classmate
Date
Page

★ Number System

① Two's Complement

→ When doing operation on 2 - n bit numbers the carry flag is set if $(n+1)^{th}$ bit is set

→ **Overflow Rule:**

When adding two numbers of same sign gives output of opposite sign overflow bit is set.

Illy for Subtraction

★ No. of representation of 0 is 1

Range : -2^{n-1} to $2^{n-1}-1$

★ A - B : can be calculated using

$$A + \underbrace{(2^n - B)}_{2^{is} \text{ complement of } B}$$

★ Two's complement of x is $(2^n - x)$ Y. 2^n

∴ $-0 = 0$

$-128 = -128$ } Special cases

classmate
Date
Page

* FP numbers with base 2

$$(-1)^S \times F \times 2^E$$

S: Sign

F: Fraction (Called Mantissa or Significand)

E: Exponent

* Float (Single Precision)

1 bit 8 bits 23 bits

* Double (Double Precision)

1 bit 11 bits 52 (20+32) bits

* Only one non-zero digit to left of decimal point no need to store it (# Base 2)

(Such numbers are normalized)

★ Single Precision: $1 \le F \le 2 - 2^{-23}$
Double Precision: $1 \le F \le 2 - 2^{-52}$
} # Sum of GP

★ IEEE 754

★ Exponent bias usually $2^{k-1} - 1$ for a k bit exponent.

For 32 bit exponent is of 8 bit

∴ bias is 127

∴ $E = E' - 127$, $V = (-1)^S \times 1.M \times 2^E$

1	8	23
1	0111 1110	1000 0000 0000 0000 0000 000
S	E'	F(M)

★ $V = -1.1 \times 2^{(126-127)} = -1.1 \times 2^{-1}$

		Max	Min
Single Pres	$-126 \le E \le 127$	$(2-2^{-23}) \times 2^{127}$	1×2^{-126}
Double Pres	$-1022 \le E \le 1023$	$(2-2^{-52}) \times 2^{1023}$	1×2^{-1022}

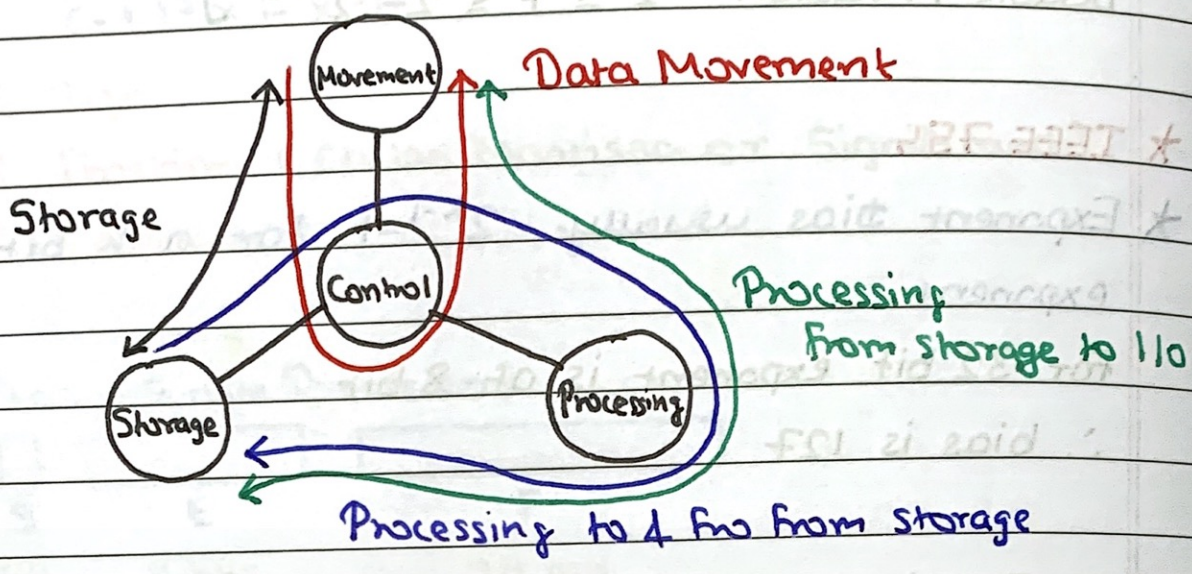
all 0's & all 1's have a special meaning.

★

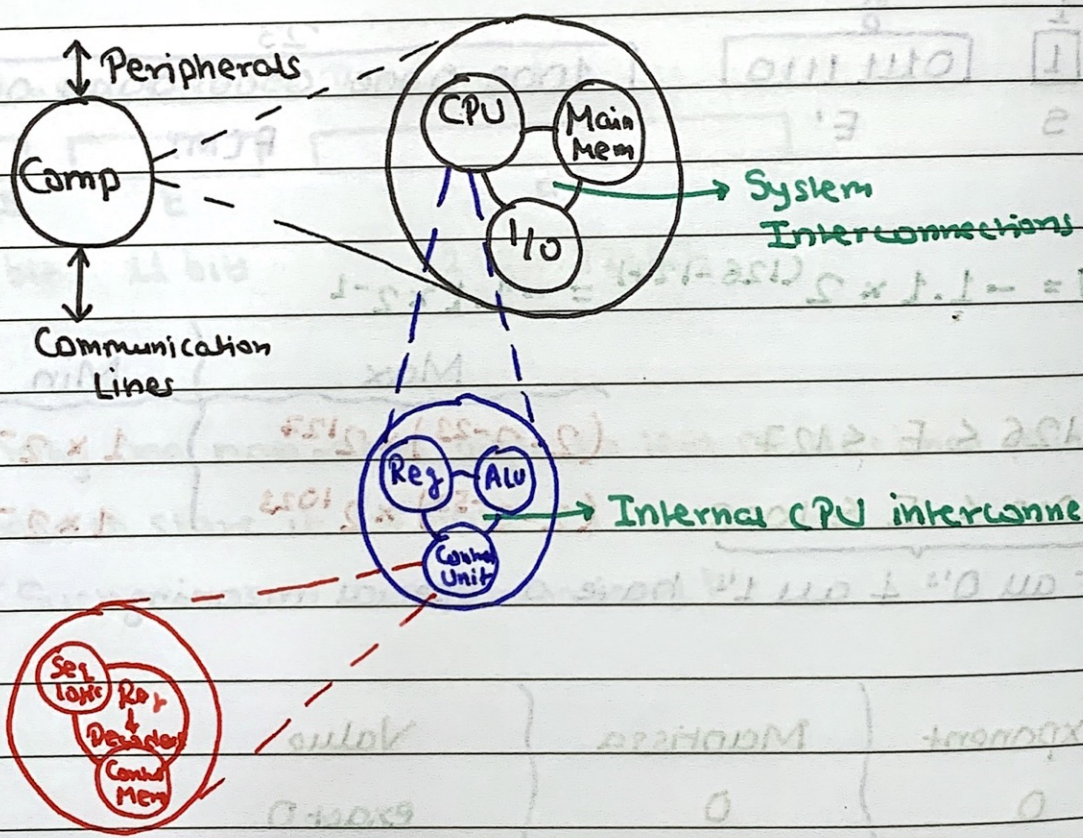
Exponent	Mantissa	Value
0	0	exact 0
255	0	∞
0	≠0	Denormalized
255	≠0	NAN

classmate
Date
Page

✶ Operations



✶



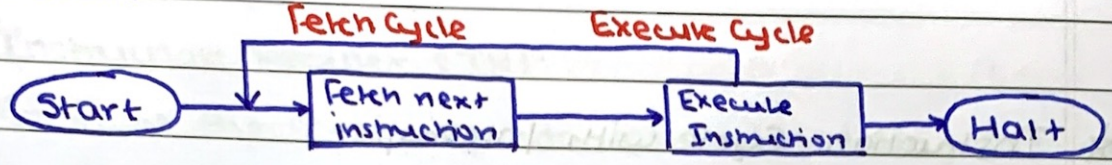
classmate
Date
Page

✶ Instruction Cycle

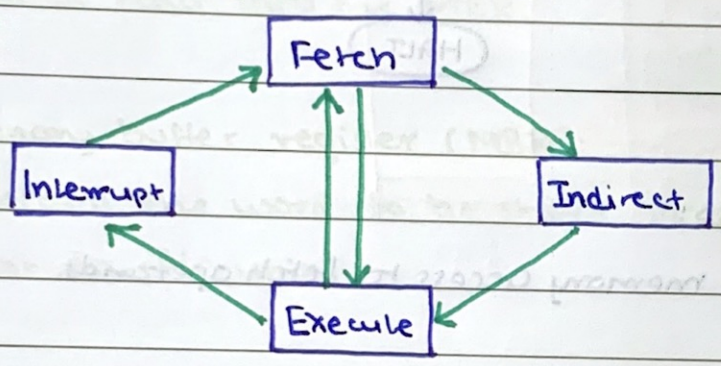
→ Two Steps:

→ Fetch

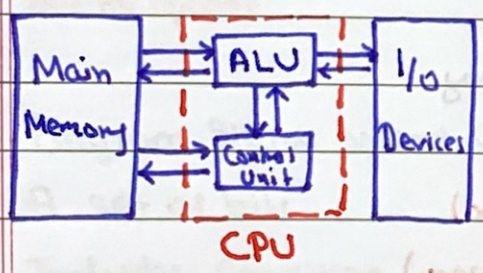
→ Execute



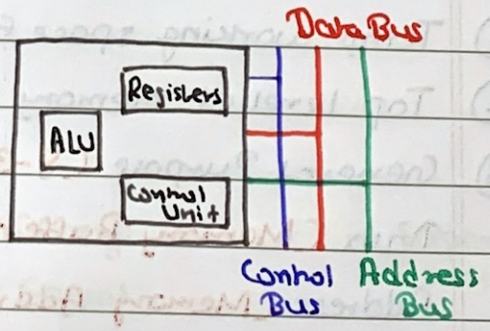
✶ Instruction Cycle with Indirect



✶ Von-Neuman Machine



✶ CPU with System Bus

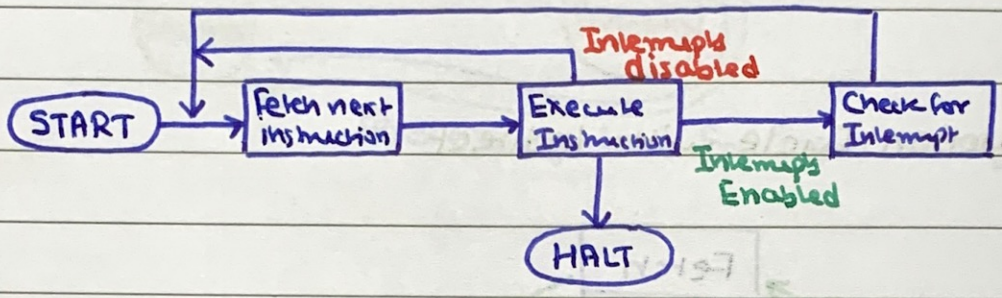


* K-bit address Bus : 2^K memory locations

n-bit data Bus : n bit memory / memory location

∴ Total memory : $2^K \times n$ bits $= \frac{2^K \times n}{8}$ bytes

* Instruction Cycle with Interrupts



* Indirect Cycle:

May require memory access to fetch operands.

* Registers

① Temp working space for CPU

② Top level of memory hierarchy

③ General Purpose (8-32)

Data (Memory Buffer Register)

Address (Memory Address Register)

Condition Codes (Usually can't be set by program)

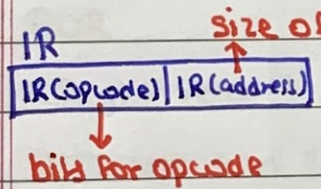
④ Control & Status Registers

→ Program Counter (PC):
Contains address of next instruction pair to be executed

→ Instruction Register (IR):
Contains the 8-bit opcode instruction being executed

→ Memory address register (MAR):
Specifies the address in memory of the word to be written from or read into the MBR.

→ Memory buffer register (MBR):
Contains the word to be stored in memory or to be sent to I/O unit.



⑤ Program Status Word (PSW)

→ A set of bits
→ Includes Condition Codes
→ Sign of last result
→ Zero
→ Carry
→ Equal
→ Overflow
→ Interrupt (Enable/Disable)
→ Supervisor (Sudo then 1)

* Bus:

① Communication pathway connecting two or more devices.

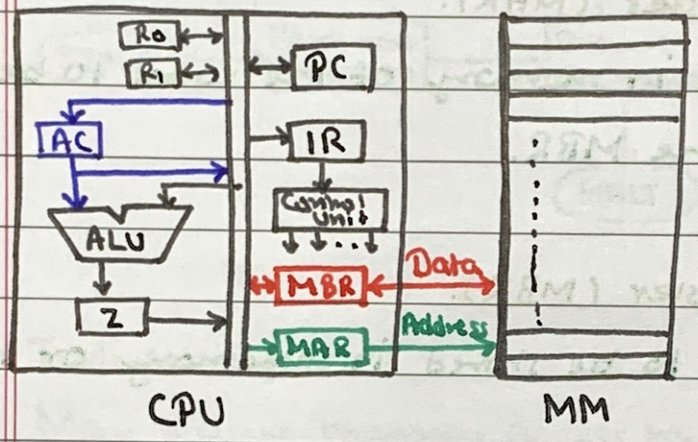
② Usually broadcast.

③ Often grouped

→ A no. of channels in one bus

→ 32 bit data bus is 32 seperate single bit channels.

*



classmate
Date
Page

* Instruction Set

① Machine Code: Binary

② Operation Code (Op Code)
- Do This

③ Source Operand Reference
- To This

④ Result Operand Reference
- Put ans here

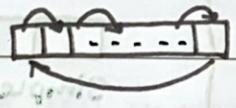
* SHR

(Logical right shift)

* ASR

(Arithmetic right shift)

* ROR



(Right rotate)

III^ly we can define for left.

* A = contents of an address field in the instruction that refers to a memory.

* R = contents of an address field in the instruction that refers to a register

* EA = Effective address of the location containing the referenced operand.

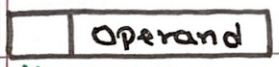
* (X) = contents of location X.

* Instruction: 

* Addressing Modes: *Principal Adv. *Principal Disadv.

a) Immediate

Operand = A

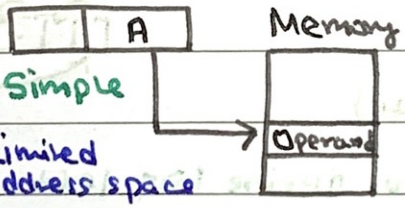


No memory reference

Limited Operand magnitude

b) Direct

EA = A

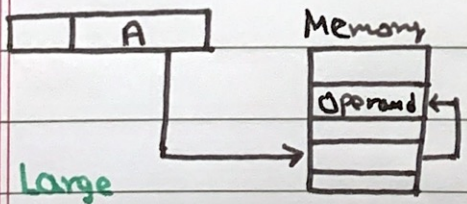


Simple

Limited Address space

c) Indirect

EA = (A) can be nested EA = ((A))n



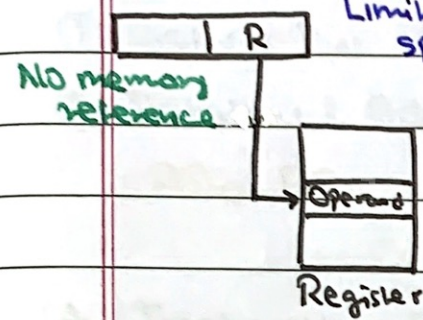
Large address space

Multiple memory reference

d) Register

EA = R

Limited address space

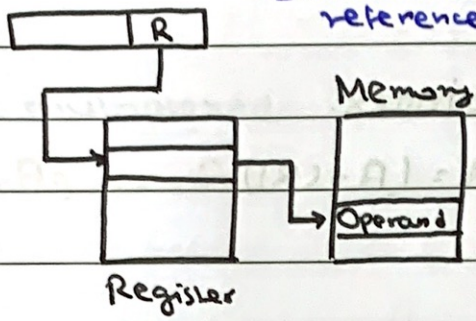


e) Register Indirect

EA = (R)

Extra memory reference

Large Address Space



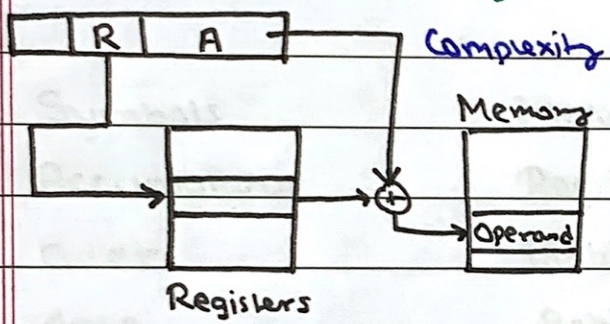
f) Displacement

EA = A + (R)

Base value

Flexibility

Complexity

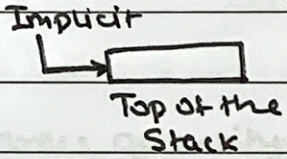


g) Stack

EA = top of the stack

No memory reference

Limited applicability



* Register Indirect has 1 less memory reference than Indirect

h) Relative addressing

EA = A + (PC)

i) Base-Register Addressing

EA = (A) + R

j) Indexed Addressing.

EA = A + R (A: base, R: displacement)

Good for accessing arrays

R++

classmate
Date
Page

★ Post index :

EA = (A) + (R)

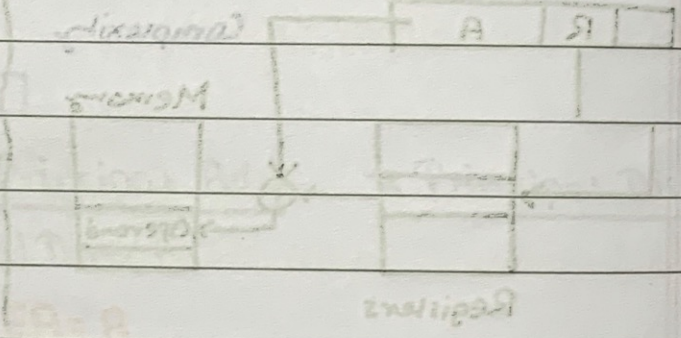
★ Auto indexing

EA = A + (R)

(R) = (R) + 1

★ Pre index:

EA = (A + (R))



★ 8085

① Data Bus: 8 bit

② Address Bus: 16 bits

③ Databus & Address Bus are multiplexed.

$AD_0, AD_1, \ldots, AD_7, A_8, A_9, \ldots, A_{15}$

★ Instruction format

→ One byte

→ Two bytes

→ Three bytes

→ Symbols	Meaning
Accumulator	Reg A
addr	16-bit address quantity
data	8-bit quantity
data 16	16-bit data quantity
byte 2	second byte of ins.
byte 3	third byte of ins.
r, r1, r2	One of the reg A, B, C, D, E, H, L
DDD, SSS	D: Destination, S: Source

A	111	H	100	10
B	000	L	101	
C	001	M	110	(Memory)
D	010	SP	11	
E	011			

00 (B, C); 01 (D, E)

★ Flag Register 8085

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
S	Z	X	AC	X	P	X	CY

① MSB = S (Sign)

② If result is zero after any arithmetic / logical op
Z = 1 (eg 01H + FFH then Z = 1) (Zero)

③ AC: Auxilary Carry
AC = 1 if ∃ carry from D_3 to D_4

eg: 2B
+ 39

∃ a carry # (B+9) ∴ AC = 1

④ P: Parity
P = 1: Ac even no. of 1's

⑤ CY: Carry
(n+1)th bit set then CY = 1

★ 1 Fetch cycle: 4 states
1 Read cycle: 3 states
1 Write cycle: 3 states

classmate
Date
Page

★ Groups

① Data Transfer:

② Arithmetic:

③ Logic:

④ Branch:

⑤ Stack, I/O, Machine control group:

★ Data Transfer

① MOV r_1, r_2 (Opcode)

$(r_1) \leftarrow (r_2)$

Opcode's M/C Code: 0 1 DDD SSS

Register, 1, 4

↓ Addressing ↓ Cycles ↓ States

None → Flags

② MOV r, M

$(r) \leftarrow ((H)(L))$

01 DDD 110

Register Indirect, 2, 7

None

③ MVI r, data

$(r) \leftarrow (data)$

Content of byte 2 moved to reg r

00 DDD 110

data

Immediate, 2, 7

None

④ MVI M, data

$((H)(L)) \leftarrow (data)$

00 110 110

data

Immed/reg indirect, 3, 10

None

⑤ LXI rp, data16 higher order reg ← (rh) ← (byte 3) eg H in HL (rl) ← (byte 2)	00 RP 0001 byte 2 byte 3 Immediate, 3, 10	} data 16	None
⑥ LHLD addr (L) ← ((byte 3)(byte 2)) (H) ← ((byte 3)(byte 2)+1) Contents of mem location moved to L & subsequent one to H	00101010 byte 2 byte 3 Direct, 5, 16		None
⑦ LDA addr (A) ← ((byte 3)(byte 2)) * 1 fetch, 3 reads	00111010 byte 2 byte 3 Direct, 4, 13		None

Like STA addr

★ Arithmetic

① ADD r (A) ← (A) + (r)	10000SSS Register, 1, 4	Z, S, P, CY, AC
② ADD M (A) ← (A) + ((H)(L))	10000110 Reg Indirect, 2, 7	Z, S, P, CY, AC

classmate
Date
Page

★ Branch

① JMP addr

(PC) ← (byte 3) (byte 2)

11000011	None
byte 2	
byte 3	
Immediate, 3, 10	

★ Condition

Condition	CCC
NZ	000
Z	001
NC	010
C	011
PO (P=0)	100
PE (P=1)	101
P (S=0)	110
M (S=1)	111

② Jcondition addr

IF (CCC) (PC) ← (byte 3) (byte 2)

11CCC010	None
byte 2	
byte 3	
Immediate, 3, 10	

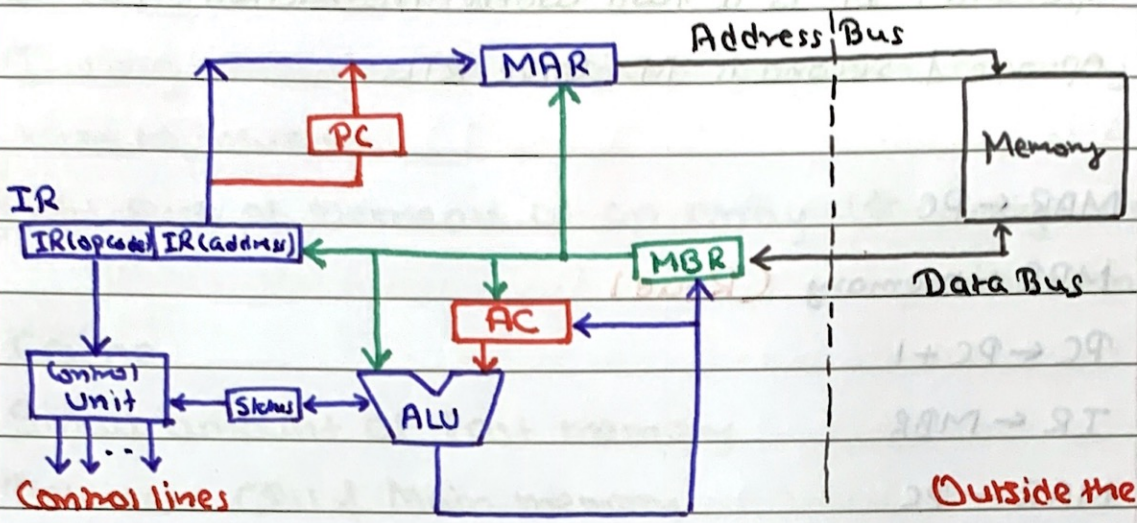
★ Stack

① Call addr	11 001101	None
	byte 2	
	byte 3	
	Res indirect, 5, 18	

→ Write a program to add 100 (64H) integers stored in memory locations starting from 3000H & store result in memory location 4000H

Label	Memory	Op-Code	MIC Code	MIC in HEX, Data
	1000	MVI B, 64H	00 000110	06 64
	1002	LXI H, 3000H	00 10 0001	21 00 30
	1005	MVI A, 00H	00 111 110	3E 00
loop	1007	ADD M	10000110	86
	1008	INX H	0010 0011	23
	1009	DCR B	00 000 101	05
	100A	JNZ loop	11 000 010	C2 07 10
	100D	STA 4000H	00110010	32 00 40
	1010	HLT	0111 0110	76

★ Register Transfer Language (RTL)



★ Instruction fetch

t1 MAR ← PC

t2 MBR ← memory (Read)

PC ← PC+1

t3 IR ← MBR

★ Execute LDA X

t4 MAR ← IR(address)

t5 MBR ← memory (Read)

t6 AC ← MBR

★ Execute JMP X

t4 PC ← IR(address)

* Consider $R_1 = R_1 + (M)$. M memory address of the operand. It is a two word instruction. First word opcode & second is M. Give RTL.

t1 MAR ← PC

t2 MBR ← memory (Read)
PC ← PC + 1

t3 IR ← MBR

t4 MAR ← PC

t5 MBR ← memory (Read)
PC ← PC + 1

t6 MAR ← MBR

t7 MBR ← memory (Read)

t8 $R_1 \leftarrow R_1 + MBR$