Experiment No : 3

Aim : To Study i) 8051 data types & directives. ii) 8051 register banks & stack.

Theory :

i) 8051 data type and directives

Data Types

The 8051 microcontroller has only one data type. It is 8 bits, and the size of each register is also 8 bits. It is the job of the programmer to break down data larger than 8 bits (00 to FFH, or 0 to 255 in decimal) to be processed by the CPU. The data types used by the 8051 can be positive or negative.

Directives

DB (define byte)

The DB directive is the most widely used data directive in the assembler. It is used to define the 8-bit data. When DB is used to define data, the numbers can be in decimal, binary, hex, or ASCII formats. For decimal, the "D" after the decimal number is optional, but using "B" (binary) and "H" (hexadecimal) for the others is required. Regardless of which is used, the assembler will convert the numbers into hex. To indicate ASCII, simply place the characters in quotation marks ('like this'). The assembler will assign the ASCII code for the numbers or characters automatically. The DB directive is the only directive that can be used to define ASCII strings larger than two characters; therefore, it should be used for all ASCII data definitions.

```
          ORG  500H
DATA1:    DB   28                ;DECIMAL(1C in hex)
DATA2:    DB   00110101B         ;BINARY (35 in hex)
DATA3:    DB   39H               ;HEX
          ORG  510H
DATA4:    DB   "2591"            ;ASCII NUMBERS
          ORG  518H
DATA6:    DB   "My name is Joe"  ;ASCII CHARACTERS
```

Either single or double quotes can be used around ASCII strings. This can be useful for strings, which contain a single quote such as "O'Leary". DB is also used to allocate memory in byte-sized chunks.

Assembler directives

The following are some more widely used directives of the 8051.

ORG (origin)

The ORG directive is used to indicate the beginning of the address. The number that comes after ORG can be either in hex or in decimal. If the number is not followed by H, it is decimal and the assembler will convert it to hex. Some assemblers use ". ORG" (notice the dot) instead of "ORG" for the origin directive. Check your assembler.

EQU (equate)

This is used to define a constant without occupying a memory location. The EQU directive does not set aside storage for a data item but associates a constant value with a data label so that when the label appears in the program, its constant value will be substituted for the label. The following uses EQU for the counter constant and then the constant is used to load the R3 register.

```
COUNT     EQU  25
...       ....
MOV       R3,#COUNT
```

END directive

Another important pseudo code is the END directive. This indicates to the assembler the end of the source (asm) file. The END directive is the last line of an 8051 program, meaning that in the source code anything after the END directive is ignored by the assembler. Some assemblers use ". END" (notice the dot) instead of "END".

ii) 8051 Register Banks & Stack

The 128 bytes of RAM inside the 8051 are assigned the address 00 to 7FH. They can be accessed directly as memory locations and are divided into three different groups as follows –

- 32 bytes from 00H to 1FH locations are set aside for register banks and the stack.
- 16 bytes from 20H to 2FH locations are set aside for bit-addressable read/write memory.
- 80 bytes from 30H to 7FH locations are used for read and write storage; it is called as **scratch pad**. These 80 locations RAM are widely used for the purpose of storing data and parameters by 8051 programmers.

Register Banks in 8051

A total of 32 bytes of RAM are set aside for the register banks and the stack. These 32 bytes are divided into four register banks in which each bank has 8 registers, R0–R7. RAM locations from 0 to 7 are set aside for bank 0 of R0–R7 where R0 is RAM location 0, R1 is RAM location 1, R2 is location 2, and so on, until the memory location 7, which belongs to R7 of bank 0.

The second bank of registers R0–R7 starts at RAM location 08 and goes to locations OFH. The third bank of R0–R7 starts at memory location 10H and goes to location to 17H. Finally, RAM locations 18H to 1FH are set aside for the fourth bank of R0–R7.

Stack in the 8051

The stack is a section of a RAM used by the CPU to store information such as data or memory address on temporary basis. The CPU needs this storage area considering limited number of registers.

As the stack is a section of a RAM, there are registers inside the CPU to point to it. The register used to access the stack is known as the stack pointer register. The stack pointer in the 8051 is 8-bits wide, and it can take a value of 00 to FFH. When the 8051 is initialized, the SP register contains the value 07H. This means that the RAM location 08 is the first location used for the stack. The storing operation of a CPU register in the stack is known as a **PUSH**, and getting the contents from the stack back into a CPU register is called a **POP**.

Programs :

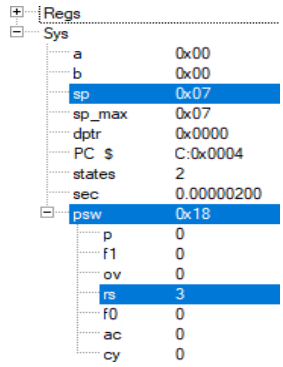
1. Select the register bank 3.

```
setb psw.3
setb psw.4
```

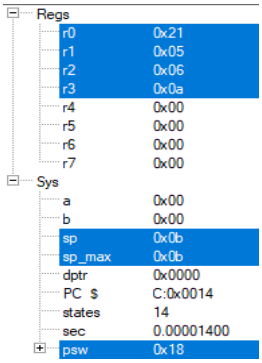
2. Push the values 21H, 05H, 06H and AH into the Stack.

```
setb psw.3
setb psw.4
mov r0,#21H
mov r1,#05H
mov r2,#06H
mov r3,#0Ah
push 0
push 1
push 2
push 3
```

Output :



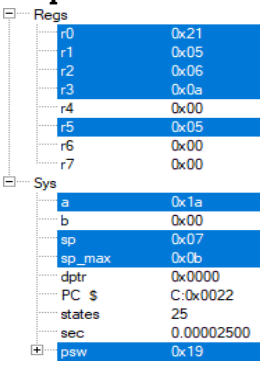
Output :



3. Pop the values from the stack.

```
setb psw.3
setb psw.4
mov r0,#21H
mov r1,#05H
mov r2,#06H
mov r3,#0AH
push 0
push 1
push 2
push 3
pop 0
pop 1
pop 2
pop 3
```

Output :

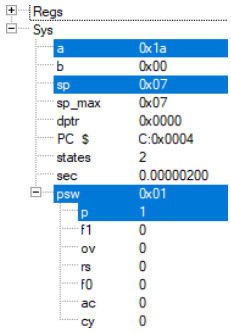


4. Add the following numbers :

a. AH and 10H

```
mov a,#0Ah
add a,#10h
```

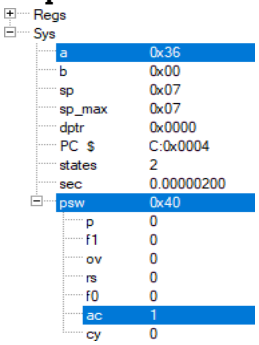
Output :



b. 2FH and 7H

```
mov a,#2Fh
add a,#7h
```

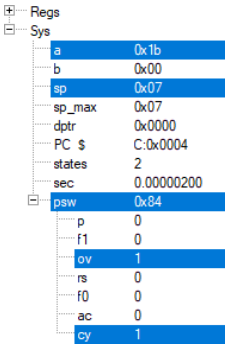
Output :



c. 88H and 93H

```
mov a,#88h
add a,#93h
```

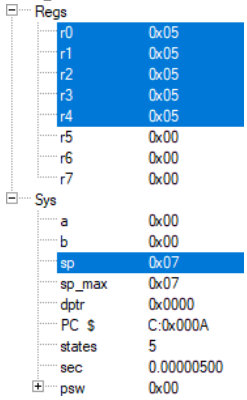
Output :



5. Use equ directive to store the value of 5H into 5 registers.

```
val equ 5h
mov r0,#val
mov r1,#val
mov r2,#val
mov r3,#val
mov r4,#val
```

Output :



Conclusion :

Programs to study 8051 Data type, Directives and Register Banks & Stack were successfully implemented.

Deepraj Bhosale

181105016