

8051 Assembly Language Programming [Part-2]

Bilal Habib
DCSE, UET

8051 DATA TYPES AND DIRECTIVES

Assembler Directives (cont')

❑ 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 and decimal
 - If the number is not followed by H, it is decimal and the assembler will convert it to hex

❑ END

- This indicates to the assembler the end of the source (asm) file
- The END directive is the last line of an 8051 program
 - Mean that in the code anything after the END directive is ignored by the assembler

8051 DATA TYPES AND DIRECTIVES

Assembler directives (cont')

- ❑ 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
 - When the label appears in the program, its constant value will be substituted for the label

8051 DATA TYPES AND DIRECTIVES

Assembler directives (cont')

❑ EQU (equate) (cont')

- Assume that there is a constant used in many different places in the program, and the programmer wants to change its value throughout
 - By the use of EQU, one can change it once and the assembler will change all of its occurrences

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

Use EQU for the
counter constant

The constant is used to
load the R3 register

FLAG BITS AND PSW REGISTER

Program Status Word

- ❑ The program status word (PSW) register, also referred to as the *flag register*, is an 8 bit register
 - Only 6 bits are used
 - These four are CY (*carry*), AC (*auxiliary carry*), P (*parity*), and OV (*overflow*)
 - They are called *conditional flags*, meaning that they indicate some conditions that resulted after an instruction was executed
 - The PSW3 and PSW4 are designed as RS0 and RS1, and are used to change the bank
 - The two unused bits are user-definable

FLAG BITS AND PSW REGISTER

Program Status Word (cont')

The result of signed number operation is too large, causing the high-order bit to overflow into the sign bit

CY	AC	F0	RS1	RS0	OV	--	P
----	----	----	-----	-----	----	----	---

CY PSW.7 Carry flag.

A carry from D3 to D4

AC PSW.6 Auxiliary carry flag.

Carry out from the d7 bit

-- PSW.5 Available to the user for general purpose

RS1 PSW.4 Register Bank selector bit 1.

RS0 PSW.3 Register Bank selector bit 0.

OV PSW.2 Overflow flag.

Reflect the number of 1s in register A

-- PSW.1 User definable bit.

P PSW.0 Parity flag. Set/cleared by hardware each instruction cycle to indicate an odd/even number of 1 bits in the accumulator.

RS1	RS0	Register Bank	Address
0	0	0	00H – 07H
0	1	1	08H – 0FH
1	0	2	10H – 17H
1	1	3	18H – 1FH

FLAG BITS AND PSW REGISTER

ADD Instruction And PSW (cont')

- ❑ The flag bits affected by the ADD instruction are CY, P, AC, and OV

Example 2-2

Show the status of the CY, AC and P flag after the addition of 38H and 2FH in the following instructions.

```
MOV A, #38H
```

```
ADD A, #2FH ;after the addition A=67H, CY=0
```

Solution:

38	00111000
+ 2F	00101111

FLAG BITS AND PSW REGISTER

ADD Instruction And PSW (cont')

- ❑ The flag bits affected by the ADD instruction are CY, P, AC, and OV

Example 2-2

Show the status of the CY, AC and P flag after the addition of 38H and 2FH in the following instructions.

```
MOV A, #38H
```

```
ADD A, #2FH ;after the addition A=67H, CY=0
```

Solution:

38	00111000
+ 2F	<u>00101111</u>
67	01100111

CY = 0 since there is no carry beyond the D7 bit

AC = 1 since there is a carry from the D3 to the D4 bi

P = 1 since the accumulator has an odd number of 1s (it has five 1s)

FLAG BITS AND PSW REGISTER

ADD
Instruction And
PSW
(cont')

Example 2-3

Show the status of the CY, AC and P flag after the addition of 9CH and 64H in the following instructions.

```
MOV A, #9CH
```

```
ADD A, #64H ;after the addition A=00H, CY=1
```

Solution:

9C	10011100
+ 64	<u>01100100</u>

FLAG BITS AND PSW REGISTER

ADD Instruction And PSW (cont')

Example 2-3

Show the status of the CY, AC and P flag after the addition of 9CH and 64H in the following instructions.

```
MOV A, #9CH
```

```
ADD A, #64H ;after the addition A=00H, CY=1
```

Solution:

9C	10011100
+ 64	<u>01100100</u>
100	00000000

CY = 1 since there is a carry beyond the D7 bit

AC = 1 since there is a carry from the D3 to the D4 bi

P = 0 since the accumulator has an even number of 1s (it has zero 1s)

FLAG BITS AND PSW REGISTER

ADD Instruction And PSW (cont')

Example 2-4

Show the status of the CY, AC and P flag after the addition of 88H and 93H in the following instructions.

```
MOV A, #88H
```

```
ADD A, #93H ;after the addition A=1BH, CY=1
```

Solution:

88	10001000
+ 93	<u>10010011</u>

FLAG BITS AND PSW REGISTER

ADD Instruction And PSW (cont')

Example 2-4

Show the status of the CY, AC and P flag after the addition of 88H and 93H in the following instructions.

```
MOV A, #88H
```

```
ADD A, #93H ;after the addition A=1BH, CY=1
```

Solution:

88	10001000
+ 93	<u>10010011</u>
11B	00011011

CY = 1 since there is a carry beyond the D7 bit

AC = 0 since there is no carry from the D3 to the D4 bi

P = 0 since the accumulator has an even number of 1s (it has four 1s)

REGISTER BANKS AND STACK

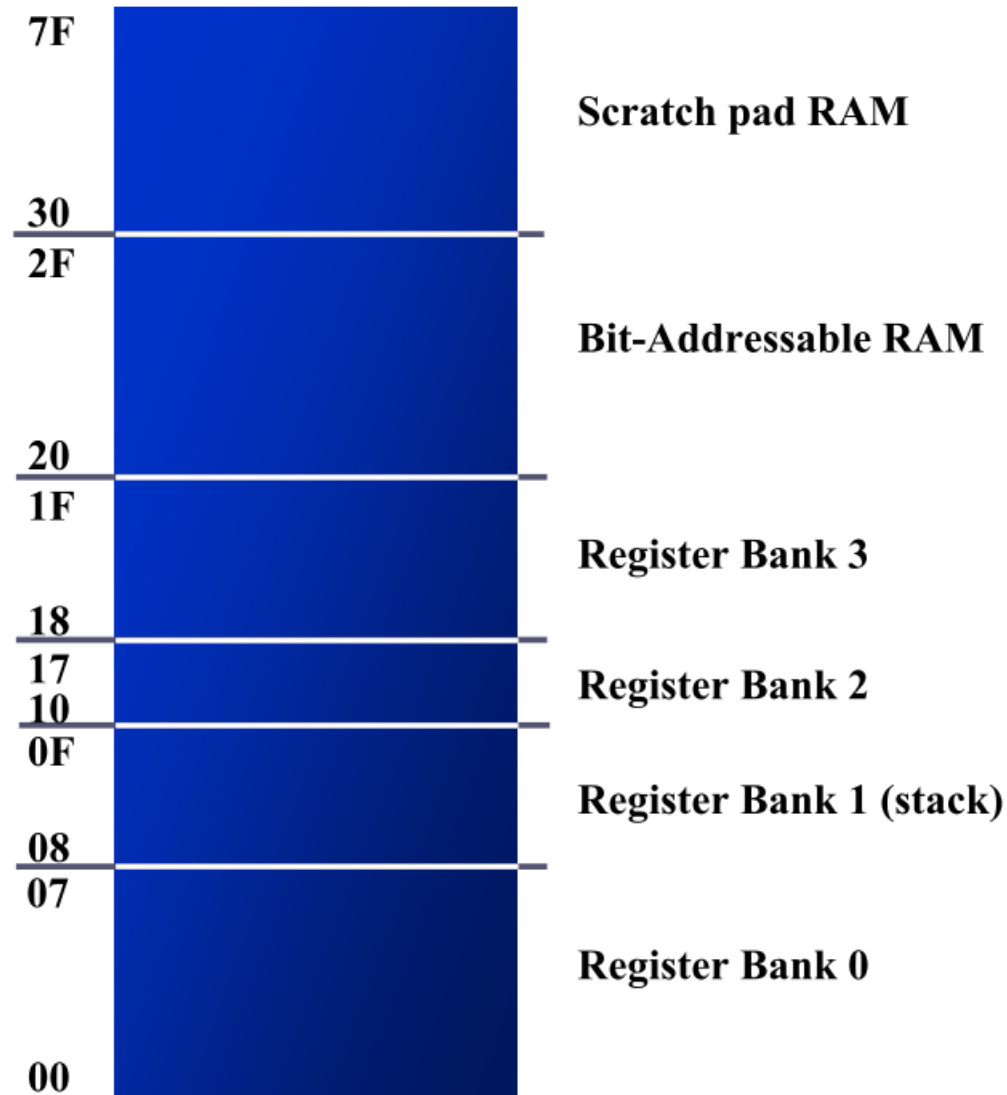
RAM Memory Space Allocation

- ❑ There are 128 bytes of RAM in the 8051
 - Assigned addresses 00 to 7FH
- ❑ The 128 bytes are divided into three different groups as follows:
 - 1) A total of 32 bytes from locations 00 to 1F hex are set aside for register banks and the stack
 - 2) A total of 16 bytes from locations 20H to 2FH are set aside for bit-addressable read/write memory
 - 3) A total of 80 bytes from locations 30H to 7FH are used for read and write storage, called *scratch pad*

8051 REGISTER BANKS AND STACK

RAM Memory Space Allocation (cont')

RAM Allocation in 8051



8051 REGISTER BANKS AND STACK

Register Banks

- ❑ These 32 bytes are divided into 4 banks of registers in which each bank has 8 registers, R0-R7
 - RAM location 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 RAM location 2, and so on, until memory location 7 which belongs to R7 of bank 0
 - It is much easier to refer to these RAM locations with names such as R0, R1, and so on, than by their memory locations
- ❑ Register bank 0 is the default when 8051 is powered up

8051 REGISTER BANKS AND STACK

Register Banks (cont')

Register banks and their RAM address

Bank 0		Bank 1		Bank 2		Bank 3	
7	R7	F	R7	17	R7	1F	R7
6	R6	E	R6	16	R6	1E	R6
5	R5	D	R5	15	R5	1D	R5
4	R4	C	R4	14	R4	1C	R4
3	R3	B	R3	13	R3	1B	R3
2	R2	A	R2	12	R2	1A	R2
1	R1	9	R1	11	R1	19	R1
0	R0	8	R0	10	R0	18	R0

8051 REGISTER BANKS AND STACK

Register Banks (cont')

- ❑ We can switch to other banks by use of the PSW register
 - Bits D4 and D3 of the PSW are used to select the desired register bank
 - Use the bit-addressable instructions SETB and CLR to access PSW.4 and PSW.3

PSW bank selection

	RS1(PSW.4)	RS0(PSW.3)
Bank 0	0	0
Bank 1	0	1
Bank 2	1	0
Bank 3	1	1

Micro-Processor Based System Design

Datatypes, Loops and masking

Bilal Habib

DCSE, UET Peshawar

Slides are adapted from Chung-Ping Young lectures

The unsigned int is a 16-bit data type

- Takes a value in the range of 0 to 65535 (0000 – FFFFH)
- Define 16-bit variables such as memory addresses
- Set counter values of more than 256
- Since registers and memory accesses are in 8-bit chunks, the misuse of int variables will result in a larger hex file

Signed int is a 16-bit data type

- Use the MSB D15 to represent – or +
- We have 15 bits for the magnitude of the number from –32768 to +32767

Write an 8051 C program to toggle bit D0 of the port P1 (P1.0) 50,000 times.

Solution:

```
#include <reg51.h>
sbit MYBIT=P1^0;
```

sbit keyword allows access to the single bits of the SFR registers

```
void main(void)
{
    unsigned int z;
    for (z=0; z<=50000; z++)
    {
        MYBIT=0;
        MYBIT=1;
    }
}
```

DATA TYPES

Bit and sfr

- ❑ The bit data type allows access to single bits of bit-addressable memory spaces 20 – 2FH
- ❑ To access the byte-size SFR registers, we use the sfr data type

Data Type	Size in Bits	Data Range/Usage
unsigned char	8-bit	0 to 255
(signed) char	8-bit	-128 to +127
unsigned int	16-bit	0 to 65535
(signed) int	16-bit	-32768 to +32767
sbit	1-bit	SFR bit-addressable only
bit	1-bit	RAM bit-addressable only
sfr	8-bit	RAM addresses 80 – FFH only

Delay using Loops

Write an 8051 C program to toggle bits of P1 continuously forever with some delay.

Solution:

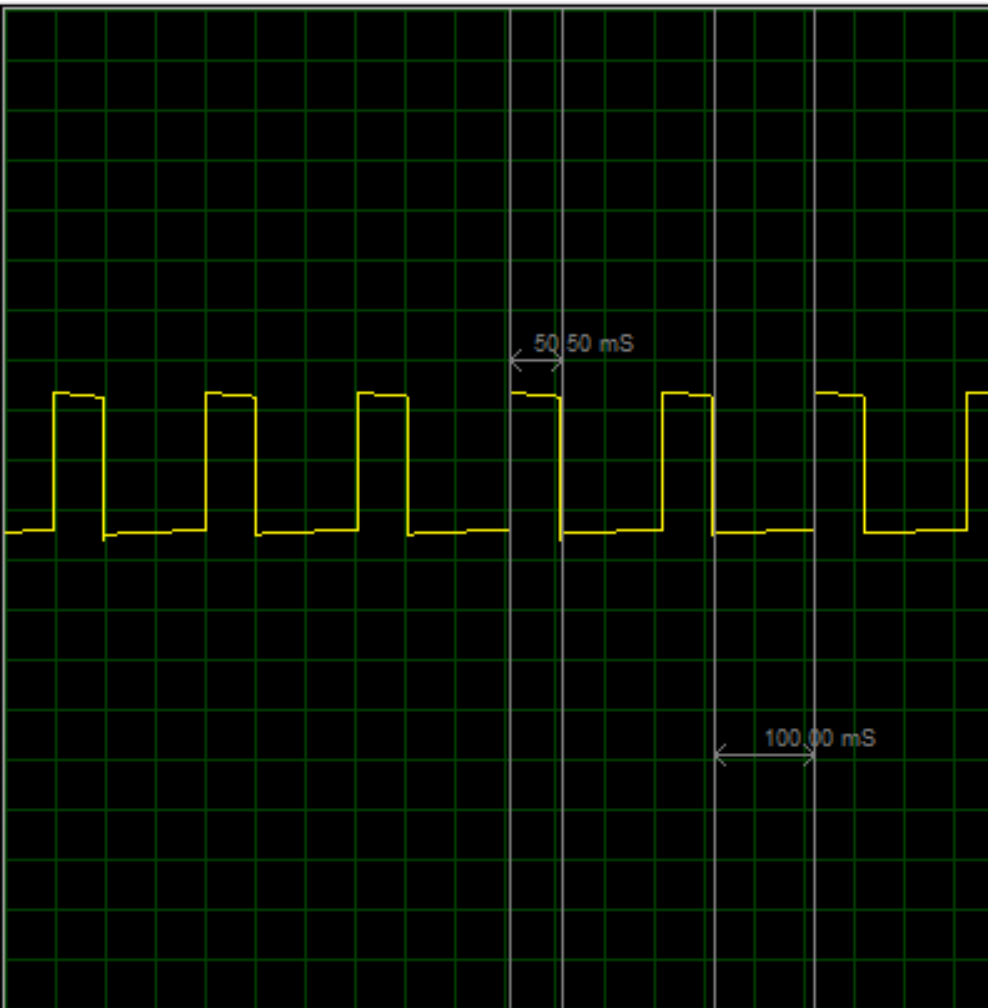
```
//Toggle P1 forever with some delay in between
//"on" and "off"
#include <reg51.h>
void main(void)
{
    unsigned int x;
    for (;;)                                //repeat forever
    {
        p1=0x55;
        for (x=0;x<40000;x++);             //delay size
                                            //unknown

        p1=0xAA;
        for (x=0;x<40000;x++);
    }
}
```

We must use the oscilloscope to measure the exact duration

Millisecond delay

Digital Oscilloscope



```
#include <reg51.h>
#include <stdio.h>
void MS_delay(unsigned int);
void main(void)
{
    while(1)
    {
        P1 = 0x01;           // 0000 0001...ON
        MS_delay(50);
        P1 = 0x00;           // 0000 0000...Off
        MS_delay(100);
    }
}

void MS_delay(unsigned int itime)
{
    unsigned int i,j;
    for (i=0;i<itime;i++)
        for (j=0;j<125;j++);
}
```

LOGIC OPERATIONS

Bit-wise Operators in C

- ❑ Logical operators
 - AND (&&), OR (||), and NOT (!)
- ❑ Bit-wise operators
 - AND (&), OR (|), EX-OR (^), Inverter (~), Shift Right (>>), and Shift Left (<<)
 - These operators are widely used in software engineering for embedded systems and control

Bit-wise Logic Operators for C

		AND	OR	EX-OR	Inverter
A	B	A&B	A B	A^B	~B
0	0	0	0	0	1
0	1	0	1	1	0
1	0	0	1	1	
1	1	1	1	0	

LOGIC OPERATIONS

Bit-wise Operators in C (cont')

Run the following program on your simulator and examine the results.

Solution:

```
#include <reg51.h>

void main(void)
{
    P0=0x35 & 0x0F;           //ANDing
    P1=0x04 | 0x68;           //ORing
    P2=0x54 ^ 0x78;           //XORing
    P0=~0x55;                 //inversing
    P1=0x9A >> 3;             //shifting right 3
    P2=0x77 >> 4;             //shifting right 4
    P0=0x6 << 4;              //shifting left 4
}
```

LOGIC OPERATIONS

Bit-wise Operators in C (cont')

Write an 8051 C program to get bit P1.0 and send it to P2.7 after inverting it.

Solution:

```
#include <reg51.h>
sbit inbit=P1^0;
sbit outbit=P2^7;
bit membit;

void main(void)
{
    while (1)
    {
        membit=inbit;    //get a bit from P1.0
        outbit=~membit;  //invert it and send
                        //it to P2.7
    }
}
```