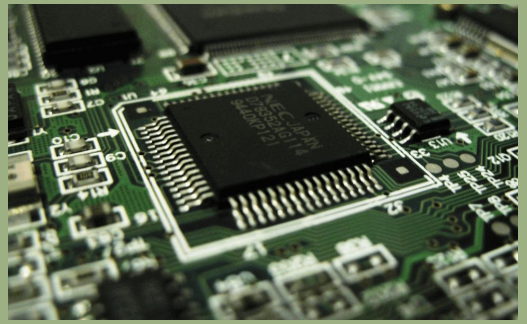


8085 Microprocessor

Dr. Manju Khurana
Assistant Professor, CSED
TIET, Patiala
manju.khurana@thapar.edu

Instruction Set

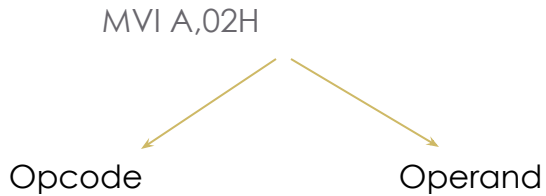
Instructions have been classified into five following groups:

- Data Transfer Group
- Arithmetic Group
- Logical Group
- Branch Control Group
- I/O and Machine Control Group

Instruction Formats

Each instruction has two parts:

- **Opcode (operation code)-** The first part is the task or operation to be performed.
- **Operand -** The second part is the data to be operated on. Data can be given in various forms.
  - It can specify in various ways: may include 8 bit/16 bit data, an internal register, memory location or 8 bit /16 bit address.
  - For Eg.:



NOTATIONS	MEANING
M	Memory location pointed by HL register pair
r	8- bit register
rp	16-bit register
rs	Source register
rd	Destination register
addr	16-bit address

Data Transfer Instructions

- These instructions move data between registers, or between memory and registers.
- These instructions copy data from source to destination.
- **While copying, the contents of source are not modified**.

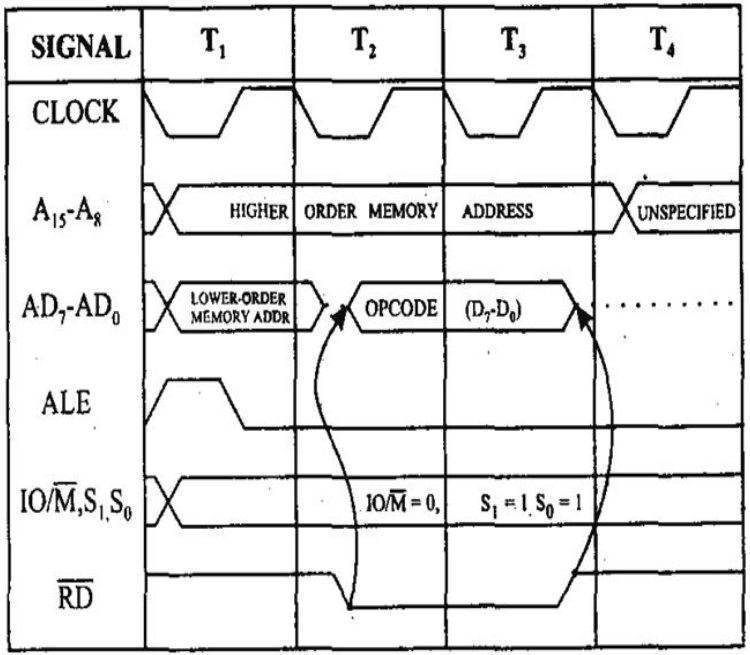
Sr. No.	Data transfer	Example
1.	Between registers.	Register B → Register D.
2.	Specific data byte to register or a memory location.	Data byte → Register B.
3.	Between memory location and register.	Memory location → Register A.
4.	Between an I/O device and the accumulator.	Input device → Register A.
5.	Between a register pair and the stack.	Register pair data → Stack locations.

1. MOV r_1, r_2

Move the contents of one register to another register.

$[r_1] \leftarrow [r_2]$

States: 4
Flags: None
Addressing: Register
Machine Cycles: 1

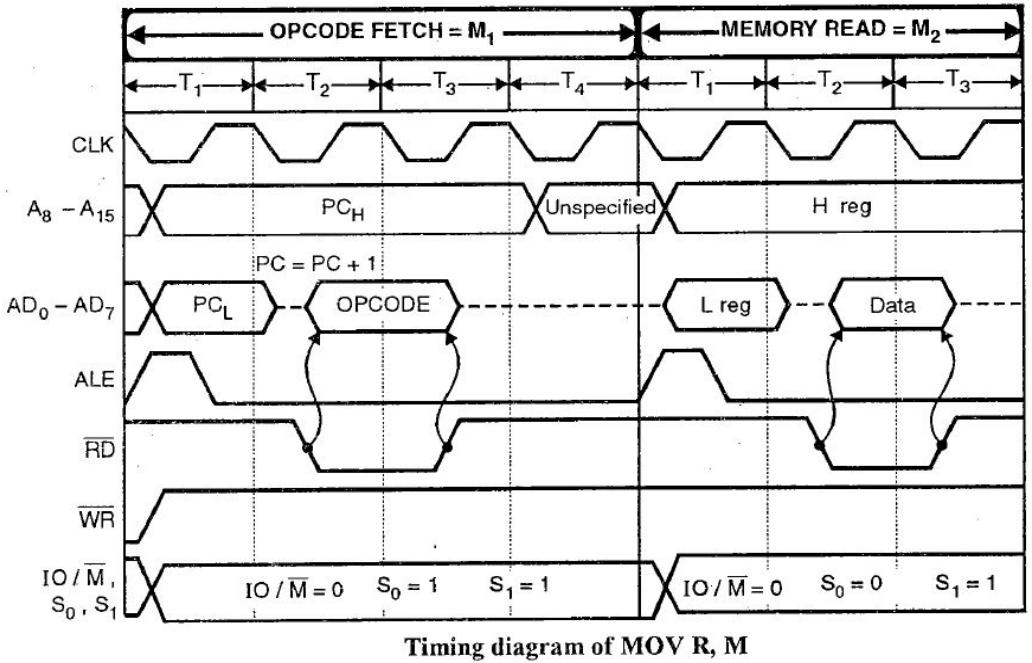


2. MOV r, M

Move the contents of memory to register.

[r] ← [H-L]

States: 7
Flags: None
Addressing: Register Indirect
Machine Cycles: 2



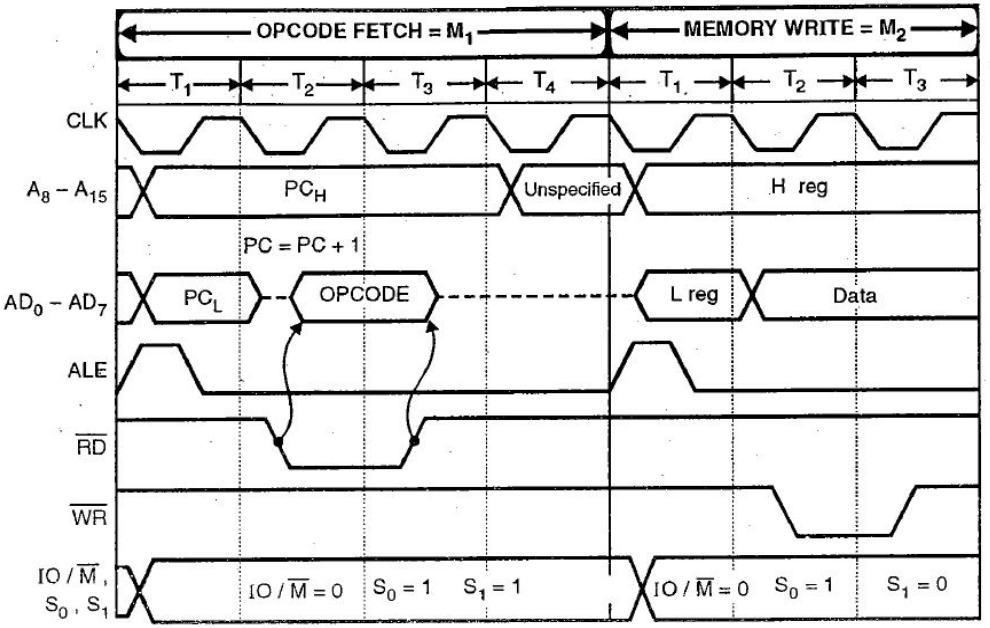
Timing diagram of MOV R, M

3. MOV M, r

Move the contents of register to memory.

[H-L] ← [r]

States: 7
Flags: None
Addressing: Register Indirect
Machine Cycles: 2

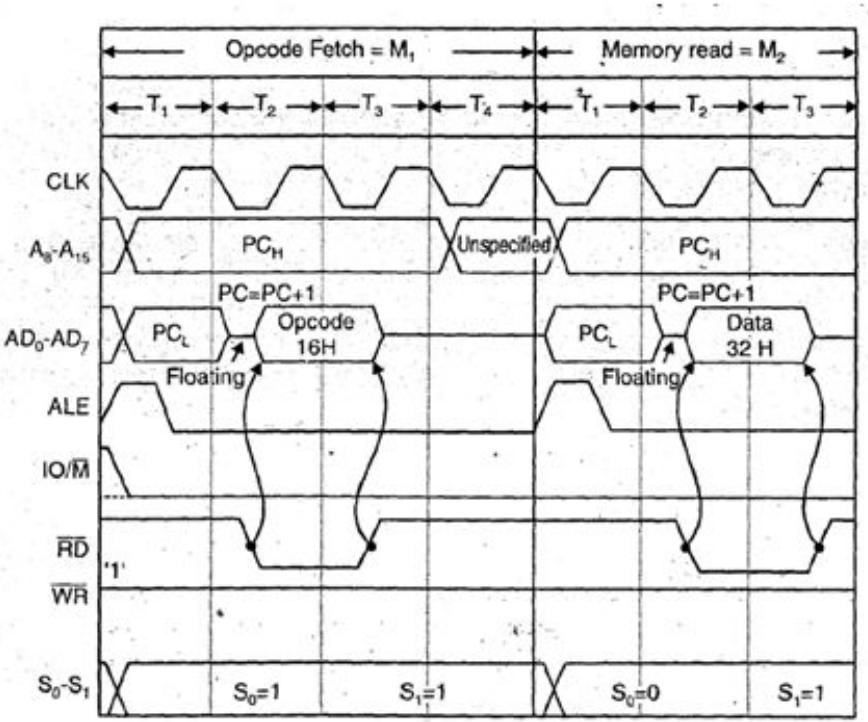


Timing diagram of MOV M, R

4. MVI r, data

Move immediate data to register.
[r] ← data

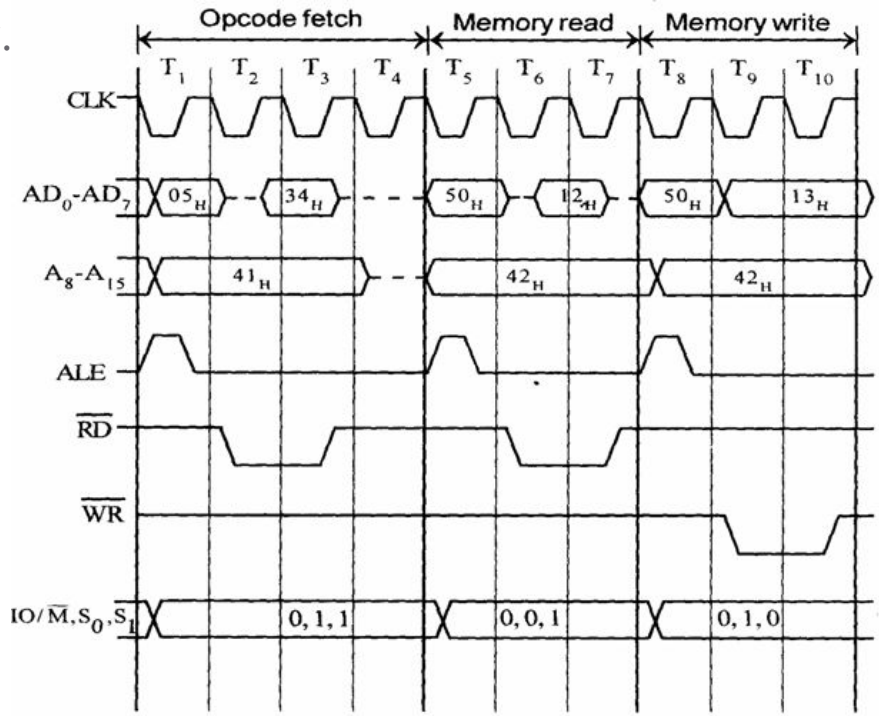
States: 7
Flags: None
Addressing: Immediate
Machine Cycles: 2



5. MVI M, data

Move immediate data to memory.
[H-L] ← data

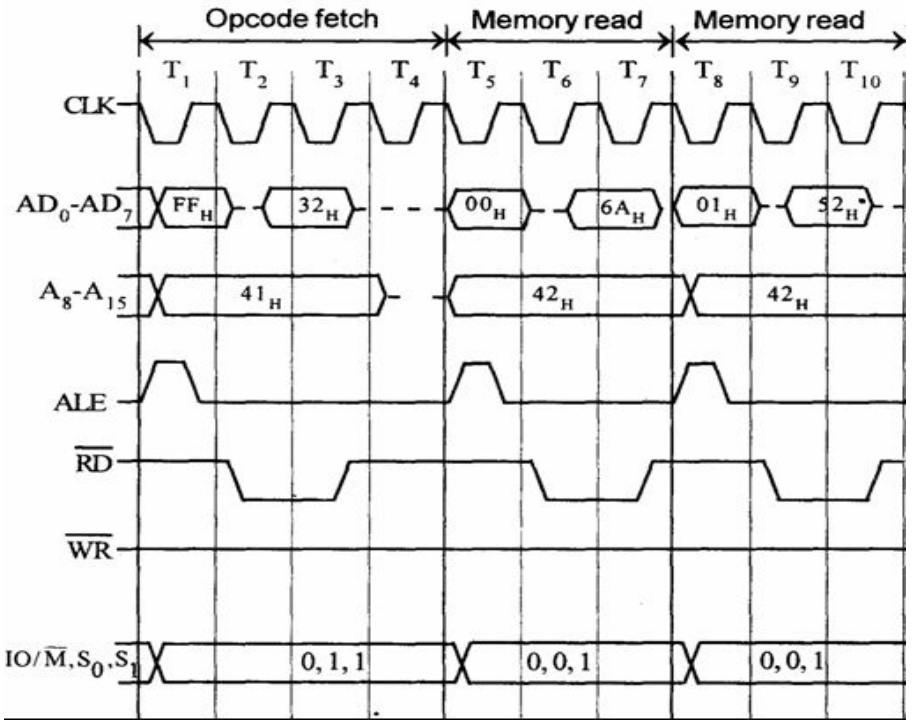
States: 10
Flags: None
Addressing: Immediate/
Register Indirect
Machine Cycles: 3



6. LXI rp, data 16

Load register pair immediate.
[rp] ← data 16 bits
[rh] ← 8 MSBs of data
[rl] ← 8 LSBs of data

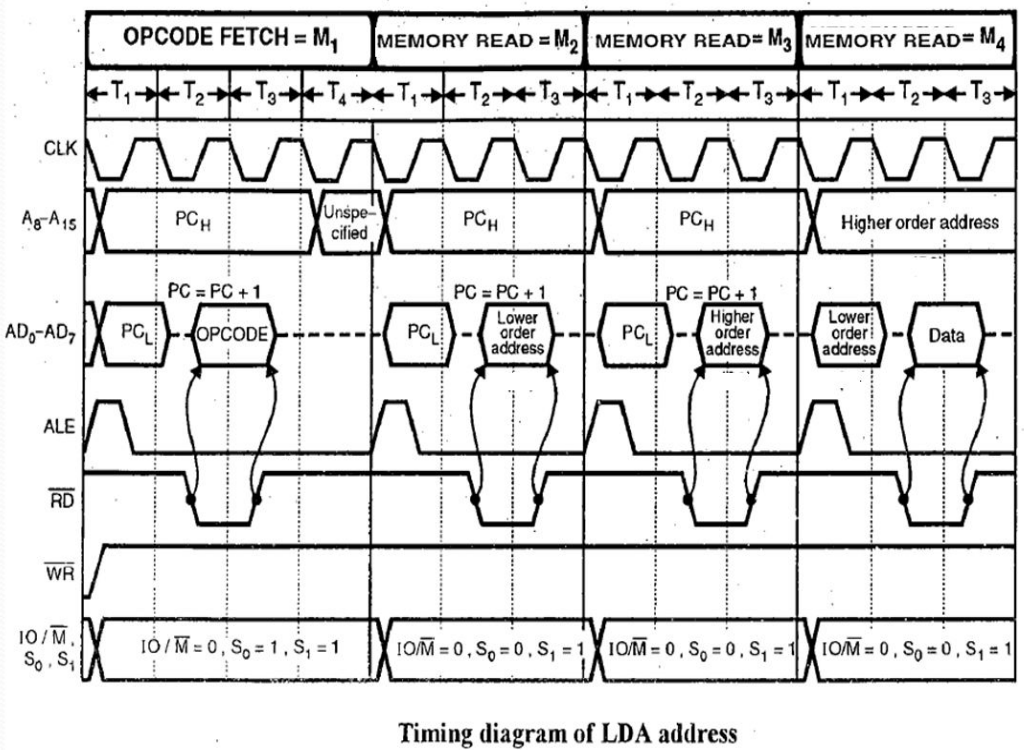
States: 10
Flags: None
Addressing: Immediate
Machine Cycles: 3



7. LDA addr

Load Accumulator direct.
[A] ← addr

States: 13
Flags: None
Addressing: direct
Machine Cycles: 4

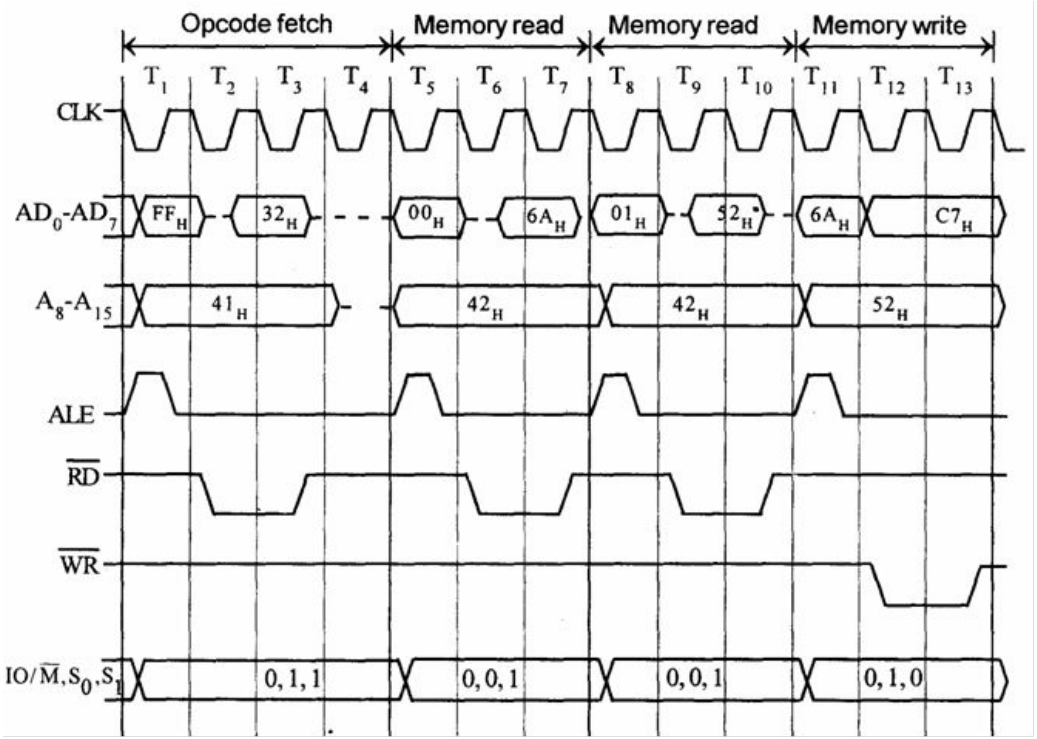


Timing diagram of LDA address

8. STA addr

Store Accumulator direct.
[addr] ← [A]

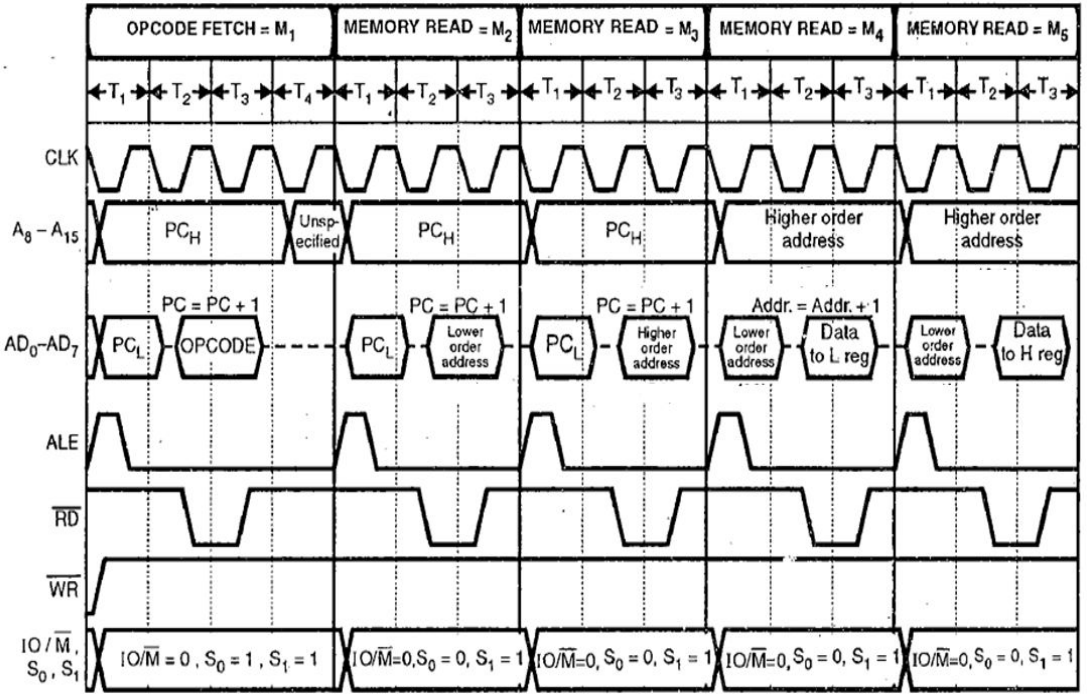
States: 13
Flags: None
Addressing: direct
Machine Cycles: 4



9. LHLD addr

Load H-L pair direct.
[L] ← [addr]
[H] ←[addr+1]

States: 16
Flags: None
Addressing: direct
Machine Cycles: 5

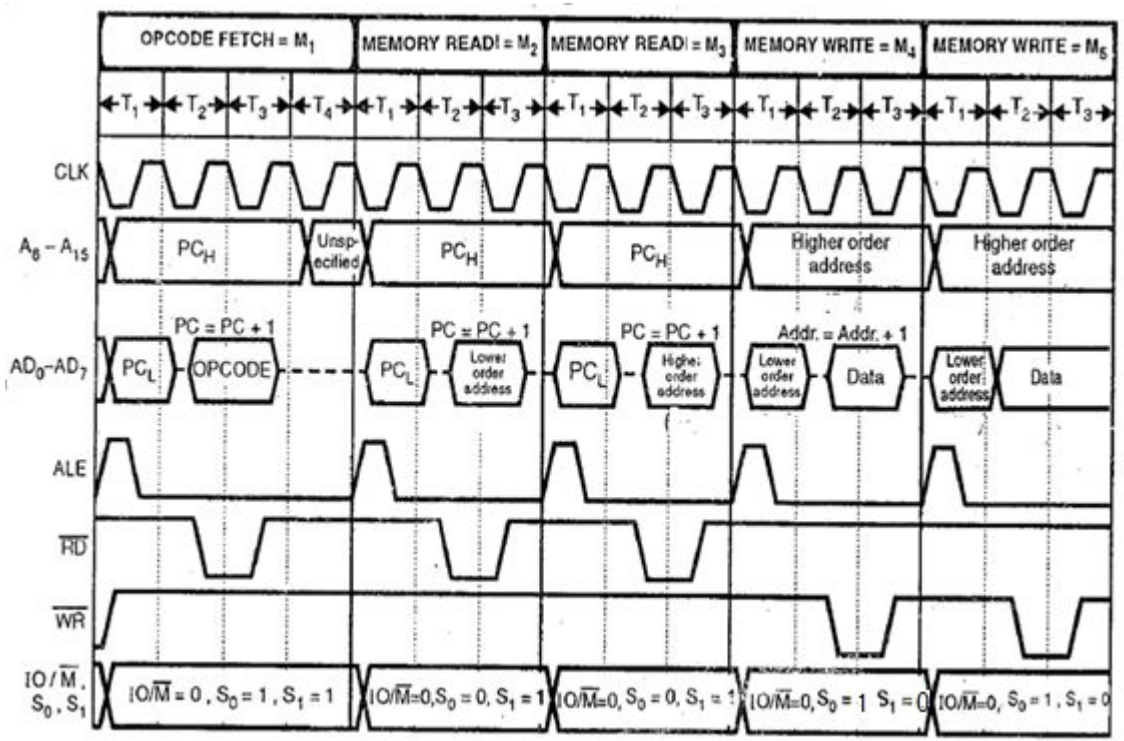


Timing diagram of LHLD address

10. SHLD addr

Store H-L pair direct.
[addr] ← [L]
[addr+1] ← [H]

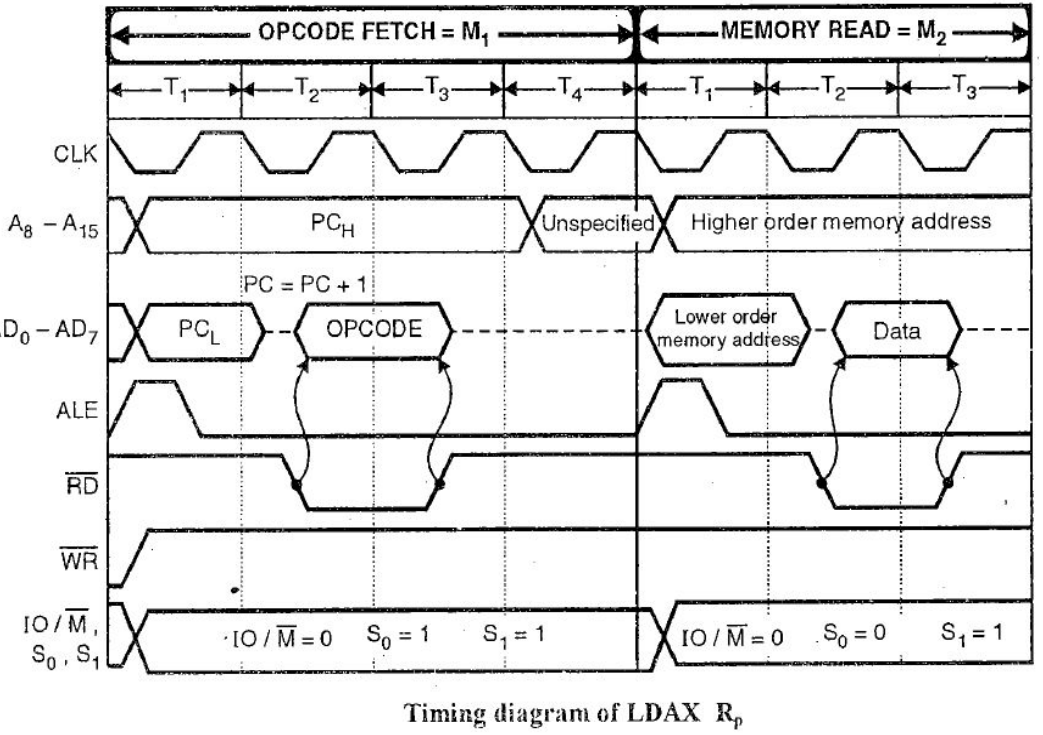
States: 16
Flags: None
Addressing: direct
Machine Cycles: 5



11. LDAX rp

Load accumulator indirect.
[A] ← [rp]

States: 7
Flags: None
Addressing: Register indirect
Machine Cycles: 2

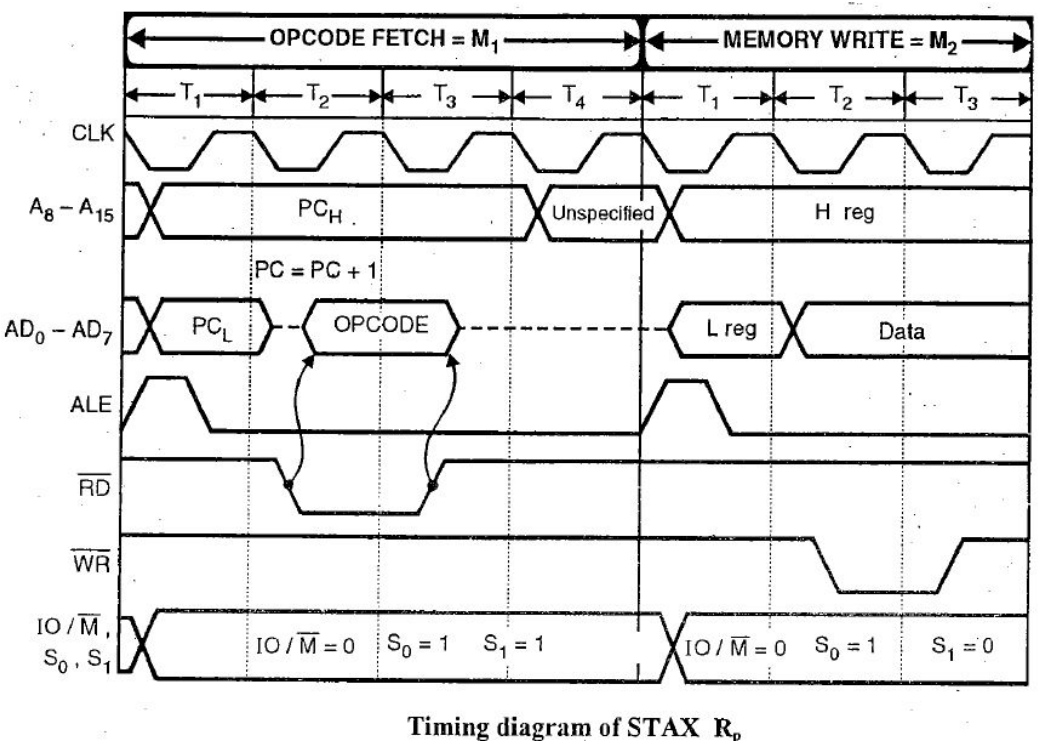


Timing diagram of LDAX R_p

12. STAX rp

Store accumulator indirect.
[rp] ← [A]

States: 7
Flags: None
Addressing: Register indirect
Machine Cycles: 2

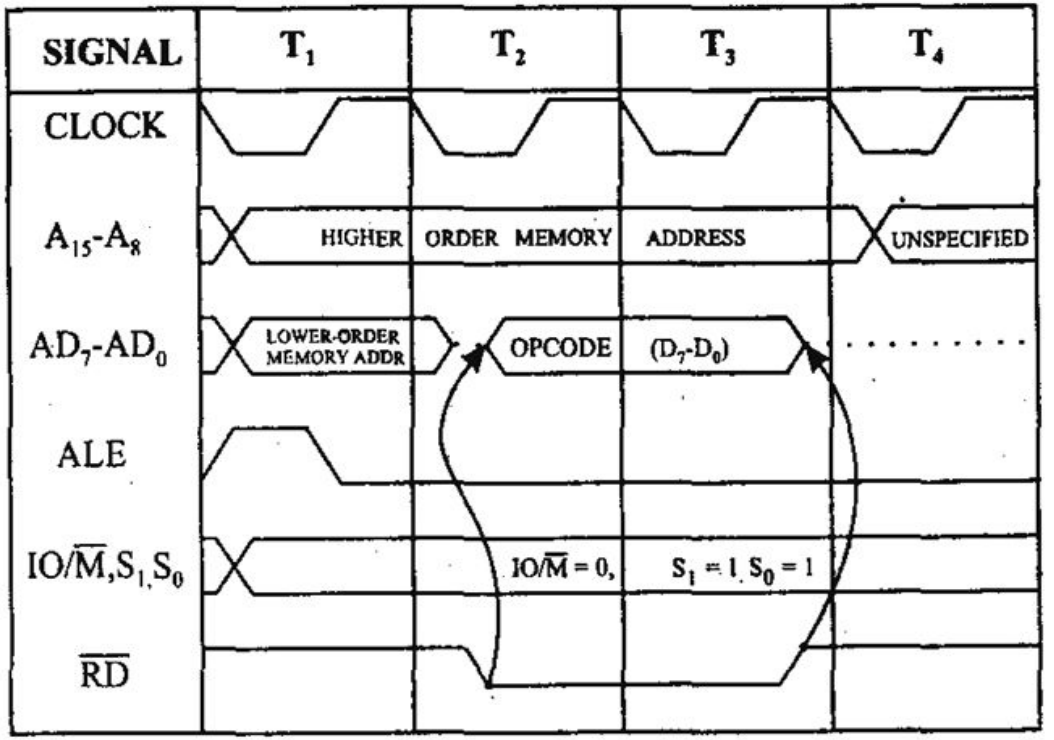


Timing diagram of STAX R_p

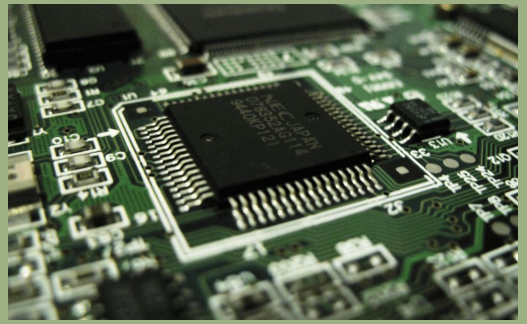
13. XCHG

Exchange the contents of H-L pair with D-E pair.
[H-L] ↔ [D-E]

States: 4
Flags: None
Addressing: Register
Machine Cycles: 1




Thank you!



8085 Microprocessor

Dr. Manju Khurana
Assistant Professor, CSED
TIET, Patiala
manju.khurana@thapar.edu

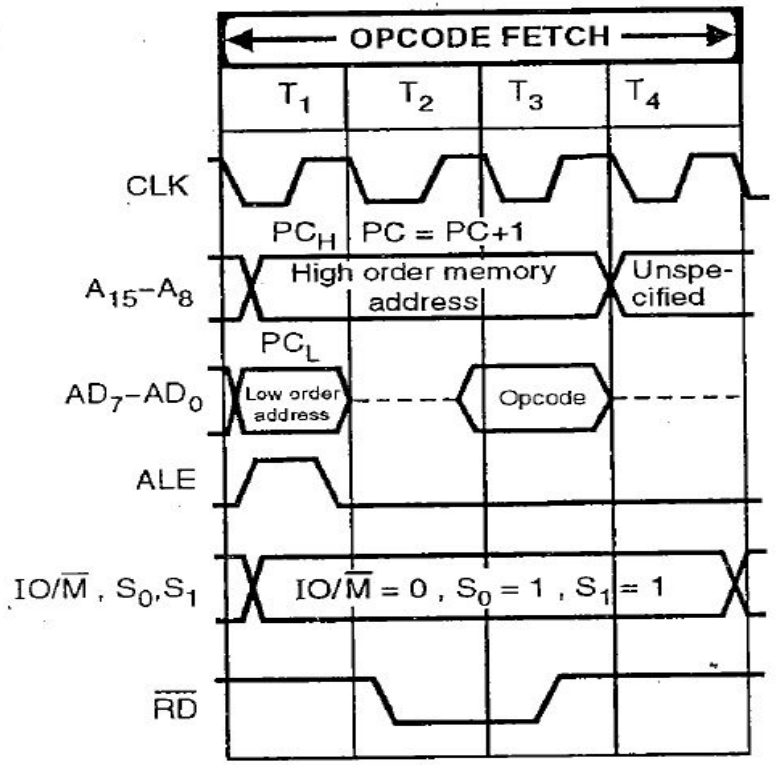
Arithmetic Instructions

- These instructions perform the operations like:
  - Addition
  - Subtraction
  - Increment
  - Decrement

1. ADD r

Add register to accumulator.
[A] ← [A] + [r]

States: 4
Flags: all
Addressing: Register
Machine Cycles: 1

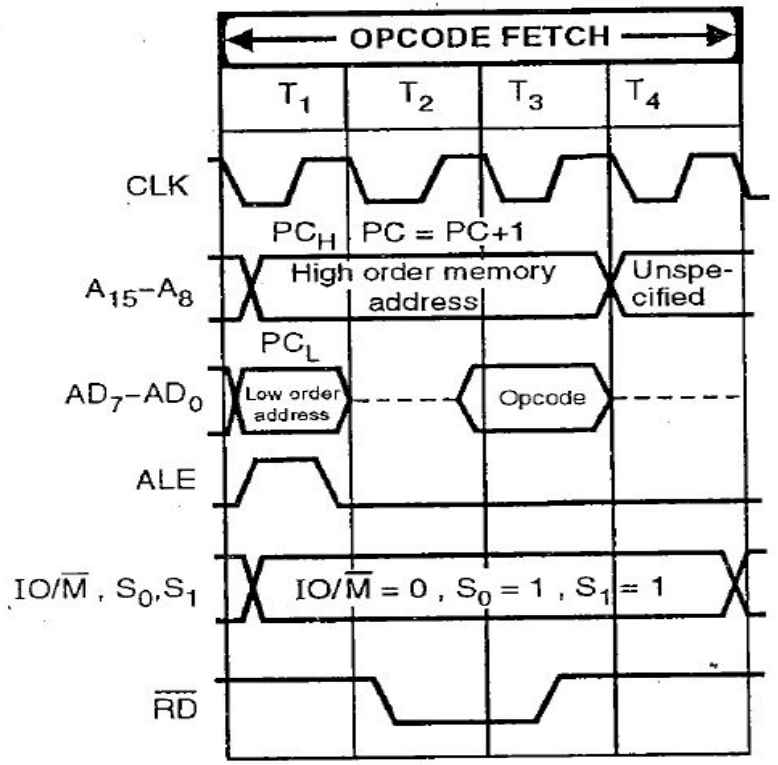


2. SUB r

Subtract register from accumulator.

[A] ← [A] - [r]

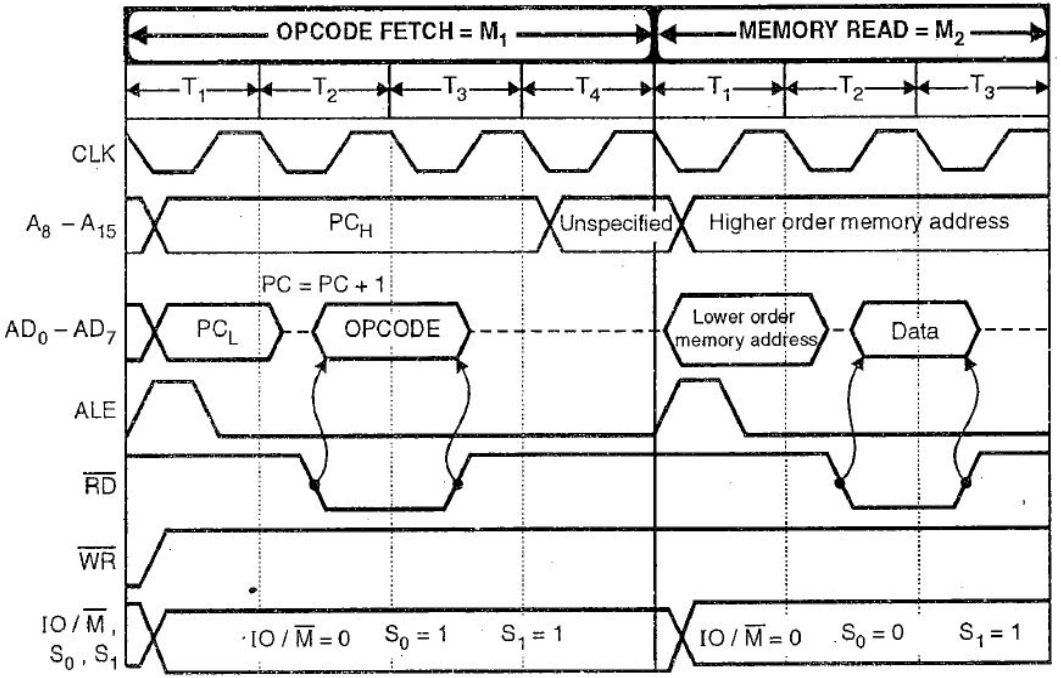
States: 4
Flags: all
Addressing: Register
Machine Cycles: 1



3. ADD M

Add memory to accumulator.
[A] ← [A] + [H-L]

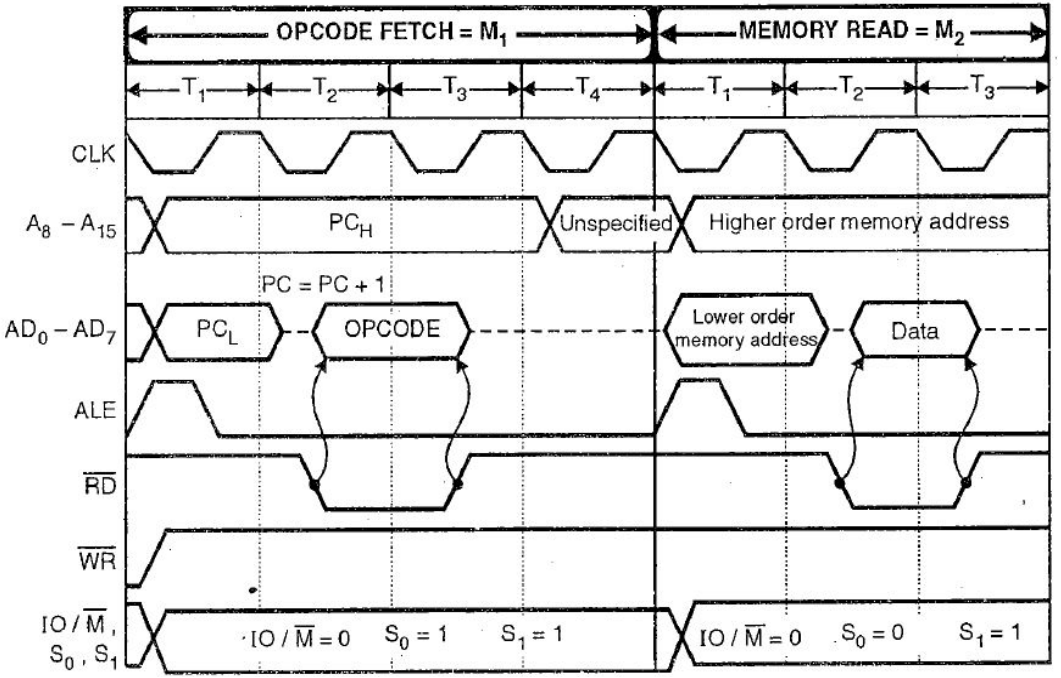
States: 7
Flags: all
Addressing: Register Indirect
Machine Cycles: 2



4. SUB M

Subtract memory from accumulator.
[A] ← [A] - [H-L]

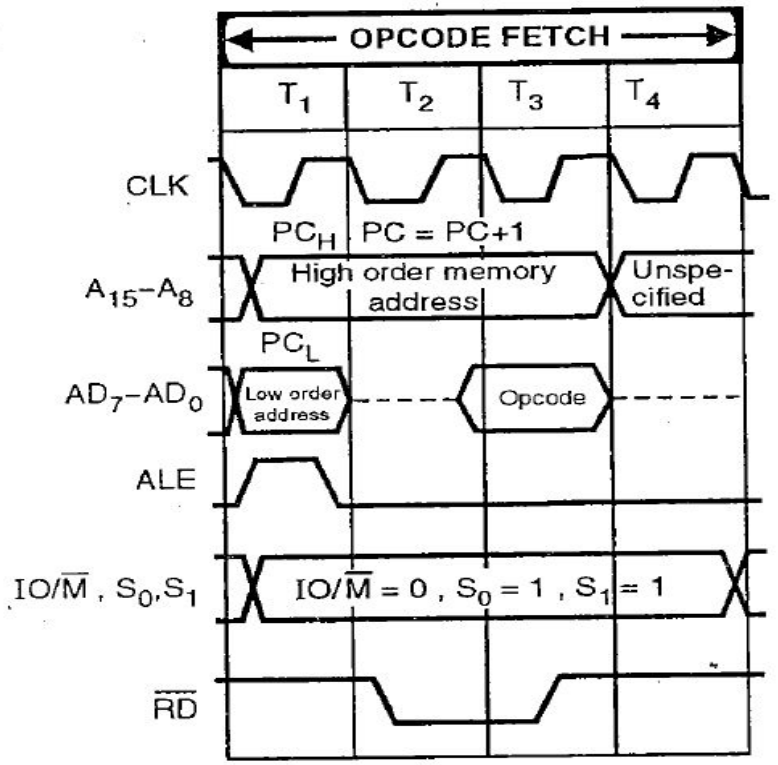
States: 7
Flags: all
Addressing: Register Indirect
Machine Cycles: 2



5. ADC r

Add register with carry to accumulator.
[A] ← [A] + [r] + [CS]

States: 4
Flags: all
Addressing: Register
Machine Cycles: 1

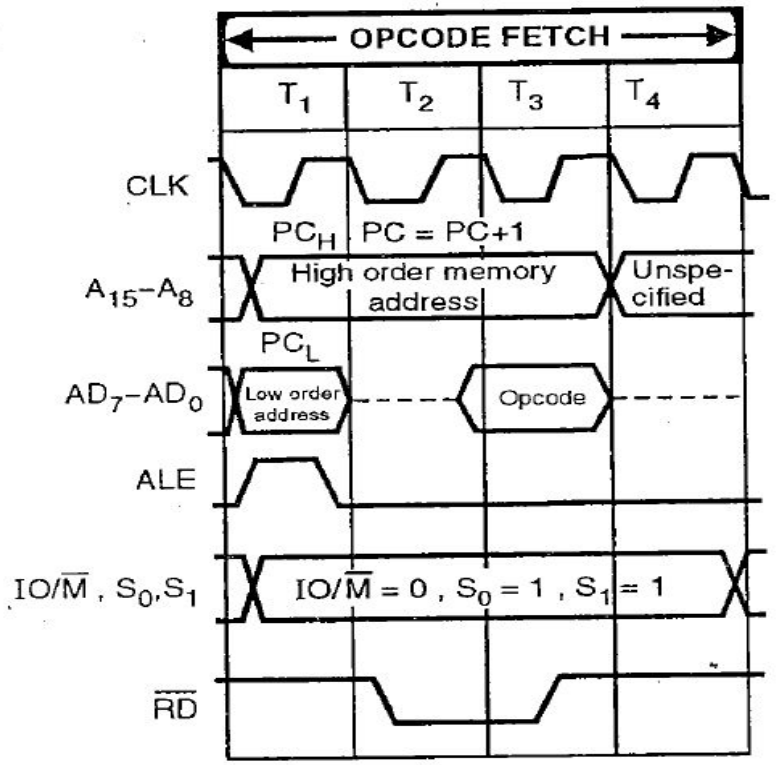


6. SBB r

Subtract register from accumulator with borrow.

[A] ← [A] - [r] - [CS]

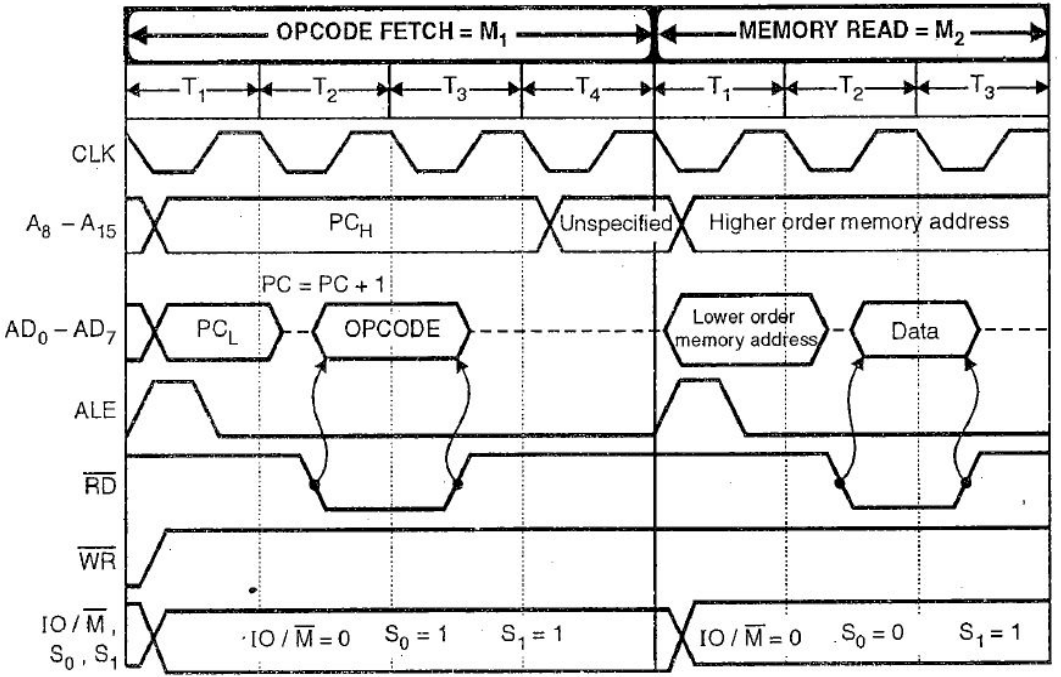
States: 4
Flags: all
Addressing: Register
Machine Cycles: 1



7. ADC M

Add memory with carry to accumulator.
[A] ← [A] + [H-L] + [CS]

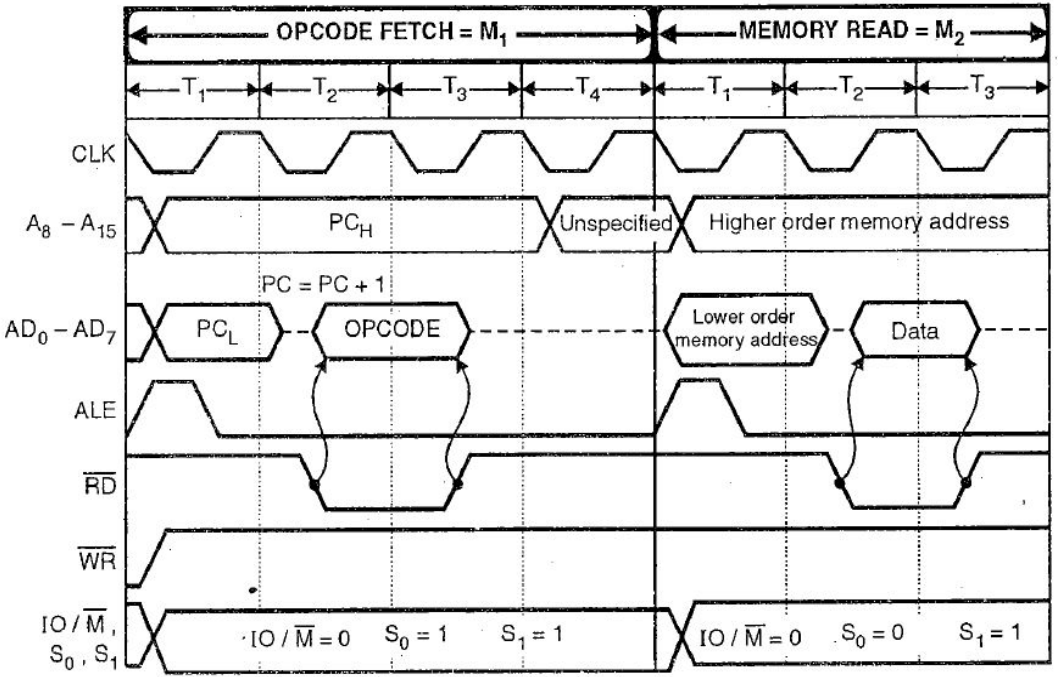
States: 7
Flags: all
Addressing: Register Indirect
Machine Cycles: 2



8. SBB M

Subtract memory from Accumulator with borrow.
[A] ← [A] - [H-L] - [CS]

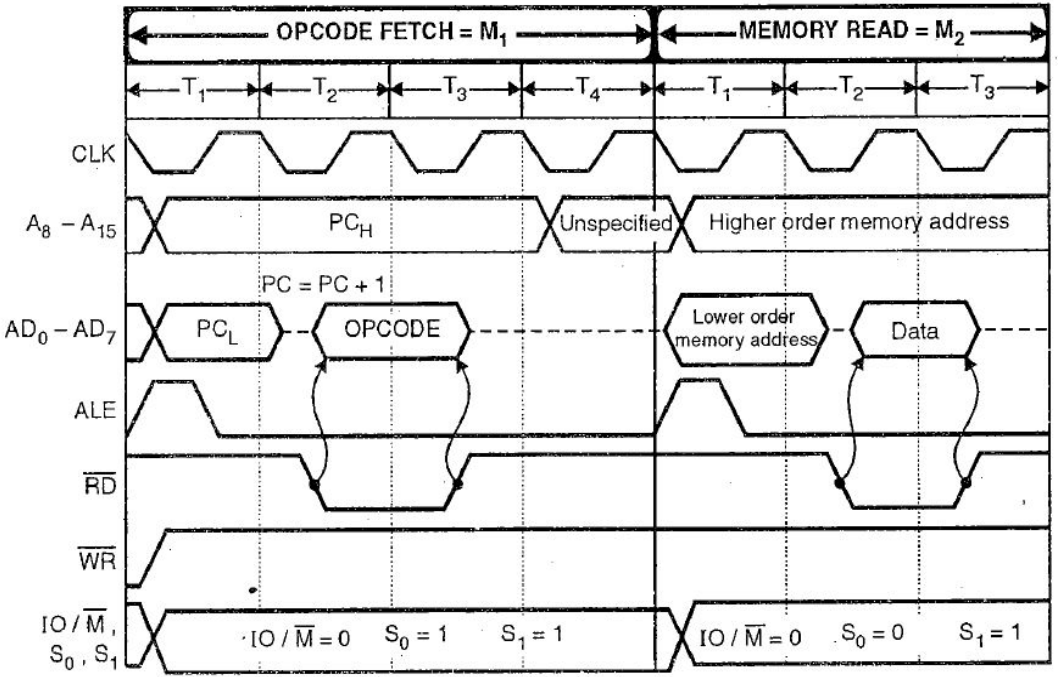
States: 7
Flags: all
Addressing: Register Indirect
Machine Cycles: 2



9. ADI data

Add immediate data to accumulator.
[A] ← [A] + data

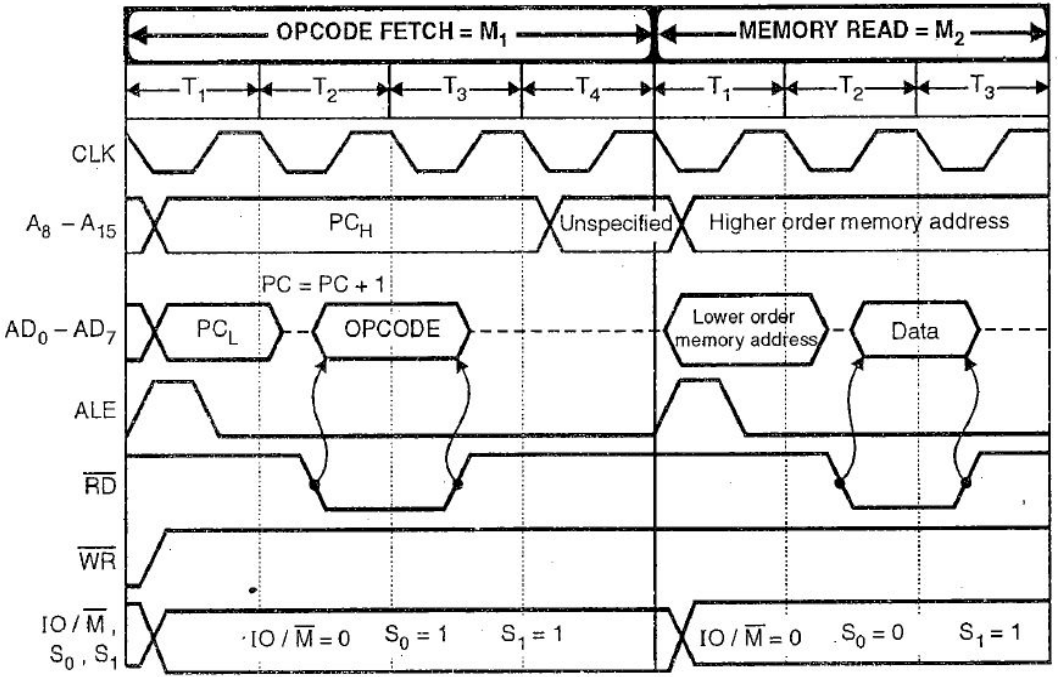
States: 7
Flags: all
Addressing: Immediate
Machine Cycles: 2



10. SUI data

Subtract immediate data from accumulator.
[A] ← [A] - data

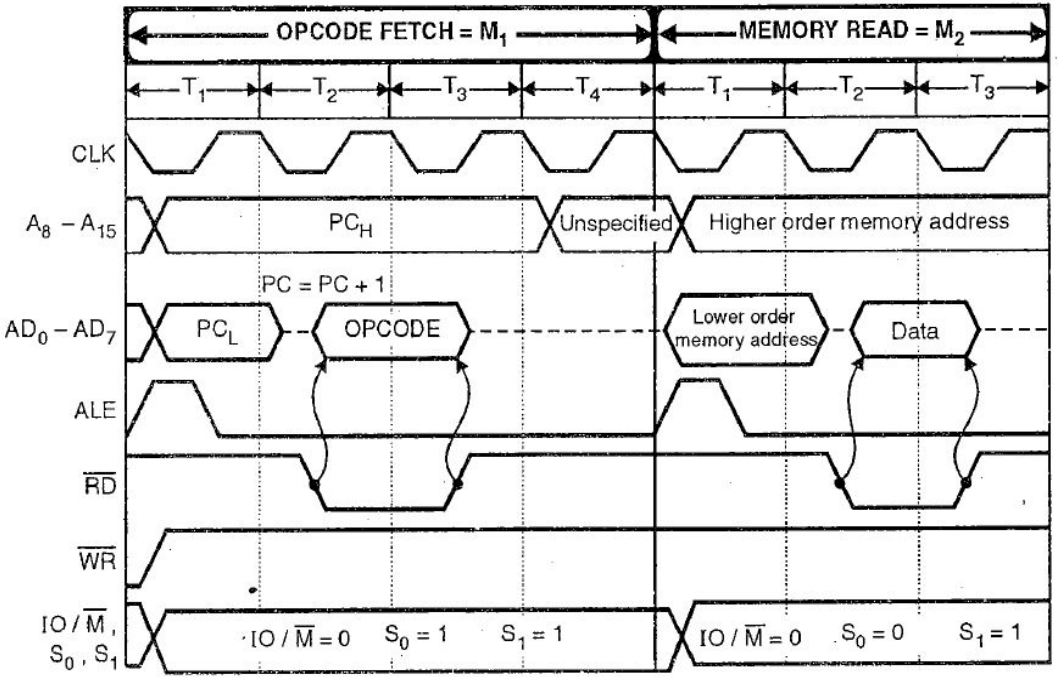
States: 7
Flags: all
Addressing: Immediate
Machine Cycles: 2



11. ACI data

Add with carry immediate data to accumulator.
[A] ← [A] + data +[CS]

States: 7
Flags: all
Addressing: Immediate
Machine Cycles: 2

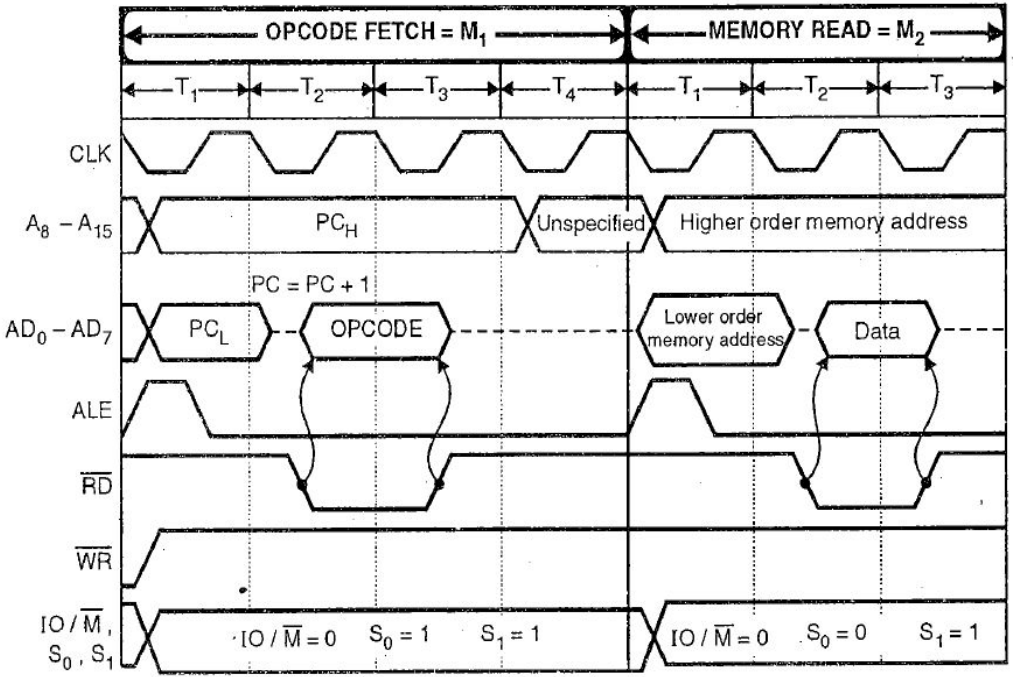


12. SBI data

Subtract immediate data from accumulator with borrow.

[A] ← [A] - data -[CS]

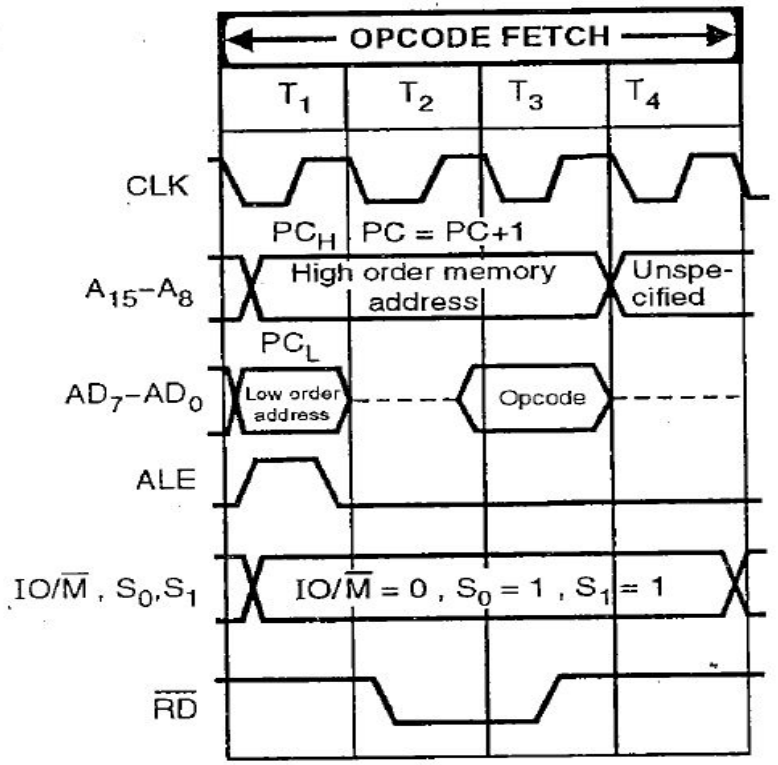
States: 7
Flags: all
Addressing: Immediate
Machine Cycles: 2



13. INR r

Increment register content.
[r] ← [r] + 1

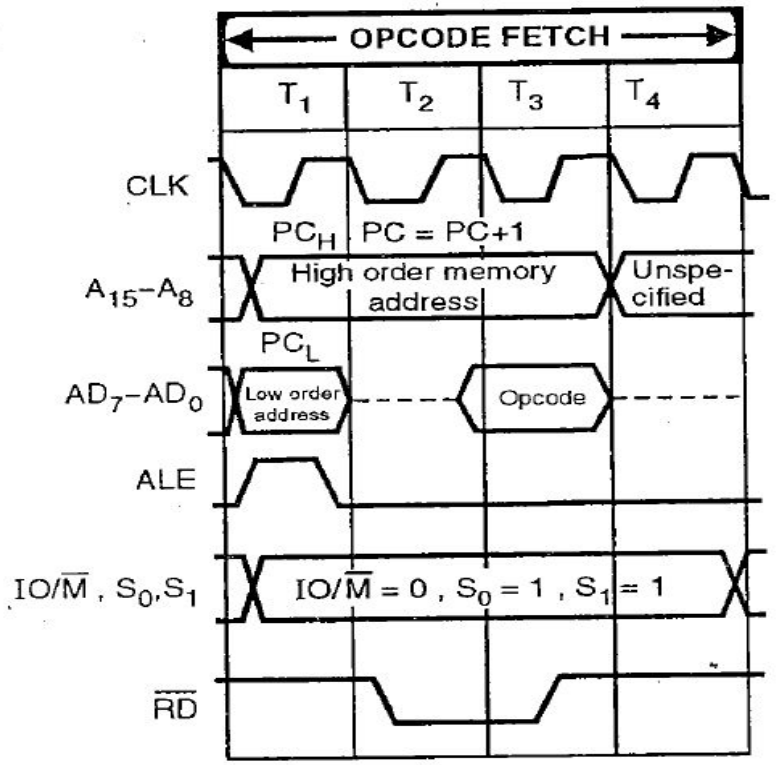
States: 4
Flags: all except carry flag
Addressing: Register
Machine Cycles: 1



14. DCR r

Decrement register content.
[r] ← [r] - 1

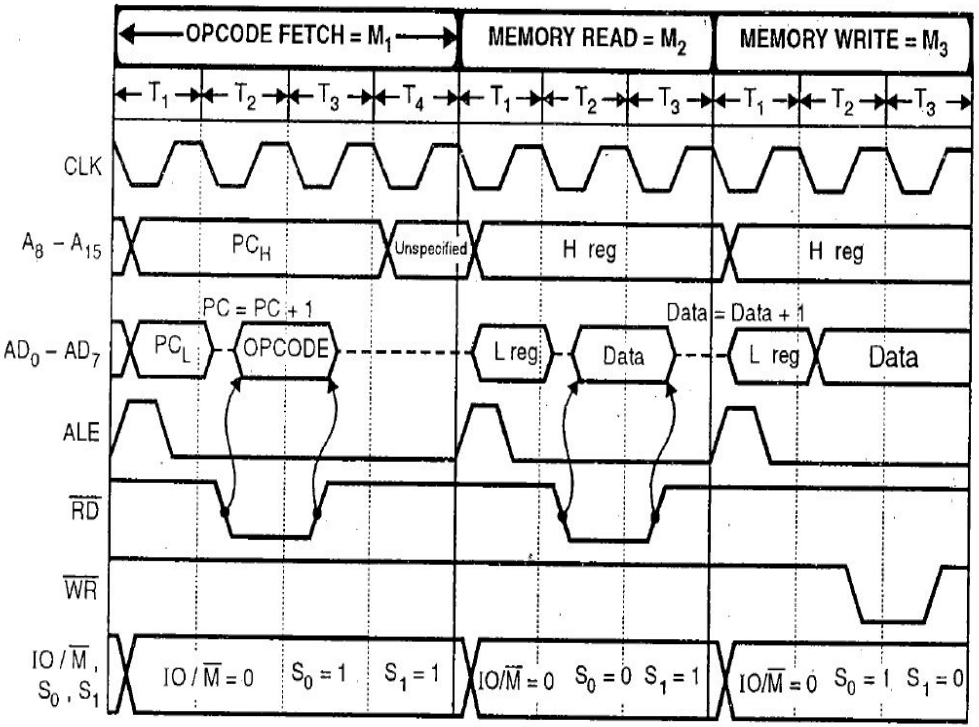
States: 4
Flags: all except carry flag
Addressing: Register
Machine Cycles: 1



15. INR M

Increment memory content.
[H-L] ← [H-L] + 1

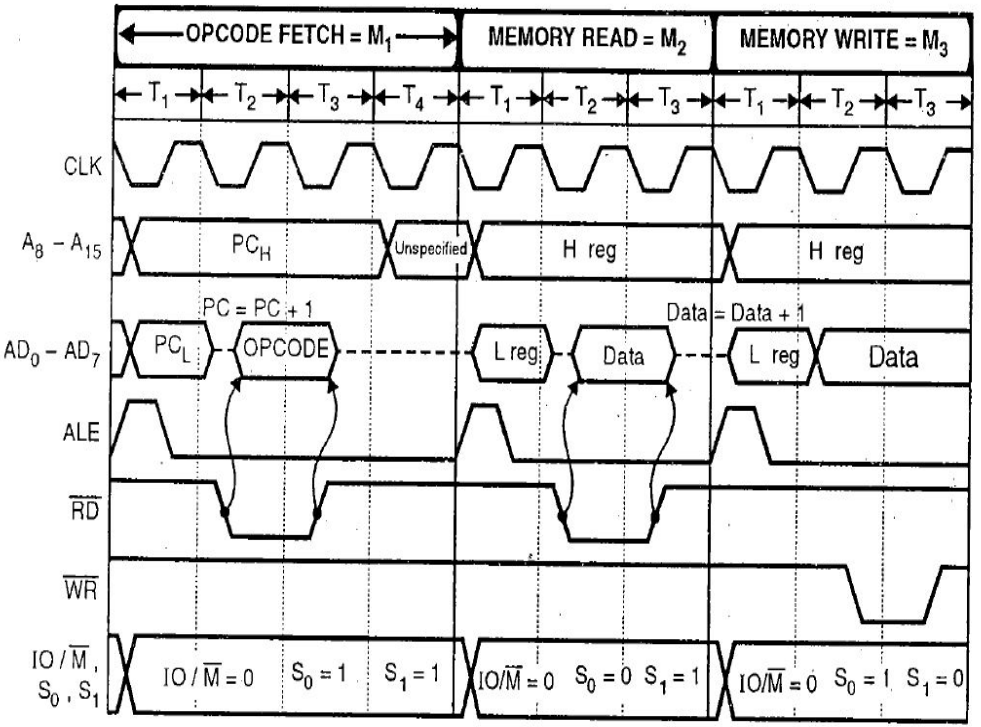
States: 10
Flags: all except carry flag
Addressing: Register Indirect
Machine Cycles: 3



16. DCR M

Decrement memory content.
[H-L] ← [H-L] - 1

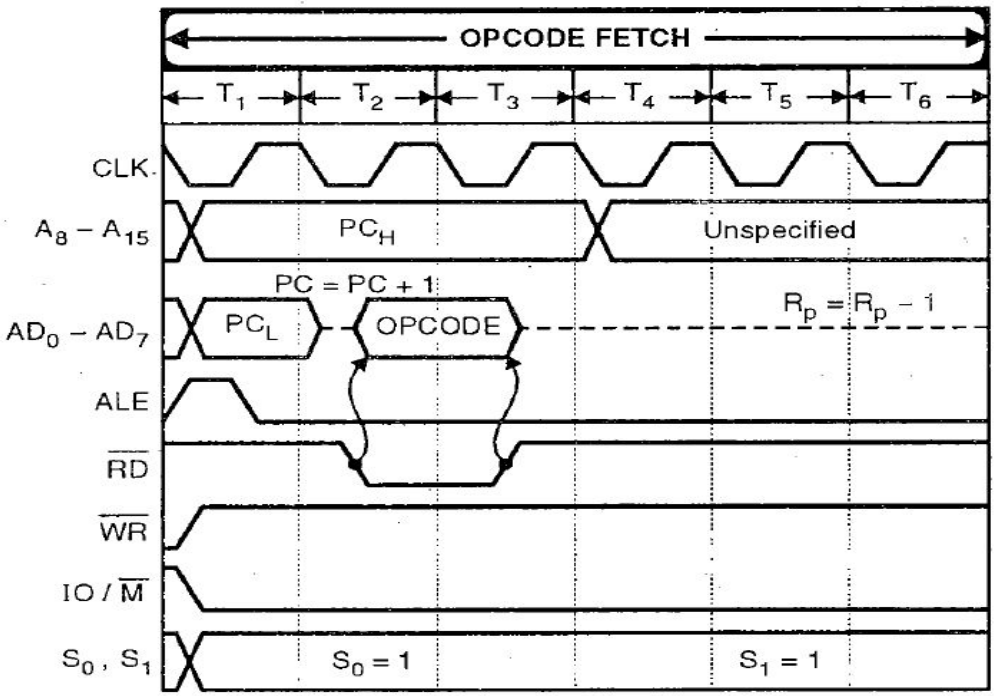
States: 10
Flags: all except carry flag
Addressing: Register Indirect
Machine Cycles: 3



17. INX rp

Increment register pair.
[rp] ← [rp] + 1

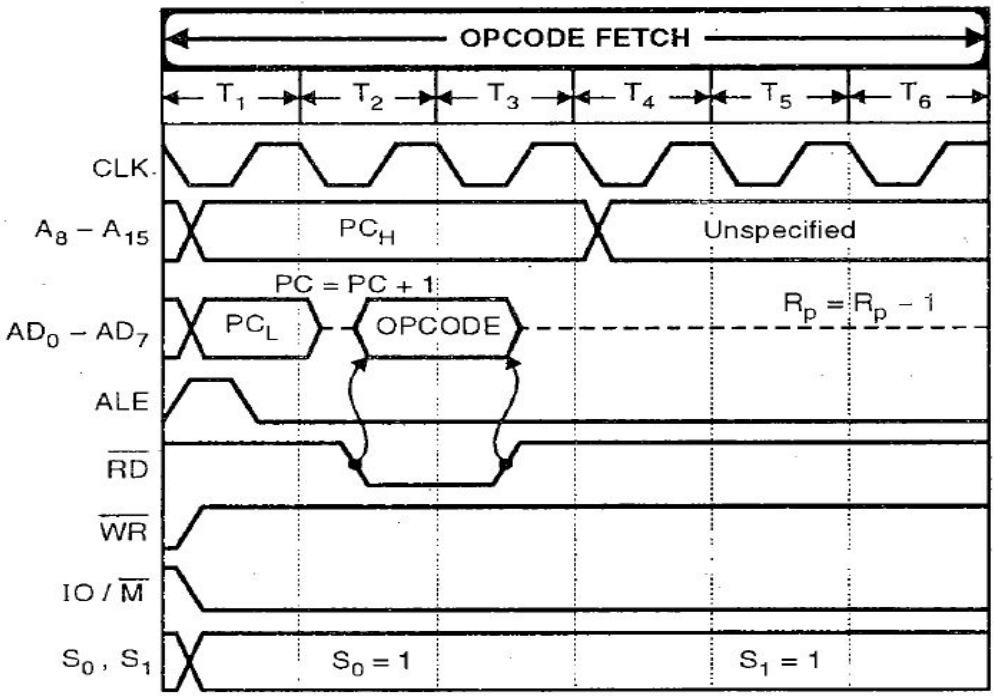
States: 6
Flags: None
Addressing: Register
Machine Cycles: 1



18. DCX rp

Decrement register pair.
[rp] ← [rp] - 1

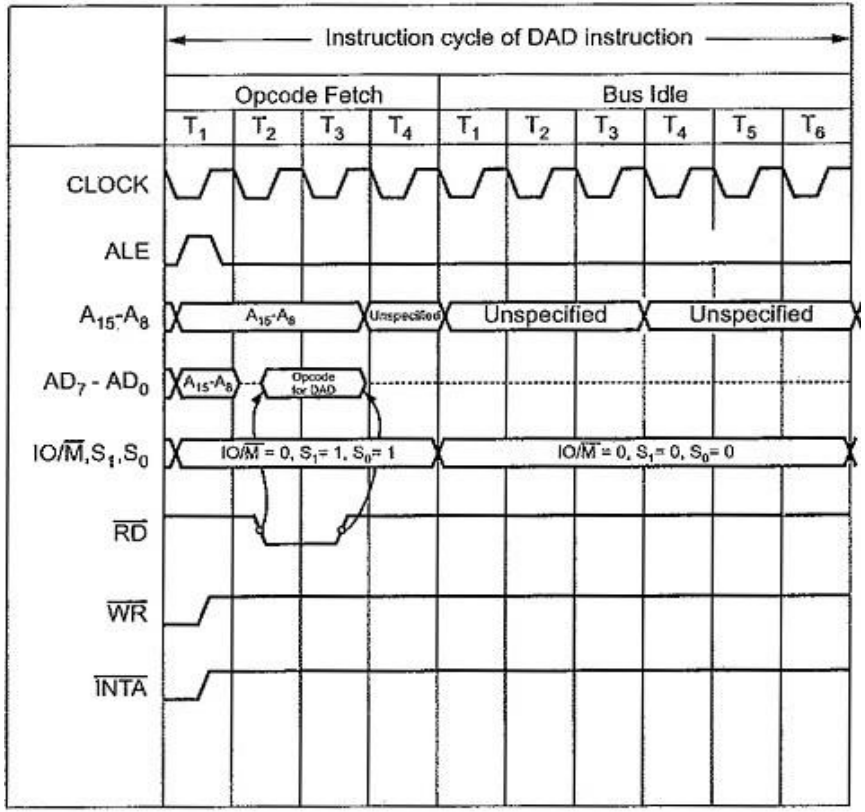
States: 6
Flags: None
Addressing: Register
Machine Cycles: 1



19. DAD rp

Add register pair to H-L pair.
[H-L] ← [H-L] +[rp]

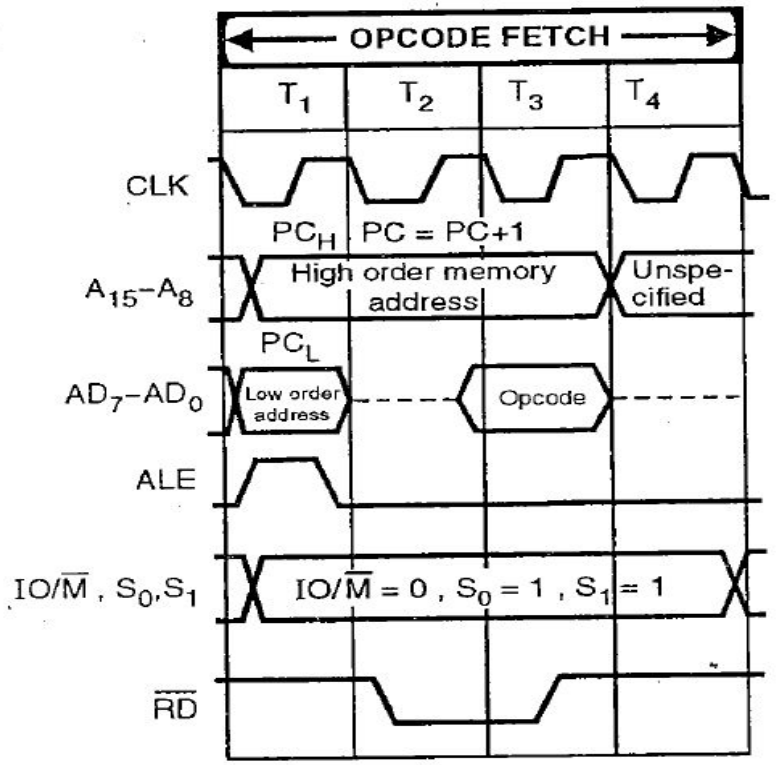
States: 10
Flags: CS
Addressing: Register
Machine Cycles: 3



20. DAA

Decimal Adjust Accumulator

States: 4
Flags: all
Addressing: Implicit
Machine Cycles: 1



20. DAA (Decimal Adjust Accumulator) contd..

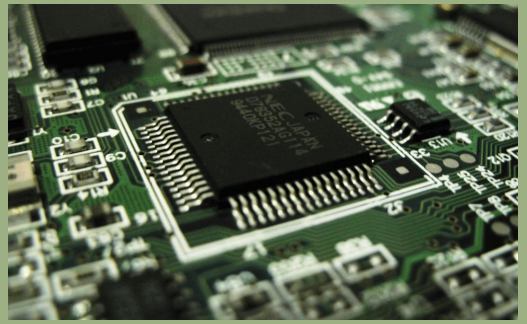
- Accumulator will automatically adjust in BCD format.
  - If lower nibble of Acc > 1001 then add 0110.
  - If lower nibble of Acc ≤1001 but AC->1, then add 0110.
  - If upper nibble of Acc >1001 then add 0110.
  - If upper nibble of Acc ≤1001 but CY->1, then add 0110.

Example:

```
LXI H, 8A79H
MOV A,L
ADD H
DAA
MOV H,A
PCHL
```

After PCHL next instruction will fetch from?


Thank you!



8085 Microprocessor

Dr. Manju Khurana
Assistant Professor, CSED
TIET, Patiala
manju.khurana@thapar.edu

Logical Instructions

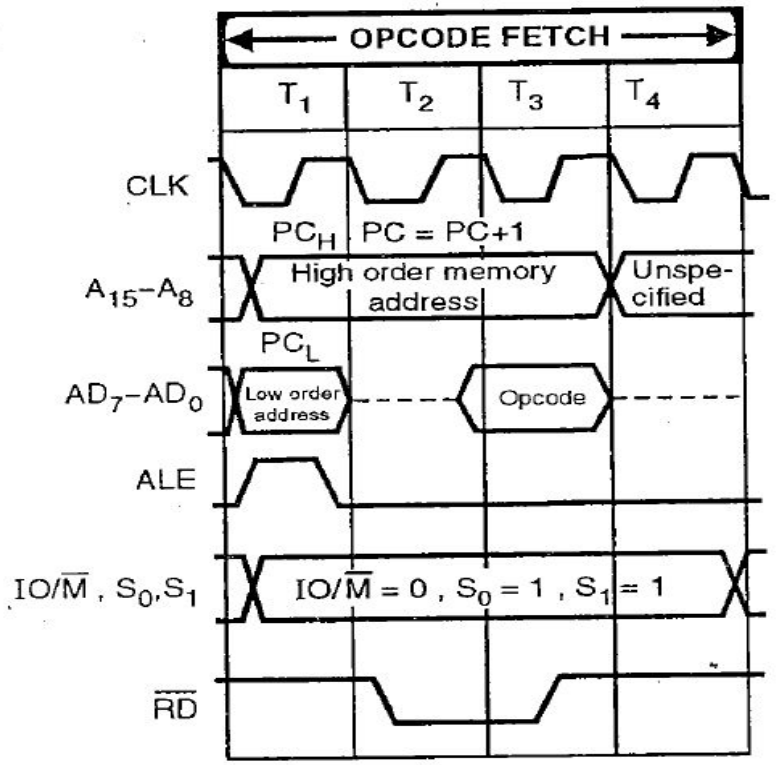
- These instructions perform logical operations on data stored in registers, memory and status flags.
- The logical operations are:
  - AND
  - OR
  - XOR
  - Rotate
  - Compare
  - Complement

1. ANA r

AND register with accumulator.

[A] ← [A] ∧ [r]

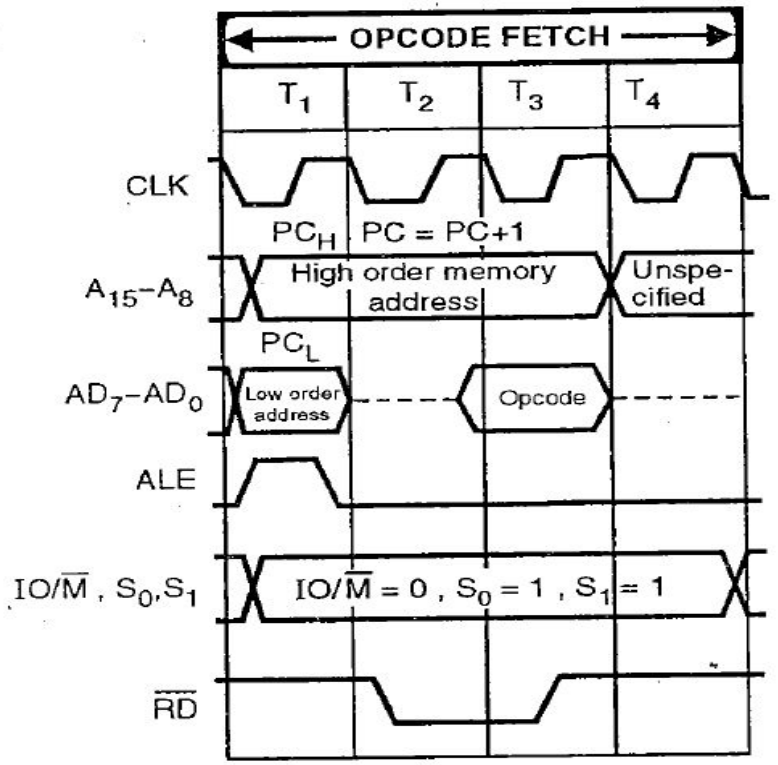
States: 4
Flags: all
Addressing: Register
Machine Cycles: 1



2. ORA r

OR register with accumulator.
[A] ← [A] V [r]

States: 4
Flags: all
Addressing: Register
Machine Cycles: 1

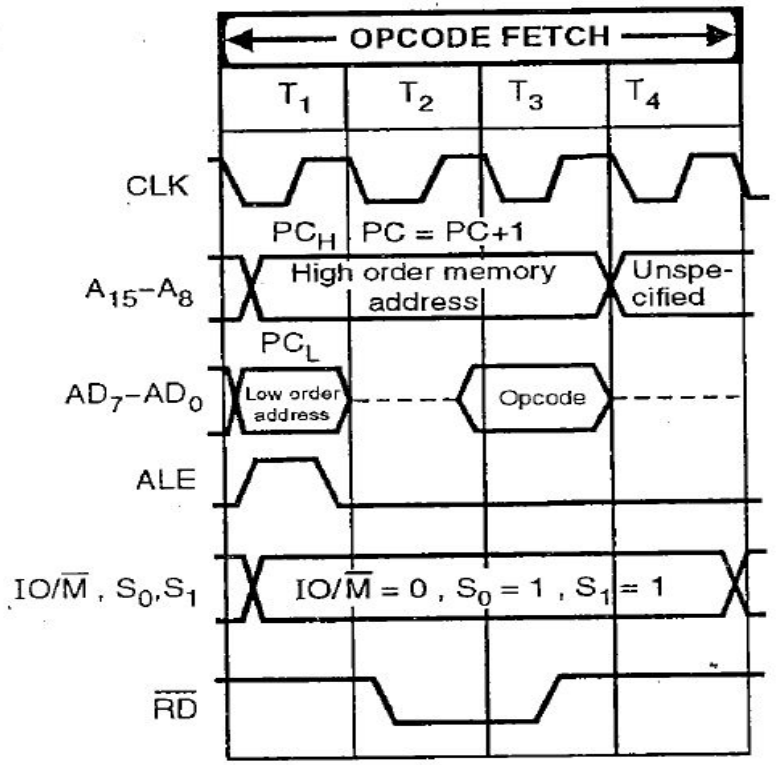


3. XRA r

Exclusive-OR register with accumulator.

[A] ← [A] ∀ [r]

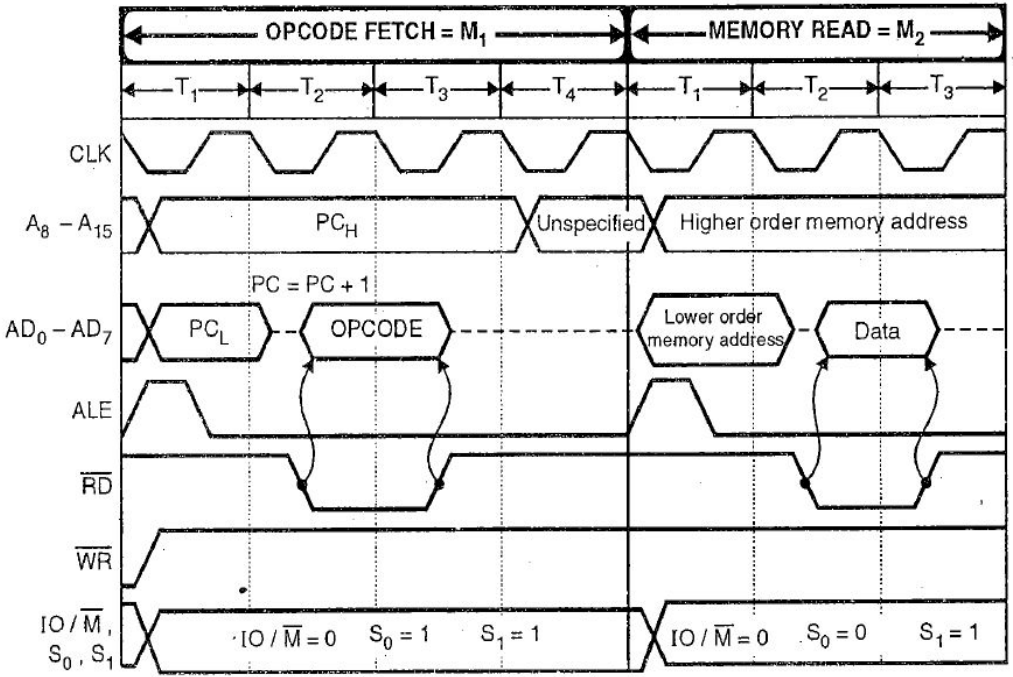
States: 4
Flags: all
Addressing: Register
Machine Cycles: 1



4. ANA M

AND memory with accumulator.
[A] ← [A] ∧ [H-L]

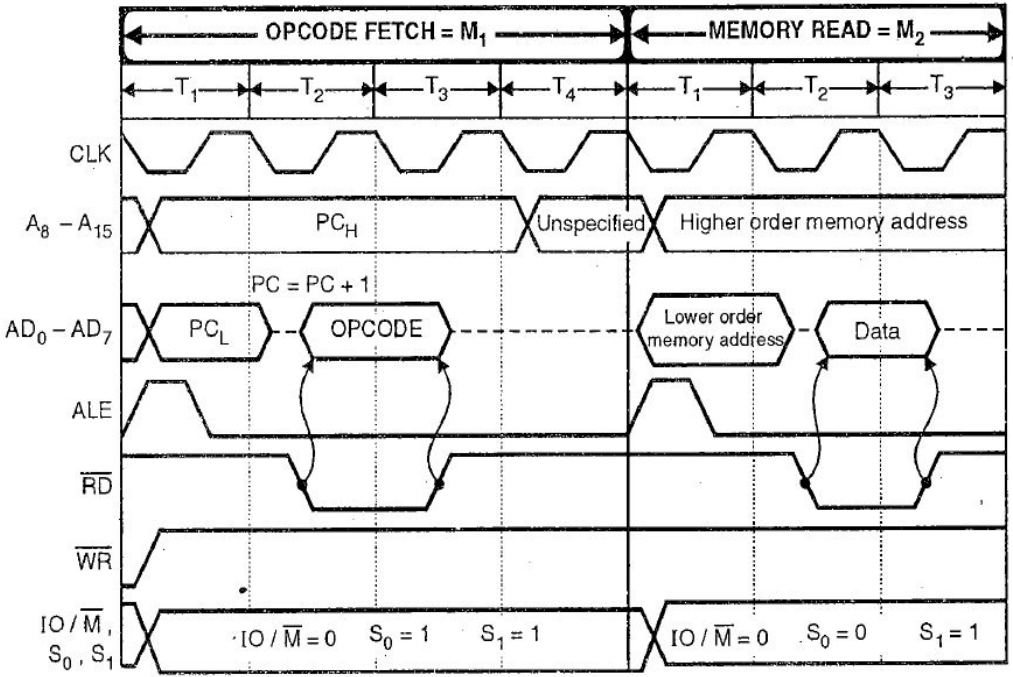
States: 7
Flags: all
Addressing: Register Indirect
Machine Cycles: 2



5. ORA M

OR memory with accumulator.
[A] ← [A] V [H-L]

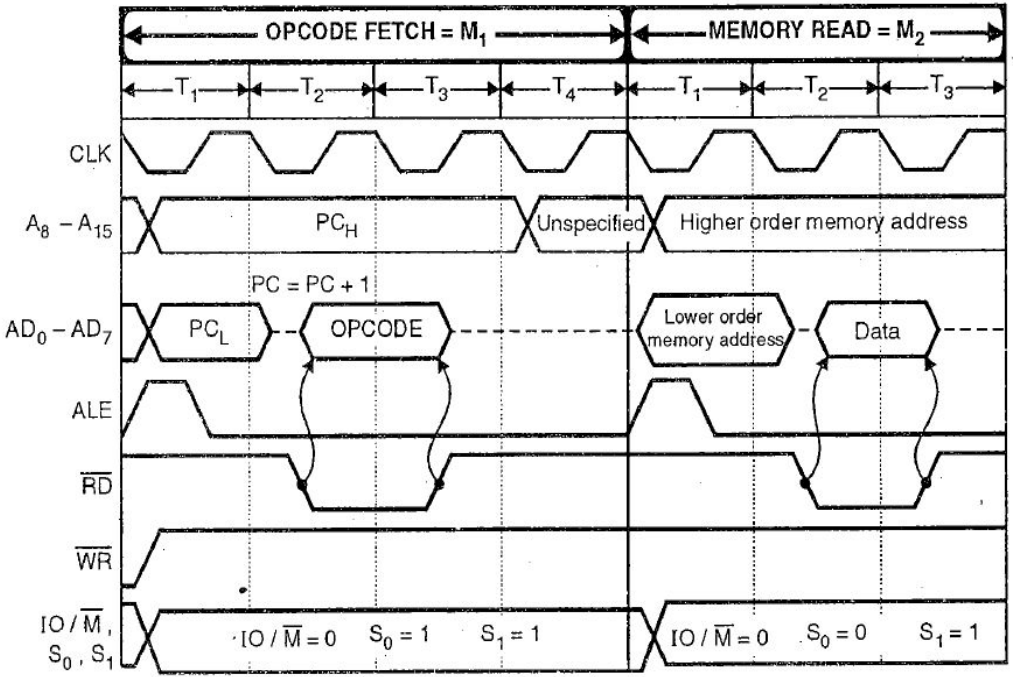
States: 7
Flags: all
Addressing: Register Indirect
Machine Cycles: 2



6. XRA M

Exclusive-OR memory with accumulator.
[A] ← [A] ∀ [H-L]

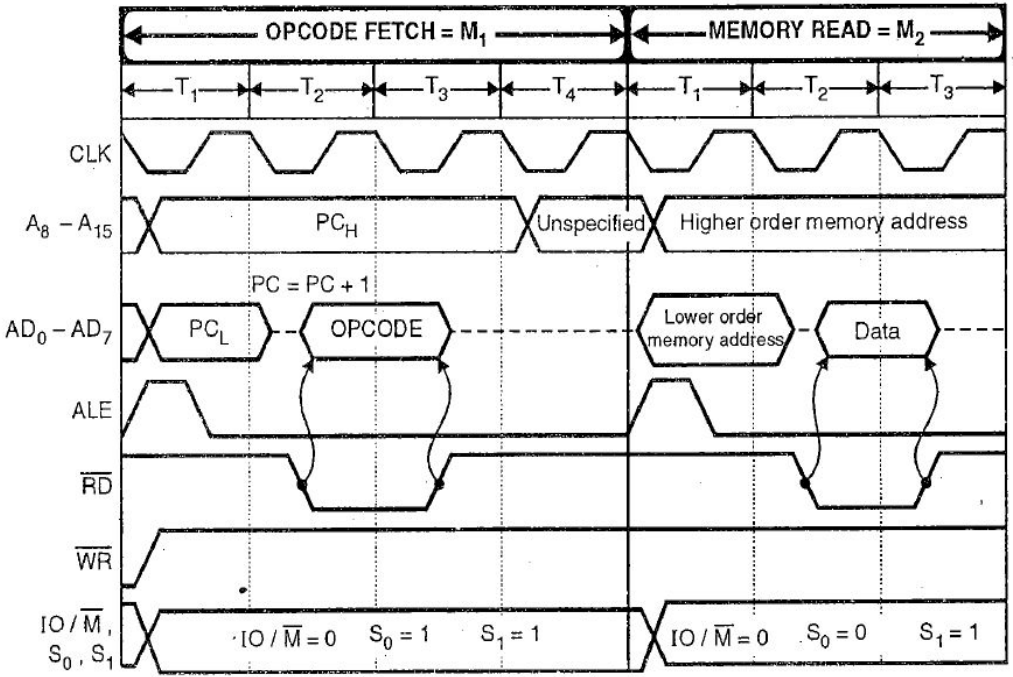
States: 7
Flags: all
Addressing: Register Indirect
Machine Cycles: 2



7. ANI data

AND immediate data with accumulator.
[A] ← [A] ∧ data

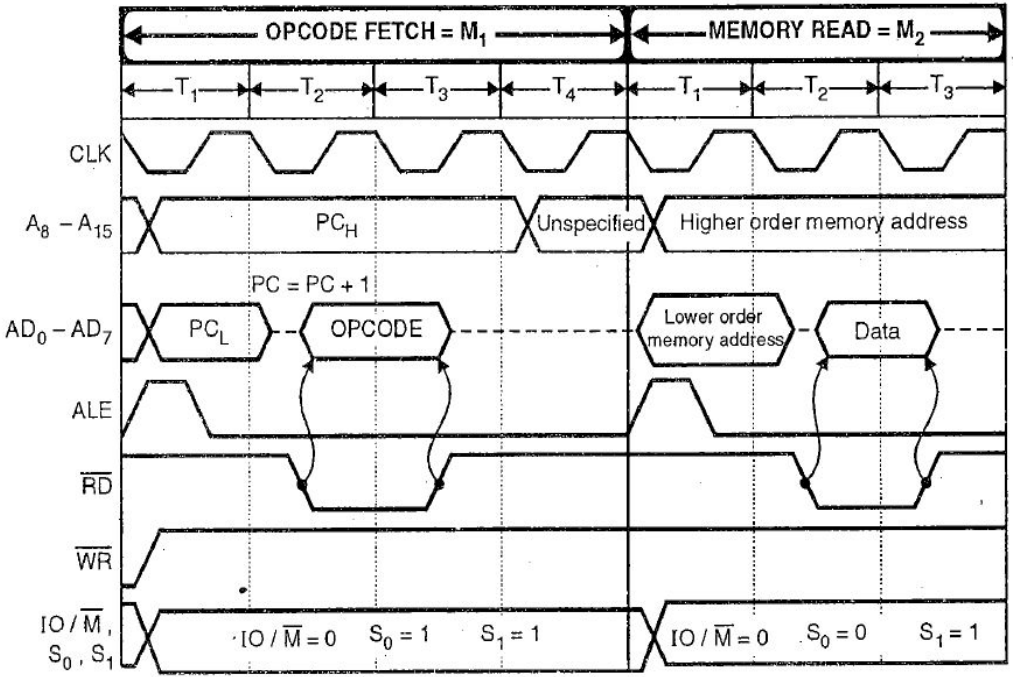
States: 7
Flags: all
Addressing: Immediate
Machine Cycles: 2



8. ORI data

OR immediate data with accumulator.
[A] ← [A] V data

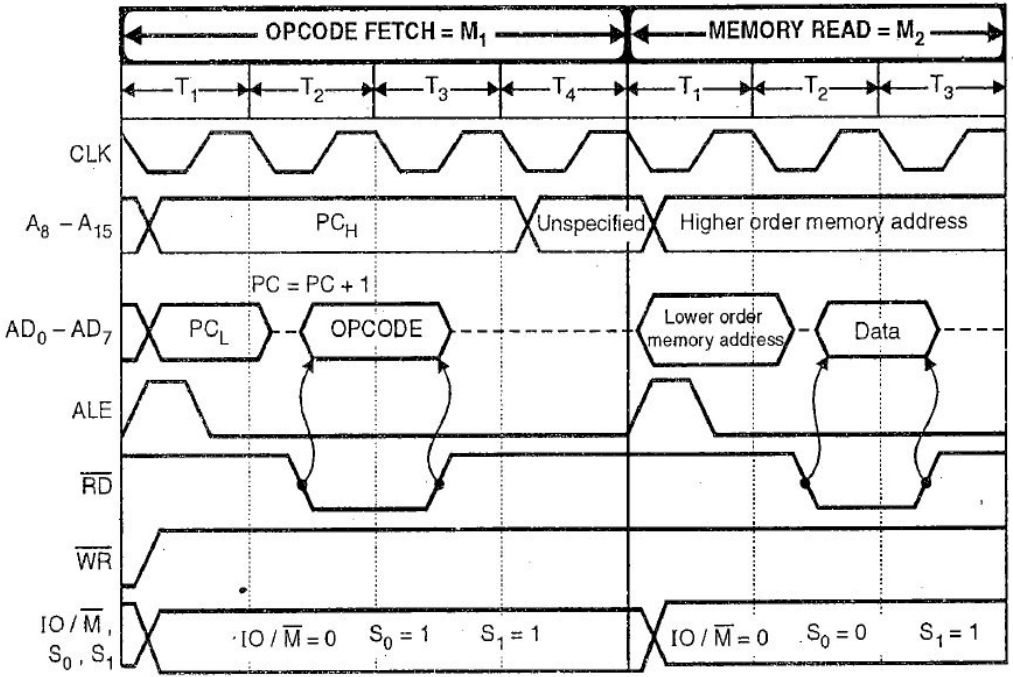
States: 7
Flags: all
Addressing: Immediate
Machine Cycles: 2



9. XRI data

Exclusive-OR immediate data with accumulator.
[A] ← [A] ∀ data

States: 7
Flags: all
Addressing: Immediate
Machine Cycles: 2

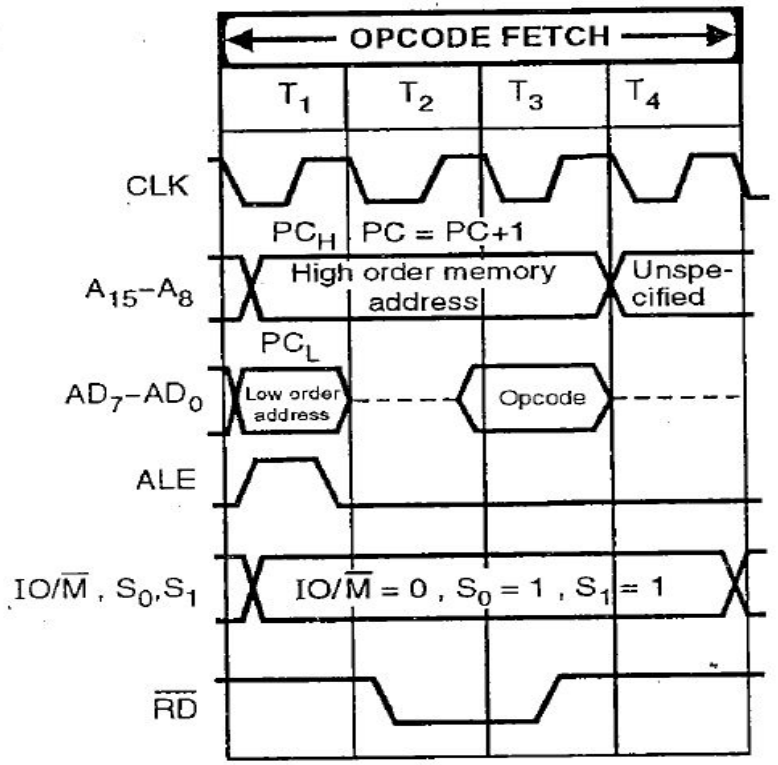


10. CMA

Complement the accumulator.

$[A] \leftarrow [\bar{A}]$

States: 4
Flags: None
Addressing: Implicit
Machine Cycles: 1



11. CMC

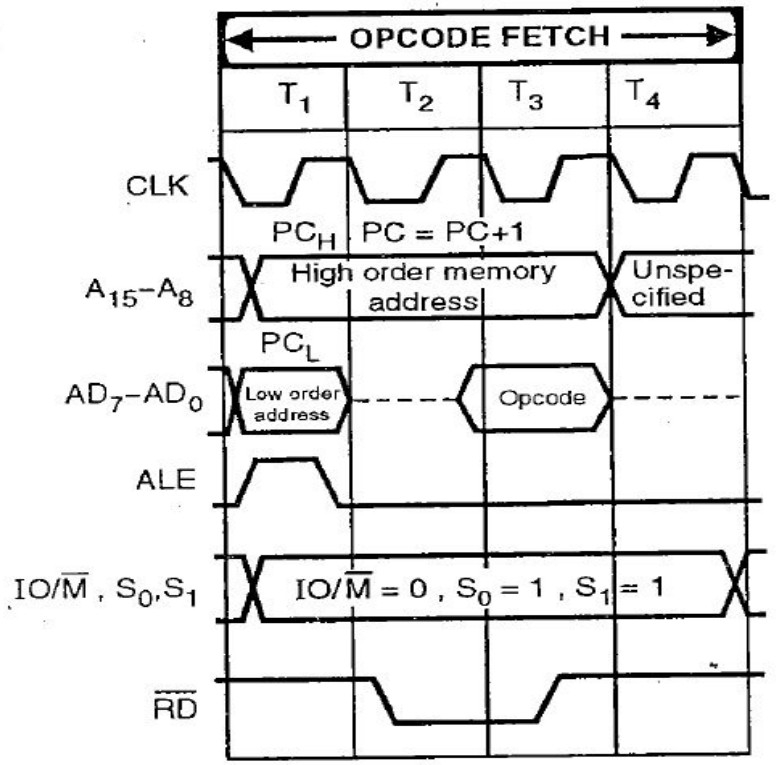
Complement the carry status.

[CS] ← [$\overline{CS}$]

States: 4

Flags: CS

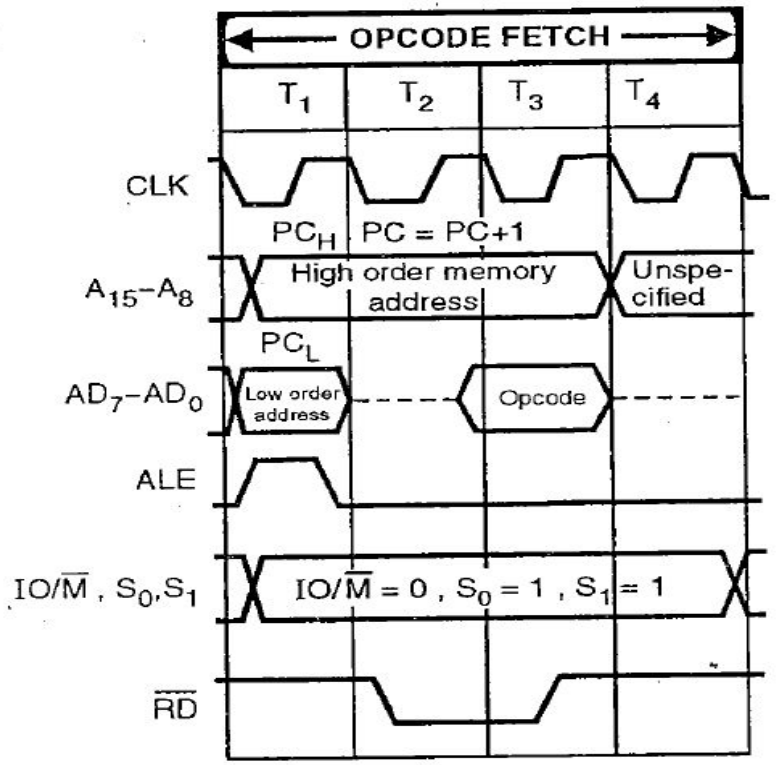
Machine Cycles: 1



12. STC

Set carry status.
[CS] ← 1

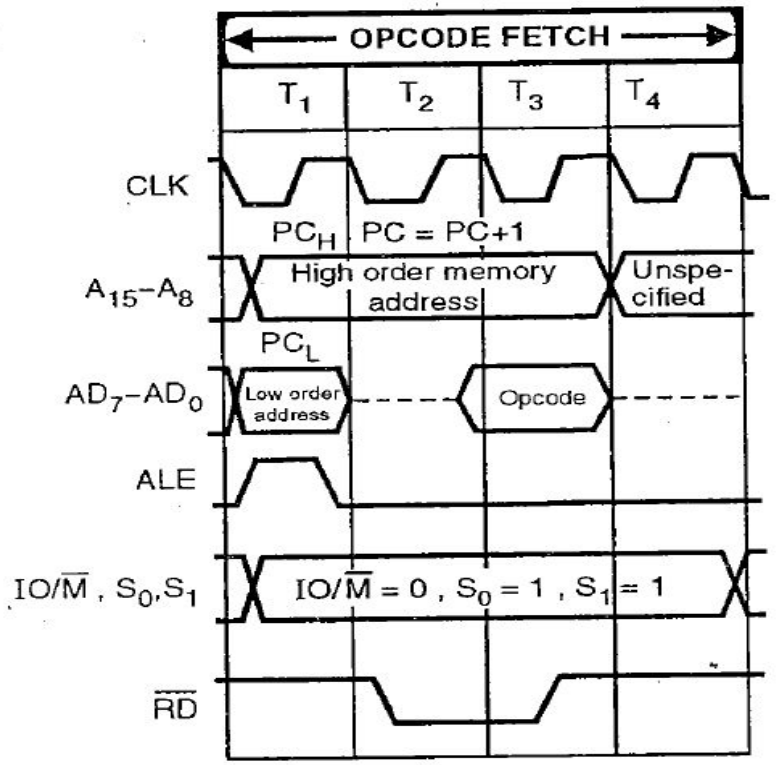
States: 4
Flags: CS
Machine Cycles: 1



13. CMP r

Compare register with accumulator.
[A] –[r]

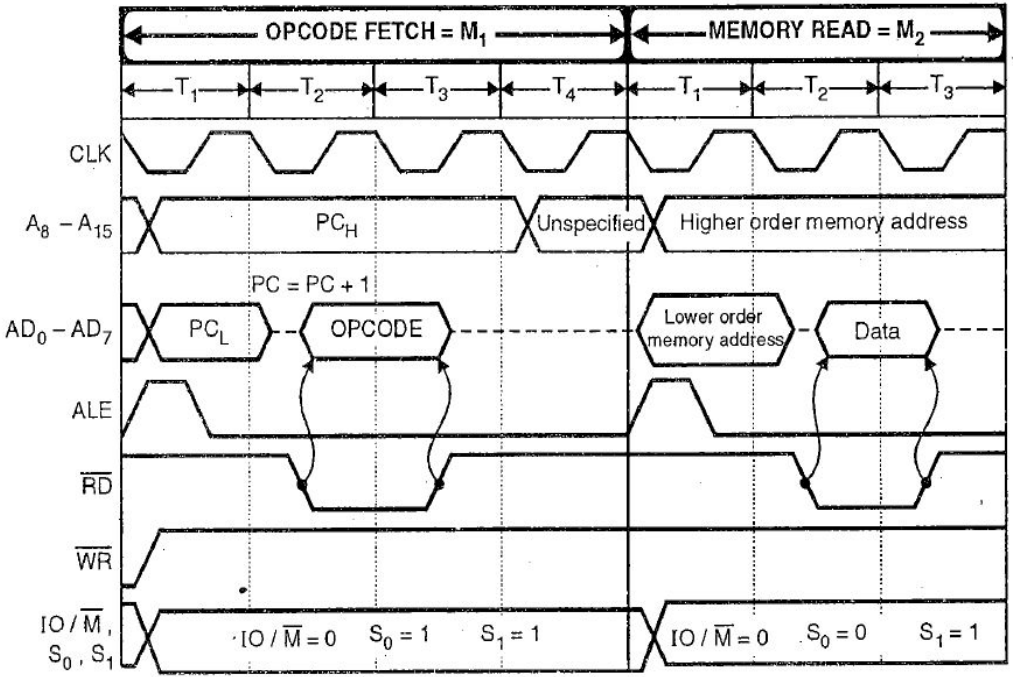
States: 4
Flags: all
Addressing: Register
Machine Cycles: 1



14. CMP M

Compare memory with accumulator.
[A] –[H-L]

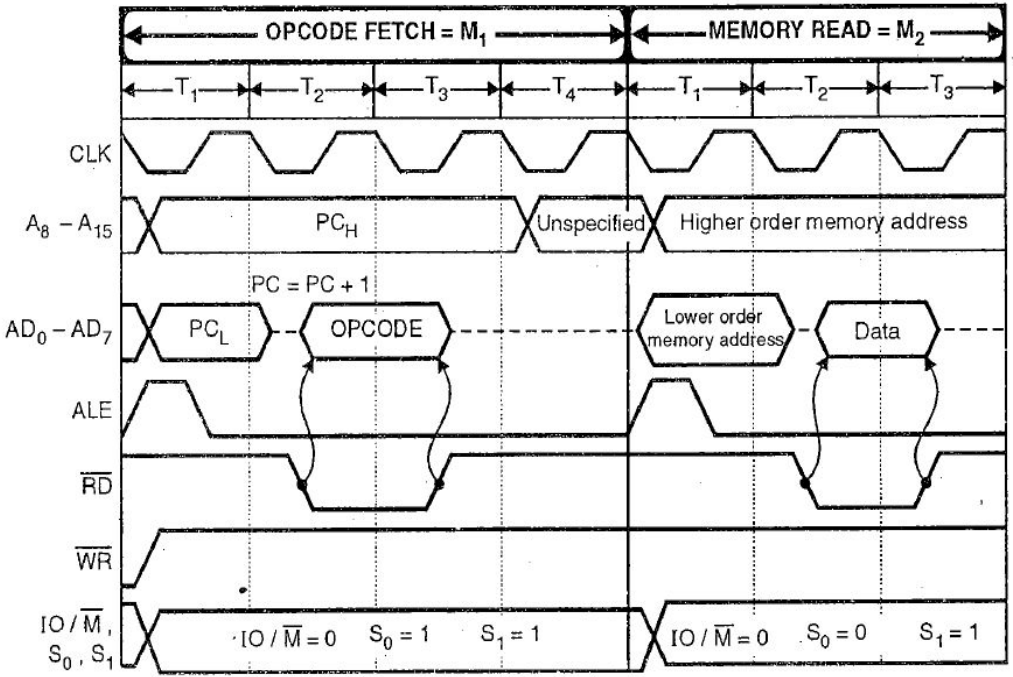
States: 7
Flags: all
Addressing: Register Indirect
Machine Cycles: 2



15. CPI data

Compare immediate data with accumulator.
[A] – data

States: 7
Flags: all
Addressing: Immediate
Machine Cycles: 2



16. RLC

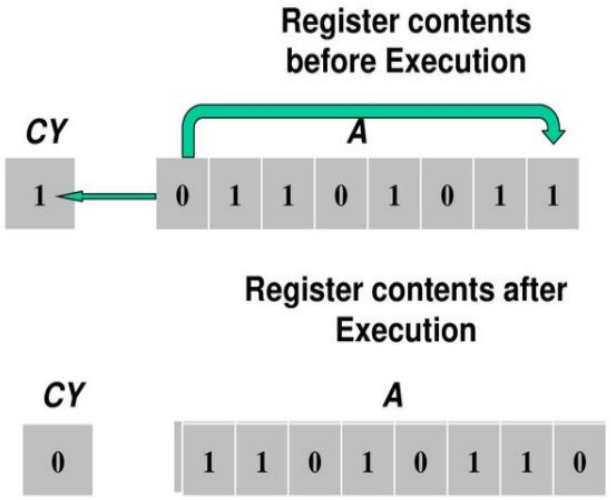
Rotate accumulator left.

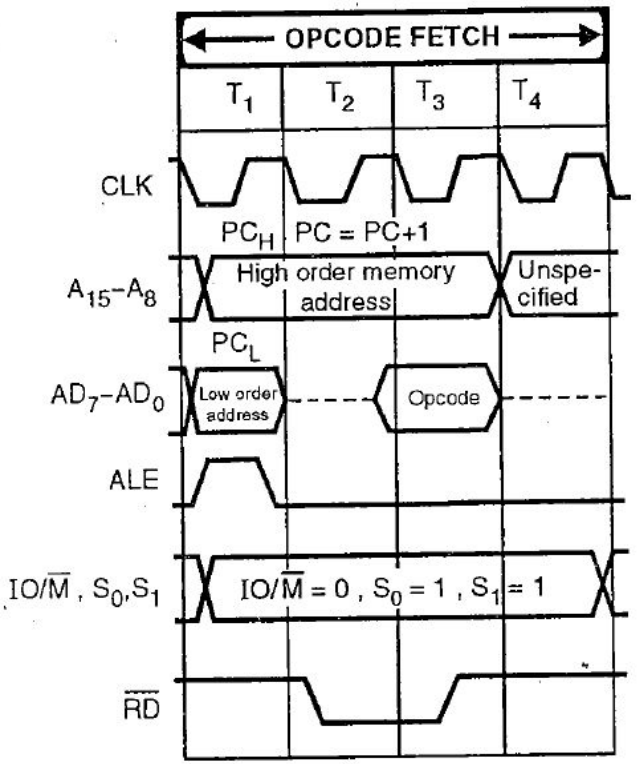
States: 4

Flags: CS

Addressing: Implicit

Machine Cycles: 1





17. RRC

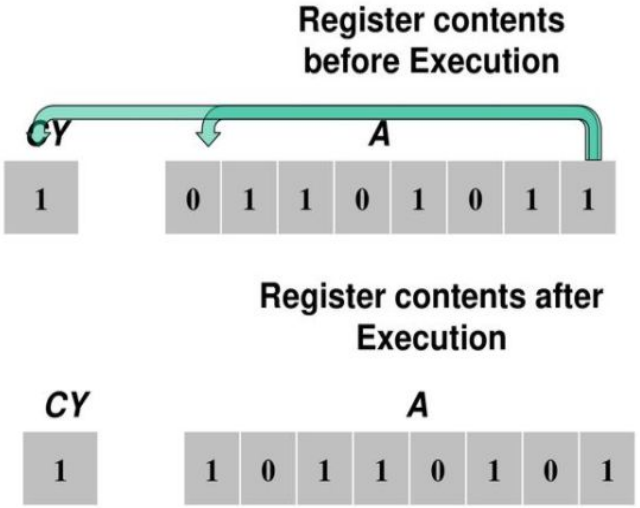
Rotate accumulator right.

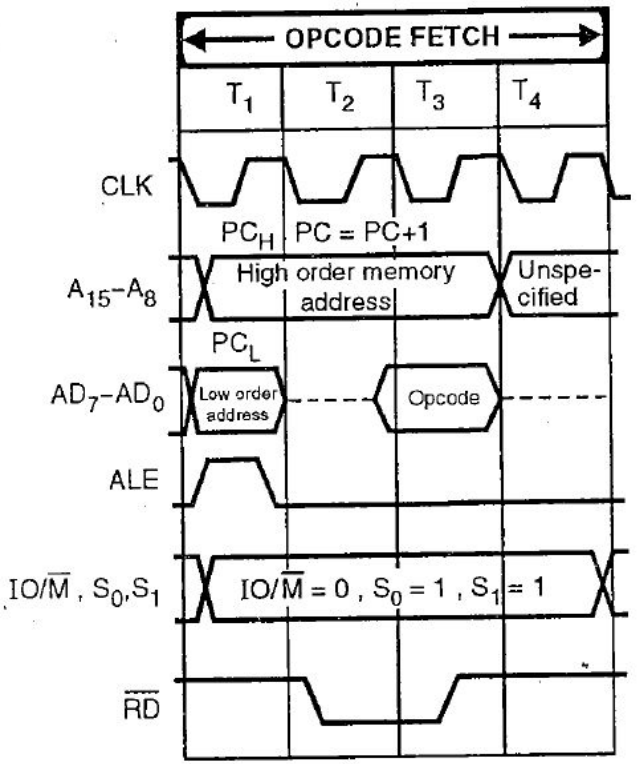
States: 4

Flags: CS

Addressing: Implicit

Machine Cycles: 1





18. RAL

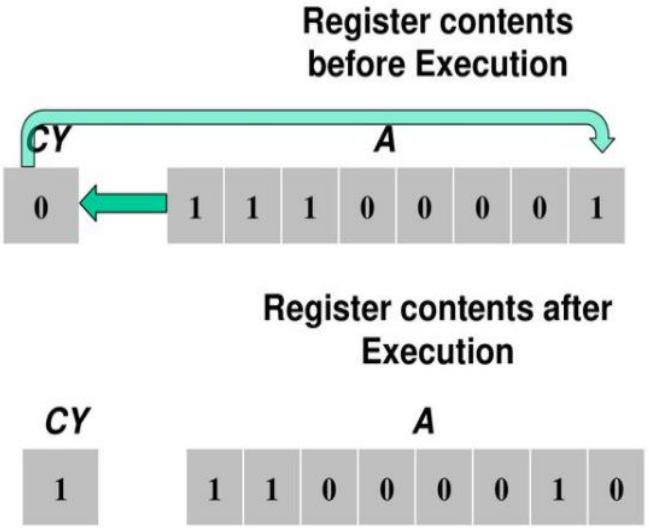
Rotate accumulator left through carry.

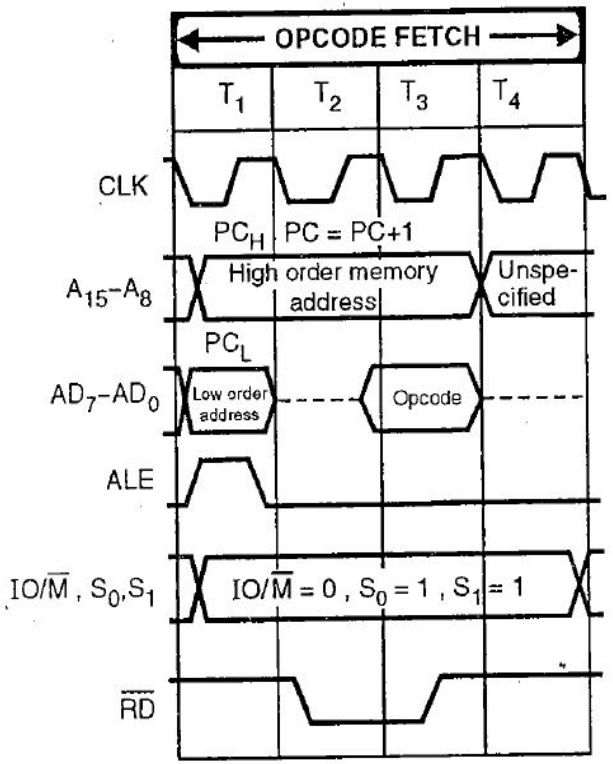
States: 4

Flags: CS

Addressing: Implicit

Machine Cycles: 1





19. RAR

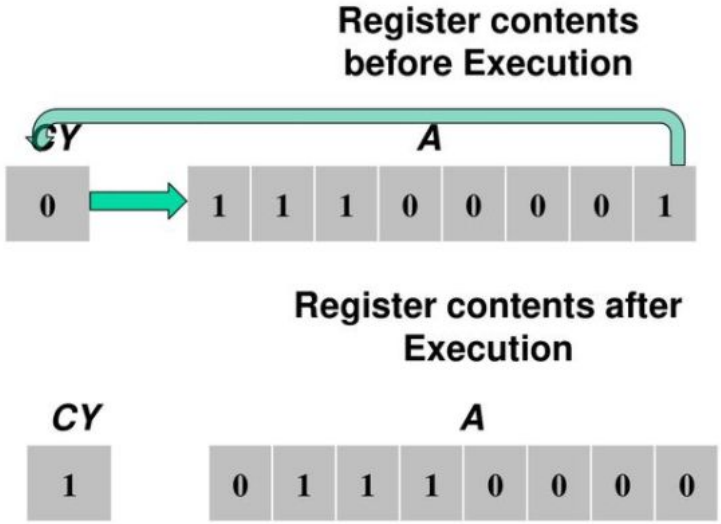
Rotate accumulator right through carry.

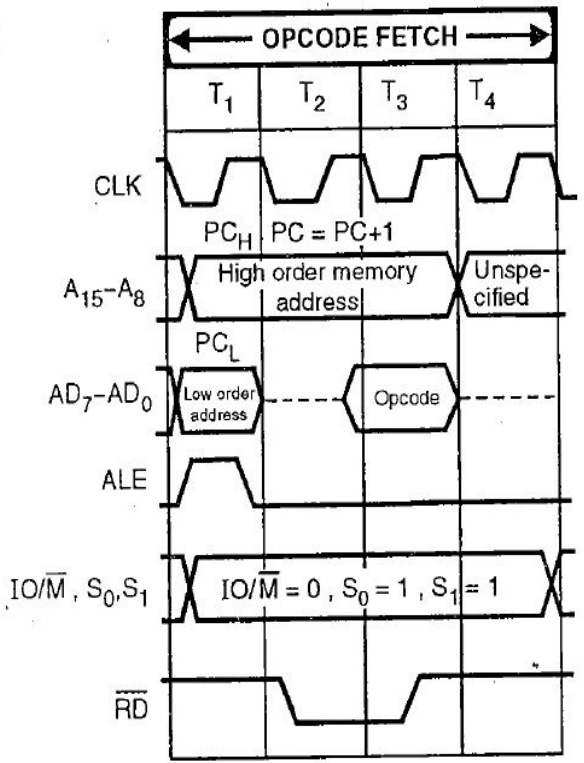
States: 4

Flags: CS

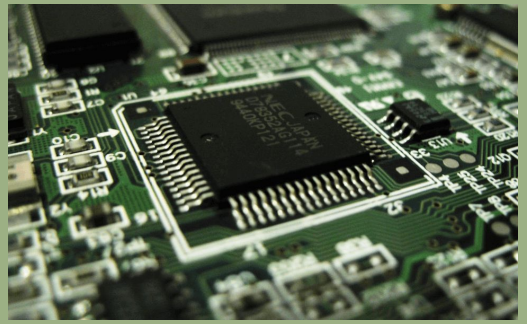
Addressing: Implicit

Machine Cycles: 1





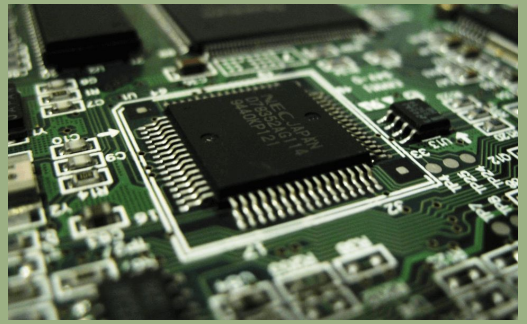

Thank you!



8085 Microprocessor

Dr. Manju Khurana
Assistant Professor, CSED
TIET, Patiala
manju.khurana@thapar.edu


Thank you!



8085 Microprocessor

Dr. Manju Khurana
Assistant Professor, CSED
TIET, Patiala
manju.khurana@thapar.edu

Branch Control Group

- The instructions of this group change the normal sequential flow of the program.
- These instructions alter either unconditionally or conditionally.

3 categories are there:

1. **JUMP instructions:** JMP addr., Conditional JMP instructions, PCHL
2. **Call and return instructions:** Call addr., Conditional call instructions, RET, Conditional RET instructions
3. **Restart instructions:** RST N

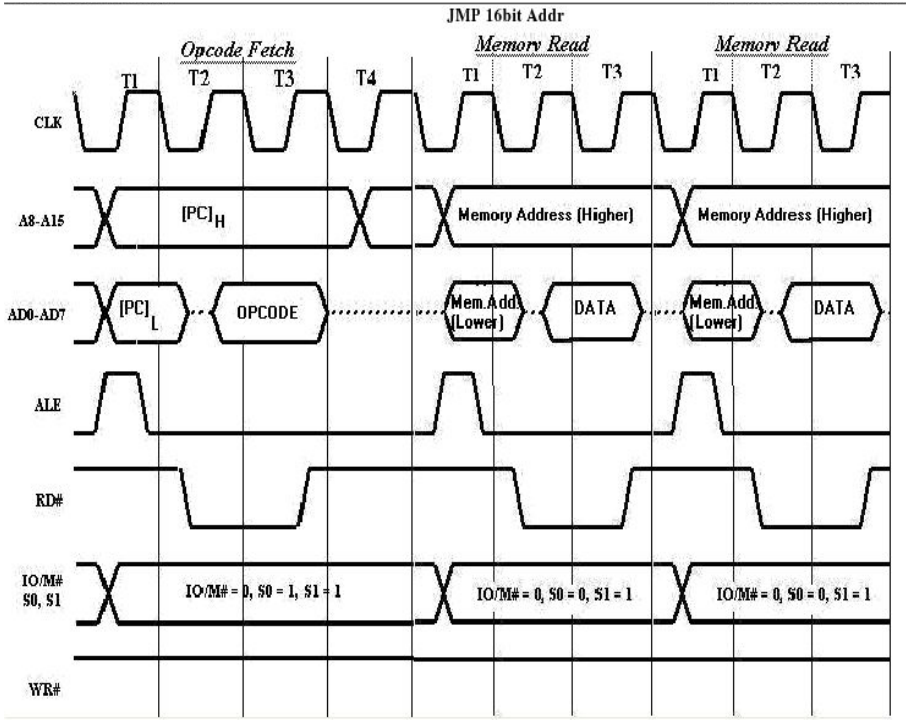
1. Jump Conditionally

Opcode	Description	Status Flags
JC	Jump if Carry	CY = 1
JNC	Jump if No Carry	CY = 0
JP	Jump if Positive	S = 0
JM	Jump if Minus	S = 1
JZ	Jump if Zero	Z = 1
JNZ	Jump if No Zero	Z = 0
JPE	Jump if Parity Even	P = 1
JPO	Jump if Parity Odd	P = 0

1. JMP addr (label)

Unconditional jump: jump to the Instruction specified by the address
[PC] ← Label

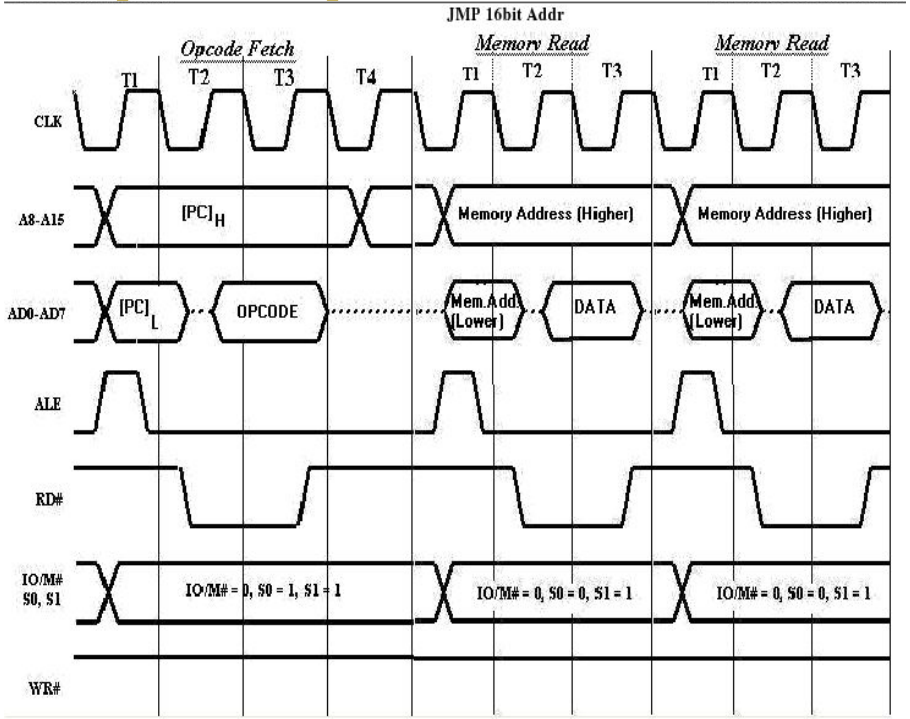
States: 10
Flags: None
Addressing: Immediate
Machine Cycles: 3



2. Conditional JMP addr (label)

[PC] ← address (Label)

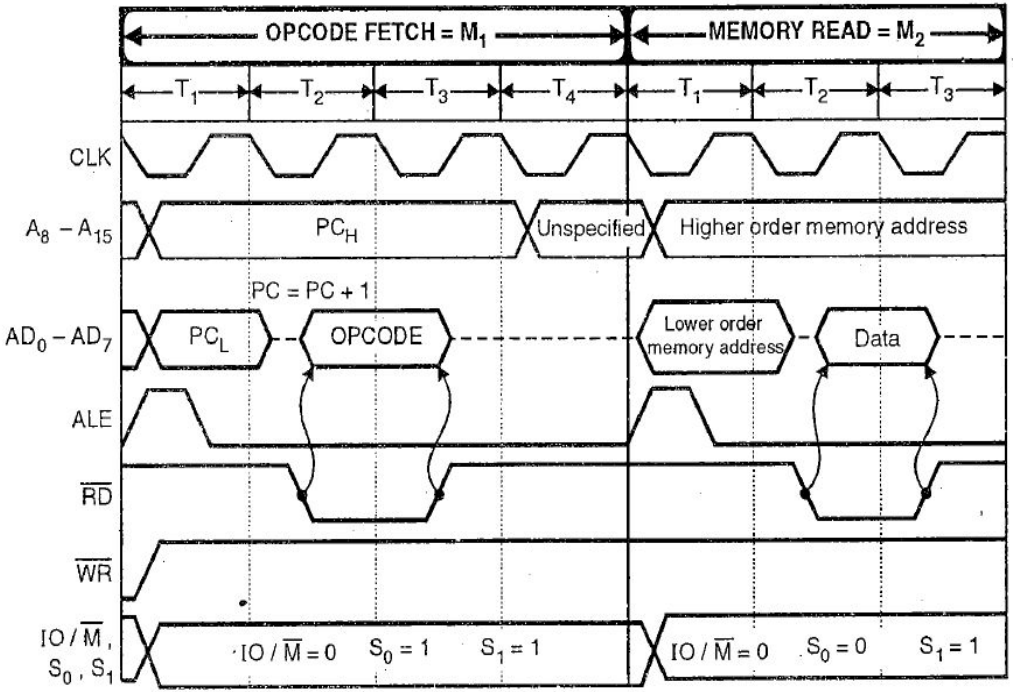
States: 7/10
Flags: None
Addressing: Immediate
Machine Cycles: 2/3



2. Conditional JMP addr (label)contd.

[PC] ← address (Label)

States: 7/10
Flags: None
Addressing: Immediate
Machine Cycles: 2/3



2. Call Conditionally

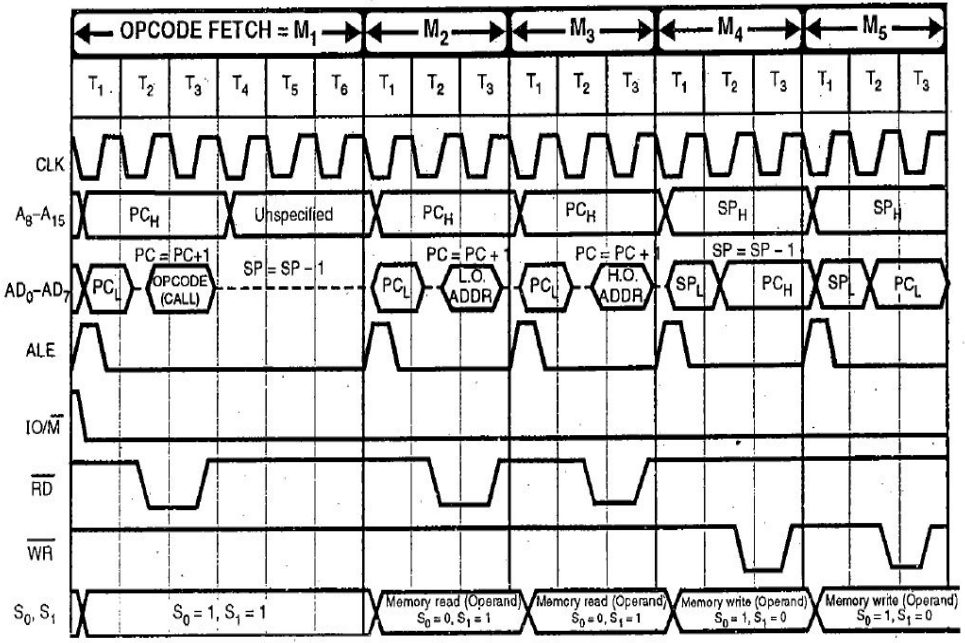
Opcode	Description	Status Flags
CC	Call if Carry	CY = 1
CNC	Call if No Carry	CY = 0
CP	Call if Positive	S = 0
CM	Call if Minus	S = 1
CZ	Call if Zero	Z = 1
CNZ	Call if No Zero	Z = 0
CPE	Call if Parity Even	P = 1
CPO	Call if Parity Odd	P = 0

1. CALL addr (label)

Unconditional call: call the Sub-routine identified by the address

$(PC)_H \rightarrow (SP - 1)$
$(PC)_L \rightarrow (SP - 2)$
$(SP - 2) \rightarrow SP$
$[PC] \leftarrow$ address (Label)

States: 18
Flags: None
Addressing: Immediate/ Register Indirect
Machine Cycles: 5

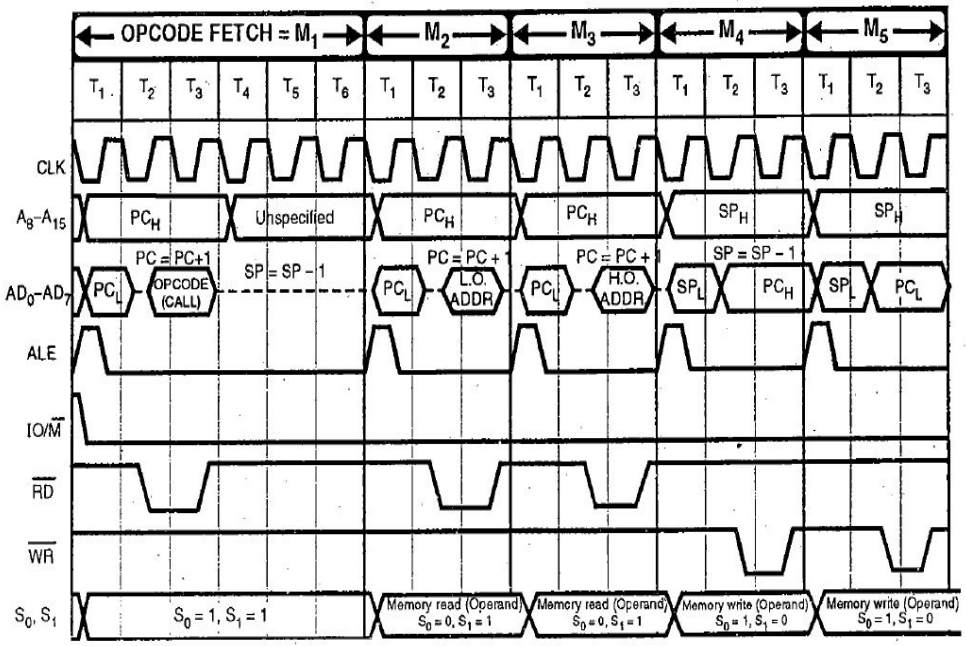


Timing diagram of CALL instruction

2. Conditional CALL addr (label)

$(PC)_H \rightarrow (SP - 1)$
$(PC)_L \rightarrow (SP - 2)$
$(SP - 2) \rightarrow SP$
$[PC] \leftarrow$ address (Label)

States: 9/18
Flags: None
Addressing: Immediate/
Register Indirect
Machine Cycle: 2/5

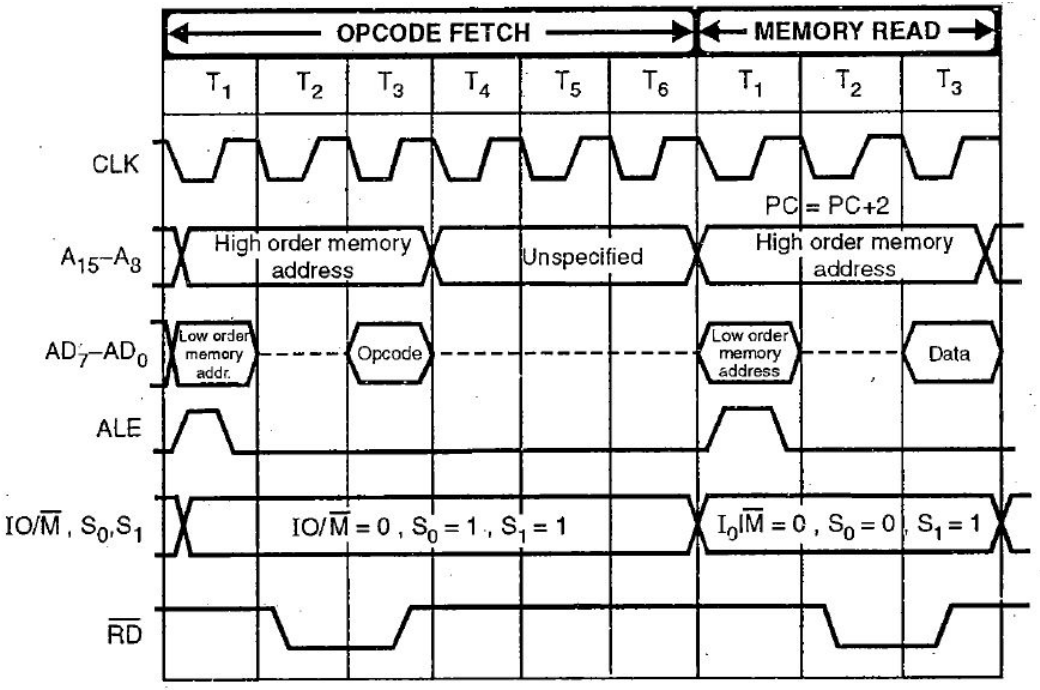


Timing diagram of CALL instruction

2. Conditional CALL addr (label)contd.

$(PC)_H \rightarrow (SP - 1)$
$(PC)_L \rightarrow (SP - 2)$
$(SP - 2) \rightarrow SP$
$[PC] \leftarrow$ address (Label)

States: 9/18
Flags: None
Addressing: Immediate/
Register Indirect
Machine Cycle: 2/5



3. Return Conditionally

Opcode	Description	Status Flags
RC	Return if Carry	CY = 1
RNC	Return if No Carry	CY = 0
RP	Return if Positive	S = 0
RM	Return if Minus	S = 1
RZ	Return if Zero	Z = 1
RNZ	Return if No Zero	Z = 0
RPE	Return if Parity Even	P = 1
RPO	Return if Parity Odd	P = 0

1. RET

Return from subroutine

$(SP) \rightarrow (PC)_L$

$(SP + 1) \rightarrow (PC)_H$

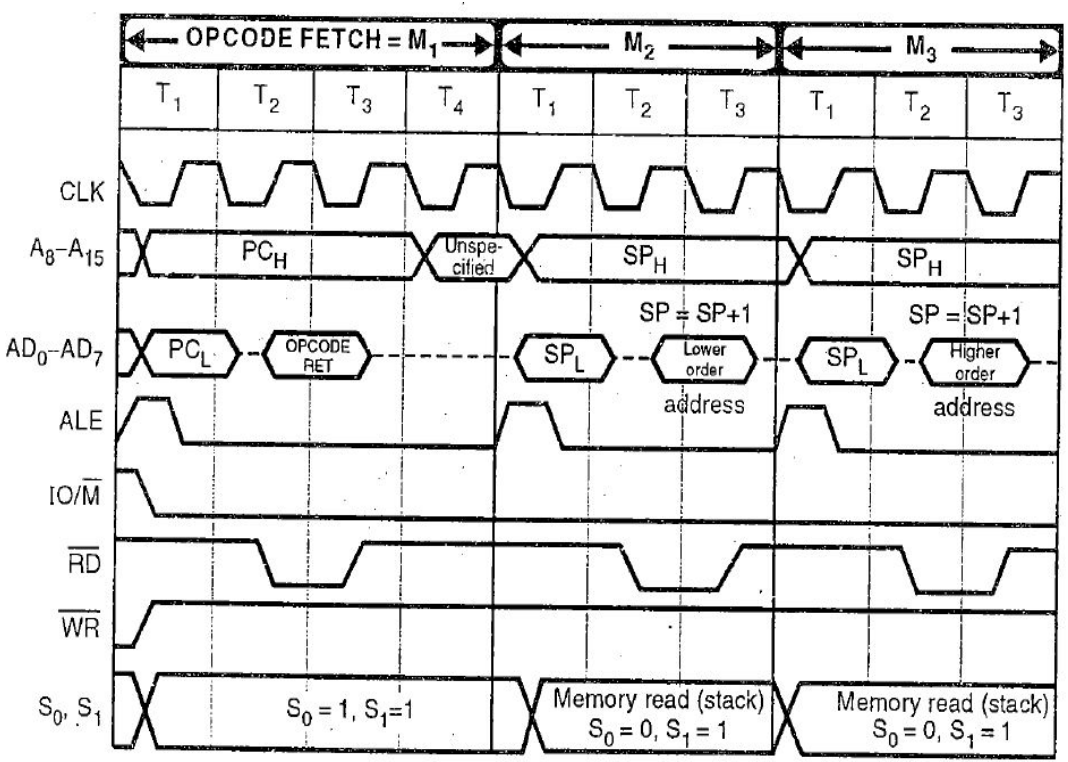
$(SP + 2) \rightarrow SP$

States: 10

Flags: None

Addressing: Register Indirect

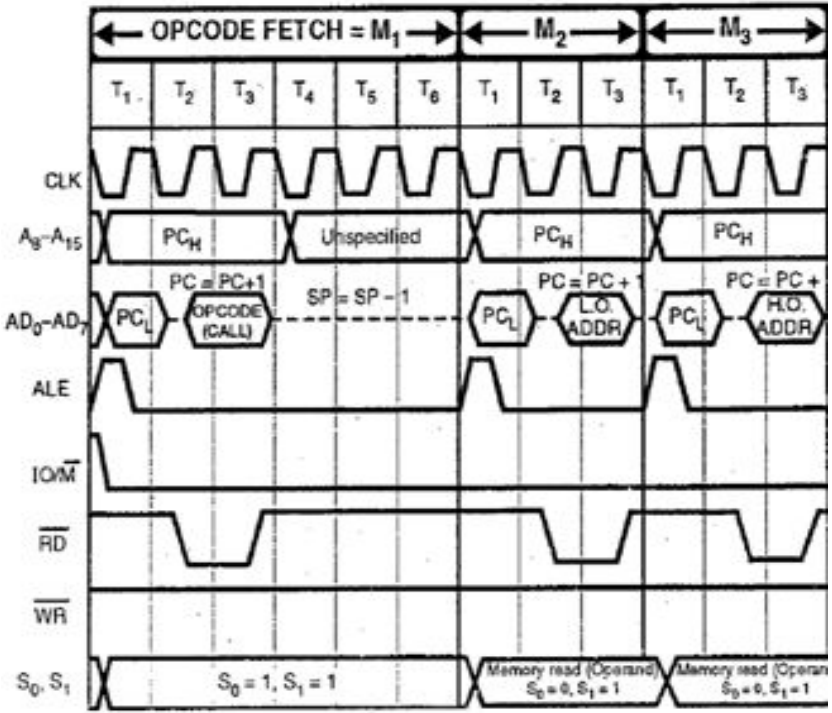
Machine Cycles: 3



2. Conditional RET

$(SP) \rightarrow (PC)_L$
$(SP + 1) \rightarrow (PC)_H$
$(SP + 2) \rightarrow SP$

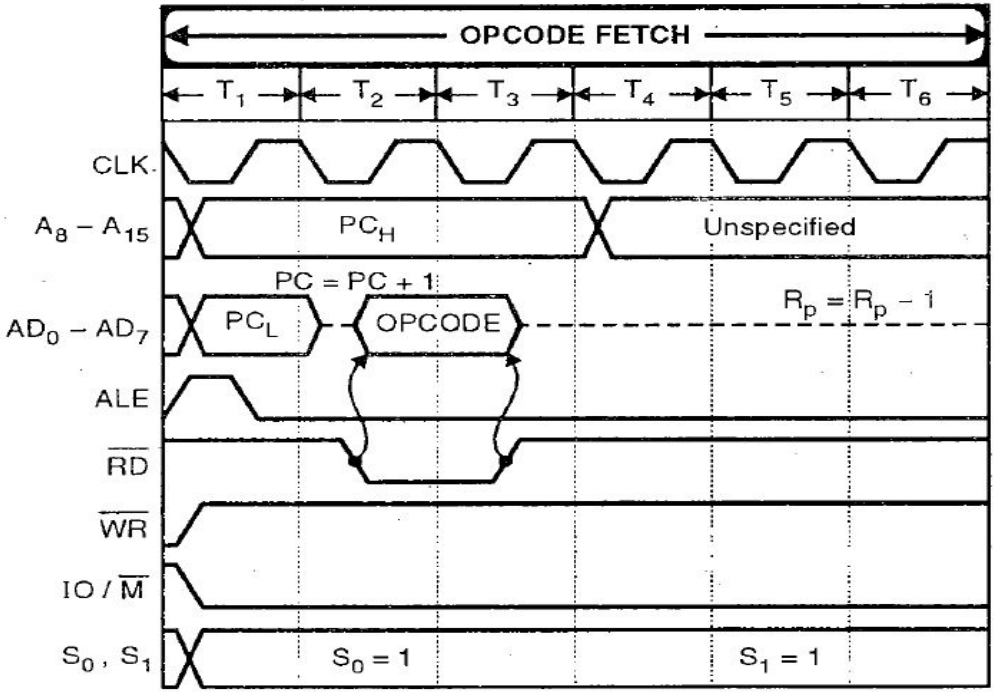
States: 6/12
Flags: None
Addressing: Register Indirect
Machine Cycles: 1/3



2. Conditional RET contd.

$(SP) \rightarrow (PC)_L$
$(SP + 1) \rightarrow (PC)_H$
$(SP + 2) \rightarrow SP$

States: 6/12
Flags: None
Addressing: Register Indirect
Machine Cycles: 1/3



4. RST n (Restart)

States: 12

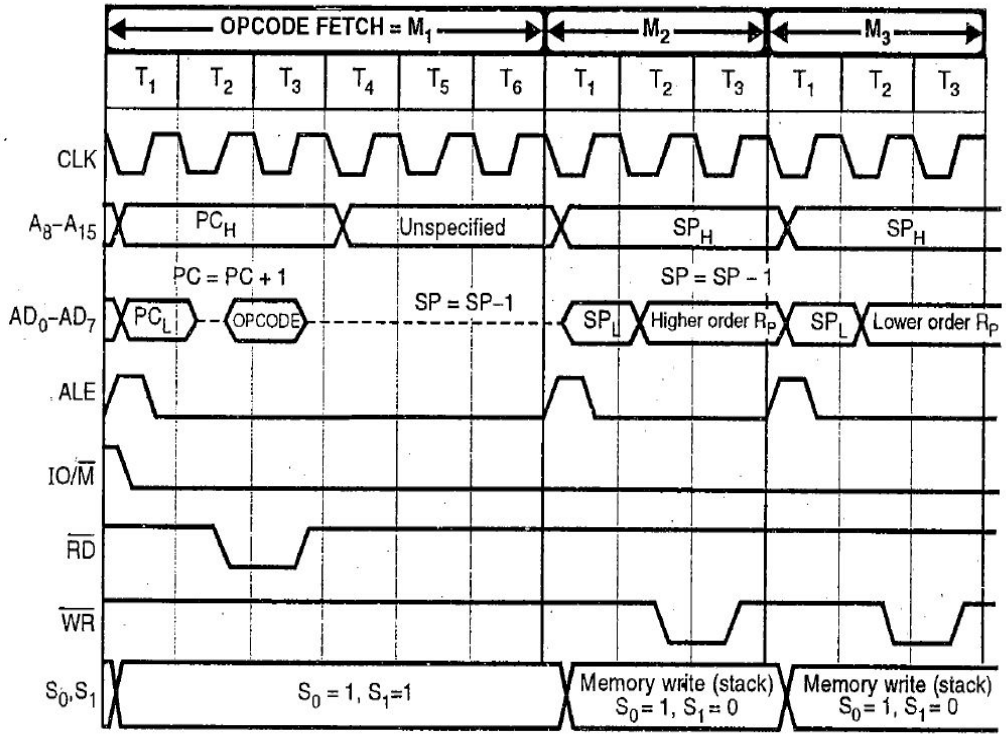
Flags: None

Addressing: Register Indirect

Machine Cycles: 3

Restart Address Table →

Instructions	Restart Address
RST 0	0000 H
RST 1	0008 H
RST 2	0010 H
RST 3	0018 H
RST 4	0020 H
RST 5	0028 H
RST 6	0030 H
RST 7	0038 H



5. PCHL

Jump to address specified by H-L pair

[PC] ← [H-L]

[PCH] ← [H]

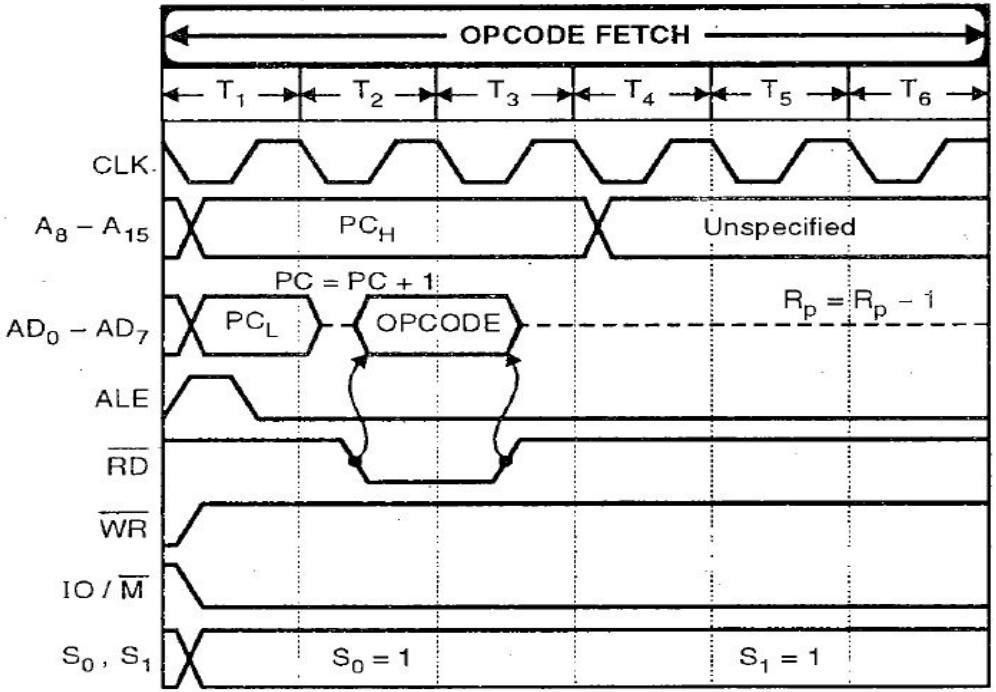
[PCL] ← [L]

States: 6

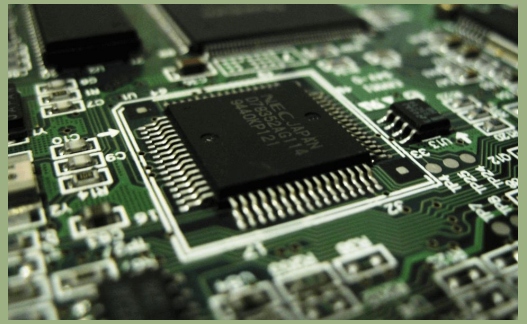
Flags: None

Addressing: Register

Machine Cycles: 1




Thank you!



8085 Microprocessor

Dr. Manju Khurana
Assistant Professor, CSED
TIET, Patiala
manju.khurana@thapar.edu

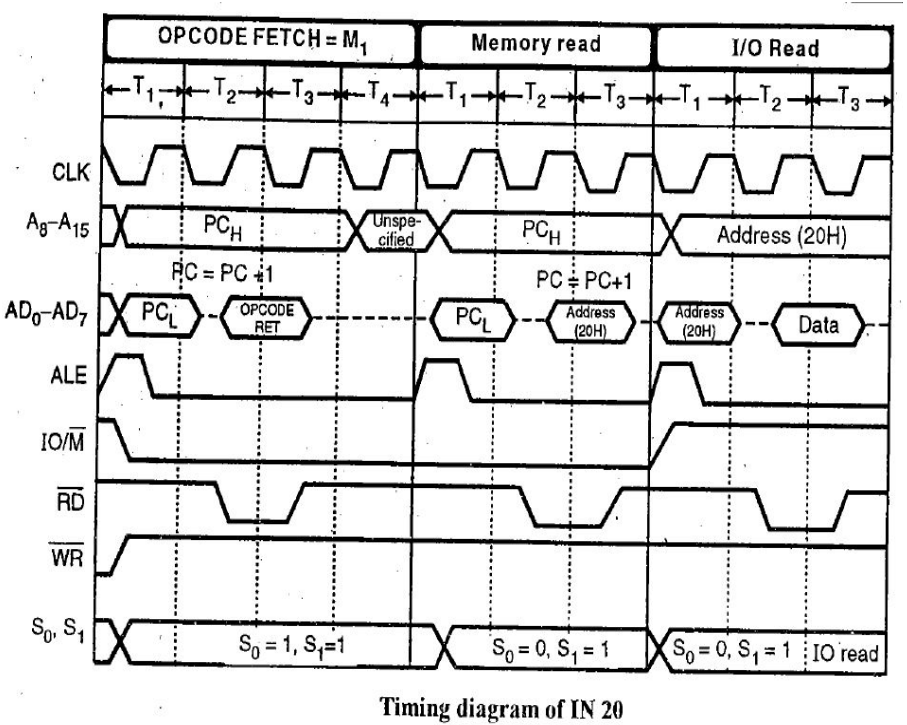
Stack, I/O and Machine Control Group

1. IN port-address

Input to accumulator from I/O port.

[A] ← [port]

States: 10
Flags: None
Addressing: Direct
Machine Cycles: 3

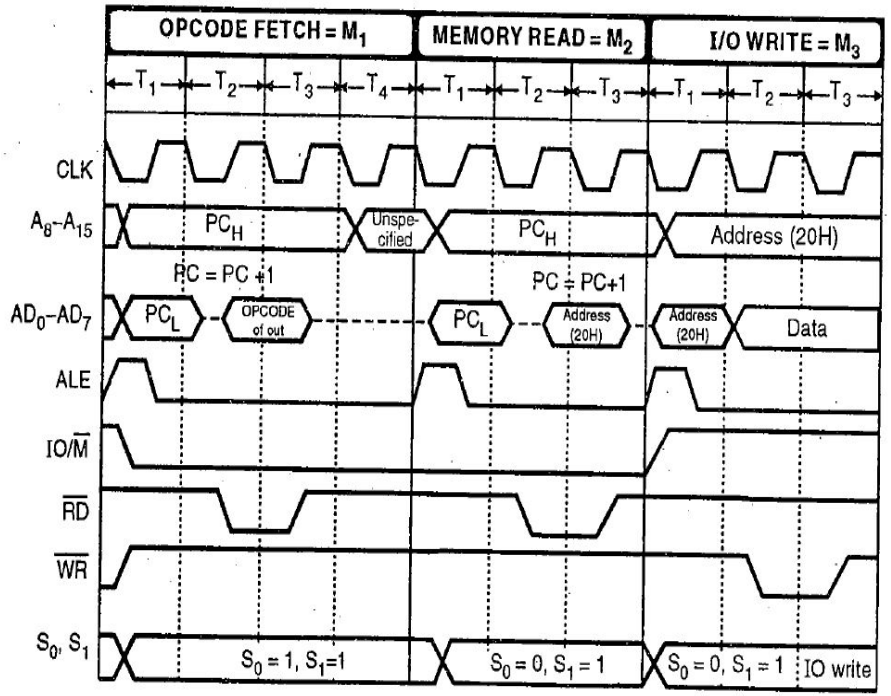


Timing diagram of IN 20

2. OUT port-address

Output from accumulator to I/O port.
[port] ← [A]

States: 10
Flags: None
Addressing: Direct
Machine Cycles: 3

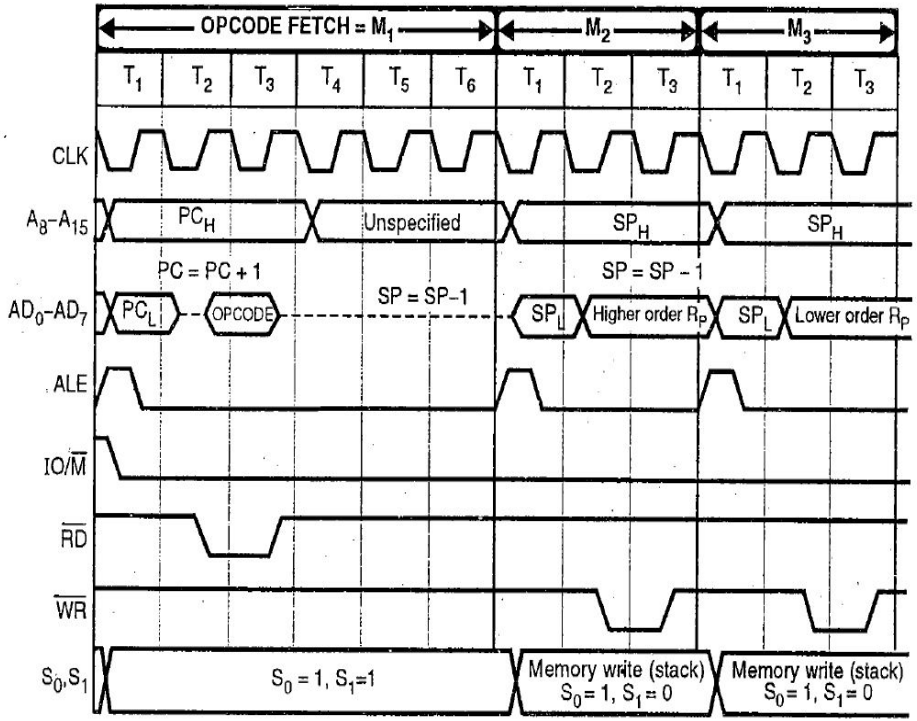


Timing diagram of OUT 30

3. PUSH Rp

Push the content of register pair to stack.
$(Rp)_H \rightarrow (SP - 1)$
$(Rp)_L \rightarrow (SP - 2)$
$(SP - 2) \rightarrow SP$

States: 12
Flags: None
Addressing: Register (Source)
Register Indirect (Destination)
Machine Cycles: 3



4. POP Rp

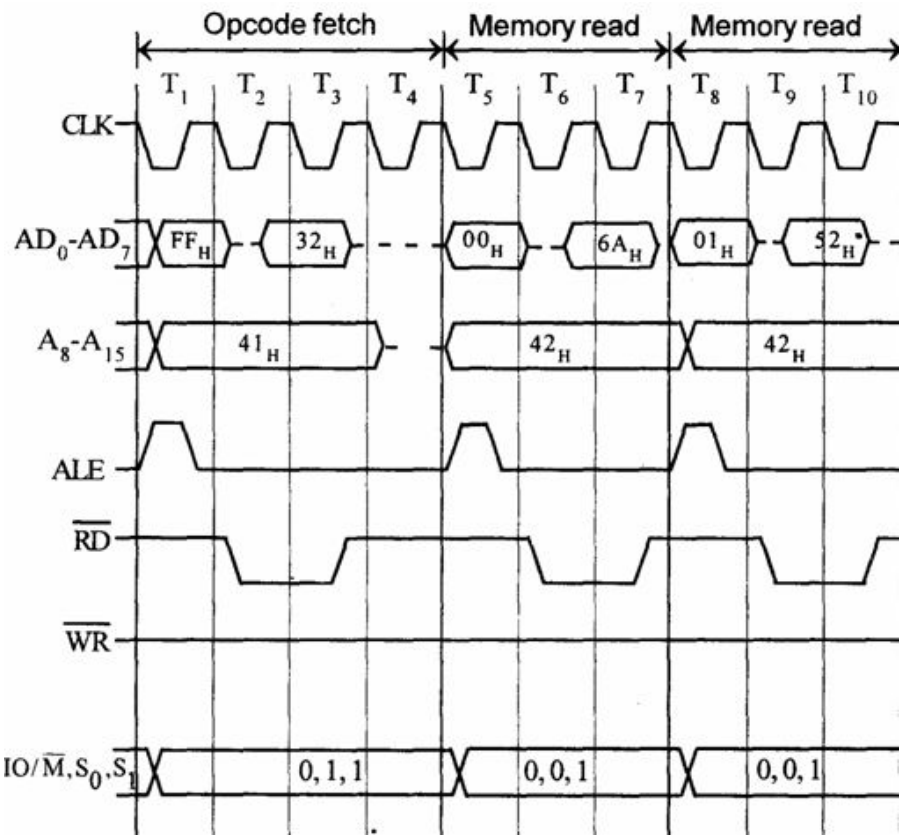
Copy two bytes from the top of the stack into the specified register.

$SP \rightarrow (Rp)_L$

$(SP + 1) \rightarrow (Rp)_H$

$(SP + 2) \rightarrow SP$

States: 10
Flags: None
Addressing: Register (Destination)
Register Indirect (Source)
Machine Cycles: 3



5. PUSH PSW

Push processor status word.

[A] → (SP – 1)

PSW → (SP – 2)

(SP – 2) → SP

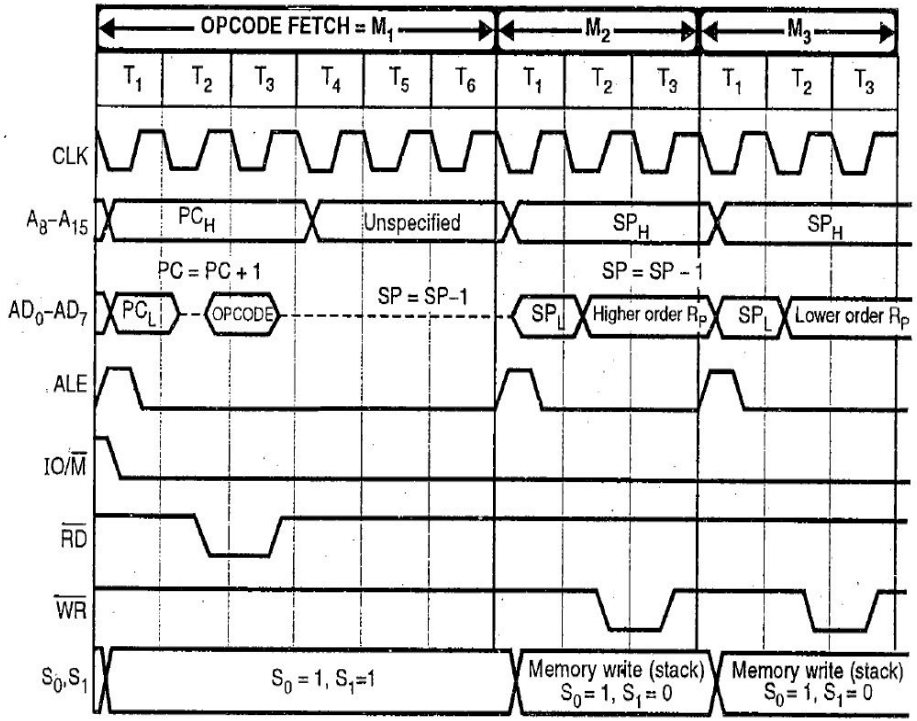
States: 12

Flags: None

Addressing: Register (Source)

Register Indirect (Destination)

Machine Cycles: 3



6. POP PSW

Copy two bytes from the top of the stack into PSW and accumulator.

SP → PSW

(SP + 1) → [A]

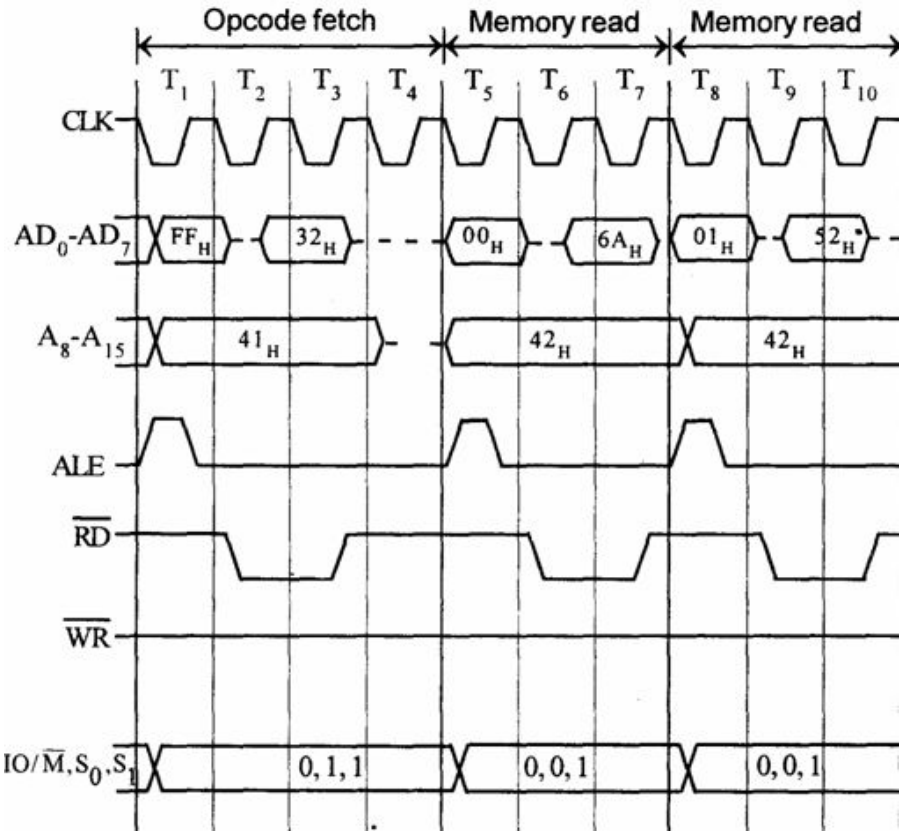
(SP + 2) → SP

States: 10

Flags: None

Addressing: Register Indirect

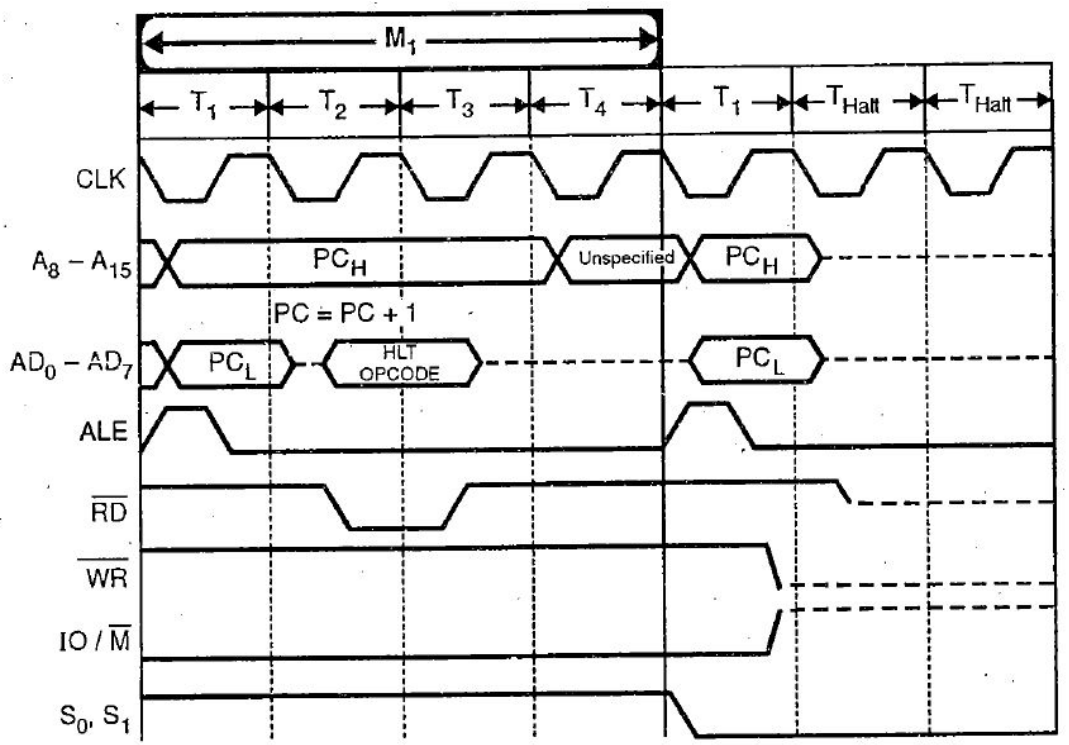
Machine Cycles: 3



7. HLT

Halt.

States: 5
Flags: None
Machine Cycles: 1



8. XTHL

Exchange stack-top with H-L.

[L] ↔ [SP]

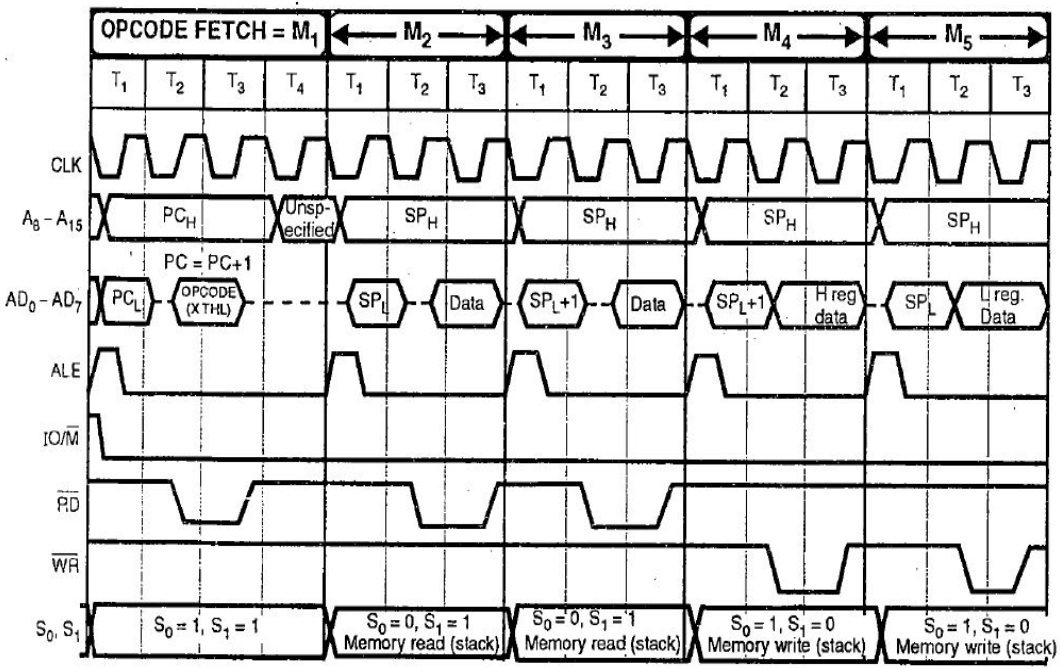
[H] ↔ [SP+1]

States: 16

Flags: None

Addressing: Register Indirect

Machine Cycles: 5

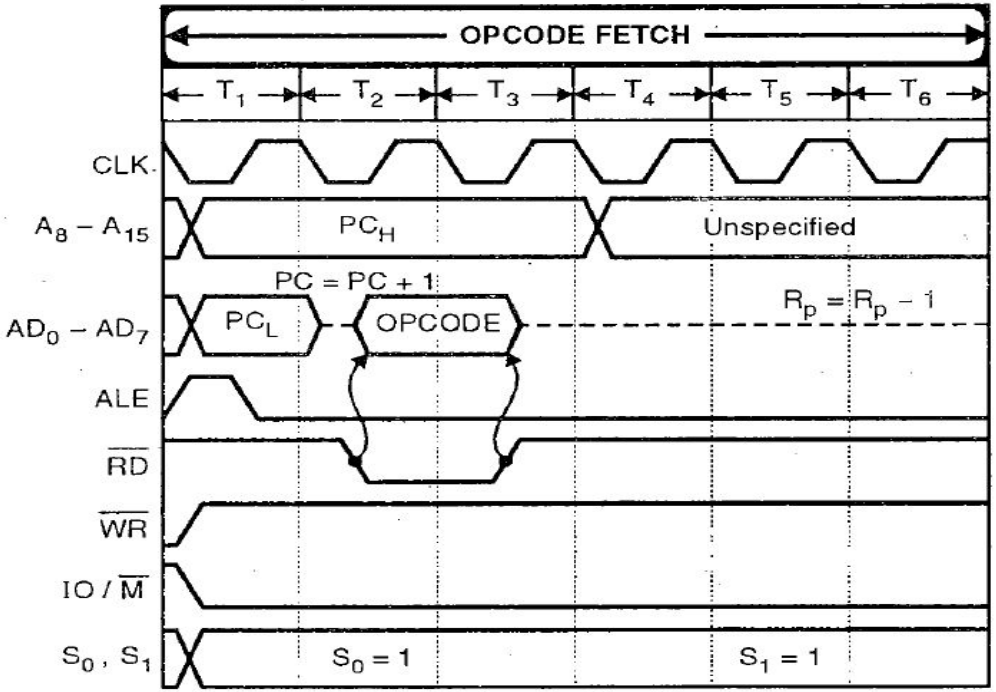


Timing diagram of XTHL instruction

9. SPHL

Move the contents of H-L Pair to stack pointer.
[H-L] → [SP]

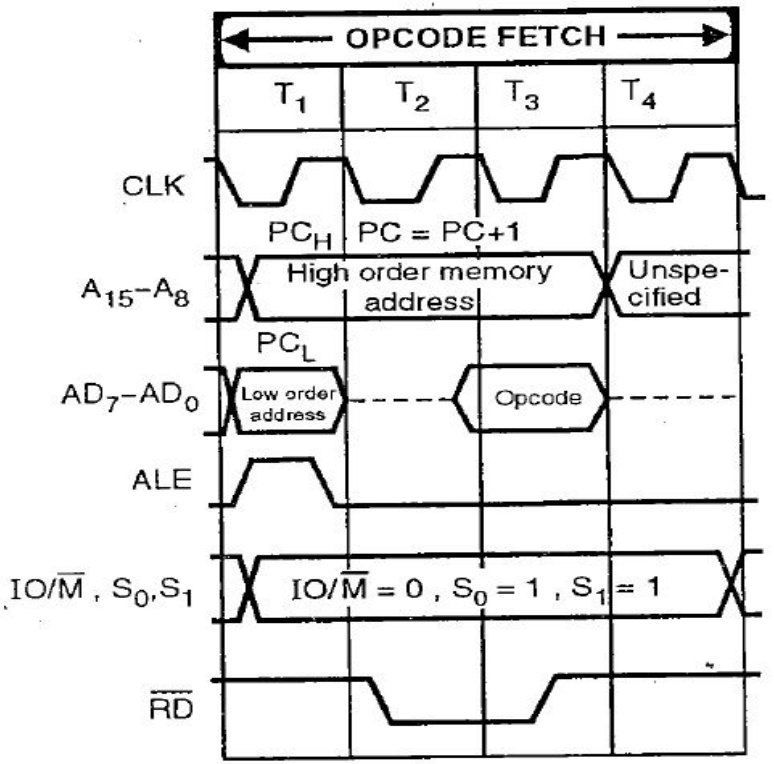
States: 6
Flags: None
Addressing: Register
Machine Cycles: 1



10. EI

Enable Interrupts.

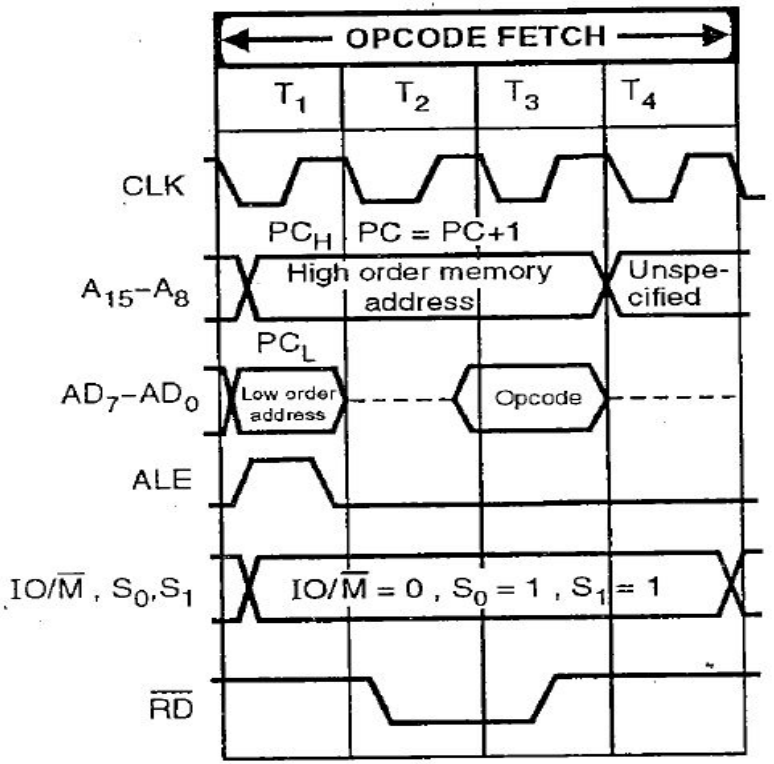
States: 4
Flags: None
Machine Cycles: 1



11. DI

Disable Interrupts.

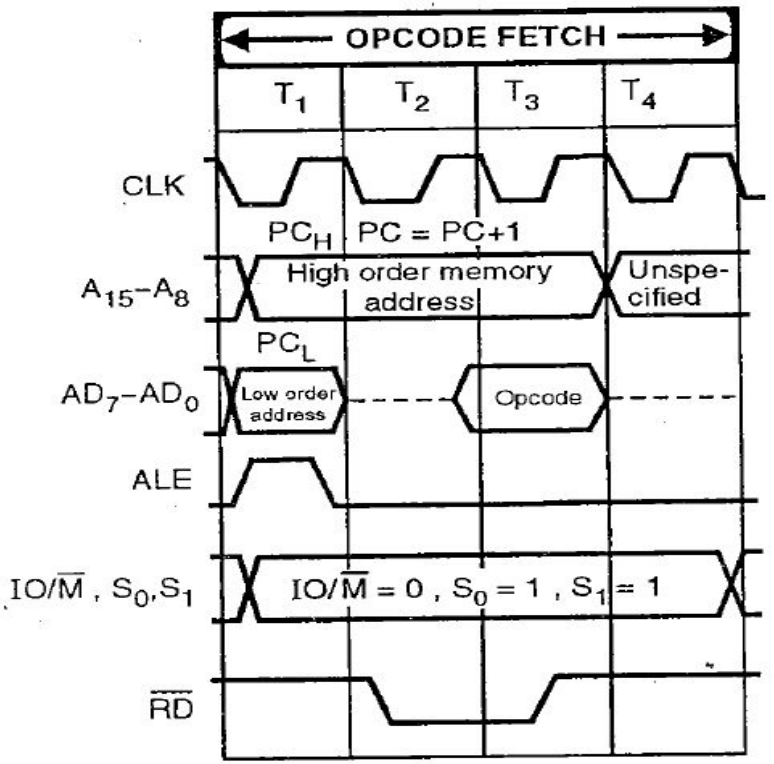
States: 4
Flags: None
Machine Cycles: 1



12. SIM

Set Interrupt Mask.

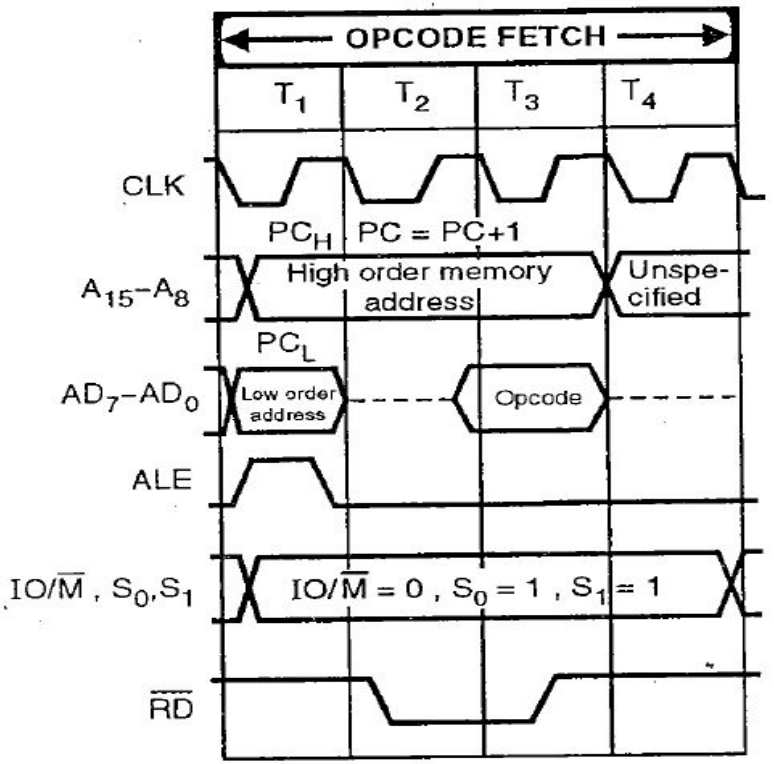
States: 4
Flags: None
Machine Cycles: 1



13. RIM

Read Interrupt Mask.

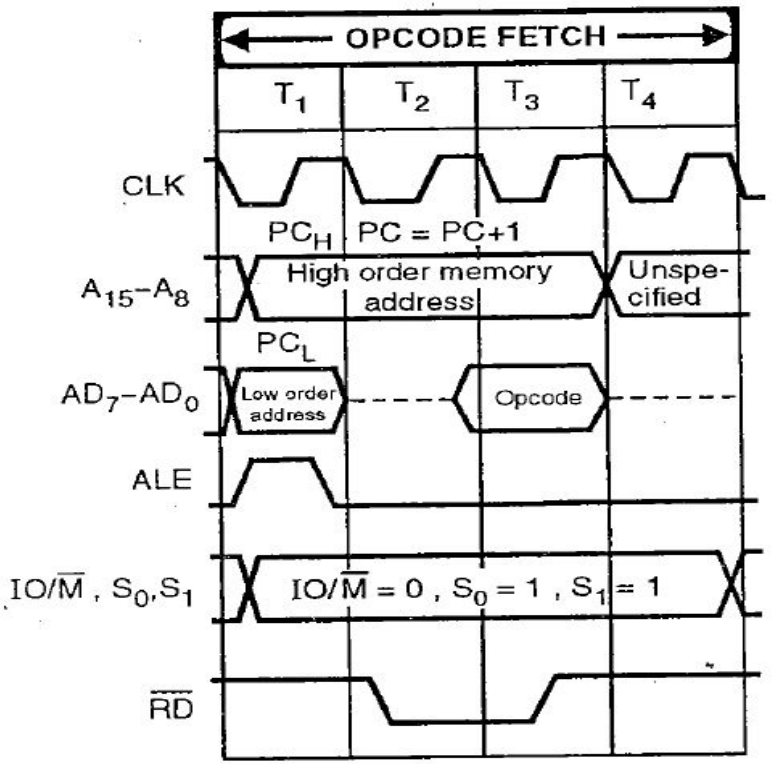
States: 4
Flags: None
Machine Cycles: 1



14. NOP

No Operation.

States: 4
Flags: None
Machine Cycles: 1




Thank you!