**Mano basic computer**

Instruction format of basic computer

|  |  |  |
| --- | --- | --- |
| 15 mode bit(I) | 12-14 opcode 4bit | 0-11 address 12bit |

Addressing mode in basic computer:

When I = 0 → direct addressing mode (the address in operand field is effective address)

When I = 1 → indirect addressing mode (the address in operand field contains memory address where effective address resides)

Types of instructions:

1. Memory reference instruction

2. Register reference instruction

3. I/O reference instruction

Memory reference instructions:

|  |  |  |  |
| --- | --- | --- | --- |
| Symbol | I = 0 | I = 1 | Description |
| AND | 0xxx | 8xxx | AND memory word to AC |
| ADD | 1xxx | 9xxx | add memory word to AC |
| LDA | 2xxx | Axxx | Load AC to memory |
| STA | 3xxx | Bxxx | Store AC content to memory |
| BUN | 4xxx | Cxxx | Branch unconditionally |
| BSA | 5xxx | Dxxx | Branch and store return address |
| ISZ | 6xxx | Exxx | Increment and skip if 0 |

Register Reference instructions:

|  |  |
| --- | --- |
| 15-12 0111 | 11-0 register operation |

|  |  |  |
| --- | --- | --- |
| Symbol | hexcode | description |
| CLA | 7800 | Clear AC |
| CLE | 7400 | Clear E flag |
| CMA | 7200 | Complement AC |
| CME | 7100 | Complement E flag |
| CIR | 7080 | Circulate right AC and E |
| CIL | 7040 | Circulate left AC and E |
| INC | 7020 | Increment AC |
| SPA | 7010 | Skip next instruction if AC is +ve |
| SNA | 7008 | Skip next instruction if AC is -ve |
| SZA | 7004 | Skip next instruction if AC is 0 |
| SZE | 7002 | Skip next instruction if E flag is zero |
| HLT | 7001 | Halt computer |

I/O reference instructions:

|  |  |
| --- | --- |
| 15-12 1111 | 11-0 I/O operation |

|  |  |  |
| --- | --- | --- |
| Symbol | Hex code | description |
| INP | F800 | Input character to AC |
| OUT | F400 | Output character form AC |
| SKI | F200 | Skip on input flaag |
| SKO | F100 | Skip on output flag |
| ION | F080 | Interupt on |
| IOF | F040 | Interupt off |

For an instrucction set of a CPU to be complete it must must have the folloeing instructions:

1. Data manipulation instructions

→ Arithmatic instructions (ADD, SUB, INC, DEC, etc)

→ Logical instructions (AND, OR, XOR, etc)

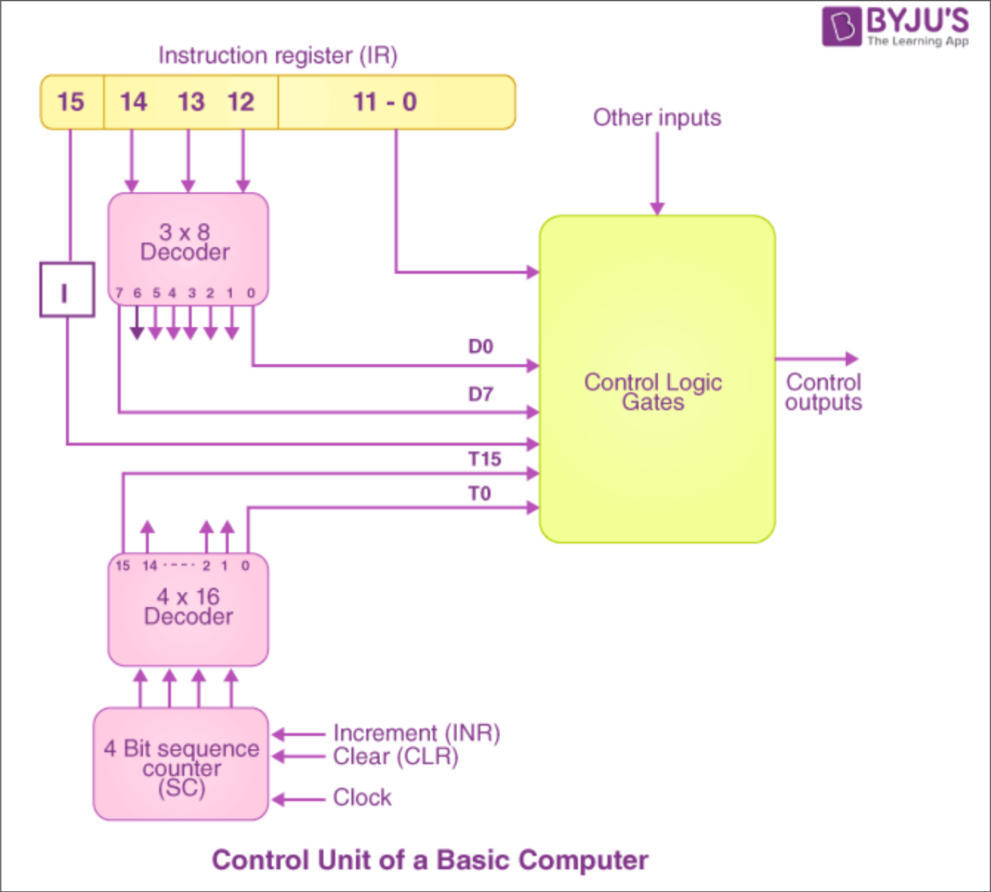
→ Shift instructions (SHR, SHL, etc)

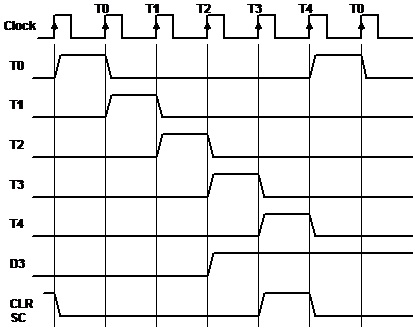
2. Data transfer instructions (MOV, LDA, STA, etc)

3. Branching instructions (BUN, BSA, ISZ, etc)

4. I/O instructions (INP, OUT, etc)

**Hardwired Control Unit:**





Timing diagram for D3T4 : SC ← 0

**Instruction cycle:**

1. Fetch Instruction

2. Decode Instruction

3. Catlculate Operand Address (read the effective address from the memory if the instruction has indirect address)

4. Execute Instruction

Fetch:

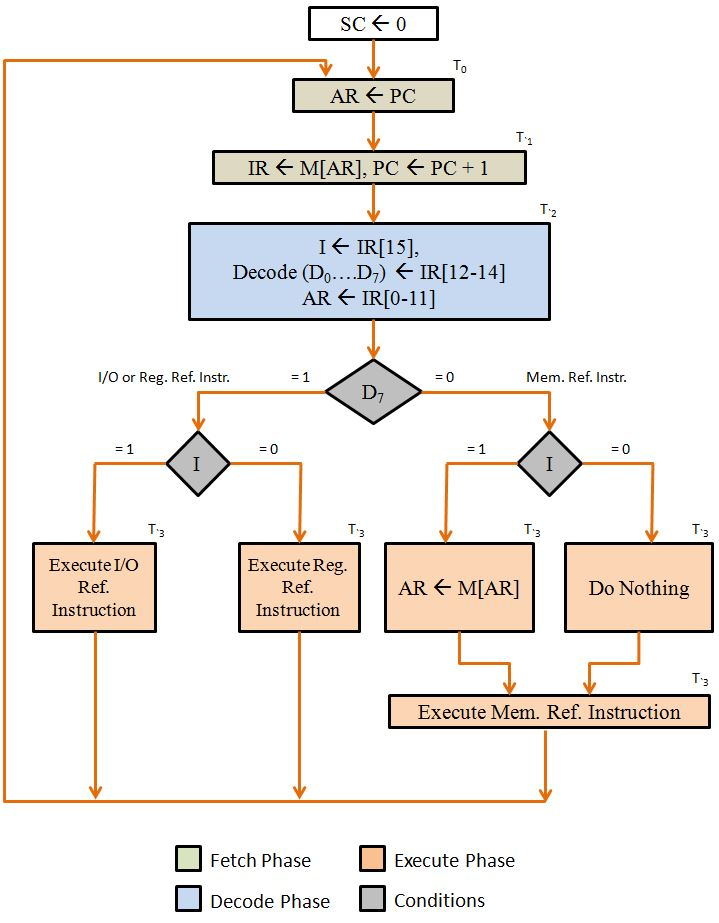
T0 : AR ← PC

T1 : IR ← M[AR] , PC ← PC + 1

Decode:

T2 : D0 , … , D7 ← Decode IR(12-14) , AR ← IR(0-11) , I ← IR(15)

Flowchart:



Register reference instructions:

D7 = 0 , I = 0

→ reg ref instructions are specified in bits b0 – b11 IR

→ execution starts in time T3

Let r = D7I’T3 common to all register reference instructions

Bi = IR(i) i = 0,1,2,…,11

CLA = rB11 AC ← 0 , SC ← 0

CLE = rB10 E ← 0 , SC ← 0

CMA = rB9 AC ← AC’ , SC ← 0

CME = rB8 E ← E’ , SC ← 0

CIR = rB7 AC ← shrAC , AC(15) ← E , E ← AC(0) , SC ← 0

CIL = rB6 AC ← shrAC , AC(0) ← E , E ← AC(15) , SC ← 0

INC = rB5 AC ← AC + 1 , SC ← 0

SPA = rB4 if(AC(15)=0) then PC ← PC + 1 , SC ← 0

SNA = rB3 if(AC(15)=1) then PC ← PC + 1 , SC ← 0

SZA = rB2 if(AC=0) then PC ← PC + 1 , SC ← 0

SZE = rB1 if(E=0) then PC ← PC + 1 , SC ← 0

HTL = rB0 S ← 0 , SC ← 0

Memory reference instructions:

AND D0 AC ← AC ^ M[AR]

ADD D1 AC ← AC + M[AR]

LDA D2 AC ← M[AR]

STA D3 M[AR] ← AC

BUN D4 PC ← AR

BSA D5 M[AR] ← PC , PC ← AR + 1

ISZ D6 M[AR] ← M[AR] + 1 , IF M[AR] + 1 =0 THEN PC ← PC + 1

AND

D0T4 = DR ← M[AR]

D0T5 = AC ← AC ^ DR , SC ← 0

ADD

D1T4 = DR ← M[AR]

D1T5 = AC ← AC + DR , SC ← 0

LDA

D2T4 = DR ← M[AR]

D2T5 = AC ← DR , SC ← 0

STA

D3T4 = M[AR] ← AC , SC ← 0

BUN

D4T4 = PC ← AR , SC ← 0

BSA

D5T4 = M[AR] ← PC , AR ← AR + 1

D5T5 = PC ← AR , AC ← 0

ISZ

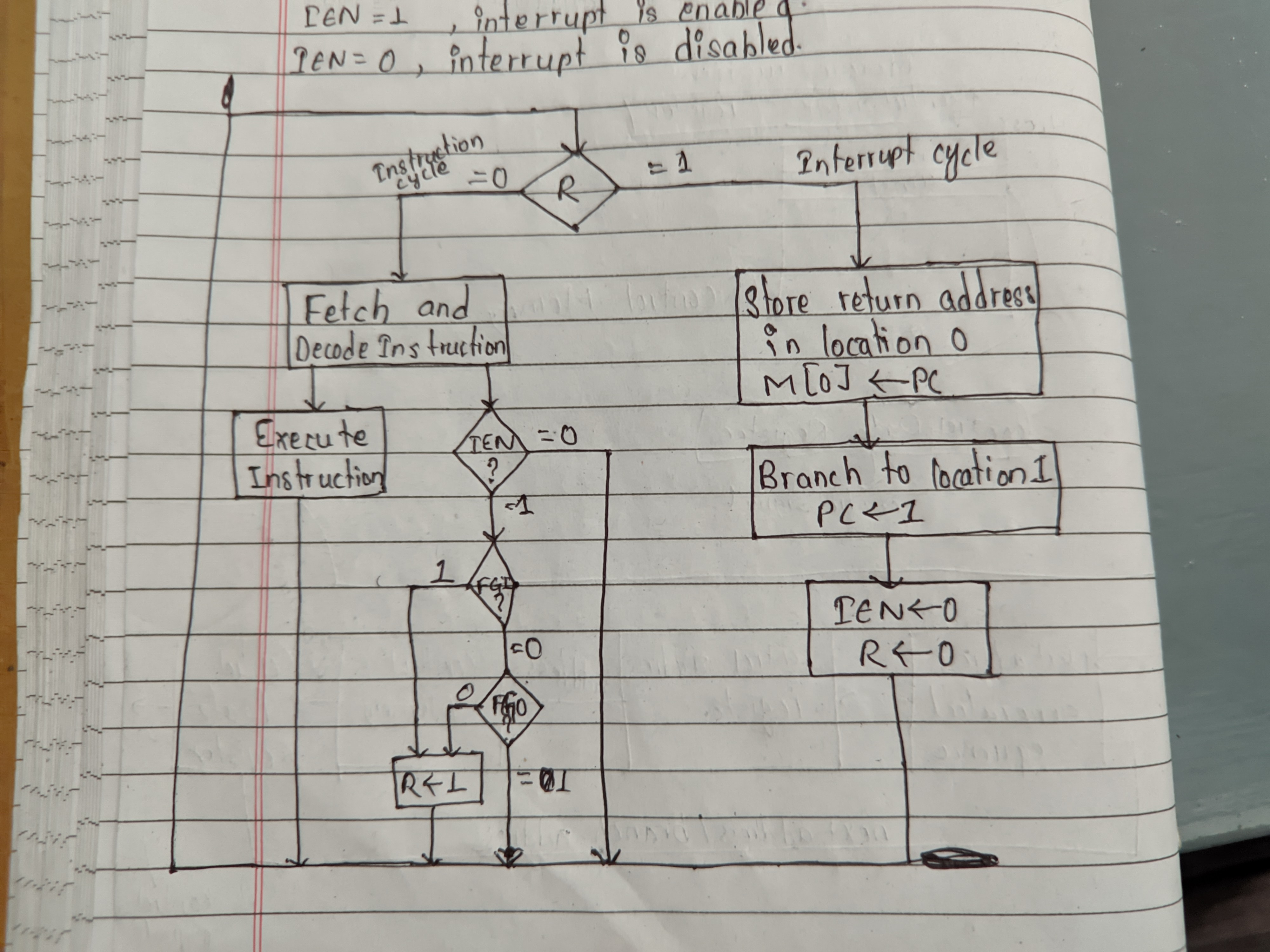
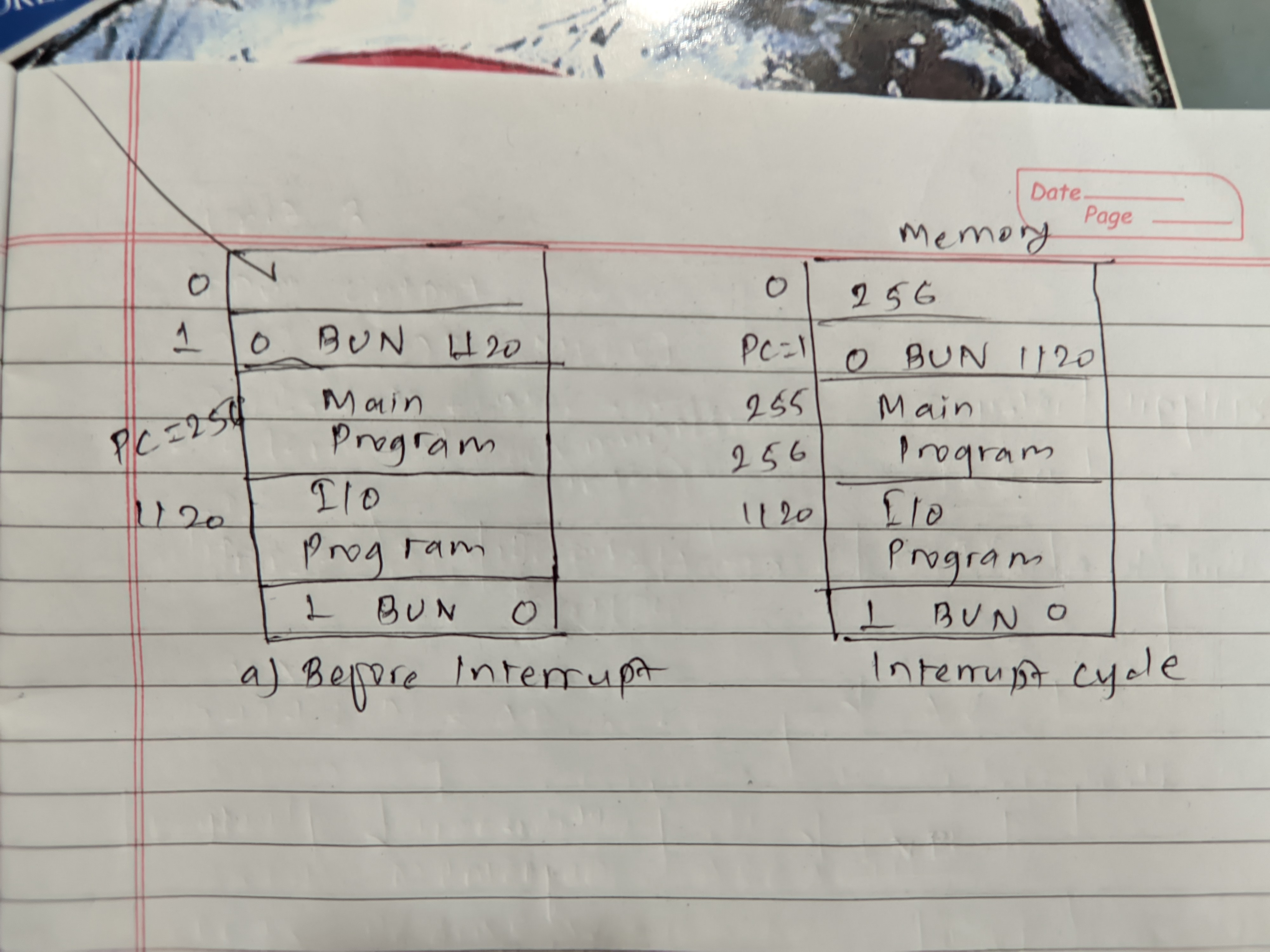
D6T4 = DR ← M[AR]

D6T5 = DR ← DR + 1

D6T6 = M[AR] ← DR , IF(DR=0) THEN (PC ← PC + 1) , SC ← 0

**Interupt:**

IEN → Interrupt enable flipflop (1:enabled,0:disabled)



Complete Flowchart:

