Date: 21/02/2021

Student's name	Anuj Shah
Roll Number	18104B0024
Name of Professor	Professor Mohit Gujar

Experiment number	1
Experiment title	Add two 8-bit numbers
Hardware requirement	-
Software requirement	Keil uVision5

Aim	To write a program to add two 8-bit numbers.
Theory	An assembler program is made up of 3 elements: • Instructions: for example, MOV, ADD, SJMP • Assembler directives: for example, ORG, END • Assembler controls The 8051 uses 4 **addressing modes**: • Immediate addressing: MOV A,#10 (moves the decimal number 10 into register A), MOV R0, #0AH (moves the Hex number 0xA into register R0) • Register addressing: MOV A,R0 (copies the contents of register R0 to register A) • Direct addressing: MOV A,20H (copies the contents of address 0x20 to register A), MOV 30H,40H (copies the contents of address 0x40 to address 0x30), MOV P1,A (moves the contents of register A to port 1) • Indirect addressing: The most powerful addressing mode. MOV R0, #20H (moves the Hex number 0x20 into register R0), MOV @R0, #55H (moves the Hex number 0x55 to the address contained in R0, which is 0x20; R0 acts as a pointer to address 0x20), MOV A,@R0 (copy the contents of address 0x20 to register A)

8051 assembly	C language	Interpretation
MOV A, #57	A = 57	copy number 57 into Accumulator
MOV A, R0	A = R0	copy number stored in register R0 into Accumulator
MOV A, 42	-	copy number stored in memory location [42] into Accumulator
MOV A, @R1	A = *R1	copy number stored in memory location stored in register R1 into Accumulator
MOV R0, #57	R0 = 57	copy number 57 into register R0
MOV R0, A	R0 = A	copy number stored in Accumulator into register R0
MOV R0, 42	-	copy number stored in memory location [42] into register R0
MOV 42, A	-	copy number stored in Accumulator into memory location [42]
MOV 42, R0	-	copy number stored in register R0 into memory location [42]
MOV 42, 43	-	copy number stored in memory location [43] into memory location [42]
MOV 42, @R1	-	copy number stored in memory location stored in register R1 into memory location [42]
MOV 42, #57	-	copy number 57 into memory location [42]
MOV @R1, A	*R1 = A	copy number stored in Accumulator into memory location stored in register R1
MOV @R1, 42	-	copy number stored in memory location [42] into memory location stored in register R1
MOV @R1, #57	*R1 = 57	copy number 57 into memory location stored in register R1

Table 8: Addressing modes in 8051. The asterix (*) is the dereference operator in C (see my Overleaf article on Memory and Pointers). Notice that unlike assembly, C doesn't allow us to directly access specific memory locations, like [42] (see the Stack Overflow article on Why can't we use direct addressing in C or C++ code?).

Here are 2 useful directives in 8051:

- END: Last line of code. Assembler will not compile after this line.
- ORG: Origin directive. Sets the location counter address for the following instructions.

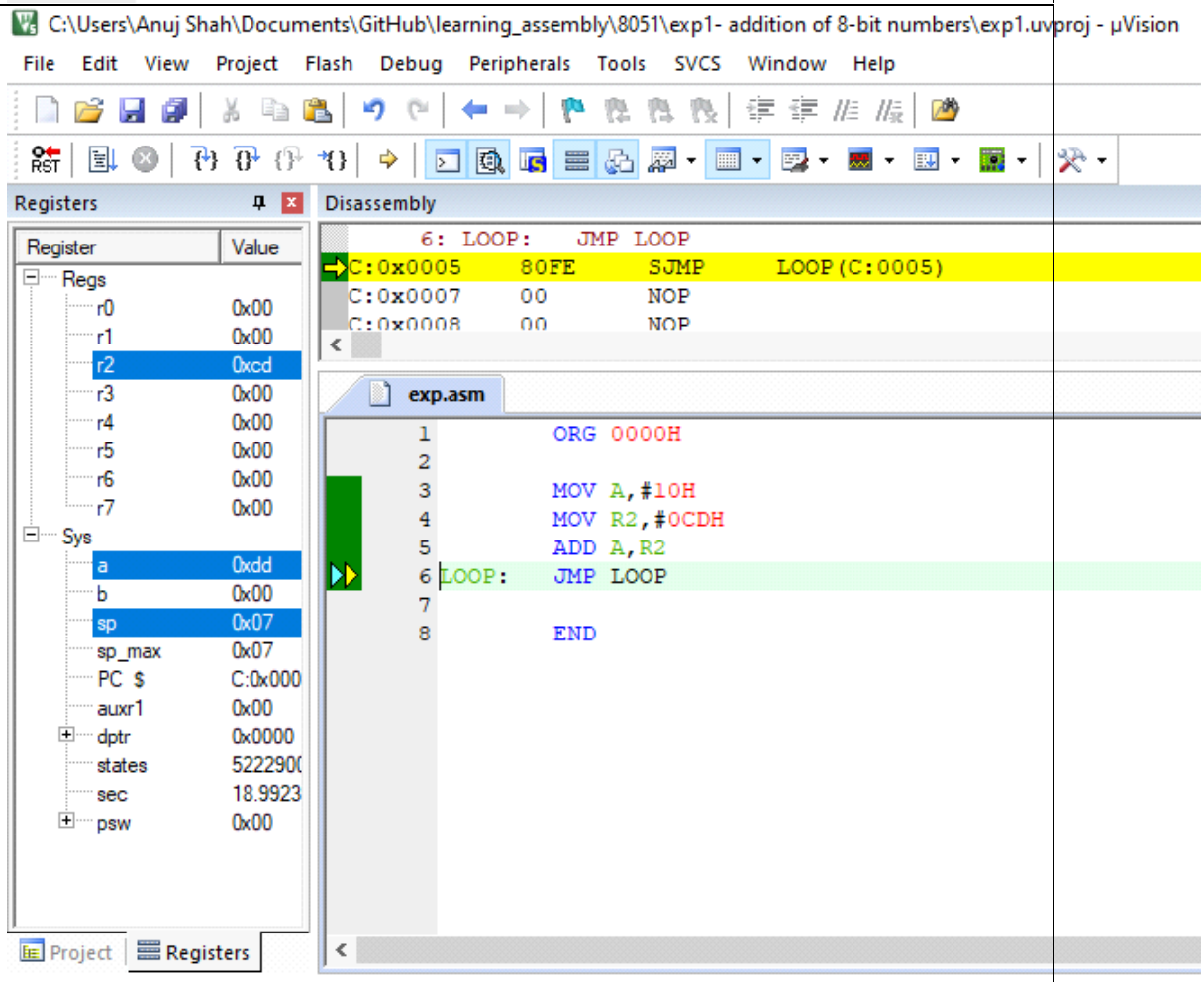
Program branching: Normal program execution is sequential. However, program branching instructions allow the programmer to alter the program execution sequence.

- Unconditional jump (JMP) : This instruction will automatically load the PC (program counter) with a new address, and will automatically jump to the instruction at that address.
- Conditional jumps: These instructions will only jump if a certain condition is true. They are similar to "If" statements in C.

Source:
http://www.polyengineeringtutor.com/8051%20Assembly%20Programming.pdf

Algorithm/ Flowchart

1. The directive "ORG 0000H" indicates the start of the program.
2. The command "MOV A,#10H" moves the hex number 10 (decimal = 16) to register A.
3. The command "MOV R2,#0CDH" moves the hex number CD (decimal = 205) to register B.
4. The command "ADD A,B" adds the value in register B with the value in register A, and stores the result in register A. In our program, the sum of 10h and CDh is DDh (decimal = 221).
5. The command "LOOP: SJMP LOOP" creates an infinite loop. This command is often included at the end of 8051 programs, because there is no other way to halt the microcontroller. https://stackoverflow.com/questions/43459467/why-do-some-8051-program-ends-with-the-code-loop-sjmp-loop
6. The directive "END" indicates the end of the program.

Program	exp.asm 1 ORG 0000H 2 3 MOV A,#10H 4 MOV R2,#0CDH 5 ADD A,R2 6 LOOP: JMP LOOP 7 8 END
Results/ Output	
Conclusion	We learned a lot of fundamentals of 8051 programming in this experiment: • ORG and END directives, used to start and stop a program. • Immediate addressing • JMP (unconditional jumps) and Infinite loops

Faculty Sign Grade received