



SCHOOL OF COMPUTER SCIENCE ENGINEERING

WINTER SEMESTER 2022-2023

LAB ASSIGNMENT - 1

Slot: L11 – L12

Class: VL2022230504038

Programme Name & Branch: B. Tech CSE

Course code & Title: BECE204P – Microprocessors and Microcontrollers Lab

Faculty Name: Venu Allapakam

Task 1: Arithmetic Operations – Addition and Subtraction

Program 1: Addition

Aim: To perform arithmetic addition- 8051 micro controllers

Software Requirement: Keil Software

Program:

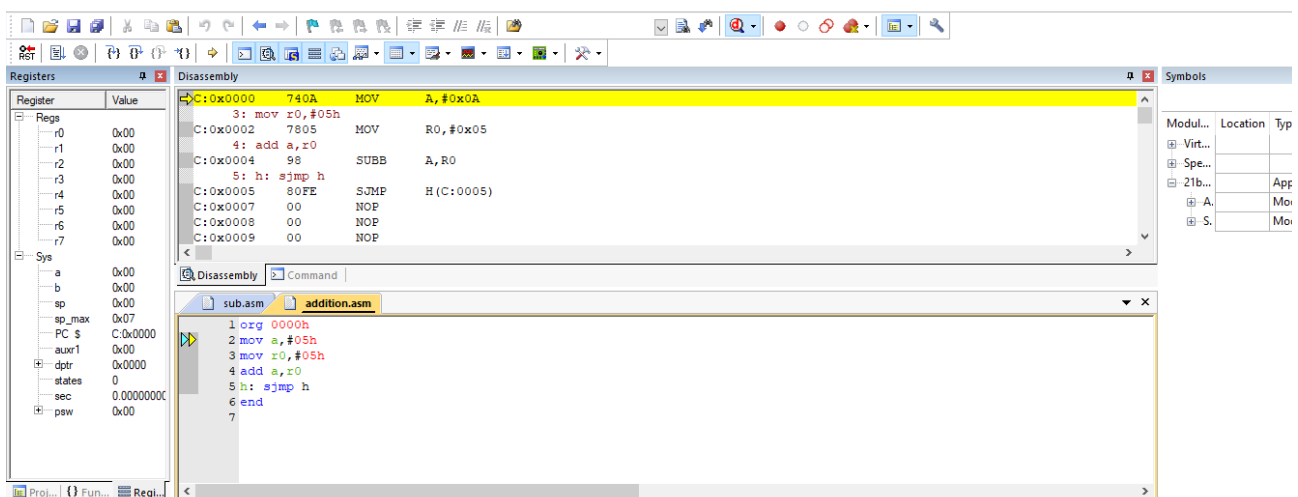
```

1  org 0000h
2  mov a,#05h
3  mov r0,#05h
4  add a,r0
5  h: sjmp h
6  end
7

```

Output:

Before Execution Register status:



After Execution:

The screenshot displays a microcontroller development environment with two main panels: 'Registers' and 'Disassembly'.

Registers Panel:

Register	Value
r0	0x05
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
Sys	
a	0x00
b	0x00
sp	0x07
sp_max	0x07
PC \$	C:0x0005
auxr1	0x00
dp1r	0x0000
states	3
sec	0.00000109
psw	0x00

Disassembly Panel:

Address	Hex	OpCode	Comment
C:0x0000	7405	MOV	A, #0x05
C:0x0002	7805	MOV	R0, #0x05
C:0x0004	98	SUBB	A, R0
4: LOOP: CLR A			
C:0x0005	80FE	SJMP	LOOP (C:0005)
6: ADD A, R2			
C:0x0007	FE	MOV	R6, A
7: JNC NEXT			
C:0x0008	7C65	MOV	R4, #0x65

Source Code Panel (ADD.asm):

```
1  ORG 0000H
2  MOV A, #05H
3  MOV R0, #05H
4  ADD A, R0
5  H: SJMP H
6  END
```

Result-

Result → On addition of 05, 05, we obtained 0x11 in accumulators.

(10)

T. S. L.

10/Jan/23

Program 2: Subtraction

Aim: To perform arithmetic subtraction- 8051 micro controllers

Software Requirement: Keil Software

Program:

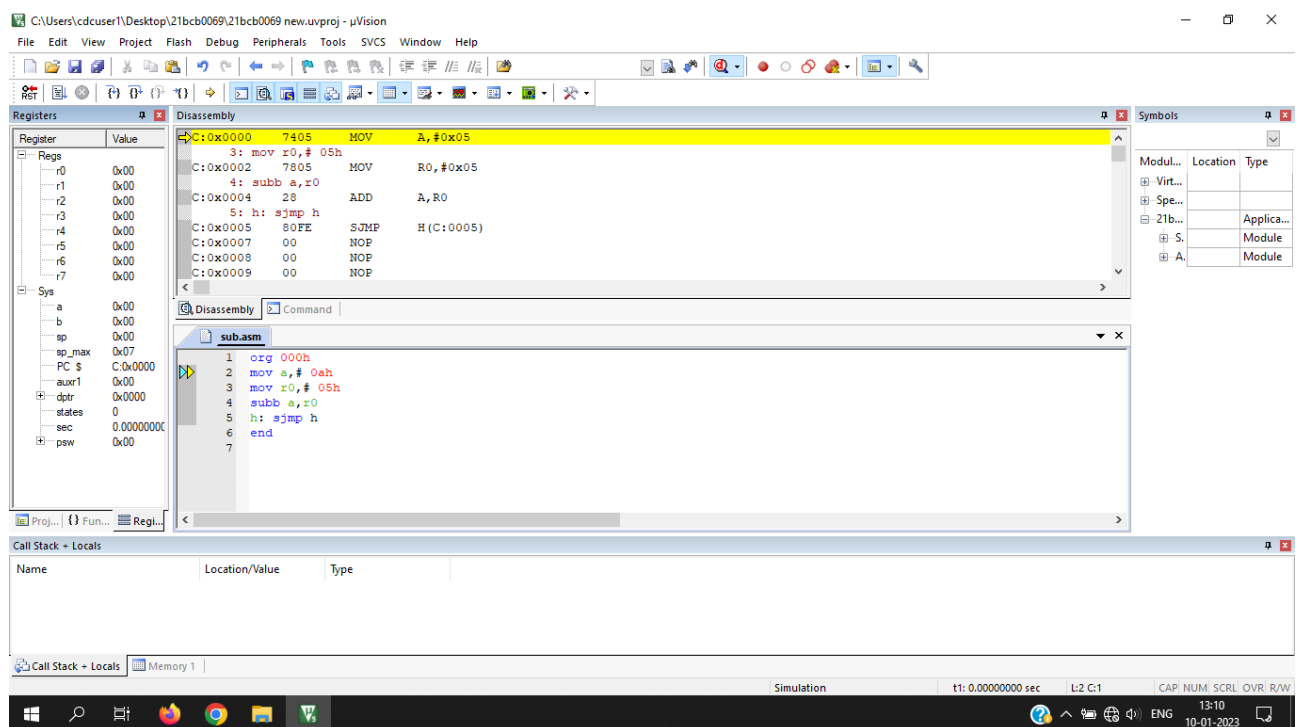
```

1  org 0000h
2  mov a,#0AH
3  mov r0,#05h
4  SUBB a,r0
5  h: sjmp h
6  end

```

Output:

Before Execution Register status:



After Execution:

Registers

Register	Value
Regs	
r0	0x00
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
Sys	
a	0x05
b	0x05
sp	0x07
sp_max	0x07
PC \$	C:0x0005
auxr1	0x00
dpnr	0x0000
states	3
sec	0.00000109
psw	0x00

Disassembly

```

C:0x0000 7405 MOV A,#0x05
C:0x0002 75F005 MOV B(0xF0),#0x05
4: LOOP: CLR A
→C:0x0005 A4 MUL AB
5: MOVC A,@A+DPTR
C:0x0006 80FE SJMP C:0006
7: JNC NEXT
C:0x0008 7C65 MOV R4,#0x65
8: INC R3

```

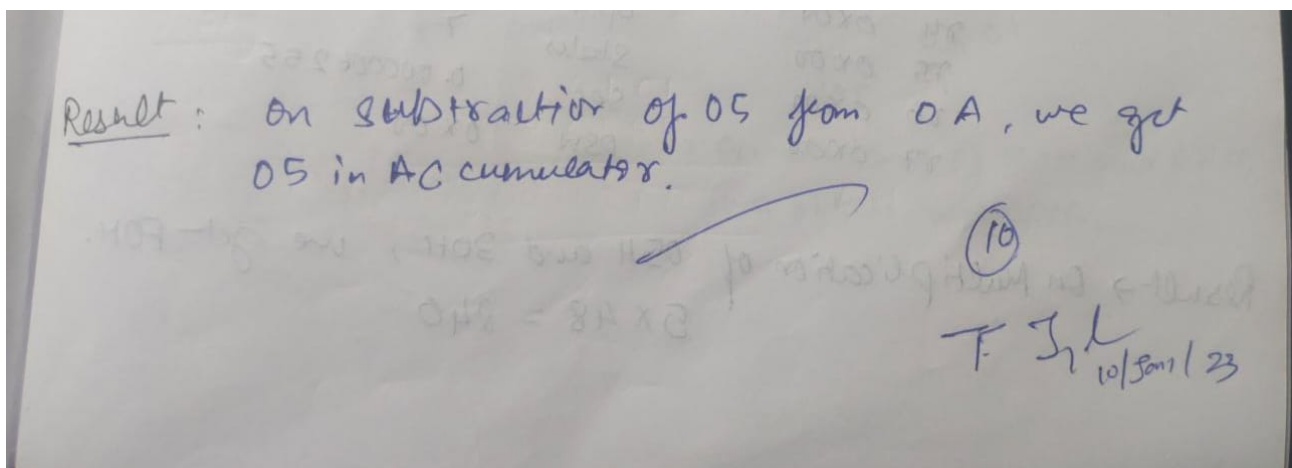
SUB.asm

```

1 ORG 0000H
2 MOV A,#05H
3 MOV R0,#05H
4 SUBB A,R0
5 H: SJMP H
6 END

```

Result—



Task 2: Arithmetic Operations – Multiplication and Division

Program 1: Multiplication

Aim: To perform arithmetic multiplication- 8051 micro controllers

Software Requirement: Keil Software

Program:

```

multi.asm
1  ORG 0000H
2  MOV A, #05H
3  MOV B, #30H
4  MUL AB
5  h: sjmp h
6  end

```

Output:

Before Execution Register status:

The screenshot displays the Keil IDE interface with two windows open: 'Registers' and 'Disassembly'.

Registers Window:

Register	Value
r0	0x00
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
a	0x00
b	0x00
sp	0x00
sp_max	0x07
PC	0x0000
auxr1	0x00
dptr	0x0000
states	0
sec	0.00000000
psw	0x00

Disassembly Window:

Address	Hex	OpCode	Comment
C:0x0000	7405	MOV	A, #0x05
C:0x0002	75F005	MOV	B(0xF0), #0x05
4: LOOP:		CLR	A
C:0x0005	84	DIV	AB
5:		MOVC	A, @A+DPTR
C:0x0006	80FE	SJMP	C:0006
7: JNC		NEXT	
C:0x0008	7C65	MOV	R4, #0x65
8:		INC	R3

The 'MUL.asm' source file is also visible, showing the assembly code:

```

1  ORG 0000H
2  MOV A, #05H
3  MOV B, #05H
4  MUL AB
5  H: SJMP H
6  END

```

After Execution:

The screenshot displays a microcontroller development environment with two main panels: 'Registers' and 'Disassembly'.

Registers Panel:

Register	Value
Regs	
r0	0x00
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
Sys	
a	0x01
b	0x00
sp	0x07
sp_max	0x07
PC \$	C:0x0006
auxr1	0x00
dpnr	0x0000
states	7
sec	0.00000255
psw	0x01

Disassembly Panel:

```

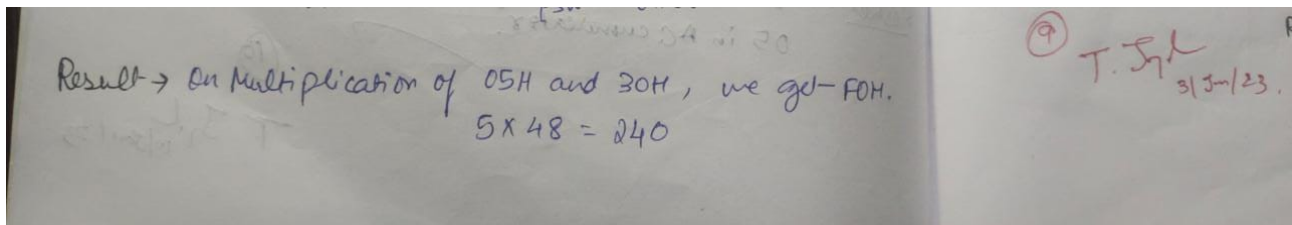
C:0x0000 7405 MOV A,#0x05
C:0x0002 75F005 MOV B(0xF0),#0x05
4: LOOP: CLR A
C:0x0005 84 DIV AB
5: MOVC A,@A+DPTR
C:0x0006 80FE SJMP C:0006
7: JNC NEXT
C:0x0008 7C65 MOV R4,#0x65
8: INC R3
  
```

MUL.asm File:

```

1 ORG 0000H
2 MOV A,#05H
3 MOV B,#05H
4 MUL AB
5 H: SJMP H
6 END
  
```

Result—



Program 2: Division

Aim: To perform arithmetic division- 8051 micro controllers

Software Requirement: Keil Software

Program:

```

1 ORG 0000H
2 MOV A, #50H
3 MOV B, #05H
4 div AB
5 h: sjmp h
6 end

```

Output:

Before Execution Register status:

The screenshot displays the Keil uVision IDE interface. On the left, the 'Registers' window shows the status of various registers before execution. On the right, the 'Disassembly' window shows the assembly code being executed.

Register	Value
r0	0x00
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
a	0x00
b	0x00
sp	0x00
sp_max	0x07
PC	0x0000
auxr1	0x00
dptr	0x0000
states	0
sec	0.00000000
psw	0x00

Address	Hex	Assembly
C:0x0000	7405	MOV A, #0x05
C:0x0002	75F005	MOV B(0xF0), #0x05
4: LOOP:	CLR A	
C:0x0005	84	DIV AB
5:	MOVC A, @A+DPTR	
C:0x0006	80FE	SJMP C:0006
7:	JNC NEXT	
C:0x0008	7C65	MOV R4, #0x65
8:	INC R3	

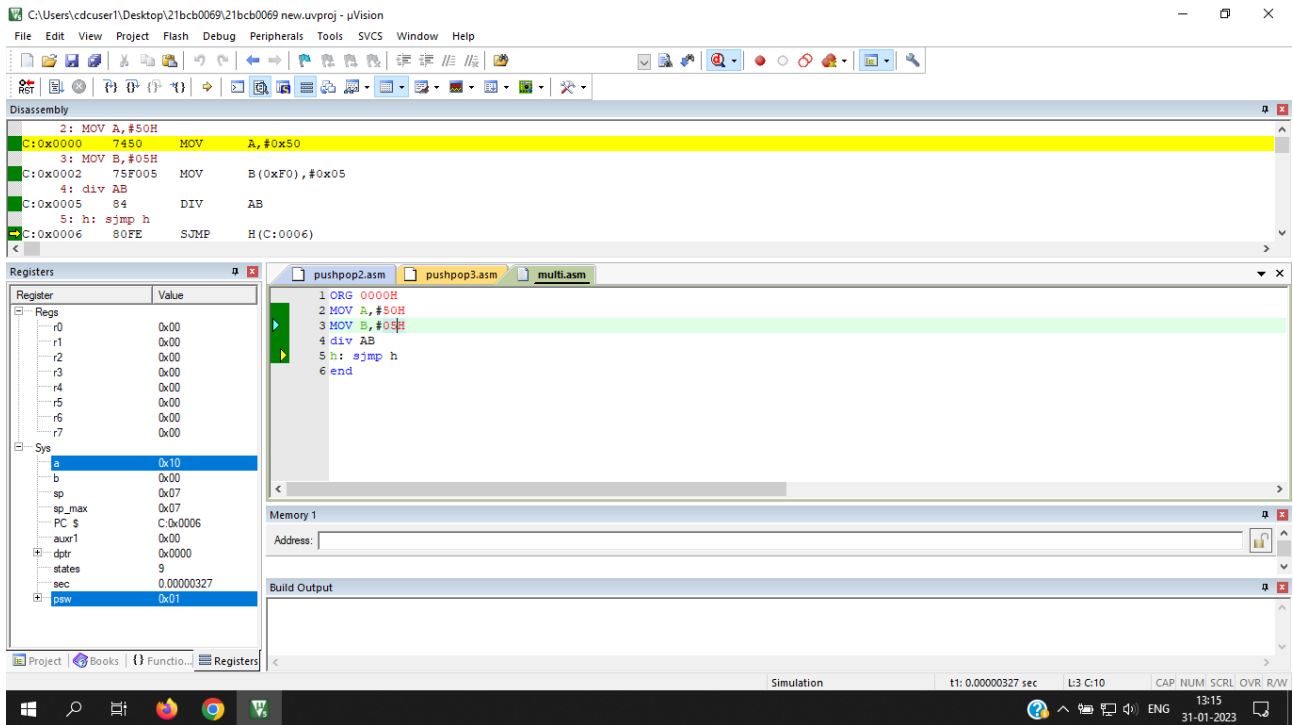
The assembly code in the 'Disassembly' window is as follows:

```

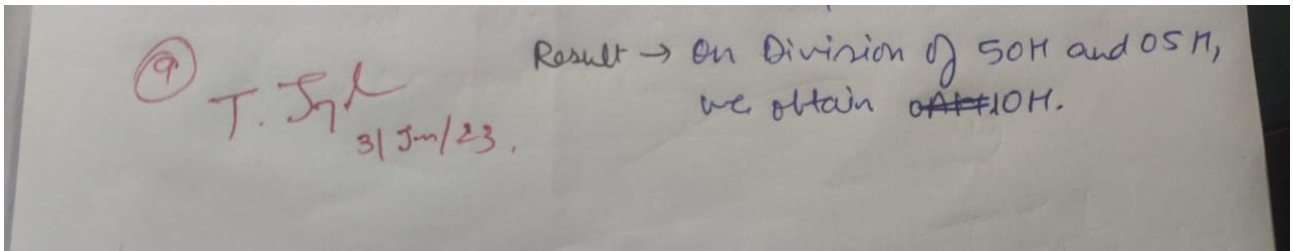
1 ORG 0000H
2 MOV A, #05H
3 MOV B, #05H
4 MUL AB
5 H: SJMP H
6 END

```


After Execution:



Result—



Task 3: Stack Operations - Push and Pop

Program 1: Push

Aim: To load values in each of the registers R0 to R4 and then push each of them into the stack

Software Requirement: Keil software

Program:

```

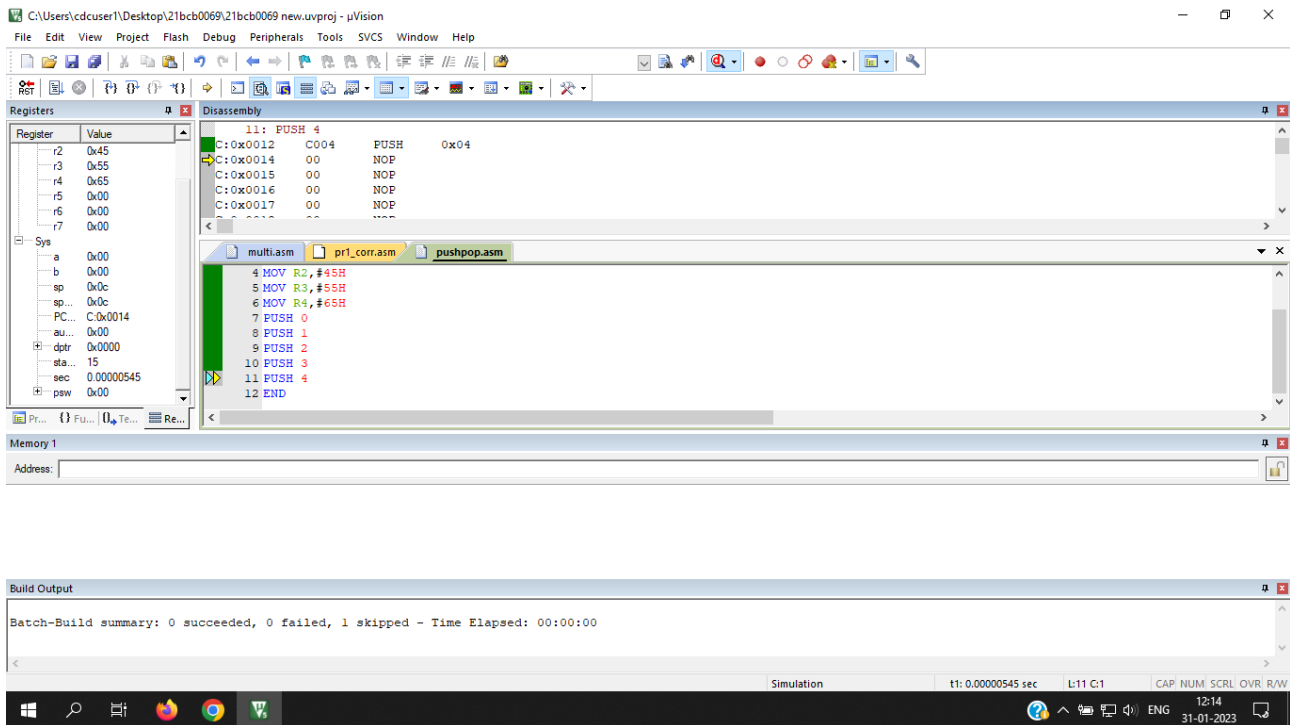
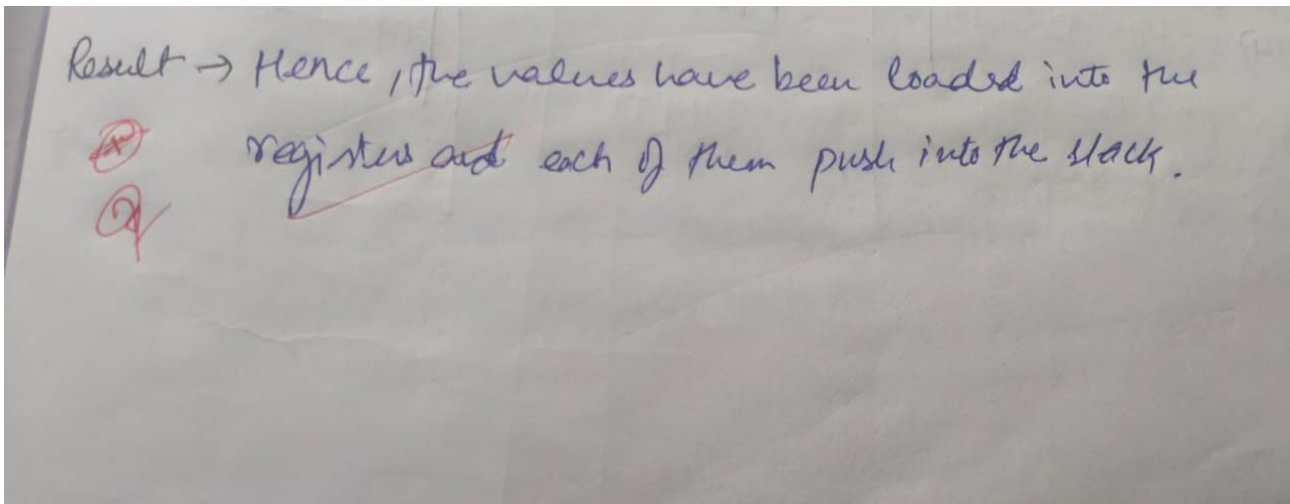
1  ORG 0000H
2  MOV R0, #25H
3  MOV R1, #35H
4  MOV R2, #45H
5  MOV R3, #55H
6  MOV R4, #65H
7  PUSH 0
8  PUSH 1
9  PUSH 2
10 PUSH 3
11 PUSH 4
12 END

```

Output:

Before Execution:

The screenshot displays the Keil uVision IDE interface before the program execution. On the left, the 'Registers' window shows the state of various registers: R0 through R7 are all 0x00; Sys, a, b, sp, and sp_max are 0x00; PC is 0x0000; auxr1 is 0x00; dptr is 0x0000; states is 0; sec is 0.00000000; and psw is 0x00. On the right, the 'Disassembly' window shows the assembly code for 'push pop 1.asm'. The first instruction, 'MOV R0, #0x25', is highlighted in yellow. Below it, the source code window shows the first few lines of the program: '1 ORG 0000H', '2 MOV R0, #25H', '3 MOV R1, #35H', '4 MOV R2, #45H', '5 MOV R3, #55H', '6 MOV R4, #65H', '7 PUSH 0', and '8 PUSH 1'.

After Execution:**Result:**

Program 2: Pop

Aim: To set SP to 0D and then put a different value in each of the RAM locations 08 to 0D and pop each location into registers R0 to R4

Software Requirement: Keil Software

Program:

```

1 ORG 0000H
2 MOV SP, #0DH
3 MOV 08H, #25H
4 MOV 09H, #35H
5 MOV 0AH, #45H
6 MOV 0BH, #55H
7 MOV 0CH, #65H
8 MOV 0DH, #75H
9 POP 0
10 POP 1
11 POP 2
12 POP 3
13 POP 4
14 END

```

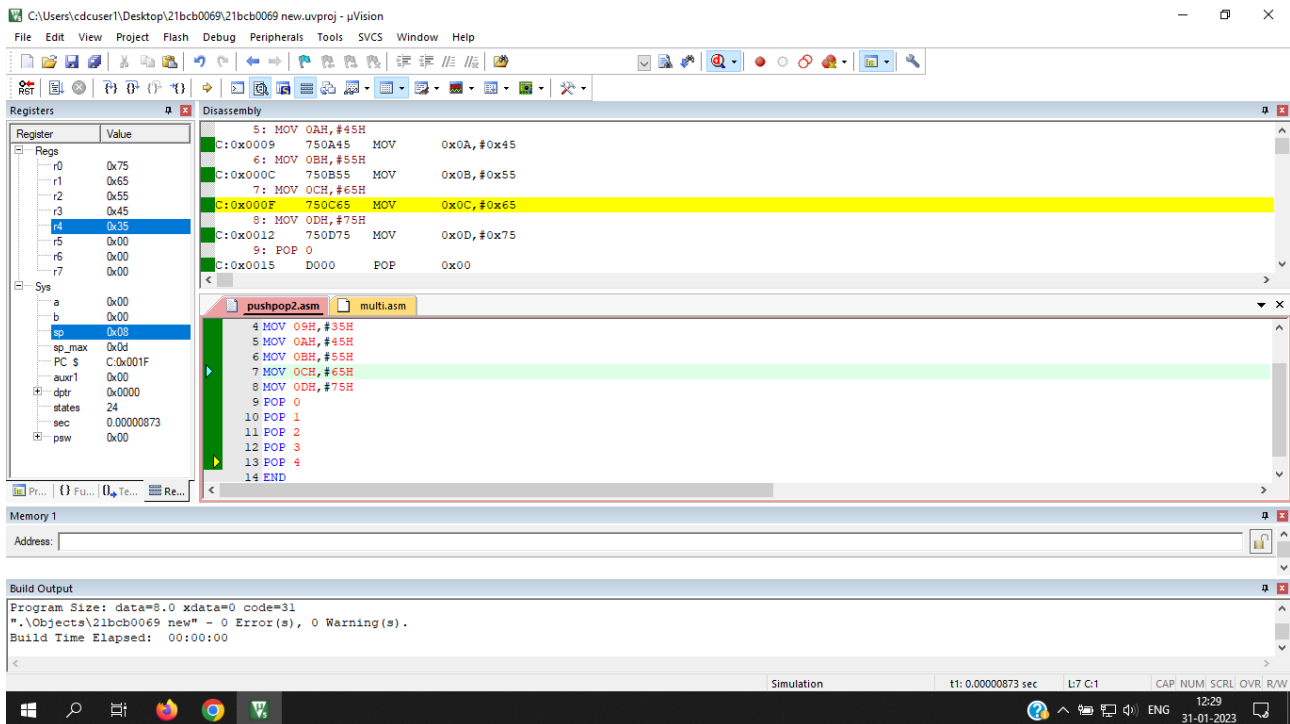
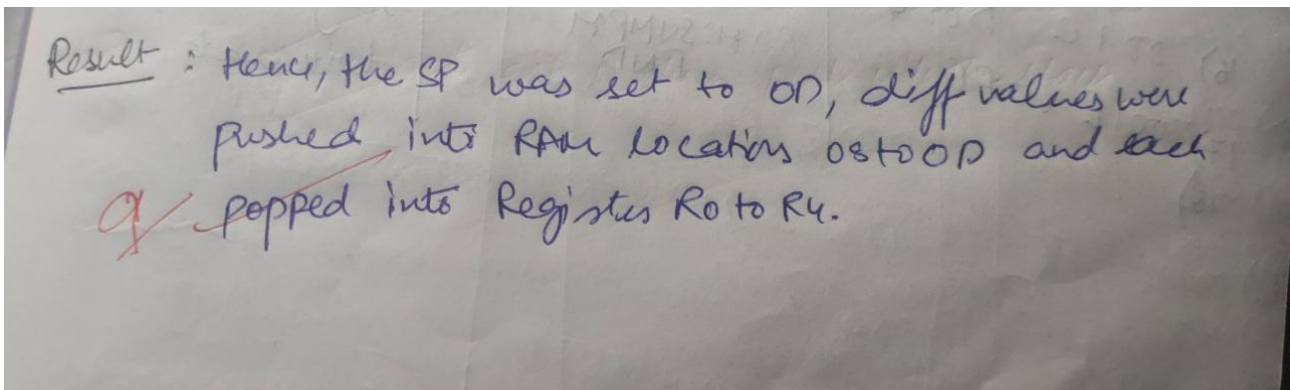
Output:

Before Execution:

Register	Value
r0	0x00
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
sp	0x00
sp_max	0x07
PC	0x0000
auxr1	0x00
dp1r	0x0000
states	0
sec	0.00000000
psw	0x00

Address	Hex	Assembly	Comment
C:0x0000	7825	MOV R0, #0x25	
3: MOV R1, #35H			
C:0x0002	7935	MOV R1, #0x35	
4: MOV R2, #45H			
C:0x0004	7A45	MOV R2, #0x45	
5: MOV R3, #55H			
C:0x0006	7B55	MOV R3, #0x55	
6: MOV R4, #65H			
C:0x0008	7C65	MOV R4, #0x65	
7: PUSH 0			
C:0x000A	C000	PUSH 0x00	
8: PUSH 1			
C:0x000C	C001	PUSH 0x01	
9: PUSH 2			

Address	Hex	Assembly	Comment
1 ORG 0000H			
2 MOV R0, #25H			
3 MOV R1, #35H			
4 MOV R2, #45H			
5 MOV R3, #55H			
6 MOV R4, #65H			
7 PUSH 0			
8 PUSH 1			

After Execution:**Result:**

Program 3: Push and Pop

Aim: To load values in each of the registers R0 to R4 and then push each of them into the stack and then pop them back

Software Requirement Keil Software

Program:

```

pushpop2.asm  pushpop3
1 org 0000h
2 mov r0, #25h
3 mov r1, #35h
4 mov r2, #45h
5 mov r3, #55h
6 mov r4, #65h
7 push 0
8 push 1
9 push 2
10 push 3
11 push 4
12 pop 0
13 pop 1
14 pop 2
15 pop 3
16 pop 4
17 end
18

```

Output:

Before Execution:

Registers

Register	Value
r0	0x00
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
Sys	
a	0x00
b	0x00
sp	0x00
sp_max	0x07
PC \$	C:0x0000
auxr1	0x00
dp1r	0x0000
states	0
sec	0.00000000
psw	0x00

Disassembly

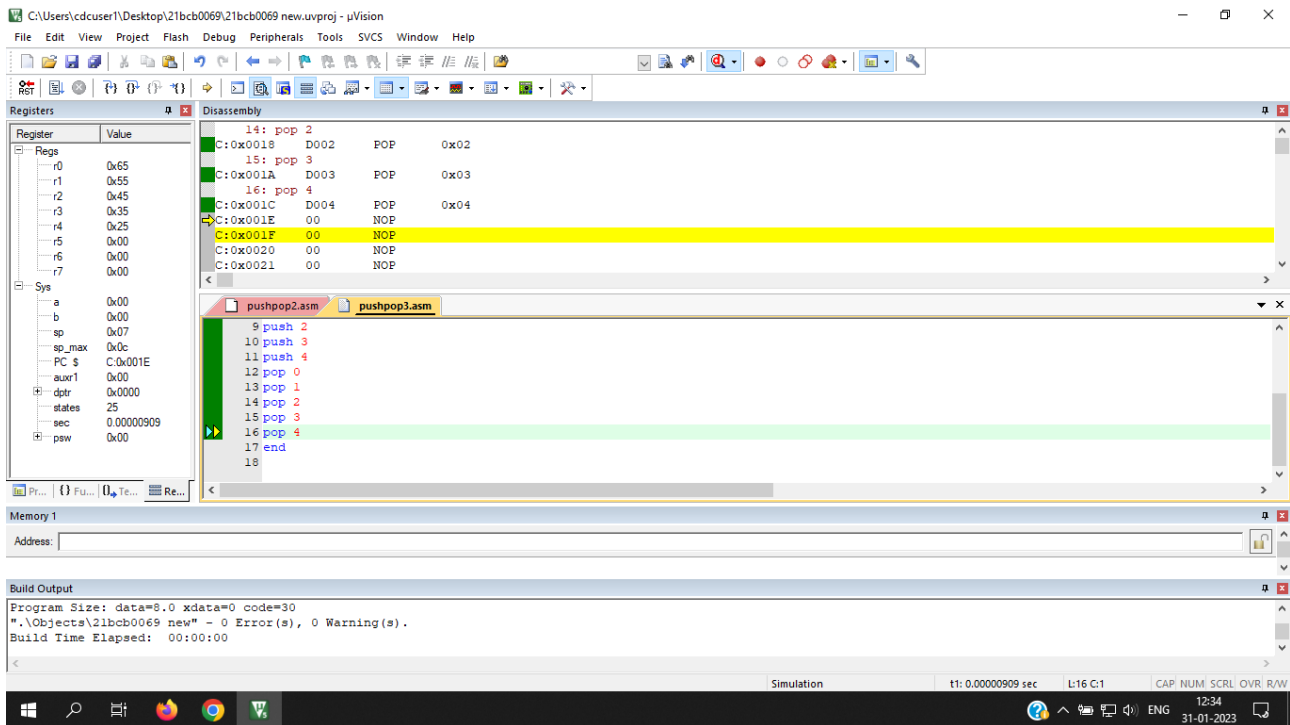
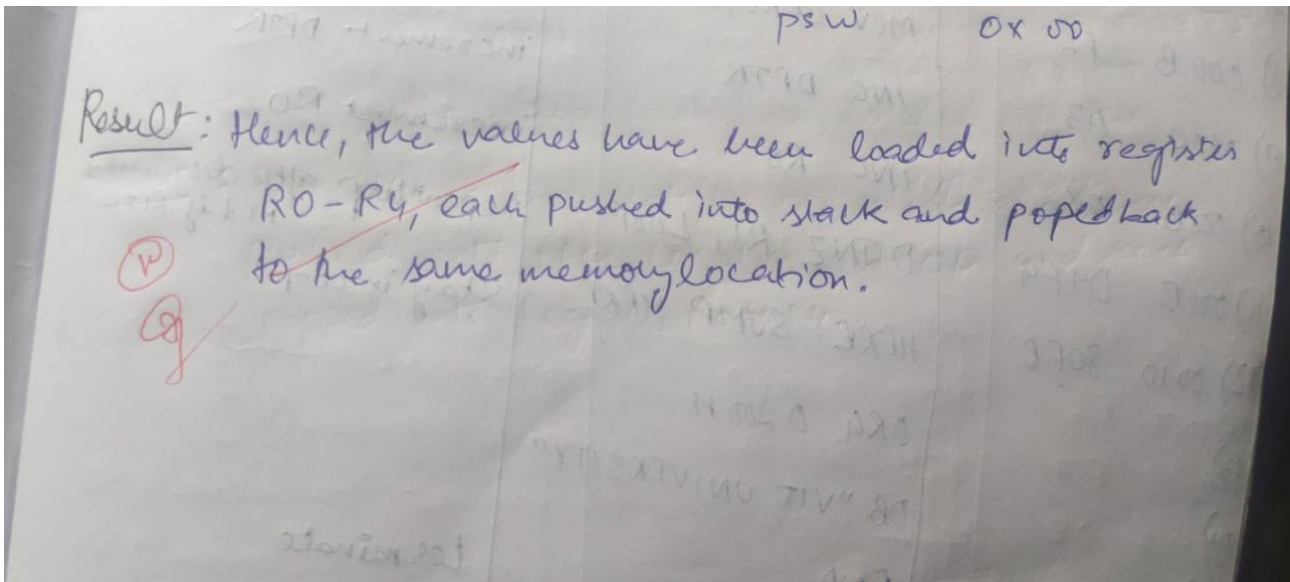
Address	Hex	Assembly	Comment
C:0x0000	7825	MOV	R0, #0x25
3:		MOV	R1, #35H
C:0x0002	7935	MOV	R1, #0x35
4:		MOV	R2, #45H
C:0x0004	7A45	MOV	R2, #0x45
5:		MOV	R3, #55H
C:0x0006	7B55	MOV	R3, #0x55
6:		MOV	R4, #65H
C:0x0008	7C65	MOV	R4, #0x65
7:		PUSH	0
C:0x000A	C000	PUSH	0x00
8:		PUSH	1
C:0x000C	C001	PUSH	0x01
9:		PUSH	2

PUSH POP 3.asm

```

1 ORG 0000H
2 MOV R0, #25H
3 MOV R1, #35H
4 MOV R2, #45H
5 MOV R3, #55H
6 MOV R4, #65H
7 PUSH 0
8 PUSH 1

```

After Execution:**Result:**

Task 4: Data Transfer Operations

Program 1:

Aim: to transfer a string of data "VIT UNIVERSITY" from ROM locations starting at 200H to RAM locations starting at 40H

Software Requirement: Keil Software

Program:

```

1  ORG 0000H
2  MOV A, #00H
3  MOV DPTR, #0200H
4  MOV R1, #0EH
5  MOV R0, #40H
6  LOOP: CLR A
7  MOVC A, @A+DPTR
8  MOV @R0, A
9  INC DPTR
10 INC R0
11 DJNZ R1, LOOP
12 HERE: SJMP HERE
13 ORG 0200H
14 DB "VIT UNIVERSITY"
15 END

```

Output:

Before Execution:

The screenshot displays the Keil IDE interface with two main panels. On the left, the 'Registers' window shows the state of various registers before execution. On the right, the 'Disassembly' window shows the assembly code being executed.

Registers Window:

Register	Value
r0	0x00
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
a	0x00
b	0x00
sp	0x00
sp_max	0x07
PC \$	C:0x0000
auxr1	0x00
dptr	0x0000
states	0
sec	0.00000000
psw	0x00

Disassembly Window:

```

C:0x0000 7400 MOV A, #0x00
3: MOV DPTR, #0200H
C:0x0002 900200 MOV DPTR, #0x0200
4: MOV R1, #0EH
C:0x0005 790E MOV R1, #0x0E
5: MOV R0, #40H
C:0x0007 7840 MOV R0, #0x40
6: LOOP: CLR A
C:0x0009 E4 CLR A

```

The assembly code in the disassembly window matches the program code shown in the previous block, indicating the state of the program before execution begins.

After Execution:

Register	Value
r0	0x4e
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00

Register	Value
a	0x59
b	0x00
sp	0x07
sp_max	0x07
PC \$	C:0x0010
auxr1	0x00
dptr	0x020e
states	131
sec	0.00004764
psw	0x00

Address	Hex	Op1	Op2
C:0x0009	E4	CLR	A
7:		MOV	A, @A+DPTR
C:0x000A	93	MOV	A, @A+DPTR
8:		MOV	@R0, A
C:0x000B	F6	MOV	@R0, A
9:		INC	DPTR
C:0x000C	A3	INC	DPTR
10:		INC	R0
C:0x000D	08	INC	R0
11:		DJNZ	R1, LOOP
C:0x000E	D9F9	DJNZ	R1, LOOP (C:0009)
12:		SJMP	HERE
C:0x0010	80FE	SJMP	HERE (C:0010)
C:0x0011	00	MOV	

Address	Hex	Op1	Op2
6		CLR	A
7		MOV	A, @A+DPTR
8		MOV	@R0, A
9		INC	DPTR
10		INC	R0
11		DJNZ	R1, LOOP
12		SJMP	HERE
13		ORG	0200H

Result:

Result: Hence, the data string "VIT UNIVERSITY" has been transferred from ROM location 200H to RAM locations 40H.

Program 2:

Aim: To Transfer a string of data "VIT UNIVERSITY" from ROM locations starting at 200H to RAM locations starting at 40H and then move it to RAM locations starting at 60H

Software Requirement: Keil Software

Program:

```

1  ORG 0000H
2  MOV A,#00H
3  MOV DPTR,#0200H
4  MOV R0,#40H
5  MOV R1,#0EH
6  LOOP: CLR A
7  MOVC A,@A+DPTR
8  MOV @R0,A
9  INC R0
10 INC DPTR
11 DJNZ R1,LOOP
12 MOV R0,#40H
13 MOV R1,#60H
14 MOV R3,#0EH
15 LOOP2: CLR A
16 MOV A,@R0
17 MOV @R1,A
18 INC R0
19 INC R1
20 DJNZ R3,LOOP2
21 HERE: SJMP HERE
22 ORG 0200H
23 DB "VIT UNIVERSITY"
24 END

```

Output:**Before Execution:**

The screenshot displays the Keil IDE interface before program execution. The **Registers** window on the left shows the state of various registers: **r0-r7** are 0x00, **sp** is 0x07, **PC** is 0x0000, and other system registers like **auxr1**, **dptr**, **states**, **sec**, and **psw** are also 0x00. The **Disassembly** window on the right shows the first instructions: **MOV A, #0x00** at address 0000, followed by **MOV DPTR, #0x0200** at 0002, **MOV R0, #40H** at 0005, and **MOV R1, #0EH** at 0007. The **DATA TRANSFER 2.asm** source code window at the bottom shows the corresponding assembly code, with the first line being **ORG 0000H** and the second line being **MOV A, #00H**.

After Execution:

Register	Value
r0	0x4e
r1	0x6e
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
Sys	
a	0x59
b	0x00
sp	0x07
sp_max	0x07
PC \$	C:0x001D
auxr1	0x00
dptr	0x020e
states	232
sec	0.00008436
psw	0x00

Address	Hex	Op	Comment
C:0x0016	E4	CLR	A
16:	MOV	A,@R0	
C:0x0017	E6	MOV	A,@R0
17:	MOV	@R1,A	
C:0x0018	F7	MOV	@R1,A
18:	INC	R0	
C:0x0019	08	INC	R0
19:	INC	R1	
C:0x001A	09	INC	R1
20:	DJNZ	R3,LOOP2	
C:0x001B	DBF9	DJNZ	R3,LOOP2 (C:0016)
21:	HERE: SJMP	HERE	
C:0x001D	80FE	SJMP	HERE (C:001D)
C:0x001E	00	MOD	

DATA TRANSFER 2.asm	
18	INC R0
19	INC R1
20	DJNZ R3,LOOP2
21	HERE: SJMP HERE
22	ORG 0200H
23	DB "VIT UNIVERSITY"
24	END

Result:

Result: Hence, the data string "VIT UNIVERSITY" has been transferred from ROM location starting at 200H to RAM location starting at 40H and then RAM location starting at 60H.

Task 5: Data Transfer and Arithmetic Operations

Program 1:

Aim: To add 10 bytes of data stored in ROM locations starting at 200H and store the results in registers R2 and R3

Software Requirement: Keil Software

Program:

```

1  ORG 0000H
2  MOV DPTR, #200H
3  MOV R0, #10H
4  LOOP: CLR A
5  MOVC A, @A+DPTR
6  ADD A, R2
7  JNC NEXT
8  INC R3
9  NEXT: INC DPTR
10 MOV R2, A
11 DJNZ R0, LOOP
12 HERE: SJMP HERE
13 ORG 200H
14 DB 22H, 23H, 24H, 25H, 26H, 27H, 28H, 22H, 23H, 24H, 25H, 26H, 27H, 28H
15 END

```

Output:

Before Execution:

The screenshot displays the Keil IDE interface. On the left, the 'Registers' window shows the state of various registers before execution. On the right, the 'Disassembly' window shows the assembly code being executed, with the current instruction highlighted.

Register	Value
r0	0x00
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
sys	
a	0x00
b	0x00
sp	0x00
sp_max	0x07
PC	0x0000
auxr1	0x00
dptr	0x0000
states	0
sec	0.00000000
psw	0x00

The Disassembly window shows the following instructions:

```

C:0x0000 900300 MOV DPTR, #0x0300
3: MOV R0, #10H
C:0x0003 7810 MOV R0, #0x10
4: LOOP: CLR A
C:0x0005 E4 CLR A
5: MOVC A, @A+DPTR
C:0x0006 93 MOVC A, @A+DPTR
6: ADD A, R2
C:0x0007 2A ADD A, R2
7: JNC NEXT
C:0x0008 D4 DA A
C:0x0009 5001 JNC C:000C
9: NEXT: INC DPTR
C:0x000B 0B INC DPTR

```

The assembly source code window shows the following code:

```

1  ORG 0000H
2  MOV DPTR, #200H
3  MOV R0, #10H
4  LOOP: CLR A
5  MOVC A, @A+DPTR
6  ADD A, R2
7  JNC NEXT
8  INC R3

```

After Execution:

Register	Value
r0	0x01
r1	0x00
r2	0x04
r3	0x04
r4	0x00
r5	0x00
r6	0x00
r7	0x00
a	0x04
b	0x00
sp	0x07
sp_max	0x07
PC \$	C:0x000D
auxr1	0x00
dptr	0x0310
states	196
sec	0.00007127
psw	0x01

Address	Hex	Op	Comment
11	DJNZ R0, LOOP		
C:0x000D	FA	MOV	R2, A
C:0x000E	D8F5	DJNZ	R0, LOOP (C:0005)
C:0x0010	80FE	SJMP	C:0010
C:0x0012	00	NOP	
C:0x0013	00	NOP	
C:0x0014	00	NOP	
C:0x0015	00	NOP	
C:0x0016	00	NOP	
C:0x0017	00	NOP	
C:0x0018	00	NOP	
C:0x0019	00	NOP	
C:0x001A	00	NOP	
C:0x001B	00	NOP	

Line	Code
9	NEXT: INC DPTR
10	MOV R2, A
11	DJNZ R0, LOOP
12	HERE: SJMP HERE
13	ORG 200H
14	DB 22H, 23H, 24H, 25H, 26H, 27H, 28H, 22H, 23H, 24H, 25H, 26H, 27H, 28H
15	END

Result:

Result:- Hence, 10 bytes of data stored in 200H ROM location has been added and result stored in Registers R2 & R3.

Program 2:

Aim: To add 10 bytes of BCD data stored in ROM locations starting at 300H and store the results in registers R2 and R3

Software Requirement: Keil Software

Program:

```

1  ORG 0000H
2  MOV DPTR, #300H
3  MOV R0, #10H
4  LOOP: CLR A
5  MOVC A, @A+DPTR
6  ADD A, R2
7  DA A
8  JNC NEXT
9  INC R3
10 NEXT: INC DPTR
11 MOV R2, A
12 DJNZ R0, LOOP
13 HERE: SJMP HERE
14 ORG 300H
15 DB 22H, 43H, 23H, 34H, 31H, 77H, 91H, 33H, 43H, 7H
16 END

```

Output:

Before Execution:

The screenshot displays the Keil IDE interface. On the left, the 'Registers' window shows the state of various registers before execution. On the right, the 'Disassembly' window shows the assembly code being executed, with the current instruction highlighted.

Register	Value
r0	0x00
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
a	0x00
b	0x00
sp	0x00
sp_max	0x07
PC	0x0000
auxr1	0x00
dptr	0x0000
states	0
sec	0.00000000
psw	0x00

The Disassembly window shows the following instructions:

```

C:0x0000 900300 MOV DPTR, #0x0300
3: MOV R0, #10H
C:0x0003 7810 MOV R0, #0x10
4: LOOP: CLR A
C:0x0005 E4 CLR A
5: MOVC A, @A+DPTR
C:0x0006 93 MOVC A, @A+DPTR
6: ADD A, R2
C:0x0007 2A ADD A, R2
7: DA A
C:0x0008 D4 DA A
8: JNC NEXT
C:0x0009 5001 JNC NEXT (C:000C)
9: INC R3

```

The assembly code window shows the following instructions:

```

1 ORG 0000H
2 MOV DPTR, #300H
3 MOV R0, #10H
4 LOOP: CLR A
5 MOVC A, @A+DPTR
6 ADD A, R2
7 DA A
8 JNC NEXT

```

After Execution:

The screenshot displays the state of the microcontroller's registers and the disassembled code after execution.

Registers Window:

Register	Value
r0	0x00
r1	0x00
r2	0x04
r3	0x04
r4	0x00
r5	0x00
r6	0x00
r7	0x00
a	0x04
b	0x00
sp	0x07
sp_max	0x07
PC \$	C:0x0010
auxr1	0x00
dptr	0x0310
states	199
sec	0.00007236
psw	0x01

Disassembly Window:

```

11: MOV R2,A
C:0x000D FA MOV R2,A
12: DJNZ R0,LOOP
C:0x000E D8F5 DJNZ R0,LOOP(C:0005)
13: HERE: SJMP HERE
C:0x0010 80FE SJMP HERE(C:0010)
C:0x0012 00 NOP
C:0x0013 00 NOP
C:0x0014 00 NOP
C:0x0015 00 NOP
C:0x0016 00 NOP
C:0x0017 00 NOP
C:0x0018 00 NOP
C:0x0019 00 NOP
  
```

DATA TRANSFER ARITHMETICS 2.asm:

```

10 NEXT: INC DPTR
11 MOV R2,A
12 DJNZ R0,LOOP
13 HERE: SJMP HERE
14 ORG 300H
15 DB 22H,43H,23H,34H,31H,77H,91H,33H,43H,7H
16 END
  
```

Result:

Result: Hence, 10 bytes of BCD data stored in ROM location starting at 300H and store the results in Register R2 & R3.

Task 6: Base Conversion Operations

Program 1:

Aim: to get a byte of hex data from P1, convert it to decimal and then to ASCII. Then place the ASCII results into RAM locations starting at 40H

Software requirement: Keil Software

Program:

```

1  ORG 0000H
2  MOV P1, #0FBH
3  MOV R0, #40H
4  MOV A, P1
5  LOOP: MOV B, #10
6  DIV AB
7  XCH A, B
8  ADD A, #30H
9  MOV @R0, A
10 XCH A, B
11 INC R0
12 JNZ LOOP
13 END

```

Output:

Before Execution:

The screenshot displays the Keil IDE interface. On the left, the 'Registers' window shows the state of various registers before execution. The 'Disassembly' window on the right shows the assembly code being executed, with the first instruction highlighted.

Register	Value
r0	0x00
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
a	0x00
b	0x00
sp	0x00
sp_max	0x07
PC	0x0000
auxr1	0x00
dptr	0x0000
states	0
sec	0.00000000
psw	0x00

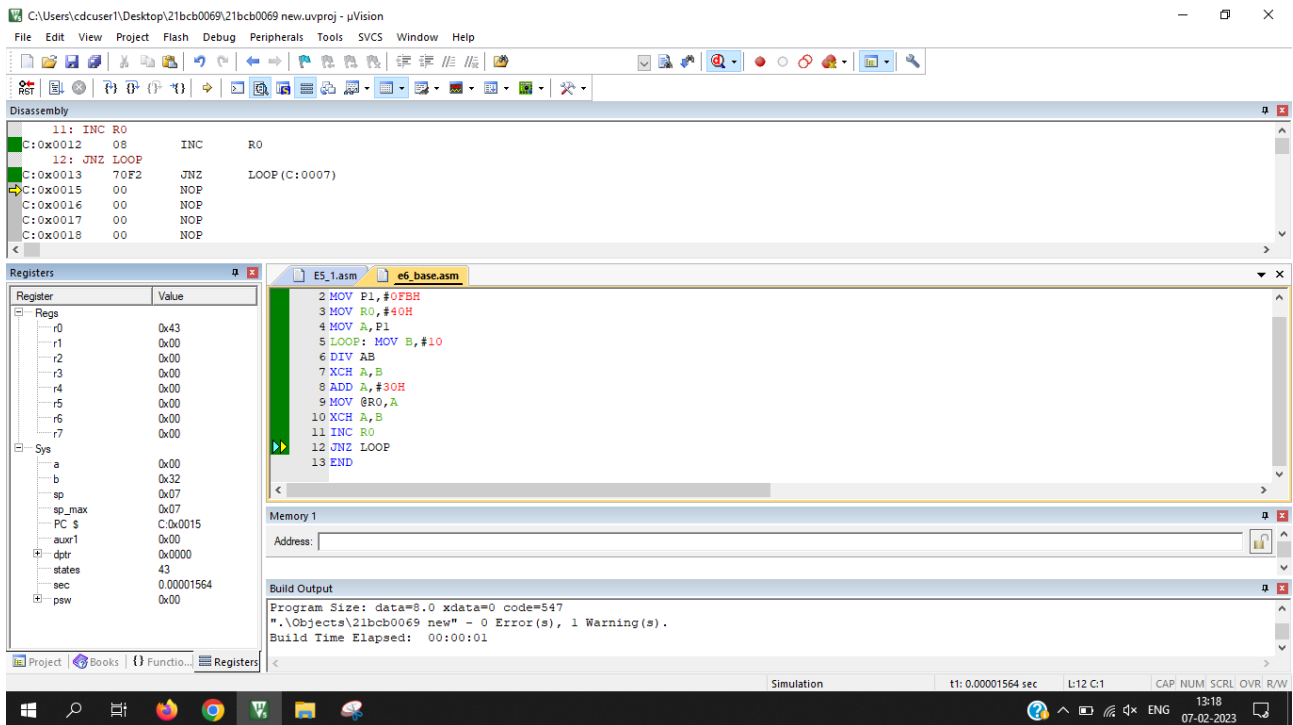
Address	Hex	Assembly	Comment
C:0x0000	7590FB	MOV	P1 (0x90), #0xFB
3:		MOV	R0, #40H
C:0x0003	7840	MOV	R0, #0x40
4:		MOV	A, P1
C:0x0005	E590	MOV	A, P1 (0x90)
5:		MOV	B, #10
C:0x0007	75F00A	MOV	B (0xF0), #0x0A
6:		DIV	AB
C:0x000A	84	DIV	AB
7:		XCH	A, B
C:0x000B	C5F0	XCH	A, B (0xF0)
8:		ADD	A, #30H
C:0x000D	2430	ADD	A, #0x30

The assembly code in the Disassembly window is as follows:

```

1  ORG 0000H
2  MOV P1, #0FBH
3  MOV R0, #40H
4  MOV A, P1
5  LOOP: MOV B, #10
6  DIV AB
7  XCH A, B
8  ADD A, #30H

```


After Execution:**Result:**