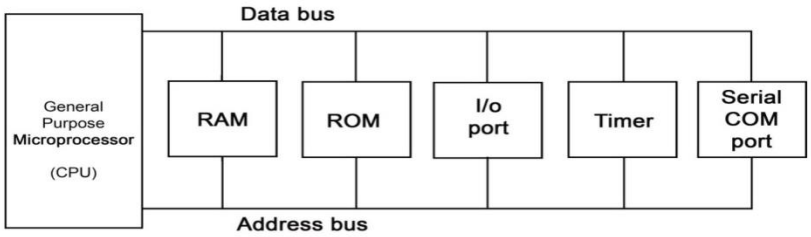
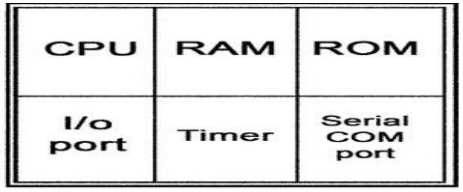
Microcontrollers

Introduction

Microprocessor



Microcontroller



Microprocessor	Microcontroller
It has just processor. Memory and I/O components has to be connected externally.	It has processor along with internal memory and I/O components.
Cost of entire system is low.	Cost of entire system increases.
Cannot be used in compact systems and hence inefficient.	Can be used in compact systems and hence it is an efficient technique.
Due to external components, the entire power consumption is high. Hence it is not suitable to be used with devices running on stored power like batteries.	Since external components are low, total power consumption is less and can be used with devices running on stored power like batteries.

Introduction Contd…

Microprocessor	Microcontroller
It has less number of registers, hence more operations are memory based.	It has more number of registers, hence the programs are easier to write.
It is based on Von Neumann Architecture.	It is based on Harvard Architecture.
It is mainly used in Personal Computer.	It is mainly used in washing machine, mp3 players.
Since memory and i/o components are all external, each instruction will need external operation, hence it is relatively slower.	Since components are internal, most of the operations are internal instructions, hence the speed is fast.
Most of the processors do not have power saving features.	Most of the microcontroller have power saving modes idle mode and power saving mode. This helps to reduce power consumption even further.

Intel Microcontroller Family

- 1976 - 1st Microcontroller by Intel 8048.
- 1980 - 8051, 8052, 8031
- 8051 is a Harvard architecture, CISC instruction set, single chip microcontroller which was developed by Intel in 1980 for the use of Embedded System.
- It has 128 bytes of RAM(Data Memory), 4K bytes of on-chip ROM(Program Memory), 2 timers, one serial port, and 4 ports(each 8-bit wide) all on a single chip.
- 8-bit processor employs that cpu can work with only 8-bit of data at a time.
- Larger data has to be broken into 8-bit pieces to be processed by the CPU.
- It has 4 I/O ports, each 8-bit wide.

Intel Microcontroller Family

8031

- ROM less 8051.
- To use 8031 we should add external ROM to it .
- Maximum size of ROM is 64 Kb.
- Adding of external ROM to 8031 we loose two ports and leave only two ports for I/O operations.

8052

All the standard features of 8051 plus 128 bytes of RAM plus an extra timer which means

- 256 bytes of RAM
- 3 – timer
- 8 – kb of ROM
- 8 – interrupts sources

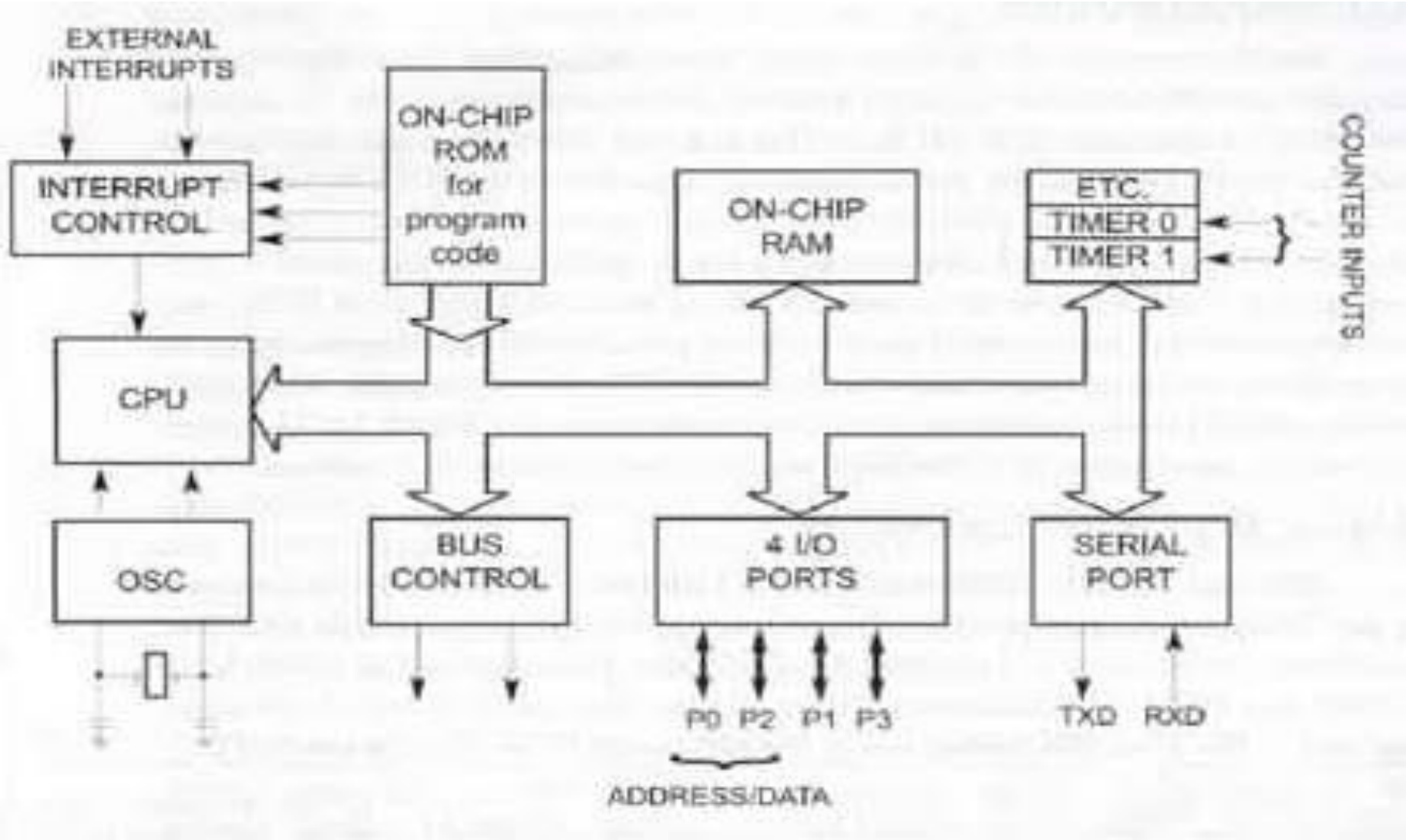
All programs that are written in 8051 will run on 8052 but reverse is not true .

Intel Microcontroller Family

Why 8051 Microcontroller ?

- Very versatile featuring powerful processor supports bit manipulation instructions for real time industrial control application.
- It handles interrupts.
- These interrupts have two priority level and each is allocated fixed 8 bytes of code supports CAN , USB , SPI , TCP/IP Interfaces , ADC , DAC , LCD controller and number of I/O ports .
- Low cost.

Architecture



Architecture

RAM:

- Volatile
- 128 bytes of internal RAM is divided into 32 working registers.
- This 32 working registers are divided into 4 register banks like Bank 0, Bank 1, Bank 2, Bank 3.
- Each bank register has 8 registers (R0-R7)

Serial Port:

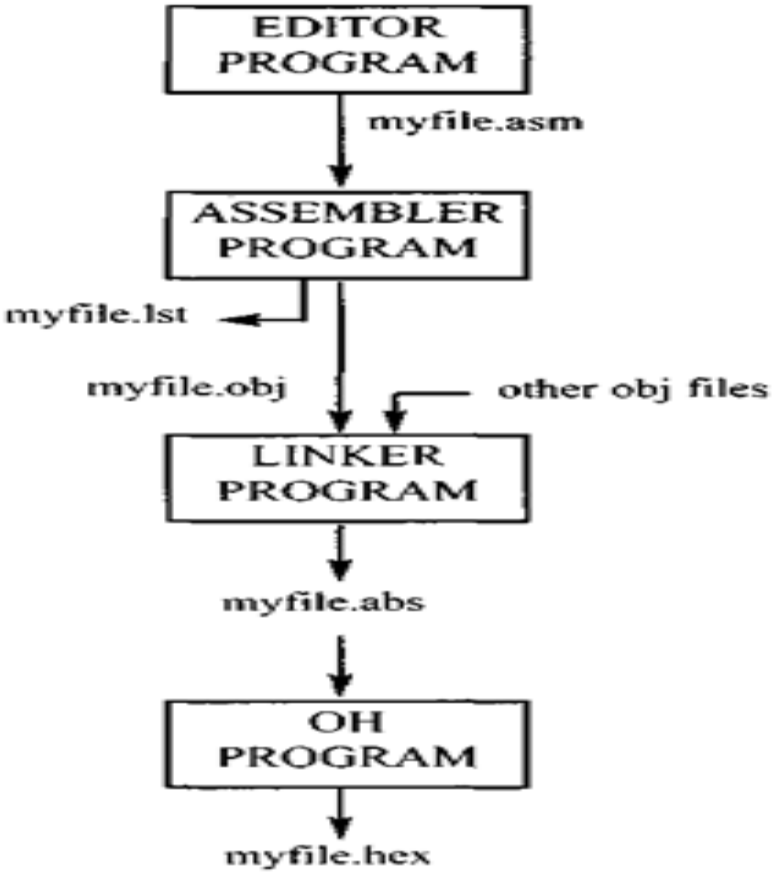
- TXD, RXD: Receive and transmit serial data.

ROM:

- Non volatile
- A code of 4K memory is on chip ROM.

Programming in Assembly Language

Assembling and running an 8051 program.



Commonly Used Data types

- Data Types DB : Defined byte is used to define 8-bit data.
  - For example data1: DB 38 ; decimal data
    data2 : db 01000100B ; Binary

- ORG specifies where your program starts.

- EQU(Equate): to define a constant without occupying a memory location.
  data: EQU 43H

- END: indicate to the assembler the end of the source(.asm) file.

Addressing Modes:

Direct Addressing : MOV A, 07h

Content of RAM address 07H is copied into Accumulator.

Indirect Addressing: MOV A, @R0

Copy the content of memory location whose address is given in R0 register.

Register Addressing: MOV A, R0

Copy the content of R0 register into Accumulator.

Immediate Addressing: MOV R0, #55H

Copy the immediate data 55h provided in instruction into R0 register.

Index Addressing:

MOV DPTR, #25F5H(Index Register)

MOV A,#00h (Base Register)

MOVC A, @A+DPTR

here 'C' means Code. Here the content of A register is added with content of DPTR and the resultant is the address of memory location from where the data is copied to A register.

Instruction Sets

Arithmetic Instructions:

ADD A,<LOC>

SUBB A,<LOC>

INC A

DEC A

MUL AB; A*B => Higher byte in B and Lower byte in A.

DIV AB; A=A/B => Quotient is stored in A, and Remainder in B.

DA;

```
MOV A, #47H    ;    A=47H
MOV B, #25H    ;    B=25H
ADD A, B       ;    A=6CH
DA             ;    A=72H
```

Instruction Sets

Logical Instructions

```
ANL A, #0FH
ORL A, #30H
XRL A, #78H
CPL A
CJNE A, #99H, NEXT(Compare and Jump if not equal)
RR A, #36H;       A=36H after Rotate A=1B
RL A, #72H;       A=72H after Rotate A=EA
MOV A,#26H
RRC A             ; A=13H CY=0
MOV A, #15H
RLC A             ; A=2A CY=0
MOV A, #72H
SWAP A            ; A=27H
```

Instruction Sets

Program Control Transfer Instructions:

Unconditional:

SJMP: Short Jump, Destination address= offset to the current pc Value.(2 byte instruction) 00-FF

LMP: Long Jump(3 byte instruction)

1st byte opcode + 2nd byte first 8-bit address + 3rd byte second 8-bit address.

2-byte target address allows a jump to any memory location from 0000H to FFFFH.

AJMP:

2 byte long with 11-bit constant(3-bits of opcode)

11-bits are substituted at the lower 11-bit of pc with 5 bit of pc unchanged.

Instruction Sets

Conditional Jump:

Instruction	Operation
JZ	Jump if A=0
JNZ	Jump if A not equal to 0
DJNZ	Decrement if register not equal to zero.
CJNE A, #Data, Location	Jump if A not equal to Data
JC	Jump if cy=1
JNC	Jump if cy=0
JB	Jump if bit=1
JNB	Jump if bit=0
JBC	Jump if bit=1 and clear bit

Instruction Sets

Subroutine call and return instructions:

ACALL – 2 byte instruction

uses 11-bit address. The subroutine must be within the same 2K block size.

LCALL – 3 byte instruction

A15-0: 1st byte opcode and 2nd byte address 1st 8-bit address and 3rd byte 2nd 8-bit address.

RET and RETI:

RET: Return from subroutine, pop the return address from the stack and continue execution from here.

RETI: Return from ISR. Pop the return address from the stack.

Programming in Assembly Language

Program Counter(PC)

❖16-bit register holding the address of code memory to be fetched Integral part of cpu and hidden from programmer.

Stack Pointer(SP)

❖8-bit register holding the current address of the stack memory.

Bank Register(R0-R7)

❖Located in the lower 32 bytes of internal ROM.

* RS0 and RS1 selects one of the 4 banks by PSW register.

Programming in Assembly Language

Special function registers

Cpu register

- Accumulator (ACC)

  Bit addressable register

  ACC.7 ACC.6 --------- ACC.0
- B register

  Acts as an operand in multiply and division operations .

  Stores REM and MSB in div and multiply.

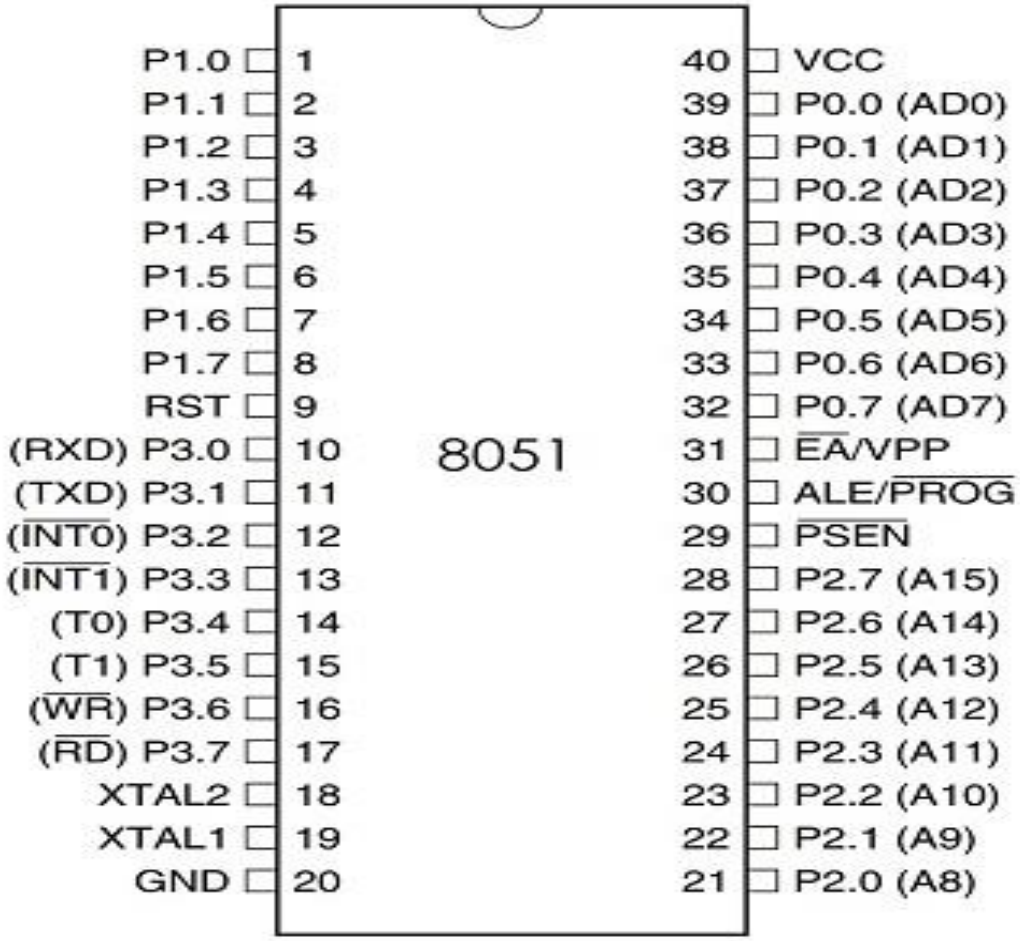
  Also used as GPR for programming.

Data Pointer(DPTR) :combination of two 8-bit register.

Holds 16-bit memory address of external memory to be read/written.

DPH and DPL can be used as 2 independent GPR for application program.

Pin Diagram


P1.0 1
P1.1 2
P1.2 3
P1.3 4
P1.4 5
P1.5 6
P1.6 7
P1.7 8
RST 9
(RXD) P3.0 10
(TXD) P3.1 11
(INT0) P3.2 12
(INT1) P3.3 13
(T0) P3.4 14
(T1) P3.5 15
(WR) P3.6 16
(RD) P3.7 17
XTAL2 18
XTAL1 19
GND 20
8051
40 VCC
39 P0.0 (AD0)
38 P0.1 (AD1)
37 P0.2 (AD2)
36 P0.3 (AD3)
35 P0.4 (AD4)
34 P0.5 (AD5)
33 P0.6 (AD6)
32 P0.7 (AD7)
31 EA/VPP
30 ALE/PROG
29 PSEN
28 P2.7 (A15)
27 P2.6 (A14)
26 P2.5 (A13)
25 P2.4 (A12)
24 P2.3 (A11)
23 P2.2 (A10)
22 P2.1 (A9)
21 P2.0 (A8)

Programming in Assembly Language

Program status word (PSW).

CY	AC	F0	RS1	RS0	OV	--	P

CY	PSW.7	Carry flag.
AC	PSW.6	Auxiliary carry flag.
--	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.
--	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.

A carry from D3 to D4

Carry out from the d7 bit

Reflect the number of 1s in register A

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

Transfer word “POKHARA” from ROM to RAM

Write an assembly language to transfer a word “pokhara” stored in ROM storing at location 250H to RAM starting at location 50H onwards.

```
        ORG OOOOH
        MOV DPTR,#250H
         MOV R7, #07H
        MOVR1,#50H                ;access the upper memory
B1:     CLR A
        MOVC A,@A+DPTR
        MOV @R1,A
        INC DPTR
        INC R1
        DJNZ R7, B1
        END
        ORG 250H
MYDATA: DB “POKHARA”
        END
```

A Simple Interfacing example with 7 Segment Display

```
        org 0000h
again:  mov a,#3fh
        acall prt
        acall delay
        mov a,#06h
        acall prt
        acall delay
        mov a,#5bh
        acall prt
        acall delay
        mov a,#4fh
        acall prt
        acall delay
        mov a,#66h
        acall prt
        acall delay
        mov a,#6dh
        acall prt
        acall delay
        mov a,#7dh
        acall prt
        acall delay
        mov a,#07h
        acall prt
        acall delay
        mov a,#7fh
        acall prt
        acall delay
        mov a,#6fh
        acall prt
        acall delay
        sjmp again
prt:    mov p1,a
        ret

delay:  mov r3,#20
here2:  mov r4,#30
here:   djnz r4, here
        djnz r3, here2
        ret
        end
```

A Simple Interfacing example with 7 Segment Display 2 digit display

```
        ORG 0000H
        MOV R7,#09H
        MOV R6,#09H
        MOV R0,#00H
        MOV R1,#00H
AGAIN:  MOV DPTR,#0300h
        ACALL display
        MOV P2,A
        ACALL display
        MOV P3,A
        ACALL delay
SECOND: INC DPTR
        ACALL display
        MOV P3,A
        DEC R7
        DJNZ R7,SECOND
```

```
ONE:    MOV DPTR,#0300h
        SJMP HERE
TWO:    INC R1
        MOV A,R1
        MOV R0,A
        ACALL display
        MOV P2,A
        MOV DPTR,#0300h
        ACALLL DISPLAY
        MOV P3,A
        DEC R6
        DJNZ R6, SECOND
        SJMP AGAIN
HERE:   INC DPTR
        MOV A,R1
        MOV R0,A
        DJNZ RO, HERE
        SJMP TWO
display: CLR A
        MOVC A, @A+DPTR
        RET
```

```
delay: MOV R3,#20
delay2: MOV R4,#30
delay1:  DJNZ R4, DELAY1
         DJNZ R3, DELAY2
         RET

org 0300h
mydata: db
3fh,06h,5BH,4FH,66H,6DH,7DH,07H
,7FH,6FH
END
```