Sub: **Microprocessors and Microcontrollers Lab**

Lab Task 2

TASK2 EXPT 1:

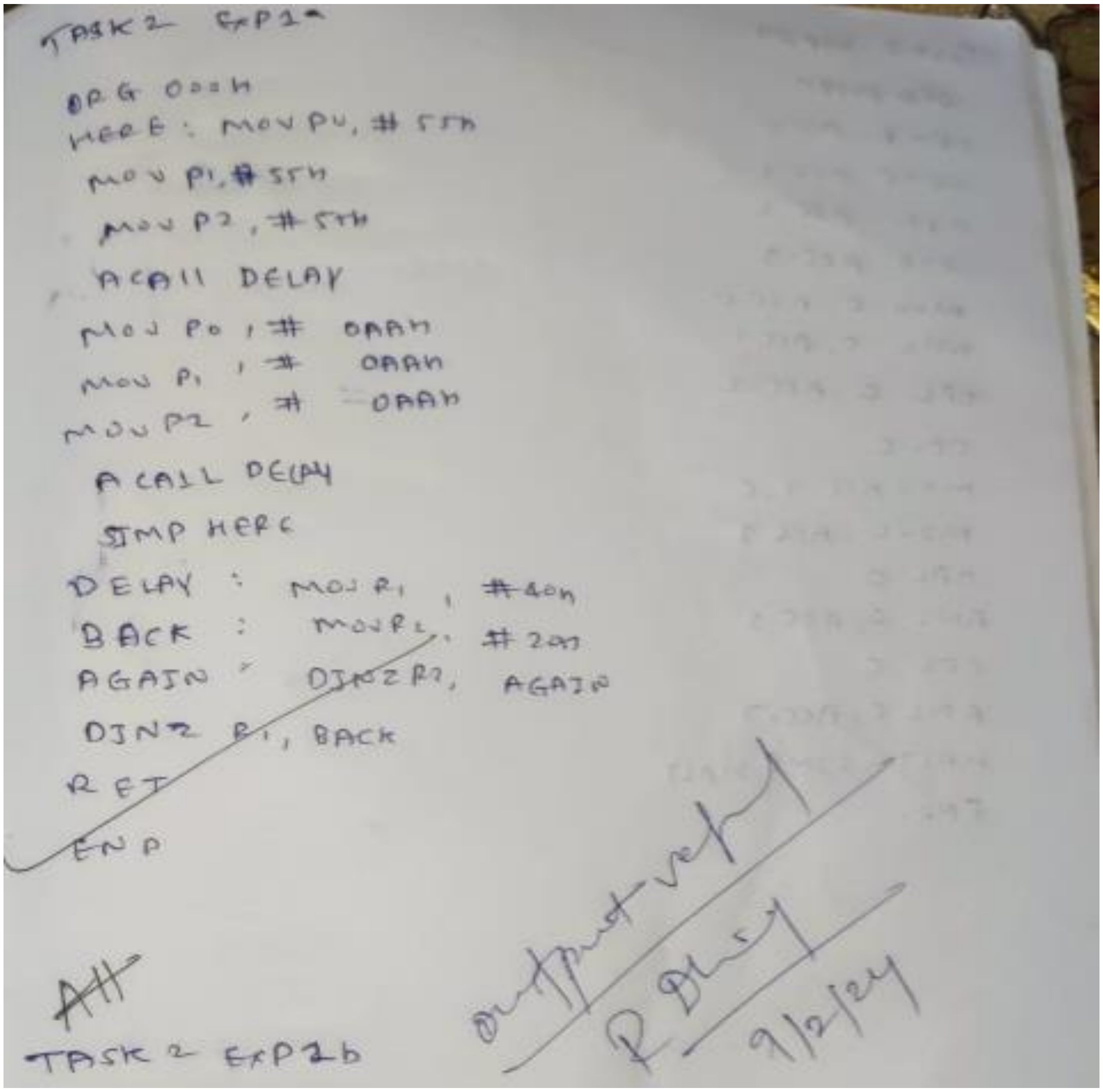
Write and assemble a program to toggle all the bits of PO, P1, and P2 continuously by sending 55H and AAH to these ports.

Put a time delay between the "on" and "off" states. Then using the simulator, single-step through the program and examine the ports. Do not single-step through the time delay call.

CODE:

```
ORG 0000H

HERE: MOV P0, #55H

MOV P1, #SSH

MOV P2,#55H

ACALL DELAY

MOV P0, #0AAH

MOV P1, #0AAH

MOV P2, #ΘΑΛΗ

ACALL DELAY

SJMP HERE

DELAY: MOV R1, #04H

BACK: MOV R2, #20H
```

```
AGAIN: DJNZ R2, AGAIN
DINZ R1, BACK
RET
END
```



TASK 2 EXP 1a

```
ORG 000H
HERE : MOV P0, # 55H
 MOV P1, # 55H
 MOV P2 , # 55H
 ACALL DELAY
 MOV P0 , # 0AAH
 MOV P1 , # 0AAH
MOV P2 , # 0AAH
 ACALL DELAY
 SJMP HERE
DELAY : MOV R1 , # 40H
BACK : MOV R2 , # 20H
AGAIN : DJNZ R2, AGAIN
 DJNZ R1, BACK
RET
END
```

Att

TASK 2 EXP 1b

Output verified

R Bh

9/2/24

Output:

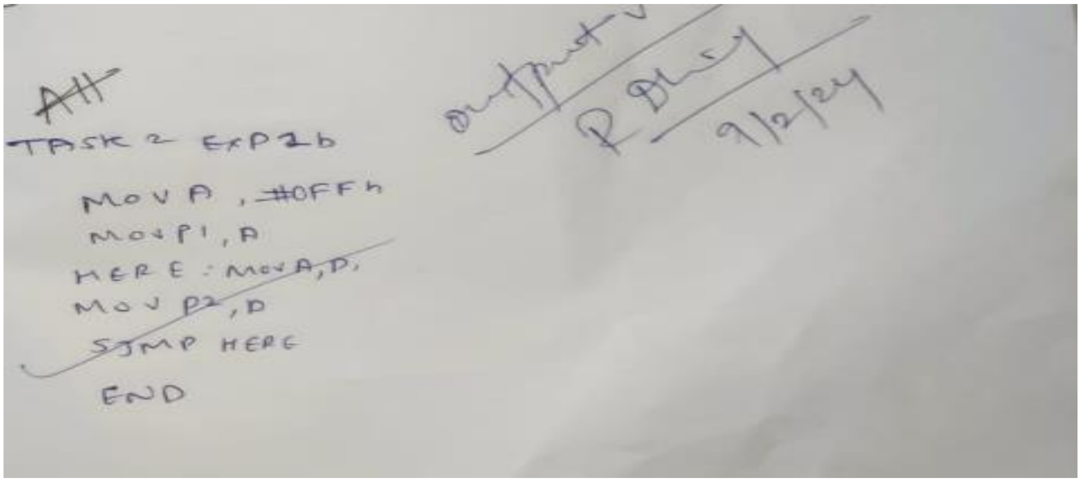
Result: The program successfully toggles all the bits of ports PO, P1, and P2 continuously by sending 55H and AAH signals, with a time delay between each state change. By using the simulator and single-stepping through the program, the ports can be examined to verify the toggling operation.

TASK2 EXPT 2:

Get the Data from Port P1 and Send it to Port P2, Note: P1 as input Port and P2 as Output Port

Code:

```
MOV A, #OFFh ;        A=FFH
MOV P1, A ;           make P1 an input port
HERE: MOV A, P1 ;     get data from P1
MOV P2, A ;           send it to P2
SJMP HERE;            keep doing this
END
```



A#

TASK 2 EXP 2b

```
MOV A , #OFFh
MOV P1 , A
HERE : MOV A, P1
MOV P2 , A
SJMP HERE
END
```

output ✓

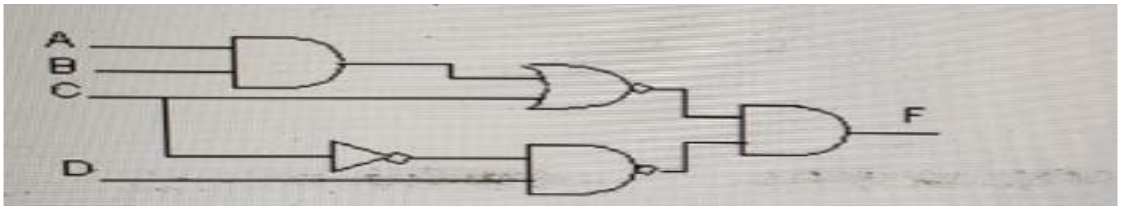
9/2/24

Output:

Result: The program reads data from input port P1 and then sends this data to output port P2.

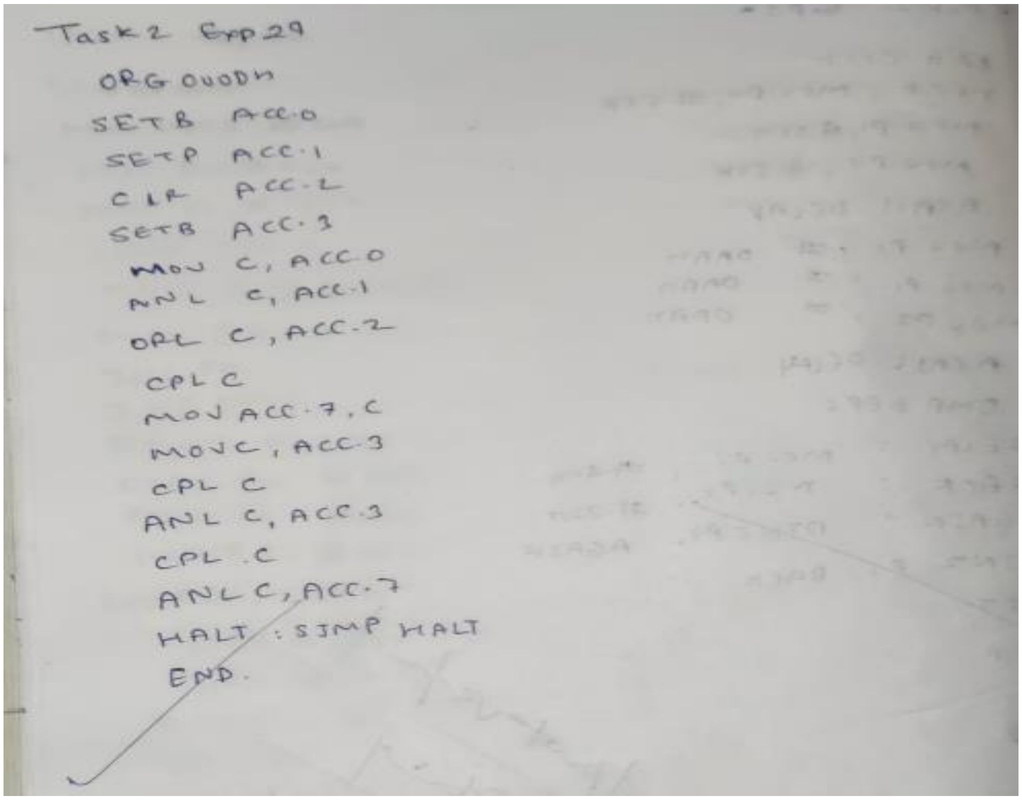
TASK2 EXPT 3:

To develop an assembly code for 8051 Microcontroller, to implement the given digital circuit using KEIL development tool.



Code:

```
ORG 0000H
SETB ACC.0; input A
SETB ACC.1; input B
CLR ACC.2; input C
SETB ACC.3; input D
MOV C, ACC.O
ANL C, ACC.1
ORL C, ACC.2
CPL C
MOV ACC.7, C
MOV C, ACC.2
CPL C
ANL C, ACC.3
CPL C
ANL C, ACC.7; Final output (F)
HALT: SJMP HALT
END
```



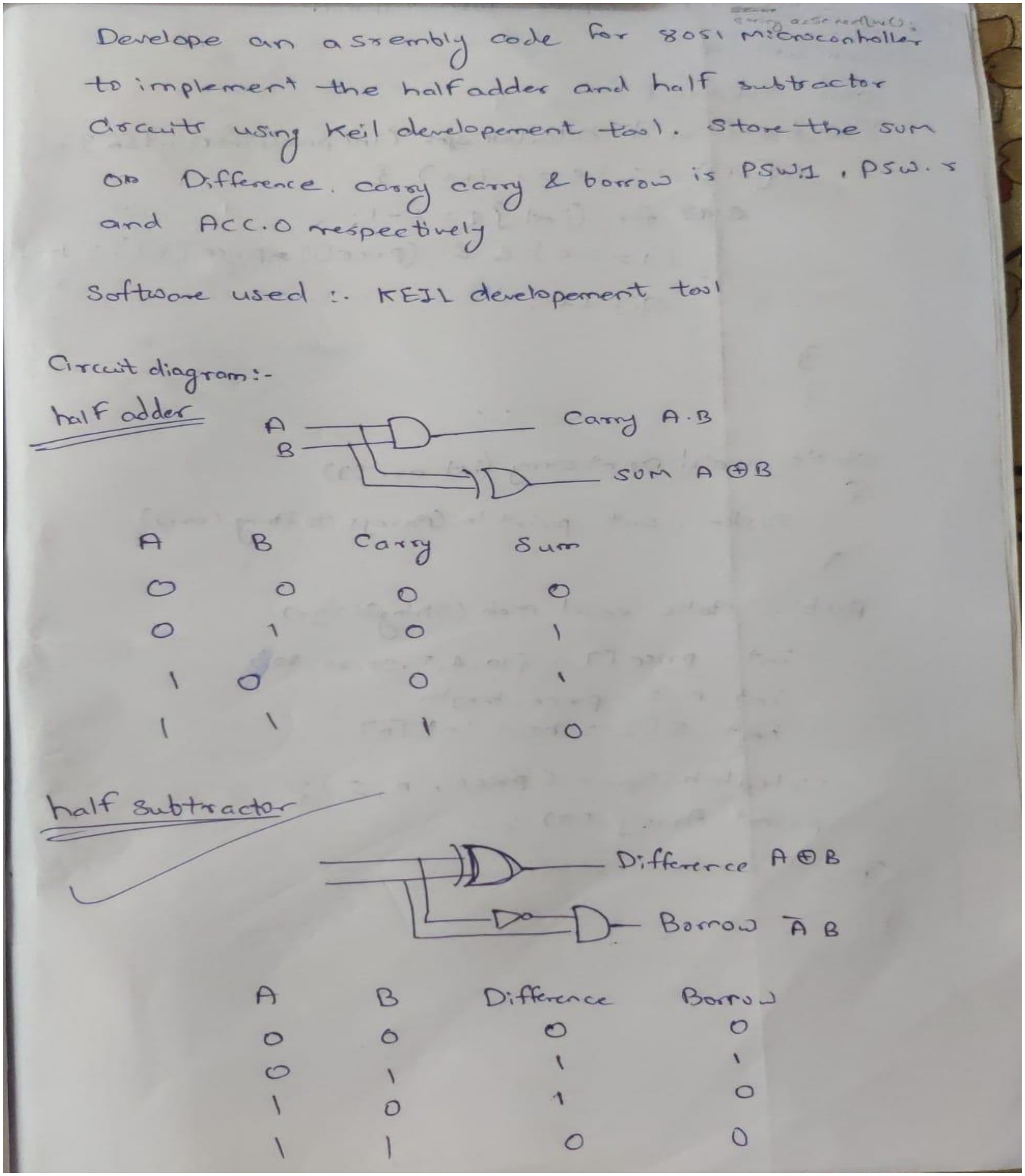
Task2 Exp 29

```
ORG 0000H
SETB ACC.0
SETP ACC.1
CLR  ACC.2
SETB ACC.3
 MOV C, ACC.0
 ANL  C, ACC.1
 ORL C, ACC.2
  CPL C
  MOV ACC.7, C
  MOVC, ACC.3
  CPL C
 ANL C, ACC.3
  CPL .C
  ANLC, ACC.7
  HALT :SJMP HALT
   END.
```

Result: with the input 1,1,0,1 we get output as 0

TASK2 EXPT 4:

Develope an assembly code for 8051 microcontroller to implement the half adder and half subtractor circuit using keil development tool. Store the sum or difference , Carry, borrow in PSW1, PSW5 and ACC.0 respectively:



string a=sc.nextLine();

Develope an assembly code for 8051 Microcontroller to implement the halfadder and half subtractor circuits using Keil developement tool. Store the sum On Difference. Carry carry & borrow is PSW.1 , PSW.5 and ACC.0 respectively

Software used :- KEIL developement tool

Circuit diagram:-

half adder

A

B

Carry A.B

SUM A ⊕B

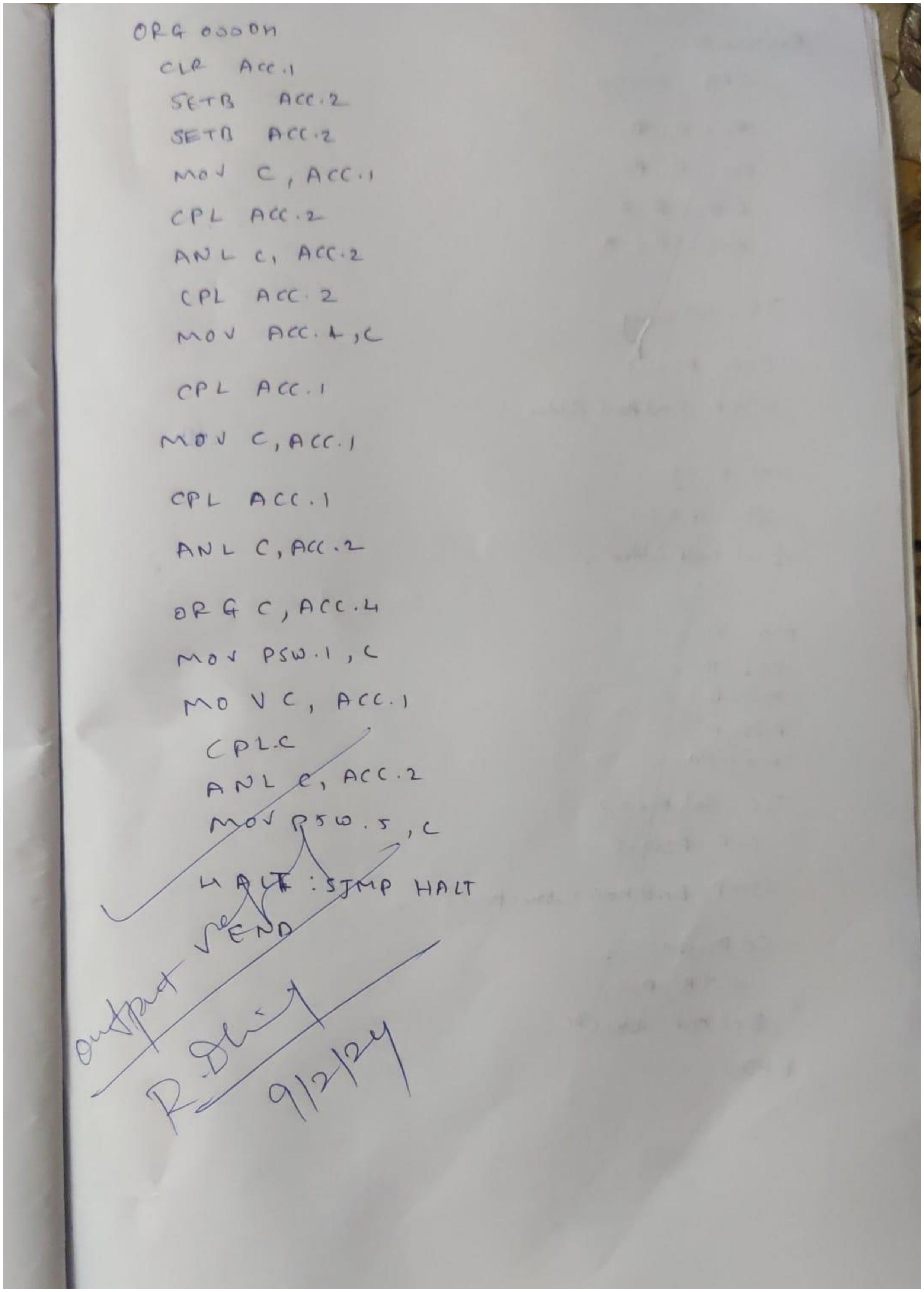
A	B	Carry	Sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

half subtractor

Difference A ⊕ B

Borrow $\bar{A}$ B

A	B	Difference	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0



```
ORG 0000H
 CLR ACC.1
  SETB  ACC.2
  SETB  ACC.2
  MOV  C, ACC.1
  CPL ACC.2
  ANL C, ACC.2
  CPL  ACC.2
  MOV  ACC.4,C

  CPL ACC.1

 MOV C,ACC.1

  CPL ACC.1

  ANL C,ACC.2

  ORG C,ACC.4
  MOV PSW.1 ,C
  MOV C, ACC.1
   CPL.C
   ANL C, ACC.2
   MOV PSW.5 ,C

   HALT :SJMP HALT
   END
```

Output ve

R. D

9/2/24

Result: The assembly code successfully implements both a half adder and half subtractor circuit on an 8051 microcontroller using the KEIL development tool. It stores the sum or difference, carry, and borrow in PSW1, PSW5, and ACC.0 respectively, enabling efficient computation and storage of arithmetic results.