

19BEC0358



VIT®

Vellore Institute of Technology

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

School of Electronics Engineering ,VIT, Vellore

Reg.No	19BEC0358		
Student Name	ARPIT PATAWAT		
Course Code	ECE3003	Slot & Semester	L35+L36 FALL -- 2021-22
Course Name	Microcontroller and its applications		
Program Title	Lab Assignment 1(ADD BCD STACK PROGRAMS)		
Date of Exp.	16-08-21	Date of Submission	30-08-21
Faculty	A.Karthikeyan		

Question

- 1. Write an 8051 ALP to add the following data and then use the simulator to examine the CY flag.**
a) INPUT DATA YOUR REG NO: 19BEC0358 Example 01 9B EC 03 58.
b) Five max 8BIT NUMBERS (FF, FF, FF, FF, FF)
- 2. Write an 8051 ALP assemble a program to load values into each of registers R0 – R5 and then push each of these registers onto the stack. Single-step the program, and examine the stack and the SP register after the execution of each instruction.**
- 3. Write an 8051 assemble language program to:**
(a) Set SP = 0D,
(b) Load a different value in each of RAM locations 0D, 0C, 0B, 0A, 09, and 08,
(c) POP each stack location into registers R0 - R4. Use the simulator to single-step and examine the registers, the stack, and the stack pointer.
- 4. Write an 8051 ALP assemble a program to load values into each of registers R0 - R4 and then push each of these registers onto the stack and pop them back. Single-step the program, and examine the stack and the SP register after the execution of each instruction.**
- 5. Write an 8051 ALP to add 10 bytes of BCD data and store the result in R2 and R3. The bytes are stored in ROM space starting at 300H.**

The data would look as follows:

MYDATA: DB ;

Input: i) (10 DATA) : 22H,43H,23H,34H,31H,77H,91H,33H,43H,07H

ii) (10 DATA) : YOUR REG NO: _____

Example 01 7B EC 02 75 00 00 00 00 00 00

Notice that you must first bring the data from ROM space into the CPU's RAM, then add them together. Use a simulator to single-step the program and examine the data.

TASK 1.a) -

Aim: To write an 8051 ALP to perform addition of 5 8BIT NUMBERS (Reg. No.) using keil software and to verify the result manually.

Tools Required: Keil Microvision Software

Algorithm:

- 1.Load first 2 value out of 5 to operands A and B.
- 2.Add both the operands and if carry flag is 1 then increment R0.
- 3.load the value of A to B and then load the third value to A.
- 4.Again add both the values and increment R0 if carry flag is 1.
- 5.Repeat the process until we add all the 5 values.

Program:

Label	Mnemonics	Operands	addressing mode used	Machine cycle Required	Memory Byte Required	Type of Instruction	Comments	Flags getting affected by the Instruction
	ORG 0000h						Assembler directive	NONE
	MOV	A, #001H	Immediate	1	2	Data transfer	Load FF to A	PARITY
	MOV	B, #09BH	Immediate	1	2	Data Transfer	Load FF to B	NONE
	ADD	A, B	Register	1	1	Arithmet ic	Add content of A and B and store it in A	PARITY,
	JNC	L1	Indexed	2	2	Boolean	Jump to label L1 if there is no carry	NONE
	INC	R0	Register	1	1	Arithmet	Register	NONE

						ic	R0 is incremented by 1	
L1	MOV	B, A	Register	1	1	Data transfer	Load the content of A into B	NONE
	MOV	A, #0ECH	Immediate	1	2	Data transfer	Load FF to A	PARITY
	ADD	A, B	Register	1	1	Arithmet ic	Add content of A and B and store it in A	PARITY, AUXILIARY CARRY FLAG, CARRY FLAG
	JNC	L2	Indexed	2	2	Boolean	Jump to label L2 if there is no carry	NONE
	INC	R0	Immediate	1	1	Arithmet ic	Register R0 is incremented by 1	NONE
L2	MOV	B, A	Register	1	1	Data transfer	Load the content of A into B	NONE
	MOV	A, #003H	Immediate	1	2	Data transfer	Load FF to A	NONE
	ADD	A, B	Register	1	1	Arithmet ic	Add content of A and B and store it in A	AUXILIARY CARRY FLAG, CARRY FLAG
	JNC	L3	Indexed	2	2	Boolean	Jump to label L3 if there is no carry	NONE

	INC	R0	Immediate	1	1	Arithmet ic	Register R0 is incremented by 1	NONE
L3	MOV	B, A	Register	1	1	Data transfer	Load the content of A into B	NONE
	MOV	A, #058H	Immediate	1	2	Data transfer	Load FF to A	PARITY
	ADD	A, B	Register	1	1	Arithmet ic	Add content of A and B and store it in A	AUXILIARY CARRY FLAG
	JNC	L4	Indexed	2	2	Boolean	Jump to label L4 if there is no carry	NONE
	INC	R0	Immediate	1	1	Arithmet ic	Register R0 is incremented by 1	NONE
L4							Label L4	
	END						End of programme	

Output: Registers containing the Result: R0=01H, A=E3H

Manual Calculation: hexa decimal to binary conversion ->

$$01 = 1, 9B = 155, EC = 236, 03 = 3, 58 = 88$$

$$1 + 155 + 236 + 3 + 88 = 483$$

$$483 = 1E3 \text{ in hexa decimal}$$

Answer – **1E3**

Results and Observations

Program and registers before execution:

Registers Window:

Register	Value
r0	0x00
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
sp	0x00
sp_max	0x00
dptr	0x0000
PC \$	C:0x0000
states	0
sec	0.00000000
pw	0x00
p	0
f1	0
ov	0
rs	0
f0	0
ac	0
cy	0

Disassembly Window:

```

C:0x0000 7401 MOV A, #0x01
C:0x0002 75F09B MOV B, #0x9B
ADD_1.ASM
1 ORG 0000h
2 MOV A, #001H ; add 5 DIGIT OF REG NO. 8 bit number(01, 9B, EC , 03, 58) h
3 MOV B, #09BH
4 ADD A,B
5 JNC L1
6 INC R0
7 L1: MOV B,A
8 MOV A, #0ECH
9 ADD A,B
10 JNC L2
11 INC R0
12 L2: MOV B,A
13 MOV A, #003H
14 ADD A,B
15 JNC L3
16 INC R0
17 L3: MOV B,A
18 MOV A, #058H
19 ADD A,B
20 JNC L4
21 INC R0
22 L4:
23 END
24

```

Program and registers after execution: FINAL STEP

Registers Window:

Register	Value
r0	0x01
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
sp	0x00
sp_max	0x07
dptr	0x0000
PC \$	C:0x0025
states	22
sec	0.00000660
pw	0x41
p	1
f1	0
ov	0
rs	0
f0	0
ac	1
cy	0

Disassembly Window:

```

C:0x0025 00 NOP
C:0x0026 00 NOP
C:0x0027 00 NOP
ADD_1.ASM
1 ORG 0000h
2 MOV A, #001H ; add 5 DIGIT OF REG NO. 8 bit number(01, 9B, EC , 03, 58) h
3 MOV B, #09BH
4 ADD A,B
5 JNC L1
6 INC R0
7 L1: MOV B,A
8 MOV A, #0ECH
9 ADD A,B
10 JNC L2
11 INC R0
12 L2: MOV B,A
13 MOV A, #003H
14 ADD A,B
15 JNC L3
16 INC R0
17 L3: MOV B,A
18 MOV A, #058H
19 ADD A,B
20 JNC L4
21 INC R0
22 L4:
23 END
24

```

Command Window:

```

Load "C:\\study\\FALL SEM\\Micro controller lab\\LAB 1\\19BEC0358\\Objects\\LAB1_A"
*** error 65: access violation at C:0x0025 : no 'execute/read' permission

```

Inferences:

1. About PSW VALUES – Parity flag is 1 since there are odd numbers of 1 in E3, auxiliary carry flag is also 1 since in the last addition so there is a carry from D3 to D4 and also there is no carry Flag.
2. ABOUT THE OUTPUT VALUES IN REGISTERS – every time when there is carry, R0 is incremented that's why we have R0 = 1 only.

Result: The 8051 ALP to perform addition of 5 8BIT NUMBERS is executed using Keil software and the results are verified manually.

-----XXXXXX-----

TASK 1.b) -

Aim: To write an 8051 ALP to perform addition of 5 8BIT NUMBERS using Keil software and to verify the result manually.

Tools Required: Keil Microvision Software

Algorithm:

- 1.Load first 2 value out of 5 to operands A and B.
- 2.Add both the operands and if carry flag is 1 then increment R0.
- 3.load the value of A to B and then load the third value to A.
- 4.Again add both the values and increment R0 if carry flag is 1.
- 5.Repeat the process until we add all the 5 values.

Program:

Label	Mnemonics	Operands	addressing mode used	Machine cycle Required	Memory Byte Required	Type of Instruction	Comments	Flags getting affected by the Instruction
	ORG 0000h						Assembler directive	NONE
	MOV	A, #0FFH	Immediate	1	2	Data transfer	Load FF to A	NONE
	MOV	B, #0FFH	Immediate	1	2	Data Transfer	Load FF to B	NONE
	ADD	A, B	Register	1	1	Arithmetic	Add content of A and B and store it in A	PARITY, AUXILIARY CARRY FLAG, CARRY FLAG
	JNC	L1	Indexed	2	2	Boolean	Jump to label L1 if there is no	NONE

							carry	
	INC	R0	Immediate	1	1	Arithmet ic	Register R0 is incremented by 1	NONE
L1	MOV	B, A	Register	1	1	Data transfer	Load the content of A into B	NONE
	MOV	A, #0FFH	Immediate	1	2	Data transfer	Load FF to A	NONE
	ADD	A, B	Register	1	1	Arithmet ic	Add content of A and B and store it in A	PARITY
	JNC	L2	Indexed	2	2	Boolean	Jump to label L2 if there is no carry	NONE
	INC	R0	Immediate	1	1	Arithmet ic	Register R0 is incremented by 1	NONE
L2	MOV	B, A	Register	1	1	Data transfer	Load the content of A into B	NONE
	MOV	A, #0FFH	Immediate	1	2	Data transfer	Load FF to A	NONE
	ADD	A, B	Register	1	1	Arithmet ic	Add content of A and B and store it in A	PARITY
	JNC	L3	Indexed	2	2	Boolean	Jump to label L3 if there is no carry	NONE

	INC	R0	Immediate	1	1	Arithmet ic	Register R0 is incremented by 1	NONE
L3	MOV	B, A	Register	1	1	Data transfer	Load the content of A into B	NONE
	MOV	A, #0FFH	Immediate	1	2	Data transfer	Load FF to A	NONE
	ADD	A, B	Register	1	1	Arithmet ic	Add content of A and B and store it in A	PARITY
	JNC	L4	Indexed	2	2	Boolean	Jump to label L4 if there is no carry	NONE
	INC	R0	Immediate	1	1	Arithmet ic	Register R0 is incremented by 1	NONE
L4							Label L4	
	END						End of programme	

Output: Registers containing the Result: R0=04H, A=FBH

Manual Calculation: FF = 255 (in binary)

FF + FF +FF + FF + FF is -

255 + 255+ 255+ 255+ 255 = 1275 in decimal = 4FB in Hexa decimal

Results and Observations

Program and registers 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
dptr	0x0000
PC \$	C.0x0000
states	0
sec	0.00000000
psw	0x00
p	0
f1	0
ov	0
rs	0
f0	0
ac	0
cy	0

Disassembly

```

6: INC R0 ; increment R0
C:0x0009 09 INC R0

ADD_2.ASM
1 ORG 0000h ; add 5 MAX 8 bit number(FF, FF, FF, FF, FF) h
2 MOV A, #0FFH ;load FF to A
3 MOV B, #0FFH ;load FF to B
4 ADD A,B ;add A and B
5 JNC L1 ;jump to L1 label if there is no carry
6 INC R0 ; increment R0
7 L1: MOV B,A
8 MOV A, #0FFH
9 ADD A,B
10 JNC L2
11 INC R0
12 L2: MOV B,A
13 MOV A, #0FFH
14 ADD A,B
15 JNC L3
16 INC R0
17 L3: MOV B,A
18 MOV A, #0FFH
19 ADD A,B
20 JNC L4
21 INC R0
22 L4:
23 END

```

Program and registers after execution: FINAL STEP

Registers

Register	Value
r0	0x04
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
Sys	
a	0xb
b	0xc
sp	0x07
sp_max	0x07
dptr	0x0000
PC \$	C.0x0025
states	25
sec	0.00000750
psw	0xc1
p	1
f1	0
ov	0
rs	0
f0	0
ac	1
cy	1

Disassembly

```

C:0x0025 00 NOP
C:0x0026 00 NOP

ADD_2.ASM
1 ORG 0000h ; add 5 MAX 8 bit number(FF, FF, FF, FF, FF) h
2 MOV A, #0FFH ;load FF to A
3 MOV B, #0FFH ;load FF to B
4 ADD A,B ;add A and B
5 JNC L1 ;jump to L1 label if there is no carry
6 INC R0 ; increment R0
7 L1: MOV B,A
8 MOV A, #0FFH
9 ADD A,B
10 JNC L2
11 INC R0
12 L2: MOV B,A
13 MOV A, #0FFH
14 ADD A,B
15 JNC L3
16 INC R0
17 L3: MOV B,A
18 MOV A, #0FFH
19 ADD A,B
20 JNC L4
21 INC R0
22 L4:
23 END

```

Inferences:

1. About PSW VALUES – Parity flag is 1 since there are odd numbers of 1 in FB, auxiliary carry flag is also 1 since in the last addition for FC+FF, there is a carry from D3 to D4 and also there is carry at D7 so carry flag is also 1.
2. ABOUT THE OUTPUT VALUES IN REGISTERS – every time when there is carry, R0 is incremented that's why we have R0 = 4.

Result: The 8051 ALP to perform addition of 5 8BIT NUMBERS is executed using Keil software and the results are verified manually.

-----XXX-----

TASK 2)

Aim: To write an 8051 ALP to load values from registers and to push them to stack using Keil software and to verify the result manually.

Tools Required: Keil Microvision Software

Algorithm:

- 1.load different numbers to registers from R0 - R5
- 2.push each value into the stack

Program:

Label	Mnemonics	Operands	addressing mode used	Machine cycle Required	Memory Byte Required	Type of Instruction	Comments	Flags getting affected by the Instruction
	ORG	0000h						
	MOV	R0, #001H	Immediate	1	2	Data transfer	Value is loaded to R0	NONE
	MOV	R1, #09BH	Immediate	1	2	Data transfer	Value is loaded to R1	NONE
	MOV	R2, #0ECH	Immediate	1	2	Data transfer	Value is loaded to R2	NONE
	MOV	R3, #003H	Immediate	1	2	Data transfer	Value is loaded to R3	NONE
	MOV	R4, #058H	Immediate	1	2	Data transfer	Value is loaded to R4	NONE
	MOV	R5, #012H	Immediate	1	2	Data transfer	Value is loaded to R5	NONE
	PUSH	0	Direct	2	2	Data transfer	Value is pushed to	NONE

							stack	
	PUSH	1	Direct	2	2	Data transfer	Value is pushed to stack	NONE
	PUSH	2	Direct	2	2	Data transfer	Value is pushed to stack	NONE
	PUSH	3	Direct	2	2	Data transfer	Value is pushed to stack	NONE
	PUSH	4	Direct	2	2	Data transfer	Value is pushed to stack	NONE
	PUSH	5	Direct	2	2	Data transfer	Value is pushed to stack	NONE
	END						Programme ends here	NONE

Output: Registers containing the Result: R0 =01, R1=9B, R2 =EC, R3=03, R4 = 58, R5 =12

Manual Calculation: NONE

Results and Observations

Program and registers before execution:

The screenshot shows the µVision IDE interface with the following components:

- Title Bar:** C:\study\FALL SEM\Micro controller lab\labb 2\19BEC035B\lab2_a.uvproj - µVision
- Menu Bar:** File, Edit, View, Project, Flash, Debug, Peripherals, Tools, SVCS, Window, Help
- Toolbar:** Includes icons for New, Open, Save, Build, Run, Stop, and Simulation.
- Registers Window:** Shows the state of various registers (r0-r7, Sys, PC \$, states, sec, psw) with their addresses and values.
- Disassembly Window:** Displays assembly code for TASK1_2_ASM starting at address 0000H. The code includes pushes, moves, and a loop.
- Memory Window:** A floating window titled "Memory 1" showing a dump of memory starting at address D00H. It displays a series of FF and 00 bytes.
- Command Window:** Shows the command "Running with Code Size Limit: 2K" and the path "Load C:\\study\\FALL SEM\\Micro controller lab\\labb 2\\19BEC035B\\Objects\\lab2_a".
- Bottom Bar:** Includes tabs for ASM, ASSIGN, BreakDisable, BreakEnable, BreakKill, BreakList, BreakAccess, COVERAGE, DEFINE, DIR, Display, Enter, EVALUate, EXIT, FUNC, Go, INCLUDE, KILL, LogicAnalyze, LOAD, LOG, MAP, MODE, Ostep, Simulation, t1: 0.0000000 sec, L1 C1, CAP NUM SCR1 OVR R/W, and a system status bar showing 36°C, AQI 108, and 02:24 PM 20-08-2021.

Program and registers after execution: FINAL STEP

The screenshot shows the µVision IDE interface with the following details:

- Title Bar:** C:\study\FALL SEM\Micro controller lab\labb 2\19BEC0358\lab2_a.uvproj - µVision
- Menu Bar:** File, Edit, View, Project, Flash, Debug, Peripherals, Tools, SVCS, Window, Help
- Toolbars:** Standard, Project, Registers, Memory, Stack, Watch, Breakpoints, Symbols, CPU, Bus, Memory, Registers, Stack, Watch, Breakpoints, Symbols, CPU, Bus.
- Registers Window:** Shows CPU registers (r0-r7, Sys, PC, dptr, psw) with their current values.
- Disassembly Window:** Displays assembly code for TASK1_2.ASM and STARTUP.AS1. The assembly code includes instructions like ORG, MOV, PUSH, and END.
- Memory Window:** Shows a memory dump from address D:0x00 to D:0xF0. The dump contains hex values corresponding to the assembly code.
- Command Window:** Displays the command line interface with the following output:

```
Running with Code Size Limit: 2K
Load "C:\\study\\FALL SEM\\Micro controller lab\\labb 2\\19BEC0358\\Objects\\lab2_a"
*** error 65: access violation at C:0x0018 : no 'execute/read' permission
```

Inferences:

1. About PSW VALUES – All are 0.
2. ABOUT THE OUTPUT VALUES IN REGISTERS – from R0 to R5 all the data is pushed to stack
Memory and stack pointer is incremented by 1 whenever data is pushed

Result: The 8051 ALP to PUSH 6 8Bit numbers is executed using Keil software and the results are verified manually.

-----XXX-----

TASK 3)

Aim: To write an 8051 ALP to load values to different RAM locations and POP each value to registers using Keil software and to verify the result manually.

Tools Required: Keil Microvision Software

Algorithm:

1. Set stack pointer location to 0D.
2. Load a different value at each RAM location.
3. POP each stack location into registers R0 – R4.

Program:

Label	Mnemonics	Operands	addressing mode used	Machine cycle Required	Memory Byte Required	Type of Instruction	Comments	Flags getting affected by the Instruction.
	ORG	0000h						
	MOV	SP, #0DH	Immediate	1	2	Data transfer	Value is loaded to stack pointer	NONE
	MOV	08H, #001H	Immediate	1	2	Data transfer	Value is loaded at the specified location	NONE
	MOV	09H, #09BH	Immediate	1	2	Data transfer	Value is loaded at the specified location	NONE
	MOV	0AH, #0ECH	Immediate	1	2	Data transfer	Value is loaded at the specified location	NONE

	MOV	0BH, #003H	Immediate	1	2	Data transfer	Value is loaded at the specified location	NONE
	MOV	0CH, #058H	Immediate	1	2	Data transfer	Value is loaded at the specified location	NONE
	MOV	011H, #00DH	Immediate	1	2	Data transfer	Value is loaded at the specified location	NONE
	POP	0	Direct	2	2	Data transfer	Value is popped from stack to R0	NONE
	POP	1	Direct	2	2	Data transfer	Value is popped from stack to R1	NONE
	POP	2	Direct	2	2	Data transfer	Value is popped from stack to R2	NONE
	POP	3	Direct	2	2	Data transfer	Value is popped from stack to R3	NONE
	POP	4	Direct	2	2	Data transfer	Value is popped from stack to R4	NONE
	END						Programme ends here	NONE

Output: Registers containing the Result: R0 =11, R1=58, R2 =03, R3=EC, R4 =9B

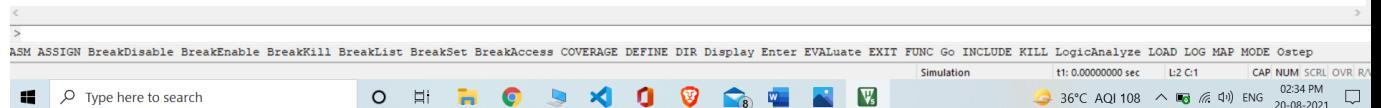
Manual Calculation: NONE

Results and Observations

Program and registers before execution:

The screenshot shows the µVision IDE interface with the following components:

- Title Bar:** C:\study\FALL SEM\Micro controller lab\lab\b2\19BEC0358\lab2_a.vproj - µVision
- Menu Bar:** File, Edit, View, Project, Flash, Debug, Peripherals, Tools, SVCS, Window, Help
- Toolbar:** Includes icons for Open, Save, Build, Run, Stop, and others.
- Registers View:** Shows registers r0 through r7 and various system registers like sp, pc, and flags.
- Disassembly View:** Displays assembly code for TASK1_3.ASM, including instructions like MOV and ADD.
- Memory View:** A floating window titled "Memory 1" showing memory starting at address D:00H.
- Command Window:** At the bottom, it says "Running with Code Size Limit: 2K" and "Load \"C:\\study\\FALL SEM\\Micro controller lab\\lab\\b2\\19BEC0358\\Objects\\lab2_a\"".



Program and registers after execution: FINAL STEP

The screenshot shows the Keil μVision IDE interface. The assembly code window displays the following instructions:

```

1 ORG 0000H
2 MOV SP, #0DH
3 MOV OBH, #001H
4 MOV O9H, #09BH
5 MOV OAH, #0ECH
6 MOV OBH, #003H
7 MOV OCH, #058H
8 MOV ODH, #011H
9 POP 0
10 POP 1
11 POP 2
12 POP 3
13 POP 4
14 END

```

The Registers window shows the following values:

Register	Value
r0	0x11
r1	0x58
r2	0x03
r3	0xec
r4	0xb
r5	0x00
r6	0x00
r7	0x00
sp	0x08
sp_max	0xd
dptr	0x0000
PC \$	C.0x001F
states	24
sec	0.00000720
psw	0x00
p	0
f1	0
ov	0
rs	0
f0	0
ac	0
cy	0

The Memory window shows the memory dump starting at address D:0x00.

The Command window shows the following error message:

```

Running with Code Size Limit: 2K
Load "C:\study\FALL SEM\Micro controller lab\labb 2\19BEC0358\Objects\lab2_a"
*** error 65: access violation at C:0x001F : no 'execute/read' permission

```

Inferences:

1. About PSW VALUES – All are 0.
2. ABOUT THE OUTPUT VALUES IN REGISTERS – all the data from stack is Pushed back to Registers R0-R4.

Result: The 8051 ALP to POP 5 8Bit numbers is executed using keil software and the results are verified manually.

-----XXXXX-----

TASK 4)

Aim: To write an 8051 ALP to PUSH values from Registers to Stack and POP them back using keil software and to verify the result manually.

Tools Required: Keil Microvision Software

Algorithm:

- 1.Load different numbers to registers from R0 -R4
- 2.PUSH each value to stack
- 3.POP each value back from stack

Program:

Label	Mnemonics	Operands	addressing mode used	Machine cycle Required	Memory Byte Required	Type of Instruction	Comments	Flags getting affected by the Instruction
	ORG	0000h						
	MOV	R0, #025H	Immediate	1	2	Data transfer	Value is loaded to R0	NONE
	MOV	R1, #035H	Immediate	1	2	Data transfer	Value is loaded to R1	NONE
	MOV	R2, #045H	Immediate	1	2	Data transfer	Value is loaded to R2	NONE
	MOV	R3, #055H	Immediate	1	2	Data transfer	Value is loaded to R3	NONE
	MOV	R4, #058H	Immediate	1	2	Data transfer	Value is loaded to R4	NONE
	MOV	R5, #065H	Immediate	1	2	Data transfer	Value is loaded to R5	NONE
	PUSH	0	Direct	2	2	Data	Value is	NONE

						transfer	pushed to stack	
	PUSH	1	Direct	2	2	Data transfer	Value is pushed to stack	NONE
	PUSH	2	Direct	2	2	Data transfer	Value is pushed to stack	NONE
	PUSH	3	Direct	2	2	Data transfer	Value is pushed to stack	NONE
	PUSH	4	Direct	2	2	Data transfer	Value is pushed to stack	NONE
	PUSH	5	Direct	2	2	Data transfer	Value is pushed to stack	NONE
	POP	0	Direct	2	2	Data transfer	Value is popped from stack to R0	NONE
	POP	1	Direct	2	2	Data transfer	Value is popped from stack to R1	NONE
	POP	2	Direct	2	2	Data transfer	Value is popped from stack to R2	NONE
	POP	3	Direct	2	2	Data transfer	Value is popped from stack to R3	NONE
	POP	4	Direct	2	2	Data transfer	Value is popped from stack to R4	NONE

	END						Programme ends here	NONE

Output: Registers containing the Result: R0 = 65, R1=55, R2 =45, R3=35, R4 = 25

Manual Calculation: NONE

Results and Observations

Program and registers before execution:

The screenshot shows the µVision IDE interface. The assembly code in the editor window is:

```

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

```

The Registers window shows initial values for R0-R4 and stack pointers. The Memory dump window shows the memory starting at address 0x0000.

Program and registers after execution: FINAL STEP

The screenshot shows the µVision IDE interface with the following components:

- Title Bar:** C:\study\FALL SEM\Micro controller lab\labb 2\19BEC0358\lab2_a.uvproj - µVision
- Menu Bar:** File, Edit, View, Project, Flash, Debug, Peripherals, Tool, SVCS, Window, Help
- Toolbar:** Includes icons for Open, Save, Build, Run, Stop, and various simulation and analysis tools.
- Registers Window:** Shows the state of CPU registers (R0-R7, Sys, P, F, OVR, RS, T0-T1, AC, CY) and memory locations (C0x001E to C0x0021).
- Disassembly Window:** Displays the assembly code for TASK1_4.ASM, starting with ORG 0000H and ending at END.
- Memory Window:** A floating window titled "Memory 1" showing the memory dump from address D:0x00 to D:0x4F. The memory contains binary data representing the assembly code.
- Command Window:** Shows the build log and error message: "Running with Code Size Limit: 2K", "Load "C:\\\\study\\\\FALL SEM\\\\Micro controller lab\\\\labb 2\\\\19BEC0358\\\\Objects\\\\lab2_a\"", and "*** error 65: access violation at C:0x001E : no 'execute/read' permission".
- Status Bar:** ASM ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE DEFINE DIR Display Enter EVALUATE EXIT FUNC Go INCLUDE KILL LogicAnalyze LOAD LOG MAP MODE Ostep Simulation t1: 0.00000750 sec L1 C4 CAP NUM SCR L R 36°C AQI 108 ENG 20-08-2021

Inferences:

1. About PSW VALUES – All are 0.
 2. ABOUT THE OUTPUT VALUES IN REGISTERS – from R0 to R4 all the data is pushed to stack Memory and again POP back, so their value gets reversed.

Result: The 8051 ALP to PUSH and POP of 5 8Bit numbers is executed using Keil software and the results are verified manually.

-XXX-

TASK 5.a)

Aim: To write an 8051 ALP to add 10 bytes of BCD data and store the result in R2 and R3 using Keil software and to verify the result manually.

Tools Required: Keil Microvision Software

Algorithm:

1. take 10 bytes of data and store it in DB at location C:300H
2. move data pointer to 300H location.
3. take data from DB, load it to A, add with R2 and store to R2.
4. check for decimal adjust accumulator content and increment R3 if it is present.
5. run a loop for 10 times to repeat this process.
6. finally the result is stored in R2 and R3.

Program:

Label	Mnemonics	Operands	addressing mode used	Machine cycle Required	Memory Byte Required	Type of Instruction	Comments	Flags getting affected by the Instruction
	ORG	000H						
	MOV	DPTR, #300H	Immediate	2	3	Data transfer	Data pointer is moved at the 300th location	NONE
	MOV	R0, #10H	Immediate	1	2	Data transfer	R0 is loaded with value 10	NONE
LOOP	CLR	A	Immediate	1	1	Logical	Clear value of A	NONE
	MOVC	A, @A+DPTR	Indexed	2	1	Data transfer	Move location to DPTR + @A	NONE
	ADD	A, R2	Register	1	1	Arithmet	Add R2 and A and	PARITY

						ic	store in A	
	DA	A	Immediate	1	1	Arithmet ic	Check for decimal adjust accumulator content for A	NONE
	JNC	NEXT	Indexed	2	2	Boolean	Jump to next if there is no carry	NONE
	INC	R3	Immediate	1	1	Arithmet ic	Increment R3	NONE
NEXT	INC	DPTR	immediate	2	1	arithmeti c	Increment DPTR	NONE
	MOV	R2, A	Register	1	1	Data transfer	Load the value of A into R2	NONE
	DZNZ	R0, LOOP	Indexed	2	2	Branchin g	Decrement and jump on non-zero	NONE
HERE	SJMP	HERE	Indexed	2	2	Branchin g	Short jump	NONE
	ORG	300H						NONE
	DB	22H, 43H, 23H, 34H, 31H, 77H, 91H, 33H, 43H, 07H						NONE

Output: Registers containing the Result: R2 =04, R3=04

Manual Calculation:

$$22+43+23+34+31+77+91+33+43+07 = 404 \text{ in decimal}$$

Results and Observations

Program and registers before execution:

The screenshot shows the µVision IDE interface with the following details:

- Title Bar:** C:\study\FALL SEM\Micro controller lab\lab2\19BEC0358\lab2_a\proj - µVision
- Menu Bar:** File, Edit, View, Project, Flash, Debug, Peripherals, Tools, SVCS, Window, Help
- Toolbars:** Standard toolbar with icons for Open, Save, Print, etc.
- Registers Window:** Shows registers r0 through r7 and system variables like Sys, PC, and PSW.
- Disassembly Window:** Displays assembly code for TASK_5_ASM:

```
        4: LOOP: CLR A
:0000005 E4      CLR    A
E: MOVE R A & RMOVED
1 ORG 000H
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,07H
16 END
17
```
- Memory Windows:** Two windows showing memory dump at address C300H and D00H.
- Command Window:** Displays build logs:

```
Running with Code Size Limit: 2K
Load "C:\study\FALL SEM\Micro controller lab\lab2\19BEC0358\Objects\lab2_a"
```
- Bottom Bar:** Includes buttons for ASM, ASSIGN, BreakDisable, BreakEnable, BreakKill, BreakList, BreakAccess, COVERAGE, DEFINE, DIR, Display, Enter, EVALUATE, EXIT, FUNC, Go, INCLUDE, KILL, LogicAnalyze, LOAD, LOG, MAP, MODE, Ostep, Simulation, t1: 0.0000000 sec, L1 C4, CAP NUM SCR L, OVR R/W, and status indicators for temperature (36°C), AQI (108), battery, signal strength, and network.
- Taskbar:** Shows the Start button, search bar, and pinned icons for File Explorer, Google Chrome, Microsoft Edge, File History, Task View, Mail, Word, Excel, and a VLC icon.

Program and registers after execution: FINAL STEP

The screenshot shows the Keil µVision IDE interface. The top menu bar includes File, Edit, View, Project, Flash, Debug, Peripherals, Tools, SVCS, Window, and Help. The left sidebar has sections for Registers (Registers, Sys) and Project. The main area displays assembly code for 'TASK1_5.ASM' and memory views for Memory 1 and Memory 2. The assembly code is as follows:

```

        4: LOOP: CLR A
        :0x0008 E4      CLR     A
        e* MVI    A,0A00H

        1 ORG 000H
        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,07H
        16 END
        17

```

The Registers window shows various registers with their current values. The Memory 1 window shows memory starting at address C:0000. The Memory 2 window shows memory starting at address D:0000. The Command window at the bottom shows the assembly code being run.

Inferences:

1. About PSW VALUES – All are 0 except parity flag which is 1
2. ABOUT THE OUTPUT VALUES IN REGISTERS – the data is taken and stored in R2 and R3.

Result: The 8051 ALP to perform ADDITION of 10 BCD NUMBERS is executed using Keil software and the results are verified manually.

-----XXX-----

TASK 5.b)

Aim: To write an 8051 ALP to add 10 bytes of BCD data and store the result in R2 and R3 using Keil software and to verify the result manually.

Tools Required: Keil Microvision Software

Algorithm:

1. take 10 bytes of data and store it in DB at location C:300H
2. move data pointer to 300H location.
3. take data from DB, load it to A, add with R2 and store to R2.
4. check for decimal adjust accumulator content and increment R3 if it is present.
5. run a loop for 10 times to repeat this process.
6. finally the result is stored in R2 and R3.

Program:

Label	Mnemonics	Operands	addressing mode used	Machine cycle Required	Memory Byte Required	Type of Instruction	Comments	Flags getting affected by the Instruction
	ORG	000H						
	MOV	DPTR, #300H	Immediate	2	3	Data transfer	Data pointer is moved at the 300th location	NONE
	MOV	R0, #10H	Immediate	1	2	Data transfer	R0 is loaded with value 10	NONE
LOOP	CLR	A	Immediate	1	1	Logical	Clear value of A	NONE
	MOVC	A, @A+DPTR	Indexed	2	1	Data transfer	Move location to DPTR + @A	NONE
	ADD	A, R2	Register	1	1	Arithmet	Add R2 and A and	PARITY

						ic	store in A	
	DA	A	Immediate	1	1	Arithmet ic	Check for decimal adjust accumulator content for A	NONE
	JNC	NEXT	Indexed	2	2	Boolean	Jump to next if there is no carry	NONE
	INC	R3	Immediate	1	1	Arithmet ic	Increment R3	NONE
NEXT	INC	DPTR	immediate	2	1	arithmeti c	Increment DPTR	NONE
	MOV	R2, A	Register	1	1	Data transfer	Load the value of A into R2	NONE
	DZNZ	R0, LOOP	Indexed	2	2	Branchin g	Decrement and jump on non-zero	NONE
HERE	SJMP	HERE	Indexed	2	2	Branchin g	Short jump	NONE
	ORG	300H						NONE
	DB	01H, 9BH, ECH, 03H, 58H, 00H, 00H, 00H, 00H						NONE

Output: Registers containing the Result: R2 =E3, R3=01

Manual Calculation: converting all numbers to decimal

01 = 1, 9B = 155, EC = 236, 03 = 3, 58 = 88

$1+155+236+3+88 = 483$ in decimal = 1E3 in hexa decimal

Results and Observations

Program and registers before execution:

The screenshot shows the µVision IDE interface with the following details:

- Title Bar:** C:\study\FALL SEM/Micro controller lab\lab2\198EC035B\lab2_a.vproj - µVision
- Menu Bar:** File, Edit, View, Project, Flash, Debug, Peripherals, Tools, SVCS, Window, Help
- Toolbars:** Standard toolbar with icons for Open, Save, Print, etc.
- Registers Window:** Shows the state of various registers (R0-R7, Sys, SP, PC, etc.) with their addresses and values.
- Disassembly Window:** Displays the assembly code for **TASK1_5.BASM**. The code includes:
 - ORG 000H
 - MOV DPTR, #300H
 - MOV R0, #10H
 - LOOP: CLR A
 - MOVC A, @A+DPTR
 - ADD A, R2
 - ;DAA
 - JNC NEXT
 - INC R3
 - NEXT: INC DPTR
 - MOV R2, A
 - DJNZ R0, LOOP
 - HERE: SJMP HERE
 - ORG 300H
 - DB 01H, 9BH, 0ECH, 03H, 58H, 00H, 00H, 00H, 00H, 00H
 - END
- Memory 1 Window:** Shows the memory dump starting at address C300H. The data is mostly zeros with some specific values at higher addresses.
- Memory 2 Window:** Shows the memory dump starting at address D00H. It contains several FF and 01 hex values.
- Bottom Status Bar:** Simulation, t1: 0.00000000 sec, L7 C5, CAP NUM SCR L OVR R/W, 26°C AQI 93, 04:44 PM, 24-08-2021
- Taskbar:** Includes icons for File Explorer, Google Chrome, Microsoft Edge, File Explorer, Task View, Taskbar settings, and a search bar.

Program and registers after execution: FINAL STEP

Inferences:

1. About PSW VALUES – All are 0 except parity flag which is 1
 2. ABOUT THE OUTPUT VALUES IN REGISTERS – the data is taken and stored in R3 and R2.

Result: The 8051 ALP to perform ADDITION of 10 BCD NUMBERS is executed using Keil software and the results are verified manually.

-XXX-