MICROPROCESSOR AND MICROCONTROLLER BASED SYSTEMS (FOR CSE & IT)

UNIT 5- 8051 MICROCONTROLLER

Introduction

A microcontroller is a computer with most of the necessary support chips onboard. All computers have several things in common, namely: . A central processing unit (CPU) that 'executes' programs. . Some random-access memory (RAM) where it can store data that is variable. Some read only memory (ROM) where programs to be executed can be stored. . Input and output (I/O) devices that enable communication to be established with the outside world i.e. connection to devices such as keyboard, mouse, monitors and other peripherals. There are a number of other common characteristics that define microcontrollers. If a computer matches a majority of these characteristics, then it can be classified as a 'microcontroller'. Microcontrollers may be: . 'Embedded' inside some other device (often a consumer product) so that they can control the features or actions of the product. Another name for a microcontroller is therefore an 'embedded controller'. . Dedicated to one task and run one specific program. The program is stored in ROM and generally does not change. A low-power device. A battery-operated microcontroller might consume as little as 50 milliwatts. A microcontroller may take an input from the device it is controlling and controls the device by sending signals to different components in the device. A microcontroller is often small and low cost. The components may be chosen to minimise size and to be as inexpensive as possible. The actual processor used to implement a microcontroller can vary widely. In many products, such as microwave ovens, the demand on the CPU is fairly low and price is an important consideration. In these cases, manufacturers turn to dedicated microcontroller chips – devices that were originally designed to be low-cost, small, low-power, embedded CPUs. The Motorola 6811 and Intel 8051 are both good examples of such chips. A typical low-end microcontroller chip might have 1000 bytes of ROM and 20 bytes of RAM on the chip, along with eight I/O pins. In large quantities, the cost of these chips can sometimes be just a few pence.

MICROPROCESSOR	MICROCONTROLLER
1.the functional blocks of a microprocessor are ALU,registers,timing and control unit.	**1**.the microcontroller includes the functional blocks of microprocessor and in addition has timer,parallel I/O port,serial I/O port,internalRAM and EPROM/EEPROM memory.some controllers has been ADC and/or DAC.

2.the microprocessor is concerned with the rapid movement of code and data between external memory and microprocessor. Hence it has large number of instructions for moving data between external memory and microprocessor.	**2**.the microcontroller is concerned with the rapid movement of code and data within microcontroller. Hence it has few instructions for data transfer between external memory and microcontroller.
3.the microprocessors mostly operate on byte/word data and so has very few bit manipulating instructions.	**3**.the microcontrollers often manipulate with bits and has large number of bit manipulating instructions.
4.the microprocessors usually require interfacing of a large number of addition ICs to form an microcomputer based system. Hence the PCB of microprocessor based system will be large and so the system will be costly.	**4**.the microcontroller can be used to form a single chip microcomputer based system without any additional ICs. Hence the PCB of microcontroller based system will be small and so system will be cheap.
5.the microprocessor are used for designing general purpose digital computing system(or computers).	**5**.the microcontrollers are used for designing application specific dedicated systems.

The 8051 is a Harvard architecture, CISC instruction set, single chip microcontroller (µC) series which was developed by Intel in 1980 for use in embedded systems.

Intel's original MCS-51 family was developed using NMOS technology, but later it is modified with CMOS technology and referred as 80C51which consume less power than their NMOS predecessors and gives high switching speed than NMOS technology. This made them more suitable for battery-powered devices.

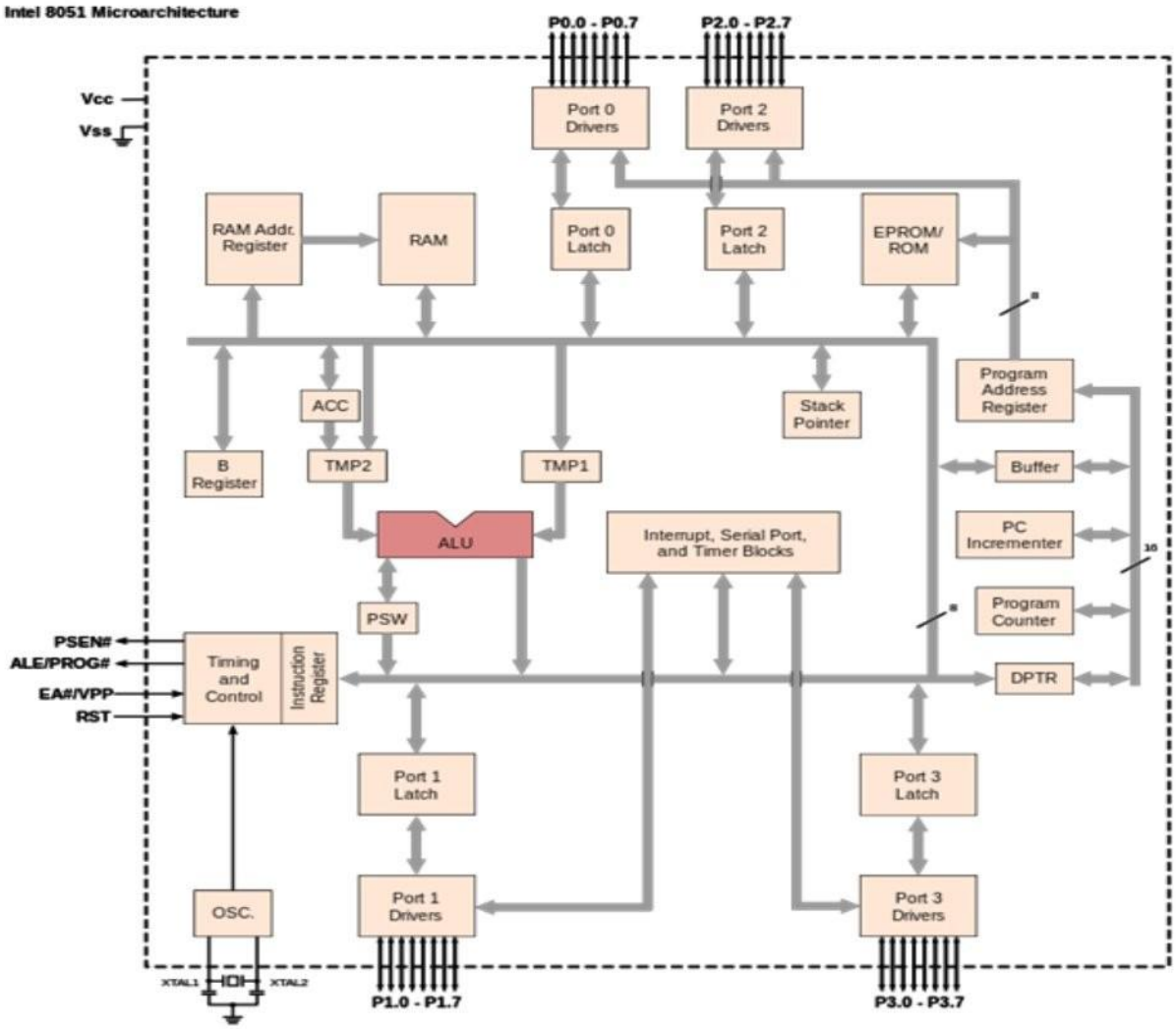
DEVICE	ON-CHIP DATA MEMORY(bytes)	ON-CHIP PROGRAM MEMORY(bytes)	16-BIT TIMER/COUNTER	NO. OF VECTORED INTERUPTS	FULL DUPLEX I/O
8051	128	4k ROM	2	5	1

Features of 8051 microcontroller

- 4 KB on chip program memory.

- 128 bytes on chip data memory(RAM).
- 128 user defined software flags.
- 8-bit data bus
- 16-bit address bus
- 32 general purpose registers each of 8 bits
- 16 bit timers (usually 2, but may have more, or less).
- 3 internal and 2 external interrupts.
- Bit as well as byte addressable RAM area of 16 bytes.
- Four 8-bit ports, (short models have two 8-bit ports).
- 16-bit program counter and data pointer.
- 1 Microsecond instruction cycle with 12 MHz Crystal.

Architecture of 8051



An 8051 microcontroller has the following 12 major components:

1. ALU (Arithmetic and Logic Unit)
2. PC (Program Counter)
3. Registers
4. Timers and counters
5. Internal RAM and ROM
6. Four general purpose parallel input/output ports
7. Interrupt control logic with five sources of interrupt
8. Serial date communication
9. PSW (Program Status Word)
10. Data Pointer (DPTR)
11. Stack Pointer (SP)

1. ALU

All arithmetic and logical functions are carried out by the ALU. Addition, subtraction with carry, and multiplication come under arithmetic operations.Logical AND, OR and exclusive OR (XOR) come under logical operations.

2.Program Counter (PC)

A program counter is a 16-bit register and it has no internal address. The basic function of program counter is to fetch from memory the address of the next instruction to be executed. The PC holds the address of the next instruction residing in memory and when a command is encountered, it produces that instruction. This way the PC increments automatically, holding the address of the next instruction.

3. Registers

Registers are usually known as data storage devices. 8051 microcontroller has 2 registers, namely Register A and Register B. Register A serves as an accumulator while Register B functions as a general purpose register. These registers are used to store the output of mathematical and logical instructions. The operations of addition, subtraction, multiplication and division are carried out by Register A. Register B is usually unused and comes into picture only when multiplication and division functions are carried out by Register A. Register A also involved in data transfers between the microcontroller and external memory.

8051 microcontroller also has 7 Special Function Registers (SFRs). They are:

1. Serial Port Data Buffer (SBUF)
2. Timer/Counter Control (TCON)
3. Timer/Counter Mode Control (TMOD)
4. Serial Port Control (SCON)
5. Power Control (PCON)
6. Interrupt Priority (IP)
7. Interrupt Enable Control (IE)

4. Timers and Counters

Synchronization among internal operations can be achieved with the help of clock circuits which are responsible for generating clock pulses. During each clock pulse a particular operation will be carried out, thereby, assuring synchronization among operations. For the formation of an oscillator, we are provided with two pins XTAL1 and XTAL2 which are used for connecting a resonant network in 8051 microcontroller device. In addition to this, circuit also consists of four more pins.

Internal operations can be synchronized using clock circuits which produce clock pulses. With each clock pulse, a particular function will be accomplished and hence synchronization is achieved. There are two pins XTAL1 and XTAL2 which form an oscillator circuit which connect to a resonant network in the microcontroller. The circuit also has 4 additional pins -

1. EA: External enable
2. ALE: Address latch enable

3. PSEN: Program store enable and

4. RST: Reset

Quartz crystal is used to generate periodic clock pulses.

5. Internal RAM and ROM

ROM

A code of 4K memory is incorporated as on-chip ROM in 8051. The 8051 ROM is a non-volatile memory meaning that its contents cannot be altered and hence has a similar range of data and program memory, i.e, they can address program memory as well as a 64K separate block of data memory.

RAM

The 8051 microcontroller is composed of 128 bytes of internal RAM. This is a volatile memory since its contents will be lost if power is switched off. These 128 bytes of internal RAM are divided into 32 working registers which in turn constitute 4 register banks (Bank 0-Bank 3) with each bank consisting of 8 registers (R0 - R7). There are 128 addressable bits in the internal RAM.

Four General Purpose Parallel Input/Output Ports

The 8051 microcontroller has four 8-bit input/output ports.

PORT P0: When there is no external memory present, this port acts as a general purpose input/output port. In the presence of external memory, it functions as a multiplexed address and data bus. It performs a dual role.

PORT P1: This port is used for various interfacing activities. This 8-bit port is a normal I/O port i.e. it does not perform dual functions.

PORT P2: Similar to PORT P0, this port can be used as a general purpose port when there is no external memory but when external memory is present it works in conjunction with PORT PO as an address bus. This is an 8-bit port and performs dual functions.

PORT P3: PORT P3 behaves as a dedicated I/O port

INSTRUCTION SET OF 8051

Data Transfer

MOV A,Rn Move register to accumulator 1

MOV A,direct Move direct byte to accumulator

MOV A,@Ri Move indirect RAM to accumulator

MOV A,#data Move immediate data to accumulator MOV

Rn,A Move accumulator to register 1

MOV Rn,direct Move direct byte to register

MOV Rn,#data Move immediate data to register

MOV direct,A Move accumulator to direct byte

MOV direct,Rn Move register to direct byte 2

MOV direct,direct Move direct byte to direct byte

MOV direct,@Ri Move indirect RAM to direct byte

MOV direct,#data Move immediate data to direct byte

MOV @Ri,A Move accumulator to indirect RAM 1

MOV @Ri,direct Move direct byte to indirect RAM

MOV @Ri, #data Move immediate data to indirect RAM

MOV DPTR, #data16 Load data pointer with a 16-bit constant

MOVC A,@A + DPTR Move code byte relative to DPTR to accumulator

MOVC A,@A + PC Move code byte relative to PC to accumulator

MOVX A,@Ri Move external RAM (8-bit addr.) to A

MOVX A,@DPTR Move external RAM (16-bit addr.) to A

MOVX @Ri,A Move A to external RAM (8-bit addr.)

MOVX @DPTR,A Move A to external RAM (16-bit addr.)

PUSH direct Push direct byte onto stack

POP direct Pop direct byte from stack

XCH A,Rn Exchange register with accumulator

XCH A,direct Exchange direct byte with accumulator

XCH A,@Ri Exchange indirect RAM with accumulator

XCHD A,@Ri Exchange low-order nibble indir. RAM with A

Arithmetic

ADD A,Rn	Adds the register to the accumulator
ADD A,direct	Adds the direct byte to the accumulator
ADD A,@Ri	Adds the indirect RAM to the accumulator
ADD A,#data	Adds the immediate data to the accumulator
ADDC A,Rn	Adds the register to the accumulator with a carry flag
ADDC A,direct	Adds the direct byte to the accumulator with a carry flag
ADDC A,@Ri	Adds the indirect RAM to the accumulator with a carry flag
ADDC A,#data	Adds the immediate data to the accumulator with a carry flag
SUBB A,Rn	Subtracts the register from the accumulator with a borrow
SUBB A,direct	Subtracts the direct byte from the accumulator with a borrow
SUBB A,@Ri	Subtracts the indirect RAM from the accumulator with a borrow

SUBB A,#data	Subtracts the immediate data from the accumulator with a borrow
INC A	Increments the accumulator by 1
INC Rn	Increments the register by 1
INC Rx	Increments the direct byte by 1
INC @Ri	Increments the indirect RAM by 1
DEC A	Decrements the accumulator by 1
DEC Rn	Decrements the register by 1
DEC Rx	Decrements the direct byte by 1
DEC @Ri	Decrements the indirect RAM by 1
INC DPTR	Increments the Data Pointer by 1
MUL AB	Multiplies A and B
DIV AB	Divides A by B
DA A	Decimal adjustment of the accumulator

Logic Operations

ANL A,Rn AND register to accumulator

ANL A,direct AND direct byte to accumulator

ANL A,@Ri AND indirect RAM to accumulator

ANL A,#data AND immediate data to accumulator

ANL direct,A AND accumulator to direct byte

ANL direct,#data AND immediate data to direct byte

ORL A,Rn OR register to accumulator

ORL A,direct OR direct byte to accumulator

ORL A,@Ri OR indirect RAM to accumulator

ORL A,#data OR immediate data to accumulator

ORL direct,A OR accumulator to direct byte

ORL direct,#data OR immediate data to direct byte

XRL A,Rn Exclusive OR register to accumulator

XRL A direct Exclusive OR direct byte to accumulator

XRL A,@Ri Exclusive OR indirect RAM to accumulator

XRL A,#data Exclusive OR immediate data to accumulator

XRL direct,A Exclusive OR accumulator to direct byte

XRL direct,#data Exclusive OR immediate data to direct byte

CLR A Clear accumulator

CPL A Complement accumulator

RL A Rotate accumulator left

RLC A Rotate accumulator left through carry

RR A Rotate accumulator right

RRC A Rotate accumulator right through carry

SWAP A Swap nibbles within the accumulator 1

Boolean Variable Manipulation

CLR C Clear carry flag

CLR bit Clear direct bit2

SETB	C	Set carry flag
SETB	bit	Set direct bit
CPL	C	Complement carry flag
CPL	bit	Complement direct bit2
ANL	C,bit	AND direct bit to carry flag
ANL	C,/bit	AND complement of direct bit to carry
ORL	C,bit	OR direct bit to carry flag
ORL	C,/bit	OR complement of direct bit to carry
MOV	C,bit	Move direct bit to carry flag
MOV	bit,C	Move carry flag to direct bit

Branch Instructions

LCALL	addr11	Absolute subroutine call
LCALL	addr16	Long subroutine call
RET		Return from subroutine
RETI		Return from interrupt
AJMP	addr11	Absolute jump2
LJMP	addr16	Long iump
SJMP	rel	Short jump (relative addr.)
JMP	@A + DPTR	Jump indirect relative to the DPTR
JZ	rel	Jump if accumulator is zero
JNZ	rel	Jump if accumulator is not zero
JC	rel	Jump if carry flag is set
JNC	rel	Jump if carry flag is not set
JB	bit,rel	Jump if direct bit is set

JNB bit,rel Jump if direct bit is not set

JBC bit,rel Jump if direct bit is set and clear bit

CJNE A,direct,rel Compare direct byte to A and jump if not equal

Machine Control

CJNE A,#data,rel Compare immediate to A and jump if not equal

CJNE Rn,#data rel Compare immed. to reg. and jump if not equal

CJNE @Ri,#data,rel Compare immed. to ind. and jump if not equal

DJNZ Rn,rel Decrement register and jump if not zero

DJNZ direct,rel Decrement direct byte and jump if not zero NOP

No operation

Assembly Language Program:

Addition of 8 bits

```
MOV A, #05H
MOV R, #09H
ADD A, R1
MOV DPTR, #2050H
MOV @ DPTR, A
AGAIN: SJMP AGAIN
```

Subtraction of 8 bits

```
ORG 4100
CLR C
MOV A,#data1
SUBB A,#data2
MOV DPTR,#4500
```

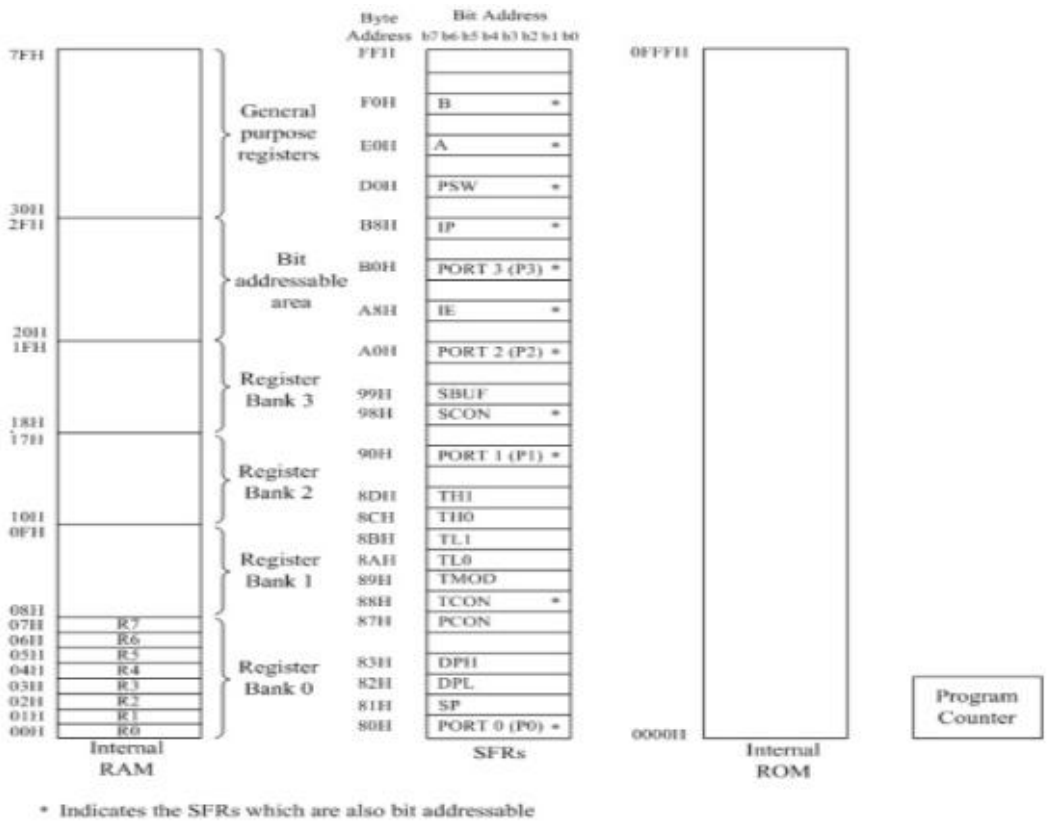
```
MOVX @DPTR,A
   HERE: SJMP HERE
```

Multiplication of 8 bits

```
ORG 4100
   CLR C
MOV    A,#data1
   MOV B,#data2
   MUL AB
MOV    DPTR,#4500
   MOVX  @DPTR,A
   INC DPTR
MOV A,B
MOVX @DPTR,A
   HERE: SJMP HERE
```

Memory organization:

Programming Model of 8051

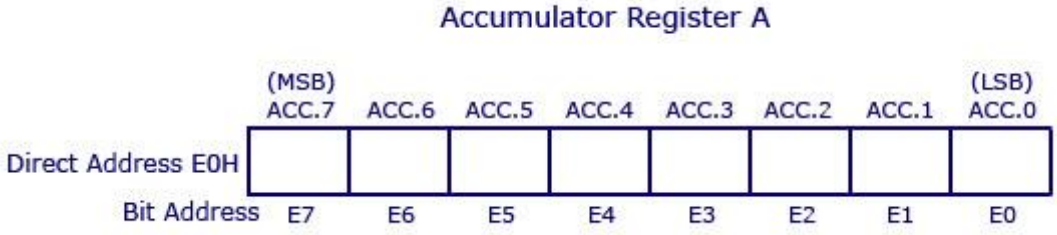


Special Function Registers and Ports

There **are 21 Special function registers (SFR)** in 8051 micro controller and this includes Register A, Register B, Processor Status Word (PSW), PCON etc etc. There are 21 unique locations for these 21 special function registers and each of these register is of 1 byte size. Some of these special function registers **are bit addressable** (which means you can access 8 individual bits inside a single byte), while some others are **only byte addressable.**

Register A/Accumulator

The Accumulator holds the result of most of arithmetic and logic operations. ACC is usually accessed by direct addressing and its physical address is **E0H.** Accumulator is **both byte and bit addressable.** To access the first bit (i.e bit 0) or to access accumulator as a single byte (all 8 bits at once), you may use the same physical address E0H. Now if you want to access the second bit (i.e bit 1), you may use E1H and for third bit E2H and so on.



Register B

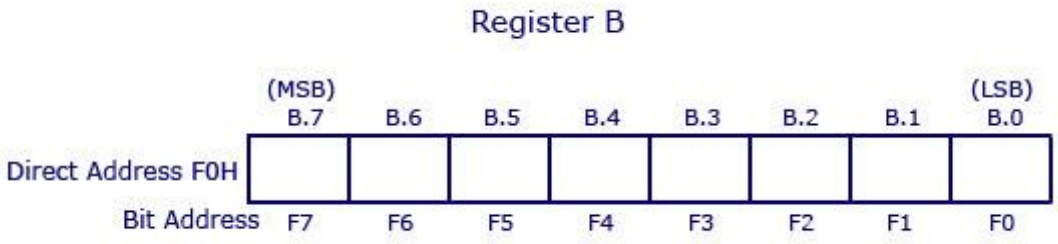
The major purpose of this register is in executing multiplication and division. The 8051 micro controller has a single instruction for multiplication **(MUL)** and division **(DIV)**. multiplication is repeated addition, where as division is repeated subtraction. While programming 8085 a loop

is written to execute repeated addition/subtraction to perform multiplication and division. Now here in 8051 you can do this with a single instruction.

Ex: MUL A,B – When this instruction is executed, data inside A and data inside B is multiplied and answer is stored in A.

Note: For MUL and DIV instructions, it is necessary that the two operands must be in A and B.

Register B is also byte addressable and bit addressable. To access bit o or to access all 8 bits (as a single byte), physical address F0 is used. To access bit 1 you may use F1 and so on. Please take a look at the picture below.

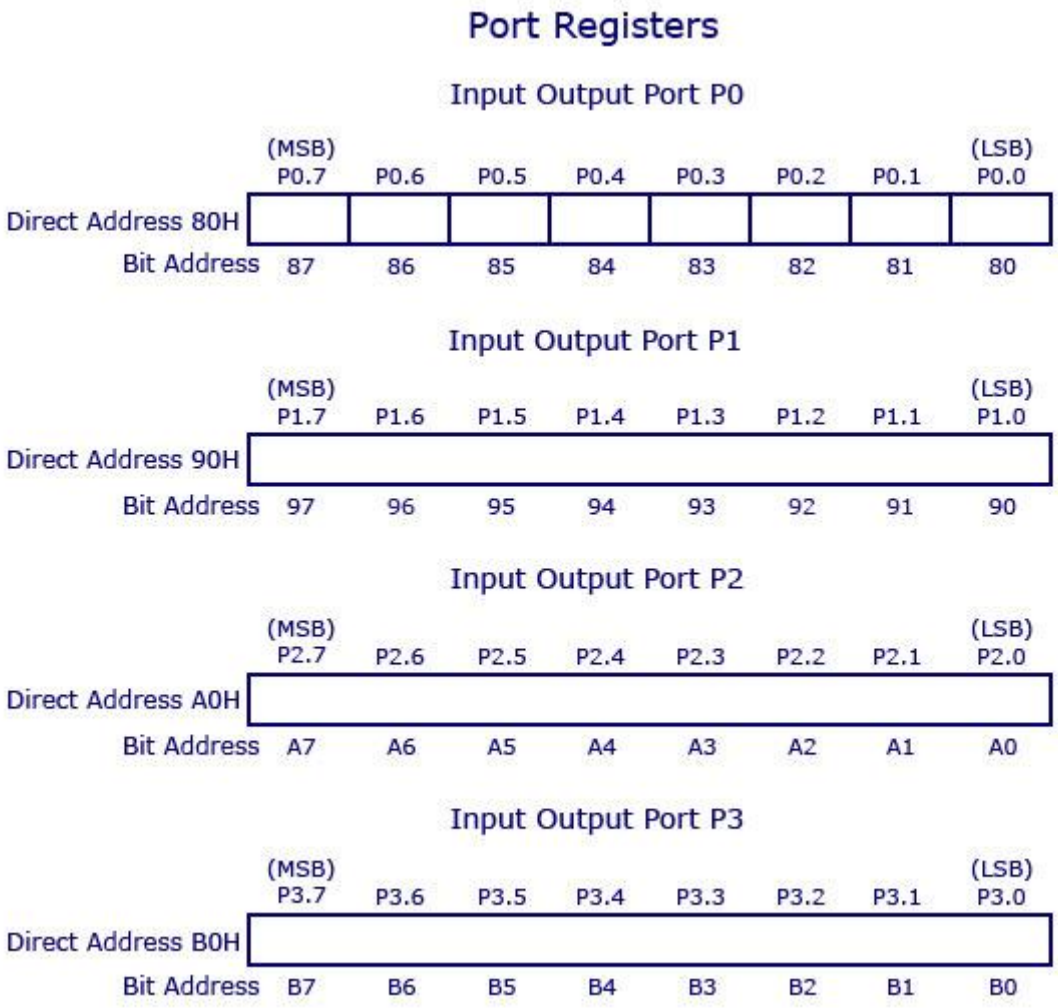


Note: Register B can also be used for other general purpose operations.

Port Registers

4 Input/Output ports named P0, P1, P2 and P3 has got four corresponding port registers with same name P0, P1, P2 and P3. Data must be written into port registers first to send it out to any other external device through ports. Similarly any data received through ports must be read from port registers for performing any operation. All 4 port registers are bit as well as byte

addressable.



From the figure:-

- The physical address of port 0 is 80
- The physical address of port 1 is 90
- And that of port 2 is A0
- And that of port 3 is B0

Stack Pointer

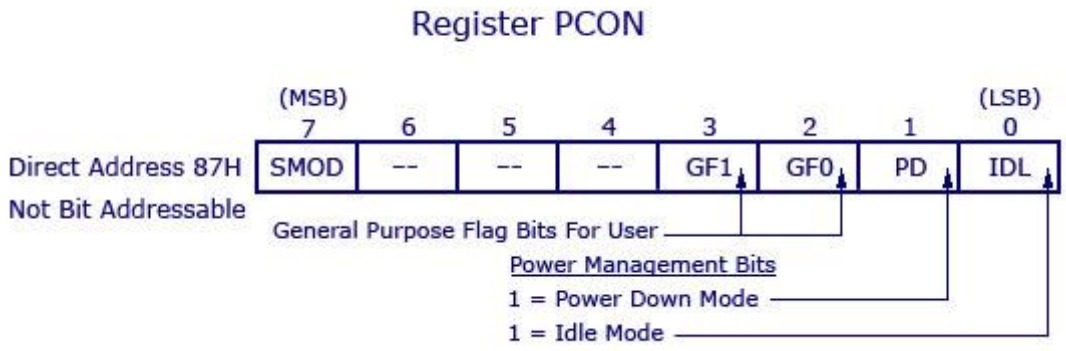
Known popularly with an acronym SP, stack pointer represents a pointer to the the system stack. Stack pointer is an 8 bit register, the direct address of SP is 81H and it is only byte addressable, which means you cant access individual bits of stack pointer. The content of the stack pointer points to the last stored location of system stack. To store something new in system stack, the SP must be incremented by 1 first and then execute the "store" command. Usually after a system reset SP is initialized as 07H and data can be stored to stack from 08H onwards. This is usually a default case and programmer can alter values of SP to suit his needs.

Power Management Register (PCON)

Power management using a microcontroller is something you see every day in mobile phones. Haven't you noticed and got wondered by a mobile phone automatically going into stand by

mode when not used for a couple of seconds or minutes ? This is achieved by power management feature of the controller used inside that phone.

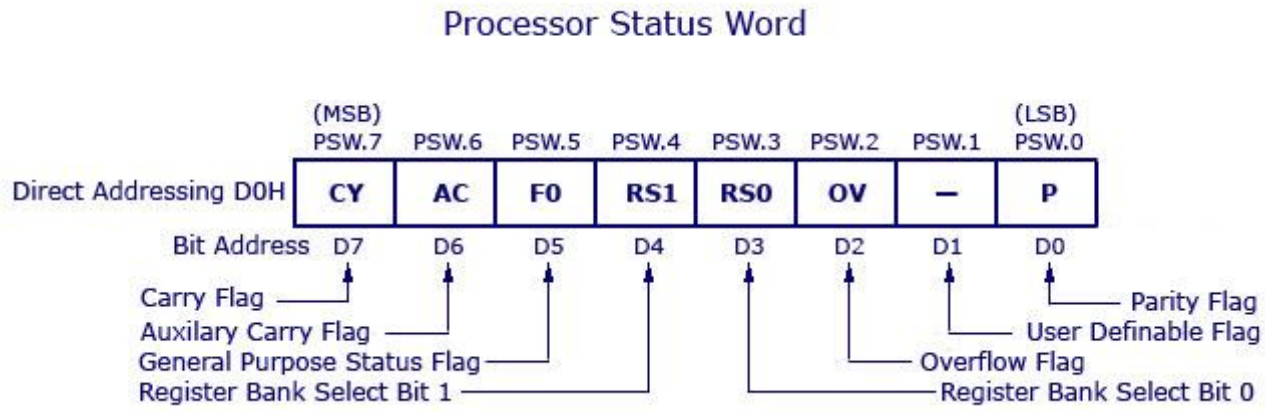
As the name indicates, this register is used for efficient power management of 8051 micro controller. Commonly referred to as PCON register, this is a dedicated SFR for power management alone. From the figure below you can observe that there are 2 modes for this register :- **Idle mode and Power down mode.**



Setting bit 0 will move the micro controller to Idle mode and Setting bit 1 will move the micro controller to Power down mode.

Processor Status Word (PSW)

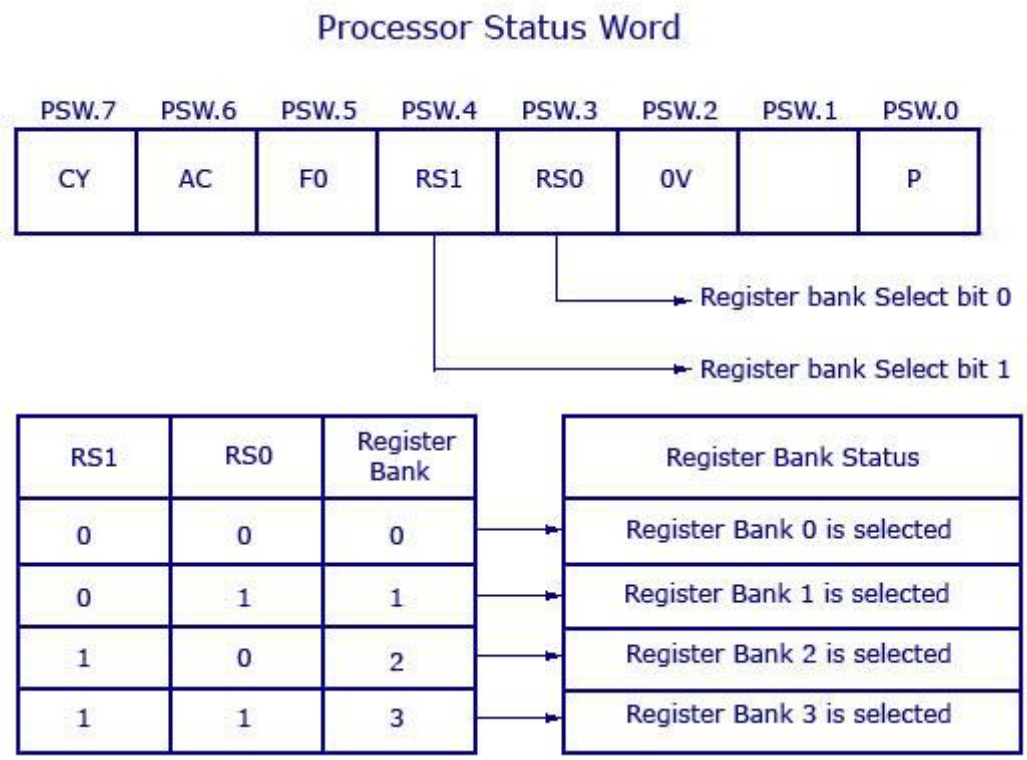
Commonly known as the PSW register, this is a vital SFR in the functioning of micro controller. This register reflects the status of the operation that is being carried out in the processor. The picture below shows PSW register and the way register banks are selected using PSW register bits – RS1 and RS0. PSW register is both bit and byte addressable. The physical address of PSW starts from D0H. The individual bits are then accessed using D1, D2 … D7. The various individual bits are explained below.



Bit No	Bit Symbol	Direct Address	Name	Function
0	P	D0	Parity	This bit will be set if ACC has odd number of 1's after an operation. If not, bit will remain cleared.

1	–	D1		User definable bit
2	OV	D2	Overflow	OV flag is set if there is a carry from bit 6 but not from bit 7 of an Arithmetic operation. It's also set if there is a carry from bit 7 (but not from bit 6) of Acc
3	RS0	D3	Register Bank select bit 0	LSB of the register bank select bit. Look for explanation below this table.
4	RS1	D4	Register Bank select bit 1	MSB of the register bank select bits.
5	F0	D5	Flag 0	User defined flag
6	AC	D6	Auxiliary carry	This bit is set if data is coming out from bit 3 to bit 4 of Acc during an Arithmetic operation.
7	CY	D7	Carry	Is set if data is coming out of bit 7 of Acc during an Arithmetic operation.

At a time registers can take value from R0,R1…to R7. You may already know there are 32 such registers. There are 4 register banks named 0,1,2 and 3. Each bank has 8 registers named from R0 to R7. At a time only one register bank can be selected. Selection of register bank is made possible through PSW register bits PSW.3 and PSW.4, named as RS0 and RS1.These two bits are known as register bank select bits as they are used to select register banks. The picture will talk more about selecting register banks.



SFR	Address	Function
DPH	83	Data pointer registers (High). Only byte addressing possible.
DPL	82	Data pointer register (Low). Only byte addressing possible.
IP	B8	Interrupt priority. Both bit addressing and byte addressing possible.
IE	A8	Interrupt enable. Both bit addressing and byte addressing possible.
SBUF	99	Serial Input/Output buffer. Only byte addressing is possible.
SCON	98	Serial communication control. Both bit addressing and byte addressing possible.
TCON	88	Timer control. Both bit addressing and byte addressing possible.
TH0	8C	Timer 0 counter (High). Only byte addressing is possible.

TL0	8A	Timer 0 counter (Low). Only byte addressing is possible.
TH1	8D	Timer 1 counter (High). Only byte addressing is possible.
TL1	8B	Timer 1 counter (Low). Only byte addressing is possible.
TMOD	89	Timer mode select. Only byte addressing is possible.

5 ADDRESSING MODES FOR 8051.
1) Immediate addressing mode **2)**Direct addressing mode **3)** Register direct addressing mode **4)** Register indirect addressing mode **5)** Indexed addressing mode.

Immediate Addressing Mode

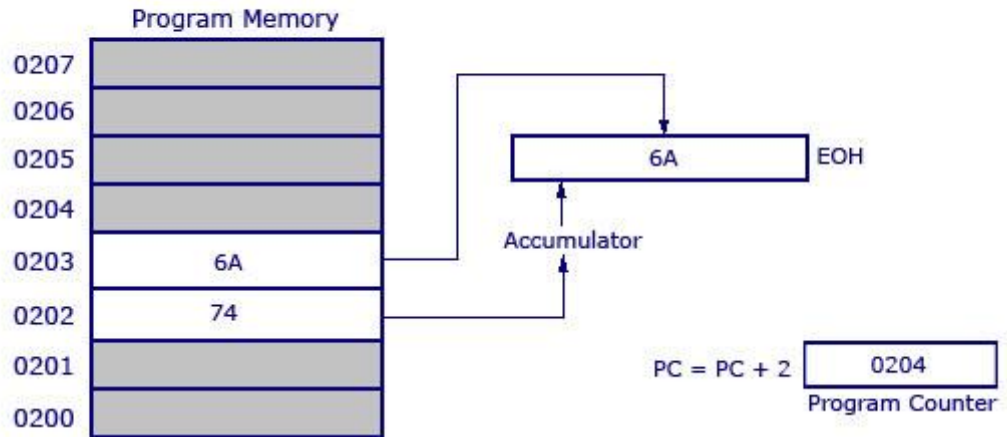
MOV A, #6AH

In general we can write MOV A, #data

This addressing mode is named as ***"immediate"*** because it transfers an 8-bit data immediately to the accumulator (destination operand).

Immediate Addressing Mode

Instruction	Opcode	Bytes	Cycles
MOV A, #6AH	74H	2	1



The opcode for MOV A, # data is 74H. The opcode is saved in program memory at 0202 address. The data 6AH is saved in program memory 0203. When the opcode 74H is read, the next step taken would be to transfer whatever data at the next program memory address (here at 0203) to accumulator A (E0H is the address of accumulator). This instruction is of two bytes and is executed in one cycle. So after the execution of this instruction, program counter will add 2 and move to o204 of program memory.

Note: The **'#'** symbol before 6AH indicates that operand is a data (8 bit). If **'#'** is not present then the hexadecimal number would be taken as address.

Direct Addressing Mode

The address of the data (source data) is given as operand..

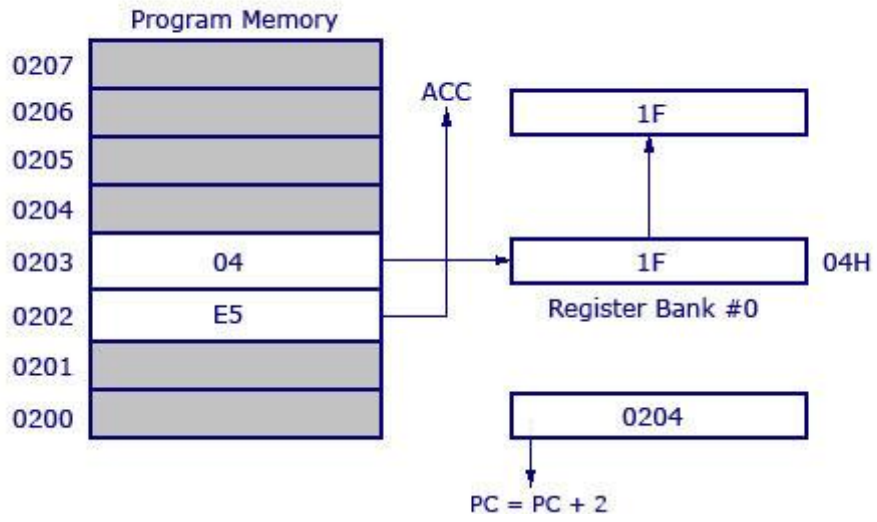
MOV A, 04H

Here 04H is the address of register 4 of register bank#0. When this instruction is executed, what ever data is stored in register 04H is moved to accumulator. In the picture below we can see, register 04H holds the data 1FH. So the data 1FH is moved to accumulator.

Note: We have not used **'#'** in direct addressing mode, unlike immediate mode. If we had used **'#'**, the data value 04H would have been transferred to accumulator instead 0f 1FH.

Direct Addressing Mode

Instruction	Opcode	Bytes	Cycles
MOV A, #04H	E5	2	1



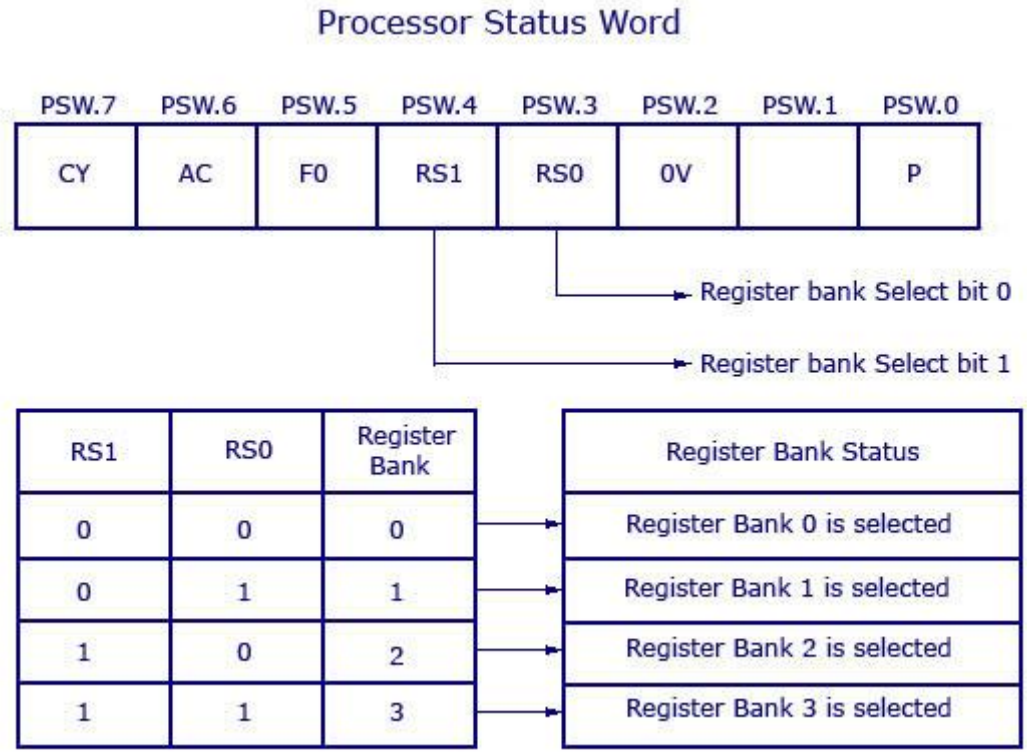
As shown in picture above this is a 2 byte instruction which requires 1 cycle to complete. Program counter will increment by 2 and stand in 0204. The opcode for instruction ***MOV A, address*** is E5H. When the instruction at 0202 is executed (E5H), accumulator is made active and ready to receive data. Then program control goes to next address that is 0203 and look up the address of the location (04H) where the source data (to be transferred to accumulator) is located. At 04H the control finds the data 1F and transfers it to accumulator and hence the execution is completed.

Register Direct Addressing Mode

In this addressing mode we use the register name directly (as source operand). An example is shown below.

MOV A, R4

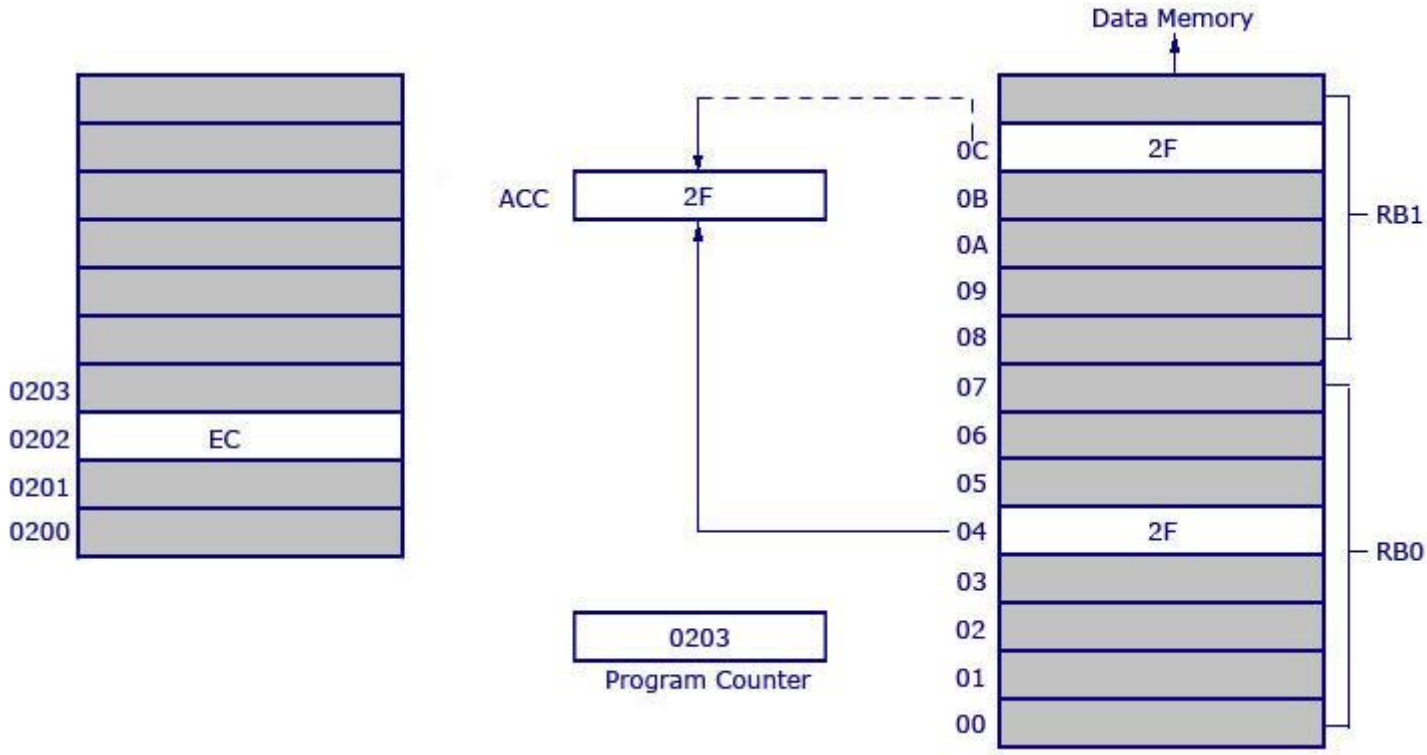
At a time registers can take value from R0,R1…to R7. You may already know there are 32 such registers. . There are 4 register banks named 0,1,2 and 3. Each bank has 8 registers named from R0 to R7. At a time only one register bank can be selected. Selection of register bank is made possible through a Special Function Register (SFR) named Processor Status Word (PSW). PSW is an 8 bit SFR where each bit can be programmed. Bits are designated from PSW.0 to PSW.7 Register banks are selected using PSW.3 and PSW.4 These two bits are known as register bank select bits as they are used to select register banks. A picture below shows the PSW register and the Register Bank Select bits with status.



So in register direct addressing mode, data is transferred to accumulator from the register (based on which register bank is selected).

Register Direct Addressing Mode

Instruction	Opcode	Bytes	Cycles
MOV A, R4	ECH	1	1



So we see that opcode for MOV A, R4 is EC. The opcode is stored in program memory address 0202 and when it is executed the control goes directly to R4 of the respected register bank (that is selected in PSW). If register bank #0 is selected then the data from R4 of register bank #0 will be moved to accumulator. (Here it is 2F stored at 04 H). 04 H is the address of R4 of register bank #0. Movement of data (2F) in this case is shown as bold line. Now please take a look at the dotted line. Here 2F is getting transferred to accumulator from data memory location 0C H. Now understand that 0C H is the address location of Register 4 (R4) of register bank #1. Programmers usually get confused with register bank selection. Also keep in mind that data at R4 of register bank #0 and register bank #1 (or even other banks) will not be same. So wrong selection of register banks will result in undesired output.

Also note that the instruction above is 1 byte and requires 1 cycle for complete execution. This means using register direct addressing mode can save program memory.

Register Indirect Addressing Mode

Address of the data (source data to transfer) is given in the register operand.

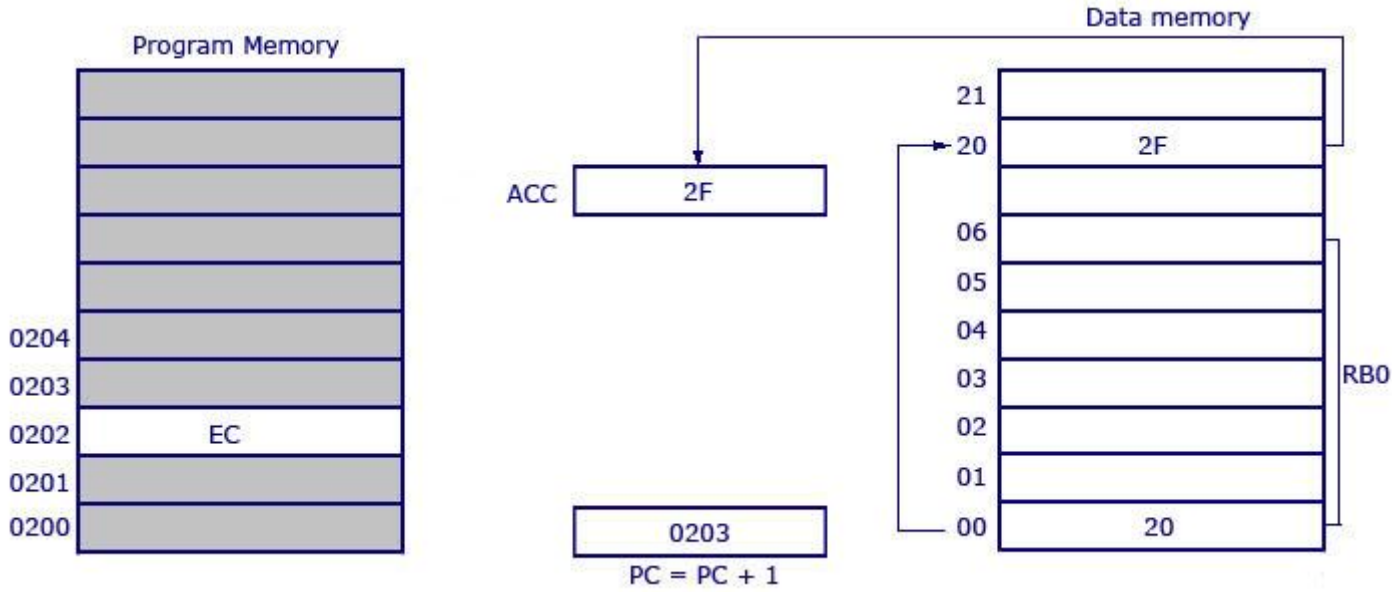
MOV A, @Ro

Here the value inside R0 is considered as an address, which holds the data to be transferred to accumulator.

Example: If R0 holds the value 20H, and we have a data 2F H stored at the address 20H, then the value 2FH will get transferred to accumulator after executing this instruction.

Register Indirect Addressing Mode

Instruction	Opcode	Bytes	Cycles
MOV A, @ R0	E6H	1	1



So the opcode for **MOV A, @R0** is E6H. Assuming that register bank #0 is selected. So the R0 of register bank #0 holds the data 20H. Program control moves to 20H where it locates the data 2FH and it transfers 2FH to accumulator.
This is a single byte instruction and the program counter increments 1 and moves to 0203 of program memory.

Note: Only R0 and R1 are allowed to form a register indirect addressing instruction. In other words programmer can must make any instruction either using @R0 or @R1. All register banks are allowed.

Implied addressing modes

The instruction itself is specifies the data to be operated by the instruction.
Example
CPL C

Relative addressing

The instruction specifies the address relative to program counter. The instruction will carry an offset whose range is -128 to +127. The offset is added to PC to generate 16-bit physical address

Case study - Microcontroller based Washing Machine

1. MICROCONTROLLER-BASED WASHING-MACHINE CONTROL PRESENTED BY, SANGEETHA BHARATH.G

What Is a Washing Machine? A washing machine is an electronic device that is designed to wash laundry like clothes, sheets, towels and other bedding. A washing machine is built with two steel tubs which are the inner tub and the outer tub whose main role is to prevent water from spilling to other parts of the machine.

2. Control knobs in washing machine: • Load select knob • Water inlet select knob • Mode select knob • Program select knob

3. Load select knob:- load Number of clothes low medium high Load select

4. Water inlet select knob:- hot cold both-mixed Water inlet

5. Mode select knob:- Save mode Normal mode Mode

6. Program select knob:- Heavy Clothes very dirty Normal Normal dirty clothes LIGHT For light dirty clothes Delicate For silk clothes

7. Operations:- • Fill:- water will be filled by the pump as per the load knob selected. • Agitate:- The wash basket will rotate in a clockwise direction for 10 revolutions, After that basket will stop for 2 seconds, then rotate 10 revolutions in anticlockwise direction. The process will be continued for specified minutes in cycle table.

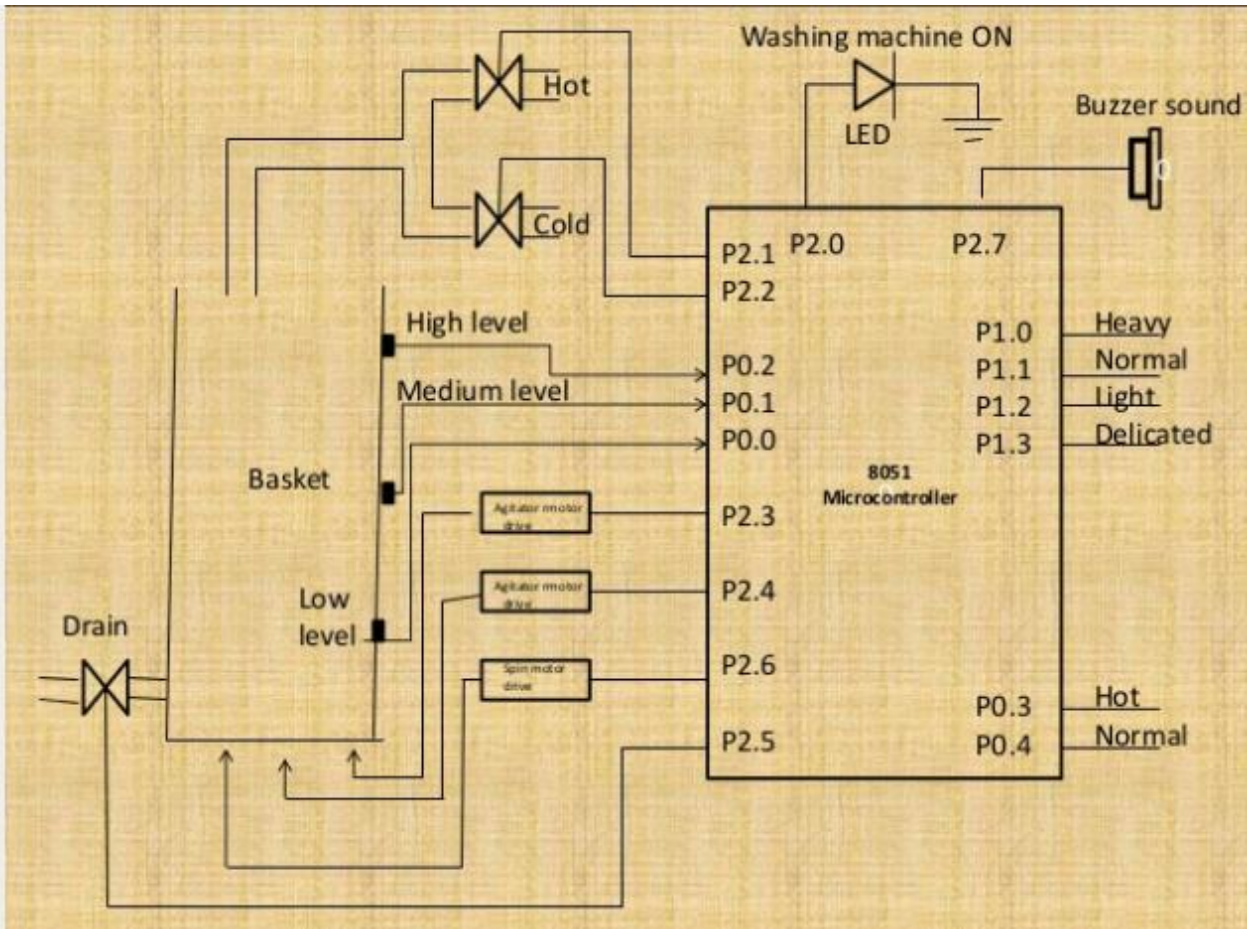
8. LED ON After completion of washing cycle, buzzer sound will be generated.◊Drain:- After agitation, the water and detergent are drained. Spin:- During spin, agitator will be stationary, only the basket will rotate at high speed. Then the moisture of clothes are removed through holes in the inner metallic basket. Indicator:- Machine ON

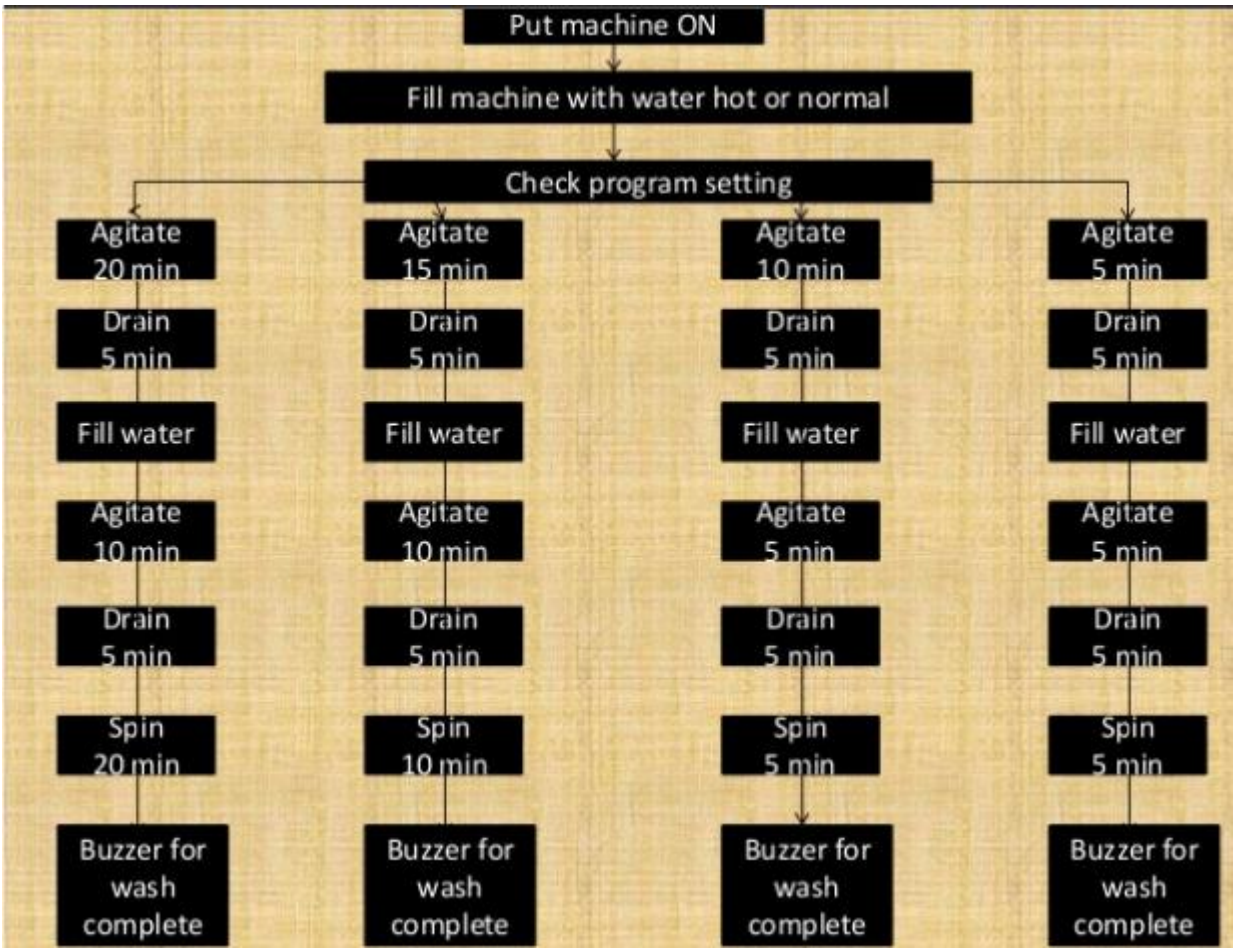
9. Washing cycle for Heavy, Normal, Light and Delicate setting Operation Heavy Normal Light Delicate Fill water Set by load Select knob Set by load Select knob Set by load Select knob Set by load Select knob Agitate 20 minutes 15 minutes 10 minutes 5 minutes Drain 5 minutes 5 minutes 5 minutes 5 minutes Fill water Set by load Select knob Set by load Select knob Set by load Select knob Set by load Select knob Agitate 10 minutes 10 minutes 5 minutes 5 minutes

Drain 5 minutes 5 minutes 5 minutes 5 minutes Spin 10 minutes 10 minutes 5 minutes 5 minutes

Operation	Signal	Input/output	Port pin no.
Load / water level select	Water level low Water level med Water level high	Input Input Input	P0.0 P0.1 P0.2
Water inlet	Hot water knob Normal water knob	Input Input	P0.3 P0.4
Program select	Heavy Normal Light Dedicate	Input Input Input Input	P1.0 P1.1 P1.2 P1.3
Machine ON	Machine on indic	Output	P2.0
Fill water	Hot water inlet Normal water inlet	Output Output	P2.1 P2.2
Agitation control	Motor rotate in cloc direction Motor rotate in anticlock direc	Output Output	P2.3 P2.4
Drain	Drain valve open	Output	P2.5
Spin	Spin motor ON/OFF	Output	P2.6
Washing ccomplete	Washing comp indic	Output	P2.7

10. Circuit diagram


Washing machine ON
Hot
LED
Buzzer sound
Cold
P2.1
P2.0
P2.7
P2.2
High level
P1.0
Heavy
P0.2
P1.1
Normal
Medium level
P0.1
P1.2
Light
P0.0
P1.3
Delicated
Basket
8051
Microcontroller
P2.3
Low
level
P2.4
Drain
P2.6
P0.3
Hot
P2.5
P0.4
Normal


Put machine ON
Fill machine with water hot or normal
Check program setting
Agitate 20 min
Drain 5 min
Fill water
Agitate 10 min
Drain 5 min
Spin 20 min
Buzzer for wash complete
Agitate 15 min
Drain 5 min
Fill water
Agitate 10 min
Drain 5 min
Spin 10 min
Buzzer for wash complete
Agitate 10 min
Drain 5 min
Fill water
Agitate 5 min
Drain 5 min
Spin 5 min
Buzzer for wash complete
Agitate 5 min
Drain 5 min
Fill water
Agitate 5 min
Drain 5 min
Spin 5 min
Buzzer for wash complete

Labels	Mnemonics	Operands	Comments
	SETB LCALL JNB SJMP	P2.0 FILL_1 P1.0,LOOP_1 HEAVY	Machine ON indication Machine fill with water 1st time Chk prog setng knob for heavy. if P1.0 is not set,jump to LOOP_1 If P1.0 is set,jump to HEAVY
LOOP_1	JNB SJMP	P1.1,LOOP_2 NORMAL	Check prog setng knob for normal.if P1.1 is not set,jump to LOOP_2 If P1.1 is set, jump to NORM
LOOP_2	JNB SJMP	P1.2,LOOP_3	Chck prog setng knob for normal.if P1.2 is not set,jump to LOOP_3 If P1.2 is set,jump to LIGHT
LOOP_3	JNB SJMP	P1.3,LOOP_4 DELICATE	Check prog set knob for delicate. If P1.3is not set,jump to LOOP_4 If P1.2 is set,jump to delicate
DISPLAY	SETB	P2.7	Indicate the completion of wash cycle.
LOOP_4	NOP LJMP	 0000	 End of program