना।



[2] Execution Unit -

- Queue से instruction code obtain करना और उसे decode करके execute करना।
- जब EU instruction को decode और execute कर रहा होता है उसीका समय BIU instruction code को memory से fetch करता है और उसे queue में store करता है।
- 8086 के segment registers और IP को छोड़कर सारे reg. EU के होते है। इसमें 16-bit ALU होता है जो Arithmatic & logical operation perform करता है।
- 16 bit flag reg. होता है जो ALU operation के result के status को indicate करता है।
- decoding unit, Instruction byte queue से opcode bytes को decode करता है। Timing & control unit instruction को execute करने के लिए control signal generate करता है।

PIPELINING -

- 8085 microprocessor के case में जब भी opcode fetch & decode होता है उसी time external BUS कुछ time के लिए free रहता है क्योंकि उसी time Processor, Instruction को internally execute कर रहा होता है।
- इस time slot को 8085 में Overlapped Fetch & execution cycle के form में utilize किया जाता है।
- जब भी कोई fetch instruction internally execute हो रहा होता है तभी external bus, next instruction के machine code को fetch करता है और इसे एक Predecoded instruction byte queue में arrange करता है।
- Instruction byte queue, 6 byte का FIFO structure होता है। queue से instruction को decode करने के लिये sequentially भेजा जाता है।

• जब भी कोई byte [opcode] decode होता है तब यह queue इस byte को बाहर भेजकर अपने आप को rearrange कर लेता है और इसके बाद queue के status को check किया जाता है कि क्या इसके पास कोई और opcode available है या नही।

• जब opcode को BIU (Bus Interface Unit) fetch करता है उसी समय EU previously decoded instruction को internally execute कर रहा होता है। इस तरह BIU, EU के साथ एक pipeline form करता है। इस concept को 8086 मे pipeline कहा जाता है।

28/4/23

Unit-04

Comparision of I/O mapped I/O & Memory mapped I/O

S.NO.	Parameter	Memory mapped I/O	I/O mapped I/O
(1)	Device Address	इस method में किसी भी input output device को एक memory की तरह treat किया जाता है और उसे 16 bit address का use करके identify किया जाता है।	इस mapping में किसी भी I/O device को एक separate 8 bit port address का use करके identify किया जाता है।
(2)	Address space	इस mapping में I/O address space और system memory space memory map को share करते है।	इस mapping में I/O address space और system memory space separately present होते है और वे memory map में share नही करते है।
(3)	Max. no of input output device connection possible	For a 16 bit system 2^{16} = 65,536 input-output device can be connected	For 8 bit system $2^8 = 256$ I/O devices can be connected.
(4)	Control signal required to identify to device	MEMR व MEMW control signal का use करके I/O device से communicate किया जाता है।	IOR एवं IOW control signal का use करके device से communicate किया जाता है।
(5)	Instruction available	Memory related सभी instruction available होते है। जैसे - STA, LDA, ADD, SUB	Input-output based instruction available रहते है। जैसे - IN OUT.
(6)	Data Transfer	इस mapping technique में I/O device व processor register के बीच data transfer possible होता है।	इस mapping technique में I/O device व accumulator के बीच ही data transfer possible है।

7	Arithmetic Logical operation	इस mapping technique में directly arithmetic व logical operation perform किए जा सकते है।	I/O device के data के साथ directly arithmetic व logical operation perform नही किए जा सकते है। Arithmetic व logical operation के लिए पहले I/O के data को accumulator में save करना पड़ता है।
8	Hardware requirement for decoding	memory mapped I/O method में decoding के लिए ज्यादा व complex hardware की need होती है।	इस technique में decoding के लिए कम hardware की need होती है।
9	Execution speed of instruction	13T state for LDA STA ₹ states for MOV M,R for memory mapped I/O	10T state for IN ₹ OUT instruction in I/O mapped I/O
10	Program memory	चूकि इसमे I/O device को memory की तरह treat किया जाता है इसलिए programming के लिए available memory कम हो जाती है।	इसमे complete system memory को programming के लिए use किया जा सकता है।

Input - Output Interface -

Input - Output interface internal storage और externally connected device के बीच information transfer करने का एक method है computer से connected peripheral external device को CPU से interface करने के लिए special communication links की need पड़ती है। communication link की need इसलिए पड़ती है ताकि peripheral और computer के बीच के difference को reduce किया जा सके।

Peripheral और computer के बीच कई difference होते है—

(a) peripheral electromechanical / electromagnetic device होते है जबकि computer electronic device है।
(b) peripheral का data transfer rate computer के transfer rate से कम होता है।
(c) peripheral का data codes/ format computer के word format से अलग होता है।
(d) computer से connected सभी peripheral का operating modes एक दूसरे से अलग होता है। इसलिए इन सभी peripheral को ऐसे control किया जाना चाहिए ताकि वे एक-दूसरे के working को disturb ना करे।

इन सभी differences को solve करने के लिए computer system में एक special hardware componet होता है जो CPU और peripheral के बीच होने वाले सभी input और output transfer को supervise और synchronize करता है इन्ही components को interfacing unit कहा जाता है क्योंकि ये processor bus और peripheral के बीच interface provide करता है।

Interface का अर्थ system के किन्हीं दो part के बीच point of contact से है interface का मतलब दो या दो से ज्यादा components या system को आपस मे जोड़ने से है। जिससे इनके interface points के द्वारा उनके बीच data transfer हो सके।

किसी input - output device को CPU से attach करने के लिए input - output circuit को device और system bus के बीच रखा जाता है। Input - output interface के मुख्य कार्य है -

(a) Data Conversion - इसका मतलब analog और digital data conversion और serial और parallel data transfer के बीच conversion से है।

(b) Synchronization - इसका मतलब CPU और peripheral के operating speed की matching से है।

(c) Device selection - इसका मतलब एक queue के अनुसार I/O device के selection से है।

यह diagram एक processor और peripheral के बीच एक communication link को show करता है।

Figure - Connection of Input-Output device Interfacing to processor

Input - Output Processor - (IOP)

Input-output processor एक ऐसा processor होता है जिसके पास direct memory access capability होती है जिससे कि वह input-output device के साथ communication कर सके।

हर एक interface के द्वारा CPU के साथ communication करने से ज्यादा efficient और convenient यह होता है कि CPU एक internal processor को select कर ले कि सभी I/O device के साथ communication कर सके। इस type के computer system में एक memory unit एक CPU और एक या ज्यादा Input-Output processor (IOP) होते है। इन IOP का main work Input-Output transfer task की responsibility से CPU को free करना होता है।

I/O Processing -

IOP CPU की तरह ही एक processor होता है जिसे specially I/O processing के लिए design किया जाता है। IOP अपने instructions को specially I/O operations करने के स्वंय ही fetch और execute कर सकता है। IOP instructions को specially I/O operation करने के लिए design किया जाता है। IOP I/O processing operations के अलावा कुछ और operations जैसे - arithmetic, logic, branching एंव code translation operation भी perform कर सकता है।

इस block diagram में computer system को दो processors (CPU & IOP) के दिखाया गया है, जिसमे memory unit central में है जो दोनो units से direct memory access method से communicate कर सकता है। इसमे

(i) किसी भी computational task का solution provide करने के लिए, जरूरत पड़ने वाले data की processing की जिम्मेदारी CPU की होती है।

(ii) IOP, memory unit और विभिन्न peripheral devices के बीच data transfer के लिए path provide करता है।

CPU usually I/O program को initiate करता है जिसके बाद IOP CPU से independently work करता है और external devices और memory के बीच data transfer करता है।

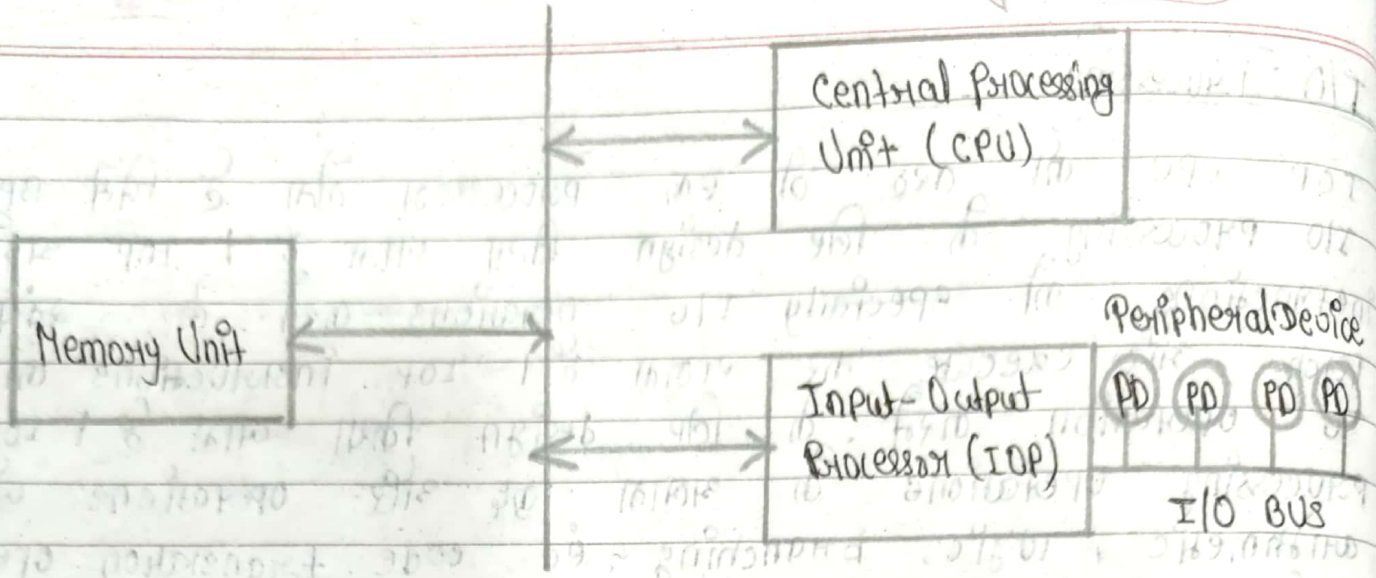


Fig - Block diagram of Computer with I/O Processor

IOP का main work memory और peripheral device के बीच data transfer का coordination करना होता है। peripheral devices, CPU और memory सभी के अपने-अपने अलग data formats होते है। IOP peripheral के data को memory मे transfer करने से पहले, इस data को memory के word format मे change करता है। peripheral device और IOP के बीच data transfer peripheral device की rate और bit capacity के हिसाब से होता है। जब IOP के पास peripheral device का data gather / collect हो जाता है तो IOP cycle stealing technique से उसे directly memory मे place कर देता है।

DIRECT MEMORY ACCESS (DMA) -

DMA एक data transfer method होता है जिसमे memory operations को speed up (fast) करने के लिए CPU को bypass करके, किसी Input - output device को directly memory के साथ data send या receive करने के लिए allow करते है।

इस process को एक chip के द्वारा manage किया जाता है जिसे DMA controller कहते है। DMA, faster processing को allow करता है क्योंकि जब कोई Input-output devices memory से data access या memory से communicate कर रहा होता है, उसी time CPU कोई दूसरा task perform कर रहा होता है।
e.g. वैसे कई ऐसे Hardware system है जो DMA का use करते है जैसे disk drive controller, Graphics card, Network cards, sound card etc.

DMA Transfer Operation / DMA Process / Working of DMA

यह figure show करता है कि जब DMA का use किया जाता है तो disk read operation कैसे होता है या दूसरे शब्दो मे memory write operation कैसे perform किया जाता है।

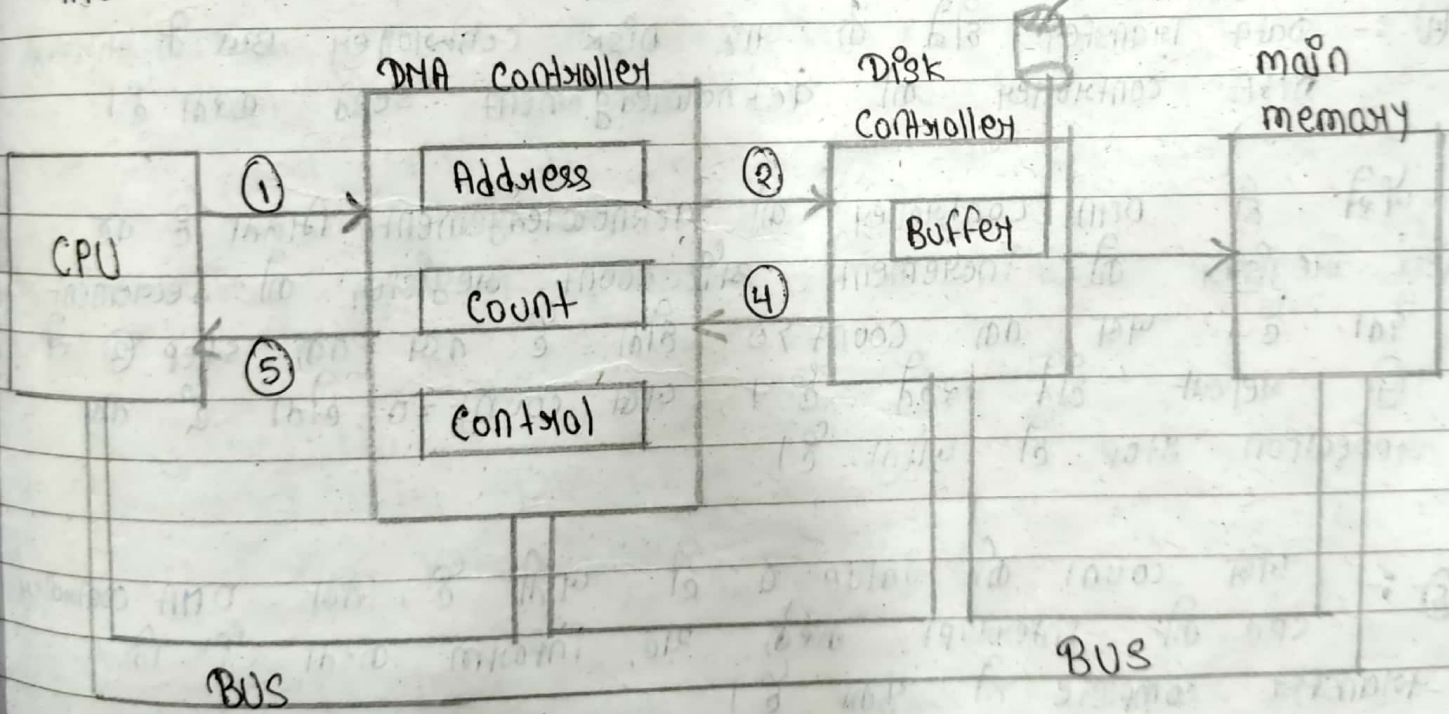


Fig- Operation of DMA Transfer

Step ① :- CPU, DMA controller के addressing register और count register को setting करके DMA controller को program कर देता है। जिससे DMA controller को यह information मिलती है कि किसे memory location से data transfer करना है और कितने byte का data transfer करना है।

Step ② :- Data transfer initiate करने के लिए DMA controller, Bus के through Disk controller को read request भेजता है ताकि Disk controller Disk से data को read करके Buffer में रख दे।

Step ③ :- Disk controller, Buffer में available data को memory को send कर देता है।

Step ④ :- Data Transfer होने के बाद Disk controller Bus के through DMA controller को acknowledgement send करता है।

जैसे ही DMA controller को acknowledgement मिलता है वह address register को increment और count register को decrement कर देता है। जब तक count > 0 होता है तब तक step ② से step ④ repeat होते रहते हैं। जब count = 0 होता है तब यह repeation stop हो जाता है।

Step ⑤ :- जब count की value 0 हो जाती है, तब DMA controller CPU को Interrupt करके यह inform करता है कि data transfer complete हो चुका है।

Advantage of DMA :-

① Transfer rate high होता है एक transfer के लिए कम CPU cycle की need होती है।
② जब memory और peripheral के बीच large data transfer होता है, तब यह एक efficient transfer method होता है।

Mode of Transfer

Binary information जो किसी external device से receive होती है, इसे usually memory unit में store किया जाता है। CPU से जब information किसी external device को भेजी जाती है तो वह भी memory unit से ही originate होती है। CPU generally memory को process करता है - source और target दोनों memory unit होते है।

Data transfer central computer और I/O devices के बीच कई modes में हो सकता है। कुछ modes में CPU को intermediate path की तरह use किया जाता है जबकि कुछ में data directly to और from memory unit transfer होता है।

Data transfer to/from peripherals तीन possible mode में हो सकता है।

① Programmed I/O
② Interrupt - initiated I/O
③ Direct Memory Access (DMA)

① Programmed I/O :-

इस case में I/O device का memory unit तक direct access नहीं होता। Device से memory तक data transfer करने के लिए CPU को कई instructions execute करनी पड़ती है - जैसे input

Instruction to transfer data from device to CPU और फिर store instruction to transfer data from CPU to memory.
Programmed I/O में CPU continuously program loop में रहता है जब तक कि I/O unit indicate नहीं करता कि वह data transfer के लिए ready है। ये एक time-consuming process है क्योंकि यह CPU को busy रखता है। इस problem को interrupt facility से avoid किया जा सकता है।

Programmed I/O Example -

Programmed I/O mode में data transfer (I/O से memory या vice versa) CPU द्वारा कुछ instruction के execution से होता है। नीचे दिए गए steps एक data type byte के transfer के लिए follow होते है।

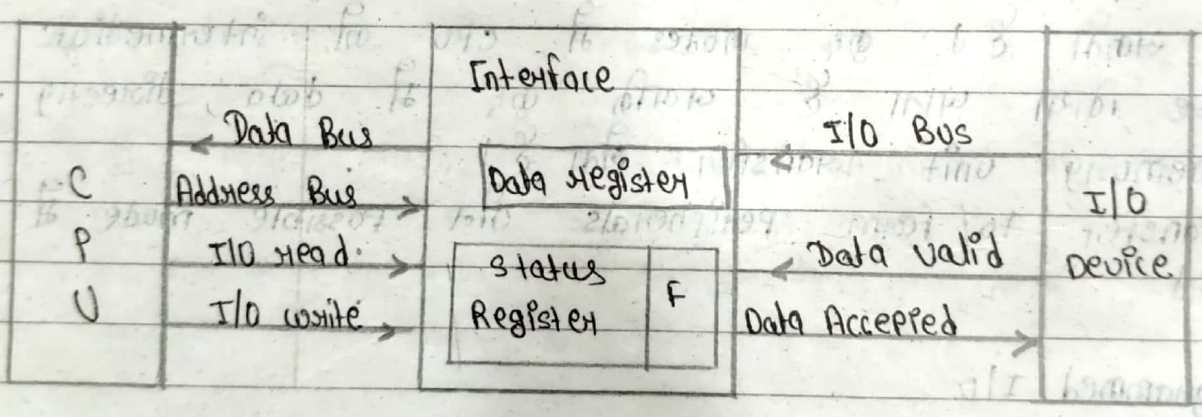


Fig - Interface used between IO and CPU

(1) जब device ready होता है और data available होता है, तब वह data को I/O bus पर रखता है और data valid line को enable करता है।

② Interface data byte को data register में accept करता है और data accepted line को enable करता है।
③ Interface status flag को 1 पर set करता है और data valid line को disable कर देता है।
④ Device data accepted line को disable करता है और अगला transfer करने के लिए ready हो जाता है।
⑤ CPU status register को read करता है और continuously check करता है कि I/O ने flag set किया है या नहीं।
⑥ अगर flag set है, तो CPU data register से data को read करता है नहीं तो step 5 repeat करता है।

Disadvantage -

- slow speed
- CPU का critical resource time waste होता है।

Advantages of Programmed I/O -

- simple और easy interface provide करता है।
- Peripheral devices को process में involve किए बिना, CPU से direct interact करता है।

Disadvantages of Programmed I/O -

- synchronization CPU time के हिसाब से efficient नहीं है क्योंकि device ready होने तक CPU को loop में wait करना पड़ता है।
- CPU idle रहता है और कोई दूसरा task handle नहीं कर सकता जब तक I/O device processing complete नहीं करता।

② Interrupt Initiated I/O :-

इस method मे, एक Interrupt command का use device को data transfer की start और end के बारे मे inform करने के लिए किया जाता है। इस दौरान CPU कोई और program execute करता रहता है। जब interface detect करता है कि device data transfer के लिए ready है तब वह एक interrupt request generate करता है और इसे CPU को भेजता है। जब CPU को यह signal मिलता है तो वह temporarily अपना current program रोक देता है और एक service program पर switch करता है जो I/O transfer process करता है। Transfer complete होने के बाद, CPU वापस original task पर लौट जाता है। CPU interrupt signal का response इस तरह देता है। वो return address को memory stack मे store करता है। फिर control एक service routine पर चला जाता है जो requested I/O operations perform करती है।

There are two method for accomplishing this :

- Vectored Interrupt - इसमे interrupt source CPU को खुद ही branch address भेजता है।
- Nonvectored Interrupt - इसमे branch address एक fixed memory location पर assign होता है।

Interrupt I/O :- यह mode programmed I/O की disadvantage को overcome करता है। Programmed I/O मे CPU को बार-बार check करना पड़ता है जबकि इस method मे device खुद CPU को interrupt signal भेजता है जब वो ready होता है और data transfer करना होता है।

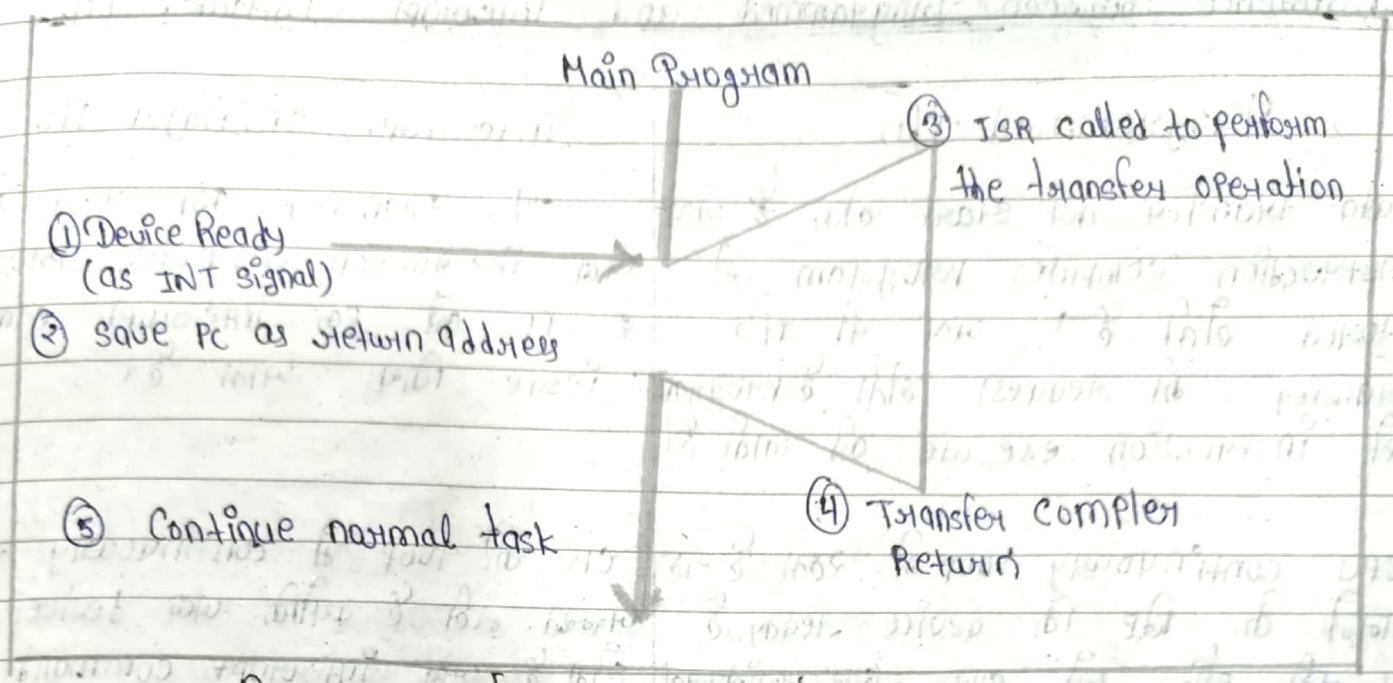


Figure - Flow of the interrupt system

Steps

① जब device (interface) ready होती है, वो अपनी data register में flag set कर देती है।
② flag set होने पर Interface एक interrupt signal CPU को भेजता है।
③ Processor return address (PC द्वारा दिया गया) को stack में store करता है।
④ एक ISR call होती है जो transfer operation perform करती है।
⑤ Transfer complete होने के बाद common stack से PC को POP करके main program पर लौटा देता है।
⑥ अब CPU अपने main program को normal तरीके से execute करता है।

Difference Between Programmed and Interrupt Initiated I/O

Programmed I/O	Interrupt Initiated I/O
Data transfer तब start होता है जब instruction computer program में stored होती है। जब भी I/O transfer की request होती है program से instruction execute की जाती है।	I/O transfer तब start होता है जब ~~instruction computer program~~ & CPU को एक interrupt command issue किया जाता है।
CPU continuously loop में रहता है यह जानने के लिए कि device ready है या नही और उसे बार-बार peripheral device को monitor करना पड़ता है।	CPU को loop में continuously रहने की जरूरत नही है क्योंकि जब device ready होता है जब interrupt command CPU को interrupt कर देता है।
इससे CPU cycles की wastage होती है क्योंकि CPU बेकार मे busy रहता है और system की efficiency reduce हो जाती है।	CPU cycles waste नही होती है क्योंकि CPU इस time मे दूसरे काम करता रहता है इसलिए यह method ज्यादा efficient है।
जब तक transfer complete नही हो जाता, CPU कोई और काम नही कर सकता क्योंकि उसे continously peripheral device को monitor करना पड़ता है।	CPU कोई और काम कर सकता है जब तक कि दूसरे उसे interrupt करके device के ready होने की information न दी जाए।
इसका module एक slow module को बहुत treat किया जाता है।	इसका module Programmed I/O module से fast होता है।

इसे programm और समझना काफी easy होता है।	अगर low level language use की जाये तो यह tricky और complicated हो सकता है।
system की performance काफी ज्यादा degrade हो जाती है।	system की performance कुछ हद तक enhance हो जाती है।

Asychronous Data Transfer

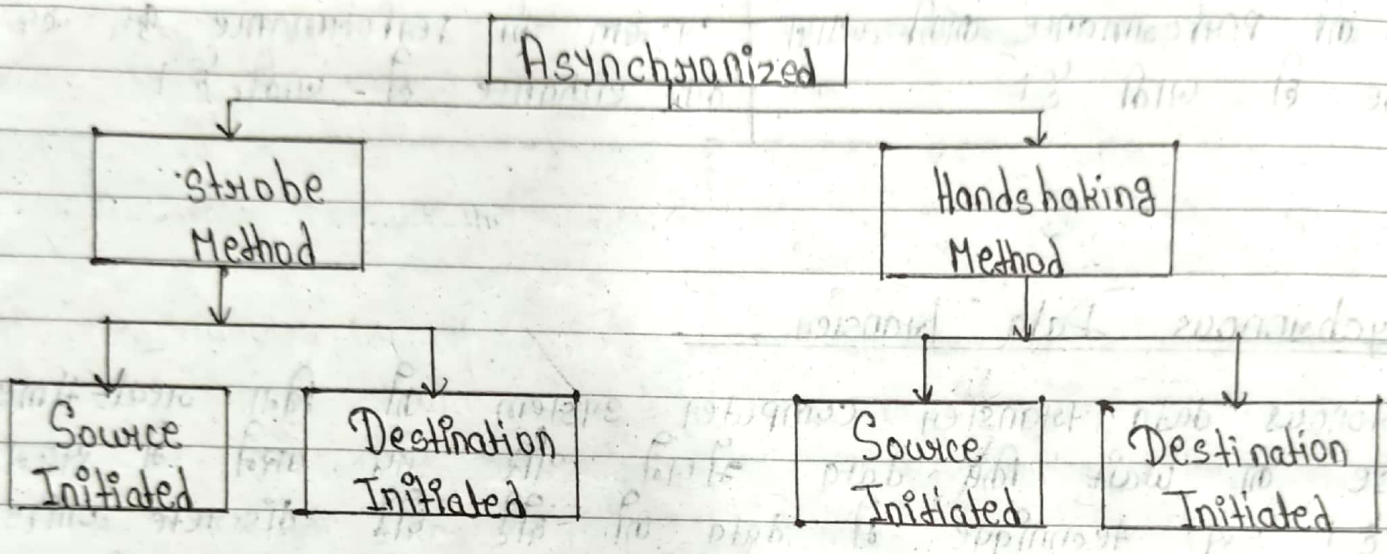
Asychronous data transfer computer system को बिना real-time response का wait किए data भेजने और प्राप्त करने में सक्षम बनाता है। इस technique में data को छोटे छोटे discrete units (Packet) में भेजा जाता है, जिन्हे अलग अलग handle किया जा सकता है।

Terminologies used in asychronous data transfer -

- Sender - वह machine या gadget जो data transfer करता है।
- Receiver - वह device या computer जो data receive करता है।
- Packet - Data का एक discrete unit जो transmit या receive किया जाता है।
- Buffer - short-term storage जो incoming या outgoing data को temporarily hold करता है।

Classification of Asynchronous Data Transfer

- Strobe Control Method
- Handshaking Method



Strobe Control Method For Data Transfer

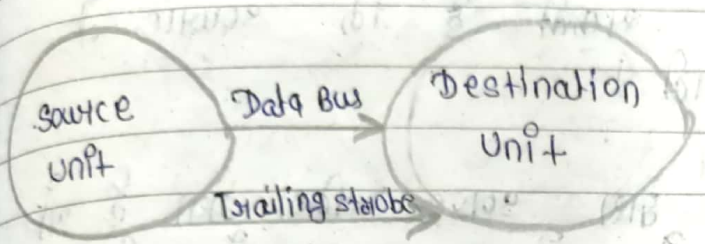
strobe control asynchronous data transfer की एक method है जो दो devices के बीच data flow को synchronize करती है। Bits को एक-एक करके भेजा जाता है और यह एक दूसरे से independent होते है इसमे किसी clock signal का use नही होता (asynchronous Commonication)। Data सही से receive करने के लिए receiving device को transmitting device के साथ synchronize होना चाहिए।

Strobe control मे data के साथ एक अलग signal भेजा जाता है जिसे Strobe signal कहा जाता है।

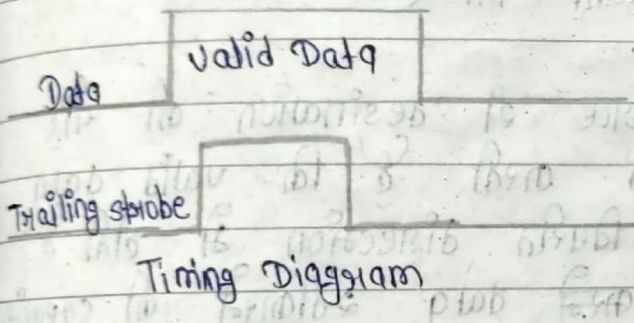
यह signal receiver को बताता है कि data valid है और उसे read किया जा सकता है। Receiver तब तक data read नही करता जब तक Strobe signal न आ जाए Strobe signal को आमतौर पर transmitter device generate करता है। यह signal data से पहले भेजा जाए तो इसे Leading Strobe कहते है। अगर यह

signal data के बाद भेजा जाए तो इसे Trailing strobe कहते हैं।

Source Initiated strobe

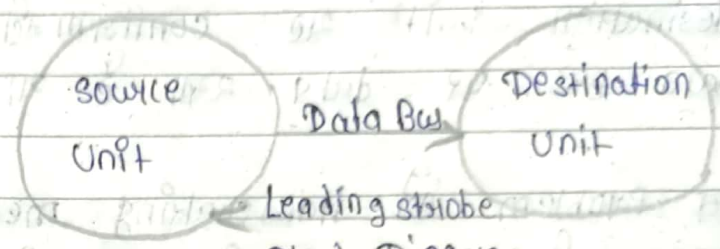


Block Diagram

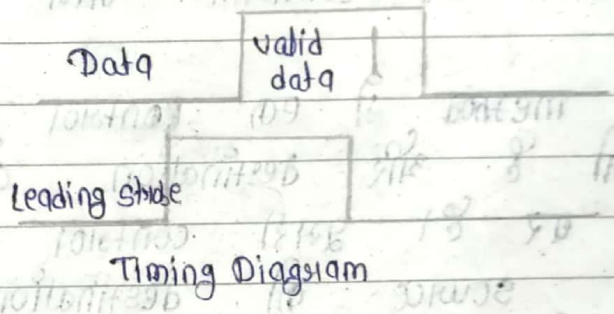


Timing Diagram

Destination Initiated strobe



Block Diagram



Timing Diagram

Types of strobes

strobes control का use करना फायदेमंद है क्योंकि यह asynchronous data transfer को possible बनाता है जो तब useful होता है जब communicating devices की clock rates अलग-अलग हो या synchronized न हो। Data transfer का timing भी strobe control की वजह से ज्यादा flexible होता है क्योंकि receiving device को transmitting device की clock के साथ synchronize होने की जरूरत नहीं होती। वो बस strobe signal का wait कर सकती है और फिर data को read कर सकती है। strobe control, जो कि बहुत से electronic devices और systems में commonly use होता है, एक useful technique है जिससे asynchronous communication में reliable data flow सुनिश्चित करने की एक useful technique है।

② Handshaking Method For Data Transfer :-

Strobe method की एक limit है कि start करने वाला source unit यह confirm नही कर सकता कि destination unit bus पर रखे गये data को receive कर लिया है या नही। इसी तरह destination unit यह confirm नही कर सकती है कि source ने data Bus पर data रखा है या नही।

इस problem को Handshaking method द्वारा solve किया गया है जो एक दूसरी control signal line को जोड़कर units के बीच transfer की confirm करने की अनुमति देती है।

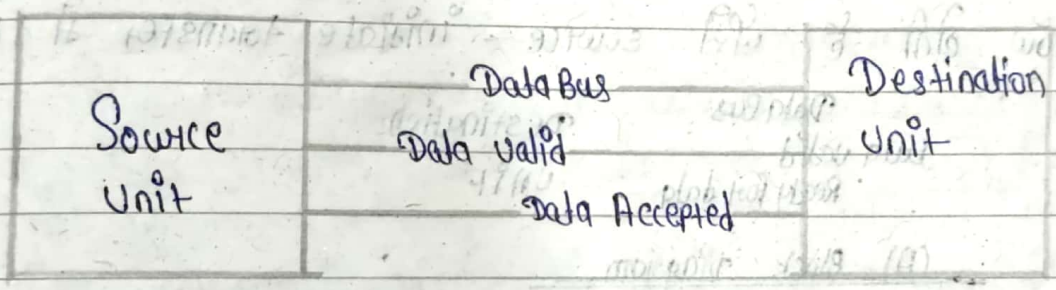
इस method मे एक control line source से destination की ओर जाती है और destination को सूचित करती है कि valid data Bus पर है। दूसरी control line विपरीत दिशा मे होती है और source को destination को करने data स्वीकारने की capacity के बारे मे बताती है।

यह control unit पर depend करता है कि transfer किसके द्वारा आरम्भ होता है- source या destination, asynchronous data transfer के दौरान दो device handshaking द्वारा अपने communication का management करते है। यह सुनिश्चित करता है कि transmitting और receive device data send और receive करने के लिए ready है।
synchronous communication के विपरीत इसमे clock signal की need नही होती।

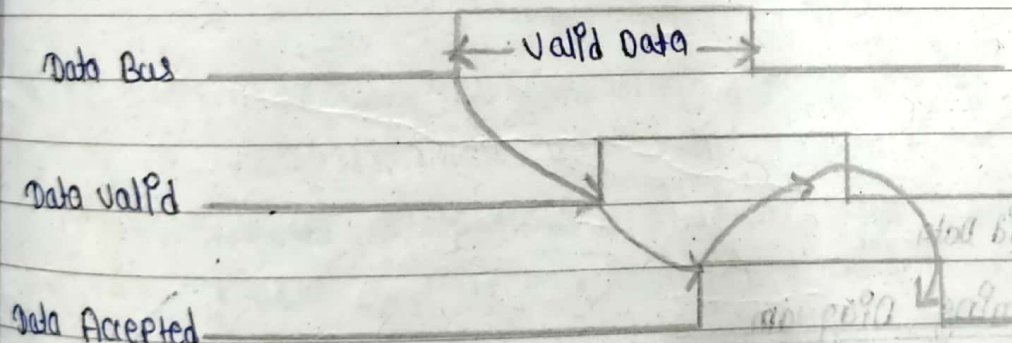
Source - Initiated Handshaking

इस प्रकार की handshaking में data transfer के दौरान दो units के बीच signal exchange दिखाया गया है। source unit bus पर data रखता है और 'data valid' signal सक्षम करता है। फिर destination unit 'data accepted' signal को achive करती है जिससे data स्वीकारने की confirm होती है इसके बाद source unit 'data valid' को अक्षम करता है।

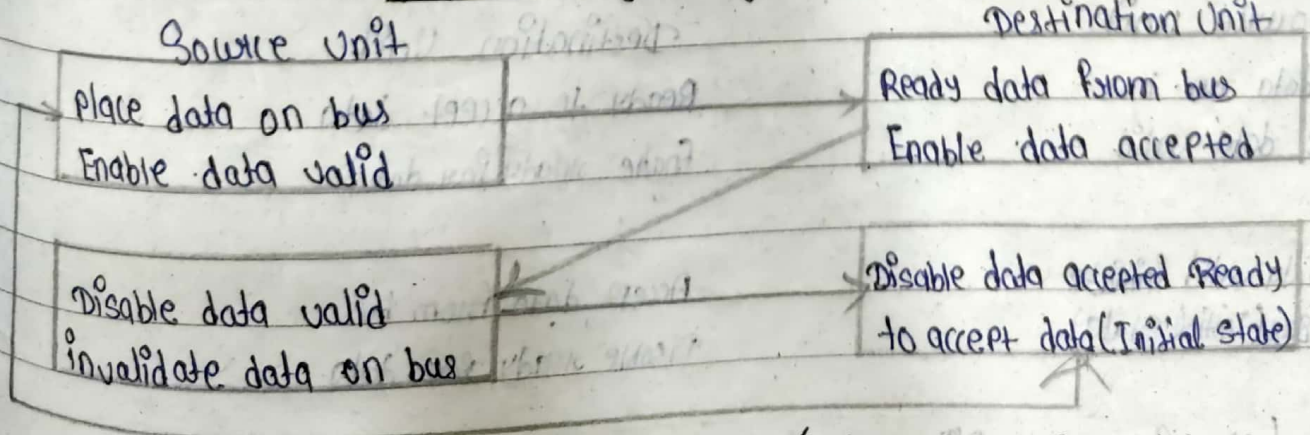
अगला data भेजने से पहले destination unit 'data accepted' को NULL करके तैयार रहने का संकेत देती है। यह sequence (block diagram, timing diagram और sequence diagram) में दिखाया गया है।



(a) Block Diagram



(B) Timing Diagram

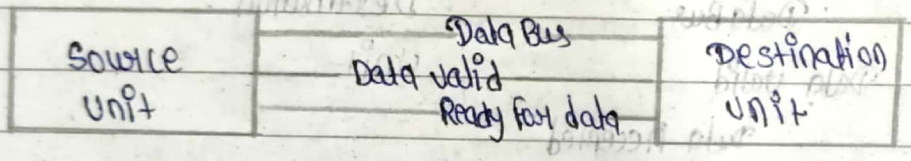


(C) Sequence Diagram (sequence of events

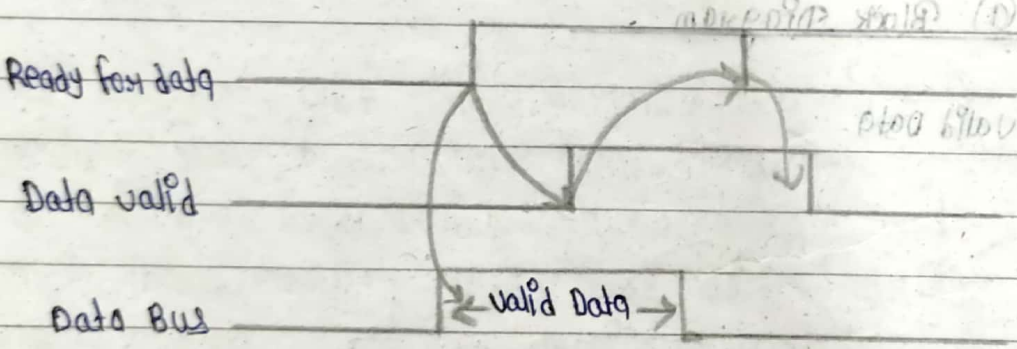
Destination Initiated Handshaking :-

Block diagram मे दो handshaking signal होते है - एक "data valid" जो source unit से आता है और दूसरा "ready for data" जो destination unit से आता है। यहा "data accepted" को rename करके "ready for data" कहा गया है ताकि इसका updated meaning reflect हो सके।

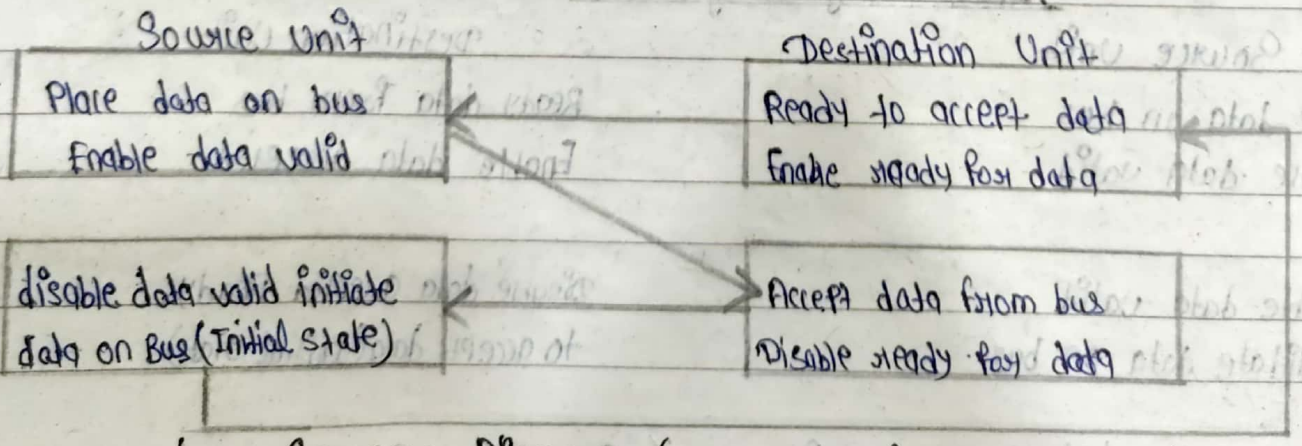
Destination Initiate Transfer मे source तब तक wait करता है जब तक उसे destination से "ready for data" signal नही मिल जाता। जब destination ready signal भेज देता है तब source data को bus पर रखता है। इसके बाद handshaking process उसी sequence मे follow होती है जैसे source - initiate transfer मे होती है।



(A) Block Diagram



(B) Timing Diagram



(C) Sequence Diagram (sequence of events)

Advantage of Asynchronous Data Transfer -

① Asynchronous data transfer data को छोटे - छोटे, independently process होने वाले parts मे भेजता है जिससे data transfer speed तेज हो जाती है

② ये method synchronous transfer से ज्यादा effective है क्योंकि receiver को हर बार respond करने की need नही होती।

③ बड़े files या datasets को छोटे packet मे divide करके parallel भेजने से transmission का total time कम हो जाता है।

Disadvantage of Asynchronous Data Transfer -

① Asynchronous data transfer मे programming थोड़ी complex होती है और possibility रहती है कि कुछ data corrupt या lost हो सकता है अगर packet सही order मे ना पहुंचे या transmission के दौरान खो जाए।

② real-time communication नही होता, इस वजह से synchronous transfer के मुकाबले ज्यादा easy-prone हो सकता है।

RISC और CISC दो computing system है जो computer के लिए develop किए गए है। Instruction set या instruction set Architecture (ISA) एक computer की structure होती है जो उसे command देती है कि data को process और manipulate कैसे करना है। Instruction set मे निम्न शामिल होते है - Instructions, Addressing modes, Native data types, registers, interrupt और exception handling, memory architecture

Instruction set को software को interpreter की help से emulate किया जा सकता है। Instruction set architecture को software और hardware के बीच की limit के रूप मे माना जाता है।

Comparision Between RISC and CISC

Parameters	RISC	CISC
Acronym	It stands for 'Reduced Instruction set computer'.	It stands for 'Complex Instruction set computer'.
Definition	RISC processor में smaller set of instruction होते है और कम addressing mode होते है।	CISC processor में larger set of instruction होते है, और कई addressing mode होते है
memory unit	इसमे memory unit नहीं होते और एक अलग hardware होता है instruction implement करने के लिए।	इसमे memory unit होता है complex instruction implement करने के लिये।
Program	इसमे hard-wired programming unit होता है।	इसमे micro-programming unit होता है।
Design	इसका compiler design complex होता है।	इसका compiler design easy होता है
Calculation	calculation faster और precise होती है।	calculation slow और precise होती है।
Decoding	Instruction decoding simple होता है	decoding complex होता है।
time	Execution time बहुत कम होता है।	Execution time बहुत ज्यादा होता है।
External	इसे external memory की need नही होती।	इसे external memory की need होती है।

memory	memory calculations के लिये use होती है।	calculation के लिए memory use होती है।
pipelining	pipelining function ठीक से work करता है।	pipelining function ठीक से work नहीं करता है।
stalling	processor में stalling कम होती है	processor अक्सर busy हो जाता है।
code expansion	code expansion problem नहीं है।	code expansion एक problem हो सकता है।
Diskspace	space ज्यादा waste होता है।	space save होता है।
Application	Low end applications में use होता है जैसे - security system, home automation etc.	High end application में use होता है जैसे video & processing tele communication और image processing

CISC Vs RISC (Additional Points)

CISC	RISC
① Hardware पर ज्यादा emphasis	Software पर ज्यादा emphasis
② Multiple instruction size और formats	same set instruction with few formats
③ कम registers	ज्यादा registers
④ ज्यादा addressing mode	कम addressing mode
⑤ Extensive microprogramming use होता है।	compiler में ज्यादा complexity होती है
⑥ Instruction को execute करने का time ज्यादा करता है।	हर instruction cycle में time होती है।
⑦ Pipelining difficult है।	pipelining easy है।

Unit - 05

Memory Hierarchy

एक computer system में कई तरह के memory devices होते है। इन memory types को organize करने का तरीका जिससे access time कम हो जाता है, उसे memory hierarchy कहते है। Computer storage को response time के आधार पर hierarchy में बांटा जाता है।

Memory hierarchy levels है :-

- CPU register
- Cache memory
- Primary memory या main memory
- Secondary memory या magnetic disk
- Optical disk या magnetic tape या tertiary memory

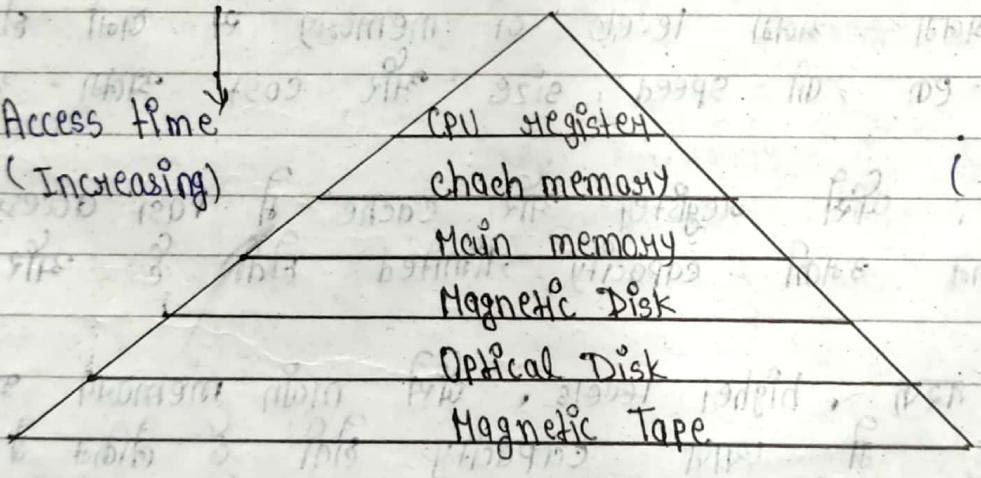


Fig :- Memory Hierarchy

- Memory hierarchy एक computing system में विभिन्न type के storage को उनकी access speed के आधार पर organize करने का तरीका है। यह response time के अनुसार computer storage को organize करता है।

- चूंकि response time, complexity और capacity आपस में जुड़े हुए है, लेकिन उन्हे उनके performance और controlling technologies के आधार पर अलग किया जा सकता है। जैसे कि ऊपर दिए गए diagram में दिखाया गया है, computer memory की एक pyramid-like structure होती है।
- इसका use memory के different levels को समझाने के लिए किया जाता है। यह hierarchy के आधार पर computer storage को अलग करता है।

Why is Memory Hierarchy used in Systems?

- Memory hierarchy का use computer system में उपलब्ध memory resource के use को optimize करने के लिए किया जाता है।
- यह hierarchy अलग-अलग levels of memory से बनी होती है, जिनमे से हर एक की speed, size और cost अलग-अलग होती है।
- Lower levels, जैसे register और cache में fast access time होता है, लेकिन उनकी capacity limited होती है और वे महंगे होते है।
- वही दूसरी तरफ, higher levels, जैसे main memory और secondary storage में ज्यादा capacity होती है लेकिन वे slow और कम खर्चीले होते है।
- Frequently accessed data को fast और small memory levels में store करके और less frequently accessed data को slow- larger memory level में रखने से, memory hierarchy efficient data access सुनिश्चित करता है। Processing time को कम करता है और overall system performance को बढ़ता है।

Advantage Of Memory Hierarchy :-

① Faster Access - memory hierarchy में कई levels होने के कारण, processor को बार-बार use किए जाने वाले data तक तेजी से access मिलता है।

② Cost-effective - Expensive memory का use सिर्फ जरूरत की जगह पर करने से cost कम की जा सकती है। cache memory main memory या secondary storage से अधिक मंहगा होती है। memory hierarchy की help से, इसे हर जगह use करने की बजाय, सिर्फ critical areas में use किया जाता है।

③ Efficient use of Resource - memory hierarchy की मदद से, computer resource का efficient use किया जा सकता है, जिससे wastage कम होती है।

④ Increased Capacity - memory hierarchy की help से, OS system बड़े data sets को secondary storage devices जैसे hard disk और solid-state drives (SSD) में store कर सकता है।

⑤ Increased Processing Speed - memory hierarchy की help से, fast memory का use बार-बार access किए जाने वाले data के लिए किया जाता है, जिससे operations बहुत fast से किए जा सकते है।

Memory Hierarchy Design

memory hierarchy design को दो type में divide किया गया है।

(i) Primary / Internal Memory - इसमें registers, cache memory और main memory शामिल होते है और इन्हे CPU द्वारा सीधे access किया जा सकता है।

(i) Cache Memory - यह सबसे fast type की memory होती है और यह frequently accessed data और instruction को store करती है जिससे उन्हे main memory से retrieve करने की तुलना में fast से प्राप्त किया जा सकता है।
आमतौर पर, cache memory के दो या तीन levels होते है, जहां हर स्तर की capacity पिछले स्तर से ज्यादा होती है लेकिन access speed धीमी होती है।

(ii) Registers - Register सबसे fast और small memory होती है, जो CPU के अंदर स्थित होता है।
ये CPU द्वारा frequently accessed data की छोटी मात्रा को store करते है।
ये data, instructions, और control information रखते है जिनकी CPU को operation perform करने की need होती है।
Register की storage capacity आमतौर पर 16-bit से 64-bit तक होती है।

(iii) Main Memory - यह computer system की primary memory होती है। यह cache memory से slow लेकिन secondary storage से faster होती है। Main memory में storage data और instructions को CPU द्वारा directly access किया जा सकता है।

② Secondary / External Memory -

यह Processor द्वारा I/O module के माध्यम से access की जाने वाली magnetic disks, optical disk और magnetic tapes जैसे storage device से बनी होती है।

① Secondary Storage - secondary storage जैसे hard disk का use large amount of data को long-term storage के लिए किया जाता है ; क्योंकि यह main memory में फिट नहीं हो सकता।
यह main memory से slow होता है लेकिन इसकी capacity अधिक होती है और यह persistent (non-volatile) होता है, यानि power off होने पर भी data safe रहता है।

② Magnetic Disk - magnetic disks, जैसे HDD, spinning platters पर magnetic रूप में data store करते हैं।
ये high storage capacity प्रदान करते हैं और sequential और random दोनों तरह की access को support करते हैं।
हालांकि, ये solid-state drives (SSD) की compare में काम विकास और धीमे होते हैं।
फिर भी, कुछ systems में bulk data storage के लिए इनका use किया जाता है क्योंकि ये cost-effective storage option प्रदान करते हैं।

③ Magnetic Tape - magnetic tape एक sequential data storage माध्यम है, जिसमें एक long plastic ribbon होता है जो magnetic material से coated रहता है।
यह अपनी high capacity, low cost per gigabyte और long-term durability के लिए जाना जाता है।
magnetic tape का use main रूप से archival storage और data

backup के लिए किया जाता है।
Tape पर data access करना disk drive की compare में slow होता है, लेकिन यह infrequently accessed data के लिए best option है, क्योंकि इसमें बार-बार data access नहीं किया जाता।

Auxiliary Memory (Secondary Storage)

Auxiliary memory, जिसे secondary storage भी कहा जाता है, एक प्रकार की computer memory है जो long-term storage के लिए data और programs को hold करती है।
Primary memory (RAM) के विपरीत, जो volatile और temporary होती है, auxiliary memory non-volatile होती है और computer के off होने पर भी data को retain रखती है।

Key characteristics :-

- Large capacity - RAM की compare में अधिक data store कर सकती है।
- Slower access - RAM की compare में data को retrieve करना slow होता है।
- Lower cost - प्रति storage unit की कीमत RAM से कम होती है।

Common types :-

- Hard Disk Drive (HDD) - magnetic platters का use करके data store करती है। आजकल SSD की fast speed के कारण HDD को replace किया जा रहा है।

- Solid State Drive (SSD) - Flash memory chips का use करके data store करती है। यह HDD से fast होती है लेकिन storage पर मंहगी होती है।

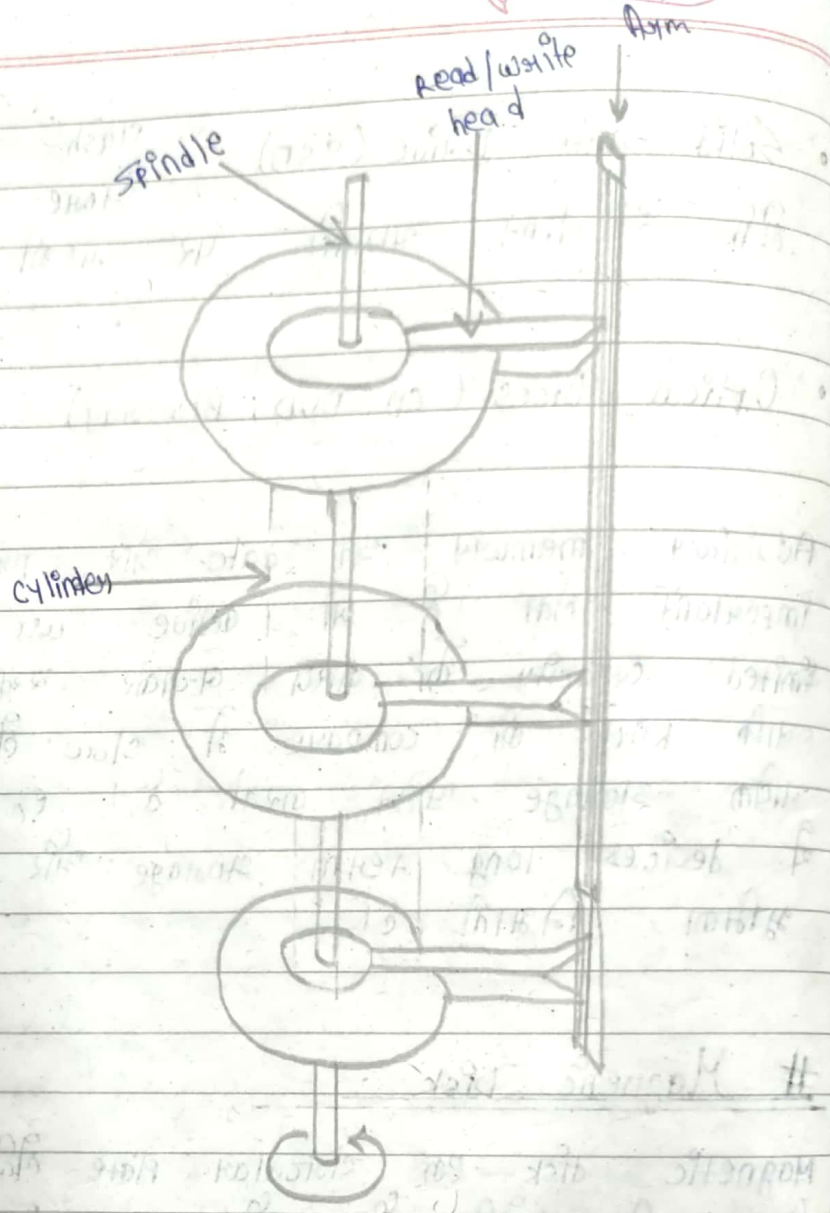
- Optical Discs (CD, DVD, Blu-ray) - Laser का use करके data को read और write करती है।

Auxiliary memory उन data और programs को store करने के लिए important होती है जो active use में नही होते लेकिन RAM की limited capacity के कारण बचाकर रखना जरूरी होता है।
हालांकि RAM की compare में slow होती है, लेकिन यह कम लागत में अधिक storage प्रदान करती है। Ex - HDDs, SSDs & Optical Discs है। ये devices long-term storage और data retrieval में important भूमिका निभाती है।

Magnetic Disk

Magnetic disk एक circular plate होती है, जो metal या plastic की बनी होती है और magnetized material से coated होती है। इनकी दोनो side का use किया जाता है, और एक से अधिक disks को एक spindle पर stack किया जाता है, जिनमें read/write heads हर surface पर मौजूद होती है।
सभी disk एक साथ high speed पर घूमती है और access के लिए बार-बार start/stop नही की जाती।

(A) A single disk



(B) A hard disk drive

Bits को magnetized surface पर concentric circle के रूप में store किया जाता है, जिन्हे tracks कहा जाता है ये tracks आगे sector में divide होते है। ज्यादातर system में, minimum quantity of information जिसे store किया जा सकता है, वह sector के रूप में होती है। एक unit में single read/write head होता है, जो प्रत्येक disk surface के लिए use किया जाता है। इस type की unit में, track address bits का use mechanical assembly द्वारा किया जाता है ताकि read/write operation से पहले head को सही track position पर move किया जा सके।

एक disk system को address bits के माध्यम से specify किया जाता है, जो disk number, disk surface, sector number और track जैसी sector को define करता है। जब read/write head को specified track position पर किया जाता है, तो system को wait करना पड़ता है जब तक कि rotating disk उस specified sector तक न पहुंच जाए read/write head के नीचे

एक track जो किसी given sector में circumference के पास होता है, वह disk के center के पास वाले track से longer होता है। अगर bits को equal density में record किया जाए, तो कुछ tracks में more recorded bits होंगे others की compare में। सभी records को equal length वाले sector में store करने के लिए कुछ disks एक variable recording density का use करते हैं जिसमें track के पास higher density होती है।

जो disk हमेशा unit assembly से attached रहती है और जिन्हे एक सामान्य user द्वारा remove नहीं किया जा सकता, उन्हे hard disk कहा जाता है।
एक disk drive में जिसमें removable disks होती है उसे floppy disk कहा जाता है। Floppy disk drive में use की जाने वाली disks होती, removable disks होती है, जो plastic की बनी होती है और उन पर magnetic recording material की coating होती है। Floppy disk का personal computer में बड़े पैमाने पर use किया जाता है, खासकर software distribution के लिए computer users को।

Magnetic Tape

magnetic tape एक लंबी और पतली plastic film होती है, जिसे magnetizable material से कोट किया जाता है। यह data storage history में एक important भूमिका निभाता है। इसे 1928 में विकसित किया गया था और इसने audio, video और computer data storage को पूरी तरह बदल दिया।

Structure of Magnetic Tape -

- **Base film** - यह एक पतली, लचीली plastic film (polyester या mylar) होती है जो tape की backbone बनाती है और magnetic coating के लिए support प्रदान करती है।
- **Magnetic Coating** - iron oxide microscopic particles की एक पतली परत base film पर जमा की जाती है। ये कण अलग-अलग दिशाओं में align किए जाते हैं ताकि digital information (0s और 1s) को दर्शाया जा सके।
- **Backing layer** - कुछ tapes में एक back-coating होती है, जो magnetic layer को wear और tear से बचाती है।

Working Principle -

magnetic tape magnetism principle पर work करता है। इसका magnetic coating छोटे iron oxide particles से बना होता है, जिसे north या south direction में magnetize किया जा सकता है। magnetic tape drive इन particles को read और write के लिए magnetic head का use करता है।

Writing

- Drive एक pulsating magnetic field उत्पन्न करता है।
- Magnetic field की दिशा (north या south) यह निर्धारित करती है कि लिखा गया data 0 या 1 होगा।
- Magnetic particles उस direction में खुद को align करते हैं और data store किया जाता है।

Reading

- Drive tape को head के सामने से move करता है।
- magnetic field में सूक्ष्म परिवर्तन के कारण tiny electric currents उत्पन्न होते हैं।
- current का direction यह बताता है कि store किया गया data 0 है या 1।
- Drive इस current को बढ़ाता है और original digital data को पुनः प्राप्त करता है।

Types of Magnetic Tape -

- Audio Cassette Tape - sound recording और playback के लिए use किया जाता है।
- Video Cassette Tape - video signals जैसे (VHS, Betamax etc) को record करने और play चलाने के लिए use किया जाता है।
- Computer Tape - पुराने computers में data storage के लिए use किया जाता था और आज भी data backups के लिए प्रासंगिक है।

Advantage of Magnetic Tape -

- High Capacity - बहुत अधिक मात्रा में data store कर सकता है, जिससे यह large data backups के लिए उपयुक्त बनता है।
- Low cost - अन्य storage options की compare में magnetic tape सस्ता होता है।
- Durability - यदि ठीक से संग्रहित किया जाए, तो दशकों तक चल सकता है।
- Offline Storage - यह offline storage प्रदान करता है, जिससे यह cyberattacks से safe रहता है।

Disadvantage of Magnetic Tape -

- Sequential Access - Data को sequential access किया जाता है, जिससे किसी specific data point तक पहुचने के लिए fast-forwarding या rewinding करनी पड़ती है, जिससे कि यह slow हो जाता है।
- Degradation - समय के साथ, यदि इसे सही तरीके से संग्रहीत नही किया गया तो खराब हो सकता है।
- Obsolescence - Tape drives और formats पुराने हो सकते हैं जिससे access करना complex होता है।

CACHE MEMORY MAPPING TECHNIQUES

cache memory का basic characteristics इसका fast access time होता है इसलिए किसी word को cache memory में ढूंढने के लिए minimum time waste होना चाहिए। वह process जिससे main memory से cache memory में data का transformation होता है उसे mapping process कहते है।

cache memory organization में तीन type के mapping techniques होती है-

(1) Associative mapping
(2) Direct mapping
(3) Set - Associative mapping

इन mapping technique को समझने के लिए हम एक specific memory organization का example लेते है जिसमे

(a) main memory की size 32 k words है जो 1 word में 12 bit का data store कर सकता है।
(b) cache memory की size 512 words की है यह भी 12 bit data store कर सकता है।

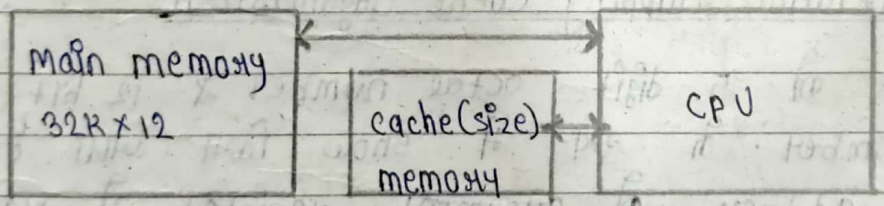


Fig :- Example of cache memory

Cache memory में stored हर एक word की एक duplicate copy main memory में होती है। CPU दोनों memories से communicate करता है। CPU पहले 15 bit address को cache memory को send करता है। यदि hit हो जाता है तो CPU इस 15 bit address से 12-bit data को accept करता है। यदि miss हो जाता है तो CPU main memory को read

करके word को, cache memory में transfer कर देता है।

(1) Associative Mapping -

यह mapping technique fastest और most flexible mapping technique है जो associative memory का use किया जाता है। Associative memory में memory word के address और content दोनो को store किया जाता है।

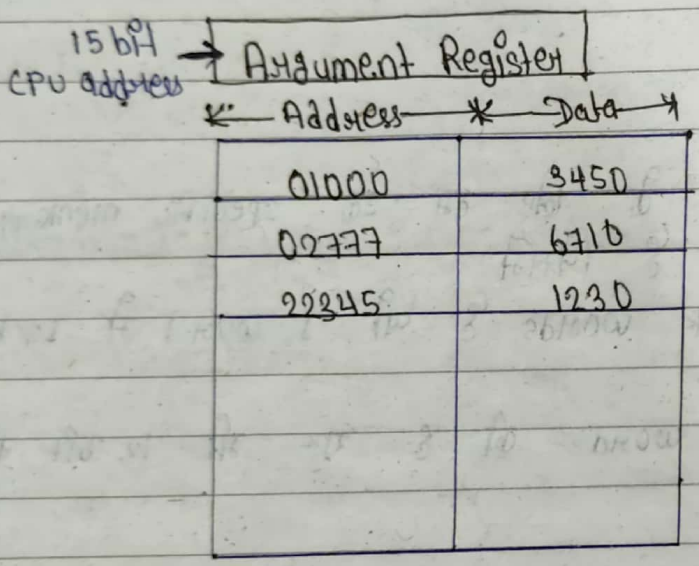


Fig :- Associative Mapping Cache Organization

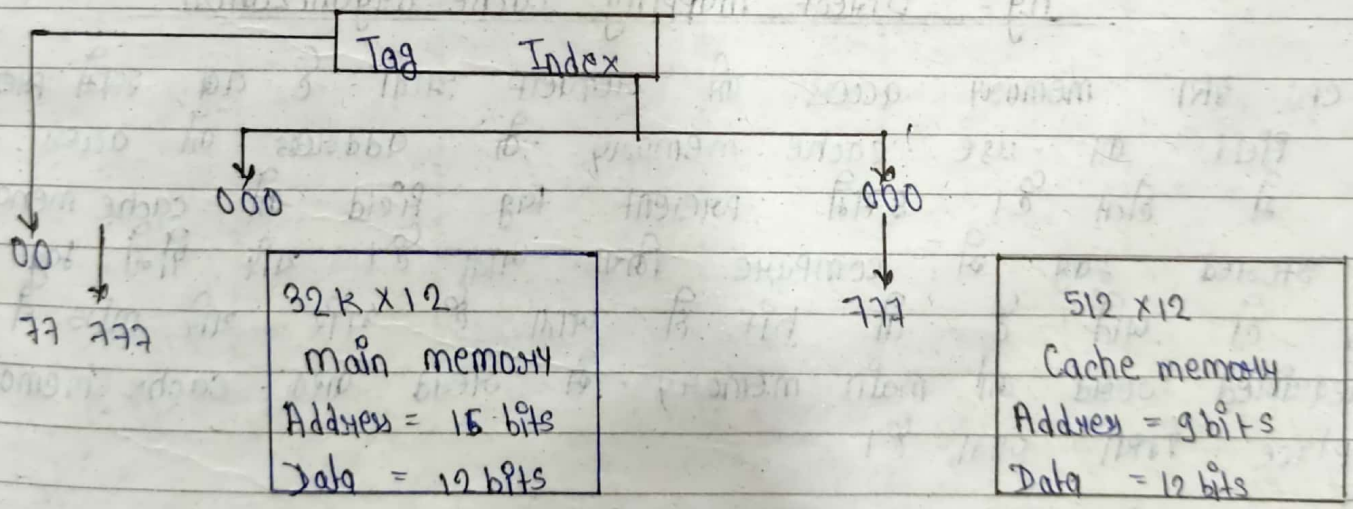
* 15 bit address को 5 digit octal number व 12 bit data को 4 digit octal number के रूप में show किया जाता है।
* CPU, 15 bit address को argument register में place करता है इसके बाद associative memory को matching address के लिए search किया जाता है। यदि matching address मिल जाता है तो उस address के 12 bit data को CPU को send कर दिया जाता है।

यदि matching address नही मिलता है तो main memory को given 15 bit address के लिए search किया जाता है। जब यह 15 bit address main memory में मिल जाता है तब इस

address और उसमे present data को cache memory मे transfer कर दिया जाता है।
यदि cache memory full हो तो नए address और data को cache मे place करने के लिए, cache memory से किसी पुराने address और data को delete / replace करना पड़ता है। cache memory FIFO replacement technique को follow करता है।

(a) Direct Mapping - Direct mapping cache organization मे RAM (Random Access Memory) का use किया जाता है क्योंकि associative memories की compare मे RAM सस्ती होती है। इस mapping technique मे 15 bit CPU address को दो part मे divide किया जाता है।

(a) 9 least significant bits को index field के रूप मे,
(b) 6 Remaining bits को Tag field के रूप में



इस figure मे direct mapping cache organization को show किया गया है जब cache memory से किसी new word की entry होती है तब इस word मे present data और उसके corresponding tag को store किया जाता है।

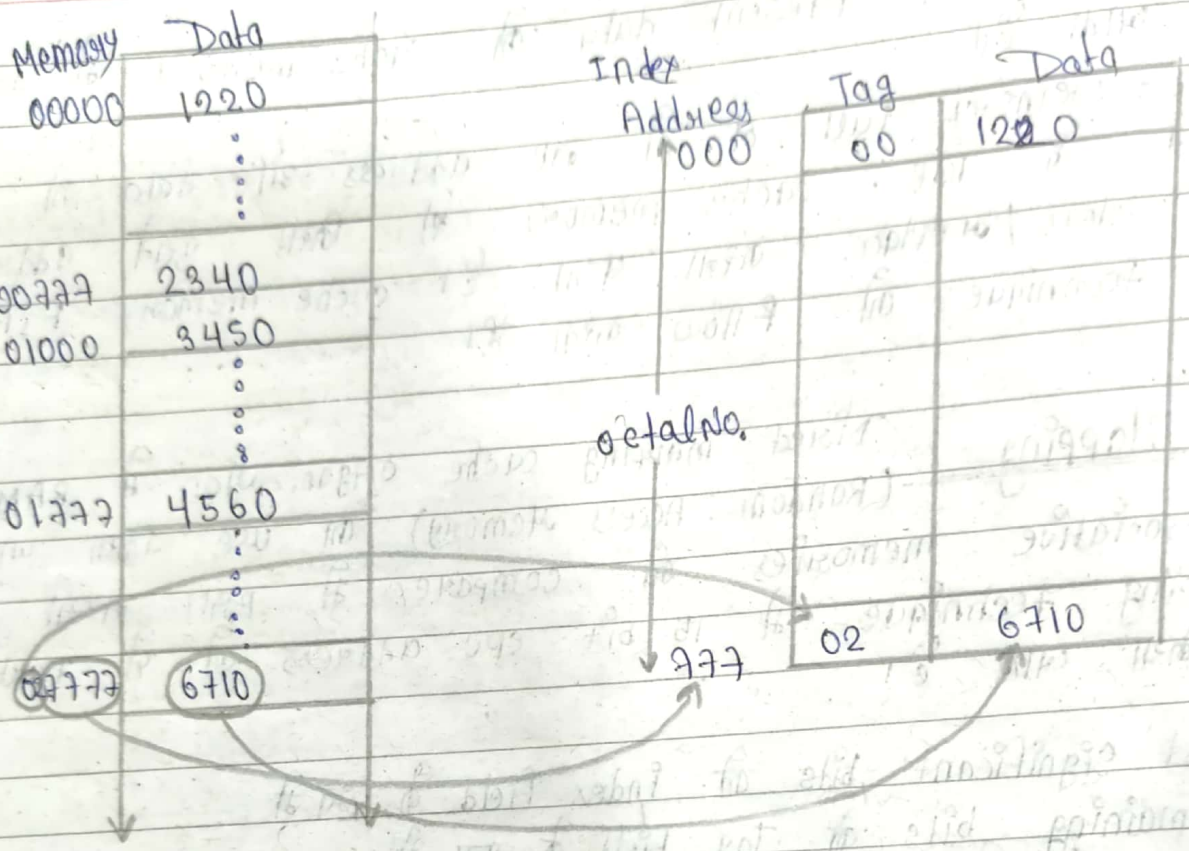


fig - Direct mapping cache organization

जब CPU द्वारा memory access की request आती है तब इसमें present index field का use cache memory के address को access करने में होता है। इसमें present tag field को cache memory में stored tag से compare किया जाता है। यदि दोनों tag match हो जाते हैं तो hit हो जाता है और यदि miss हो जाता तब requested word को main memory से read करके cache memory में place किया जाता है।

Disadvantage of direct mapping - यदि दो या दो से ज्यादा memory word में same index field और different tag field हो और उन्हे बार-बार access किया जाए तो इस mapping technique का hit ratio drastically (बहुत ज्यादा) reduce हो जाता है।

(3) Set Associative Mapping

set associative mapping technique, direct mapping technique का improvement है। Direct mapping technique में cache memory में किसी एक time में किसी एक index address में एक ही data और उसके tag को store किया जा सकता है। जबकि set Associative mapping में एक time में cache memory के किसी index address में दो या दो से ज्यादा data को उसके tag के साथ store किया जा सकता है। cache memory में किसी locations में जितने tag-data pair होते है वो मिलकर एक set बनाते है। इस figure में two-way set associative cache organization को show किया गया है।

Index	Tag '1'	Data '1'	Tag '2'	Data '2'
000	01	3450	02	5670
777	02	6710	00	2340

Fig :- Two way set associative cache memory organization

* इसमे हर Index address में 2 data और उनके tags होते है। हर एक tag के लिए 6 bits और data के लिए 12 bits की arrangement होती है। इसलिए two way set associative cache में word size = 2(6+12) = 36 bit लेती है।

* Direct mapping में cache memory की size 512 x 12 होता है जिसमे हर location मे एक data और 1 tag present होता है जबकि set associative mapping में cache memory की size 512 x 36 होती है जिसमे हर location मे 2 data items और 2 tag items होते है। इस type इसमे 1024 data item को memory मे store किया जा सकता है।

Generally set associative cache memory मे, जिसका set size "k" है वह एक index address मे main memory के पास data item को store करके रख सकता है।

Working :- main memory के 01000 और 02000 address पर present data को cache memory के index address 000 पर store किया गया है similiarly address 00777 & 02777 पर present data को index address 777 पर store किया गया है।

जब CPU से memory access की request आती है तब index address का use cache को access करने के लिए किया जाता है। CPU के tag field का comparison cache memory के दोनो tags से किया जाता है। यह comparison ठीक वैसे ही किया जाता है जैसे associative memory के search मे किया जाता है, इसलिए इसे set associative कहते है।

यदि "hit" होता है तो इस data को CPU को send कर दिया जाता है यदि "miss" होता है तो इसका मतलब यह है कि यह address cache memory मे present नही है तब CPU इसे cache memory मे place कर देता है। set associative mapping मे cache memory FIFO principle पर work करता है।