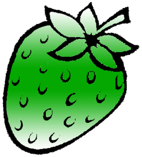
STRAWBERRY



/strawberrydevelopers

/strawberry_app

For more visit:

Strawberrydevelopers.weebly.com

8051 Microcontrollers

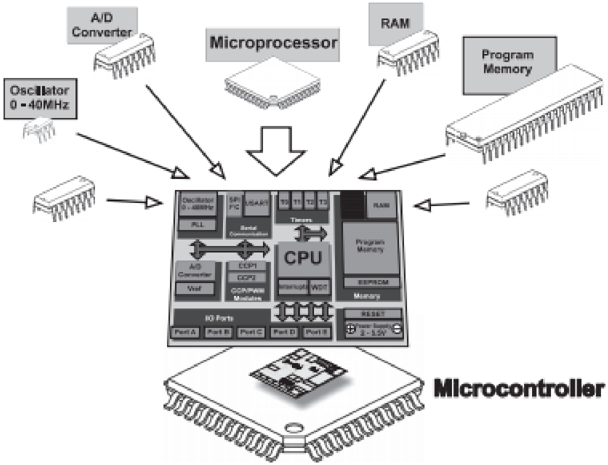
Richa Upadhyay Prabhu

NMIMS's MPSTME

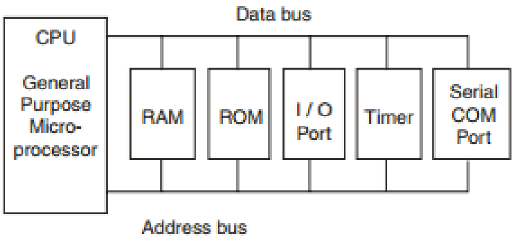
richa.upadhyay@nmims.edu

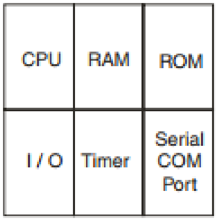
March 8, 2016

Controller vs Processor



Controller vs Processor





Introduction to 8051 Micro-controller

- In 1981,Intel corporation introduced an 8 bit microcontroller called 8051.
- 8 bit processor: CPU can only wwork on 8-bit data at a time
- 128 bytes of RAM
- 4K bytes of on-chip ROM
- 4 Input/Output ports, 8 bit wide

Also refereed as a "*system on a chip*"

Defining Micro-controller

Microcontroller may be called computer on chip since it has basic features of microprocessor with internal ROM, RAM, Parallel and serial ports within single chip. Or we can say microprocessor with memory and ports is called as microcontroller. This is widely used in washing machines, vcd player, microwave oven, robotics or in industries.

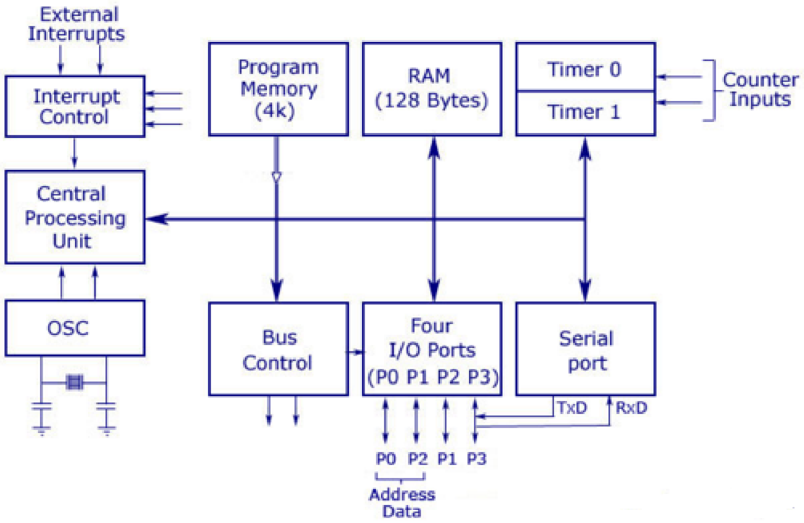
Key Features of 8051 Micro-controller

Feature	**Quantity**
ROM	4K bytes
RAM	128 bytes
Timer	2
I/O pins	32
Serial Port	1
Interrupt sources	6

8051 Family Members

Feature	**8051**	**8052**	**8031**
ROM	4K	8K	8K
RAM(bytes)	128	256	128
Timer	2	3	2
I/O pins	32	32	32
Serial Port	1	1	1
Interrupt sources	6	8	6

8051 block diagram



Inside 8051

128 Bytes of RAM for Data Storage

- RAM is volatile memory
- Used to store data during execution
- Normally Micro-controller has 256 bytes of RAM of which:
  - 128 bytes is available for user (Normal register banks and stack)
  - 128 bytes contains special function registers (SFR)
- 128 bytes means memory locations from 00H to FFH

Inside 8051

4 KB ROM

- Is available for program storage
- Used for permanent data storage or data which is not changed during processing
- Non-volatile memory
- can interface up to 64 KB ROM externally

Inside 8051

TIMERS and COUNTERS

- Timer : can give delay of particular time between some events
- Two hardware pins are available for delay generation but also can be done by programming
- These pins can be used for counting some external events
- Two special function timer registers are available : T0 and T1
- Both are 16 bit registers but can work on 8 bit also (TH0, TL0, TH1, TL1)
- TMOD and TCON registers are used for controlling timer operation.

Inside 8051

SERIAL PORT

- There are two pins available for serial communication TXD and RXD.
- Normally TXD is used for transmitting serial data which is in SBUF register, RXD is used for receiving the serial data.
- SCON register is used for controlling the operation
- There are four modes of serial communication

Inside 8051

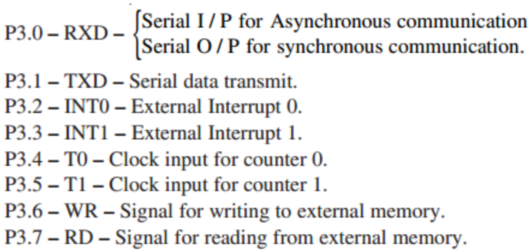
INPUT/OUTPUT PORTS

- There are four input output ports available P0, P1, P2, P3.
- Each port is 8 bit wide and has special function register P0, P1, P2, P3 which are bit addressable
- The port 0 can perform dual works. It is also used as Lower order address bus (A0 to A7) multiplexed with 8 bit data bus P0.0 to P0.7 is AD0 to AD7 respectively the address bus and data bus is demultiplex by the ALE signal and latch
- Port 2 can be used as I/O port as well as higher order address bus A8 to A15.

Inside 8051

INPUT/OUTPUT PORTS cont...

- Port 3 also have dual functions it can be worked as I/O as well as each pin of P3 has specific function.



P3.0 – RXD – { Serial I / P for Asynchronous communication
Serial O / P for synchronous communication.

P3.1 – TXD – Serial data transmit.
P3.2 – INT0 – External Interrupt 0.
P3.3 – INT1 – External Interrupt 1.
P3.4 – T0 – Clock input for counter 0.
P3.5 – T1 – Clock input for counter 1.
P3.6 – WR – Signal for writing to external memory.
P3.7 – RD – Signal for reading from external memory.

When external memory is interfaced with 8051 then P0 and P2 cant be worked as I/O port they works as address bus and data bus, otherwise they can be accessed as I/O ports.

Inside 8051

OSCILLATOR

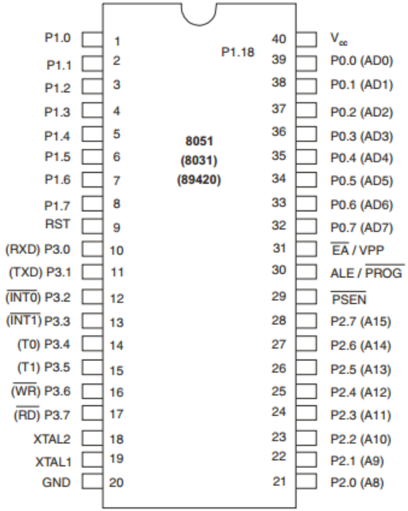
- It is used for providing the clock to 8051 which decides the speed or baud rate of Micro-controller.
- We use crystal which frequency vary from 4MHz to 30 MHz, normally we use 11.0592(or 12) MHz frequency.

Inside 8051

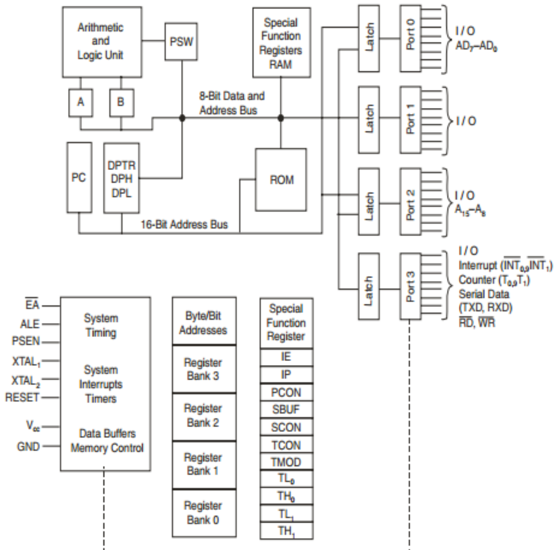
INTERRUPT

- Interrupts are defined as requests because they can be refused (masked)
- ISR are located at special memory locations in the memory
- INT0 and INT1 are the pins for external interrupts.

PIN DIAGRAM 8051



Architecture of 8051



Inside 8051

REGISTERS: 8 bit registers

Most widely used registers of 8051 are :

Accumulator (A) : used for all arithmetic and logic instructions; 8-bit

Register - B : 8 bit, bit-addressable register; used in only two instructions MUL AB and DIV AB

Registers R_0 to R_7 : all are 8-bit registers

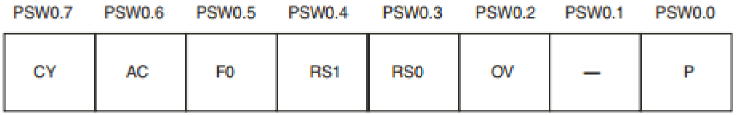
Program Counter(PC) : 16-bit register, points to address of next instruction to be executed

Data Pointer (DPTR) : 16-bit register, can be divided into two parts DPH and DPL

Inside 8051

8051 FLAG and PSW Register

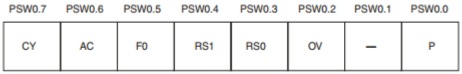
- Flag register in 8051 is called as program status word (PSW).
- Used to indicate the Arithmetic condition of ACC
- PSW is also bit addressable and 8 bit wide



PSW0.7	PSW0.6	PSW0.5	PSW0.4	PSW0.3	PSW0.2	PSW0.1	PSW0.0
CY	AC	F0	RS1	RS0	OV	—	P

FLAG Register

8051 FLAG



PSW0.7	PSW0.6	PSW0.5	PSW0.4	PSW0.3	PSW0.2	PSW0.1	PSW0.0
CY	AC	F0	RS1	RS0	OV	—	P

FLAG Register

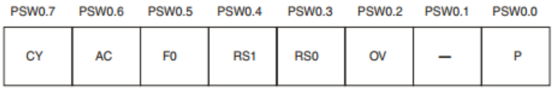
P⇒ Parity ⇒ PSW 0.0 : 1 if odd number of 1 in ACC and 0 if even number of 1 in ACC

OV⇒ Overflow ⇒ PSW 0.2 : this is used to detect error in signed arithmetic operation. This is similar to carry flag but difference is only that carry flag is used for unsigned operation.

F0 ⇒ PSW 0.5 : user definable bit

AC⇒ Auxillary Carry ⇒ PSW 0.6 : when carry is generated from D3 to D4, it is set to 1

8051 FLAG



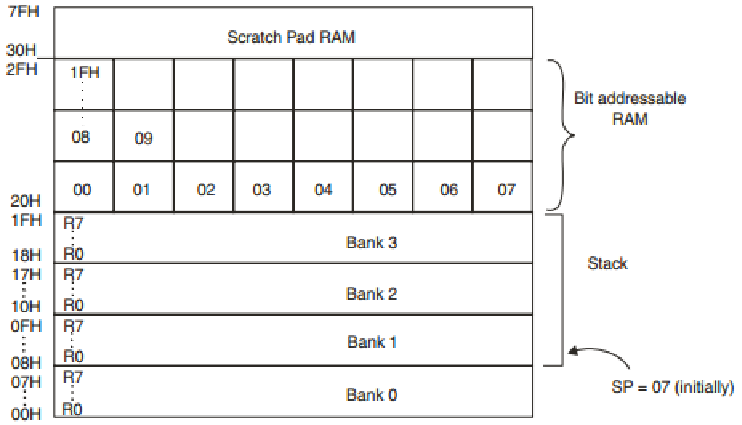
FLAG Register

CY$\Rightarrow$ Carry $\Rightarrow$ PSW 0.7 : Affected after 8 bit addition and subtraction. It is used to detect error in unsigned arithmetic operation.

RS1	**RS0**	**Register Bank select**
0	0	Bank 0
0	1	Bank 1
1	0	Bank 2
1	1	Bank 3

Structure of RAM

In 8051, 128 byte visible or user accessible RAM is available which is shown in figure. Extra 128B RAM which is not user accessible. 80H to FFH used for storage of SFR (special function register)



Structure of RAM

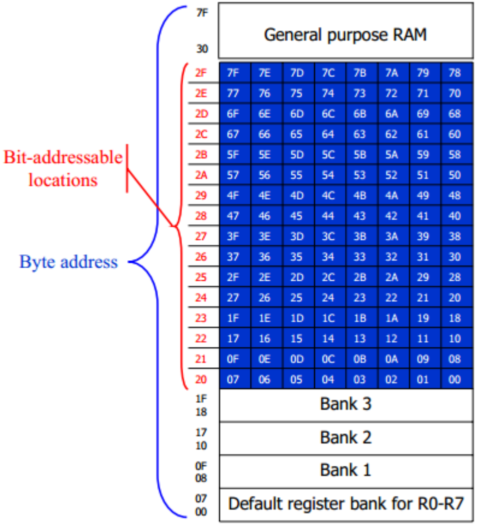
FOUR REGISTER BANKS

- each register bank there are eight 8 bit register available from R0 to R7
- By default Bank 0 is selected. For Bank 0, R0 has address 00H
- For selecting banks we use RS0 and RS1 bit of PSW.
- R0 to R7 registers are byte addressable.

Locations 20H to 2FH is bit addressable RAM means each bit from 00H to FFH in this we can set or reset CF rather than changing whole byte.

Locations 30H to 7FH is used as scratch pad means we can use this space for data reading and writing or for data storage

20H - 2FH : bit addressable RAM



7F 30	General purpose RAM							
2F	7F	7E	7D	7C	7B	7A	79	78
2E	77	76	75	74	73	72	71	70
2D	6F	6E	6D	6C	6B	6A	69	68
2C	67	66	65	64	63	62	61	60
2B	5F	5E	5D	5C	5B	5A	59	58
2A	57	56	55	54	53	52	51	50
29	4F	4E	4D	4C	4B	4A	49	48
28	47	46	45	44	43	42	41	40
27	3F	3E	3D	3C	3B	3A	39	38
26	37	36	35	34	33	32	31	30
25	2F	2E	2D	2C	2B	2A	29	28
24	27	26	25	24	23	22	21	20
23	1F	1E	1D	1C	1B	1A	19	18
22	17	16	15	14	13	12	11	10
21	0F	0E	0D	0C	0B	0A	09	08
20	07	06	05	04	03	02	01	00
1F 18	Bank 3							
17 10	Bank 2							
0F 08	Bank 1							
07 00	Default register bank for R0-R7							

20H - 2FH : bit addressable RAM

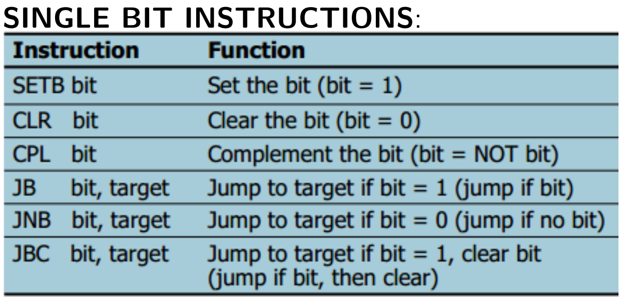
- Internal RAM locations 20 - 2FH are both byte-addressable and bit-addressable
- Bit address 00 - 7FH belong to RAM byte addresses 20 - 2FH

To avoid confusion regarding the addresses 00 - 7FH

- The 128 bytes of RAM have the byte addresses of 00 - 7FH can be accessed in byte size using various addressing modes
- The 16 bytes of RAM locations 20 - 2FH have bit address of 00 - 7FH. We can use only the single-bit instructions and these instructions use only direct addressing mode

20H - 2FH : bit addressable RAM

Instructions that are used for single-bit operations are as following



SINGLE BIT INSTRUCTIONS:

Instruction	Function
SETB bit	Set the bit (bit = 1)
CLR bit	Clear the bit (bit = 0)
CPL bit	Complement the bit (bit = NOT bit)
JB bit, target	Jump to target if bit = 1 (jump if bit)
JNB bit, target	Jump to target if bit = 0 (jump if no bit)
JBC bit, target	Jump to target if bit = 1, clear bit (jump if bit, then clear)

20H - 2FH : bit addressable RAM

Find out to which by each of the following bits belongs. Give the address of the RAM byte in hex

(a) `SETB 42H`, (b) `CLR 67H`, (c) `CLR 0FH`
(d) `SETB 28H`, (e) `CLR 12`, (f) `SETB 05`

Solution:

(a) D2 of RAM location 28H

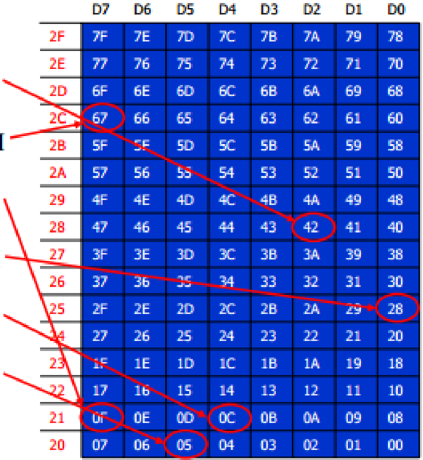
(b) D7 of RAM location 2CH

(c) D7 of RAM location 21H

(d) D0 of RAM location 25H

(e) D4 of RAM location 21H

(f) D5 of RAM location 20H



	D7	D6	D5	D4	D3	D2	D1	D0
2F	7F	7E	7D	7C	7B	7A	79	78
2E	77	76	75	74	73	72	71	70
2D	6F	6E	6D	6C	6B	6A	69	68
2C	67	66	65	64	63	62	61	60
2B	5F	5E	5D	5C	5B	5A	59	58
2A	57	56	55	54	53	52	51	50
29	4F	4E	4D	4C	4B	4A	49	48
28	47	46	45	44	43	42	41	40
27	3F	3E	3D	3C	3B	3A	39	38
26	37	36	35	34	33	32	31	30
25	2F	2E	2D	2C	2B	2A	29	28
24	27	26	25	24	23	22	21	20
23	1F	1E	1D	1C	1B	1A	19	18
22	17	16	15	14	13	12	11	10
21	0F	0E	0D	0C	0B	0A	09	08
20	07	06	05	04	03	02	01	00

STACK in 8051

STACK

- RAM locations from 08H to 1FH (24 bytes)can be used as stack. Stack is used to store the data temporarily.
- Stack is last in first out (LIFO)

STACK POINTER

- 8bit register, it points the top of stack
- Initially by default at 07H because first location of stack is 08H
- After each PUSH instruction the SP is incremented by one and after each POP instruction the SP is decremented.

Conflicting Register banks and Stack

- locations from 08H to 1FH is used as stack and it is also used as register bank
- If in the program, we use the Register Bank 1 to 3 and also use the stack then conflicts exist and error can be possible
- For removing this situation we use the stack from location 30H to 7FH by shifting SP to 2FH.
- MOV SP,#2FH;

8051 Assembly Language Programming

Structure of Assembly Language:

- In the early days of the computer, programmers coded in machine language, consisting of 0s and 1s
    - Tedious, slow and prone to error
- Assembly languages, which provided mnemonics for the machine code instructions, plus other features, were developed
- Assembly language is referred to as a lowlevel language. It deals directly with the internal structure of the CPU

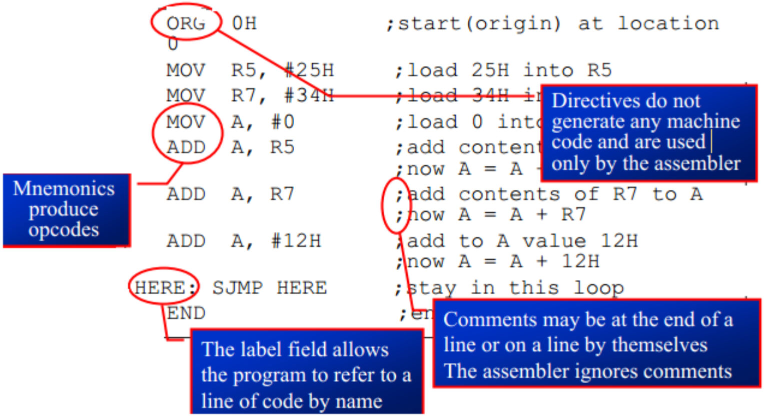
8051 Assembly Language Programming

Structure of Assembly Language:

- A given Assembly language program is a series of statements, or lines
  - Assembly language instructions : Tell the CPU what to do
  - Directives (or pseudo-instructions) : Give directions to the assembler
- Assembly language instruction consists of four fields:

[label:] Mnemonic [operands] [;comment]

8051 Assembly Language Programming



```
 ORG  0H              ;start(origin) at location
 0
 MOV  R5, #25H         ;load 25H into R5
 MOV  R7, #34H         ;load 34H i
 MOV  A, #0            ;load 0 int
 ADD  A, R5            ;add content
                       ;now A = A +
 ADD  A, R7            ;add contents of R7 to A
                       ;now A = A + R7
 ADD  A, #12H          ;add to A value 12H
                       ;now A = A + 12H
HERE: SJMP HERE        ;stay in this loop
 END                   ;en
```

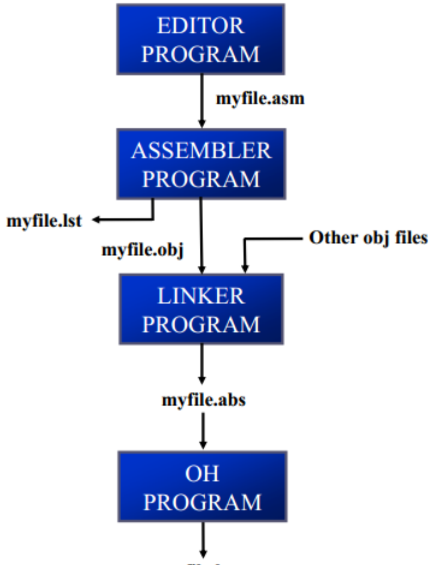
Directives do not generate any machine code and are used only by the assembler

Mnemonics produce opcodes

Comments may be at the end of a line or on a line by themselves
The assembler ignores comments

The label field allows the program to refer to a line of code by name

8051 Assembly Language Programming



8051 Assembly Language Programming

8051 DATATYPES: only one data type; 8 bit

DB: define byte : most widely used data type in the assembler. It is used to define 8 bit data.

- DATA1: DB 55 ; DECIMAL
- DATA2: DB 10100011B ; BINARY
- DATA3: DB "HELLO WORLD" ;ASCII CHARACTERS

8051 Assembly Language Programming

ASSEMBLER DIRECTIVES:

ORG : origin or starting address of program

- 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

END : Termination of program

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

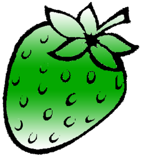
8051 Assembly Language Programming

ASSEMBLER DIRECTIVES:

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

STRAWBERRY



/strawberrydevelopers

/strawberry_app

For more visit:

Strawberrydevelopers.weebly.com