MODULE - 3

8051 ARCHITECTURE

MODULE-3

8051 Architecture

8051 - organization and architecture, RAM-ROM organization, Machine cycle

INTRODUCTION TO INTEL MCS 51 µC

INTEL MCS 51 MICROCONTROLLER

- The Intel MCS-51 (commonly termed 8051) is a Harvard architecture, complex instruction set computing (CISC) Architecture, single chip microcontroller (μC) series developed by Intel in 1980 for use in embedded systems
- Intel's original MCS-51 family was developed using N-type metal-oxide-semiconductor (NMOS) technology
- But later versions, identified by a letter C in their name (e.g., 80C51) used complementary metal-oxide-semiconductor (CMOS) technology and consume less power

INTEL MCS 51 MICROCONTROLLER

FEATURES

- 8-bit CPU
- 64K bytes on-chip program memory (ROM)
- 128 bytes on-chip data memory (RAM)
- 32 I/O pins arranged as four 8-bit ports (P0 - P3)
- 32 general purpose registers each of 8-bit
- Special Function Registers (SFRs) of 128 bytes
- 16-bit Program Counter
- 8-bit Processor Status Word (PSW) & Stack Pointer
- Two 16-bit timer/counters : T0 and T1
- Two external and three internal vectored interrupts
- One full duplex serial I/O (UART)

INTEL MCS 51 MICROCONTROLLER

8051 FAMILY SERIES

Feature	8051	8052	8031
ROM(bytes)	4K	8K	0K
RAM(bytes)	128	256	128
Timers	2	3	2
I/O pins	32	32	32
Serial port	1	1	1
Interrupt sources	6	8	6

INTEL MCS 51 MICROCONTROLLER

8051 FAMILY SERIES

- Although the 8051 is the most popular member of the 8051 family, you will not see "8051" in the part number.
- This is because the 8051 is available in different memory types, such as UV-EPROM, flash, and NV-RAM, all of which have different part numbers.
- The UV-EPROM version of the 8051 is the 8751. The flash ROM version is marketed by many companies including Atmel Corp. and Dallas Semiconductor.

INTEL MCS 51 MICROCONTROLLER

8051 FAMILY SERIES

- The Atmel Flash 8051 is called AT89C51, while Dallas Semiconductor calls theirs DS89C4xO (DS89C420/430/440).
- The NV-RAM version of the 8051 made by Dallas Semiconductor is called DS5000.
- There is also an OTP (one-time programmable) version of the 8051 made by various manufacturers.

INTEL MCS 51 MICROCONTROLLER

8051 MAJOR MANUFACTURERS

- ATMEL
- ANALOG DEVICES
- ST MICROELECTRONICS
- DALLAS
- MAXIM
- SILICON LABS
- TEXAS INSTRUMENTS
- MICROCHIP
- ZILOG

INTEL MCS 51 MICROCONTROLLER

8051 ATMEL SERIES

Part Number	ROM	RAM	I/O pins	Timer	Interrupt	Vcc	Packaging
AT89C51	4K	128	32	2	6	5V	40
AT89C52	8K	256	32	3	8	5V	40
AT89C1051	1K	64	15	1	3	3V	20
AT89C2051	2K	128	32	3	8	3V	20
AT89LV51	4K	128	32	2	6	3V	40
AT89LV52	8K	128	32	3	8	3V	40

8051 MICROCONTROLLER PIN DETAILS

8051 MICROCONTROLLER PIN DETAILS

8051 PIN DETAILS

- Although 8051 family members (e.g., 8751, 89C51, 89C52, DS89C4xO) come in different packages, such as DIP (dual in-line package), QFP (quad flat package), and LLC (leadless chip carrier)
- They all have 40 pins that are dedicated to various functions such as I/O, RD, WR, address, data, and interrupts.
- Some companies provide a 20-pin version of the 8051 with a reduced number of I/O ports for less demanding applications.

8051 MICROCONTROLLER PIN DETAILS

8051 PIN DETAILS

- However, since the vast majority of developers use the 40-pin chip, we will concentrate on that.
- In 40 pin package, a total of 32 pins are set aside for the four ports PO, P1, P2, and P3, where each port takes 8 pins.
- The rest of the pins are designated as Vcc, GND, XTAL1, XTAL2, RST, EA, PSEN, and ALE.

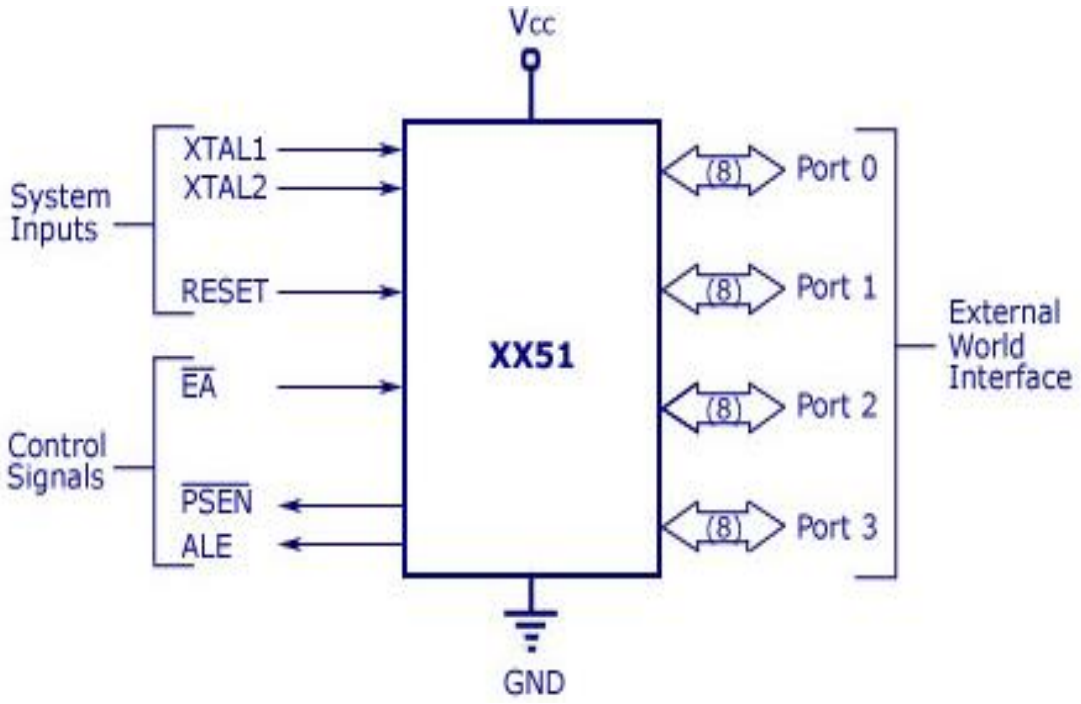
8051 MICROCONTROLLER PIN DETAILS

8051 PIN DETAILS

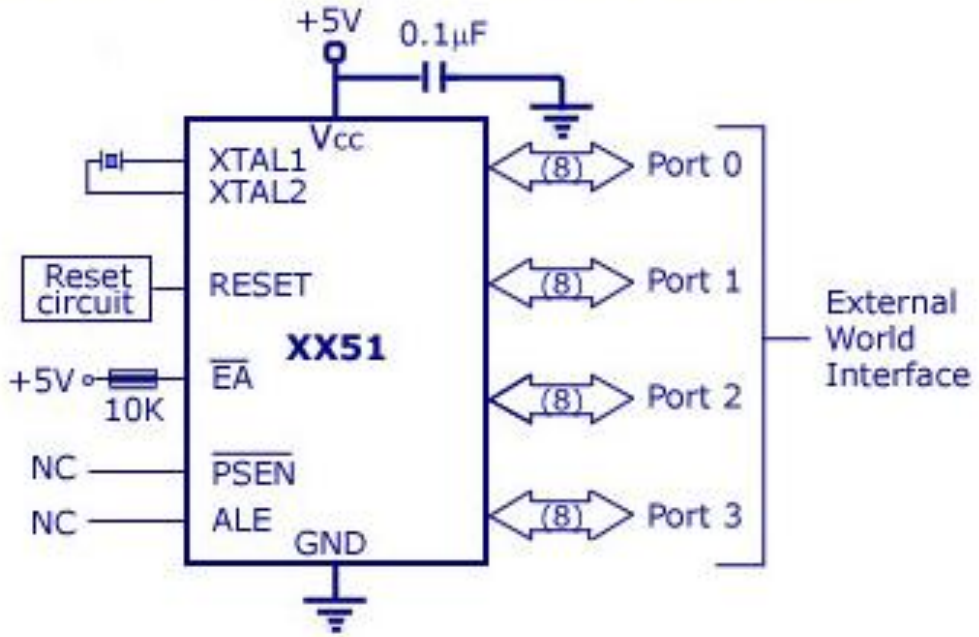
- Of these pins, six (Vcc, GND, XTAL1, XTAL2, RST, and EA) are used by all members of the 8051 and 8031 families.
- In other words, they must be connected in order for the system to work, regardless of whether the microcontroller is of the 8051 or 8031 family.
- The other two pins, PSEN and ALE, are used mainly in 8031-based systems.

8051 MICROCONTROLLER PIN DETAILS

8051 SCHEMATICS

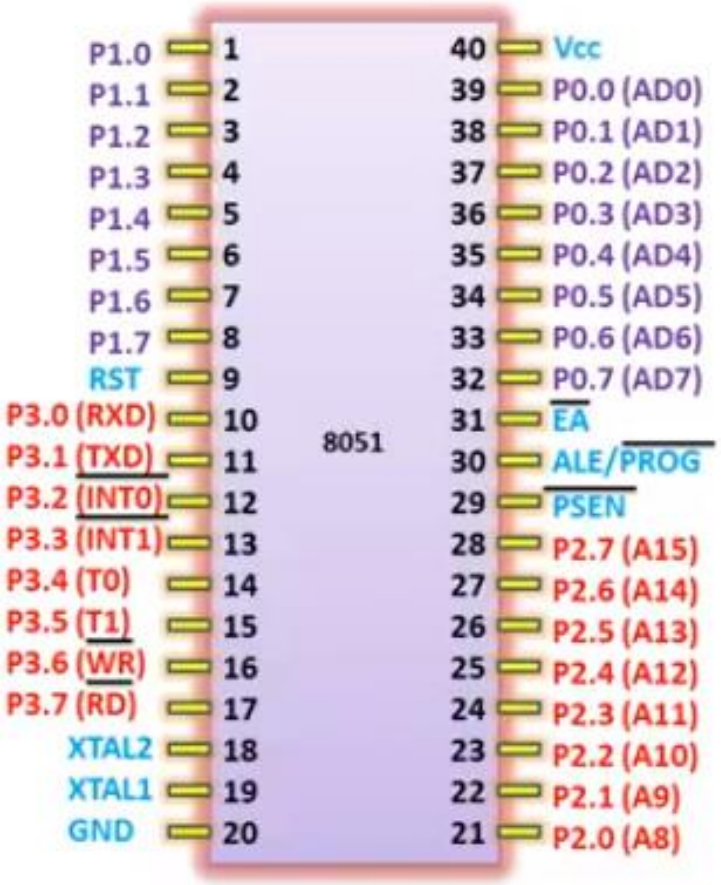


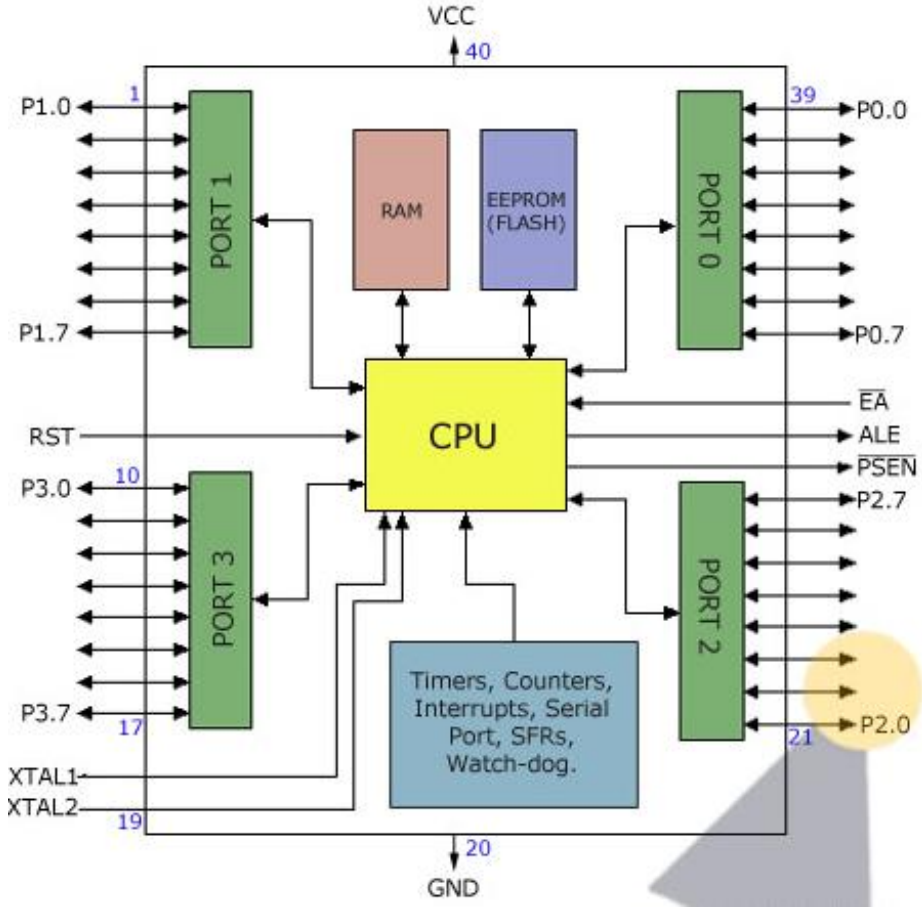
Schematic System Inputs for Stand Alone Operation



8051 MICROCONTROLLER PIN DETAILS

8051 PIN DIAGRAM





8051 MICROCONTROLLER PIN DETAILS

8051 PIN DESCRIPTION

- Pin-40 : Vcc is the main power source. Usually its +5V DC.

- Pins 32-39: Port 0 (P0.0 to P0.7) – In addition to serving as I/O port, lower order address and data bus signals are multiplexed with this port (to serve the purpose of external memory interfacing). This is a bi directional I/O port (the only one in 8051) and external pull up resistors are required to function this port as I/O.

8051 MICROCONTROLLER PIN DETAILS

- **Pin-31:- ALE - Address Latch Enable.** It is especially used for 8031 IC to connect it to the external memory. It can be used while deciding whether P0 pins will be used as Address bus or Data bus. When ALE = 1, then the P0 pins work as Data bus and when ALE = 0, then the P0 pins act as Address bus.

- **Pin-30:- EA - External Access** input is used to enable or disallow external memory interfacing. If there is no external memory requirement, this pin is pulled high by connecting it to Vcc.

8051 MICROCONTROLLER PIN DETAILS

- Pin- 29:- PSEN or Program Store Enable. This is an active low pin, i.e., it gets activated after applying a low pulse. It is an output pin and used along with the EA pin in 8031 based (i.e. ROMLESS) Systems to allow storage of program code in external ROM.

- Pins- 21-28: Port 2 (P 2.0 to P 2.7) – in addition to serving as I/O port, higher order address bus signals are multiplexed with this quasi bi directional port.

- Pin 20:- Vss – it represents ground (0 V) connection.

8051 MICROCONTROLLER PIN DETAILS

- **Pins 18 and 19:- XTAL1 & XTAL2** Used for interfacing an external crystal to provide system clock.

- **Pins 10 – 17:- Port 3 (P 3.0 to P 3.7)** It is also of 8 bits and can be used as Input/Output. This port provides some extremely important signals. P3.0 and P3.1 are RxD (Receiver) and TxD (Transmitter) respectively and are collectively used for Serial Communication. P3.2 and P3.3 pins are used for external interrupts. P3.4 and P3.5 are used for timers T0 and T1 respectively. P3.6 and P3.7 are Write (WR) and Read (RD) pins.

8051 MICROCONTROLLER PIN DETAILS

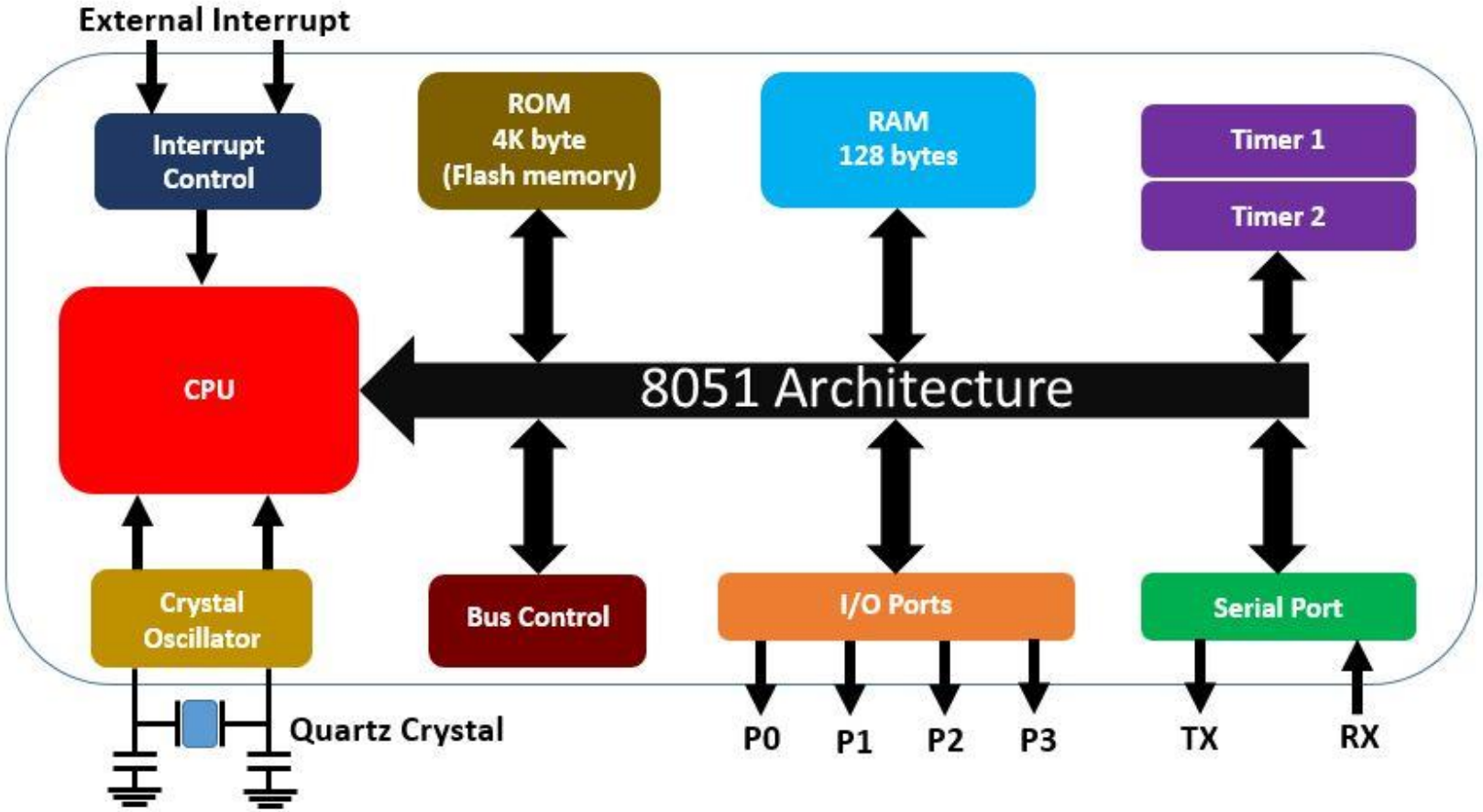
- **Pin 9:- RESET** pin is used to set the 8051 microcontroller to its initial values, while the microcontroller is working or at the initial start of application. The RESET pin must be set high for 2 machine cycles.

- **Pins 1 – 8:- Port 1 (P 1.0 to P 1.7)**. It is an 8-bit port (pin 1 through 8) and can be used either as input or output. Unlike other ports, this port does not serve any other functions. Port 1 is an internally pulled up, quasi bi directional I/O port.

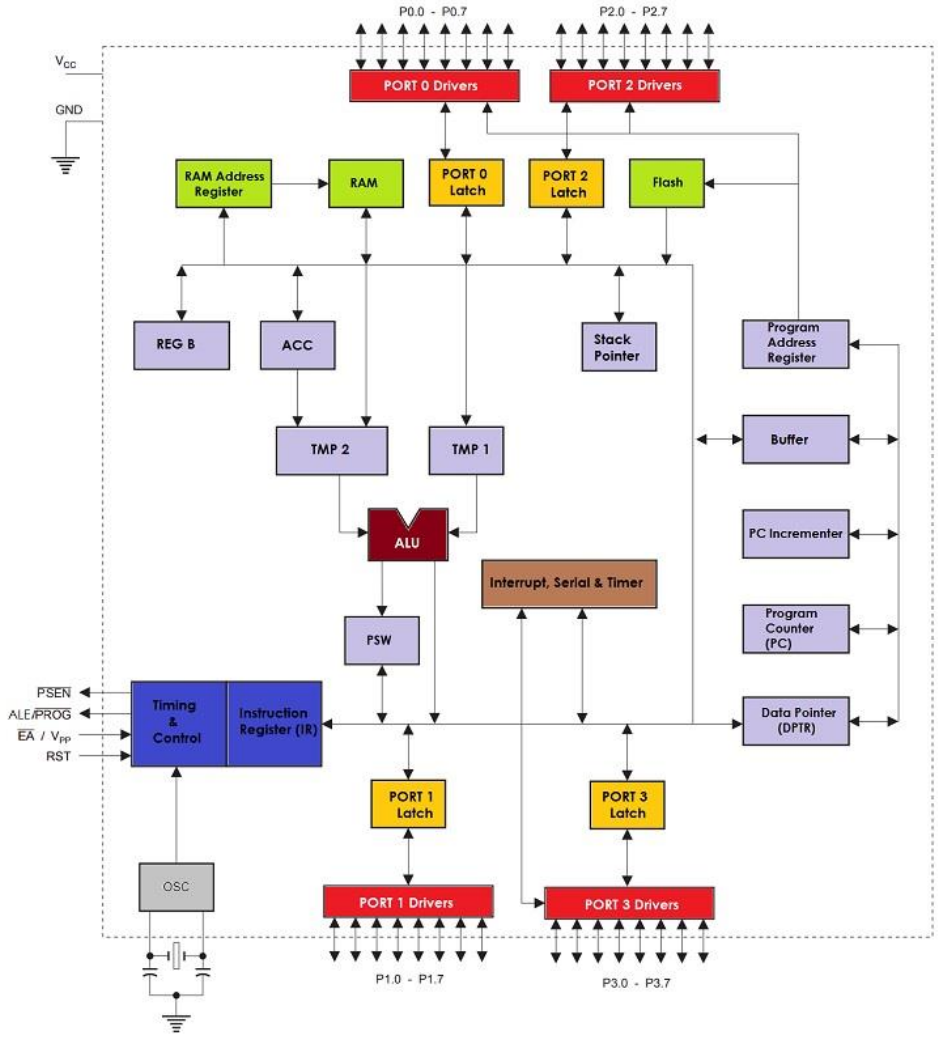
8051 MICROCONTROLLER ARCHITECTURE

8051 MICROCONTROLLER ARCHITECTURE

8051 ARCHITECTURE (Simplified)



8051 MICROCONTROLLER ARCHITECTURE



8051 ARCHITECTURE (Detailed)

8051 MICROCONTROLLER ARCHITECTURE

CENTRAL PROCESSING UNIT (CPU)

- Central Processor Unit (CPU) is the brain of any processing device of the microcontroller.
- It monitors and controls all operations that are performed on the Microcontroller units.
- The User has no control over the work of the CPU directly.
- It reads program written in ROM memory and executes them and do the expected task of that application.

8051 MICROCONTROLLER ARCHITECTURE

INTERRUPT CONTROL

- There are five interrupt sources in 8051 Microcontroller and **interrupt control** section control all these interrupts.
- Two external interrupts (INT0 & INT1), two timer (TF0 & TF1) interrupts and one serial port (RI / TI) interrupt.
- The Microcontroller 8051 can be configured in such a way that it temporarily terminates or pause the main program at the occurrence of interrupt. When subroutine is completed then the execution of main program starts as usual.

8051 MICROCONTROLLER ARCHITECTURE

RAM & ROM

- Microcontroller 8051 has 4K of Code Memory or Program memory that is it has **4KB ROM** and **RAM of 128 bytes.**
- The memory which is used to store the program of Microcontroller, is known as code memory or Program memory . It is known as 'ROM'(Read Only Memory).
- Microcontroller also requires a memory to store data or operands temporarily. This memory is known as Data Memory and we use 'RAM'(Random Access Memory) for this purpose.

8051 MICROCONTROLLER ARCHITECTURE

BUS CONTROL

- **Bus control** section of 8051 is responsible for controlling the operation of address and data bus.
- Bus: Basically Bus is a collection of wires which work as a communication channel or medium for transfer of Data.
- Buses are of two types:
  - Address Bus: Microcontroller 8051 has a 16 bit address bus. It used to address memory locations.
  - Data Bus: Microcontroller 8051 has 8 bits data bus. It is used to carry data.

8051 MICROCONTROLLER ARCHITECTURE

CRYSTAL OSCILLATOR

- **Crystal Oscillator:** Since Microcontroller is a digital circuit device, therefore it requires clock for its operation.
- For this purpose, Microcontroller 8051 has oscillator circuitry section which works as a clock source for Central Processing Unit.
- As the output pulses of oscillator are stable therefore it enables synchronized work of all parts of 8051 Microcontroller.

8051 MICROCONTROLLER ARCHITECTURE

I/O PORTS

- **I/O Ports:** To connect any external devices or peripherals we require I/O interfacing ports in the microcontroller.
- All 8051 microcontrollers have 4 I/O ports each comprising 8 bits which can be configured as input (1) or an output (0), depends on its logic state.
- Accordingly, in total of 32 input/output pins enabling the microcontroller to be connected to peripheral devices are available for use.

8051 MICROCONTROLLER ARCHITECTURE

TIMERS/COUNTERS

- **Timers/Counters**: 8051 Microcontroller 8051 two 16-bit timers and counters: Timer 0 and Timer 1.

- They can be used either as timers to generate a time delay or as counters to count events happening outside the microcontroller.

- Since the 8051 has an 8-bit architecture, each 16-bit is accessed as two separate registers of low byte and high byte.

8051 MICROCONTROLLER ARCHITECTURE

SERIAL PORT

- **Serial port**: The 8051 contains one Serial port or UART (Universal Asynchronous Receiver Transmitter)
- The serial port is full-duplex so, it can transmit and receive simultaneously
- Two port pins are used to provide the serial interface P3.0 is the receive pin (RXD) P3.1 is the transmit pin (TXD)
- This serial port that can be programmed to operate in one of four different modes and at a range of frequencies.

8051 MEMORY ORGANIZATION

(RAM-ROM ORGANIZATION)

8051 MEMORY ORGANIZATION

- The 8051 has two types of memory and these are Program Memory and Data Memory.
- Program Memory (ROM) is used to permanently save the program being executed, while Data Memory (RAM) is used for temporarily storing data and intermediate results created and used during the operation of the microcontroller.
- Depending on the model in use (8051 microcontroller family in general) at most a few Kb of ROM and 128 or 256 bytes of RAM is used.

8051 MEMORY ORGANIZATION

PROGRAM MEMORY (ROM)

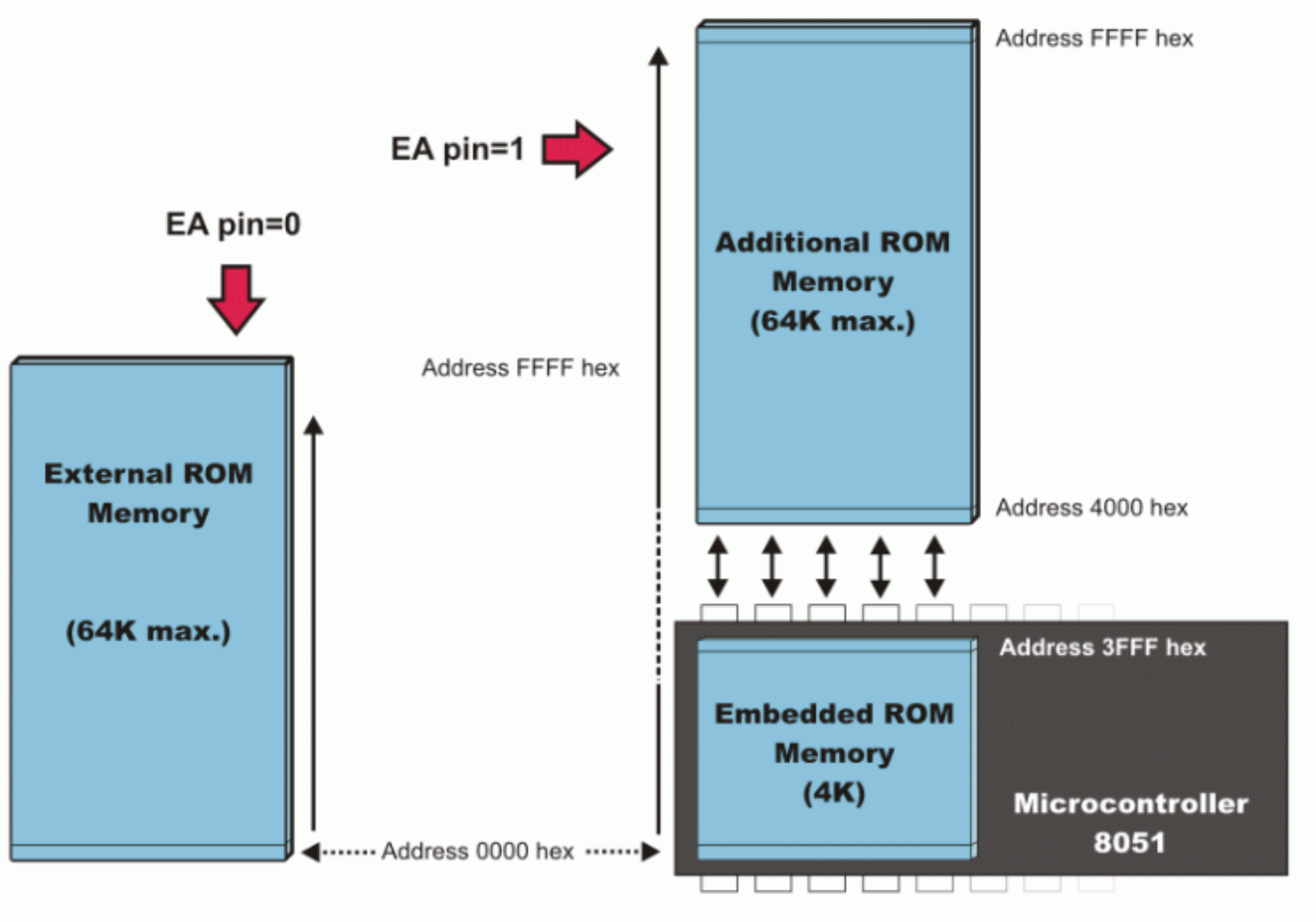
- The 8051 has 4K (4096 locations) of internal or on-chip ROM. And it can be expanded up to 64K.
- This is used for storing the system program. $2^{12} = 4096$, therefore the internal ROM locations go from 0000H to 0FFFH.
- Even though such an amount of memory is sufficient for writing most of the programs, there are situations when it is necessary to use additional memory as well.

8051 MEMORY ORGANIZATION

PROGRAM MEMORY (ROM)

- EA=0 In this case, the microcontroller completely ignores internal program memory and executes only the program stored in external memory.

- EA=1 In this case, the microcontroller executes first the program from built-in ROM, then the program stored in external memory



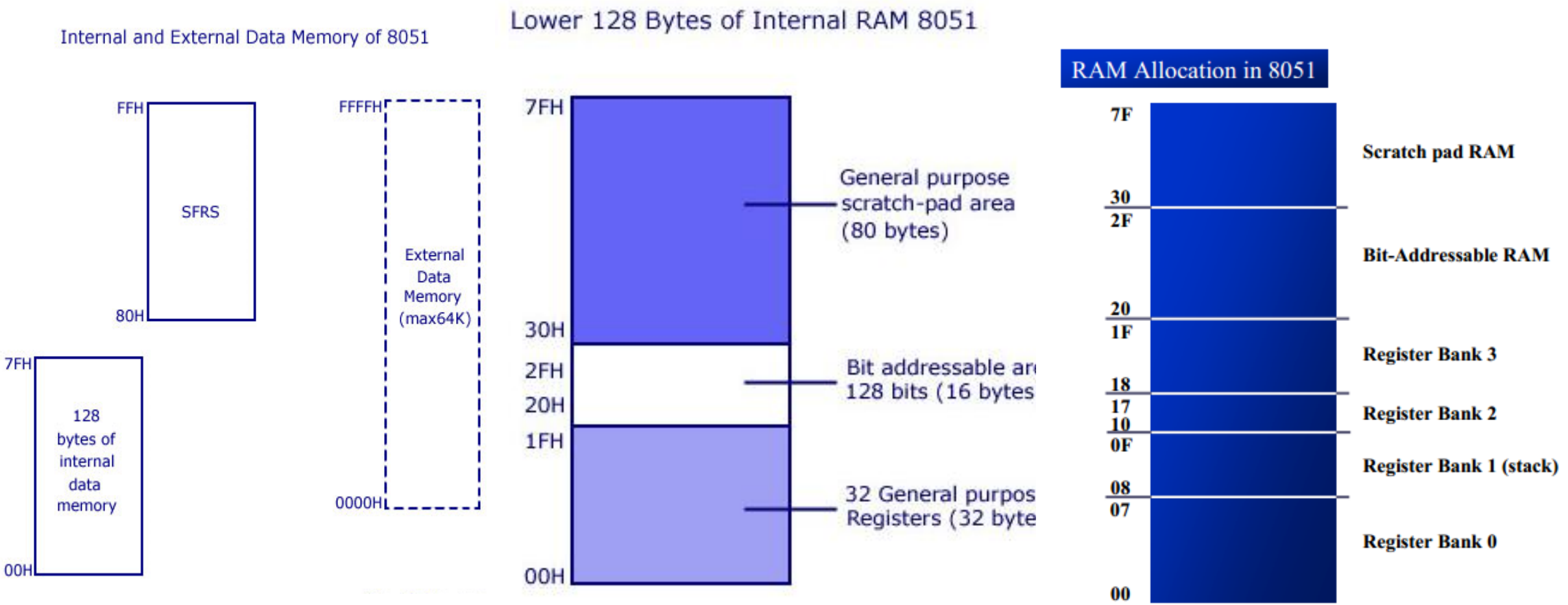
8051 MEMORY ORGANIZATION

DATA MEMORY (RAM)

- In the MCS-51 family, 8051 has 128 bytes of internal data memory and it allows interfacing external data memory of maximum size up to 64K.
- So the total size of data memory in 8051 can be upto 64K (external) + 128 bytes (internal).
- So there are 3 separations/divisions of the data memory:-
  1) Register banks
  2) Bit addressable area
  3) Scratch pad area

8051 MEMORY ORGANIZATION

DATA MEMORY (RAM)



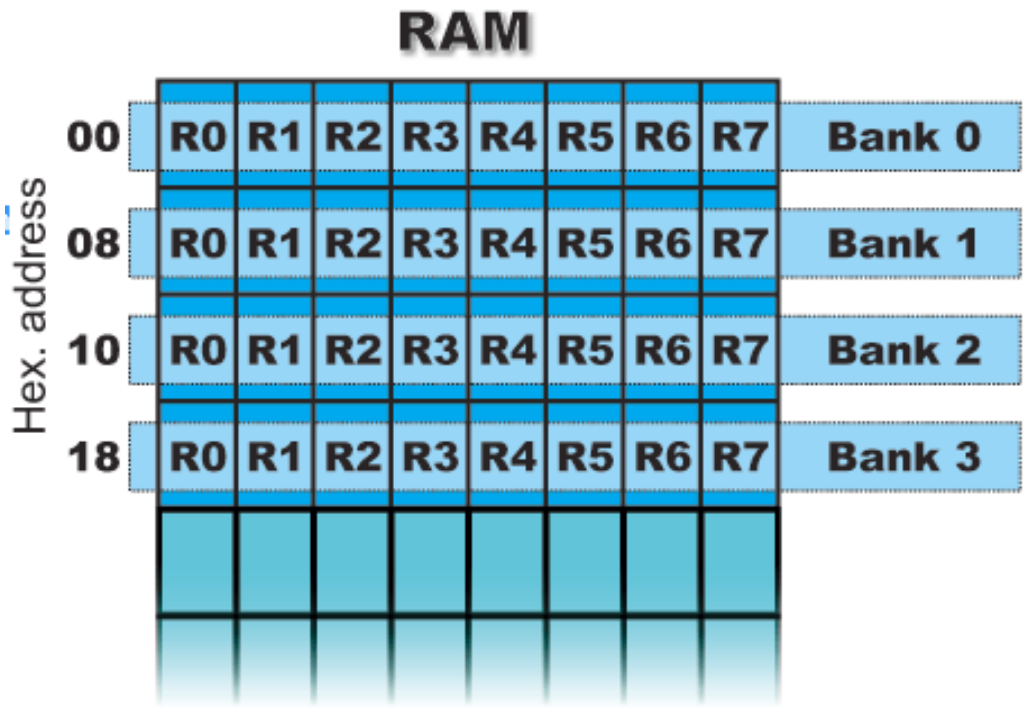
8051 MEMORY ORGANIZATION

DATA MEMORY – REGISTER BANKS

- Registers are used to store data or operands during executions. Register banks form the lowest 32 bytes on internal RAM memory.
- There are 4 register banks designated bank #0,#1, #2 and #3. Each bank has 8 registers which are designated as R0,R1…R7.
- At a time only one register bank is selected (using RS1 & RS0 bits in PSW register) for operations and the registers inside the selected bank are accessed using mnemonics R0..R1.. etc.
- By default register bank #0 is selected (after a system reset).

8051 MEMORY ORGANIZATION

DATA MEMORY – REGISTER BANKS



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

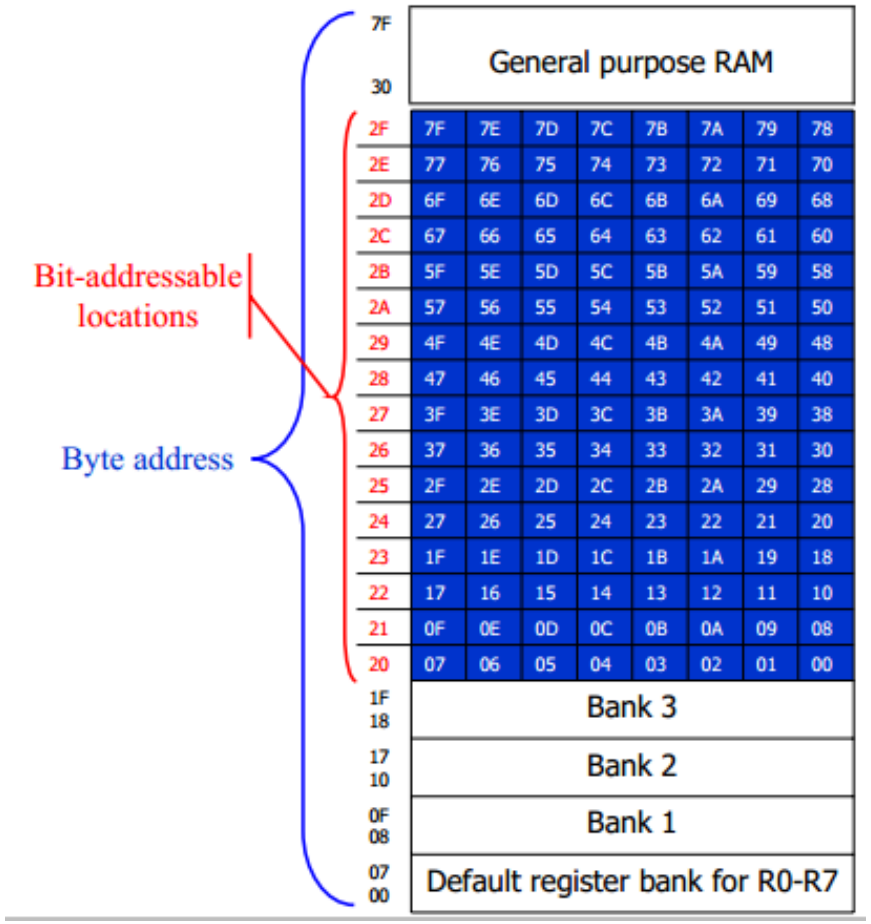
8051 MEMORY ORGANIZATION

DATA MEMORY – BIT ADDRESSABLE AREA

- The 8051 supports a special feature which allows access to bit variables. This is where individual memory bits in Internal RAM can be set or cleared.
- The Bit Addressable area of the RAM is 16 bytes (128 bits) next to register banks of Internal RAM located between 20h and 2Fh. In all there are 128 bits numbered 00h to 7Fh.
- Being bit variables any one variable can have a value 0 or 1. A bit variable can be set with a command such as SETB and cleared with a command such as CLR.

8051 MEMORY ORGANIZATION

DATA MEMORY – BIT ADDRESSABLE AREA



8051 MEMORY ORGANIZATION

DATA MEMORY – BIT ADDRESSABLE AREA

- Example instructions are

```
SETB 25h        ; sets the bit 25h (becomes 1)
CLR 25h         ; clears bit 25h (becomes 0)
```

- Bit addressable area is mainly used to store bit variables from application program, like status of an output device like LED or Motor (ON/OFF) etc.

- We need only a bit to store this status and using a complete byte addressable area for storing this is really bad programming practice, since it results in wastage of memory.

8051 MEMORY ORGANIZATION

DATA MEMORY – SCRATCH PAD RAM

- These 80 bytes of Internal RAM memory scratch pad RAM are available for general-purpose data storage. Scratch pad RAM is from 30H to 7FH and this includes stack too.
- However, these 80 bytes are used by the system stack and in practice little space is left for general storage.
- Access to this area of memory is fast compared to access to the main memory and special instructions with single byte operands are used.

8051 MEMORY ORGANIZATION

DATA MEMORY – SCRATCH PAD RAM

- The scratch pad RAM can be accessed using direct or indirect addressing modes.

- Examples of direct addressing:

```
MOV A, 6Ah        ;reads contents of address 6Ah to accumulator
```

- Examples for indirect addressing (use registers R0 or R1):

```
MOV R1, #6Ah      ; move immediate 6Ah to R1
MOV A, @R1        ; move indirect: R1 contains address of Internal
                    RAM which contains data that is moved to A.
```

8051 MEMORY ORGANIZATION

DATA MEMORY – SFRs (Special Function Registers)

- SFRs are accessed just like normal Internal RAM locations.
- The SFR registers are located within the Internal Memory in the address range 80h to FFh.
- Each SFR has a very specific function. Note some of the SFR registers are bit addressable.
- Each SFR has an address (within the range 80h to FFh) and a name which reflects the purpose of the SFR.

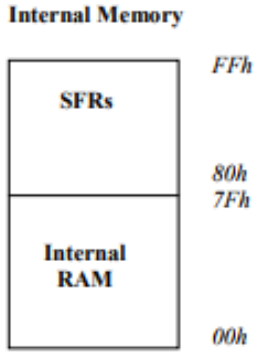
8051 MEMORY ORGANIZATION

DATA MEMORY – SFRs

- Although 128 byes of the SFR address space is defined only 21 SFR registers are defined in the standard 8051.
- Rest of locations are intentionally left unoccupied in order to enable the manufacturers to further develop microcontrollers keeping them compatible with the previous versions.
- Main function of SFR is to control timers, counters, serial I/O, port I/O, and peripherals that are present in 8051 microcontroller.

8051 MEMORY ORGANIZATION

Byte address	Bit address b7 b6 b5 b4 b3 b2 b1 b0
FFh	
F0h	B *
E0h	A (accumulator) *
D0h	PSW *
B8h	IP *
B0h	Port 3 (P3) *
A8h	IE *
A0h	Port 2 (P2) *
99h	SBUF
98h	SCON *
90h	Port 1 (P1) *
8Dh	TH1
8Ch	TH0
8Bh	TL1
8Ah	TL0
89h	TMOD
88h	TCON *
87h	PCON
83h	DPH
82h	DPL
81h	SP
80h	Port 0 (P0) *



* indicates the SFR registers which are bit addressable

DATA MEMORY – SFRs register layout

8051 MEMORY ORGANIZATION

DATA MEMORY – SFRs

Name of the Register	*Function*	*Internal RAM Address (HEX)*
ACC	Accumulator	E0H
B	B Register (for Arithmetic)	F0H
DPH	Addressing External Memory	83H
DPL	Addressing External Memory	82H
IE	Interrupt Enable Control	A8H
IP	Interrupt Priority	B8H
P0	PORT 0 Latch	80H
P1	PORT 1 Latch	90H
P2	PORT 2 Latch	A0H
P3	PORT 3 Latch	B0H
PCON	Power Control	87H
PSW	Program Status Word	D0H
SCON	Serial Port Control	98H
SBUF	Serial Port Data Buffer	99H
SP	Stack Pointer	81H
TMOD	Timer / Counter Mode Control	89H
TCON	Timer / Counter Control	88H
TL0	Timer 0 LOW Byte	8AH
TH0	Timer 0 HIGH Byte	8CH
TL1	Timer 1 LOW Byte	8BH
TH1	Timer 1 HIGH Byte	8DH

8051 MEMORY ORGANIZATION

SFRs A- Register

- The 21 Special Function Registers of 8051 Microcontroller are categorized in to seven groups.

1) Math or CPU Registers: A and B
2) Status Register: PSW (Program Status Word)
3) I/O Port Latches: P0 (Port 0), P1 (Port 1), P2 (Port 2) and P3 (Port 3)
4) Pointer Registers: DPTR (Data Pointer – DPL,DPH), SP (Stack Pointer)
5) Peripheral Control Registers: PCON, SCON, TCON, TMOD, IE and IP
6) Peripheral Data Registers: TL0, TH0, TL1, TH1 and SBUF

8051 MEMORY ORGANIZATION

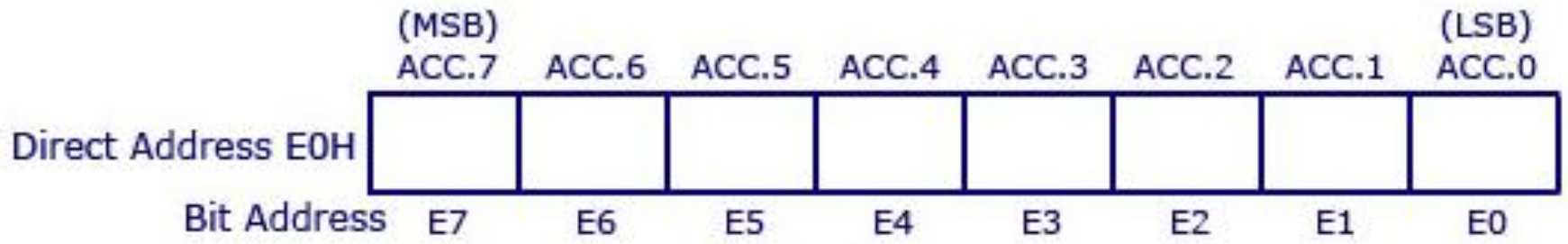
SFRs A- Register

- The most important of all special function register is Accumulator which is also known as ACC or A.
- The Accumulator holds the result of most of arithmetic and logic operations. It is also used to store 8 bit data and to hold one of operand of ALU units during arithmetical and logical operations.
- More than half instructions used by the 8051 microcontroller use somehow use the accumulator.
- ACC is usually accessed by direct addressing and its physical address is E0H.

8051 MEMORY ORGANIZATION

SFRs A- Register

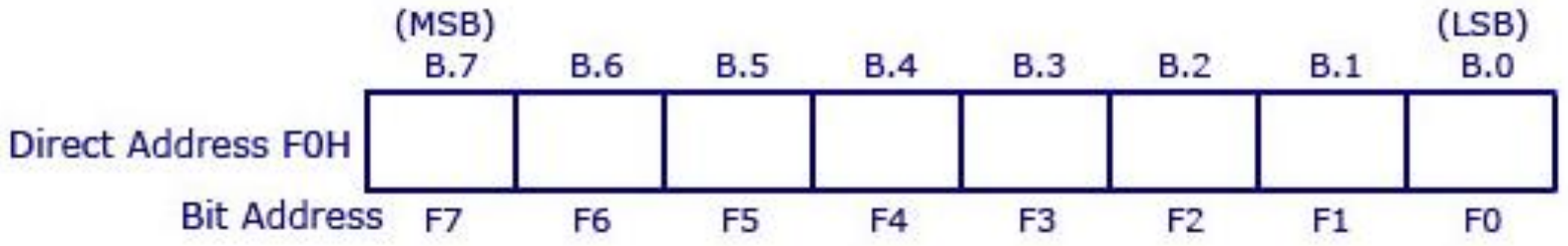
- Accumulator is an 8-bit register and it is both byte and bit addressable.
- To access the individual bits of accumulator, use the format ACC.X in the instruction where "X" denotes bit to be accessed.



8051 MEMORY ORGANIZATION

SFRs B- Register

- It is special 8-bit math register and it is bit and byte accessible.
- It is used in conjunction with A register as an input operand for ALU to perform multiplication and division operation.
- It can also be used as general purpose register to store 8-bit data.



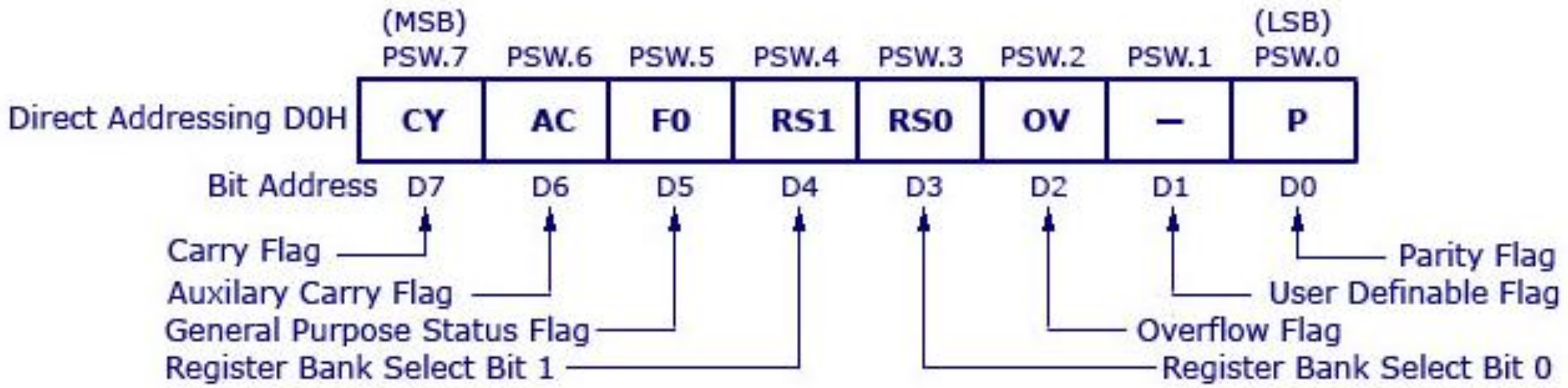
8051 MEMORY ORGANIZATION

SFRs - PSW REGISTER

- It is 8 bit register and it is bit and byte accessible.
- It contains several status bits that reflects the status of the operation that is being carried out in the processor.
- It has 4 conditional flags or math flags (CY, AC, OV, P)which sets or resets according to condition of result.
- It has 3 control flags (F0, RS0, RS1) by setting or resetting bit required operation or function can be achieved.

8051 MEMORY ORGANIZATION

SFRs - PSW REGISTER



8051 MEMORY ORGANIZATION

SFRs - PSW REGISTER

- CY, the carry flag: This flag is set whenever there is a carry out from the D7 bit. This flag bit is affected after an 8-bit addition or subtraction.

- AC, the auxiliary carry flag: If there is a carry from D3 to D4 during an ADD or SUB operation, this bit is set; otherwise, it is cleared. This flag is used by instructions that perform BCD (binary coded decimal) arithmetic.

- F0, the Flag 0 : The PSW.5 and PSW.1 bits are general-purpose status flag bits and can be used by the programmer for any purpose. In other words, they are user definable.

8051 MEMORY ORGANIZATION

SFRs - PSW REGISTER

- RS0, RS1 - Register bank select bits. These two bits are used to select one of four register banks of RAM. By setting and clearing these bits, registers R0-R7 are stored in one of four banks of RAM.

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

8051 MEMORY ORGANIZATION

SFRs - PSW REGISTER

- OV, overflow flag:
  - This flag is set whenever the result of a signed number operation is too large, causing the high-order bit to overflow into sign bit.
  - In general, the carry flag is used to detect errors in unsigned arithmetic operations.
  - The overflow flag is only used to detect errors in signed arithmetic operations

- P, the parity flag: The parity flag reflects the number of 1 s in the A (accumulator) register only. If the A register contains an odd number of 1's, then P = 1. Therefore, P = 0 if A has an even number of 1s.

8051 MEMORY ORGANIZATION

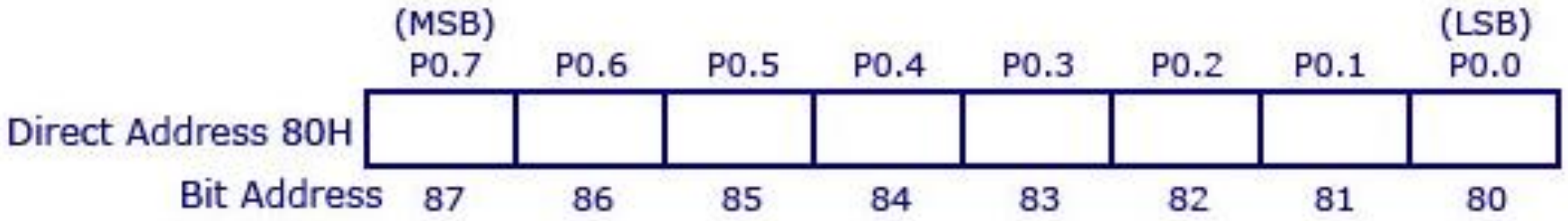
SFRs - P0, P1, P2, P3 - Input/Output Registers

- There are 4 ports with in total of 32 input/output pins are available for connection to peripheral environment.
- So 4 Input/Output ports named P0, P1, P2 and P3 has got four corresponding port registers P0, P1, P2 and P3. All 4 port registers are bit as well as byte addressable.
- 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.

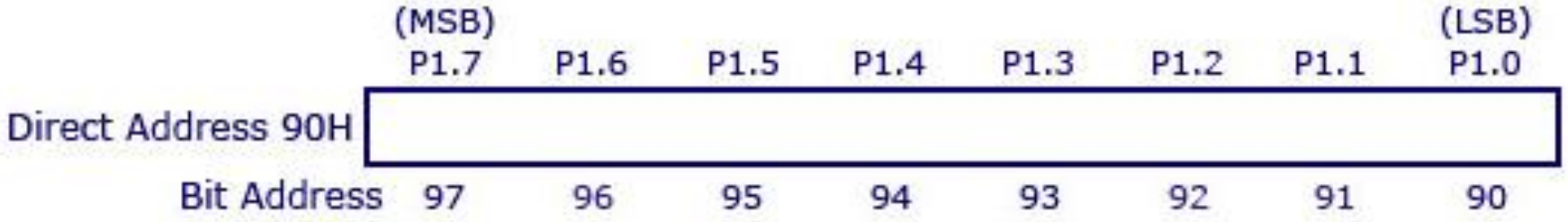
8051 MEMORY ORGANIZATION

SFRs - P0, P1, P2, P3 - Input/Output Registers

Input Output Port P0



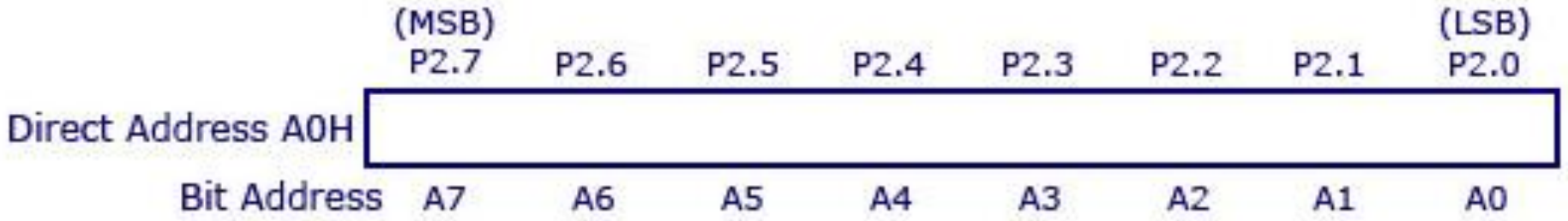
Input Output Port P1



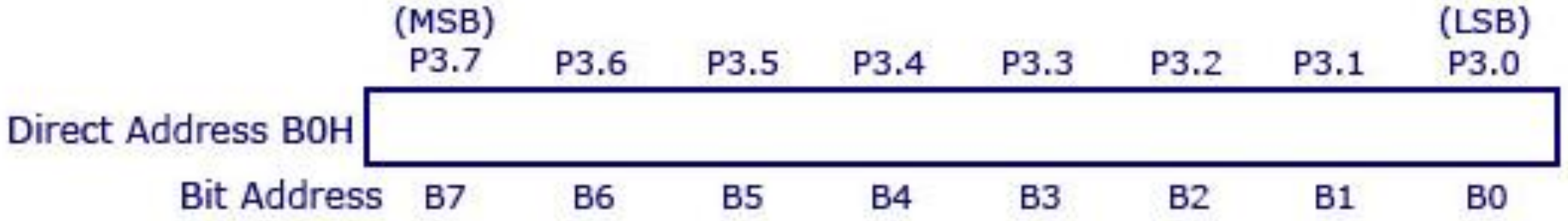
8051 MEMORY ORGANIZATION

SFRs - P0, P1, P2, P3 - Input/Output Registers

Input Output Port P2



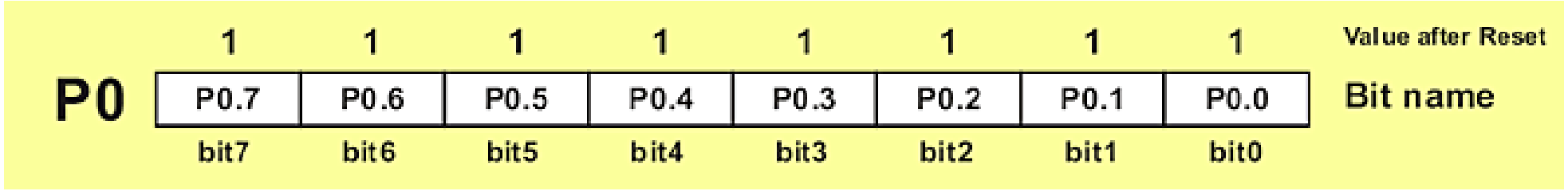
Input Output Port P3



8051 MEMORY ORGANIZATION

SFRs - P0, P1, P2, P3 - Input/Output Registers

- If a bit is cleared (0), the appropriate pin will be configured as an output, while if it is set (1), the appropriate pin will be configured as an input.
- Upon reset and power-on, all port bits are set (1), which means that all appropriate pins will be configured as inputs.



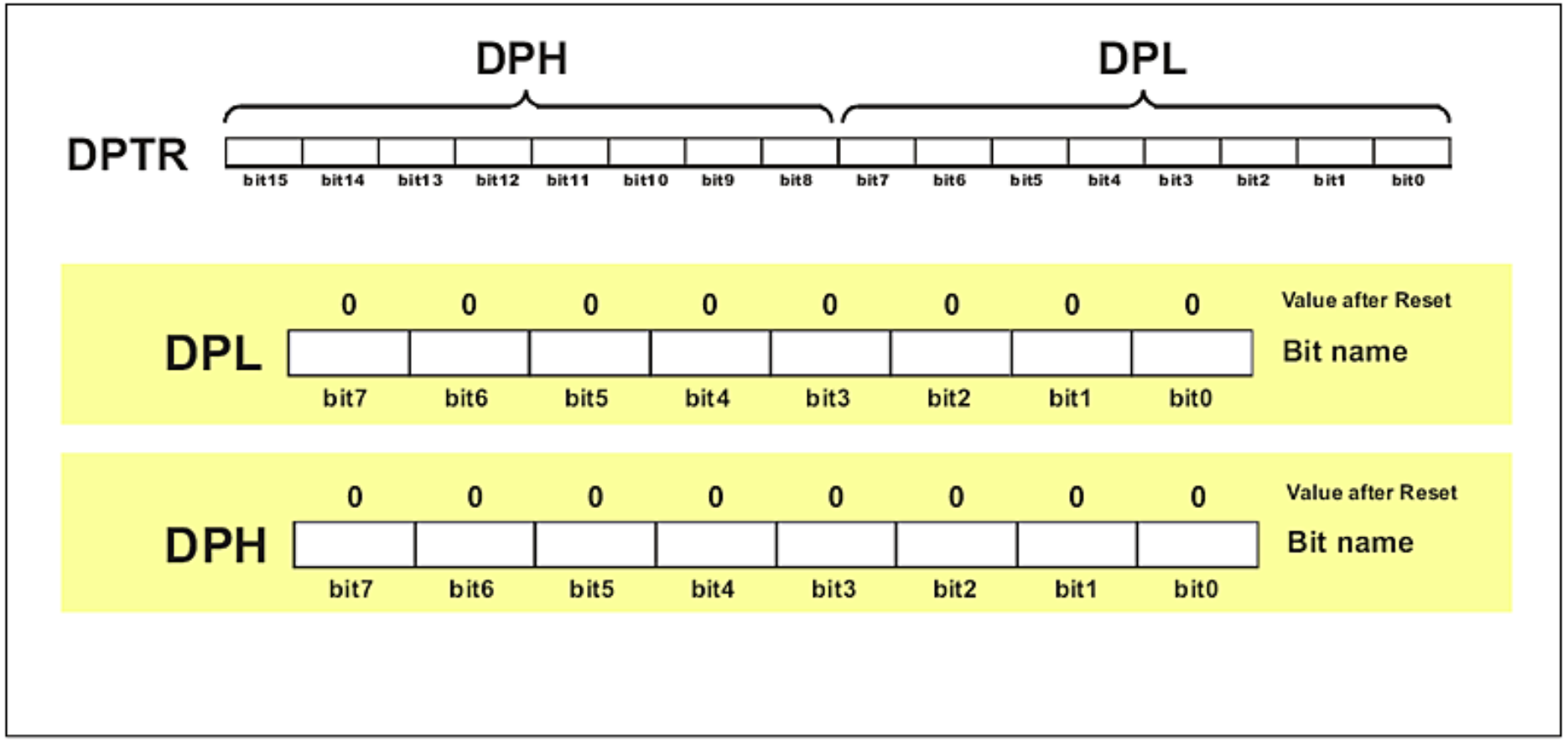
8051 MEMORY ORGANIZATION

SFRs - DPTR REGISTER

- It is a 16 bit register used to hold address of external or internal RAM where data is stored or result is to be stored.
- It can be divided into two 8-bit registers, DPH-data pointer higher order and DPL-data pointer lower order.
- Each register can be used as general purpose register to store 8 bit data and can also be used to store memory location.
- It functions as Base register in base relative addressing mode and indirect jump.

8051 MEMORY ORGANIZATION

SFRs - DPTR REGISTER



8051 MEMORY ORGANIZATION

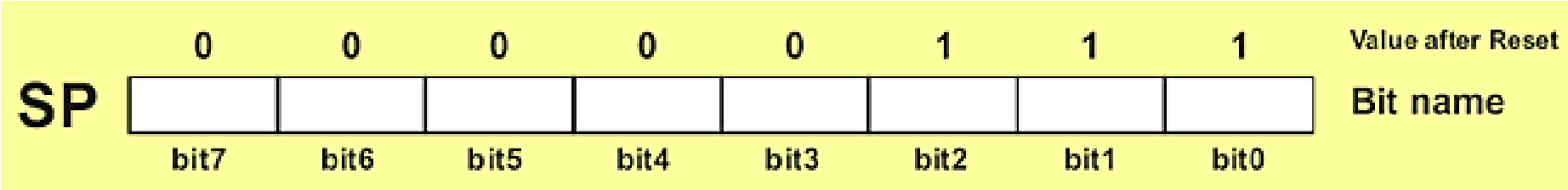
SFRs - STACK POINTER

- It is 8-bit register. It is byte addressable.
- It is used to hold the internal RAM memory location addresses which are used as stack memory.
- When the data is to be placed on stack by push instruction, the content of stack pointer is incremented by 1.
- When data is retrieved from stack, content of stack of stack pointer is decremented by 1.

8051 MEMORY ORGANIZATION

SFRs - STACK POINTER

- A value stored in the Stack Pointer points to the first free stack address and permits stack availability.
- Upon any reset and power-on, the value 7 is stored in the Stack Pointer, which means that the space of RAM reserved for the stack starts at this location.



8051 MEMORY ORGANIZATION

PC - PROGRAM COUNTER (not a part of SFRs)

- It is used to hold 16 bit address of internal RAM, external RAM or external ROM locations.
- The Program Counter (PC) is a 2-byte address which tells the 8051 where the next instruction to be executed from memory.
- When the 8051 is initialized PC always starts at 0000h and is incremented each time an instruction is executed.

8051 MACHINE CYCLE

8051 MACHINE CYCLES

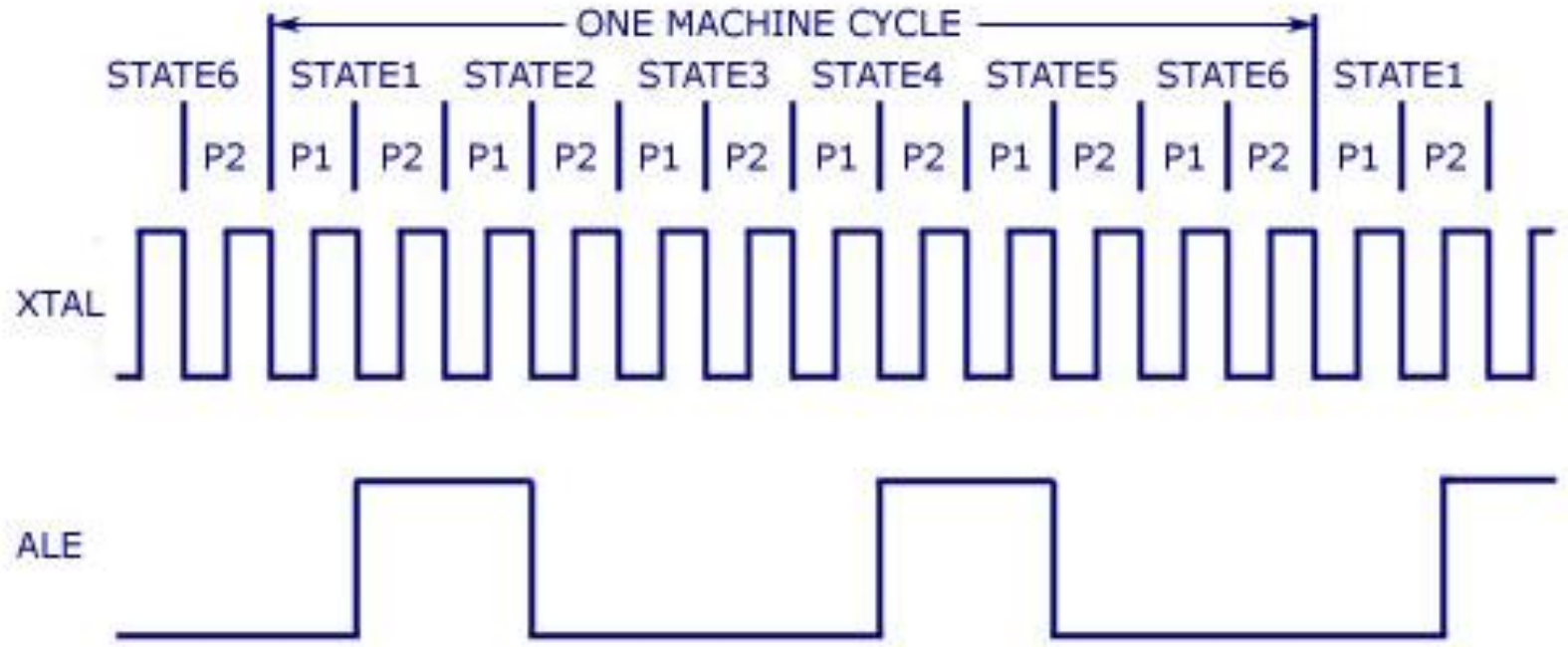
MACHINE CYCLES

- The CPU takes a certain number of clock cycles to execute an instruction.
- In the 8051 family, these clock cycles are referred to as machine cycles.
- A single machine cycle is the minimum amount of time in which a single 8051 instruction can be executed, although many instructions take multiple cycles.

8051 MACHINE CYCLES

MACHINE CYCLES

- Pulse - One complete oscillation of the clock source; State - Two pulses; Machine Cycle - six states.



8051 MACHINE CYCLES

MACHINE CYCLES

- If an instruction takes one machine cycle to execute, it will take 12 pulses of the crystal to execute.
- 8051 is designed to operate between 4MHz to 40MHz and generally operates with a crystal frequency 11.0592 MHz.
- Since we know the crystal is pulsing 11,059,000 times per second and that one machine cycle is 12 pulses. we can calculate how many instruction cycles the 8051 can execute per second:

11,059,000 / 12 = 921,583 and 1/ 921,583=1.085μs

8051 MACHINE CYCLES

MACHINE CYCLES

- This means that the 8051 can execute 921,583 single-cycle instructions per second.
- For example, if you are using exclusively 2-cycle instructions you would find that the 8051 would execute 460,791 instructions per second.
- The 8051 also has two really slow instructions that require a full 4 cycles to execute-those instructions you'd find performance to be about 230,395 instructions per second.

8051 MACHINE CYCLES

EXAMPLE - 1

- Lets find the time period of the machine cycle in each case for the following crystal frequency of different 8051 based systems: 11.0592 MHz, 16 MHz, 20 MHz.

Answer:

11.0592 MHz:

11.0592/12 = 921.6 KHz

Machine cycle = 1/921.6 KHz = 1.085us [us=microsecond]

16 MHz:

16MHz/12 = 1.333 MHz

Machine cycle = 1/1.333 MHz = 0.75us [us=microsecond]

20MHz:

20MHz/12 = 1.66 MHz

Machine Cycle = 1/1.66 MHz = 0.60us [us=microsecond]

8051 MACHINE CYCLES

EXAMPLE - 2

- Lets find how long it takes to execute each of the following instructions, for a crystal frequency of 11.0592 MHz. The machine cycle of a system of 11.0592.

INSTRUCTION	MACHINE CYCLE	TIME TO EXECUTE
MOV R2,#55H	1	1x1.085 us = 1.085 us
DEC R2	1	1x1.085 us = 1.085 us
DJNZ R2,target	2	2x1.085 us = 2.17 us
LJMP	2	2x1.085 us = 2.17 us
SJMP	2	2x1.085 us = 2.17 us
NOP	1	1x1.085 us = 1.085 us
MUL AB	4	4x1.085 us = 4.34 us

MODULE-III

8051 INSTRUCTION SET

Data Processing-Stack, Arithmetic, Logical ; Branching-unconditional, conditional

8051
INSTRUCTION SET

8051 INTRUCTION SET

- The process of writing program for the microcontroller mainly consists of giving instructions (commands) in the specific order in which they should be executed in order to carry out a specific task.

- As electronics cannot "understand" what for example an instruction "if the push button is pressed- turn the light on" means, then a certain number of simpler and precisely defined orders that decoder can recognize must be used.

- All commands are known as INSTRUCTION SET.

8051 INTRUCTION SET

- All microcontrollers compatible with the 8051 have in total of 255 instructions, i.e. 255 different words available for program writing.

- Many instructions are considered to be "different", even though they perform the same operation, so there are only 111 truly different commands.

- For example: ADD A,R0, ADD A,R1, ... ADD A,R7 are instructions that perform the same operation (addition of the accumulator and register). Taking into account that all instructions perform only 53 operations (addition, subtraction, copy etc.)

8051 INTRUCTION SET

- 8051 instructions have 8-bit opcode
  - There are 256 possible instructions of which 255 are implemented
  - Some instructions have one or two additional bytes for data or address
  - There are 139 1-byte instructions, 92 2-byte instructions, and 24 3-byte instruction
  - Where does the data for an instruction come from? – Addressing modes

8051 INTRUCTION SET

- Depending on operation they perform, all instructions are divided in several groups:
  - Arithmetic Instructions
  - Branch Instructions
  - Data Transfer Instructions
  - Logic Instructions
  - Bit-oriented Instructions

http://www.mikroe.com/chapters/view/66/chapter-3-the-8051-instruction-set/

8051 INTRUCTION SET

DA A - Decimal adjust accumulator

Description: Instruction adjusts the contents of the accumulator to correspond to a BCD number after two BCD numbers have been added by the ADD and ADDC instructions. The result in form of two 4-digit BCD numbers is stored in the accumulator.

Syntax: DA A;

Byte: 1 (instruction code);

STATUS register flags: C;

8051 INTRUCTION SET

DA A - Decimal adjust accumulator

EXAMPLE:

```
0322    ...
0323    ADDC     A,B
0324    DA       A
```

Before execution: A=47h
B=25h
After execution: A=6Ch
After BCD conversion: A=72h (00100011), C=0(No Overflow)

ADDRESSING MODES

ADDRESSING MODES

- Addressing mode is a way to address an operand. Operand means the data we are operating upon (in most cases source data).

- It can be a direct address of memory, it can be register names, it can be any numerical data etc.

ADDRESSING MODES

- There are eight addressing modes available in the 8051:
  - Register
  - Direct
  - Indirect
  - Immediate
  - Relative
  - Absolute
  - Long
  - Indexed

ADDRESSING MODES

IMMEDIATE ADDRESSING MODE

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

- In general we can write MOV A, #data.

MOV A, #6AH

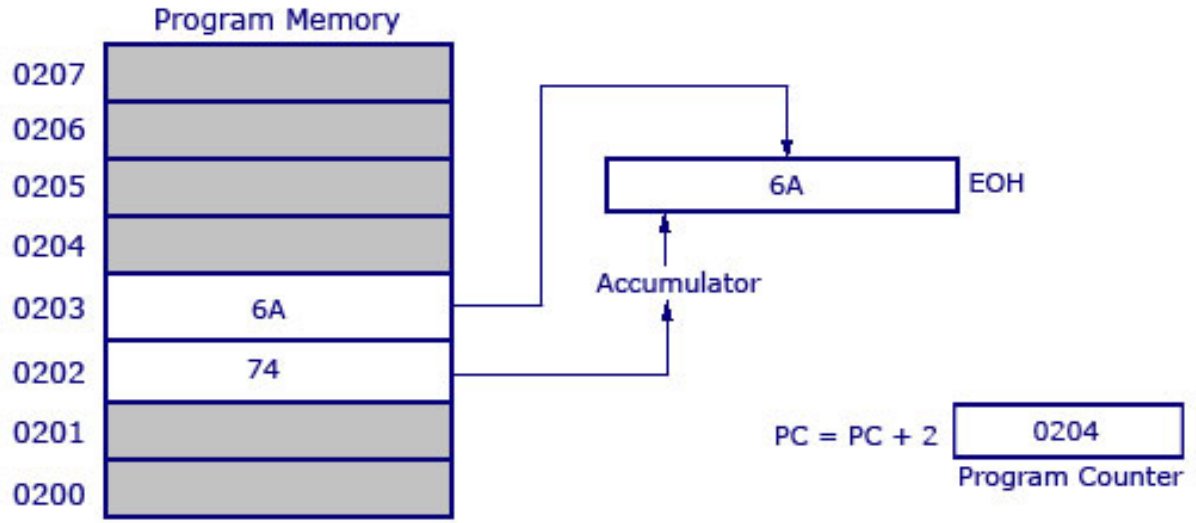
- The '#' symbol before 6AH indicates that operand is a data (8 bit). If '#' is not present then the hexadecimal number would be taken as address.

ADDRESSING MODES

IMMEDIATE ADDRESSING MODE

Immediate Addressing Mode

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



ADDRESSING MODES

IMMEDIATE ADDRESSING MODE

- 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.

ADDRESSING MODES

DIRECT ADDRESSING MODE

- Here the address of the data (source data) is given as operand. Lets take an example.

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.

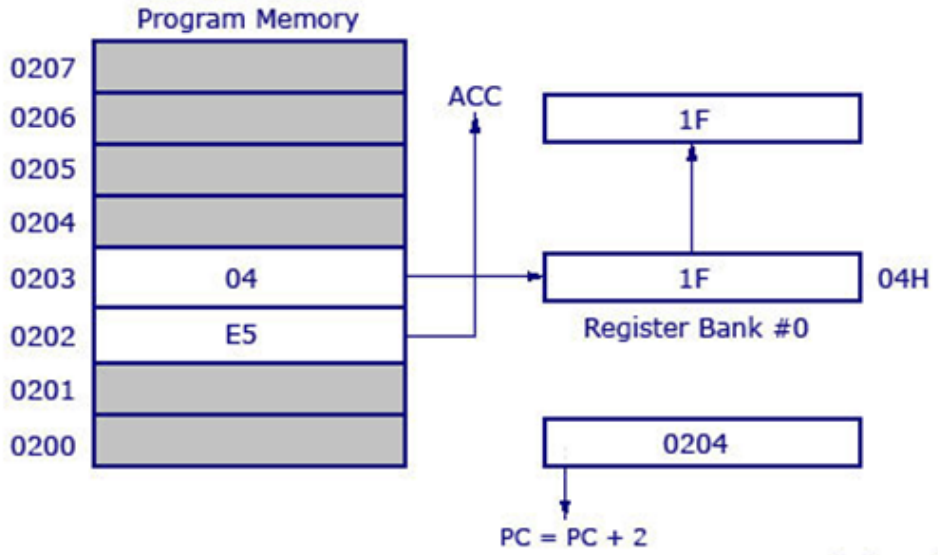
- In the figure register 04H holds the data 1FH. So the data 1FH is moved to accumulator.

ADDRESSING MODES

DIRECT ADDRESSING MODE

Direct Addressing Mode

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



ADDRESSING MODES

REGISTER ADDRESSING MODE

- In this addressing mode we use the register name directly (as source operand). At a time registers can take value from R0,R1…to R7.An example is shown below.

MOV A, R4

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

- PSW.3 and PSW.4bits are known as register bank select bits as they are used to select register banks.

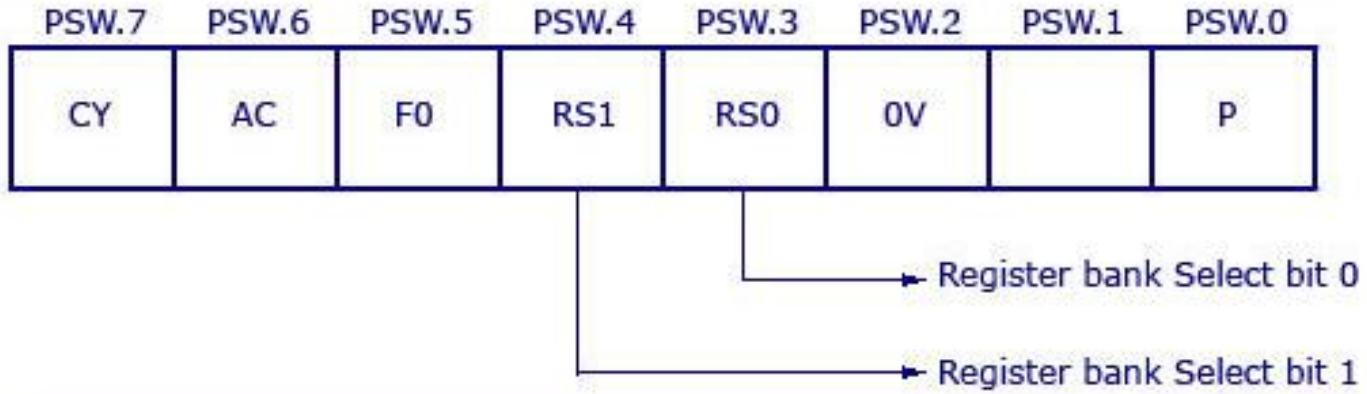
ADDRESSING MODES

REGISTER ADDRESSING MODE

- 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.

ADDRESSING MODES

REGISTER ADDRESSING MODE



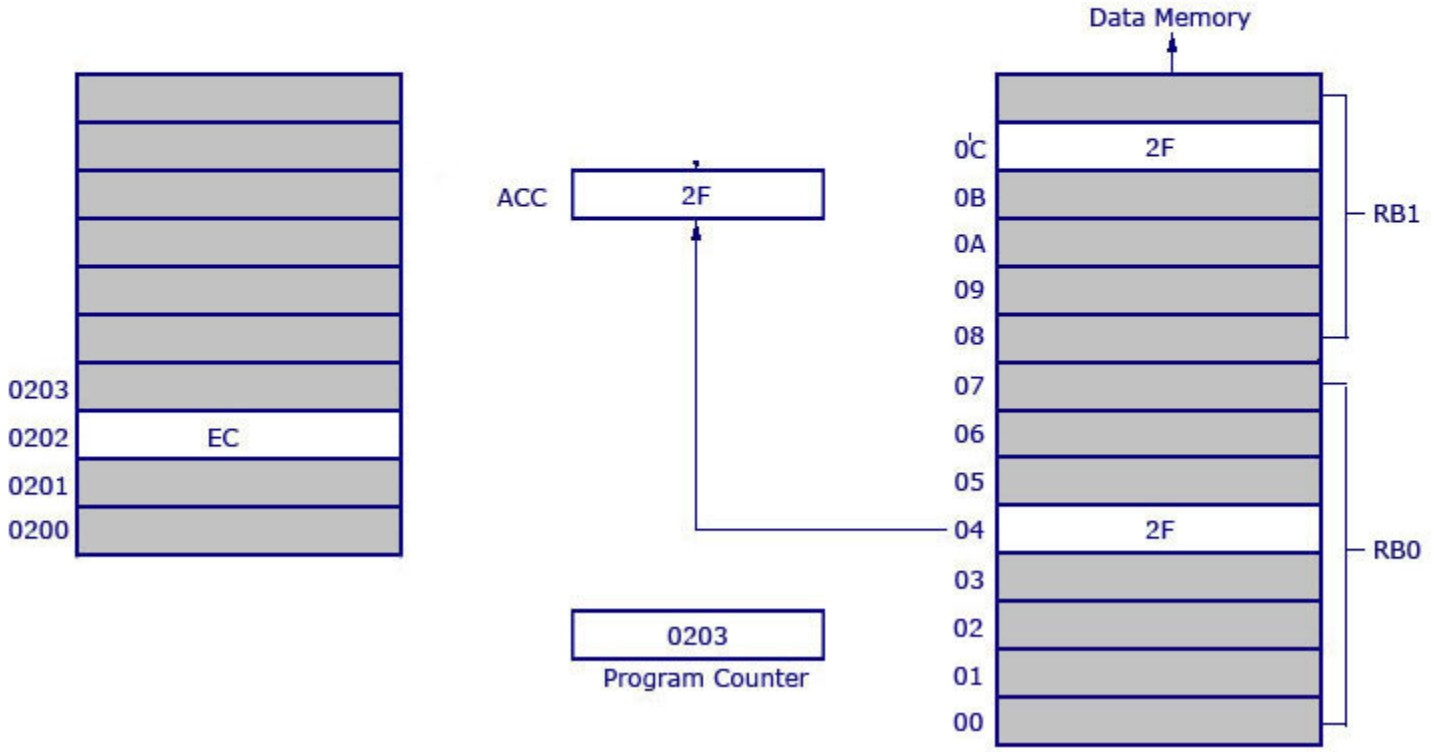
RS1	RS0	Register Bank
0	0	0
0	1	1
1	0	2
1	1	3

Register Bank Status
Register Bank 0 is selected
Register Bank 1 is selected
Register Bank 2 is selected
Register Bank 3 is selected

ADDRESSING MODES

REGISTER ADDRESSING MODE

Instruction	Opcode	Bytes	Cycles
MOV A, R4	ECH	1	1



ADDRESSING MODES

REGISTER INDIRECT ADDRESSING MODE

- In this addressing mode, address of the data (source data to transfer) is given in the register operand.

  MOV A, @R0

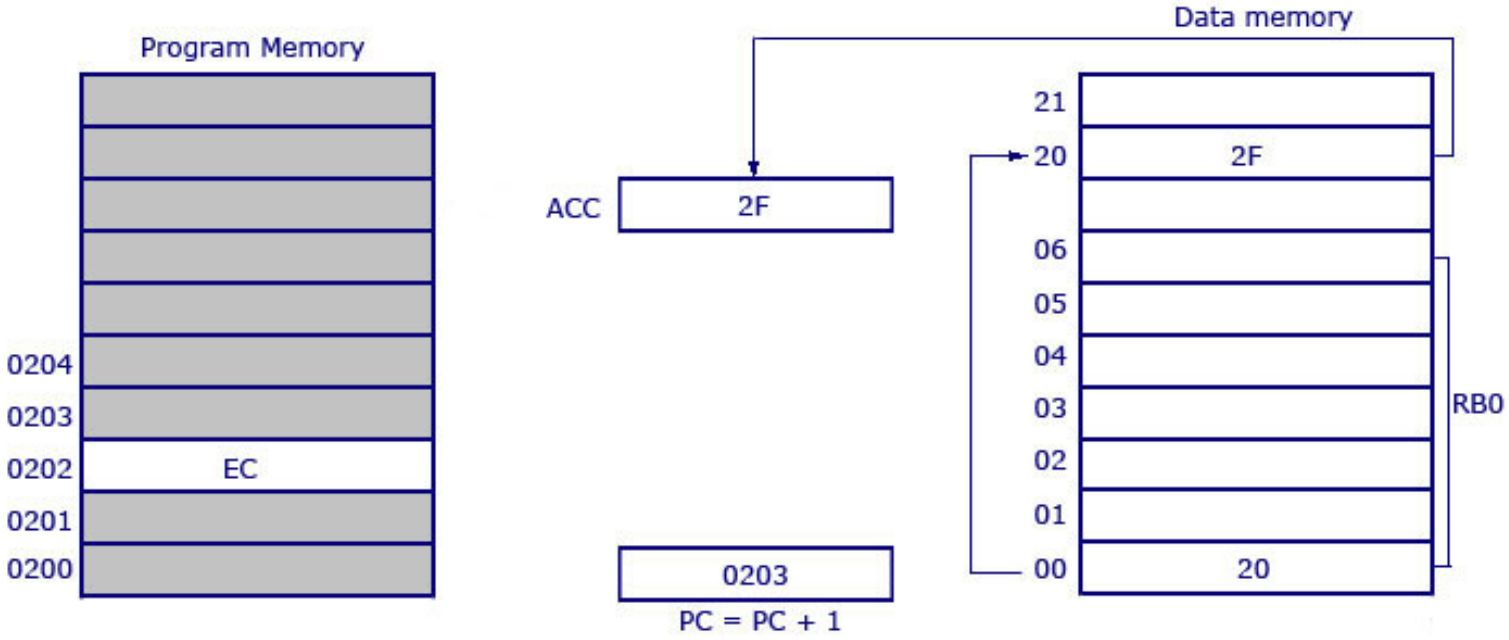
- Here the value inside R0 is considered as an address, which holds the data to be transferred to accumulator

- 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.

ADDRESSING MODES

REGISTER INDIRECT ADDRESSING MODE

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



ADDRESSING MODES

REGISTER INDIRECT ADDRESSING MODE

- 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.
- Only R0 and R1 are allowed to form a register indirect addressing instruction. All register banks are allowed.

ADDRESSING MODES

INDEXED ADDRESSING MODE

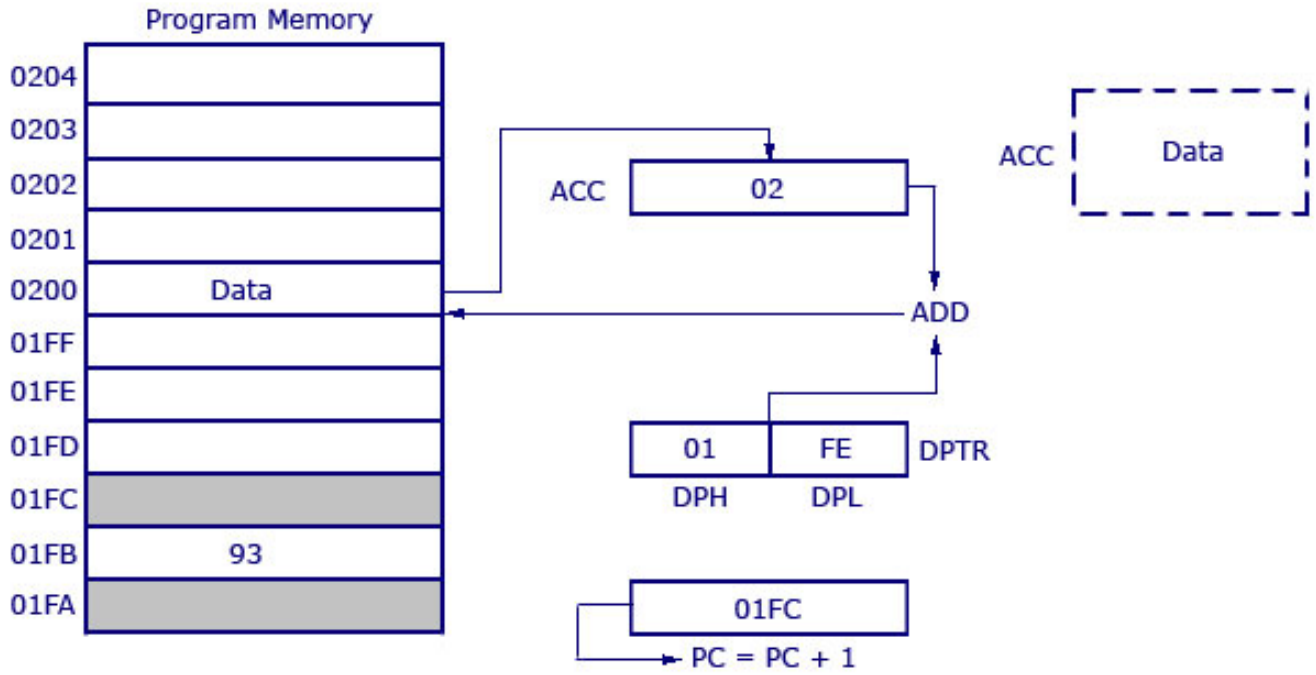
MOVC A, @A+DPTR and MOVC A, @A+PC

- where DPTR is data pointer and PC is program counter (both are 16 bit registers). Lets take the first example.
- The source operand is @A+DPTR and we know we will get the source data (to transfer) from this location.
- It is nothing but adding contents of DPTR with present content of accumulator. This addition will result a new data which is taken as the address of source data (to transfer). The data at this address is then transferred to accumulator.

ADDRESSING MODES

INDEXED ADDRESSING MODE

Instruction	Opcode	Bytes	Cycles
MOVC A,@A +DPTR	93H	1	2



ADDRESSING MODES

INDEXED ADDRESSING MODE

- The opcode for the instruction is 93H. DPTR holds the value 01FE, where 01 is located in DPH (higher 8 bits) and FE is located in DPL (lower 8 bits).
- Accumulator now has the value 02H. A 16 bit addition is performed and now 01FE H+02 H results in 0200 H.
- What ever data is in 0200 H will get transferred to accumulator. The previous value inside accumulator (02H) will get replaced with new data from 0200H. *New data in the accumulator is shown in dotted line box.*

ADDRESSING MODES

INDEXED ADDRESSING MODE

- This is a 1 byte instruction with 2 cycles needed for execution. So, the execution time required for this instruction is high compared to previous instructions (which all were 1 cycle).

- The other example MOVC A, @A+PC works the same way as above example. The only difference is, instead of adding DPTR with accumulator, here data inside program counter (PC) is added with accumulator to obtain the target address.

ADDRESSING MODES

RELATIVE ADDRESSING MODE

- This addressing mode is used only with certain jump instructions.
- A relative address (or offset) is an *8-bit signed value*, which is added to the program counter to form the address of the next instruction executed.
- The range for such a jump instruction is ±128 to+127 locations. Although the range is rather limited, relative addressing does offers the advantage of providing position-independent code (since absolute addresses are not used).

ADDRESSING MODES

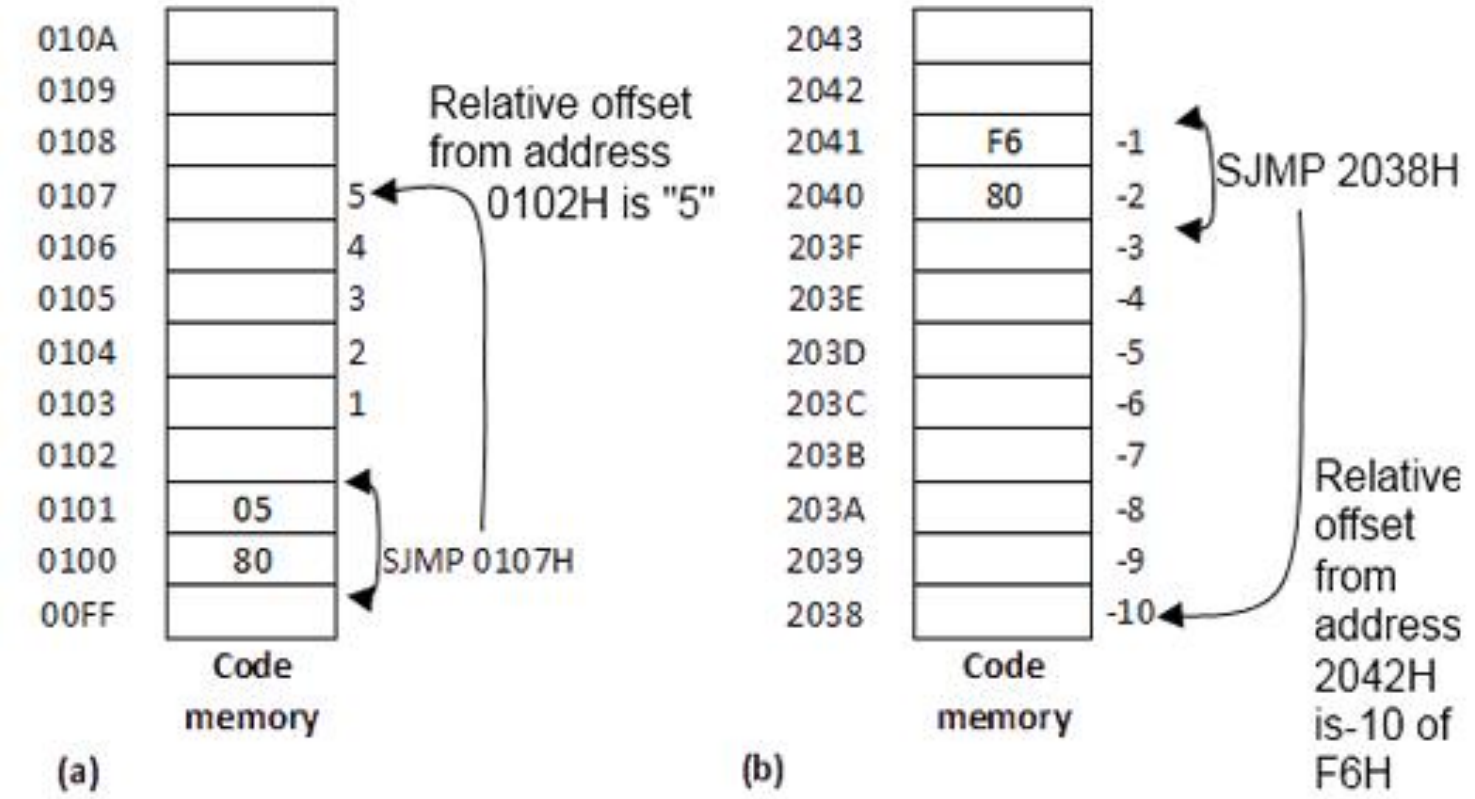
RELATIVE ADDRESSING MODE

- Prior to addition the PC is incremented to the address following jump instruction, thus, the new address is relative to the next instruction, not the address of the jump instruction.
- Since the destination are usually specified as labels and the assembler determines the relative offset accordingly.
- For example, if the label THERE represents an instruction at location 1040H, and the instruction **SJMP THERE** is in memory locations 1000h and 1001H, the assembler will assign a offset of 3E H as byte 2 of the instruction (1002 H + 3E H = 1040 H)

SJMP	LJMP	AJMP
Short jump, relative address is 8 bit it support 127 location forward	Long jump range is 64 kb	Absolute jump to anywhere within 2k block of program memory
It uses 8 bit address.	It uses 16 bit address	It uses an 11 bit address
2 byte instruction.	3 byte instruction	2 byte instruction

ADDRESSING MODES

RELATIVE ADDRESSING MODE



ADDRESSING MODES

ABSOLUTE ADDRESSING MODE

- This addressing mode used only with ACALL and AJUMP instructions.
- This two byte instruction allow within the current 2K page of the code memory by providing the 11 least significant bits of the destination address in the opcode (A10-A8) and byte-2 of the instruction (A7-A0).
- The upper five bits of the destination address are the current upper five bits in the program counter, so the instruction following the branch instruction and the destination for the branch instruction must be with in the same 2K page, since A15-A11 do not change.

ADDRESSING MODES

ABSOLUTE ADDRESSING MODE

- For example the label THERE represents an instruction at address 0F46H and the instruction

  AJMP THERE

- Is the memory location 0900 H and 0901 H, the assembler will encode the instruction as

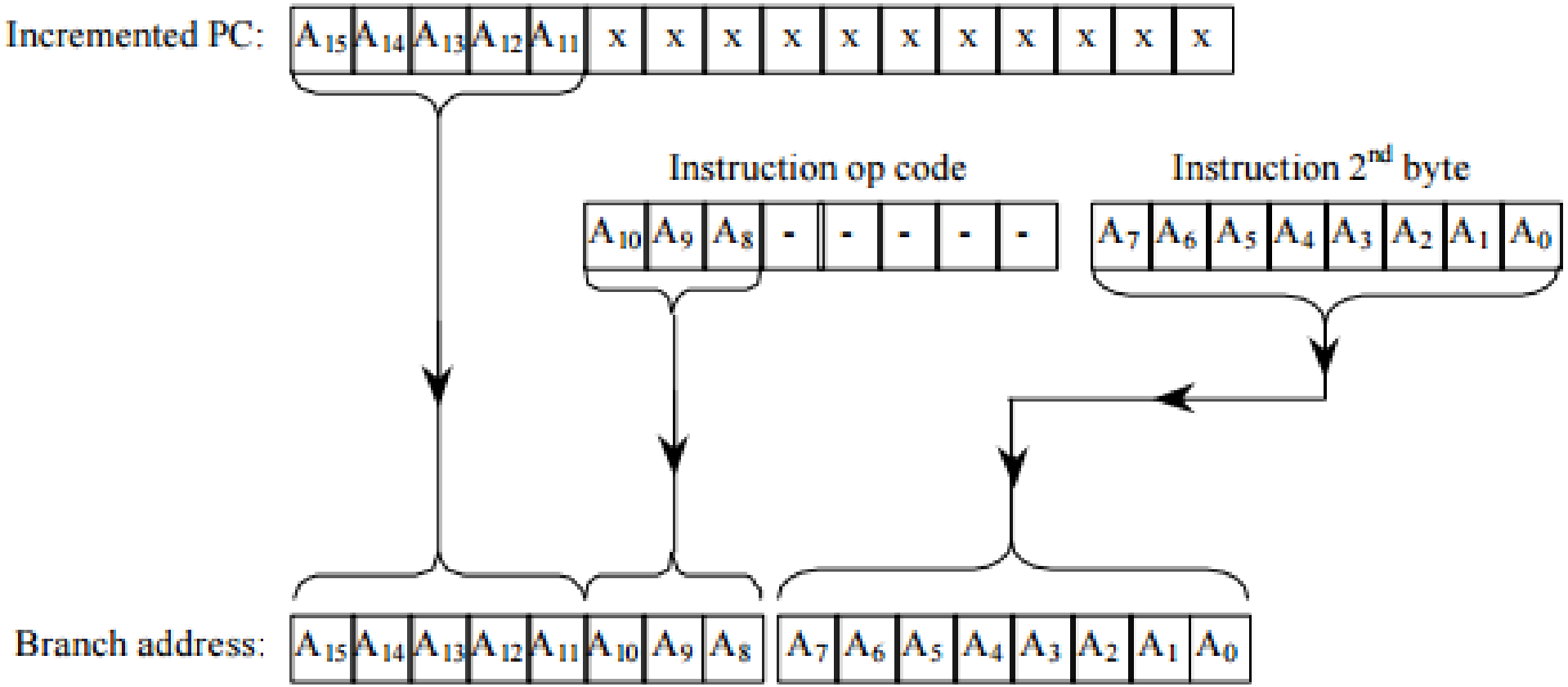
  11100001 – 1st byte (A10 – A8 of opcode)

  01000110 – 2nd byte (A7 – A0)

- The underline bit are the lower order 11 bits of the destination address 0F46 H = 00001<u>11101000110 B</u>

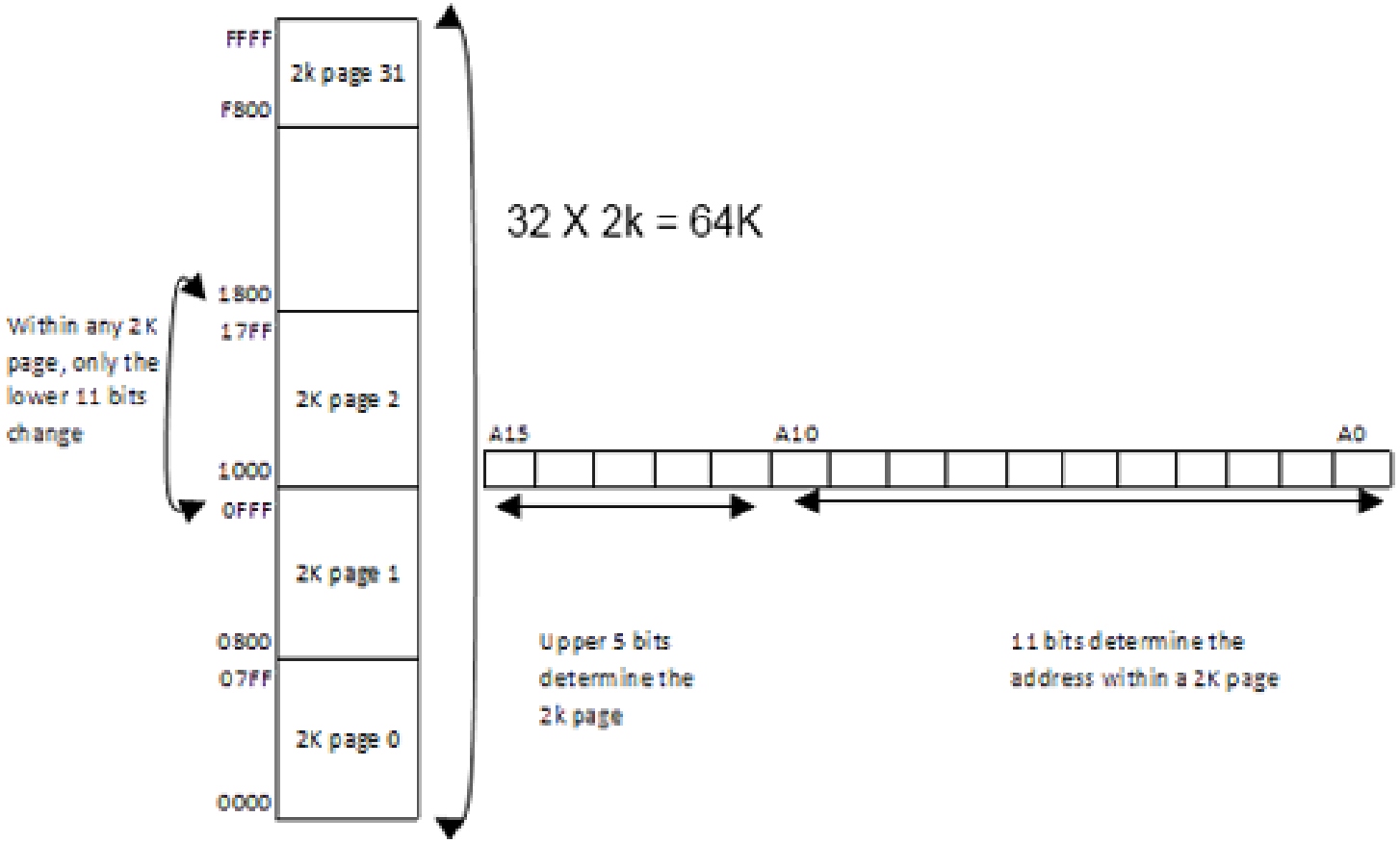
ADDRESSING MODES

ABSOLUTE ADDRESSING MODE



ADDRESSING MODES

ABSOLUTE ADDRESSING MODE



ADDRESSING MODES

LONG ADDRESSING MODE

- Long addressing mode is used only with LCALL and LJMP instruction , the advantage includes a full 16-bit destination address as bytes 2 or 3 of the instruction
- The position-dependence is the disadvantage because the program cannot execute at different address
- For example, a program begins at 2000H and an instruction such as LJMP 2040 H appears, then the program cannot be moved to 4000 H.
- The LJMP instruction would still jump to 2040 H which is not the correct location after the program has been moved.

ADDRESSING MODES

Addressing modes of MOV instructions

Instruction	Addressing mode	Function
MOV A, #data	Immediate	Load accumulator immediate
MOV Rn, #data	Immediate	Load Rn (R0–R7) immediate
MOV direct, #data	Immediate	Load direct address immediate
MOV @Ri, #data	Immediate	Load the location whose address is in Ri (R0 or R1) immediate
MOV A, Rn	Register direct	Copy to accumulator from Rn (R0–R7)
MOV A, direct	Direct	Copy to accumulator from direct address
MOV A, @Ri	Register indirect	Copy to accumulator from address in Ri (R0 or R1)
MOV Rn, A	Register direct	Copy to Rn (R0–R7) from accumulator
MOV Rn, direct	Direct	Copy to Rn (R0–R7) from direct address
MOV direct, A	Register direct	Copy to direct address from accumulator
MOV direct, Rn	Register direct	Copy to direct address from Rn (R0–R7)
MOV direct, direct	Direct	Copy to direct address from direct address
MOV direct, @Ri	Register indirect	Copy to direct address from address in Ri (R0 or R1)
MOV @Ri, A	Register direct	Copy to address in Ri (R0 or R1) from accumulator
MOV @Ri, direct	Direct	Copy to address in Ri (R0 or R1) from direct address
MOV DPTR, #data16	Immediate	Load DPTR (16 bit) immediate

ADDRESSING MODES

Addressing modes of ADD instruction

Instruction	Addressing mode	Function
ADD A, #data	Immediate	Add immediate with accumulator
ADD A, Rn	Register direct	Add content of specified register with accumulator
ADD A, direct	Direct	Add content of indicated address with accumulator
ADD A, @Ri	Register indirect	Add content of address indicated by the register with accumulator

Addressing modes of ADDC instruction

Instruction	Addressing mode	Function
ADDC A, #data	Immediate	Add immediate with accumulator with carry
ADDC A, Rn	Register direct	Add content of specified register with accumulator with carry
ADDC A, direct	Direct	Add content of indicated address with accumulator with carry
ADDC A, @Ri	Register indirect	Add content of address indicated by the register with accumulator with carry

Addressing modes of SUBB instruction

Instruction	Addressing mode	Function
SUBB A, #data	Immediate	Subtract immediately from accumulator with borrow
SUBB A, Rn	Register direct	Subtract indicated register content from accumulator with borrow
SUBB A, direct	Direct	Subtract content of indicated address from accumulator with borrow
SUBB A, @Ri	Register indirect	Subtract content of address indicated by the register from accumulator with borrow

ADDRESSING MODES

Addressing modes of INC instruction

Instruction	Addressing mode	Function
INC A	Register direct	Increment accumulator by one
INC Rn	Register direct	Increment content of indicated register by one
INC direct	Direct	Increment content of indicated address by one
INC @Ri	Register indirect	Increment content of address indicated by the register by one
INC DPTR	Register direct	Increment content of DPTR by one

Addressing modes of DEC instruction

Instruction	Addressing mode	Function
DEC A	Register direct	Decrement content of accumulator by one
DEC Rn	Register direct	Decrement content of indicated register by one
DEC direct	Direct	Decrement content of indicated address by one
DEC @Ri	Register indirect	Decrement content of address indicated by the register by one

ADDRESSING MODES

Addressing modes of CJNE instruction

Instruction	Addressing mode	Function
CJNE A, direct, rel	Direct	Compare A with direct address and jump if not equal
CJNE A, #data, rel	Immediate	Compare A with immediate data and jump if not equal
CJNE Rn, #data, rel	Immediate	Compare register with immediate data and jump if not equal
CJNE @Ri, #data, rel	Immediate	Compare content of address indicated by the register with immediate data and jump if not equal

Addressing modes of DJNZ instruction

Instruction	Addressing mode	Function
DJNZ Rn, rel	Register direct	Decrement register by one and jump if not zero
DJNZ direct, rel	Direct	Decrement content of direct address by one and jump if not zero

ARITHMETIC FLAGS

ARITHMETIC FLAGS

- Flag: It is a 1-bit register that indicates the status of the result from an operation
- Flags are either at a flag-state of value 0 or 1
- Arithmetic flags indicate the status of the results from mathematical operations (+, −, *, /)
- There are 4 arithmetic flags in the 8051:Carry (C), Auxiliary Carry (AC), Overflow (OV), Parity (P)
- All the above flags are stored in the Program Status Word(PSW)

ARITHMETIC FLAGS

INSTRUCTIONS THAT AFFECTING FLAGS

Instruction Mnemonic	Flags Affected		
ADD	C	AC	OV
ADDC	C	AC	OV
SUBB	C	AC	OV
MUL	C = 0		OV
DIV	C = 0		OV
DA A	C		
SETB C	C = 1		
MOV C, bit	C		

ARITHMETIC FLAGS

INSTRUCTIONS THAT AFFECTING FLAGS

Instruction Mnemonic	Flags Affected		
ORL C, bit	C		
ANL C, bit	C		
RLC	C		
RRC	C		
CLR C	C = 0		
CPL C	C = /C		
CJNE	C		

ARITHMETIC FLAGS

EXAMPLE

Show how the flag register is affected by the following instructions.

```
MOV  A, #0F5h        ; A = F5h
ADD  A, #0Bh         ; A = F5 + 0B = 00
```

Solution	F5h	1111 0101
	+ 0Bh	+ 0000 1011
	100h	0000 0000

After the addition, register A (destination) contains 00 and the flags are:

CY = 1 since there is a carry out from D7

P = 0 because the number of 1s is zero

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

BOOLEAN PROCESSING INSTRUCTIONS

Mnemonic		Description	Byte	Cyc
SETB	C	Set Carry flag	1	1
SETB	bit	Set direct Bit	2	1
CLR	C	Clear Carry flag	1	1
CLR	bit	Clear direct bit	2	1
CPL	C	Complement Carry flag	1	1
CPL	bit	Complement direct bit	2	1
MOV	C.bit	Move direct bit to Carry flag	2	1
MOV	bit.C	Move Carry flag to direct bit	2	2
ANL	C.bit	AND direct bit to Carry flag	2	2
ANL	C.bit	AND complement of direct bit to Carry flag	2	2
ORL	C.bit	OR direct bit to Carry flag	2	2
ORL	C.bit	OR complement of direct bit to Carry flag	2	2
JC	rel	Jump if Carry is flag is set	2	2
JNC	rel	Jump if No Carry flag	2	2
JB	bit.rel	Jump if direct Bit set	3	2
JNB	bit.rel	Jump if direct Bit Not set	3	2
JBC	bit.rel	Jump if direct Bit is set & Clear bit	3	2

Data transfer		Byte	Cycle
MOV A,Rn	Move register to accumulator	1	1
MOV A,direct")	Move direct byte to accumulator	2	1
MOV A,@Ri	Move indirect RAM to accumulator	1	1
MOV A,#data	Move immediate data to accumulator	2	1
MOV Rn,A	Move accumulator to register	1	1
MOV Rn,direct	Move direct byte to register	2	2
MOV Rn,#data	Move immediate data to register	2	1
MOV direct,A	Move accumulator to direct byte	2	1
MOV direct,Rn	Move register to direct byte	2	2
MOV direct,direct	Move direct byte to direct byte	3	2
MOV direct,@Ri	Move indirect RAM to direct byte	2	2
MOV direct,#data	Move immediate data to direct byte	3	2
MOV @Ri,A	Move accumulator to indirect RAM	1	1
MOV @Ri,direct	Move direct byte to indirect RAM	2	2
MOV @Ri,#data	Move immediate data to indirect RAM	2	1
MOV DPTR,#data16	Load data pointer with a 16-bit constant	3	2
MOVC A,@A+DPTR	Move code byte relative to DPTR to accumulator	1	2
MOVC A,@A+PC	Move code byte relative to PC to accumulator	1	2
MOVX A,@Ri	Move external RAM (8-bit addr.) to A	1	2
MOVX A,@DPTR	Move external RAM (16-bit addr.) to A	1	2
MOVX @Ri,A	Move A to external RAM (8-bit addr.)	1	2
MOVX @DPTR,A	Move A to external RAM (16-bit addr.)	1	2

Mnemonic		Description	Byte	Cycle
Arithmetic operations				
ADD	A,Rn	Add register to accumulator	1	1
ADD	A,direct	Add direct byte to accumulator	2	1
ADD	A,@Ri	Add indirect RAM to accumulator	1	1
ADD	A,#data	Add immediate data to accumulator	2	1
ADDC	A,Rn	Add register to accumulator with carry flag	1	1
ADDC	A,direct	Add direct byte to A with carry flag	2	1
ADDC	A,@Ri	Add indirect RAM to A with carry flag	1	1
ADDC	A,#data	Add immediate data to A with carry flag	2	1
SUBB	A,Rn	Subtract register to accumulator with borrow	1	1
SUBB	A,direct	Subtract direct byte to A with carry borrow	2	1
SUBB	A,@Ri	Subtract indirect RAM to A with carry borrow	1	1
SUBB	A,#data	Subtract immediate data to A with carry borrow	2	1
INC	A	Increment accumulator	1	1
INC	Rn	Increment register	1	1
INC	direct	Increment direct byte	2	1
INC	@Ri	Increment indirect RAM	1	1
DEC	A	Decrement accumulator	1	1
DEC	Rn	Decrement register	1	1
DEC	direct	Decrement direct byte	2	1
DEC	@Ri	Decrement indirect RAM	1	1
INC	DPTR	Increment data pointer	1	2
MUL	AB	Multiply A and B -> [B hi]:[A lo]	1	4
DIV	AB	Divide A by B -> A=result, B=remainder	1	4
DA	A	Decimal adjust accumulator	1	1
CLR	A	Clear accumulator	1	1

Boolean variable manipulation			Byte	Cycle
CLR	C	Clear carry flag	1	1
CLR	bit	Clear direct bit	2	1
SETB	C	Set carry flag	1	1
SETB	bit	Set direct bit	2	1
CPL	C	Complement carry flag	1	1
CPL	bit	Complement direct bit	2	1
ANL	C,bit	AND direct bit to carry flag	2	2
ANL	C,/bit	AND complement of direct bit to carry	2	2
ORL	C,bit	OR direct bit to carry flag	2	2
ORL	C,/bit	OR complement of direct bit to carry	2	2
MOV	C,bit	Move direct bit to carry flag	2	1
MOV	bit,C	Move carry flag to direct bit	2	2

Program and machine control			Byte	Cycle
ACALL	addr11	Absolute subroutine call	2	2
LCALL	addr16	Long subroutine call	3	2
RET		Return from subroutine	1	2
RETI		Return from interrupt	1	2
AJMP	addr11	Absolute jump	2	2
LJMP	addr16	Long jump	3	2
SJMP	rel	Short jump (relative address)	2	2
JMP	@A+DPTR	Jump indirect relative to the DPTR	1	2
JZ	rel	Jump if accumulator is zero	2	2
JNZ	rel	Jump if accumulator is not zero	2	2
JC	rel	Jump if carry flag is set	2	2
JNC	rel	Jump if carry flag is not set	2	2
JB	bit,rel	Jump if bit is set	3	2
JNB	bit,rel	Jump if bit is not set	3	2
JBC	bit,rel	Jump if direct bit is set and clear bit	3	2
CJNE	A,direct,rel	Compare direct byte to A and jump if not equal	3	2
CJNE	A,#data,rel	Compare immediate to A and jump if not equal	3	2
CJNE	Rn,#data,rel	Compare immed. to reg. and jump if not equal	3	2
CJNE	@Rn,#data,rel	Compare immed. to ind. and jump if not equal	3	2
DJNZ	Rn,rel	Decrement register and jump in not zero	2	2
DJNZ	direct,rel	Decrement direct byte and jump in not zero	3	2
NOP		No operation	1	1

Mnemonic		Description	Byte	Cycle
CPL	A	Complement accumulator	1	1
RL	A	Rotate accumulator left	1	1
RLC	A	Rotate accumulator left through carry	1	1
RR	A	Rotate accumulator right	1	1
RRC	A	Rotate accumulator right through carry	1	1
SWAP	A	Swap nibbles within the accumulator	1	1
Logic operations				
ANL	A,Rn	AND register to accumulator	1	1
ANL	A,direct	AND direct byte to accumulator	2	1
ANL	A,@Ri	AND indirect RAM to accumulator	1	1
ANL	A,#data	AND immediate data to accumulator	2	1
ANL	direct,A	AND accumulator to direct byte	2	1
ANL	direct,#data	AND immediate data to direct byte	3	2
ORL	A,Rn	OR register to accumulator	1	1
ORL	A,direct	OR direct byte to accumulator	2	1
ORL	A,@Ri	OR indirect RAM to accumulator	1	1
ORL	A,#data	OR immediate data to accumulator	2	1
ORL	direct,A	OR accumulator to direct byte	2	1
ORL	direct,#data	OR immediate data to direct byte	3	2
XRL	A,Rn	Exclusive OR register to accumulator	1	1
XRL	A,direct	Exclusive OR direct byte to accumulator	2	1
XRL	A,@Ri	Exclusive OR indirect RAM to accumulator	1	1
XRL	A,#data	Exclusive OR immediate data to accumulator	2	1
XRL	direct,A	Exclusive OR accumulator to direct byte	2	1
XRL	direct,#data	Exclusive OR immediate data to direct byte	3	2

Lets find the time period of the machine cycle in each case for the following crystal frequency of different 8051 based systems: 11.0592 MHz, 16 MHz, 20 MHz

11.0592 MHz:

11.0592/12 = 921.6 KHz

Machine cycle = 1/921.6 KHz = 1.085us [us=microsecond]

16 MHz:

16MHz/12 = 1.333 MHz

Machine cycle = 1/1.333 MHz = 0.75us [us=microsecond]

20MHz:

20MHz/12 = 1.66 MHz

Machine Cycle = 1/1.66 MHz = 0.60us [us=microsecond]

Lets find how long it takes to execute each of the following instructions, for a crystal frequency of 11.0592 MHz. The machine cycle of a system of 11.0592 MHz is 1.085 us.

INSTRUCTION	MACHINE CYCLE	TIME TO EXECUTE
MOV R2,#55H	1	1x1.085 us = 1.085 us
DEC R2	1	1x1.085 us = 1.085 us
DJNZ R2,target	2	2x1.085 us = 2.17 us
LJMP	2	2x1.085 us = 2.17 us
SJMP	2	2x1.085 us = 2.17 us
NOP	1	1x1.085 us = 1.085 us
MUL AB	4	4x1.085 us = 4.34 us

Lets find the size of the delay if the crystal frequency of 11.0592 MHz is connected.

```
MOV A,#55H
AGAIN: MOV P1,A
ACALL DELAY
CPL A
SJMP AGAIN
;----------------Time Delay
DELAY: MOV R3,#225
HERE: DJNZ R3,HERE
RET
```

We have the following machine cycles for each instruction of the DELAY subroutine.

DELAY: MOV R2,#255	Machine Cycle = 1
HERE: DJNZ R2,HERE	Machine Cycle = 2
RET	Machine Cycle = 1

Therefore, we have a time delay of [(255 x 2) + 1 + 1] x 1.085 us = 555.52 us

Very often we used to calculate the time delay based on the instructions inside the loop and ignore the clock cycles associated with the instructions outside the loop.

Lets find the time delay for the following subroutine with 11.0592 MHz crystal frequency is connected to the 8051 system.

DELAY: MOV R2,#255	Machine Cycle = 1
HERE: NOP	**Machine Cycle = 1**
NOP	**Machine Cycle = 1**
NOP	**Machine Cycle = 1**
NOP	**Machine Cycle = 1**
DJNZ R2,HERE	**Machine Cycle = 2**
RET	1

- The time delay inside the HERE loop is [255(1+1+1+1+2)] x 1.085 us = 1660.05 us
- The time delay of the two instructions outside the loop is: [1660.05 us+1+1)]x1.085 us = 1803.32425 us

For a crystal frequency of 11.0592 MHz, lets find the time delay in the following subroutine. The machine cycle is 1.085 us.

DELAY: MOV R2,#200	Machine Cycle = 1
AGAIN: MOV R3,#250	Machine Cycle = 1
HERE: NOP	Machine Cycle = 1
NOP	Machine Cycle = 1
DJNZ R3,HERE	Machine Cycle = 2
DJNZ R2,AGAIN	Machine Cycle = 2
RET	Machine Cycle = 1

DELAY: MOV R2,#200	Machine Cycle = 1
AGAIN: MOV R3,#250	Machine Cycle = 1
HERE: NOP	Machine Cycle = 1
NOP	Machine Cycle = 1
DJNZ R3,HERE	Machine Cycle = 2
DJNZ R2,AGAIN	Machine Cycle = 2
RET	Machine Cycle = 1

'HERE' Loop Calculations: 1+1+2, so [(1+1+2)x250] x 1.085 us = 1085 us.

'AGAIN' Loop Calculations: In this loop "MOV R3,#250" and "DJNZ R2,AGAIN" at the begining and end of the AGAIN loop add [(1+2)x200]x1.085 us = 651us to the time delay. The AGAIN loop repeats the HERE loop 200 times so 200x1085 us = 217000 us. As a result the total time delay will be 217000 us + 651 us = 217651 us or 217.651 milliseconds. The time is approximate as we have ignored the first and the last instructions in the subroutine i.e. DELAY: "MOV R2,#200" and "RET".

Program to delay 1mS.

For an 8051 microcontroller clocked by a 12MHz crystal,

```
DELAY:  MOV  R6,#250D
        MOV  R7,#250D
LABEL1: DJNZ R6,LABEL1
LABEL2: DJNZ R7,LABEL2
        RET
```

1 instruction cycle= 12 /12MHz = 1µS.

The above program roughly produces a delay of 1mS. The instruction DJNZ R*x*,*LABEL* is a two cycle instruction and it will take 2µS to execute. So repeating this instruction 500 times will generate a delay of 500 x 2µS = 1mS.

Program to delay 1 second.

```
DELAY1: MOV R5,#250D
LABEL:  ACALL DELAY
        ACALL DELAY
        ACALL DELAY
        ACALL DELAY
        DJNZ R5,LABEL
        RET
DELAY:  MOV R6,#250D
        MOV R7,#250D
LOOP1:  DJNZ R6,LOOP1
LOOP2:  DJNZ R7,LOOP1
        RET
```

The program shown below produces a delay of around 1 second. In this program subroutine for delaying 1mS (DELAY) is called 4 times back to back and the entire cycle is repeated 250 times. As result, a delay of 4 x 1mS x 250 = 1000mS = 1 second is produced.

8051 ALP program to find number of negative numbers in an array

```
ORG 000h
MOV DPTR,#0AH   ; an external address
MOV R0,#0AH     ;array counter
MOV R1,#00H     ;to count number neg numbers
MOV A,#00H
Loop: MOVC A,@A+DPTR
INR DPTR
RLC A
JNC Loop1
INC R1      ;Counter
DJNZ R0,loop
SJMP Exit
Loop1: DJNZ R0, Loop
Exit: NOP
END
```


THANK YOU