


TASK 1 (** file has been converted to pdf because it exceeds 5Mb)

School of Electronics Engineering
VIT, Vellore



Registration No.	17BEC0656		
Student Name	SPARSH ARYA		
Course Code	ECE3003	Slot & Semester	L49+L50, FALL SEM 2019-20
Course Name	Microcontroller and its applications		
Program Title	TASK 1		
Date of Exp.	25-07-19	Date of Submission	10-08-19
Faculty	A.Karthikeyan		

Questions

1. Write and assemble a program to add the following data and then use the simulator to examine the CY flag.
DATA REG NO: Example 01 7B EC 02 75. 2. Five max 8BIT NUMBERS, 3. 5 8BIT Random numbers.
2. Write and assemble a program to load values into each of registers R0 - R4 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 and 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 a program 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 ;pick your REG NO AS data.
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.

Question 1.

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

Tools Required

Algorithm:

1. Move 5 numbers in 4 registers and one accumulator.

2. Add two numbers and store the result in accumulator, if a carry is generated store it in R7. Repeat the above process for 4 iteration

Part1

Memory Address	Label	Mnemonics	Operands	addressing mode used	Machine cycle Required	Memory Byte Required	Type of Instruction	Comments	Flags getting affected by the Instruction
-	-	MOV A,#01H	A	Immediate	1	1	Data Transfer	[A] 01	-
D0x00	-	MOV R0,#7BH	R0	Immediate	1	2	Data Transfer	[R0] 7B	-
-	-	ADD A,R0	A,R0	-	1	1	Arithmetic	[A] A+R0	-
-	-	JNC L1	-	-	2	2	Bit Manipulation	Jump if carry is NOT set	-
-	-	INC R7	-	-	1	1	Arithmetic	Increment	-
D0x01	-	L1: MOV R1,#0ECH	R1	Immediate	1	2	Data Transfer	[R1] EC	-
-	-	ADD A,R1	A,R1	-	1	1	Arithmetic	[A] A+R1	-

-	-	INC R7	-	-	1	1	Arithmetic	Increment	-
D0x02	-	L2: MOV R2,#06H	R2	Immediate	1	2	Data Transfer	[R2] 06	-
-	-	ADD A,R2	A,R2	-	1	1	Arithmetic	[A] A+R2	-
-	-	JNC L3	-	-	2	2	Bit Manipulation	Jump if carry is NOT set	-
-	-	INC R7	-	-	1	1	Arithmetic	Increment	-
D0x03	-	L3: MOV R3,#56H	-	-	1	1	Data Transfer	[R3] 56	-
-	-	ADD A,R3	A,R3	-	1	1	Arithmetic	[A] A+R3	-
-	-	JNC L4	-	-	2	2	Bit Manipulation	Jump if carry is NOT set	-
D0x07	-	INC R7	-	-	1	1	Arithmetic	Increment	-
-	-	L4: END	-	-	-	-	-	-	-

Part2

Memory Address	Label	Mnemonics	Operands	addressing mode used	Machine cycle Required	Memory Byte Required	Type of Instruction	Comments	Flags getting affected by the Instruction
-	-	MOV A,#0FFH	A	Immediate	1	1	Data Transfer	[A] FF	-
D:0x00	-	MOV R0,#0FFH	R0	Immediate	1	2	Data Transfer	[R0] FF	-
-	-	ADD A,R0	A,R0	-	1	1	Arithmetic	[A] A+R0	-
-	-	JNC L1	-	-	2	2	Bit Manipulation	Jump if carry is NOT set	-
-	-	INC R7	-	-	1	1	Arithmetic	Increment	-
D:0x01	-	L1: MOV R1,#0FFH	R1	Immediate	1	2	Data Transfer	[R1] FF	-
-	-	ADD A,R1	A,R1	-	1	1	Arithmetic	[A] A+R1	-
-	-	JNC L2	-	-	2	2	Bit Manipulation	Jump if carry is NOT set	-

-	-	INC R7	-	-	1	1	Arithmetic	Increment	-
D:0x02	-	L2: MOV R2,#0FFH	R2	Immediate	1	2	Data Transfer	[R2] FF	-
-	-	ADD A,R2	A,R2	-	1	1	Arithmetic	[A] A+R2	-
-	-	JNC L3	-	-	2	2	Bit Manipulation	Jump if carry is NOT set	-
-	-	INC R7	-	-	1	1	Arithmetic	Increment	-
D:0x03	-	L3: MOV R3,#0FFH	-	-	1	1	Data Transfer	[R3] FF	-
-	-	ADD A,R3	A,R3	-	1	1	Arithmetic	[A] A+R3	-
-	-	JNC L4	-	-	2	2	Bit Manipulation	Jump if carry is NOT set	-
-	-	INC R7	-	-	1	1	Arithmetic	Increment	-
-	-	L4: END	-	-	-	-	-	-	-

Part 3

Memory Address	Label	Mnemonics	Operands	addressing mode used	Machine cycle Required	Memory Byte Required	Type of Instruction	Comments	Flags getting affected by the Instruction
-	-	MOV A,#92H	A	Immediate	1	1	Data Transfer	[A] 92	-
D:0x00	-	MOV R0,#23H	R0	Immediate	1	2	Data Transfer	[R0] 23	-
-	-	ADD A,R0	A,R0	-	1	1	Arithmetic	[A] A+R0	-
-	-	JNC L1	-	-	2	2	Bit Manipulation	Jump if carry is NOT	-

								set	
-	-	INC R7	-	-	1	1	Arithmetic	Increment	-
D:0x01	-	L1: MOV R1,#66H	R1	Immediate	1	2	Data Transfer	[R1] 66	-
-	-	ADD A,R1	A,R1	-	1	1	Arithmetic	[A] A+R1	-
-	-	JNC L2	-	-	2	2	Bit Manipulation	Jump if carry is NOT set	-

-	-	INC R7	-	-	1	1	Arithmetic	Increment	-
D:0x02	-	L2: MOV R2,#87H	R2	Immediate	1	2	Data Transfer	[R2] 87	-
-	-	ADD A,R2	A,R2	-	1	1	Arithmetic	[A] A+R2	-
-	-	JNC L3	-	-	2	2	Bit Manipulation	Jump if carry is NOT set	-
-	-	INC R7	-	-	1	1	Arithmetic	Increment	-
D:0x03	-	L3: MOV R3,#0F5H	-	-	1	1	Data Transfer	[R3] F5	-
-	-	ADD A,R3	A,R3	-	1	1	Arithmetic	[A] A+R3	-
-	-	JNC L4	-	-	2	2	Bit Manipulation	Jump if carry is NOT set	-
-	-	INC R7	-	-	1	1	Arithmetic	Increment	-
-	-	L4: END	-	-	-	-	-	-	-

Output:

Part1

REGISTER OUTPUTS:

R0=7BH R1=ECH R2 = 06H R3=56H R4=R5=R6=0

R7=01H A =C5H SP = 07 PSW = 05H

• MANUAL OUTPUTS:

A – 01H – 0000 0001

+ R0 – 7BH - 0111 1011

A – 7CH – 0111 1100; R7 – 00H; CY=0; AC=0; P=1

+ R1 – ECH - 1110 1100

A – 68H – 0110 1000; R7 – 01H; CY=1; AC=1; P=1

+ R2 – 06H - 0000 110

	A – 6EH – 0110 1110;	R7 – 01H;	CY=0; AC=0; P=0
+	R3 – 56H - 0101 0110		
	A – C5H – 1100 0101;	R7 – 01H	CY=0; AC=0; P=1

Part2

REGISTER OUTPUTS:

R0=FFH	R1 =FFH	R2 = FFH	R3=FFH	R4 =R5 =R6 = 0
R7=04H	A =FBH	SP = 07	PSW = C1H	

- MANUAL OUTPUTS:

	A – FFH – 1111 1111		
+	R0 – FFH - 1111 1111		
	A – FEH – 1111 1110;	R7 – 01H;	CY=1; AC=1; P=1
+	R1 – FFH - 1111 1111		
	A – FDH – 1111 1101;	R7 – 02H;	CY=1; AC=1; P=1
+	R2 – FFH - 1111 1111		
	A – FCH – 1111 1100;	R7 – 03H;	CY=1; AC=1; P=0
+	R3 – FFH - 1111 1111		
	A – FBH – 1111 1011;	R7 – 04H	CY=1; AC=1; P=1

Part3

REGISTER OUTPUTS:

R0=23H	R1 =66H	R2 = 87H	R3=F5H	R4 =R5 =R6 =0
R7=02H	A =97H	SP = 07	PSW = 81H	

- MANUAL OUTPUTS:

	A – 92H – 1001 0010		
+	R0 – 23H - 0010 0011		
	A – B5H – 1011 0101;	R7 – 00H;	CY=0; AC=0; P=1
+	R1 – 66H - 0110 0110		
	A – 1BH – 0001 1011;	R7 – 01H;	CY=1; AC=0; P=0

+ R2 – 87H - 1000 0111

A – A2H – 1010 0010; R7 – 01H; CY=0; AC=1; P=1

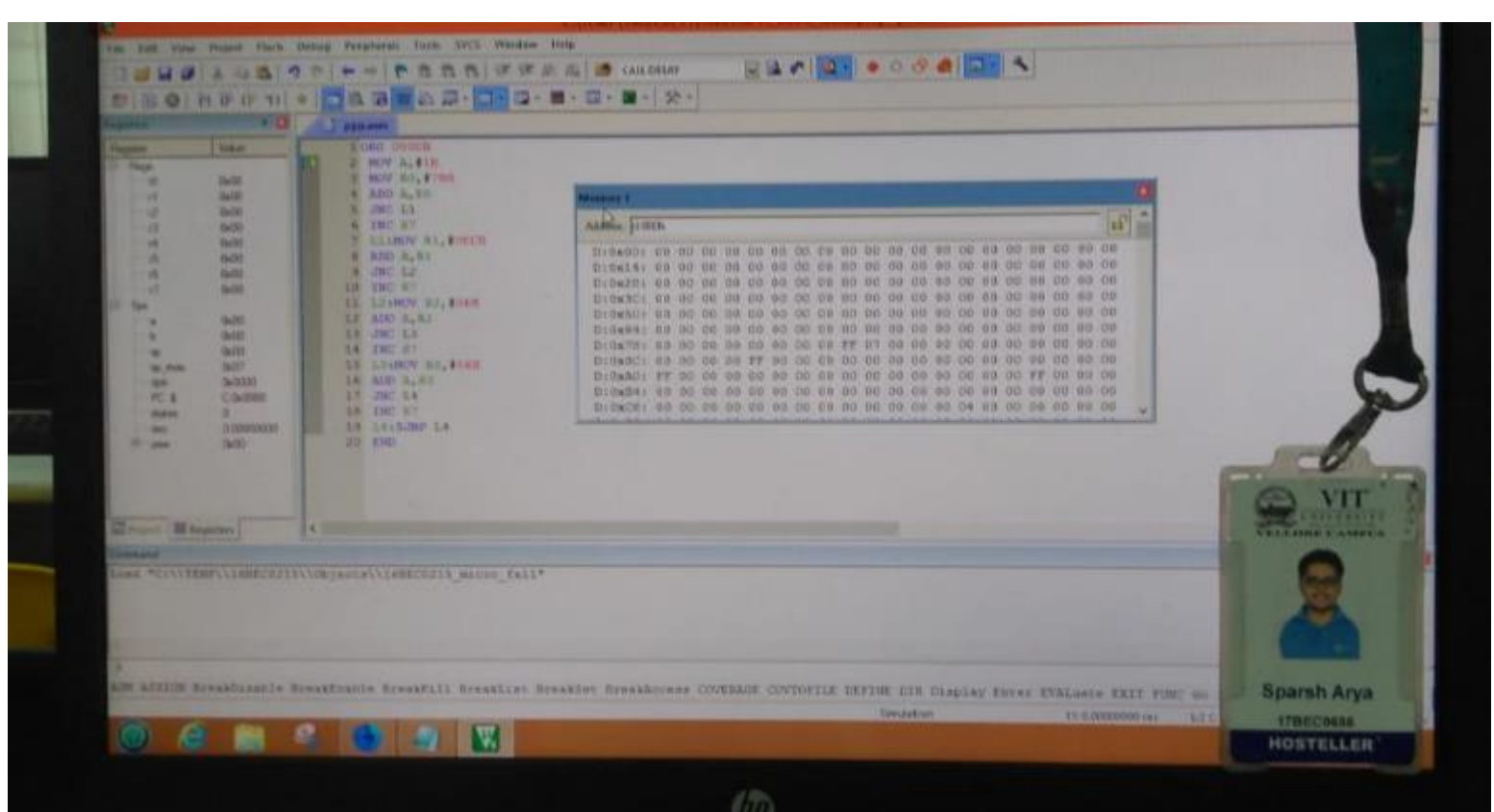
+ R3 – F5H - 1111 0101

A – 97H – 1001 0111; R7 – 02H CY=1; AC=0; P=1

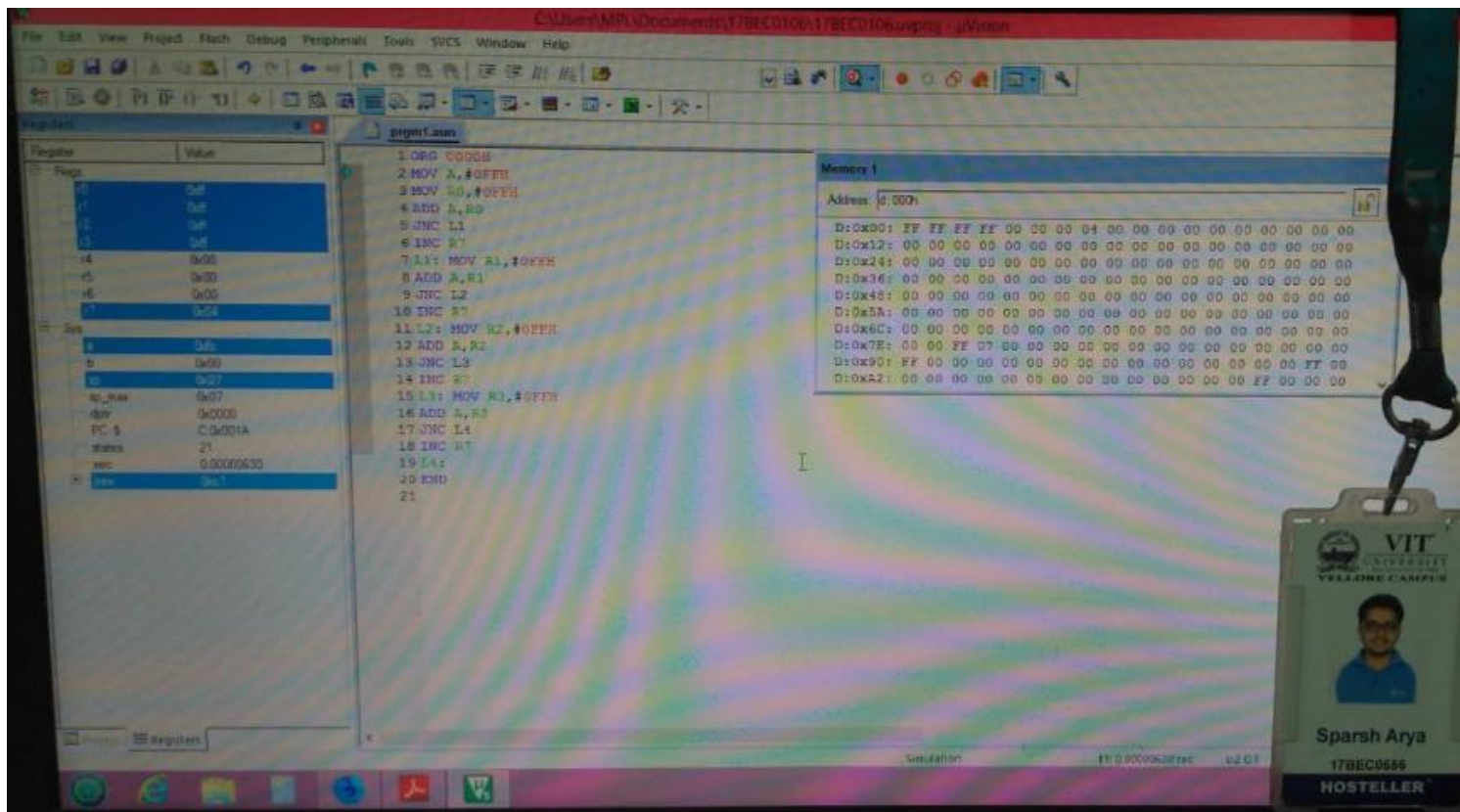
Results and Observations

Part 1

Print Screen of the Program and registers before execution:

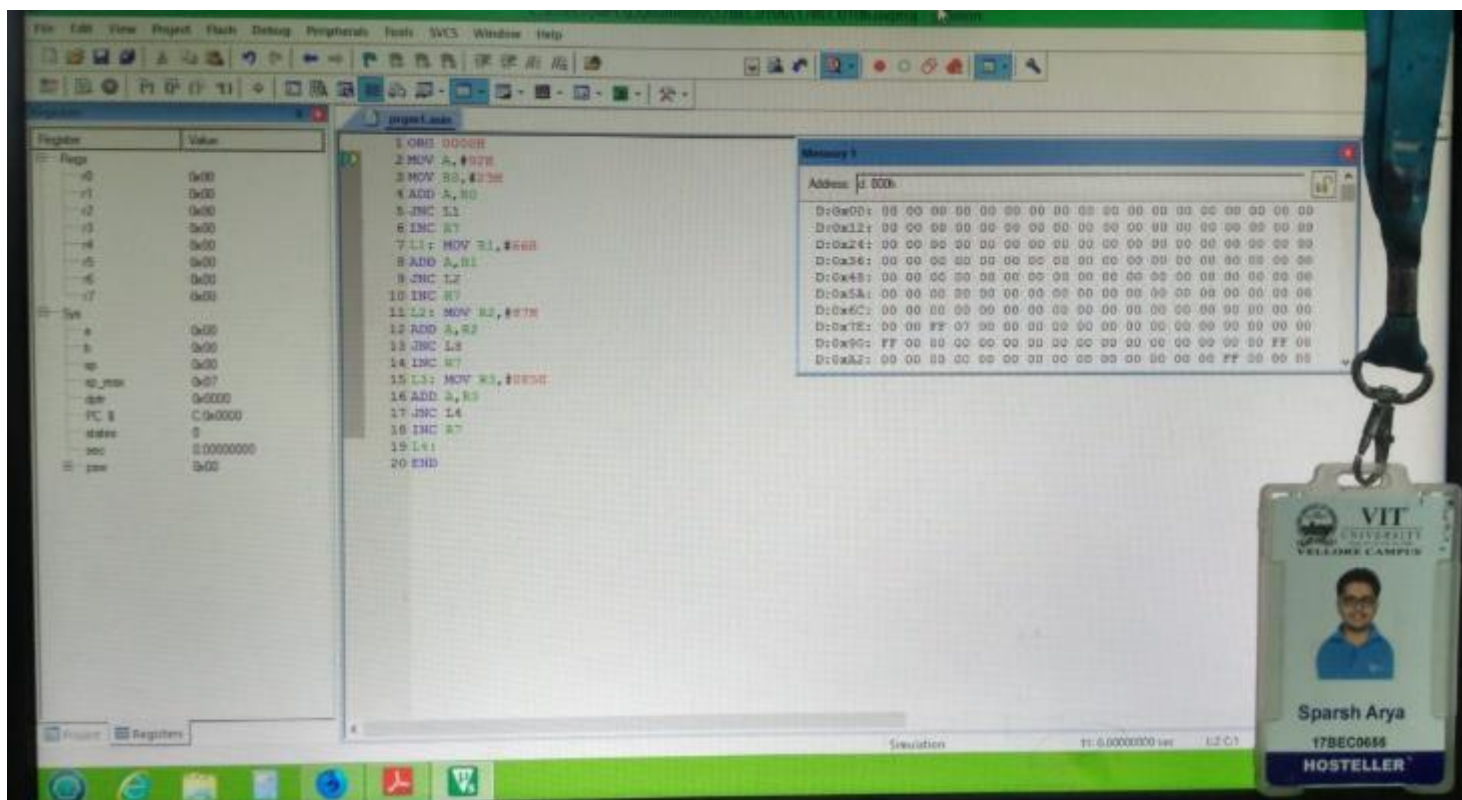


Print Screen of the Program and registers after execution:

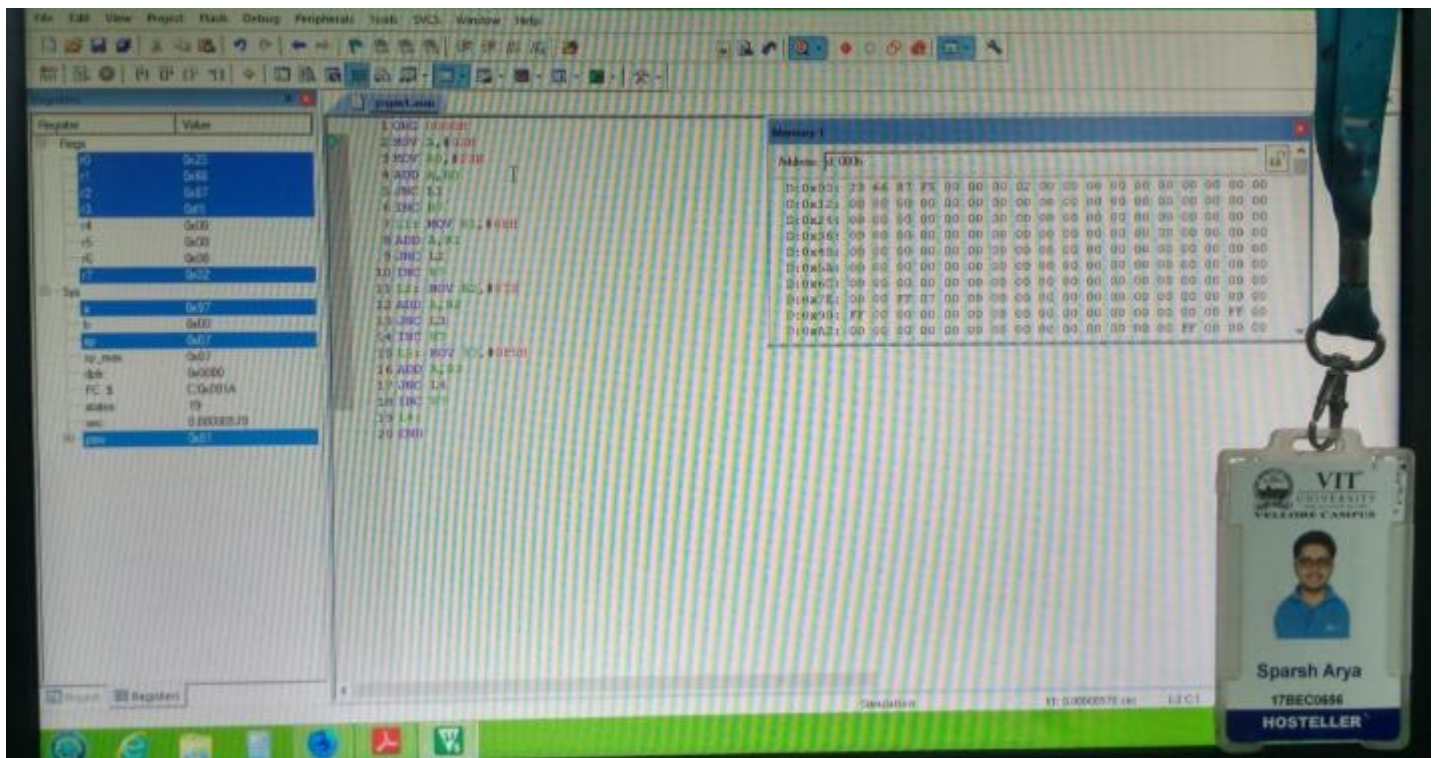


Part 3

Print Screen of the Program and registers before execution:



Print Screen of the Program and registers after execution:



Inferences:

Thus, 3 different sets of numbers have been added with successfully along with storage of carry in a separate location.

Result:

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

Question 2.

Aim: To write an 8051 ALP to perform push operation using Keil software and to verify the result manually.

Tools Required

Algorithm:

1. Store the numbers in the stack memory in general purpose registers.
2. Use the push command to push data onto the stack.
3. The stack pointer initially points to 07H and increments after every push operation.

Memory Address	Label	Mnemonics	Operands	addressing mode used	Machine cycle Required	Memory Byte Required	Type of Instruction	Comments	Flags getting affected by the Instruction
D:0x00	25	MOV R0,#25H	R0	Immediate	1	2	Data Transfer	-	-
D:0x01	35	MOV R1,#35H	R1	Immediate	1	2	Data Transfer	-	-
D:0x02	45	MOV R2,#45H	R2	Immediate	1	2	Data Transfer	-	-
D:0x03	55	MOV R3,#55H	R3	Immediate	1	2	Data Transfer	-	-
D:0x04	65	MOV R4,#65H	R4	Immediate	1	2	Data Transfer	-	-
D:0x08	25	PUSH 0	-	-	2	2	Data Transfer	-	-
D:0x09	35	PUSH 1	-	-	2	2	Data Transfer	-	-
D:0x0A	45	PUSH 2	-	-	2	2	Data Transfer	-	-

D:0x0 B	55	PUSH 3	-	-	2	2	Data Transfe r	-	-
D:0x0 C	65	PUSH 4	-	-	2	2	Data Transfe r	-	-
-	-	END	-	-	-	-	-	-	-

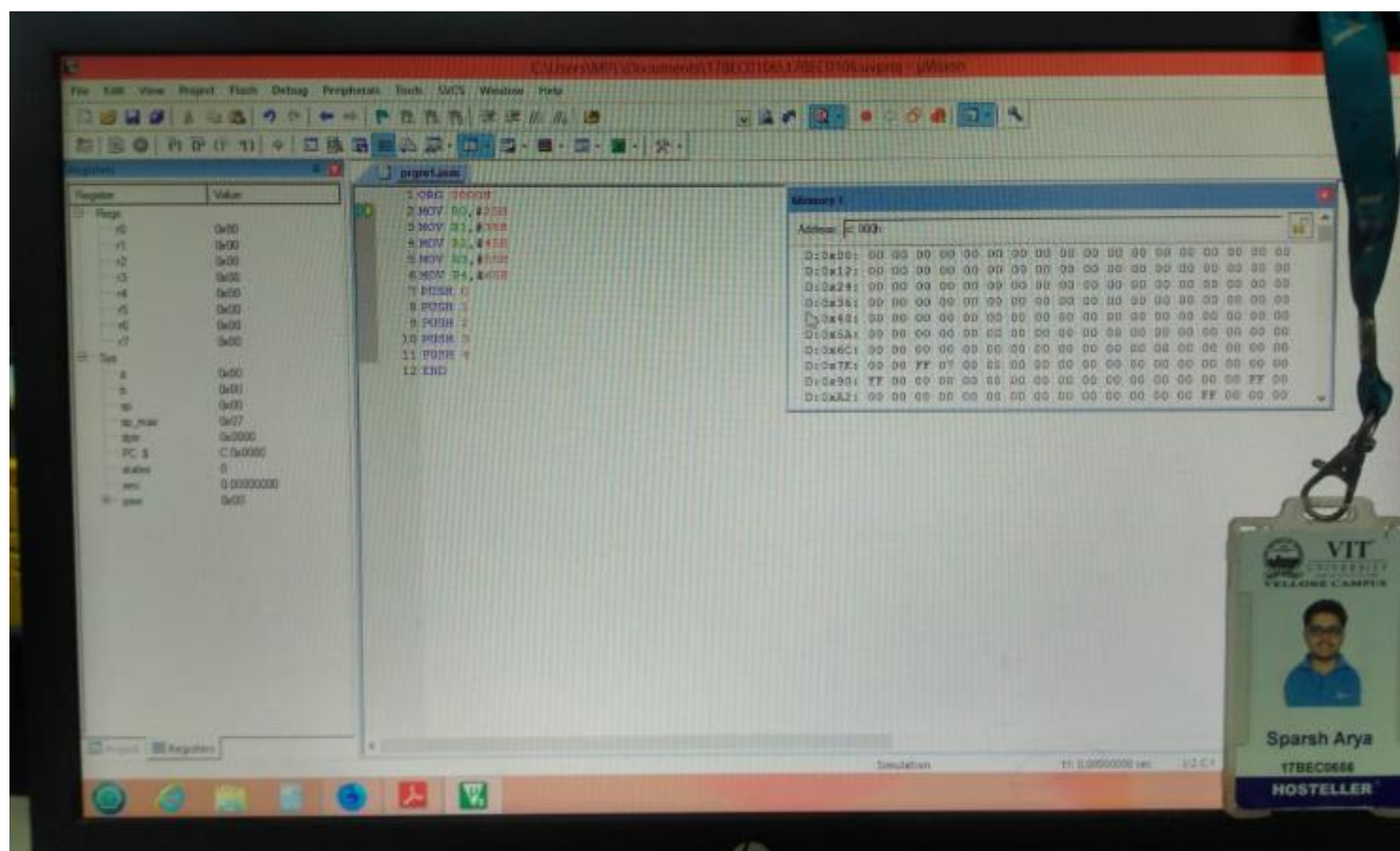
Output:

REGISTER OUTPUTS:

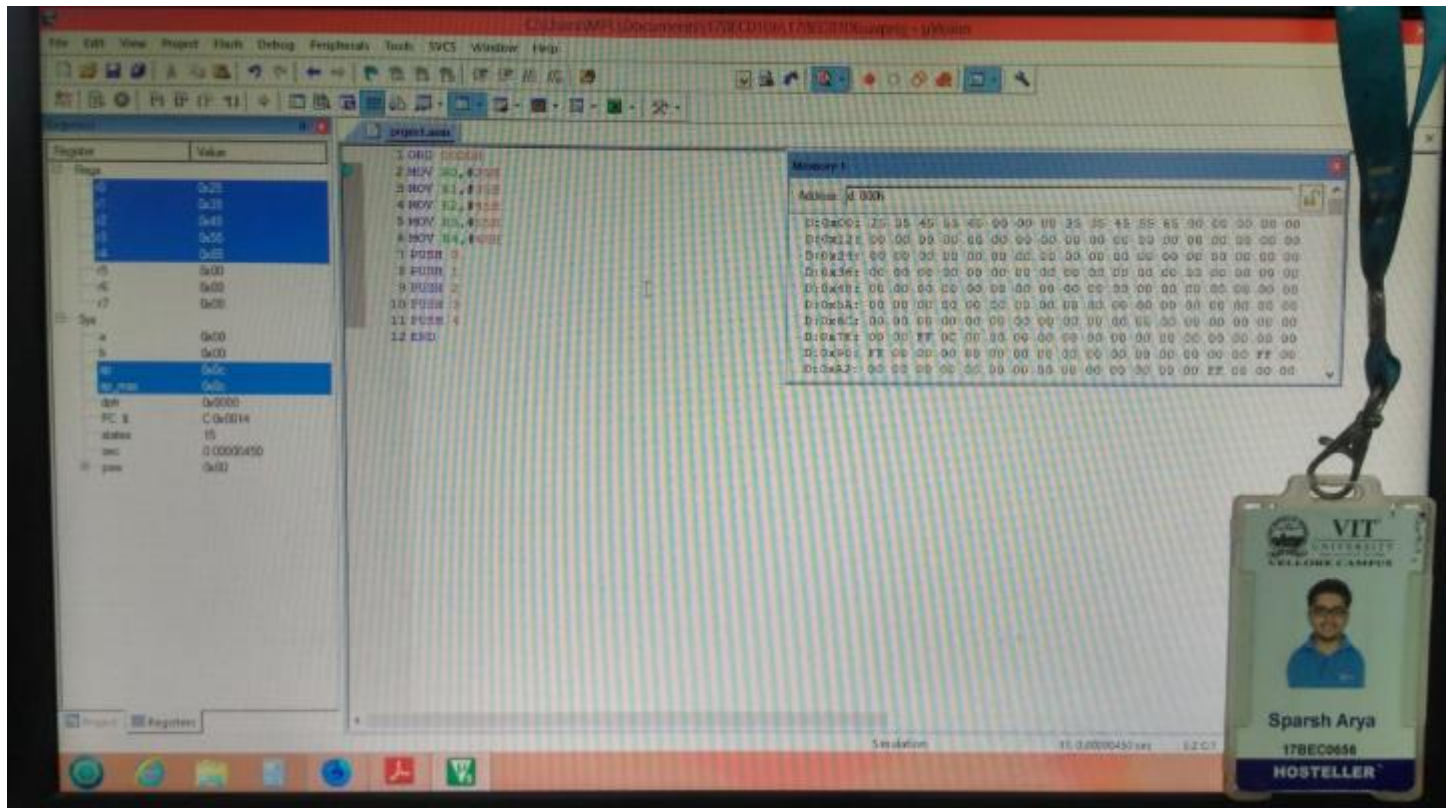
$$R_0 = 25H \quad R_1 = 35H \quad R_2 = 45H \quad R_3 = 55H \quad R_4 = 65H \quad R_5 = R_6 = R_7 = 0$$
$$SP = 0CH \quad SP_max = 0CH$$

Results and Observations

Print Screen of the Program and registers before execution:



Print Screen of the Program and registers after execution:



Inferences:

Thus, values have been pushed into the stack memory from general purpose registers using the push command.

Result:

The 8051 ALP to perform pushing of 5 8BIT NUMBERS onto the stack is executed using Keil software and the results are verified manually

Question 3.

Aim: To write an 8051 ALP to perform pop operation using Keil software and to verify the result manually.

Tools Required

Algorithm:

1. Store 5 values in from locations 08H to 0DH. These locations have been specifically allocated for stack operations
2. Then, pop the data from the stack using the pop instruction.
3. Display the values in the memory address.

Memory Address	Label	Mnemonics	Operands	addressing mode used	Machine cycle Required	Memory Byte Required	Type of Instruction	Comments	Flags getting affected by the Instruction
		MOV SP,#0DH	-	Immediate memory	2	3	Data Transfer		
D:0x08	10	MOV 08H, #10H	-	Immediate memory	2	3	Data Transfer		
D:0x09	11	MOV 09H, #11H	-	Immediate memory	2	3	Data Transfer		
D:0x0A	12	MOV 0AH, #12H	-	Immediate memory	2	3	Data Transfer		
D:0x0B	13	MOV 0BH, #13H	-	Immediate memory	2	3	Data Transfer		
D:0x0C	14	MOV 0CH, #14H	-	Immediate memory	2	3	Data Transfer		
D:0x0D	15	MOV 0DH, #15H	-	Immediate memory	2	3	Data Transfer		

							r		
D:0x00	15	POP 0	-	-	2	2	Data Transfe r		

14	POP 1	-	-	2	2	Data Transfe r		
13	POP 2	-	-	2	2	Data Transfe r		
12	POP 3	-	-	2	2	Data Transfe r		
11	POP 4	-	-	2	2	Data Transfe r		
10	POP 5	-	-	2	2	Data Transfe r		
-	END	-	-	-	-	-	-	-

Output:

REGISTER OUTPUTS:

R0 = 15H R1 = 14H R2 = 13H R3 = 12H R4 = 11H R5 = 10H

R6 = R7 = 0 SP = 07H SP_max = 0DH

Results and Observations

Print Screen of the Program and registers before execution:

Inferences:

Thus, values have been popped from stack memory using the stack operation.

Result:

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

Question 4.

Aim: To write an 8051 ALP to perform Push and pop operation using Keil software and to verify the result manually.

Tools Required


Algorithm:

1. Push 5 values into the stack.
2. Then, pop the data from the stack using the pop instruction.
3. Display the values in the memory address.

Memory Address	Label	Mnemonics	Operands	addressing mode used	Machine cycle Required	Memory Byte Required	Type of Instruction	Comments	Flags getting affected by the Instruction
C:0001	25	MOV R0,#25H	R0	Immediate	1	2	Data Transfer	-	-
C:0003	35	MOV R1,#35H	R1	Immediate	1	2	Data Transfer	-	-
C:0005	45	MOV R2,#45H	R2	Immediate	1	2	Data Transfer	-	-
C:0007	55	MOV R3,#55H	R3	Immediate	1	2	Data Transfer	-	-
C:0009	65	MOV R4,#65H	R4	Immediate	1	2	Data Transfer	-	-
D:0x08	25	PUSH 0	-	-	2	2	Data Transfer	-	-
D:0x09	35	PUSH 1	-	-	2	2	Data Transfer	-	-
D:0x0A	45	PUSH 2	-	-	2	2	Data Transfer	-	-

D:0x0B	55	PUSH 3	-	-	2	2	Data Transfer	-	-
D:0x0C	65	PUSH 4	-	-	2	2	Data Transfer	-	-
D:0x00	65	POP 0	-	-	2	2	Data Transfer	-	-
D:0x01	55	POP 1	-	-	2	2	Data Transfer	-	-
D:0x02	45	POP 2	-	-	2	2	Data Transfer	-	-
D:0x03	35	POP 3	-	-	2	2	Data Transfer	-	-
D:0x04	25	POP 4	-	-	2	2	Data Transfer	-	-
-	-	END	-	-	-	-	-	-	-

Output:

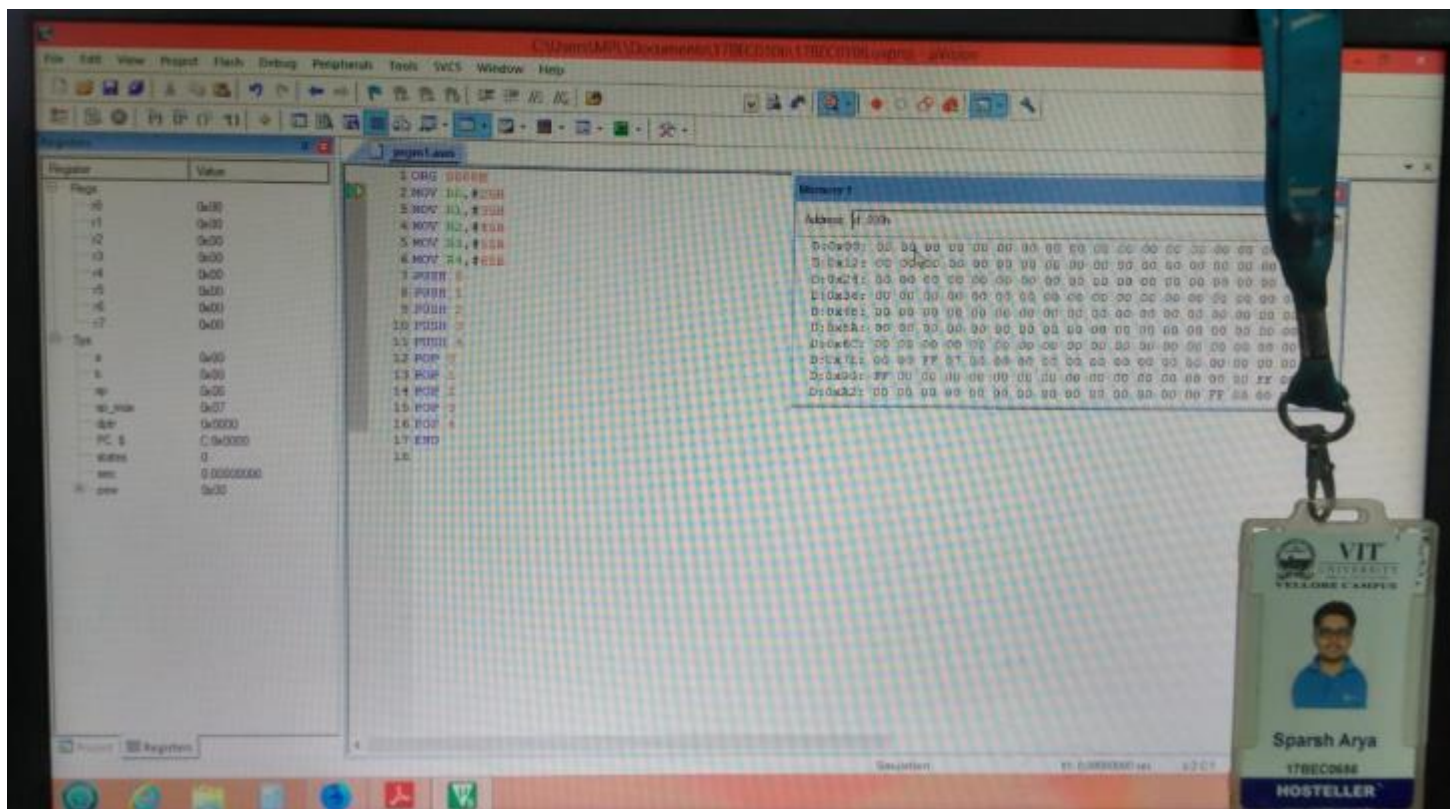

REGISTER OUTPUTS:

R0 = 65H
R1 =55H
R2 = 45H
R3 = 35H
R4 =25H
R5=R6=R7=0

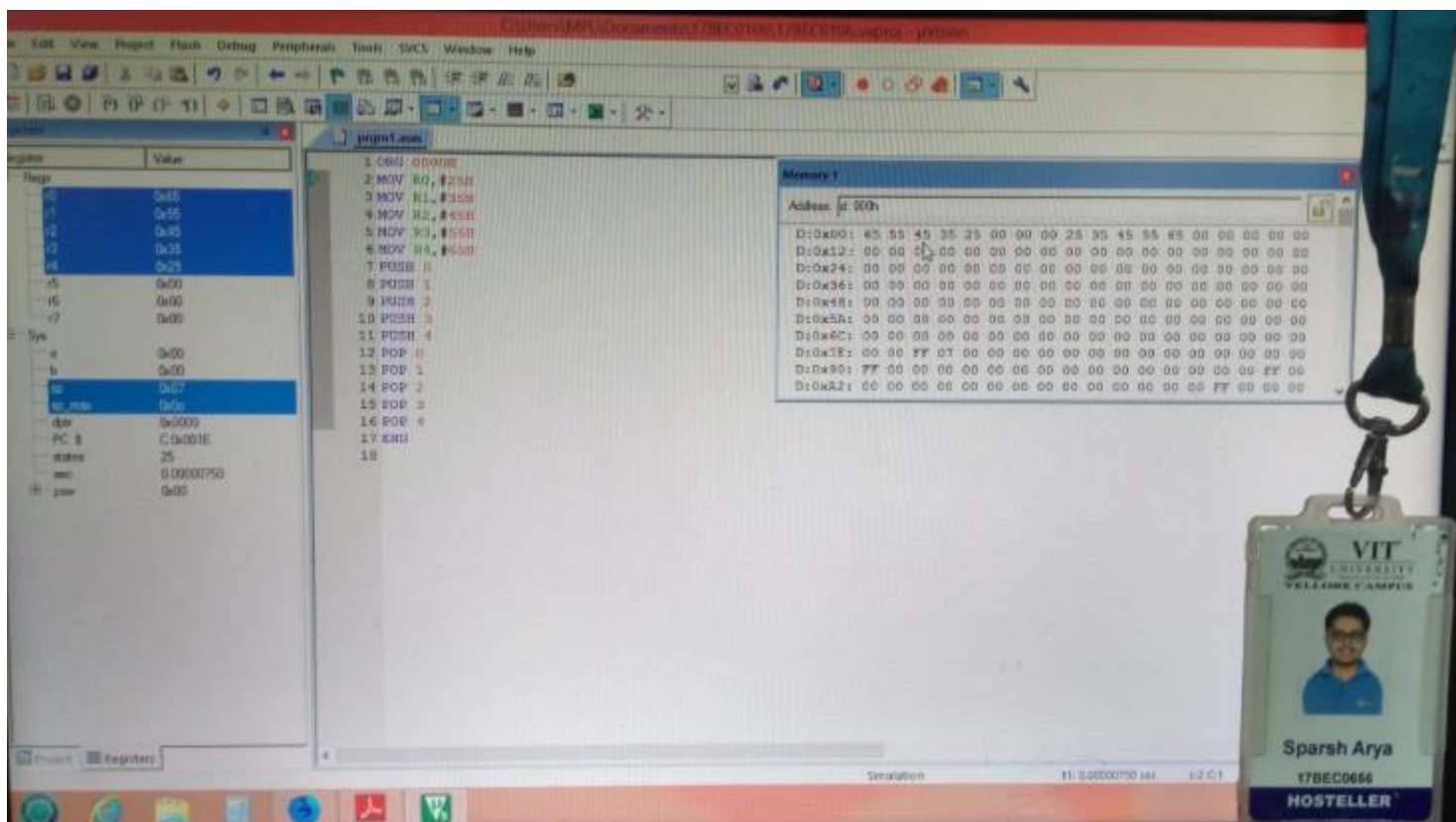
SP = 07H
SP_max = 0CH

Results and Observations

Print Screen of the Program and registers before execution:



Print Screen of the Program and registers after execution:



Inferences:

Thus, values have been pushed and popped from stack memory using the stack operation.

Result:

The 8051 ALP to perform pushing and popping of 5 8BIT NUMBERS is executed using Keil software and the results are verified manually

Question 5.

Aim: To write an 8051 ALP to perform addition of 10 BCD numbers using Keil software and to verify the result manually.

Tools Required

Algorithm:

1. Store 10 BCD values in DB.
2. Then, add the BCD numbers using the DAA operation.
3. Display the values in the memory address.

Memory Address	Label	Mnemonics	Operands	addressing mode used	Machine cycle Required	Memory Byte Required	Type of Instruction	Comments	Flags getting affected by the Instruction
-	-	ORG 0000H	-	-	-	-	-	A 0	-
	-	MOV DPTR,#300H	DPTR	-	2	3	Data Transfer	[DPTR] 300	-
	-	MOV R0,#10H	R0	Immediate	2	2	Data Transfer	[R0] 10	-
		LOOP: CLR A	A	-	1	1	Bit Manipulation	A 0	-
		MOVC A,@A+DPTR	A,DPTR	Indexed Memory	2	1	Data Transfer	[A] A + DPTR	-
		ADD A,R2	A,R2	-	1	1	Arithmetic	[A] A+R2	
		DA A	A	-	1	1	Arithmetic	Decimal adjustment A	-
		JNC NEXT	NEXT	-	2	2	Bit Manipulation	Jump if no	-

							ion	carry	
		INC R3	R3	-	1	1	Arithmetic	Increment	-
		NEXT:INC DPTR	DPTR	-	1	1	Arithmetic	Increment	-
		MOV R2,A	R2,A	Register	1	1	Data Transfer	[R2] [A]	-
		DJNZ R0,LOOP	R0	-	2	2	Branch	Decrement register, jump if not zero	-
		HERE: SJMP HERE	-	-	2	2	Branch	Short jump (+/- 127 bytes)	-
-	-	ORG 300H	-	-	-	-	-	C:300H	-
		DB 00H,01H,07H, 0BH,0EH,0CH, 00H, 06H, 56H, 02H	Database [300H – 30AH]	-	-	-	-	-	-
-	-	END	-	-	-	-	-	-	-

Output:

REGISTER OUTPUTS:

R0=R1=0 R2=51H R3=00H R4=R5=R6=R7=0 SP = 07H

SP_max = 07H

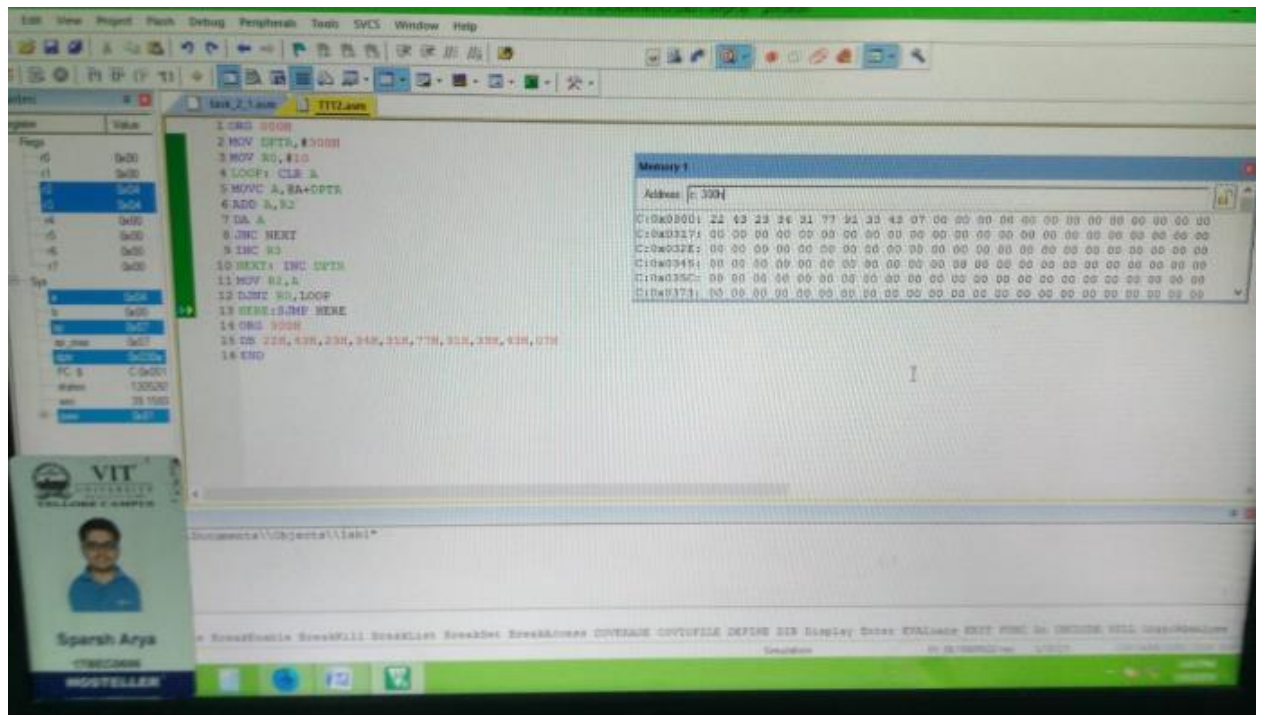


MANUAL OUTPUTS:

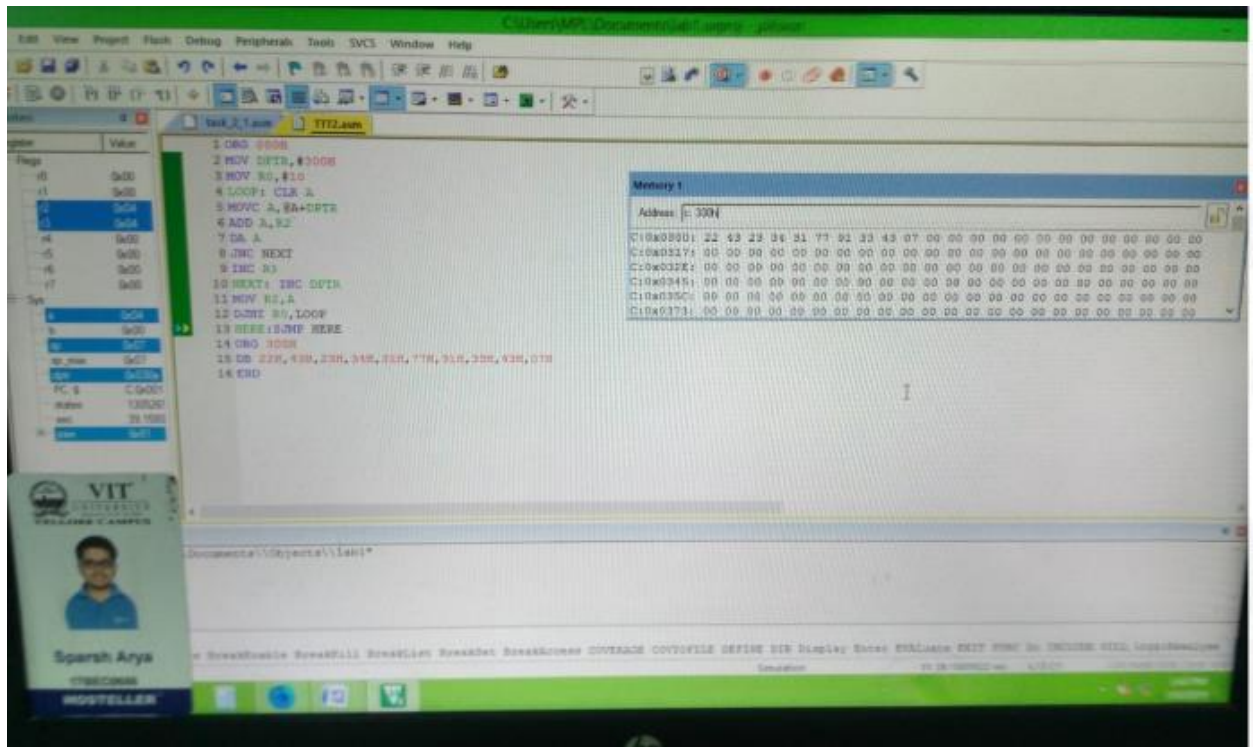
	00H	0000 0000		
+	01H	0000 0001		
A	01H	0000 0001;	CY=0; AC=0; P=1	
+	07H	0000 0111		
A	08H	0000 1000;	CY=0; AC=0; P=1	
+	0BH	0000 1011		
A	13H	0001 0011;	CY=0; AC=1; P=1	+0000 0110
A	19H	0001 1001;	CY=9; AC=0; P=1	
+	0EH	0000 1110		
A	21H	0010 0111;	CY=0; AC=1; P=0	+0000 0110
A	2DH	0010 1101;	CY=0; AC=0; P=0	
+	0CH	0000 1100		
A	39H	0011 1001;	CY=0; AC=1; P=0	+0000 0110
A	45H	0100 0101;	CY=0; AC=1; P=1	+0000 0110
A	4BH	0100 1011;	CY=0; AC=0; P=0	
+	00H	0000 0000		
A	4BH	0100 1011;	CY=0; AC=0; P=0	
+	02H	0000 0100		
A	4FH	0100 1111;	CY=0; AC=0; P=1	
+	09H	0000 0010		
A	51H	0101 0001;	CY=0; AC=1; P=0	
+	02H	0000 0000		
A	51H	0101 0001;	CY=0; AC=0; P=0	

Results and Observations

Print Screen of the Program and registers before execution:



Print Screen of the Program and registers after execution:



Inferences:

Thus, values have been added successfully

Result:

The 8051 ALP to perform adding of 10 8BIT NUMBERS is executed using Keil software and the results are verified manually

