VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

School of Computer Science and Engineering

Microprocessors and Microcontrollers

WINTER SEMESTER 2024-25

Digital Assignment - 1

NAME- Joy Sen

REG No.- 22BCB0060

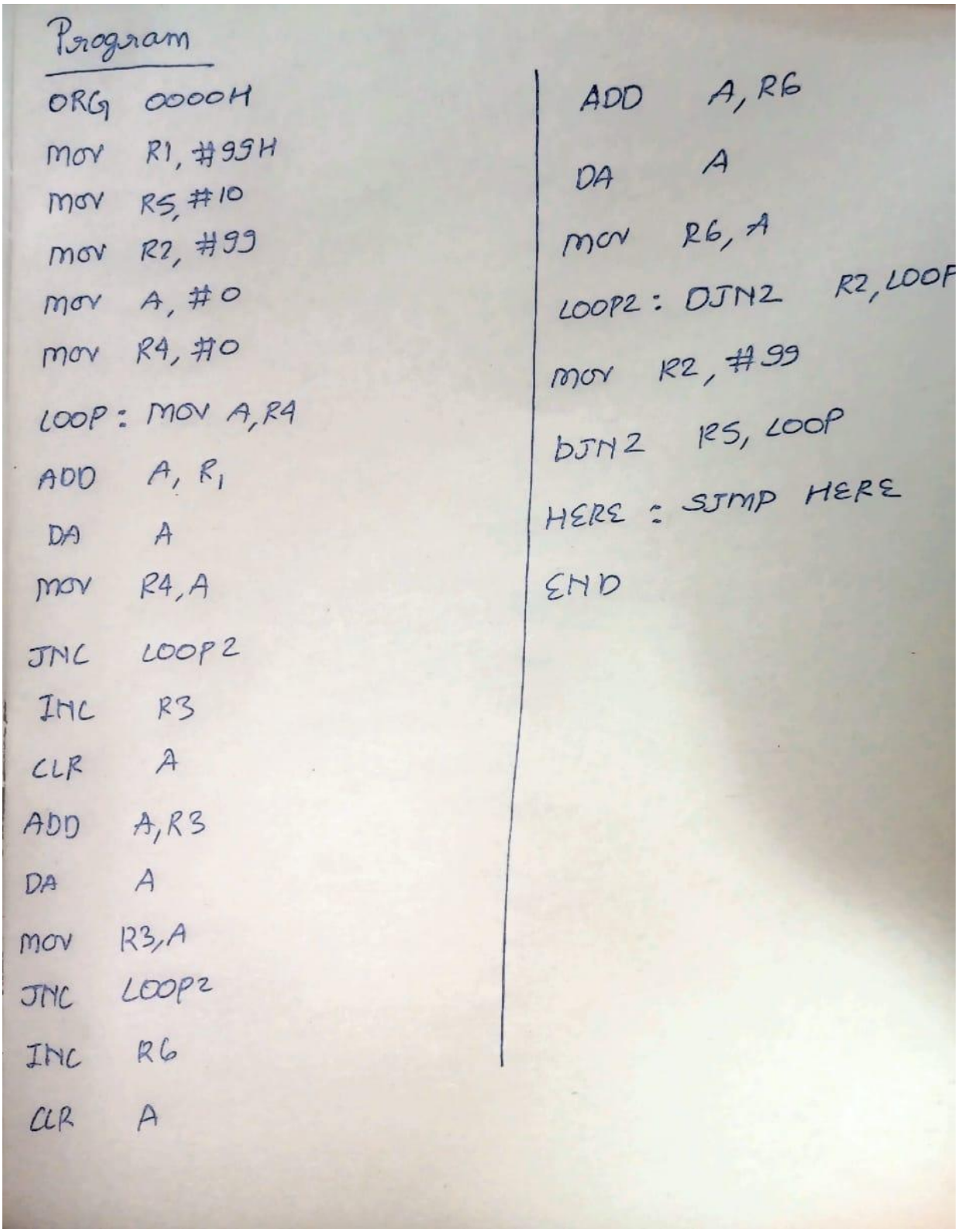
Faculty- Nisha JS

Course- Microprocessors and Microcontrollers Lab

Course code: BECE204P

1. Write an assembly language program for multiplication of two 8-bit BCD numbers using the repeated addition logic. (NOTE: Without using MUL AB instruction)

Code -



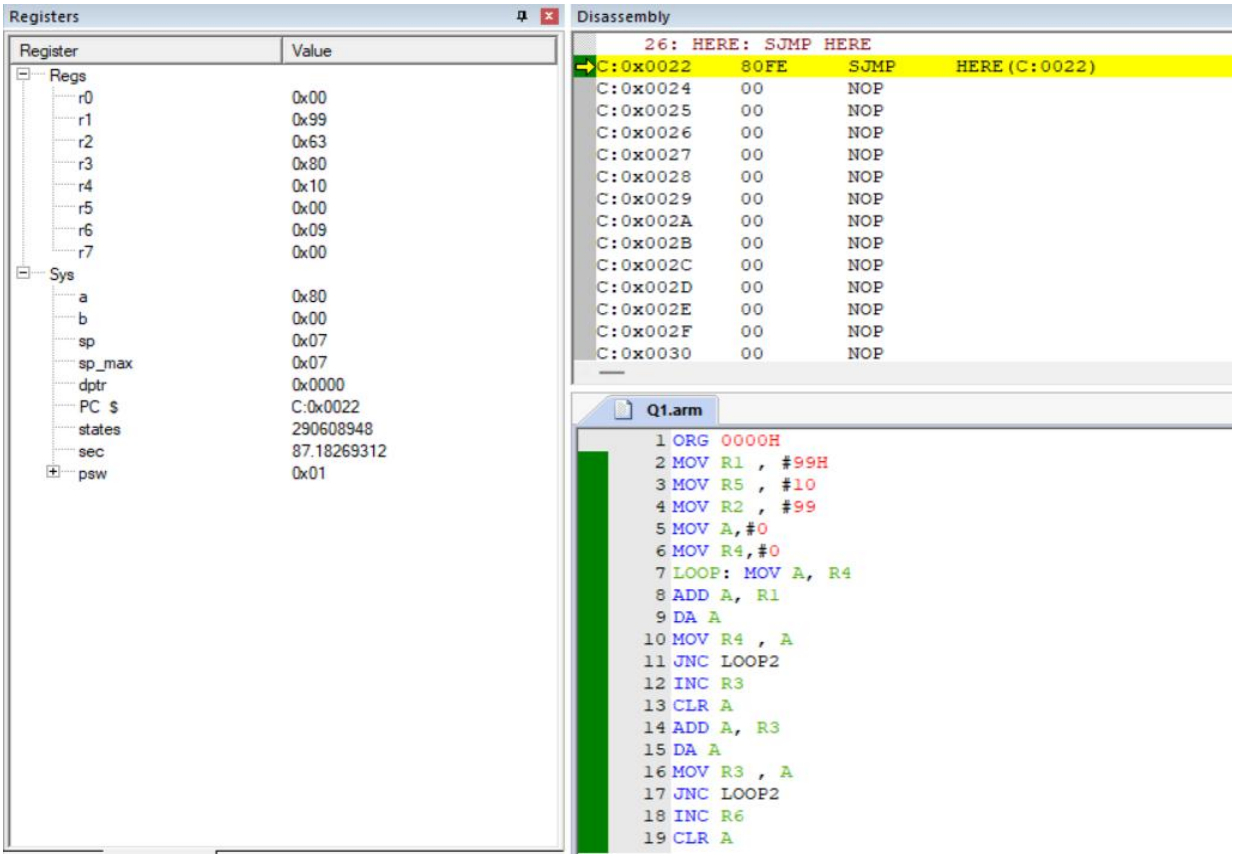
```
Program

ORG   0000H
MOV   R1, #99H
MOV   R5, #10
MOV   R2, #99
MOV   A, #0
MOV   R4, #0
LOOP: MOV A,R4
ADD   A, R1
DA    A
MOV   R4,A
JNC   LOOP2
INC   R3
CLR   A
ADD   A,R3
DA    A
MOV   R3,A
JNC   LOOP2
INC   R6
CLR   A
ADD   A, R6
DA    A
MOV   R6, A
LOOP2: DJNZ   R2,LOOP
MOV   R2, #99
DJNZ  R5, LOOP
HERE: SJMP HERE
END
```

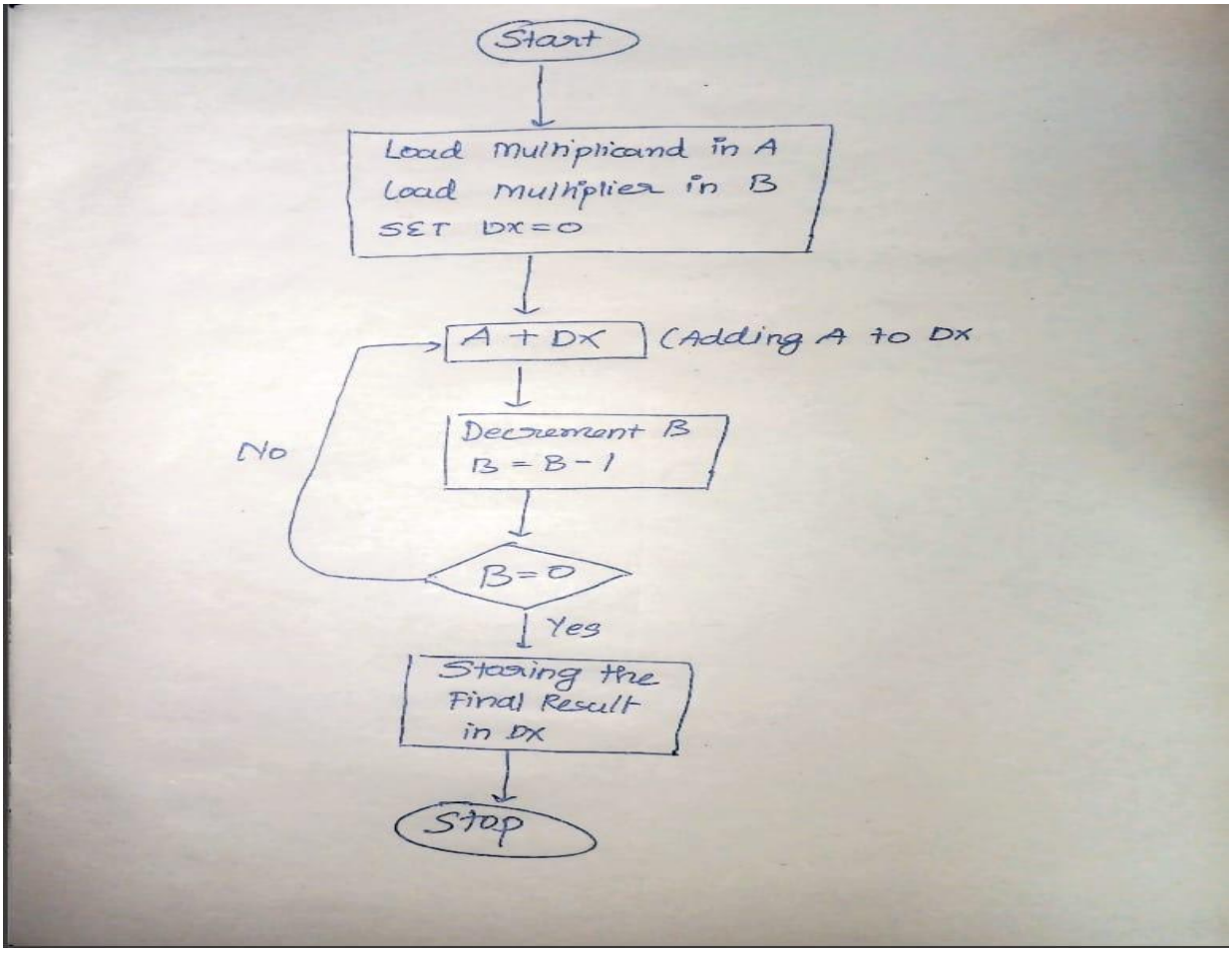
Typed Code -

```
ORG 0000H
MOV R1 , #99H
MOV R5 , #10
MOV R2 , #99
MOV A,#0
MOV R4,#0
LOOP: MOV A, R4
ADD A, R1
DA A
MOV R4 , A
JNC LOOP2
INC R3
CLR A
ADD A, R3
DA A
MOV R3 , A
JNC LOOP2
INC R6
CLR A
ADD A, R6
DA A
MOV R6 , A
LOOP2: DJNZ R2 , LOOP
MOV R2,#99
DJNZ R5 , LOOP
HERE: SJMP HERE
END
```

Output -



Flowchart -



Algorithm -

Algorithm -

1) Initialize Registers:
   - Load the multiplicand into the register R1.
   - Load the multiplier into the register R5.
   - Set the accumulator (R4) to zero to store the partial sum.
   - Initialize a higher byte (R_3 or R_6) to handle any carry generated during the addition.

2) Loop for Repeated addition:
   - Use a loop to repeatedly add the value in R1
   - Decrement R_5 (multiplier) by 1.

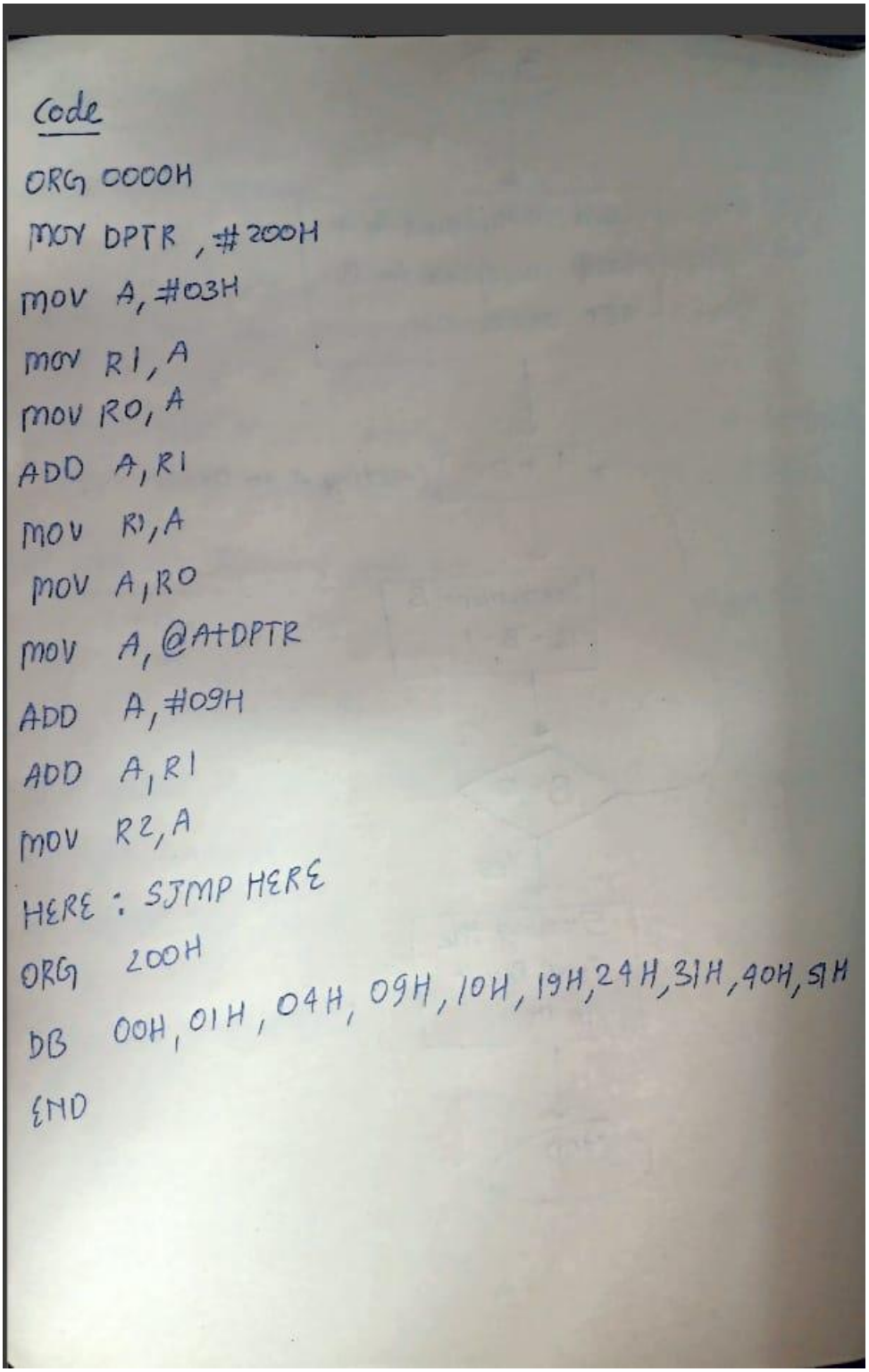
3) Handle the carry by incrementing the higher byte register R_3 or R_6

4) Adjust the result to BCD format

5) Store the final result.

2. Write a program to calculate y where y = x2 + 2x + 9. x is between 0 and 9 and the lookup table for x2 is located at the address (code space) of 200H. Register R0 has the x, and at the end of the program R2 should have y. Use the simulator to change the x value and single-step through the program, examining the registers as you go.

Code -

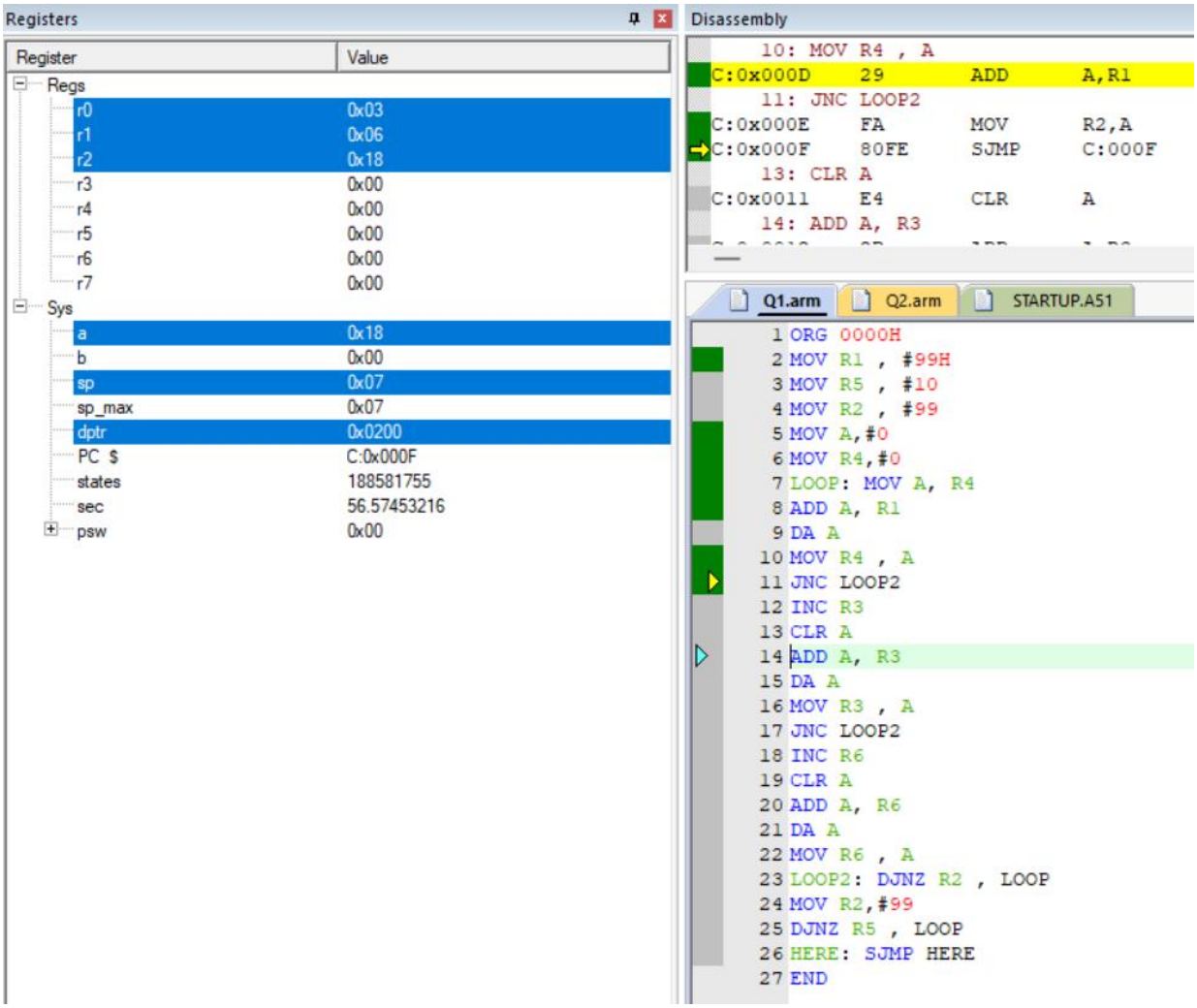


```
Code
ORG 0000H
MOV DPTR, #200H
MOV A, #03H
MOV R1, A
MOV R0, A
ADD A, R1
MOV R1, A
MOV A, R0
MOV A, @A+DPTR
ADD A, #09H
ADD A, R1
MOV R2, A
HERE: SJMP HERE
ORG 200H
DB 00H, 01H, 04H, 09H, 10H, 19H, 24H, 31H, 40H, 51H
END
```

Typed Code -

```
ORG 000H
MOV DPTR, #200H
MOV A, #03H
MOV R1, A
MOV R0,A
ADD A, R1
MOV R1, A
MOV A, R0
MOVC A, @A+DPTR
ADD A,#09H
ADD A, R1
MOV R2, A
HERE: SJMP HERE
ORG 200H
DB 00H,01H,04H,09H,10H,19H,24H,31H,40H,51H
END
```

Output -


Registers
Register
Value
Regs
r0 0x03
r1 0x06
r2 0x18
r3 0x00
r4 0x00
r5 0x00
r6 0x00
r7 0x00
Sys
a 0x18
b 0x00
sp 0x07
sp_max 0x07
dptr 0x0200
PC $ C:0x000F
states 188581755
sec 56.57453216
psw 0x00
Disassembly
10: MOV R4 , A
C:0x000D 29 ADD A,R1
11: JNC LOOP2
C:0x000E FA MOV R2,A
C:0x000F 80FE SJMP C:000F
13: CLR A
C:0x0011 E4 CLR A
14: ADD A, R3
Q1.arm
Q2.arm
STARTUP.A51
1 ORG 0000H
2 MOV R1 , #99H
3 MOV R5 , #10
4 MOV R2 , #99
5 MOV A,#0
6 MOV R4,#0
7 LOOP: MOV A, R4
8 ADD A, R1
9 DA A
10 MOV R4 , A
11 JNC LOOP2
12 INC R3
13 CLR A
14 ADD A, R3
15 DA A
16 MOV R3 , A
17 JNC LOOP2
18 INC R6
19 CLR A
20 ADD A, R6
21 DA A
22 MOV R6 , A
23 LOOP2: DJNZ R2 , LOOP
24 MOV R2,#99
25 DJNZ R5 , LOOP
26 HERE: SJMP HERE
27 END

Algorithm -

Algorithm -

1) Initial DPTR: Set the Data Pointer (DPTR) to the point to the lookup table containing x^2 values located at memory address 200H

2) Load x into register: Load the value of x into the accumulator (A) and copy it to registers R0 and R1

3) Calculate 2x: Compute 2x by adding x to itself and storing in R1.

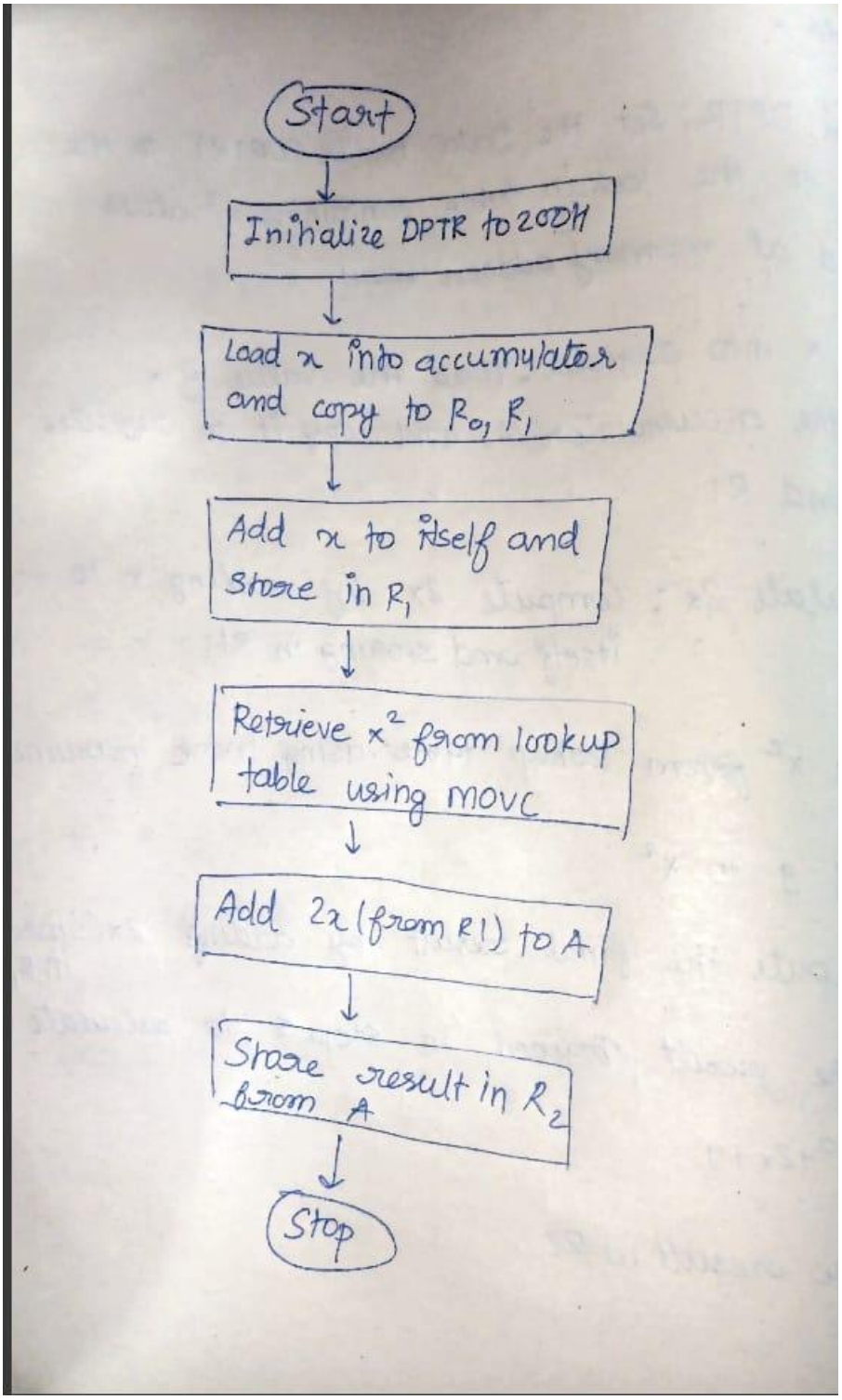
4) Fetch x^2 from lookup table using MOVC instruction

5) Add 9 to x^2

6) Compute the final result by adding 2x (stored in R_1) to the result obtained in step 5 to calculate $y = x^2 + 2x + 9$

7) Store result in R2.

Flowchart -


Start
Initialize DPTR to 200H
Load x into accumulator and copy to R0, R1
Add x to itself and store in R1
Retrieve x^2 from lookup table using MOVC
Add 2x (from R1) to A
Store result in R2 from A
Stop