8085 Instruction Set

Batch :BEC2A, BEC2B

Year:2021

Outline

- What is instruction set? Machine Cycles? T state?
- How to calculate T state?
- Types of Instructions and elaborate description with examples
- Summary and further study

Instruction Set of 8085

- An instruction is a binary pattern designed inside a microprocessor to perform a specific function.
- The entire group of instructions that a microprocessor supports is called ***Instruction Set***.
- 8085 has **246** instructions.
- Each instruction is represented by an 8-bit binary value.
- These 8-bits of binary value is called ***Op-Code*** or ***Instruction Byte***.

Machine Cycles and T-state

- You can find Name of the instruction, Operation of the instruction, Machine cycles, T-state.
- Machine cycles are sub part of any instruction.
- Machine cycles: It can be
  - Opcode Fetch OF requires (4 T-states/6 T-states)
  - Memory Read MR requires (3 T-state)
  - Memory Write MW requires (3 T-state)
  - Input/Output Read I/OR requires (3 T-state)
  - Input/Output Write I/O W requires (3 T-state)

Contd…

- T state calculation:
  - If Clock Frequency(f) of 8085 processor is 3.2 MHz, then the Clock Time period (T) is :
    - T= 1/f =1/3.2MHz= 0.3125 micro second
    - So T= 0.3125 micro Second

    Now calculate the time required for different operations:

Machine cycle	T state	Time in micro sec.
Opcode fetch	4/6	1.25 / 1.875
Memory Read/Write	3	0.9375
I/O Read/Write	3	0.9375

Classification of Instruction Set

- Data Transfer Instruction
- Arithmetic Instructions
- Logical Instructions
- Branching Instructions
- Control Instructions

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.

Data Transfer Instructions

Opcode	Operand	Description
MOV	Rd, Rs M, Rs Rd, M	Copy from source to destination.

- This instruction copies the contents of the source register into the destination register.
- The contents of the source register are not altered.
- If one of the operands is a memory location, its location is specified by the contents of the HL registers.
- **Example:** MOV B, C or MOV B, M

Data Transfer Instructions

Opcode	Operand	Description
MVI	Rd, Data M, Data	Move immediate 8-bit

- The 8-bit data is stored in the destination register or memory.
- If the operand is a memory location, its location is specified by the contents of the H-L registers.
- **Example:** MVI B, 57H or MVI M, 57H

Data Transfer Instructions

Opcode	Operand	Description
LDA	16-bit address	Load Accumulator

- The contents of a memory location, specified by a 16-bit address in the operand, are copied to the accumulator.
- The contents of the source are not altered.
- **Example:** LDA 2034H

Data Transfer Instructions

Opcode	Operand	Description
LDAX	B/D Register Pair	Load accumulator indirect

- The contents of the designated register pair point to a memory location.
- This instruction copies the contents of that memory location into the accumulator.
- The contents of either the register pair or the memory location are not altered.
- **Example:** LDAX B

Data Transfer Instructions

Opcode	Operand	Description
LXI	Reg. pair, 16-bit data	Load register pair immediate

- This instruction loads 16-bit data in the register pair.
- **Example:** LXI H, 2034 H

Data Transfer Instructions

Opcode	Operand	Description
LHLD	16-bit address	Load H-L registers direct

- This instruction copies the contents of memory location pointed out by 16-bit address into register L.
- It copies the contents of next memory location into register H.
- **Example:** LHLD 2040 H

Data Transfer Instructions

Opcode	Operand	Description
STA	16-bit address	Store accumulator direct

- The contents of accumulator are copied into the memory location specified by the operand.
- **Example:** STA 2500 H

Data Transfer Instructions

Opcode	Operand	Description
STAX	Reg. pair	Store accumulator indirect

- The contents of accumulator are copied into the memory location specified by the contents of the register pair.
- **Example:** STAX B

Data Transfer Instructions

Opcode	Operand	Description
SHLD	16-bit address	Store H-L registers direct

- The contents of register L are stored into memory location specified by the 16-bit address.
- The contents of register H are stored into the next memory location.
- **Example:** SHLD 2550 H

Data Transfer Instructions

Opcode	Operand	Description
XCHG	None	Exchange H-L with D-E

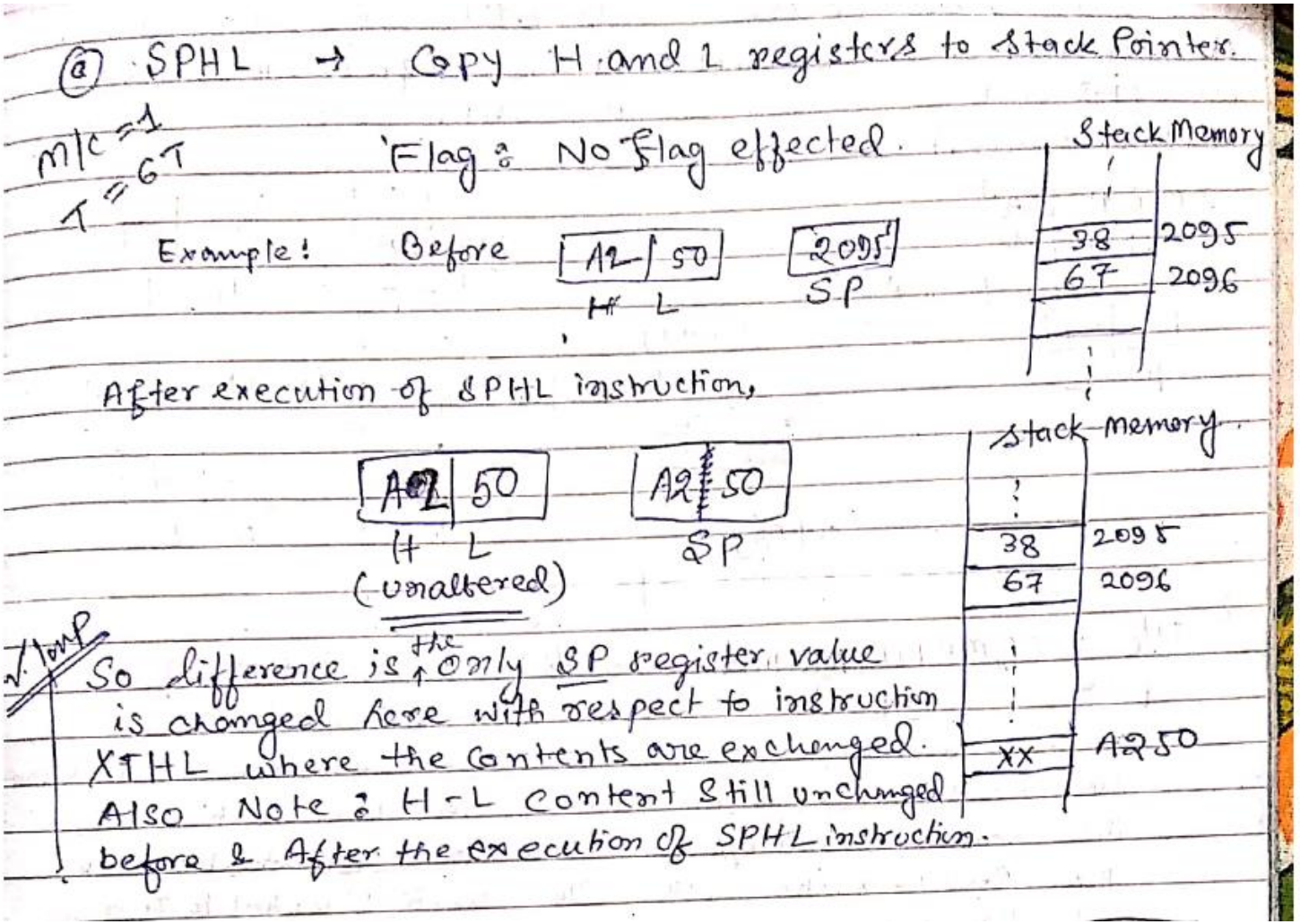
- The contents of register H are exchanged with the contents of register D.
- The contents of register L are exchanged with the contents of register E.
- **Example:** XCHG

Data Transfer Instructions

Opcode	Operand	Description
SPHL	None	Copy H-L pair to the Stack Pointer (SP)

- This instruction loads the contents of H-L pair into SP.
- **Example:** SPHL

Reference slide(SPHL EXPLAINED)

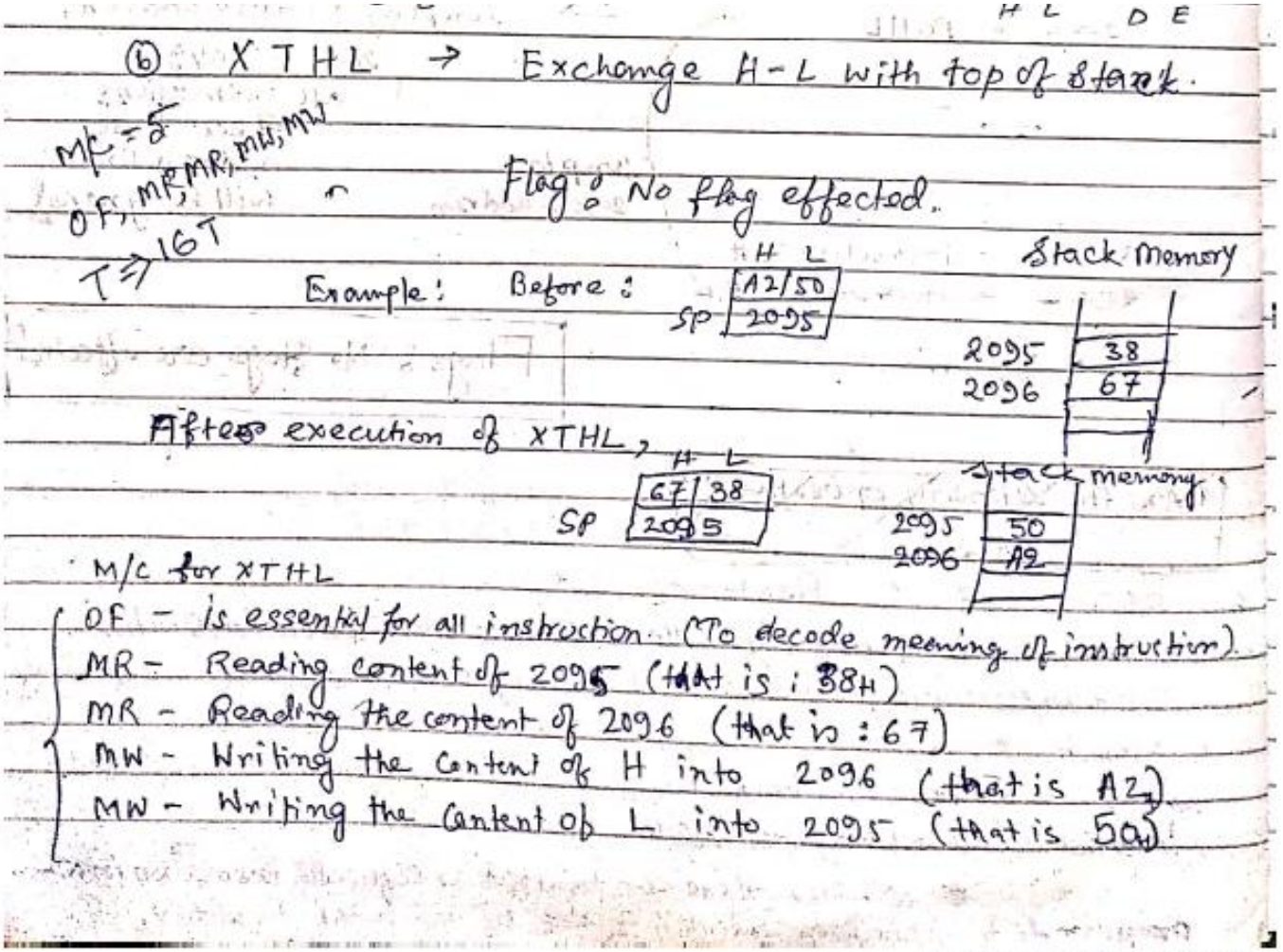


Data Transfer Instructions

Opcode	Operand	Description
XTHL	None	Exchange H–L with top of stack

- The contents of L register are exchanged with the location pointed out by the contents of the SP.
- The contents of H register are exchanged with the next location (SP + 1).
- **Example:** XTHL

Reference slides (XTHL explained)



Scanned with CamScanner

Data Transfer Instructions

Opcode	Operand	Description
PCHL	None	Load program counter with H-L contents

- The contents of registers H and L are copied into the program counter (PC).
- The contents of H are placed as the high-order byte and the contents of L as the low-order byte.
- **Example:** PCHL

Data Transfer Instructions

Opcode	Operand	Description
PUSH	Reg. pair	Push register pair onto stack

- The contents of register pair are copied onto stack.
- SP is decremented and the contents of high-order registers (B, D, H, A) are copied into stack.
- SP is again decremented and the contents of low-order registers (C, E, L, Flags) are copied into stack.
- **Example:** PUSH B

Data Transfer Instructions

Opcode	Operand	Description
POP	Reg. pair	Pop stack to register pair

- The contents of top of stack are copied into register pair.
- The contents of location pointed out by SP are copied to the low-order register (C, E, L, Flags).
- SP is incremented and the contents of location are copied to the high-order register (B, D, H, A).
- **Example:** POP H

Data Transfer Instructions

Opcode	Operand	Description
OUT	8-bit port address	Copy data from accumulator to a port with 8-bit address

- The contents of accumulator are copied into the I/O port.
- **Example:** OUT 78 H

Data Transfer Instructions

Opcode	Operand	Description
IN	8-bit port address	Copy data to accumulator from a port with 8-bit address

- The contents of I/O port are copied into accumulator.
- **Example:** IN 8C H

Addition

- Any 8-bit number, or the contents of register, or the contents of memory location can be added to the contents of accumulator.
- The result (sum) is stored in the accumulator.
- No two other 8-bit registers can be added directly.
- **Example:** The contents of register B cannot be added directly to the contents of register C.

Subtraction

- Any 8-bit number, or the contents of register, or the contents of memory location can be subtracted from the contents of accumulator.
- The result is stored in the accumulator.
- Subtraction is performed in 2's complement form.
- If the result is negative, it is stored in 2's complement form.
- No two other 8-bit registers can be subtracted directly.

Increment / Decrement

- The 8-bit contents of a register or a memory location can be incremented or decremented by 1.
- The 16-bit contents of a register pair can be incremented or decremented by 1.
- Increment or decrement can be performed on any register or a memory location.

Arithmetic Instructions

Opcode	Operand	Description
ADD	R M	Add register or memory to accumulator

- The contents of register or memory are added to the contents of accumulator.
- The result is stored in accumulator.
- If the operand is memory location, its address is specified by H-L pair.
- All flags are modified to reflect the result of the addition.
- **Example:** ADD B or ADD M

Arithmetic Instructions

Opcode	Operand	Description
ADC	R M	Add register or memory to accumulator with carry

- The contents of register or memory and Carry Flag (CY) are added to the contents of accumulator.
- The result is stored in accumulator.
- If the operand is memory location, its address is specified by H-L pair.
- All flags are modified to reflect the result of the addition.
- **Example:** ADC B or ADC M

Arithmetic Instructions

Opcode	Operand	Description
ADI	8-bit data	Add immediate to accumulator

- The 8-bit data is added to the contents of accumulator.
- The result is stored in accumulator.
- All flags are modified to reflect the result of the addition.
- **Example:** ADI 45 H

Arithmetic Instructions

Opcode	Operand	Description
ACI	8-bit data	Add immediate to accumulator with carry

- The 8-bit data and the Carry Flag (CY) are added to the contents of accumulator.
- The result is stored in accumulator.
- All flags are modified to reflect the result of the addition.
- **Example:** ACI 45 H

Arithmetic Instructions

Opcode	Operand	Description
DAD	Reg. pair	Add register pair to H-L pair

- The 16-bit contents of the register pair are added to the contents of H-L pair.
- The result is stored in H-L pair.
- If the result is larger than 16 bits, then CY is set.
- No other flags are changed.
- **Example:** DAD B

Arithmetic Instructions

Opcode	Operand	Description
SUB	R M	Subtract register or memory from accumulator

- The contents of the register or memory location are subtracted from the contents of the accumulator.
- The result is stored in accumulator.
- If the operand is memory location, its address is specified by H-L pair.
- All flags are modified to reflect the result of subtraction.
- **Example:** SUB B or SUB M

Arithmetic Instructions

Opcode	Operand	Description
SBB	R M	Subtract register or memory from accumulator with borrow

- The contents of the register or memory location and Borrow Flag (i.e. CY) are subtracted from the contents of the accumulator.
- The result is stored in accumulator.
- If the operand is memory location, its address is specified by H-L pair.
- All flags are modified to reflect the result of subtraction.
- **Example:** SBB B or SBB M

Arithmetic Instructions

Opcode	Operand	Description
SUI	8-bit data	Subtract immediate from accumulator

- The 8-bit data is subtracted from the contents of the accumulator.
- The result is stored in accumulator.
- All flags are modified to reflect the result of subtraction.
- **Example:** SUI 45 H

Arithmetic Instructions

Opcode	Operand	Description
SBI	8-bit data	Subtract immediate from accumulator with borrow

- The 8-bit data and the Borrow Flag (i.e. CY) is subtracted from the contents of the accumulator.
- The result is stored in accumulator.
- All flags are modified to reflect the result of subtraction.
- **Example:** SBI 45 H

Arithmetic Instructions

Opcode	Operand	Description
INR	R M	Increment register or memory by 1

- The contents of register or memory location are incremented by 1.
- The result is stored in the same place.
- If the operand is a memory location, its address is specified by the contents of H-L pair.
- **Example:** INR B or INR M

Arithmetic Instructions

Opcode	Operand	Description
INX	R	Increment register pair by 1

- The contents of register pair are incremented by 1.
- The result is stored in the same place.
- **Example:** INX H

Arithmetic Instructions

Opcode	Operand	Description
DCR	R M	Decrement register or memory by 1

- The contents of register or memory location are decremented by 1.
- The result is stored in the same place.
- If the operand is a memory location, its address is specified by the contents of H-L pair.
- **Example:** DCR B or DCR M

Arithmetic Instructions

Opcode	Operand	Description
DCX	R	Decrement register pair by 1

- The contents of register pair are decremented by 1.
- The result is stored in the same place.
- **Example:** DCX H

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

- PSW (Program Status word)
- - Flag unaffected
- * affected
- 0 reset
- 1 set
- S Sign (Bit 7)
- Z Zero (Bit 6)
- AC Auxiliary Carry (Bit 4)
- P Parity (Bit 2)
- CY Carry (Bit 0)

AND, OR, XOR

- Any 8-bit data, or the contents of register, or memory location can logically have
  - AND operation
  - OR operation
  - XOR operation

  with the contents of accumulator.
- The result is stored in accumulator.

Rotate

- Each bit in the accumulator can be shifted either left or right to the next position.

Compare

- Any 8-bit data, or the contents of register, or memory location can be compares for:
  - Equality
  - Greater Than
  - Less Than

  with the contents of accumulator.
- The result is reflected in status flags.

Complement

- The contents of accumulator can be complemented.
- Each 0 is replaced by 1 and each 1 is replaced by 0.

Logical Instructions

Opcode	Operand	Description
CMP	R M	Compare register or memory with accumulator

- The contents of the operand (register or memory) are compared with the contents of the accumulator.
- Both contents are preserved .
- The result of the comparison is shown by setting the flags of the PSW as follows:

Logical Instructions

Opcode	Operand	Description
CMP	R M	Compare register or memory with accumulator

- if (A) < (reg/mem): carry flag is set
- if (A) = (reg/mem): zero flag is set
- if (A) > (reg/mem): carry and zero flags are reset.
- **Example:** CMP B or CMP M

Logical Instructions

Opcode	Operand	Description
CPI	8-bit data	Compare immediate with accumulator

- The 8-bit data is compared with the contents of accumulator.
- The values being compared remain unchanged.
- The result of the comparison is shown by setting the flags of the PSW as follows:

Logical Instructions

Opcode	Operand	Description
CPI	8-bit data	Compare immediate with accumulator

- if (A) < data: carry flag is set
- if (A) = data: zero flag is set
- if (A) > data: carry and zero flags are reset
- **Example:** CPI 89H

Logical Instructions

Opcode	Operand	Description
ANA	R M	Logical AND register or memory with accumulator

- The contents of the accumulator are logically ANDed with the contents of register or memory.
- The result is placed in the accumulator.
- If the operand is a memory location, its address is specified by the contents of H-L pair.
- S, Z, P are modified to reflect the result of the operation.
- CY is reset and AC is set.
- **Example:** ANA B or ANA M.

Logical Instructions

Opcode	Operand	Description
ANI	8-bit data	Logical AND immediate with accumulator

- The contents of the accumulator are logically ANDed with the 8-bit data.
- The result is placed in the accumulator.
- S, Z, P are modified to reflect the result.
- CY is reset, AC is set.
- **Example:** ANI 86H.

Logical Instructions

Opcode	Operand	Description
ORA	R M	Logical OR register or memory with accumulator

- The contents of the accumulator are logically ORed with the contents of the register or memory.
- The result is placed in the accumulator.
- If the operand is a memory location, its address is specified by the contents of H-L pair.
- S, Z, P are modified to reflect the result.
- CY and AC are reset.
- **Example:** ORA B or ORA M.

Logical Instructions

Opcode	Operand	Description
ORI	8-bit data	Logical OR immediate with accumulator

- The contents of the accumulator are logically ORed with the 8-bit data.
- The result is placed in the accumulator.
- S, Z, P are modified to reflect the result.
- CY and AC are reset.
- **Example:** ORI 86H.

Logical Instructions

Opcode	Operand	Description
XRA	R M	Logical XOR register or memory with accumulator

- The contents of the accumulator are XORed with the contents of the register or memory.
- The result is placed in the accumulator.
- If the operand is a memory location, its address is specified by the contents of H-L pair.
- S, Z, P are modified to reflect the result of the operation.
- CY and AC are reset.
- **Example:** XRA B or XRA M.

Logical Instructions

Opcode	Operand	Description
XRI	8-bit data	XOR immediate with accumulator

- The contents of the accumulator are XORed with the 8-bit data.
- The result is placed in the accumulator.
- S, Z, P are modified to reflect the result.
- CY and AC are reset.
- **Example:** XRI 86H.

Opcode	Operand	Description
RAL	None	Rotate accumulator left through carry

- Each binary bit of the accumulator is rotated left by one position through the Carry flag.
- Bit D7 is placed in the Carry flag, and the Carry flag is placed in the least significant position Do.
- CY is modified according to bit D7.
- S, Z, P, AC are not affected.
- **Example:** RAL.

Opcode	Operand	Description
RAR	None	Rotate accumulator right through carry

- Each binary bit of the accumulator is rotated right by one position through the Carry flag.
- Bit Do is placed in the Carry flag, and the Carry flag is placed in the most significant position D7.
- CY is modified according to bit Do.
- S, Z, P, AC are not affected.
- **Example:** RAR.

circular Left shift

Opcode	Operand	Description
RLC	None	Rotate accumulator left

- Each binary bit of the accumulator is rotated left by one position.
- Bit D7 is placed in the position of Do as well as in the Carry flag.
- CY is modified according to bit D7.
- S, Z, P, AC are not affected.
- **Example:** RLC.

- circular right shift

Opcode	Operand	Description
RRC	None	Rotate accumulator right

- Each binary bit of the accumulator is rotated right by one position.
- Bit Do is placed in the position of D7 as well as in the Carry flag.
- CY is modified according to bit Do.
- S, Z, P, AC are not affected.
- **Example:** RRC.

Logical Instructions

Opcode	Operand	Description
CMA	None	Complement accumulator

- The contents of the accumulator are complemented.
- No flags are affected.
- **Example:** CMA.

Logical Instructions

Opcode	Operand	Description
CMC	None	Complement carry

- The Carry flag is complemented.
- No other flags are affected.
- **Example:** CMC.

Logical Instructions

Opcode	Operand	Description
STC	None	Set carry

- The Carry flag is set to 1.
- No other flags are affected.
- **Example:** STC.

Branching Instructions

- The branching instruction alter the normal sequential flow.
- These instructions alter either unconditionally or conditionally.

Branching Instructions

Opcode	Operand	Description
JMP	16-bit address	Jump unconditionally

- The program sequence is transferred to the memory location specified by the 16-bit address given in the operand.
- **Example:** JMP 2034 H.

Branching Instructions

Opcode	Operand	Description
Jx	16-bit address	Jump conditionally

- The program sequence is transferred to the memory location specified by the 16-bit address given in the operand based on the specified flag of the PSW.
- **Example:** JZ 2034 H.

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

Branching Instructions

Opcode	Operand	Description
CALL	16-bit address	Call unconditionally

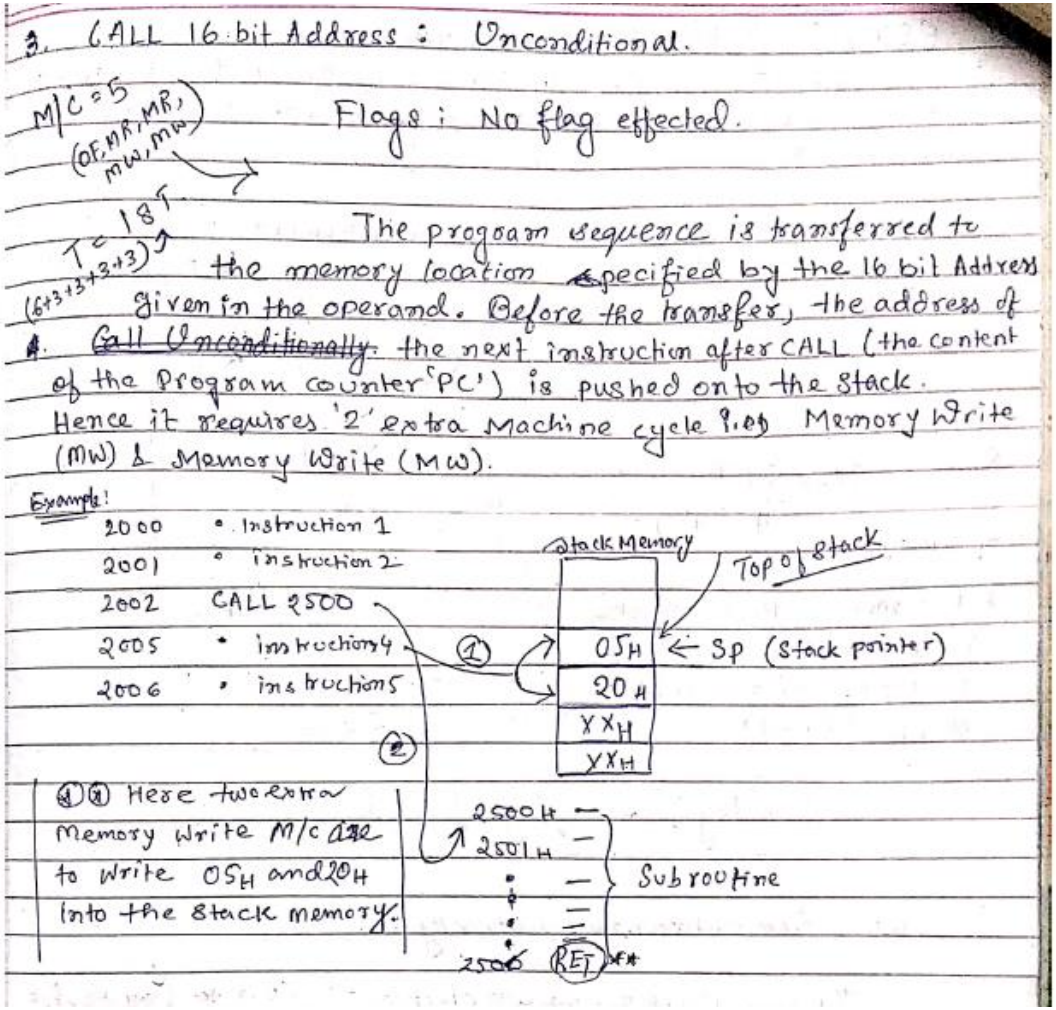
- The program sequence is transferred to the memory location specified by the 16-bit address given in the operand.
- Before the transfer, the address of the next instruction after CALL (the contents of the program counter) is pushed onto the stack.
- **Example:** CALL 2034 H.

Branching Instructions

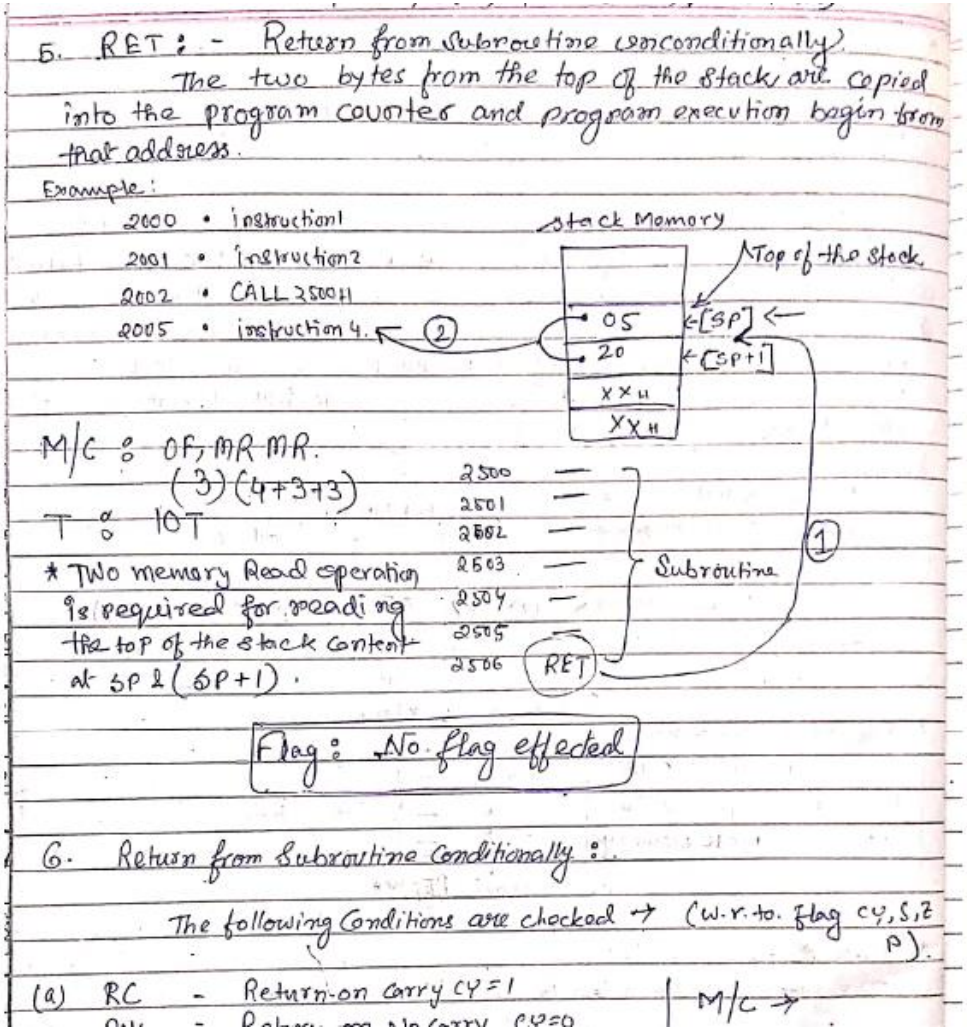
Opcode	Operand	Description
RET	None	Return unconditionally

- The program sequence is transferred from the subroutine to the calling program.
- The two bytes from the top of the stack are copied into the program counter, and program execution begins at the new address.
- **Example:** RET.

Reference slide(CALL ins. diagram)



Reference slide(RET ins. diagram)



Control Instructions

Opcode	Operand	Description
NOP	None	No operation

- No operation is performed.
- The instruction is fetched and decoded but no operation is executed.
- **Example:** NOP

Control Instructions

Opcode	Operand	Description
HLT	None	Halt

- The CPU finishes executing the current instruction and halts any further execution.
- An interrupt or reset is necessary to exit from the halt state.
- **Example:** HLT

Control Instructions

Opcode	Operand	Description
DI	None	Disable interrupt

- The interrupt enable flip-flop is reset and all the interrupts except the TRAP are disabled.
- No flags are affected.
- **Example:** DI

Control Instructions

Opcode	Operand	Description
EI	None	Enable interrupt

- The interrupt enable flip-flop is set and all interrupts are enabled.
- No flags are affected.
- This instruction is necessary to re-enable the interrupts (except TRAP).
- **Example:** EI

Summary – Data transfer

- MOV Move
- MVI Move Immediate
- LDA Load Accumulator Directly from Memory
- STA Store Accumulator Directly in Memory
- LHLD Load H & L Registers Directly from Memory
- SHLD Store H & L Registers Directly in Memory

Summary Data transfer

- An 'X' in the name of a data transfer instruction implies that it deals with a register pair (16-bits);

- LXI Load Register Pair with Immediate data
- LDAX Load Accumulator from Address in Register Pair
- STAX Store Accumulator in Address in Register Pair
- XCHG Exchange H & L with D & E
- XTHL Exchange Top of Stack with H & L

Summary - Arithmetic Group

- Add, Subtract, Increment / Decrement data in registers or memory.

- ADD Add to Accumulator
- ADI Add Immediate Data to Accumulator
- ADC Add to Accumulator Using Carry Flag
- ACI Add Immediate data to Accumulator Using Carry
- SUB Subtract from Accumulator
- SUI Subtract Immediate Data from Accumulator
- SBB Subtract from Accumulator Using Borrow (Carry) Flag
- SBI Subtract Immediate from Accumulator Using Borrow (Carry) Flag
- INR Increment Specified Byte by One
- DCR Decrement Specified Byte by One
- INX Increment Register Pair by One
- DCX Decrement Register Pair by One
- DAD Double Register Add; Add Content of Register Pair to H & L Register Pair

Summary Logical Group

- This group performs logical (Boolean) operations on data in registers and memory and on condition flags.
- These instructions enable you to set specific bits in the accumulator ON or OFF.

- ANA Logical AND with Accumulator
- ANI Logical AND with Accumulator Using Immediate Data
- ORA Logical OR with Accumulator
- OR Logical OR with Accumulator Using Immediate Data
- XRA Exclusive Logical OR with Accumulator
- XRI Exclusive OR Using Immediate Data

- The Compare instructions compare the content of an 8-bit value with the contents of the accumulator;

- CMP Compare
- CPI Compare Using Immediate Data

- The rotate instructions shift the contents of the accumulator one bit position to the left or right:

- RLC Rotate Accumulator Left
- RRC Rotate Accumulator Right
- RAL Rotate Left Through Carry
- RAR Rotate Right Through Carry

- Complement and carry flag instructions:

- CMA Complement Accumulator
- CMC Complement Carry Flag
- STC Set Carry Flag

Summary - Branch Group

- Unconditional branching
  - JMP Jump
  - CALL Call
  - RET Return
- Conditions
  - NZ Not Zero (Z = 0)
  - Z Zero (Z = 1)
  - NC No Carry (C = 0)
  - C Carry (C = 1)
  - PO Parity Odd (P = 0)
  - PE Parity Even (P = 1)
  - P Plus (S = 0)
  - M Minus (S = 1)
- Conditional branching

Summary - Stack

- PUSH Push Two bytes of Data onto the Stack
- POP Pop Two Bytes of Data off the Stack
- XTHL Exchange Top of Stack with H & L
- SPHL Move content of H & L to Stack Pointer

I/0 instructions

- IN Initiate Input Operation
- OUT Initiate Output Operation

Summary -Machine Control instructions

- EI Enable Interrupt System
- DI Disable Interrupt System
- HLT Halt
- NOP No Operation

Reference slides

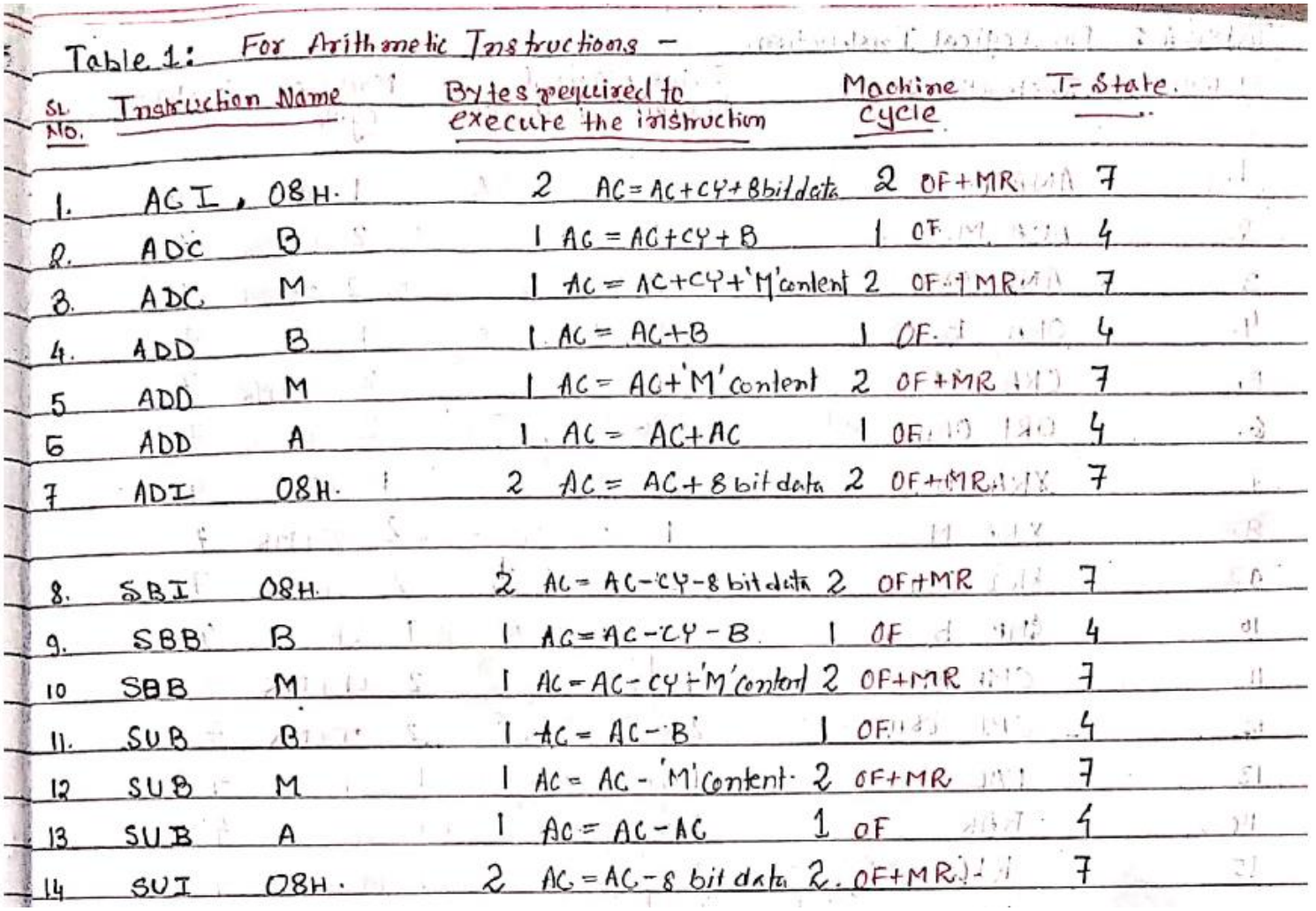
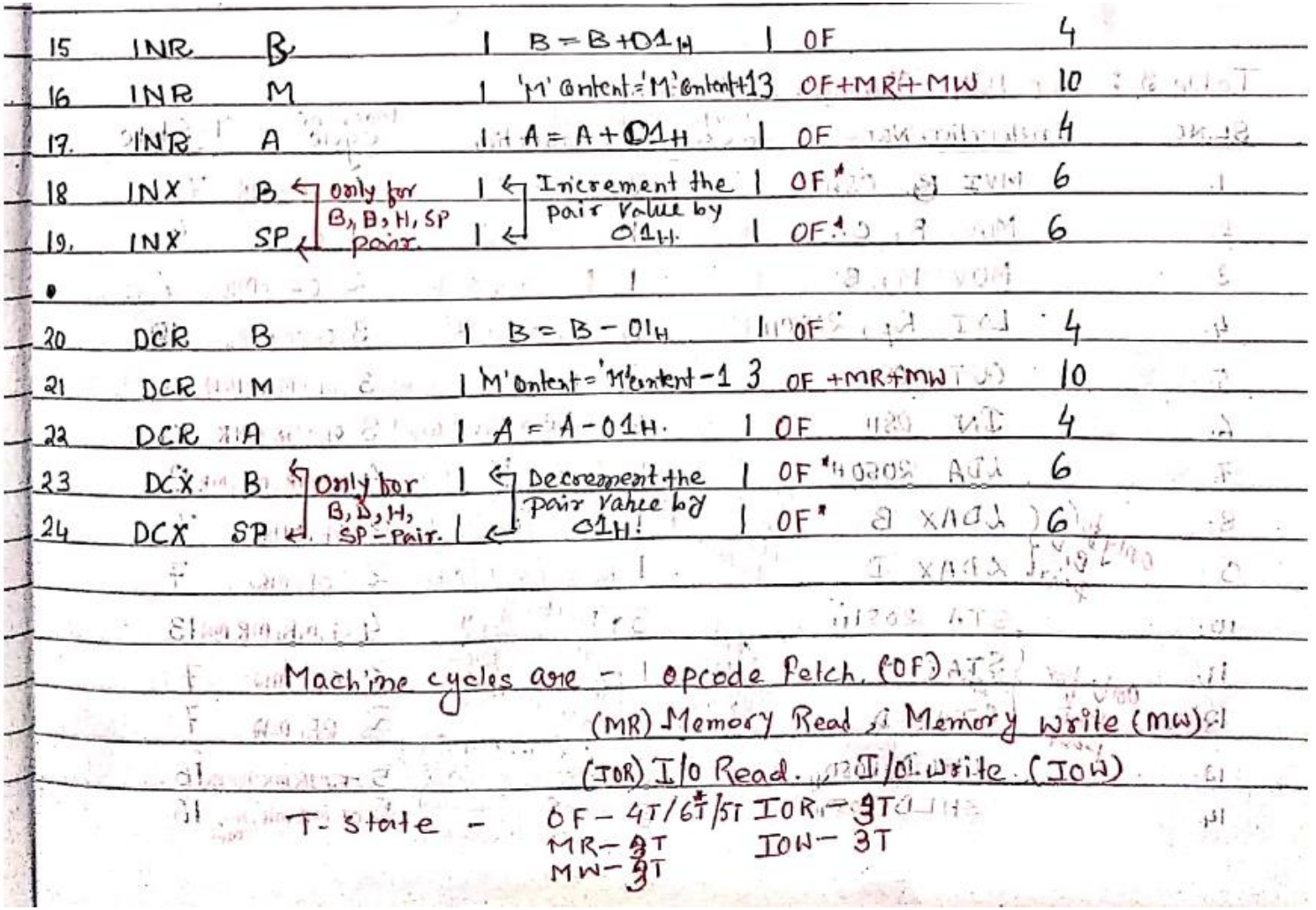


Table 1: For Arithmetic Instructions –

Sl. No.	Instruction	Name	Bytes required to execute the instruction		Machine cycle		T-State
1.	ACI	08H.	2	AC = AC + CY + 8 bit data	2	OF + MR	7
2.	ADC	B	1	AC = AC + CY + B	1	OF	4
3.	ADC	M	1	AC = AC + CY + 'M' content	2	OF + MR	7
4.	ADD	B	1	AC = AC + B	1	OF	4
5	ADD	M	1	AC = AC + 'M' content	2	OF + MR	7
6	ADD	A	1	AC = AC + AC	1	OF	4
7	ADI	08H.	2	AC = AC + 8 bit data	2	OF + MR	7
8.	SBI	08H.	2	AC = AC - CY - 8 bit data	2	OF + MR	7
9.	SBB	B	1	AC = AC - CY - B	1	OF	4
10	SBB	M	1	AC = AC - CY + 'M' content	2	OF + MR	7
11.	SUB	B	1	AC = AC - B	1	OF	4
12	SUB	M	1	AC = AC - 'M' content	2	OF + MR	7
13	SUB	A	1	AC = AC - AC	1	OF	4
14	SUI	08H.	2	AC = AC - 8 bit data	2	OF + MR	7

Reference slides



15	INR	B	B = B + 01H	OF	4
16	INR	M	'M' Content = 'M' Content + 1	3 OF + MR + MW	10
17.	INR	A	A = A + 01H	OF	4
18	INX	B ← Only for B, D, H, SP pair.	← Increment the pair value by 01H.	OF	6
19.	INX	SP ←		OF	6
20	DCR	B	B = B − 01H	OF	4
21	DCR	M	'M' Content = 'M' Content − 1	3 OF + MR + MW	10
22	DCR	A	A = A − 01H.	OF	4
23	DCX	B ← Only for B, D, H, SP-Pair.	← Decrement the pair value by 01H!	OF	6
24	DCX	SP ←		OF	6

Machine cycles are - Opcode Fetch (OF)
(MR) Memory Read, Memory Write (MW)
(IOR) I/O Read, I/O Write (IOW)

T-State - OF - 4T/6T/5T IOR - 3T
MR - 3T IOW - 3T
MW - 3T

Reference slides

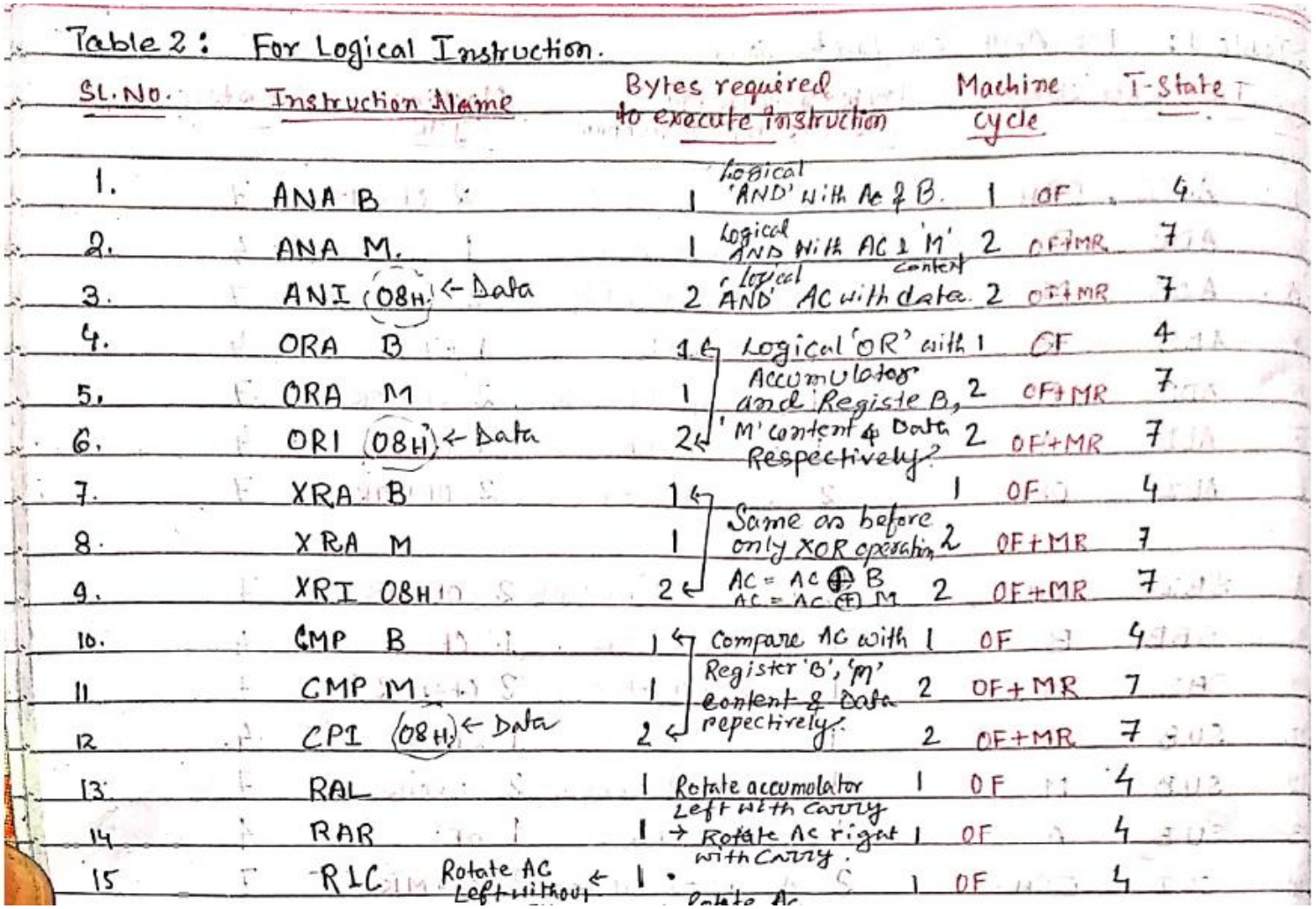
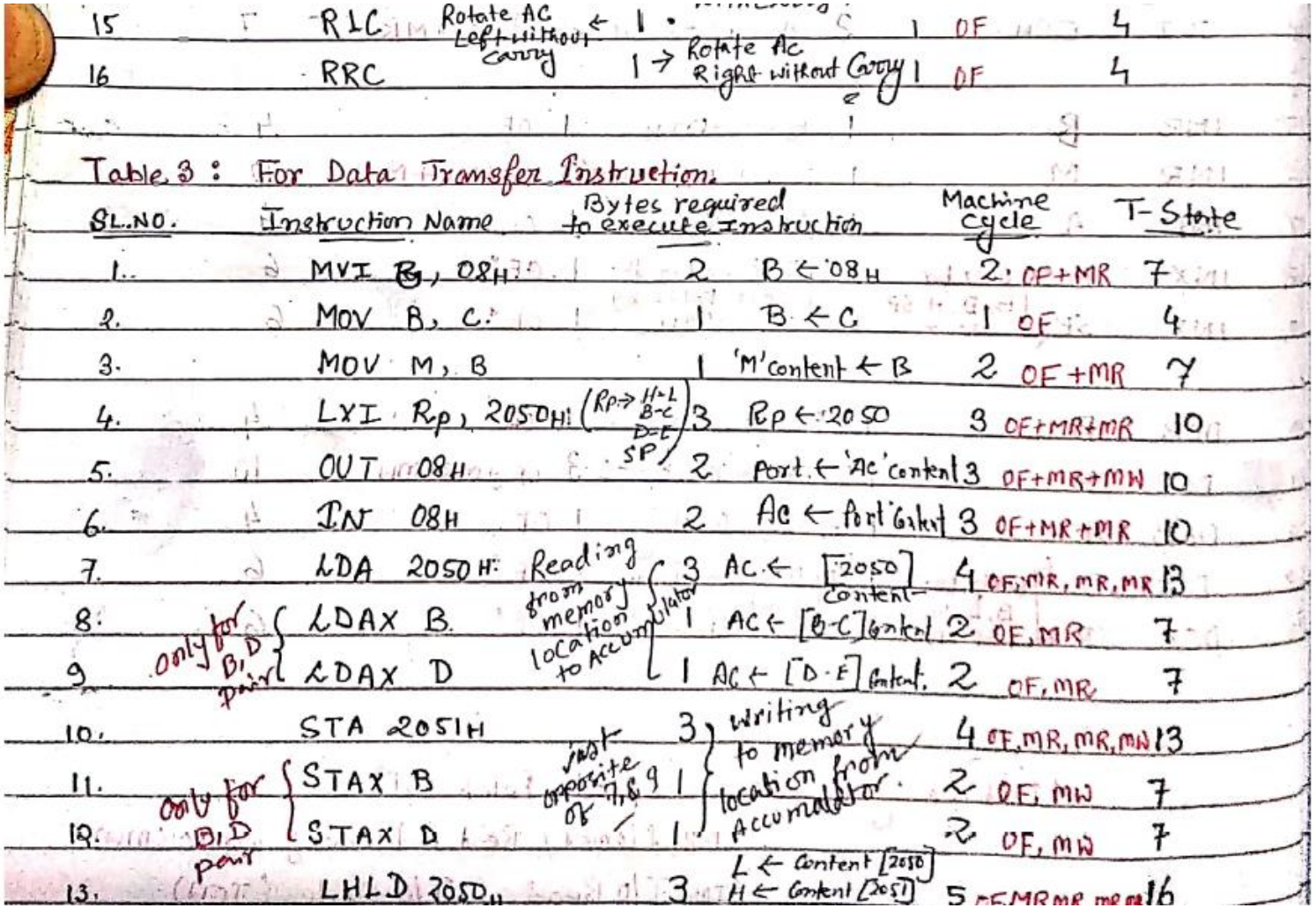


Table 2: For Logical Instruction.

Sl. No.	Instruction Name	Bytes required to execute instruction		Machine cycle		T-State
1.	ANA B	1	Logical 'AND' with Ac & B.	1	OF	4
2.	ANA M.	1	Logical AND with AC & 'M' content	2	OF+MR	7
3.	ANI (08H) ← Data	2	'Logical AND' AC with data	2	OF+MR	7
4.	ORA B	1	Logical 'OR' with Accumulator and Registe B, 'M' content & Data Respectively?	1	OF	4
5.	ORA M	1		2	OF+MR	7
6.	ORI (08H) ← Data	2		2	OF+MR	7
7.	XRA B	1	Same as before only XOR operation AC = AC ⊕ B AC = AC ⊕ M	1	OF	4
8.	XRA M	1		2	OF+MR	7
9.	XRI 08H	2		2	OF+MR	7
10.	CMP B	1	Compare AC with Register 'B', 'M' content & Data repectively.	1	OF	4
11	CMP M	1		2	OF+MR	7
12	CPI (08H) ← Data	2		2	OF+MR	7
13	RAL	1	Rotate accumolator Left with carry	1	OF	4
14	RAR	1	→ Rotate Ac right with Carry.	1	OF	4
15	RLC Rotate AC Left without ←	1	• Rotate Ac	1	OF	4

Reference slides



15	RLC	Rotate AC Left without Carry	1	OF	4
16	RRC	Rotate AC Right without Carry	1	OF	4

Table 3 : For Data Transfer Instruction.

Sl.No.	Instruction Name	Bytes required to execute Instruction		Machine cycle	T-State
1.	MVI B, 08H	2	B ← 08H	2 OF+MR	7
2.	MOV B, C	1	B ← C	1 OF	4
3.	MOV M, B	1	'M' content ← B	2 OF+MR	7
4.	LXI Rp, 2050H (Rp → H-L, B-C, D-E, SP)	3	Rp ← 2050	3 OF+MR+MR	10
5.	OUT 08H	2	Port ← 'Ac' content	3 OF+MR+MW	10
6.	IN 08H	2	Ac ← Port 'G' content	3 OF+MR+MR	10
7.	LDA 2050H: Reading from memory location to Accumulator	3	Ac ← [2050] content	4 OF, MR, MR, MR	13
8.	LDAX B (only for B, D pair)	1	Ac ← [B-C] content	2 OF, MR	7
9.	LDAX D (only for B, D pair)	1	Ac ← [D-E] content	2 OF, MR	7
10.	STA 2051H	3	writing to memory location from Accumulator (just opposite of 7, 8 & 9)	4 OF, MR, MR, MW	13
11.	STAX B (only for B, D pair)	1		2 OF, MW	7
12.	STAX D (only for B, D pair)	1		2 OF, MW	7
13.	LHLD 2050H	3	L ← Content [2050]; H ← Content [2051]	5 OF MR MR MR	16

Reference slides

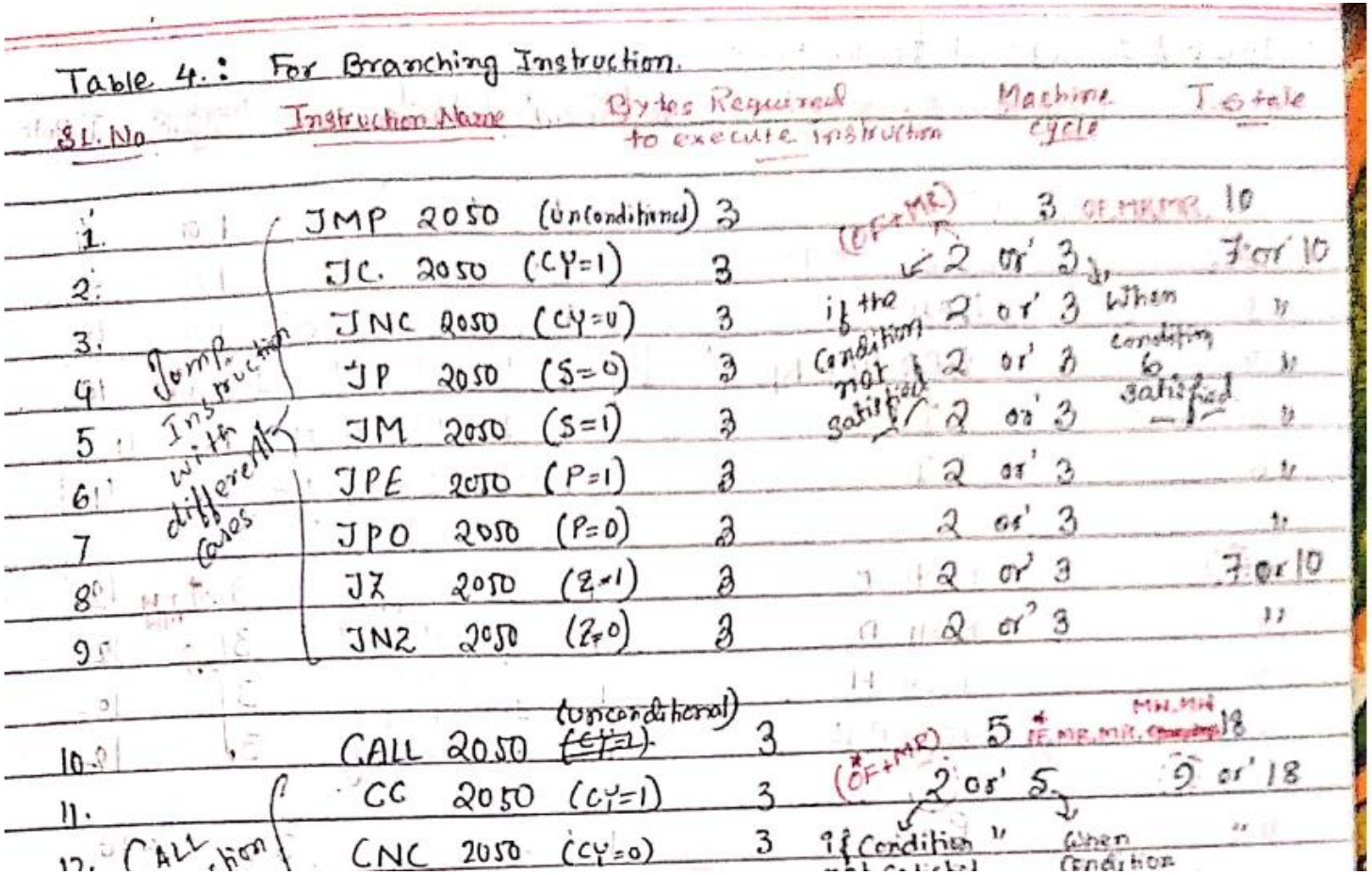
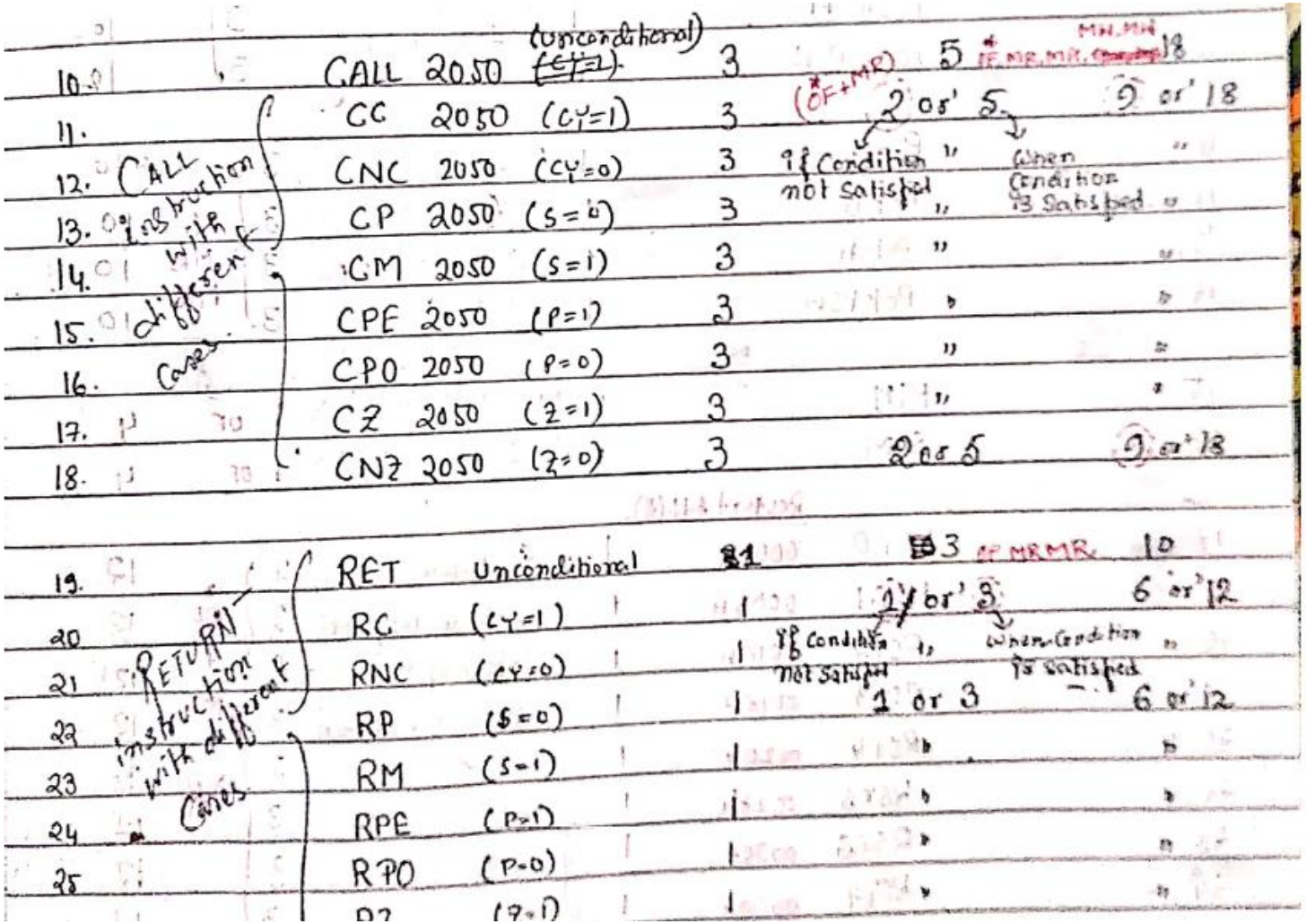


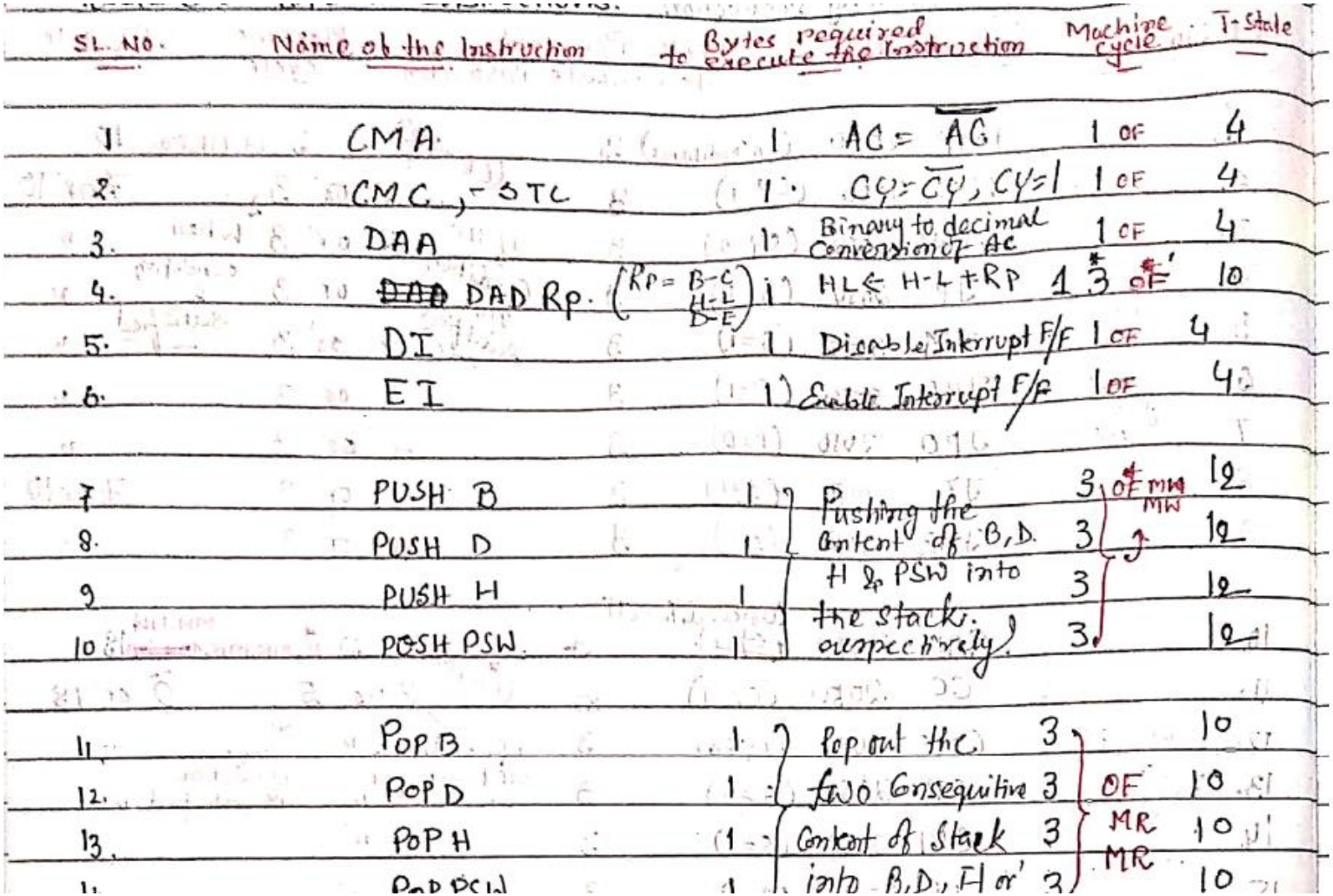
Table 4.: For Branching Instruction.

Sl. No		Instruction Name	Bytes Required to execute instruction	Machine cycle	Totale
1.	Jump Instruction with different codes	JMP 2050 (Unconditional)	3	3 (OF+MR)	10
2.		JC. 2050 (CY=1)	3	2 or 3 if the condition not satisfied / When condition satisfied	7 or 10
3.		JNC 2050 (CY=0)	3	2 or 3	"
4.		JP 2050 (S=0)	3	2 or 3	"
5.		JM 2050 (S=1)	3	2 or 3	"
6.		JPE 2050 (P=1)	3	2 or 3	"
7.		JPO 2050 (P=0)	3	2 or 3	"
8.		JZ 2050 (Z=1)	3	2 or 3	7 or 10
9.		JNZ 2050 (Z=0)	3	2 or 3	"
10.		CALL 2050 (Unconditional)	3	5 (OF+MR)	18
11.		CC 2050 (CY=1)	3	2 or 5 if Condition / When condition	9 or 18
12.	CALL	CNC 2050 (CY=0)	3	"	"

Reference slides



Reference slides



Sl. No.	Name of the Instruction	Bytes required to execute the instruction		Machine cycle	T-State
1	CMA	1	AC = $\overline{AC}$	1 OF	4
2.	CMC, STC	1	CY = $\overline{CY}$, CY = 1	1 OF	4
3.	DAA	1	Binary to decimal conversion of Ac	1 OF	4
4.	DAD Rp. (Rp = B-C, H-L, D-E)	1	HL ← H-L + RP	1 3 OF	10
5.	DI	1	Disable Interrupt F/F	1 OF	4
6.	EI	1	Enable Interrupt F/F	1 OF	4
7	PUSH B	1	Pushing the Content of B,D. H & PSW into the stack. respectively	3 OF MW MW	12
8.	PUSH D	1		3	12
9	PUSH H	1		3	12
10	PUSH PSW	1		3	12
11	POP B	1	Pop out the two consequitive Content of Stack into B,D, H or	3	10
12.	POP D	1		3 OF MR	10
13.	POP H	1		3 MR	10
14.	POP PSW	1		3	10