Design of scaled-down version of 8085 microprocessor

(Self Project submission for resume verification)

Name: Aditya Ajay Wani **Roll No: 193079020**

Github Repo: https://github.com/25Aditya25/mini8085

Introduction:

This implementation is a scaled-down version of the 8085 processor with 18 instructions based on the problem statement given as an assignment for the course EE739 Processor Design at IIT Bombay by Prof. Virendra Singh.

The GitHub repo also has the problem statement. The design was done on paper as mentioned in the problem statement. The work was done in march and april of 2020 in the pandemic. The grading of the assignment was supposed to happen when the tem resumes but it didn't happen. So I am including this assignment as a self-project.

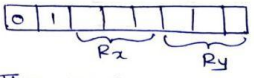
The Design details are as follows:

The level 1 flow charts for instructions is shown below:-

Name:- Aditya Ajay Wani
Roll No:- 193079020

1] mov R_x, R_y

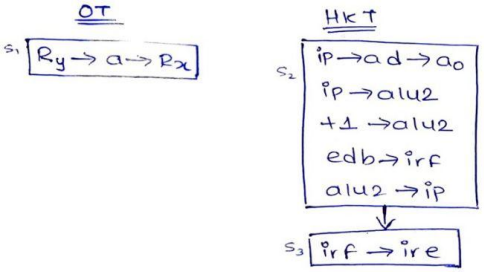
8bit instruction



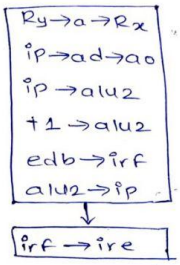
The register addresses are as below,

Register	address
B	000
C	001
D	010
E	011
A	111
H	100
L	101
memory	110

The level 1 flowchart of this instruction is shown below,

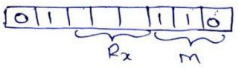


States S_1 can be merged with state S_2. The merged level 1 flowchart is as shown below,

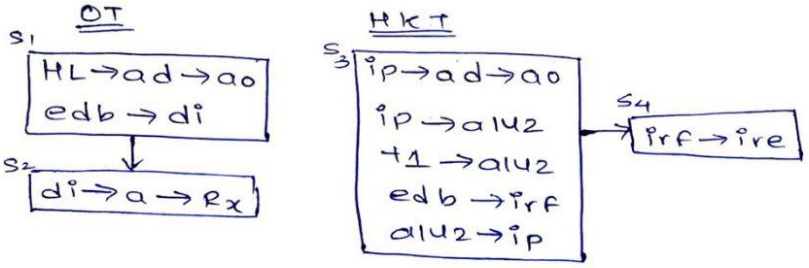


2] mov R_x, M 8bit instruction.

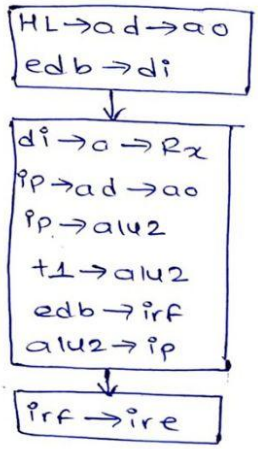
move the contents of memory into register R_x



The level 1 flowchart for the above instruction is shown below:-

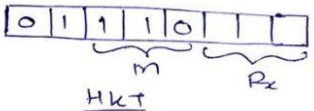


Here we can merge states S_2 & S_3 to get the merged level 1 flowchart as below

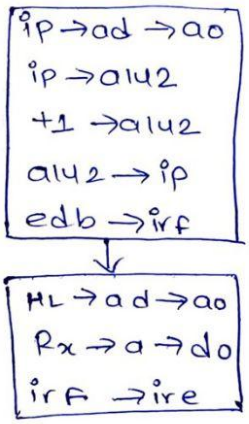


3] MOV M, Rs

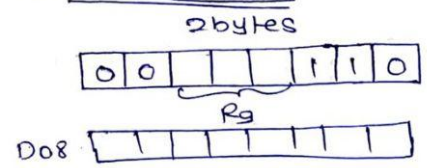
8 bit instruction:



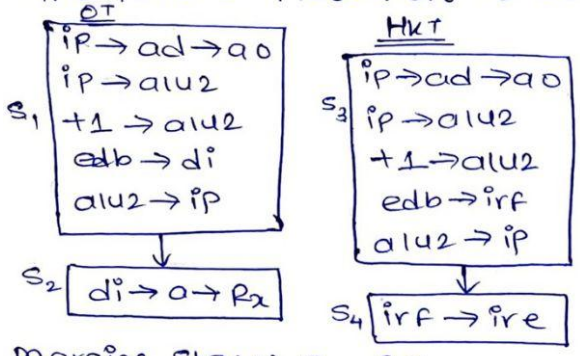
merging states S_1 & S_3 we get level 1 merged flowchart as below,



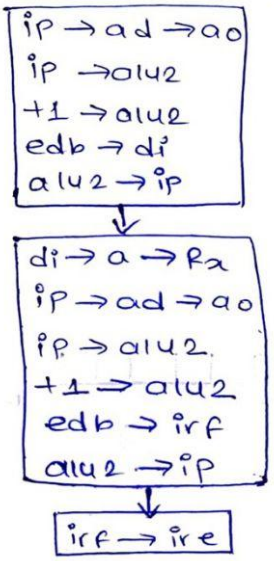
4] MVI Rg, D08



The level 1 flowchart is as shown below,

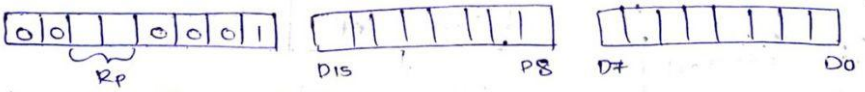


Merging states S_2 & S_3 we get, Merged level 1 flowchart.



5] LXI Rp/Sp, D16.

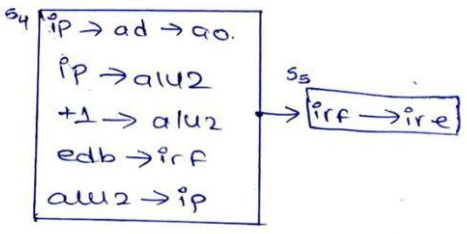
3 byte instruction:

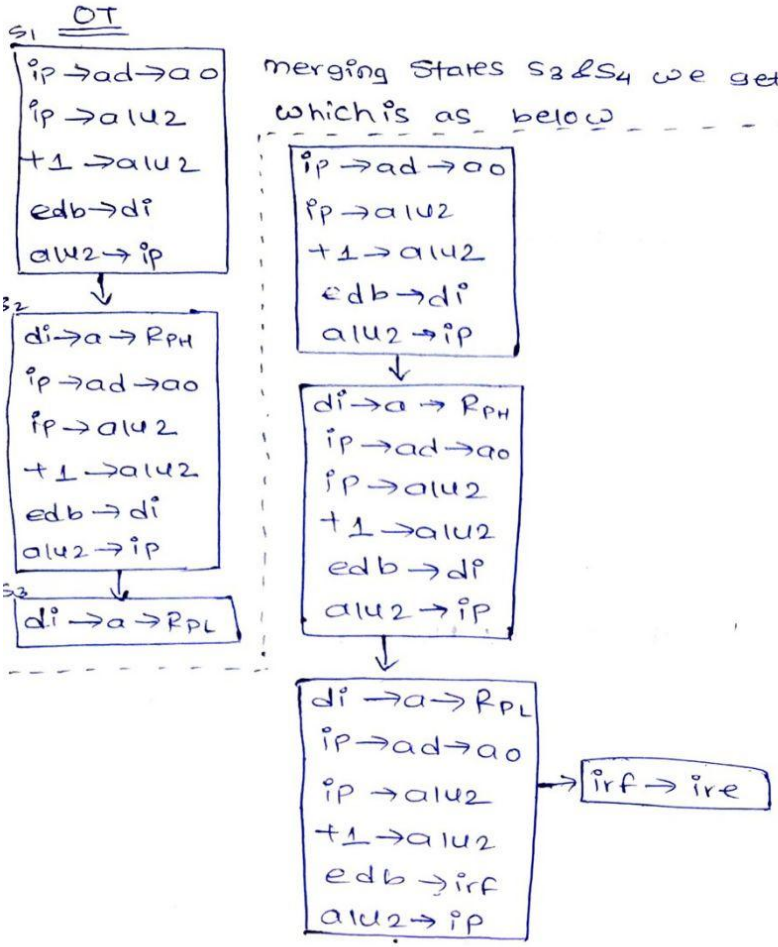


The Register pair address are as below

Rp	address
Bc	00
DE	01
HL	10
Sp	11

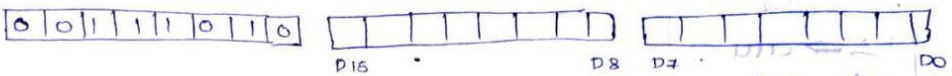
HkT

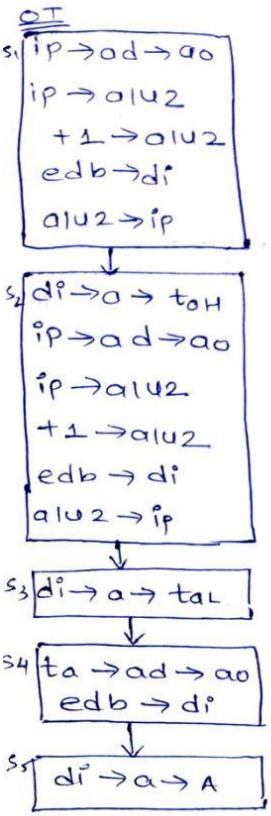


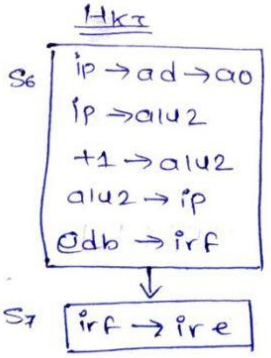


merging States S_3 & S_4 we get the Level 1 merged flowchart which is as below

6] LDA D16 3 bytes :-







combining States S_3 & S_6 & S_5 & S_7 we get the Level 1 merged flowchart as follows.

ip → ad → ao
ip → alu 2
+1 → alu 2
edb → di
alu2 → ip

↓

di → a → tDH
ip → ad → ao
ip → alu2
+1 → alu2
edb → di
alu2 → ip

↓

di → a → tAL
ip → ad → ao
ip → alu2
+1 → alu2
alu2 → ip
edb → irf

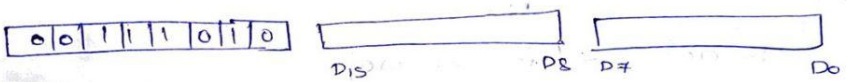
↓

ta → ad → ao
edb → di

↓

di → a → A
irf → ire

→] STA D16 3 bytes:



07

S_1 ip → ad → ao
ip → alu2
+1 → alu2
edb → di
alu2 → ip

↓

S_2 di → a → tAH
ip → ad → ao
ip → alu2
+1 → alu2
edb → di
alu2 → ip

↓

S_3 di → a → tAL

↓

S_4 tA → ad → ao
A → a → do

HKT

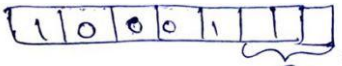
S_5 ip → ad → ao
ip → alu2
+1 → alu2
alu2 → ip
edb → irf

↓

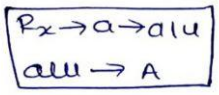
S_6 irf → ire

Combining States S_3, S_5 and S_4 & S_6 we get the merged Level 1 flowchart as below.

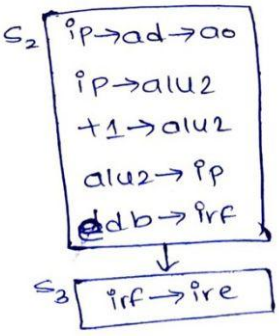
8] ADC Rg : add with carry. 1 byte:-

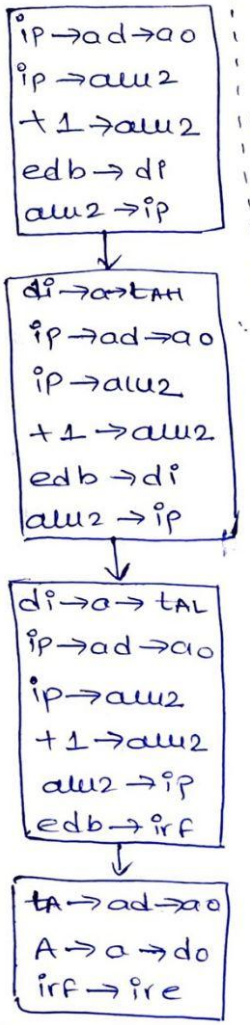


OT

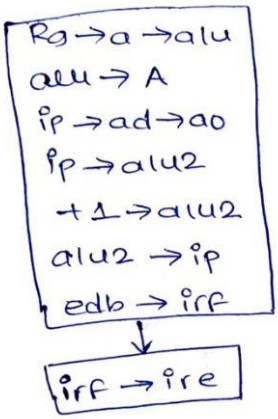


HKT

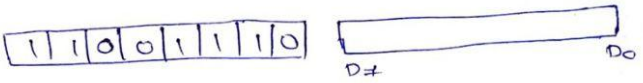




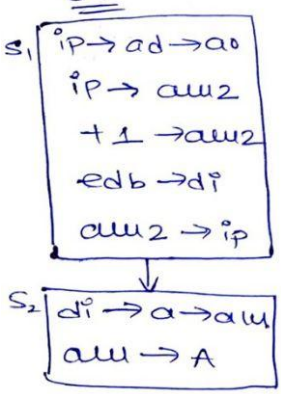
Combining S_1 & S_2 we get merged level 1 flowchart



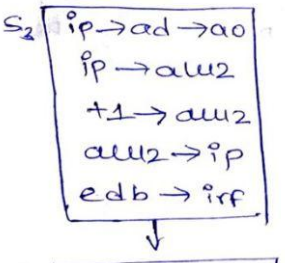
9] ACI D08 2 bytes:-



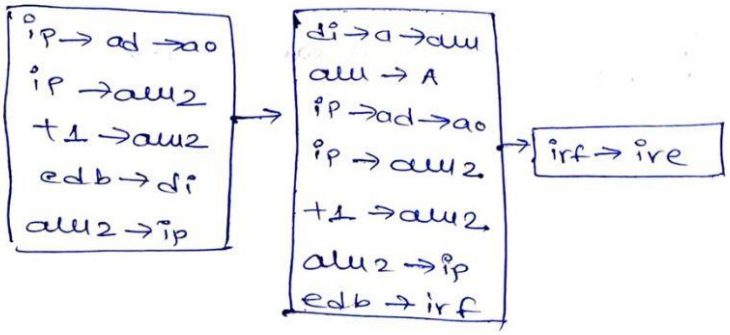
OT



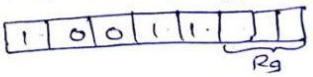
HKT



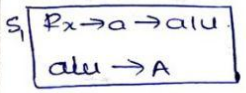
Combining States S_2 & S_3 we get merged level 1 flowchart as below



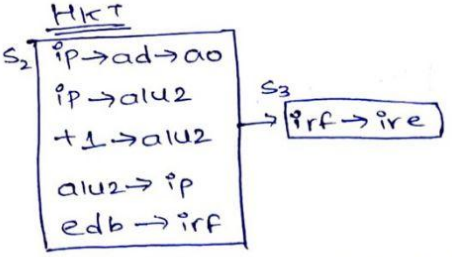
10] SSB Rg : subtract Rg and borrow from Accumulator.

1 byte : 

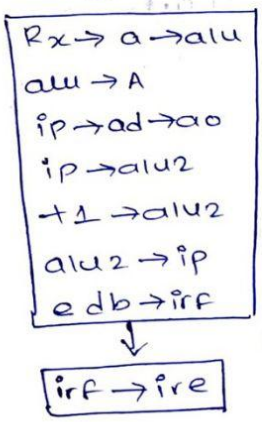
OT



HKT

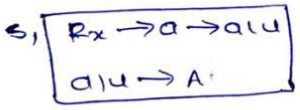


merging S_1 & S_2 we get merged level 1 flowchart.

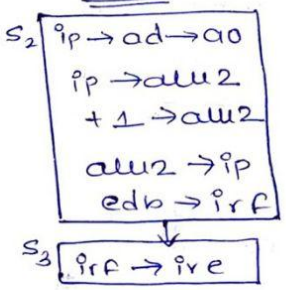


11] ANA Rg AND Accumulator with Rg

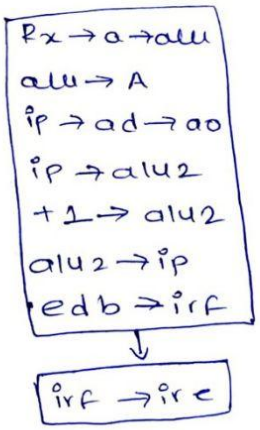
OT



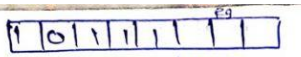
HKT



merging states. S_1 & S_2 we get merged LVL 1 flowchart



(2) cmp Rg



OT

S_1: A → a → t1

S_2: Rg → a → alu1; alu1 → A

S_3: t1 → a → A

HKT

S_4: ip → ad → ao; ip → alu2; +1 → alu2; alu2 → ip; edb → irf

S_5: irf → ire

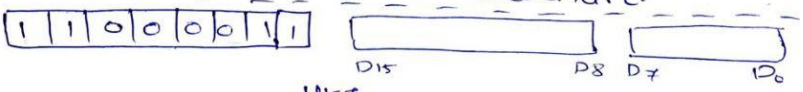
merge S_4 & S_1 & S_5 & S_2 we get merged level 1 flowchart.

A → a → t1
ip → ad → ao
ip → alu2
+1 → alu2
alu2 → ip
edb → irf

Rg → a → alu
alu → A

t1 → a → A
irf → ire

(3) Jmp D16



OT

S_1: ip → ad → ao; ip → alu2; +1 → alu2; edb → di; alu2 → ip

S_2: di → a → tAH

S_3: di → a → tAL

S_4: tA → ad → ip

ip → ad → ao
ip → alu2
+1 → alu2
edb → di
alu2 → ip

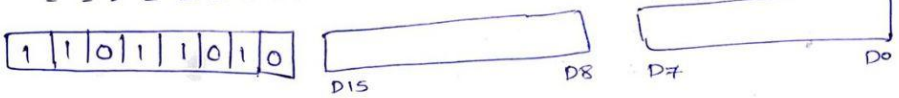
HKT

S_5: ip → ad → ao; ip → alu2; +1 → alu2; alu2 → ip; edb → irf

S_6: irf → ire

(3) Here No merging is preferred to save no. of control stores.

(4) JC D16



ip → ad → ao
ip → alu2
+1 → alu2
edb → di
alu2 → ip

Here also no merging is possible.

di → a → tAH
ip → ad → ao
+1 → alu2
edb → di
alu2 → ip

c=0

ip → ad → ao
ip → alu2
+1 → alu2
alu2 → ip
edb → irf

irf → ire

c=1

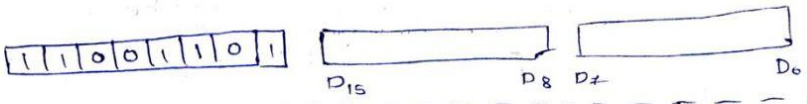
di → a → tAL

tA → ad → ip

ip → ad → ao
ip → alu2
+1 → alu2
alu2 → ip
edb → irf

irf → ire

15] CALL D16 3 bytes.



OT

S1: Sp→ad→ao; Sp→alu2; +1→alu2; alu2→Sp; iPL→a→do

S2: Sp→ad→ao; Sp→alu2; +1→alu2; alu2→Sp; iPH→a→do

S3: ip→ad→ao; ip→alu2; +1→alu2; alu2→ip

S4: di→a→tAH; ip→ad→ao; ip→alu2; +1→alu2; alu2→ip; edb→di

S5: di→a→tAL

S6: tA→ad→ip

HKT

S7: ip→ad→ao; ip→alu2; +1→alu2; alu2→ip; edb→irf

S8: irf→ire

The merged level1 flow chart is as shown below.

Sp→ad→ao; Sp→alu2; +1→alu2; alu2→Sp; iPL→a→do

Sp→ad→ao; Sp→alu2; +1→alu2; alu2→Sp; iPH→a→do

ip→ad→ao; ip→alu2; +1→alu2; edb→di; alu2→ip

di→a→tAH; ip→ad→ao; +1→alu2; alu2→ip; edb→di; ip→alu2

di→a→tAL

tA→ad→ip

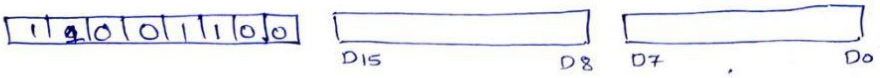
ip→ad→ao; ip→alu2; +1→alu2; alu2→ip; edb→irf

→ irf→ire

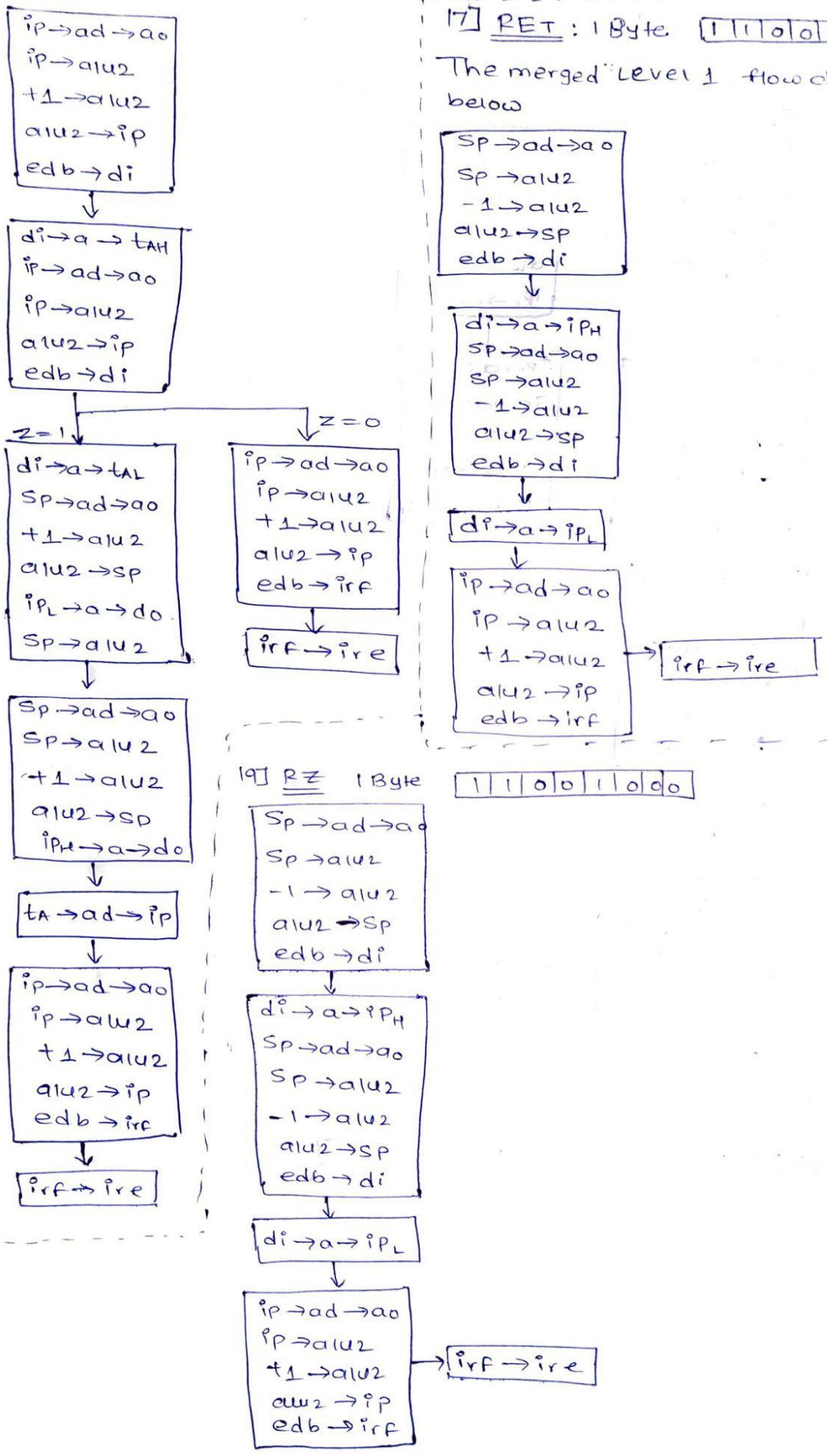
16] CZ D16

call if zero flag is set.

3 bytes



The merged level 1 flowchart is as shown below,



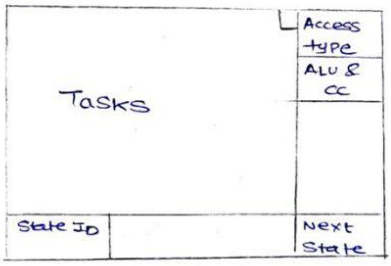
[7] RET : 1 Byte | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |

The merged level 1 flow chart is shown below

[9] RZ 1 Byte | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |

Level 2 flowcharts:-

The templates of the level 2 flowchart is shown below:-



Access Type:-

1) DR: Dataread
2) DW: Data write
3) IR: Instruction read
4) NA: No Access

Next State:-

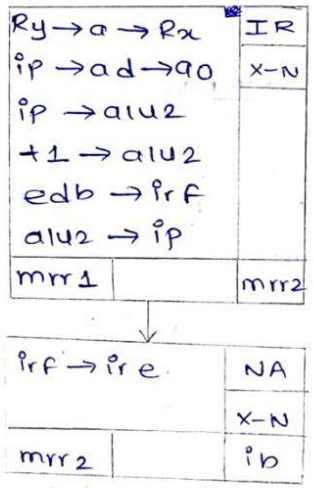
1) BC: Branch conditionally
2) IB: Instruction Branch
3) SB: Sequence Branch
4) State ID: Direct transfer

ALU & cc

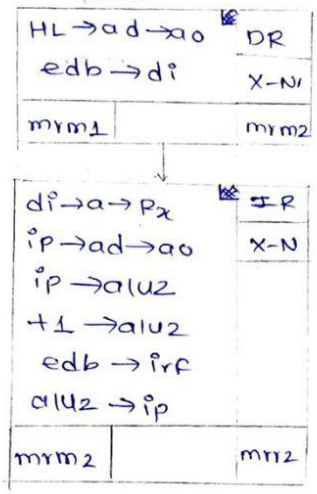
S: Set the cc
N: Do not set the cc
X: Don't care

The Level 2 flowcharts for all the instructions are as follows:

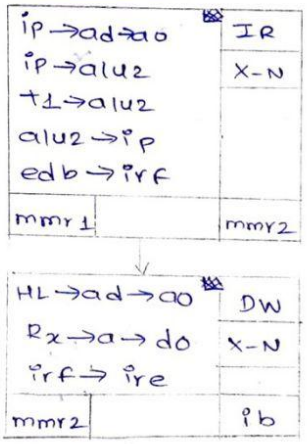
1] MOV Rg, Rg:-



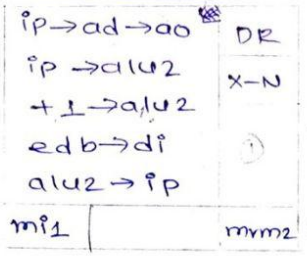
2] MOV Rg, M



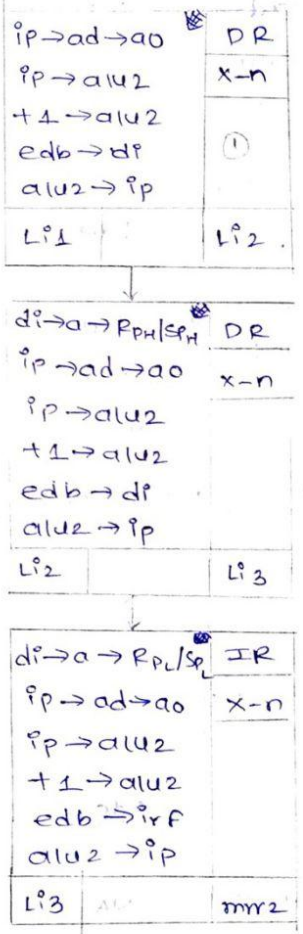
3] MOV M, Rg



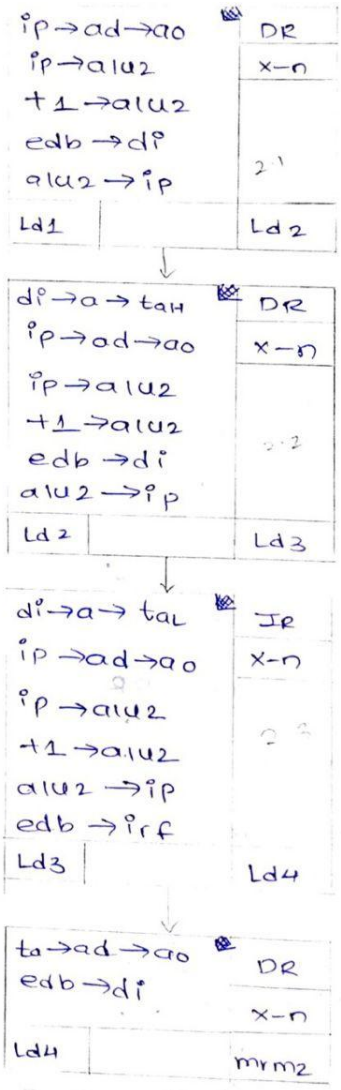
4] MVI Rg, D08



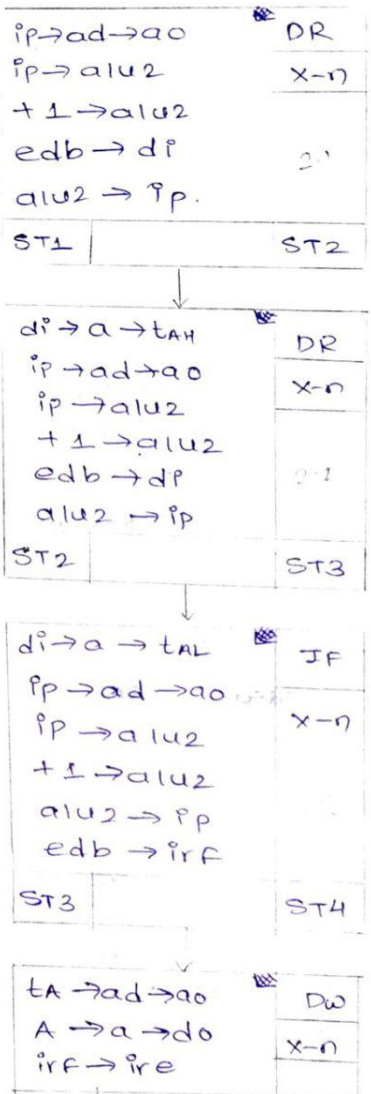
5] LXI RP/SP D16



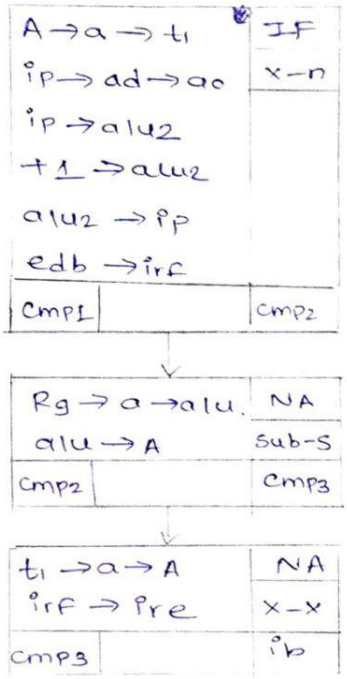
6] LDA D16



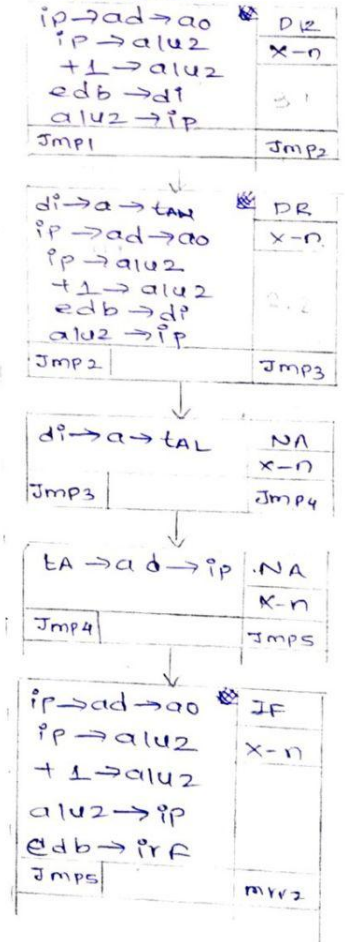
7] STA D16



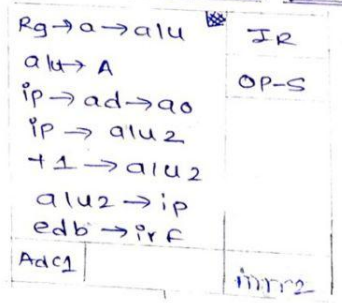
12] cmp Rg



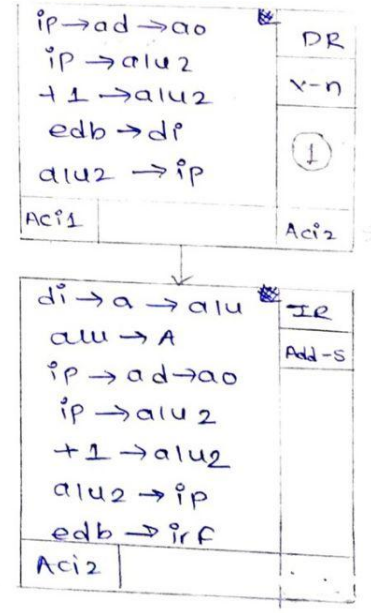
13] Jmp D16



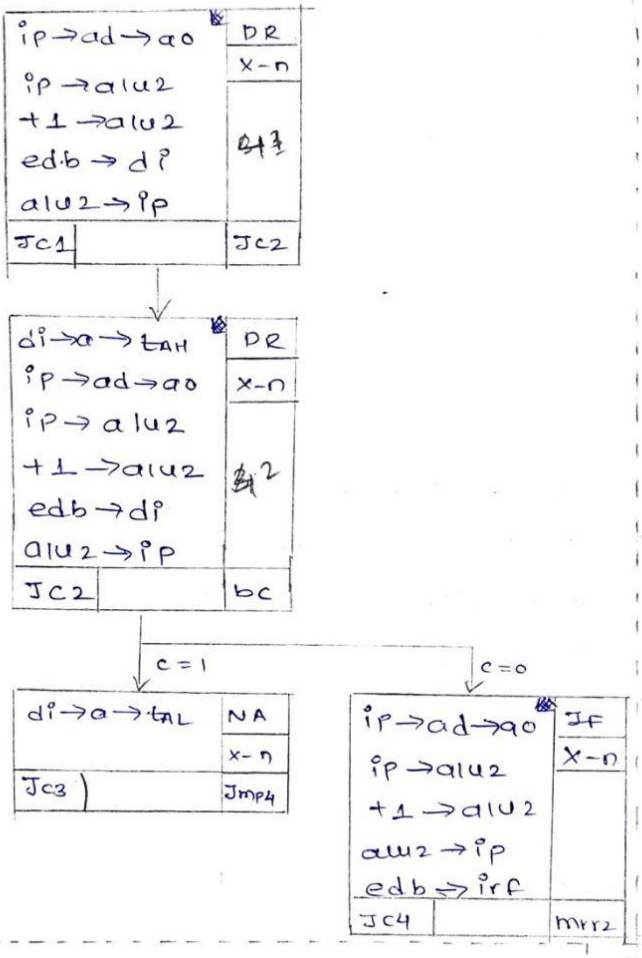
8] ADC Rg, SSB Rg, ANA Rg



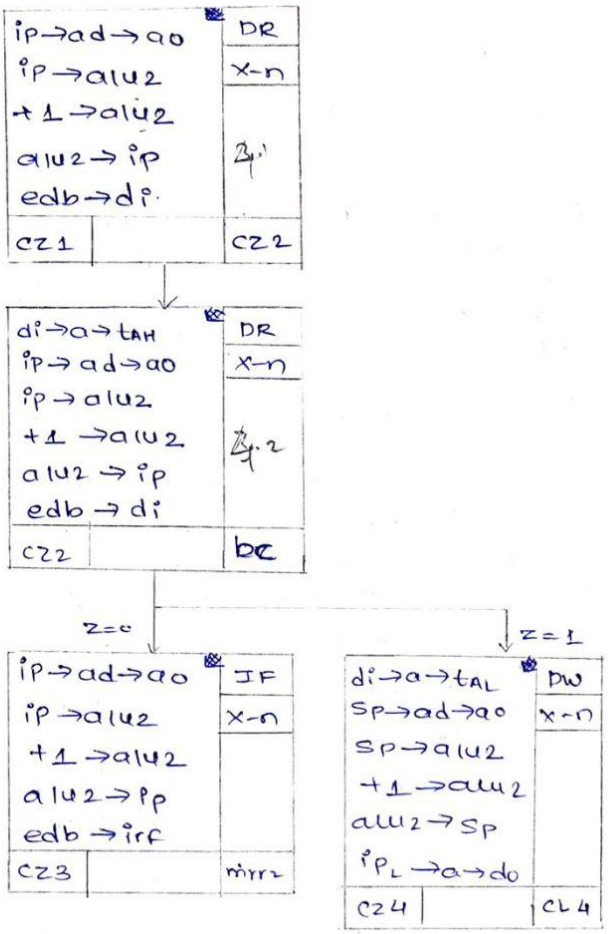
9] ACI D8



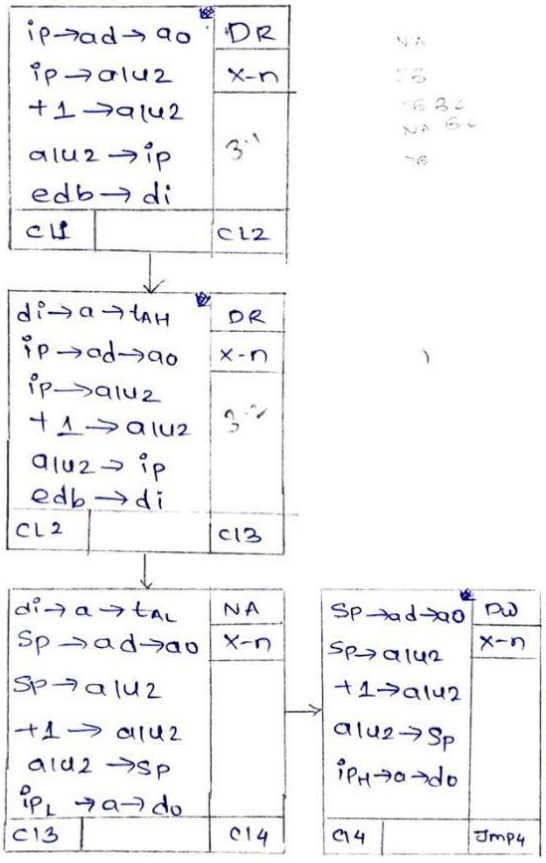
[4] JC D16



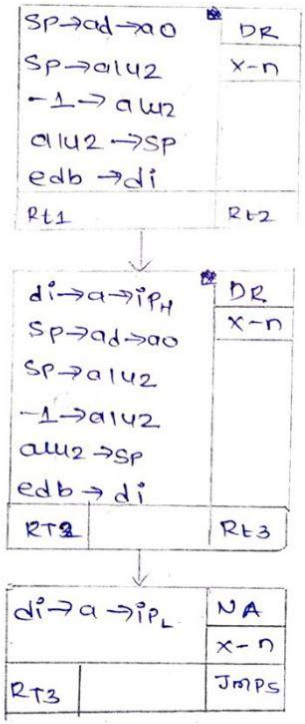
[6] CZ D16



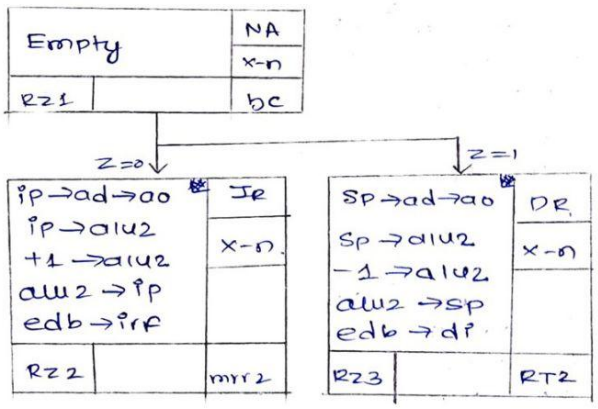
[5] call D16



[7] Ret



18] Rz



Now we have 48 different states, so each of the states will have its own control word. So we will require 48 control words. no. of bits for control store addresses = 6.

The addresses have been given to the different states as below.

000000 - mrr1
000001 - mrr2
000010 - mrm1
000011 - mrm2
000100 - mmr1
000101 - mmr2
000110 - mi1
000111 - LI1
001000 - LI2
001001 - LI3
001010 - Ld1
001011 - Ld2
001100 - Ld3
001101 - Ld4
001110 - Adc1
001111 - Adc2
010000 - ST1
010001 - ST2
010010 - ST3
010011 - ST4
010100 - Aci1
010101 - Aci2
010110 - cmp1
010111 - cmp2
011000 - cm3
011001 - Jmp1
011010 - Jmp2
011011 - Jmp3
011100 - Jmp4
011101 - Jmp5
011110 - Jc1
011111 - Jc2
100000 - Jc3
100001 - Jc4
100010 - cl1
100011 - cl2
100100 - cl3
100101 - cl4
100110 - cz1
100111 - cz2
101000 - cz3
101001 - cz4
101010 - Rt1
101011 - Rt2
101100 - Rt3
101101 - Rz1
101110 - Rz2
101111 - Rz3

Now we can further reduce the total number of control words by Naming the same ~~br~~ states as one and using sequence branch. This was done for the instructions MVI Rg, D08, LXI Rp/Sp D16, LDA D16, STA D16, ACI D08, Jmp D16, Jc D16, call D16 & cz D16.
The Level 2 flowchart for these instructions after adding sequence branches is as below.

4) MVI Rg, D08

ip→ad→ao	DR
ip→alu2	x-N
+1→alu2	
edb→di	
alu2→ip	
1	SB mrm2

5) LXI Rp/Sp D16

ip→ad→ao	DR
ip→alu2	x-n
+1→alu2	
edb→di	
alu2→ip	
1	SB Li2

di→a→RPH/SPH	DR
ip→ad→ao	x-n
ip→alu2	
+1→alu2	
edb→di	
alu2→ip	
Li2	Li3

di→a→RPL/SPL	IR
ip→ad→ao	x-n
ip→alu2	
+1→alu2	
edb→irf	
alu2→ip	
Li3	mrr2

6) LDA D16

ip→ad→ao	DR
ip→alu2	x-n
+1→alu2	
edb→di	
alu2→ip	
2.1	2.2

di→a→taH	DR
ip→ad→ao	x-n
ip→alu2	
+1→alu2	
edb→di	
alu2→ip	
2.2	2.3

di→a→taL	IR
ip→ad→ao	x-n
ip→alu2	
+1→alu2	
alu2→ip	
edb→irf	
2.3	SB Ld4

ta→ad→ao	DR
edb→di	x-n
Ld4	mrm2

7) STA D16

ip→ad→ao	DR
ip→alu2	x-n
+1→alu2	
edb→di	
alu2→ip	
2.1	2.2

di→a→taH	DR
ip→ad→ao	x-n
ip→alu2	
+1→alu2	
edb→di	
alu2→ip	
2.2	2.3

di→a→taL	IF
ip→ad→ao	x-n
ip→alu2	
+1→alu2	
alu2→ip	
edb→irf	
2.3	SB ST4

ta→ad→ao	DW
A→a→do	x-n
irf→ire.	
ST4	ib

13) Jmp D16

ip→ad→ao	PR
ip→alu2	x-n
+1→alu2	
edb→di	
alu2→ip	
3.1	3.2

di→a→tAH	DR
ip→ad→ao	x-n
ip→alu2	
+1→alu2	
edb→di	
alu2→ip	
3.2	SB Jmp3

di→a→tAL	NA
	x-n
Jmp3	Jmp4

tA→ad→ip	NA
	x-n
Jmp4	Jmp5

ip→ad→ao	IF
ip→alu2	x-n
+1→alu2	
alu2→ip	
edb→irf	
Jmp5	mrr2

14) Jc D16

ip→ad→ao	DR
ip→alu2	x-n
+1→alu2	
edb→di	
alu2→ip	
4.1	4.2

di→a→taH	DR
ip→ad→ao	x-n
ip→alu2	
edb→di	
alu2→ip	
4.2	BC

C=1

di→a→tAL	NA
	x-n
Jc3	Jmp4

C=0

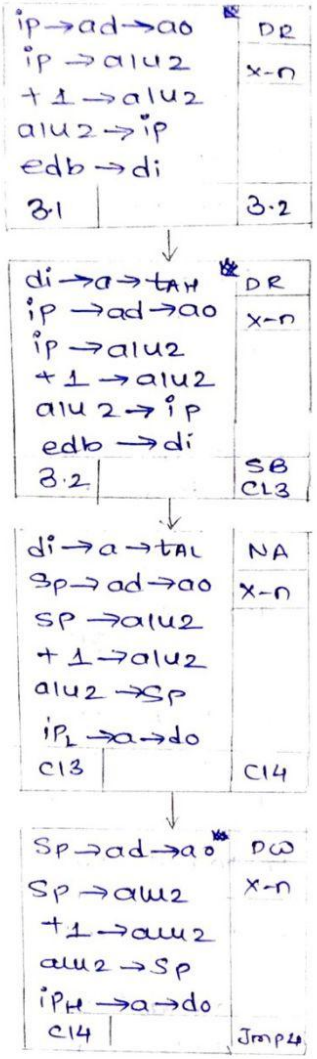
ip→ad→ao	IF
ip→alu2	x-n
+1→alu2	
alu2→ip	
edb→irf	
Jc4	mrr2

9) ACI D08

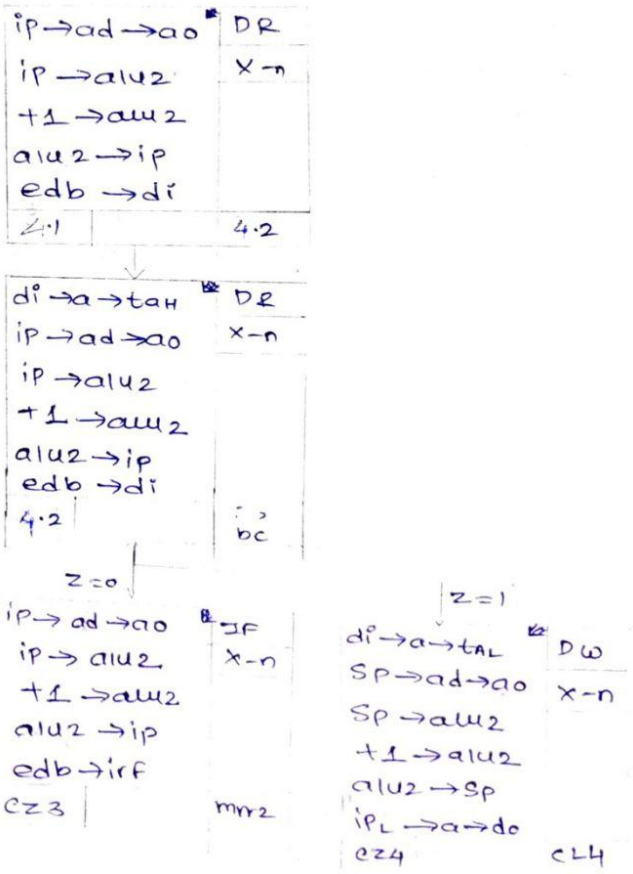
ip→ad→ao	DR
ip→alu2	x-n
+1→alu2	
edb→di	
alu2→ip	
1	SB Aci2

di→a→alu	IR
alu→A	Add-s
ip→ad→ao	
ip→alu2	
+1→alu2	
alu2→ip	
edb→irf	
Aci2	

15> Call D16



16> CZ D16



The address Select field can now be of ~~8~~ types as follows

1> NA – Next address or direct branch

2> IB – Instruction branch

3> SB-BC – sequence branch with branching

4> NA-BC – Next address with Branching

5> SB – Sequence branch

~~So the address select field will require 3 bits. T_2, T_1, T_0~~

The address select field is as below.

1> NA – Next address

2> SB – Sequence branch

3> BC – Branch control with address taken from SB

4> IB – Instruction Branch.

Now let us list all the tasks that need to be done to different components.

1) ip

ip → ad
ip → alu2
alu2 → ip
ad → ip

The first 3 tasks always occur together so we have in total 3 tasks (including none)

b \ a	0	1
0	none	ip→ad, ip→alu2, alu2→ip
1	ad→ip	X

Not included

∴ ip→ad, ip→alu2 ⇒ a
alu2→ip
ad→ip ⇒ b

2) ao

ad → ao
none
∴ ad→ao ⇒ a. ①

3) edb

edb → irf
edb → di
none

b \ a	0	1
0	none	edb→irf
1	edb→di	X

edb → irf ⇒ a
edb → di ⇒ b ②

4) irf

irf → ire
none

∴ irf → ire ⇒ a ①

5) HL

HL → ad
none
HL → ad ⇒ a

①

6) di

di → a.
none
∴ di → a ⇒ a ①

7) tA

ta → ad
none
ta → ad ⇒ a ①

8) taH

a → taH
none
a → taH ⇒ a ①

9) taL

a → taL
none
a → taL ⇒ a ①

10) A

A → a
none
A → a = a ①

11) t1

a → t1
t1 → a
none

b \ a	0	1
0	none	a→t1
1	t1→a	X

a → t1 ⇒ a
t1 → a ⇒ b ②

12) do

a → do
none
a → do ⇒ a ①

13) iPH

iPH → a
a → iPH
none.

b \ a	0	1
0	none	iPH→a
1	a→iPH	X

iPH → a ⇒ a
a → iPH ⇒ b ②

14) iPL

iPL → a
a → iPL
none

∴ iPL → a ⇒ a
a → iPL ⇒ b ①

15) Non Specific Reg

Ry → a; a → Rx
a → Rx
Rx → a.
none

b \ a	0	1
0	none	Ry→a
1	Rx→a	X

a → Rx ⇒ ab
Ry → a, a → Rx ⇒ a b̄
Rx → a ⇒ ā b ②

16) Register pair

a → RPH ⇒ a
a → RPL ⇒ b ②

17) ALU

a → alu; alu → A
a one.
∴ a → alu; alu → A ⇒ a ①

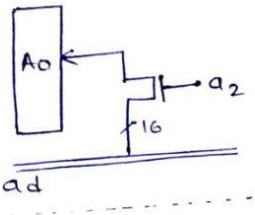
18) alu2

SP → ad; SP → alu2; +1 → alu2; alu2 → SP ⇒ a b̄ ⇒ a b
SP → ad; SP → alu2, −1 → alu2, alu2 → SP ⇒ ā b ⇒ ā b
ip → ad, ip → alu2, +1 → alu2, alu2 → ip ⇒ ā b̄ ⇒ a b
none ⇒ ā b̄ ②

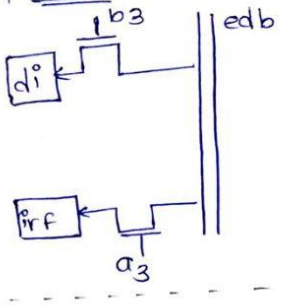
The above combination is better than the combination 1. So we dont need the control word bits for ip separate.

The connections of the different components to their peripherals is shown below,

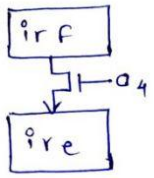
2) ao



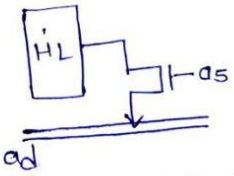
3) edb



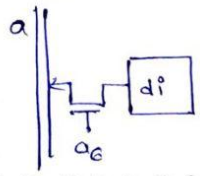
4) irf



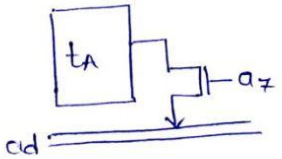
5) HL



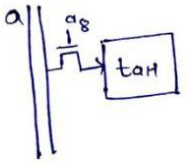
6) di



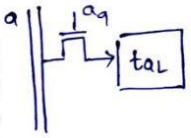
7) tA



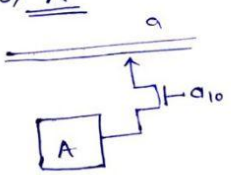
8) taH



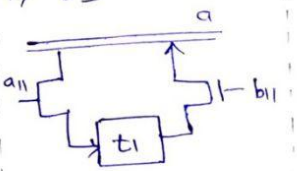
9) taL



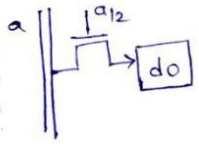
10) A



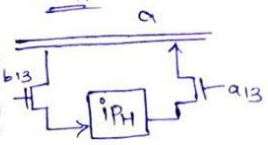
11) t1



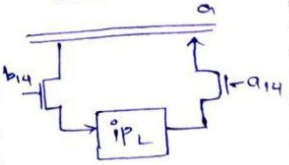
12) do



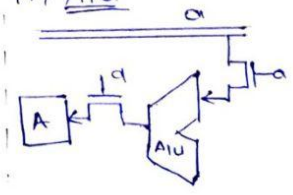
13) iPH



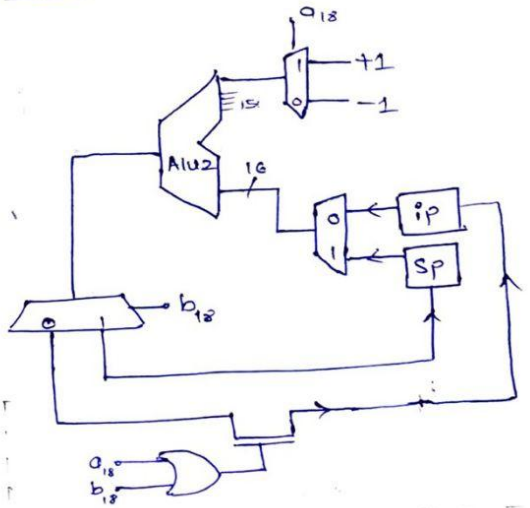
14) iPL

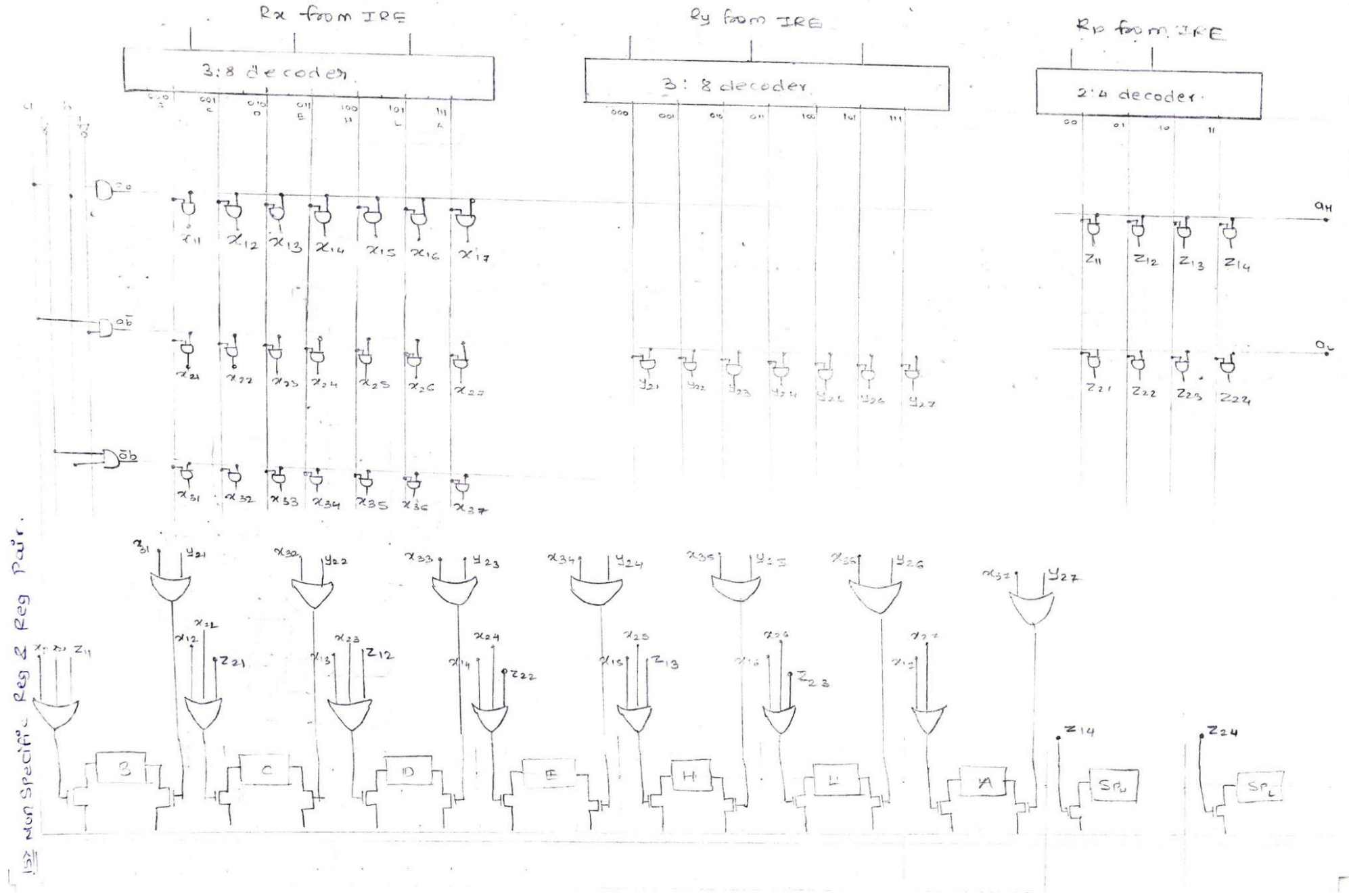


17) AIU

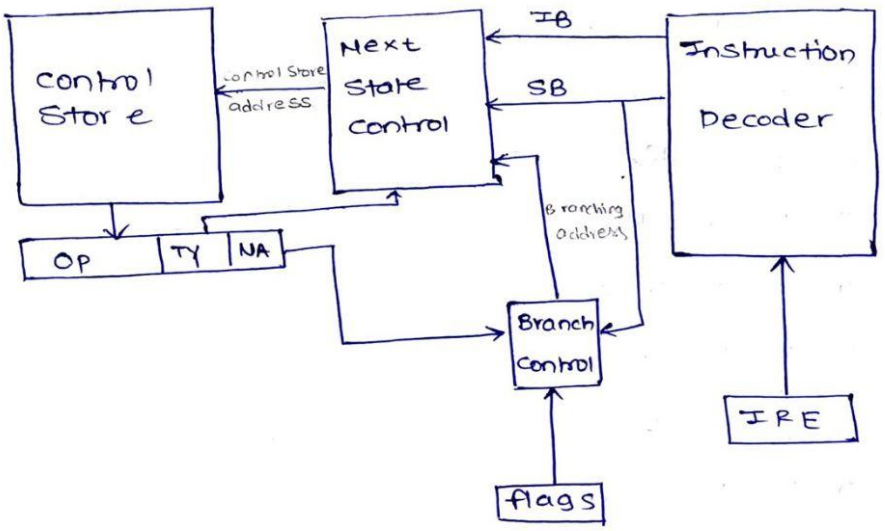


18) ALU2




Rx from IRE
3:8 decoder
Ry from IRE
3:8 decoder
Rp from IRE
2:4 decoder
aH
aL
bH
bL
Reg & Reg Pair.
B
C
D
E
H
L
A
SPH
SPL
z14
z24

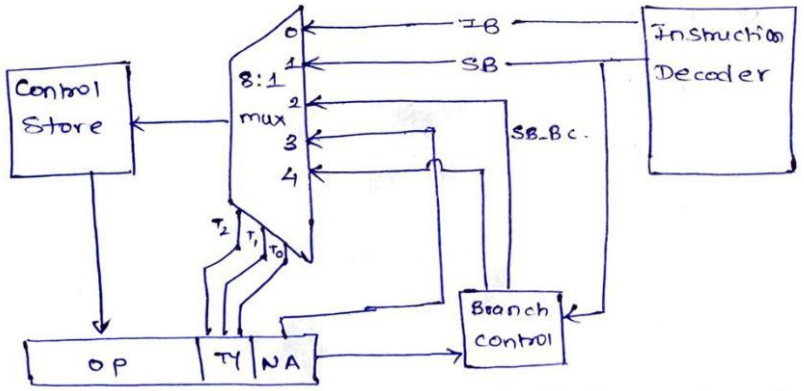
The block diagram of the microcontroller is shown below,



The TY field is of 3 bits. The assignment is as shown below

000 – IB
001 – SB
010 – SB-Bc
011 – NA
100 – NA-Bc

The Next State control block is as shocon below:-



Let us find how many total instructions can be made.

Instruction	combination
1> mov Rg, Rg	49
2> mov Rg, M	7
3> mov M, Rg	7
4> mvI Rg, D08	7
5> LXI SP/Rp	4
6> LDA D16	1
7> STA D16	1
8> ADC Rg	7
9> ACI D08	1
10> SSB Rg	7
11> ANA Rg	7

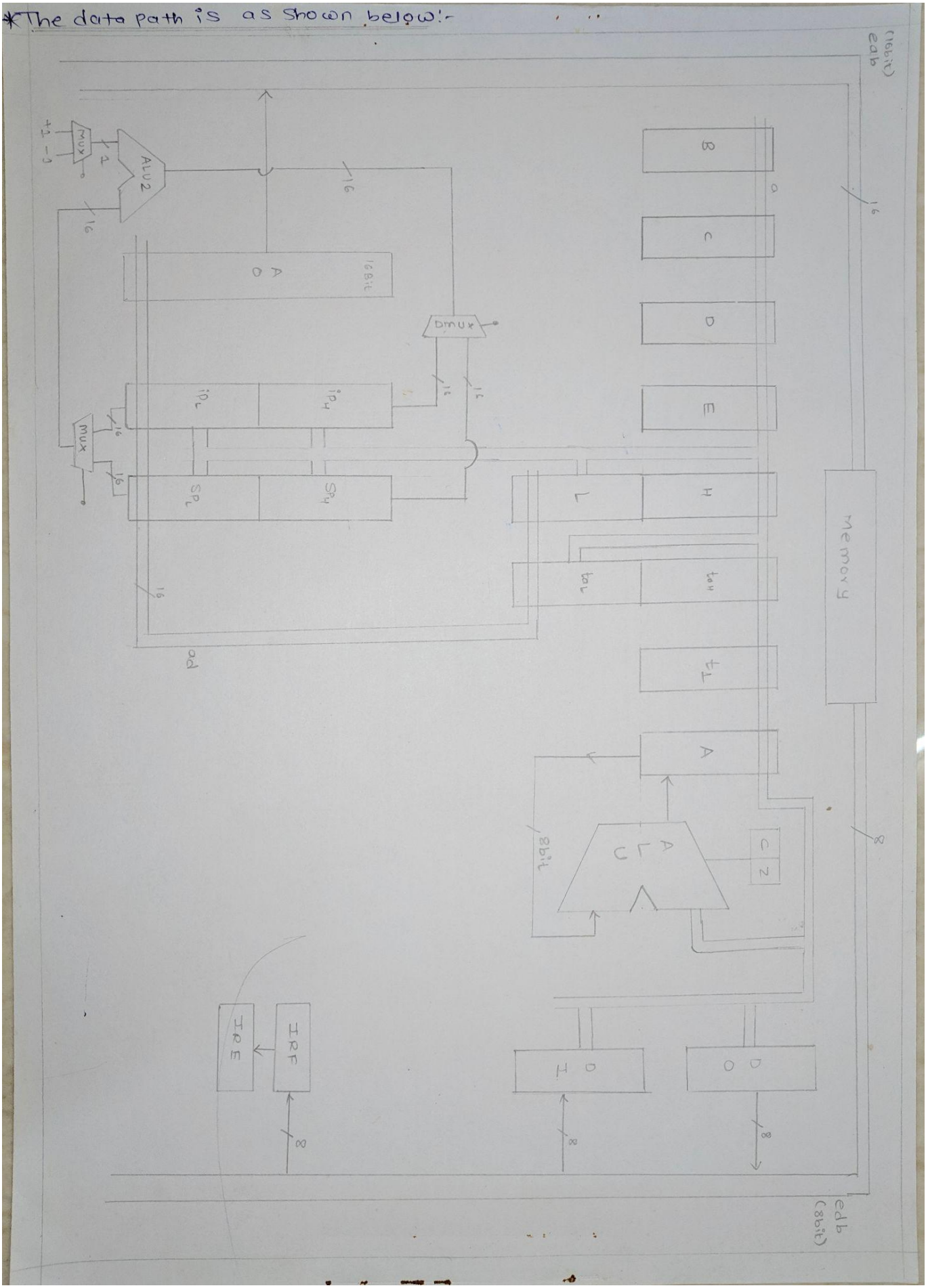
Instruction	combination
12> cmp Rg	7
13> JmP D16	1
14> Jc D16	1
15> call D16	1
16> cz D16	1
17> Ret	1
18> Rz	1

∴ Total number of unique instructions = 111

So we can represent all the instructions using 7 bits.

The assignment of the bits to the instructions is as shown below:-

1>	MOV Rg,Rg :-	0		Rx		Ry		
2>	MOV Rg,M	0		Rx		1	1	0
3>	MOV M, Rg	0	1	1	0	Rx		
4>	MVI Rg D08	1	0	0	0	Rx		
5>	LXI Sp/Rp	1	1	0	1	0	Rp	
6>	LDA D16	1	1	0	1	1	0	0
7>	STA D16	1	1	0	1	1	0	1
8>	ADC Rg	1	0	0	1	Rg		
9>	ACI D08	1	1	0	1	1	1	0
10>	SBB Rg	1	0	1	0	Rg		
11>	ANA Rg	1	0	1	1	Rg		
12>	cmp Rg	1	1	0	0	Rg		
13>	Jmp D16	1	1	0	1	1	1	0
14>	Jc D16	1	1	0	1	1	1	1
15>	Call D16	1	1	1	0	0	0	0
16>	CZ D16	1	1	1	0	0	0	1
17>	Ret	1	1	1	0	0	1	0
18>	RZ	1	1	1	0	0	1	1



The control store signals were written in a spreadsheet as there were so many of them. They can be found here: 8085 control store