

# BEE403:MICROCONTROLLERS

## MODULE – 1:8051 Microcontrollers Basics

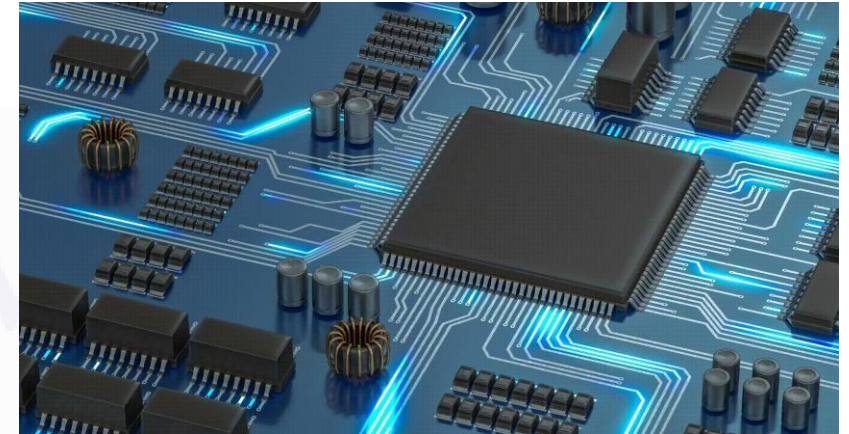


## Learning Resources prescribed by University:

- The 8051 Microcontroller and Embedded Systems Using Assembly and C , Muhammad Ali Mazadi ,Pearson 2<sup>nd</sup> Edition, 2008.
- The 8051 Microcontroller, Kenneth Ayala, Cengage Learning , 3rd Edition, 2005
- The 8051 Microcontroller and Embedded Systems ,Manish K Patel, McGraw Hill, 2014 Microcontrollers: Architecture, Programming, Interfacing and System Design, Raj Kamal , Pearson ,1st Edition, 2012

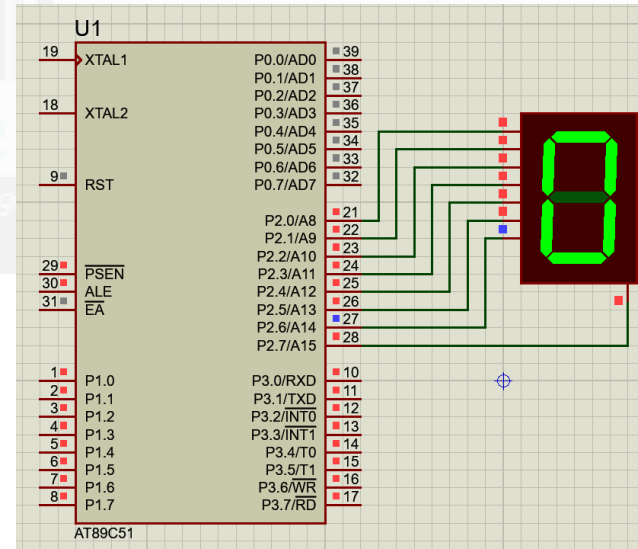
# OUTLINE

1. Inside the Computer
2. Microcontrollers and Embedded Processors
3. Block Diagram of 8051
4. PSW and Flag Bits
5. 8051 Register Banks and Stack
6. Internal Memory Organization of 8051
7. IO Port Usage in 8051
8. Types of Special Function Registers and their uses in 8051
9. Pins Of 8051
10. Memory Address Decoding, 8031/51 Interfacing With External ROM and RAM
11. 8051 Addressing Modes

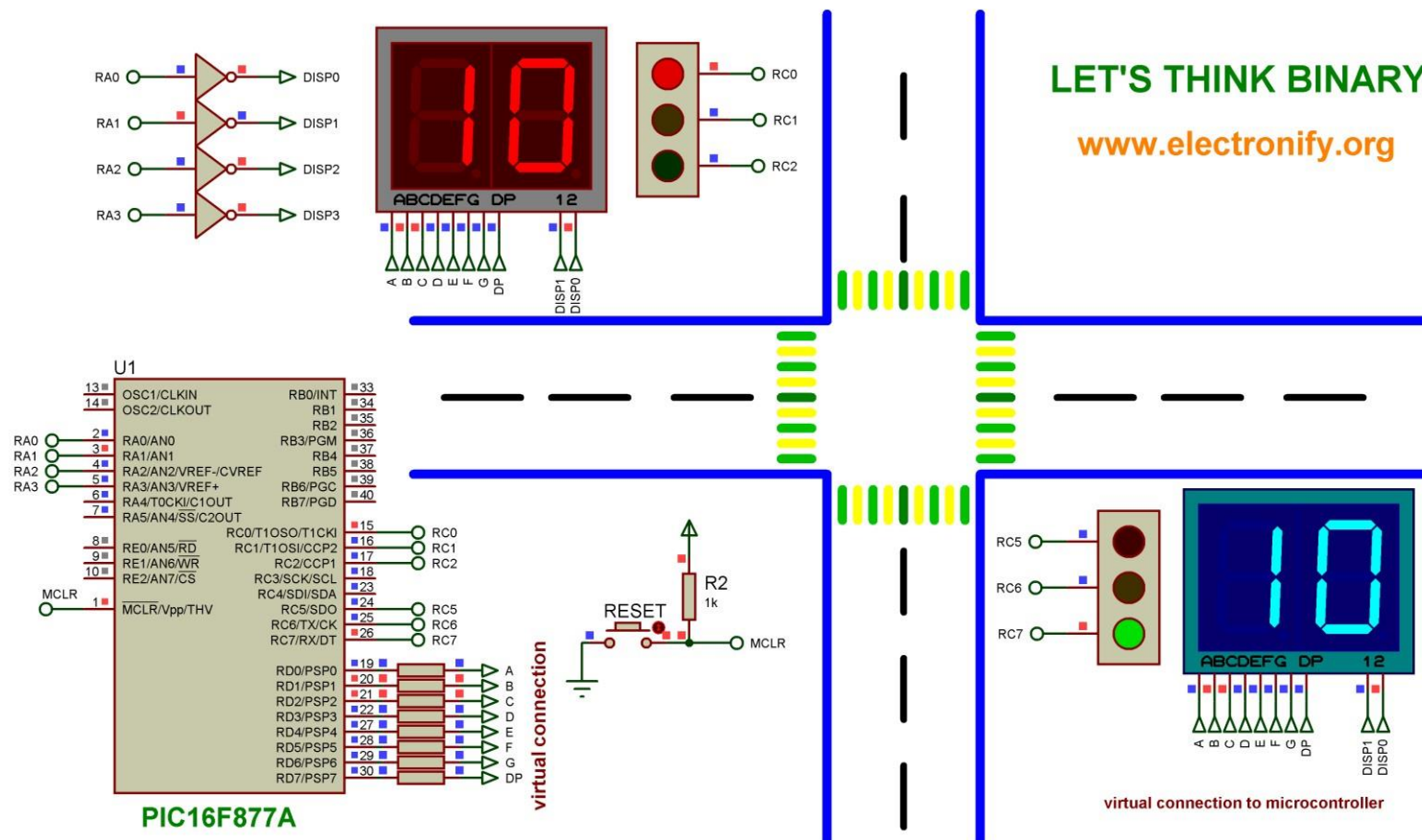




# 1.1 Inside the Computer



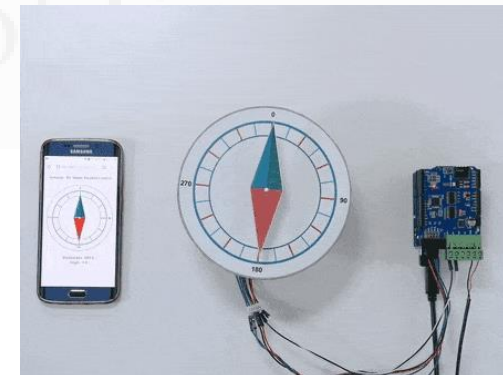
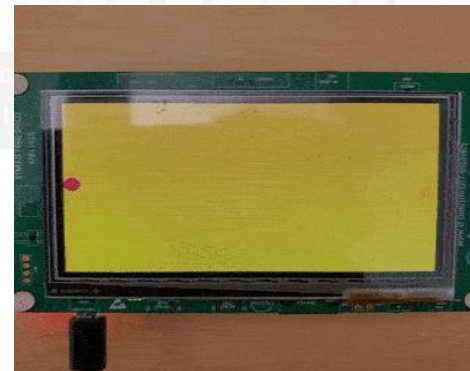
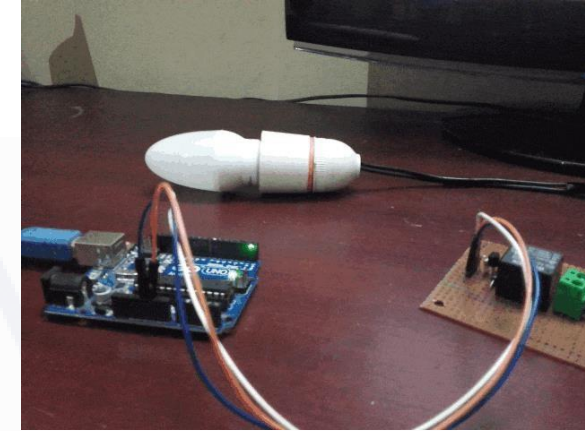
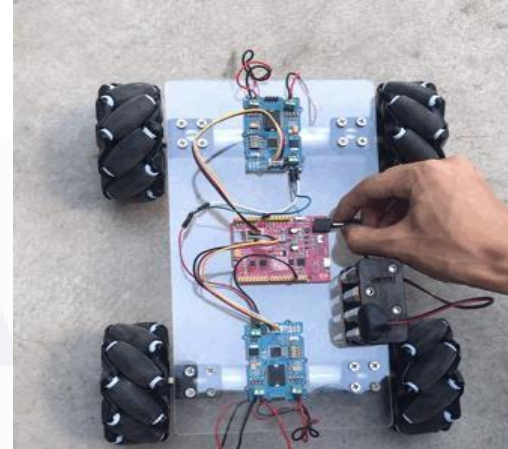
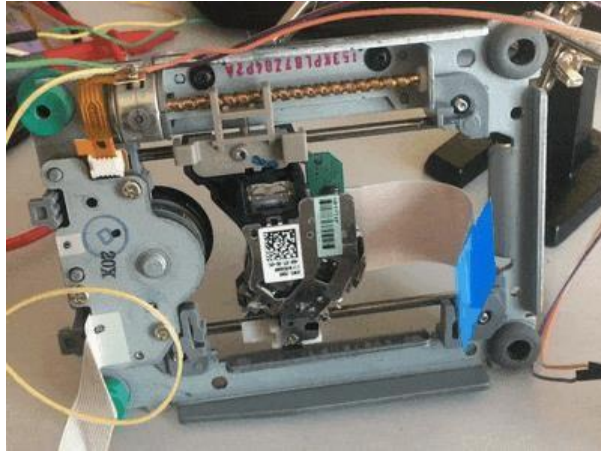
## AUTOMATIC TRAFFIC LIGHTS







A T M E  
College of Engineering



## 1.1 Inside the Computer

A computer is a multipurpose programmable machine that reads binary instructions from its memory, accepts binary data as input, processes the data according to those instructions and provides results as output.

Microprocessor or 'Computer on Chip' first became a commercial reality in 1971 with the introduction of the 4 bit 4004 by Intel.



# Why do we need to learn **Microprocessors/controllers?**

- The microprocessor is the core of **computer systems**.
- Many **communication, digital-entertainment, portable devices**, are controlled by them.
- A designer should know what types of components he needs, ways to reduce **production costs** and product reliable.



# Three criteria in Choosing a Microcontroller

## 1.Meeting the computing needs of the task efficiently and cost effectively

- Speed, the amount of ROM and RAM, the number of I/O ports and timers, size, packaging, power consumption
- Easy to upgrade
- Cost per unit

## 2.Availability of software development tools : Assemblers, debuggers, C compilers, emulator, simulator, technical support

## 3. Wide availability and reliable sources of the microcontrollers.

# Different aspects of a microprocessor/controller

- **Hardware** : Interface to the real world
- **Software** : order how to deal with inputs
- **A set of instructions** written for the computer to solve a specific task is called **program** and collection of programs is called **software**

## 1.1.1 Computer Software

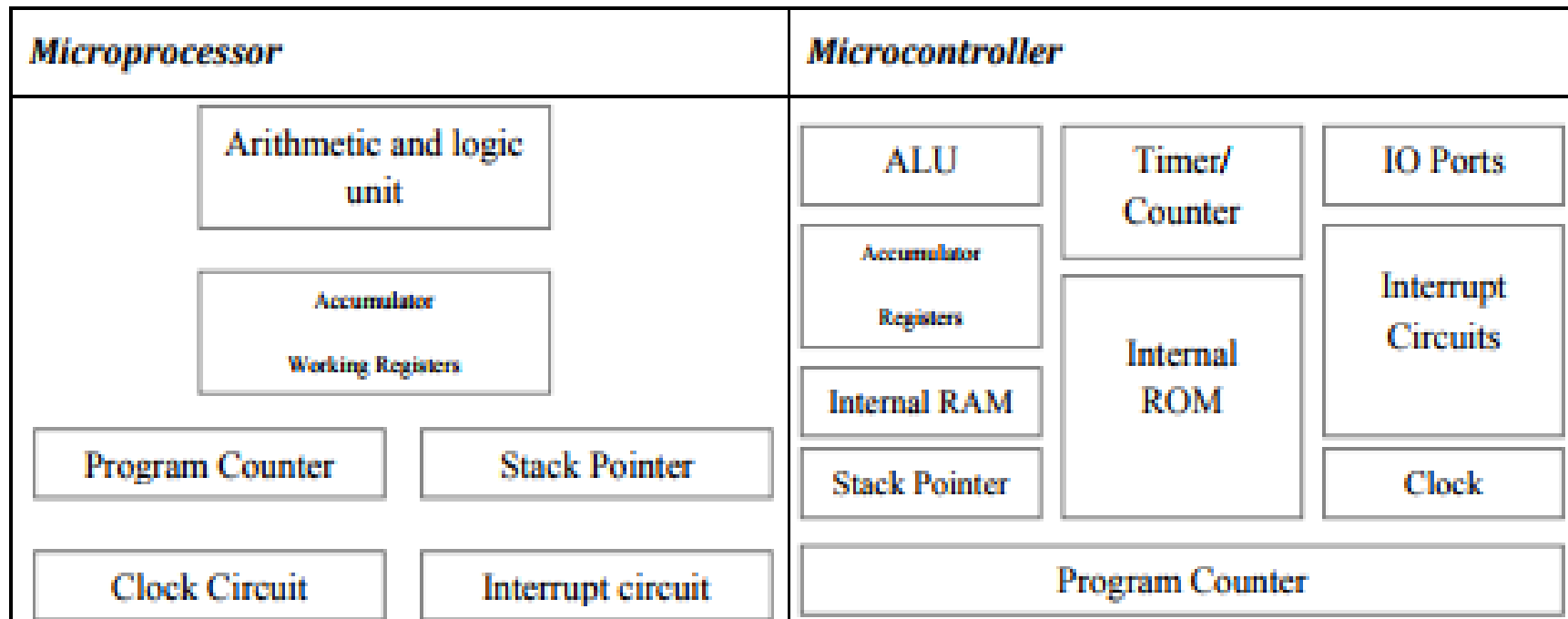
- The program stored in the computer memory in the form of **binary numbers** is called **machine instructions**. The machine language program is called **object code**.
- The assembly language program the programmer enters is called **source code**. The source code (**assembly language**) is translated to **object code** (**machine language**) using assembler

## 1.1.1 Computer Software Continued

- High level languages such as **C, C++** etc. High level language will be converted to machine language using **compiler or interpreter**
- **Compiler reads the entire program** and translate into the object code and then it is executed by the processor.
- **Interpreter takes one statement** of the high level language as input and translate it into object code and then executes.

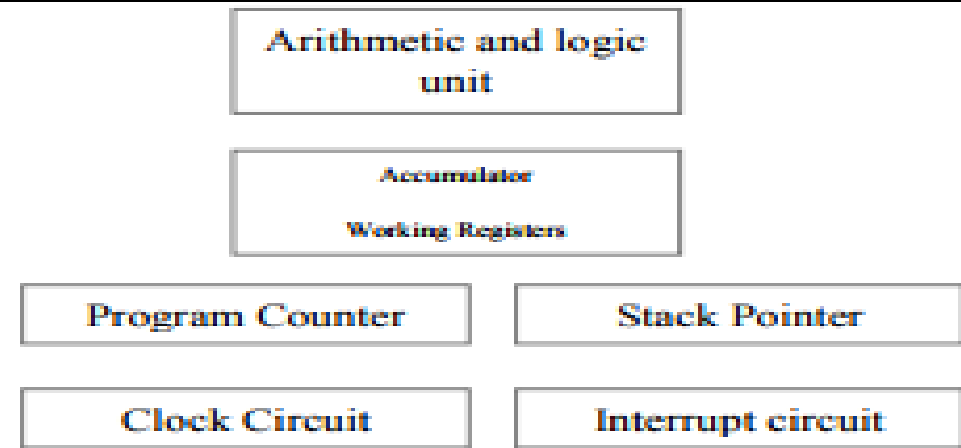
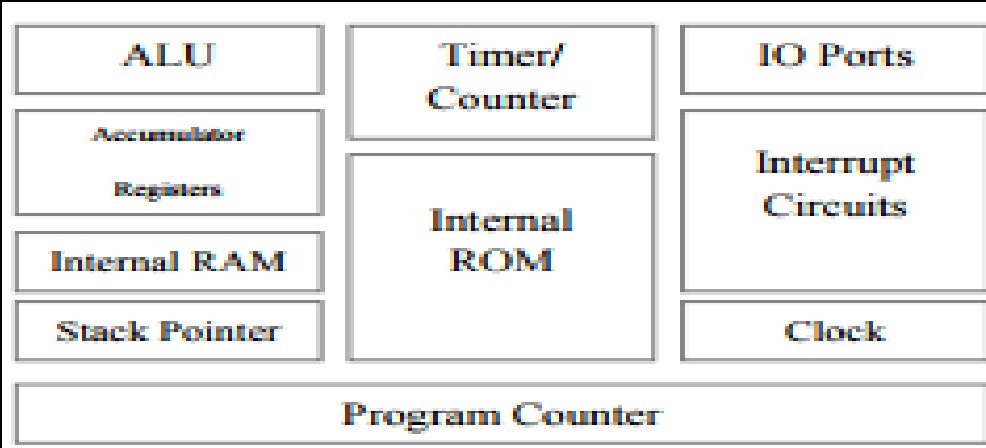


# 1.2 Microcontrollers and Embedded Processors



**Block diagram of microprocessor**

**Block diagram of microcontroller**

<b>Microprocessor</b>	<b>Microcontroller</b>
 <p>The block diagram of a microprocessor shows several components: an Arithmetic and logic unit at the top, followed by Accumulator and Working Registers. Below these are the Program Counter and Stack Pointer. At the bottom are the Clock Circuit and Interrupt circuit.</p>	 <p>The block diagram of a microcontroller shows components: ALU, Timer/Counter, and IO Ports at the top. Below these are Accumulator Registers, Internal RAM, and Stack Pointer. A large Internal ROM block is in the center. To the right are Interrupt Circuits and a Clock. At the bottom is the Program Counter.</p>
<b>Block diagram of microprocessor</b>	<b>Block diagram of microcontroller</b>
Microprocessor contains ALU, General purpose registers, stack pointer, program counter, clock timing circuit, interrupt circuit	Microcontroller contains the circuitry of microprocessor, and in addition it has built in ROM, RAM, I/O Devices, Timers/Counters etc.
It has many instructions to move data between memory and CPU	It has few instructions to move data between memory and CPU
Few bit handling instruction	It has many bit handling instructions
Less number of pins are multifunctional	More number of pins are multifunctional
Single memory map for data and code (program)	Separate memory map for data and code (program)
Access time for memory and IO are more	Less access time for built in memory and IO.
Microprocessor based system requires additional hardware	It requires less additional hardwares

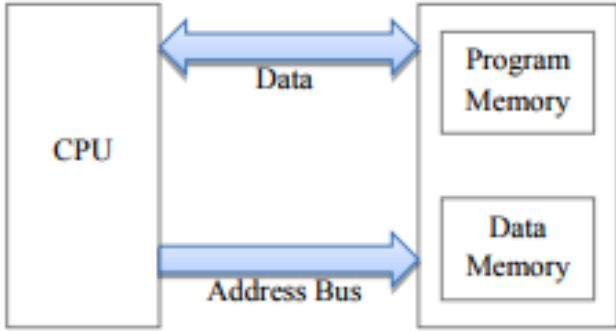
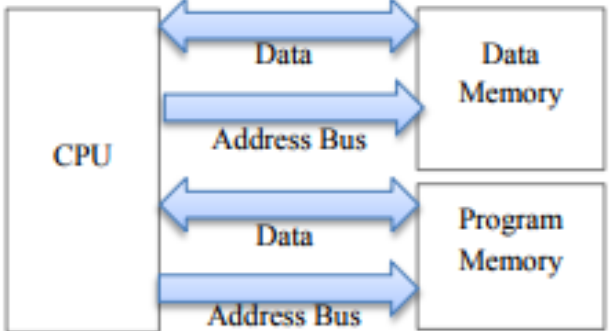
## 1.2.1 RISC AND CISC CPU ARCHITECTURES

SL.No.	RISC	CISC
1	Instruction takes one or two cycles	Instruction takes multiple cycles
2	Only load/store instructions are used to access memory	In additions to load and store instructions, memory access is possible with other instructions also
3	Instructions executed by hardware	Instructions executed by the micro program
4	Fixed format instruction	Variable format instructions
5	Few addressing modes	Many addressing modes
6	Few instructions	Complex instruction set
7	Most of the have multiple register banks	Single register bank
8	Highly pipelined	Less pipelined
9	Complexity is in the compiler	Complexity in the micro program

**Note :1 machine cycle = 12 clock pulses**

**8051 is a CISC**

## 1.2. 2 HARVARD & VON- NEUMANN CPU ARCHITECTURE

SL.No.	Von-Neumann (Princeton architecture)	Harvard architecture
1		
2	It uses single memory space for both instructions and data.	It has separate program memory and data memory
3	It is not possible to fetch instruction code and data	Instruction code and data can be fetched simultaneously
4	Execution of instruction takes more machine cycle	Execution of instruction takes less machine cycle
5	Uses CISC architecture	Uses RISC architecture
6	Instruction pre-fetching is a main feature	Instruction parallelism is a main feature
7	Also known as control flow or control driven computers	Also known as data flow or data driven computers
8	Simplifies the chip design because of single memory space	Chip design is complex due to separate memory space

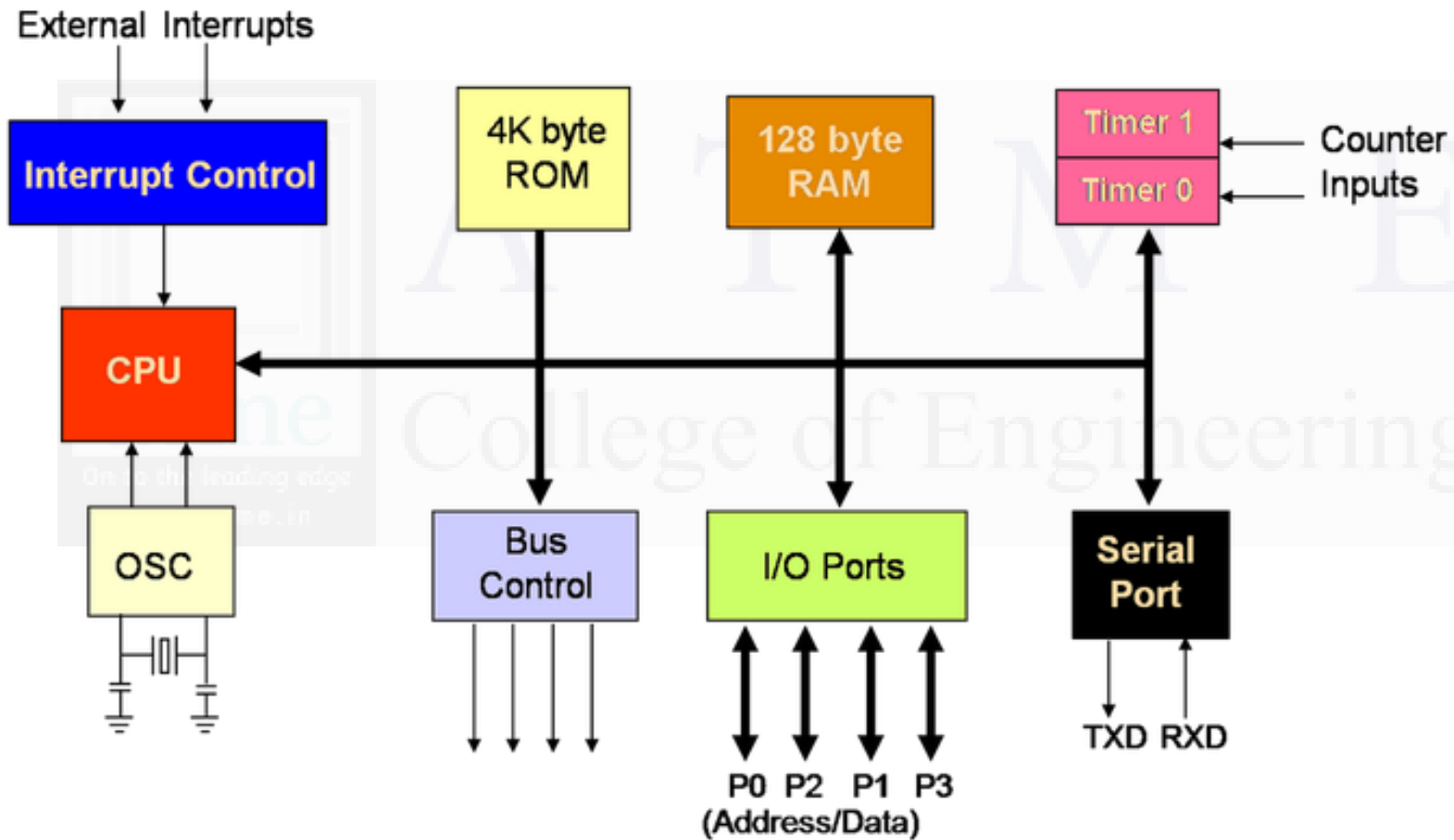




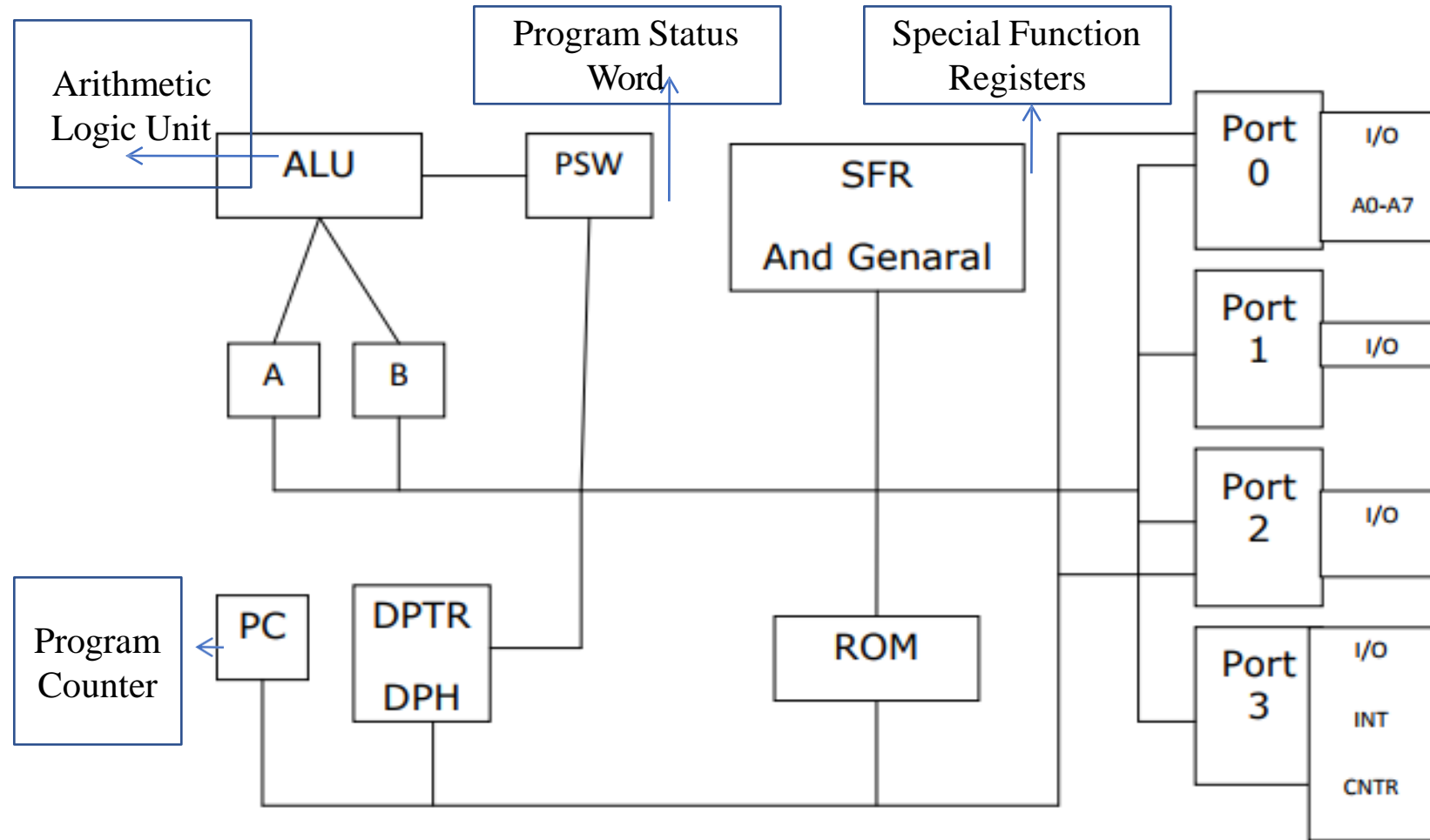
## Classification of Microcontrollers

Intel 4004	4 bit (2300 PMOS trans, 108 kHz)	1971
Intel 8048	8 bit	1976
Intel 8031	8 bit (ROM-less)	.
Intel 8051	8 bit (Mask ROM)	1980
Microchip PIC16C64	8 bit	1985
Motorola 68HC11	8 bit (on chip ADC)	.
Intel 80C196	16 bit	1982
Atmel AT89C51	8 bit (Flash memory)	.
Microchip PIC 16F877	8 bit (Flash memory + ADC)	.

## Block diagram of microcontroller



# Block diagram/ Schematic diagram of microcontroller



## Salient Features

- **Eight bit CPU** with registers A (Accumulator) and B
- Sixteen bit **Program counter (PC) and a data pointer (DPTR)**
- 8 Bit Program Status Word (PSW)
- 8 Bit Stack Pointer
- **4K Code Memory**
- **Internal Memory of 128 Bytes**
- **32 I/O Pins arranged as 4 port  $\rightarrow 4 \times 8 = 32$  I/O Pins**
- **Two 16 Bit Timer/Counter : T0, T1**
- Full Duplex serial data receiver/transmitter
- Control Registers : TCON, TMOD, SCON, PCON, IP and IE
- Two External and Internal Interrupt sources



## 1.4PSW and Flag Bits

This is an 8 bit register which contains the arithmetic status of ALU and the bank select bits of register banks.

PSW.7	PSW.6	PSW.5	PSW.4	PSW.3	PSW.2	PSW.1	PSW.0
<b>CY</b>	<b>AC</b>	<b>F0</b>	<b>RS1</b>	<b>RS0</b>	<b>OV</b>	<b>-</b>	<b>P</b>

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

CY	PSW.7	Carry flag.
AC	PSW.6	Auxiliary carry flag.
F0	PSW.5	Available to the user for general purpose.
RS1	PSW.4	Register Bank selector bit 1.
RS0	PSW.3	Register Bank selector bit 0.
OV	PSW.2	Overflow flag.
—	PSW.1	User-definable bit.
P	PSW.0	Parity flag. Set/cleared by hardware each instruction cycle to indicate an odd/even number of 1 bits in the accumulator.

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

- **P- Parity**
- **F0** - available to the user for general purpose
- **RS1, RS0** - register bank select bits
- **OV** - overflow
- **AC**- Auxiliary Carry
- **CY**- Carry

### Example 2-3

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

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

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

**Solution:**

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

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

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

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



## Example 2-2

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

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

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

**Solution:**

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

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

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

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

**Table 2-1: Instructions That Affect Flag Bits**

Instruction	CY	OV	AC
ADD	X	X	X
ADDC	X	X	X
SUBB	X	X	X
MUL	0	X	
DIV	0	X	
DA	X		
RRC	X		
RLC	X		
SETB C	1		
CLR C	0		
CPL C	X		
ANL C, bit	X		
ANL C, /bit	X		
ORL C, bit	X		
ORL C, /bit	X		
MOV C, bit	X		
CJNE	X		

*Note: X can be 0 or 1.*

## 1.5 8051 Register Banks and Stack

### Internal RAM organization

R7	1F	BANK 3
R6	1E	
R5	1D	
R4	1C	
R3	1B	
R2	1A	
R1	19	
R0	18	
R7	17	BANK 2
R6	16	
R5	15	
R4	14	
R3	13	
R2	12	
R1	11	
R0	10	
R7	0F	BANK 1
R6	0E	
R5	0D	
R4	0C	
R3	0B	
R2	0A	
R1	09	
R0	08	
R7	07	BANK 0
R6	06	
R5	05	
R4	04	
R3	03	
R2	02	
R1	01	
R0	00	

Working Registers

2F	7F							78
2E	77							70
2D	6F							68
2C	67							60
2B	5F							58
2A	57							50
29	4F							48
28	47							40
27	3F							38
26	37							30
25	2F							28
24	27							20
23	1F							18
22	17							10
21	0F							08
20	07							00

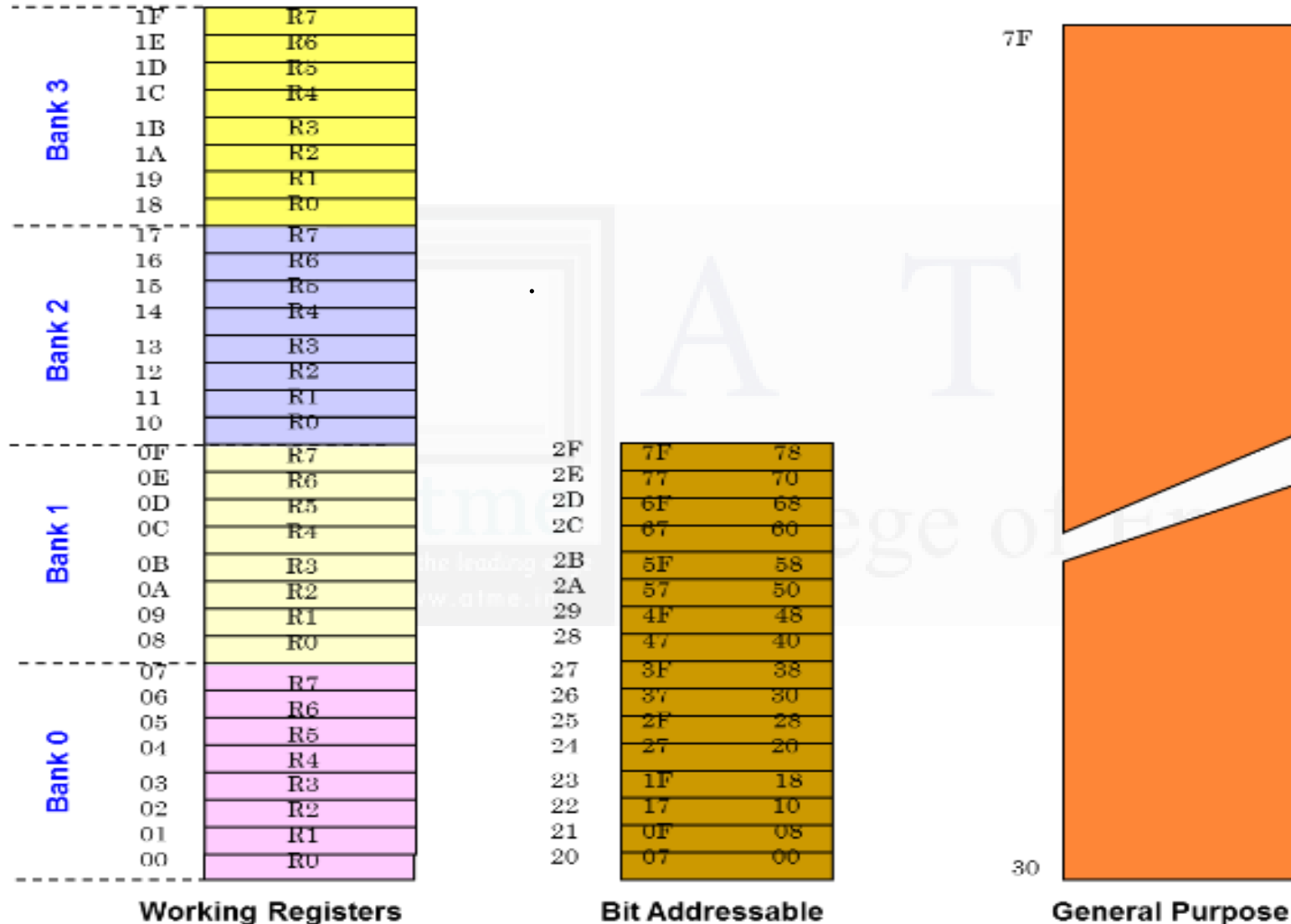
Bit addressable memory

7F
7E
.
.
.
.
.
.
.
32
31
30

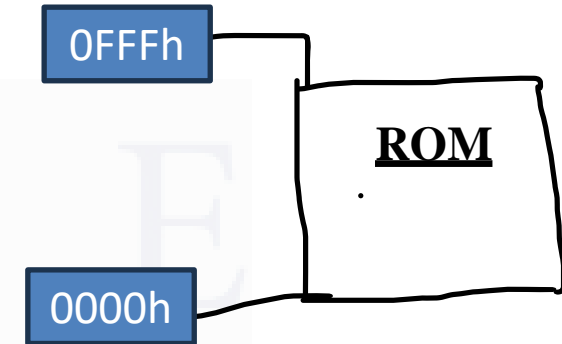
General purpose memory

## Internal RAM Structure

## Internal RAM Structure



- 4K bytes program memory ROM



## 1.5 8051 Register Banks and Stack

- **Register Banks:** 00h to 1Fh.
- The 8051 uses **8 general-purpose registers** R0 through R7 (R0, R1, R2, R3, R4, R5, R6, and R7).
- There are four such register banks.
- Selection of register bank can be done through RS1, RS0 bits of PSW.
- On reset, the default Register Bank 0 will be selected.

- **Bit Addressable RAM:** 20h to 2Fh.

- 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.

- In all there are 128 bytes 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.

### Example instructions are:

SETB 25h;            sets the bit 25h (becomes 1)

CLR 25h;            clears bit 25h (becomes 0)

## •General Purpose RAM:

- 80 Bytes Memory
- It is a good practice to use general purpose memory from 30 – 7Fh.
- The general purpose RAM can be accessed using direct or indirect addressing modes.

Internal RAM	Size
Working Registers	32Bytes
Bit Addressable	16Bytes
General Purpose	80Bytes
<b>Total</b>	<b>128 Bytes</b>



## 1.5.1 Stack

- A stack is a **last in first out memory**.
- In 8051 **internal RAM space can be used as stack**.
- The **address of the stack** is contained in a register called **stack pointer**.
- Instructions **PUSH and POP** are used for stack operations.
- When a data is to be placed on the stack, the stack pointer **increments before storing the data on the stack** so that the stack grows up as data is stored (pre-increment).
- As the data is retrieved from the stack the byte is read from the stack, and then **SP decrements to point the next available byte** of stored data (post decrement). The stack pointer is set to 07 when the 8051 resets.

# Example

Eg; Show the stack and SP for the following.

**[SP]=07 //CONTENT OF SP IS 07 (DEFAULT VALUE)**

**MOV R6, #25H ;**

**[R6] =25H //CONTENT OF R6 IS 25H**

**MOV R1, #12H ;**

**[R1] =12H //CONTENT OF R1 IS 12H**

**MOV R4, #0F3H ;**

**[R4] =F3H //CONTENT OF R4 IS F3H**

**PUSH 6 ;**

**[SP] =08 [08] = [06] =25H //CONTENT OF 08 IS 25H**

**PUSH 1 ;**

**[SP] =09 [09] = [01] =12H //CONTENT OF 09 IS 12H**

**PUSH 4 ;**

**[SP] =0A [0A] = [04] =F3H //CONTENT OF 0A IS F3H**

# QUIZ

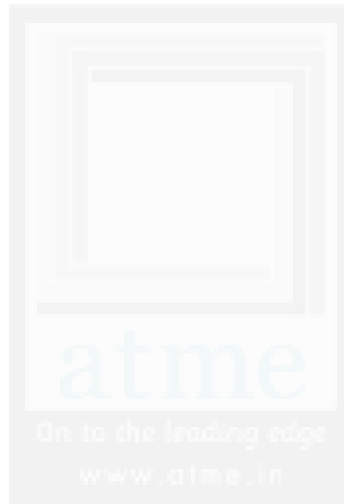
**a. 8051 has Internal RAM of capacity**

1. 4K Bytes
2. 128 Bytes
3. 8K Bytes
4. 100K Bytes

# QUIZ

**b. 8051 has how many ports?**

1. 3
2. 4
3. 5
4. 8



**A T M E**  
College of Engineering

# QUIZ

**c. 8051 has how many 16Bit registers?**

1. 2
2. 3
3. 4
4. 5

# QUIZ

**d. When Carry flag is set there is a bit set at?**

1. D6
2. D3
3. D7
4. D1



# QUIZ

**e. The address of Stack is stack pointer and it uses instructions**

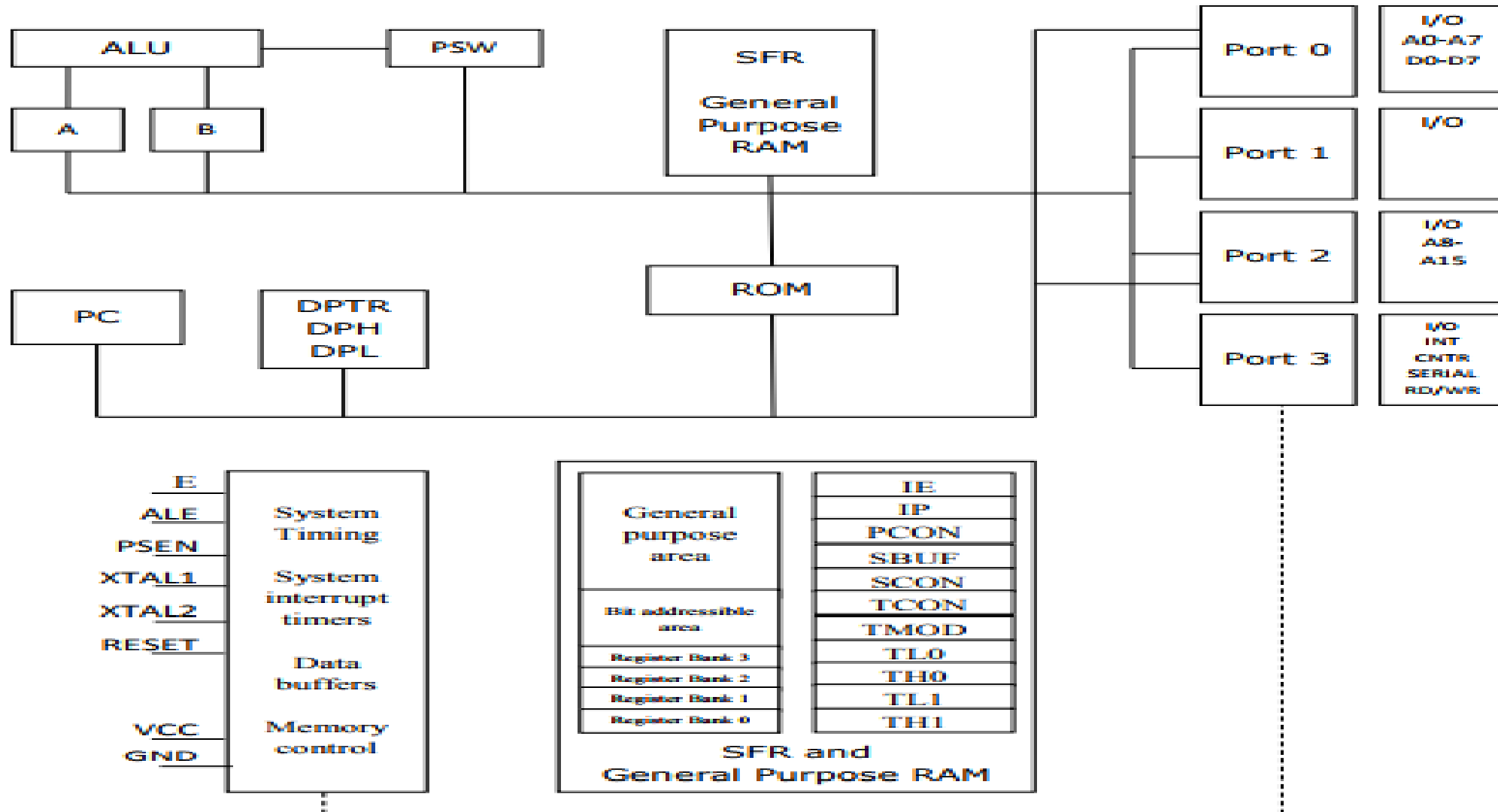
1. PRINT & POP
2. PAUSE & POP
3. PUSH & POP
4. PLAY & POP

# QUIZ

**f. PSW is an 8 bit register and contains**

1. Time of the register
2. Address of the register
3. Arithmetic status of the register
4. None

## Block Diagram of 8051- Architecture of 8051



# Features

- **Eight bit CPU**
- On chip clock oscillator
- **4Kbytes of internal program memory** (code memory) [ROM]
- **128 bytes of internal data memory** [RAM]
- **64 Kbytes of external program memory** address space.
- **64 Kbytes of external data memory** address space.
- 32 bi-directional I/O lines (can be used as four 8 bit ports or 32 individually addressable I/O lines) Two 16 Bit Timer/Counter :T0, T1
- **Full Duplex serial data receiver/transmitter**

- Four Register banks with 8 registers in each bank.
- **Sixteen bit Program counter (PC) and a data pointer (DPTR)**
- **8 Bit Program Status Word (PSW)**
- **8 Bit Stack Pointer**
- **Five vector interrupt structure** (RESET not considered as an interrupt.)
- **8051 CPU consists of 8 bit ALU** with associated registers like accumulator 'A', B register,
- **PSW, SP, 16 bit program counter, stack pointer.** ALU can perform arithmetic and logic functions on 8 bit variables.
- 8051 has 128 bytes of internal RAM which is divided into
- Working registers [00 – 1F] ;Bit addressable memory area [20 – 2F] ; General purpose memory area (Scratch pad memory) [30-7F]

- 8051 has 4 K Bytes of internal ROM. The address space is from 0000 to 0FFFh. **If the program size is more than 4 K Bytes 8051 will fetch the code automatically from external memory.**
- Accumulator is an 8 bit register widely used for all arithmetic and logical operations.
- **Accumulator is also used to transfer data between external memory.**
- **B register is used along with Accumulator** for multiplication and division. A and B registers together is also called MATH registers.



## 1.5 IO Port Usage in 8051

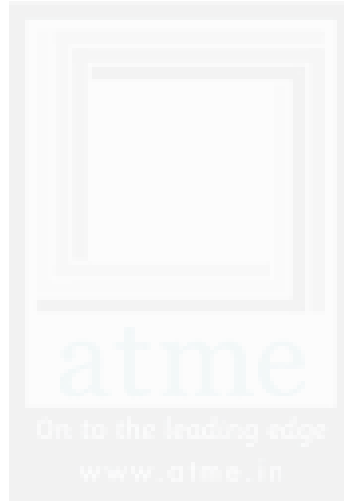
- Out of 40 pins 24 pins may each be used for one or two entirely different functions
- But the port pins have been multiplexed to perform different functions to make 8051 as 40 Pin IC



A T M E  
College of Engineering



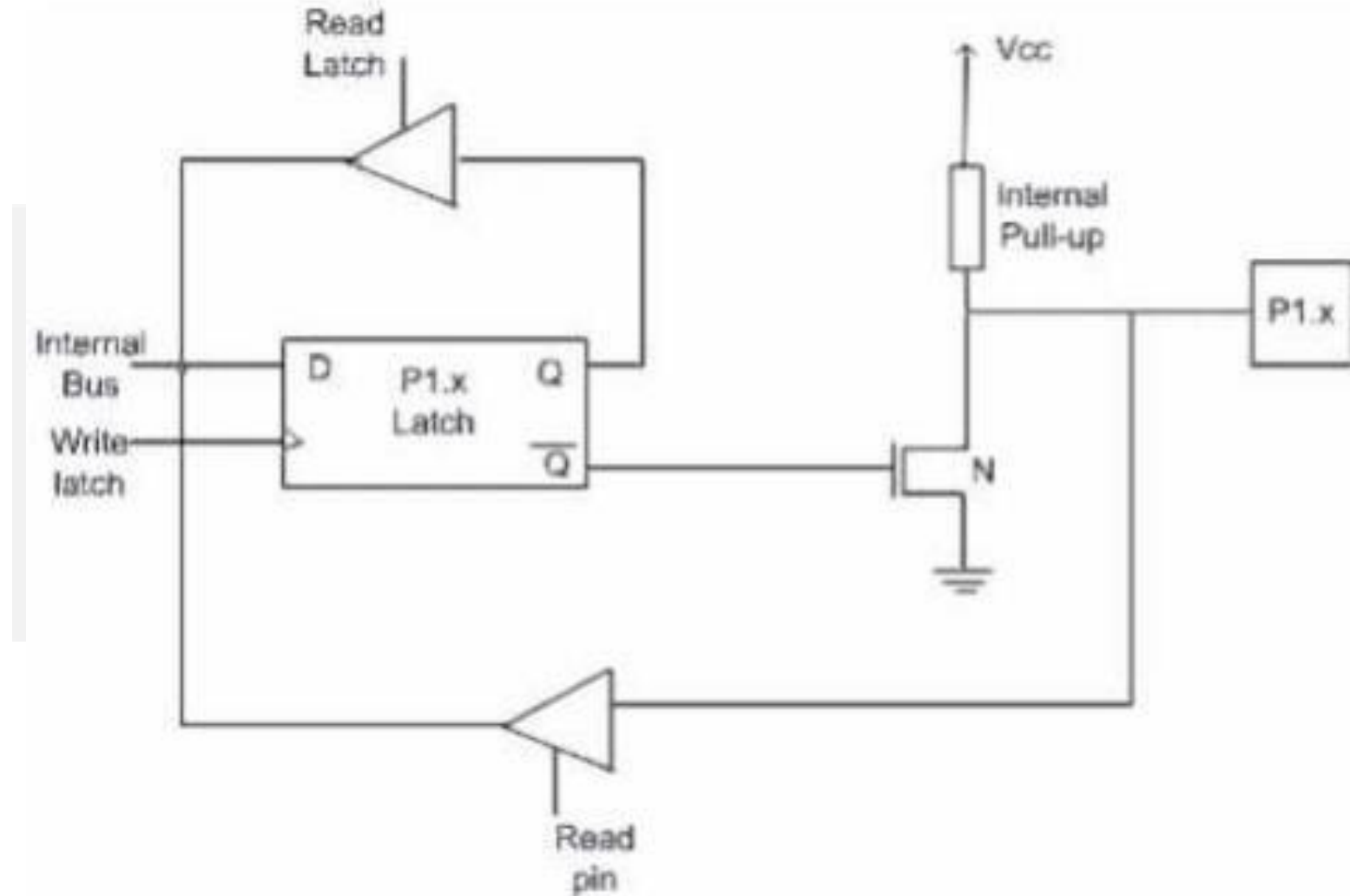
# PORT -1



A T M E  
College of Engineering



Working of 8051 Input - Output Ports.mp4

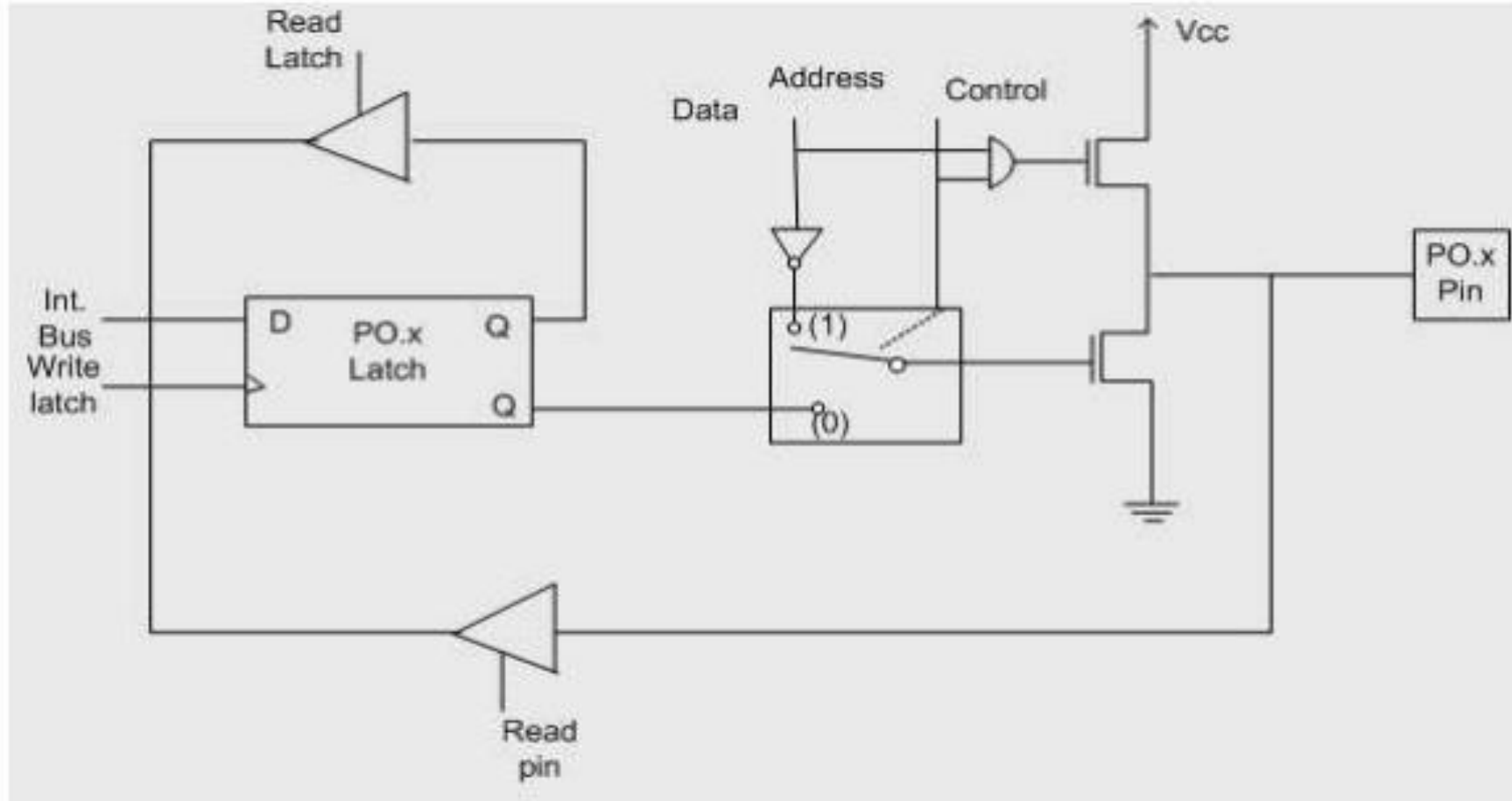


- Port-1 has 8 pins (P1.1-P1.7) .The structure of a port-1 pin is shown in Fig.
- Port-1 **does not have any alternate function** i.e. it is dedicated solely for **I/O interfacing**. When used as output port, the pin is pulled up or down through internal pull-up.
- **To use port- 1 as input port, '1' has to be written to the latch**. In this input mode when '1' is written to the pin by the external device then it reads fine.
- **But when '0' is written to the pin by the external device then the external source must sink current due to internal pull-up**. If the external device is not able to sink the current the pin voltage may rise, leading to a possible wrong reading.

# Port-0



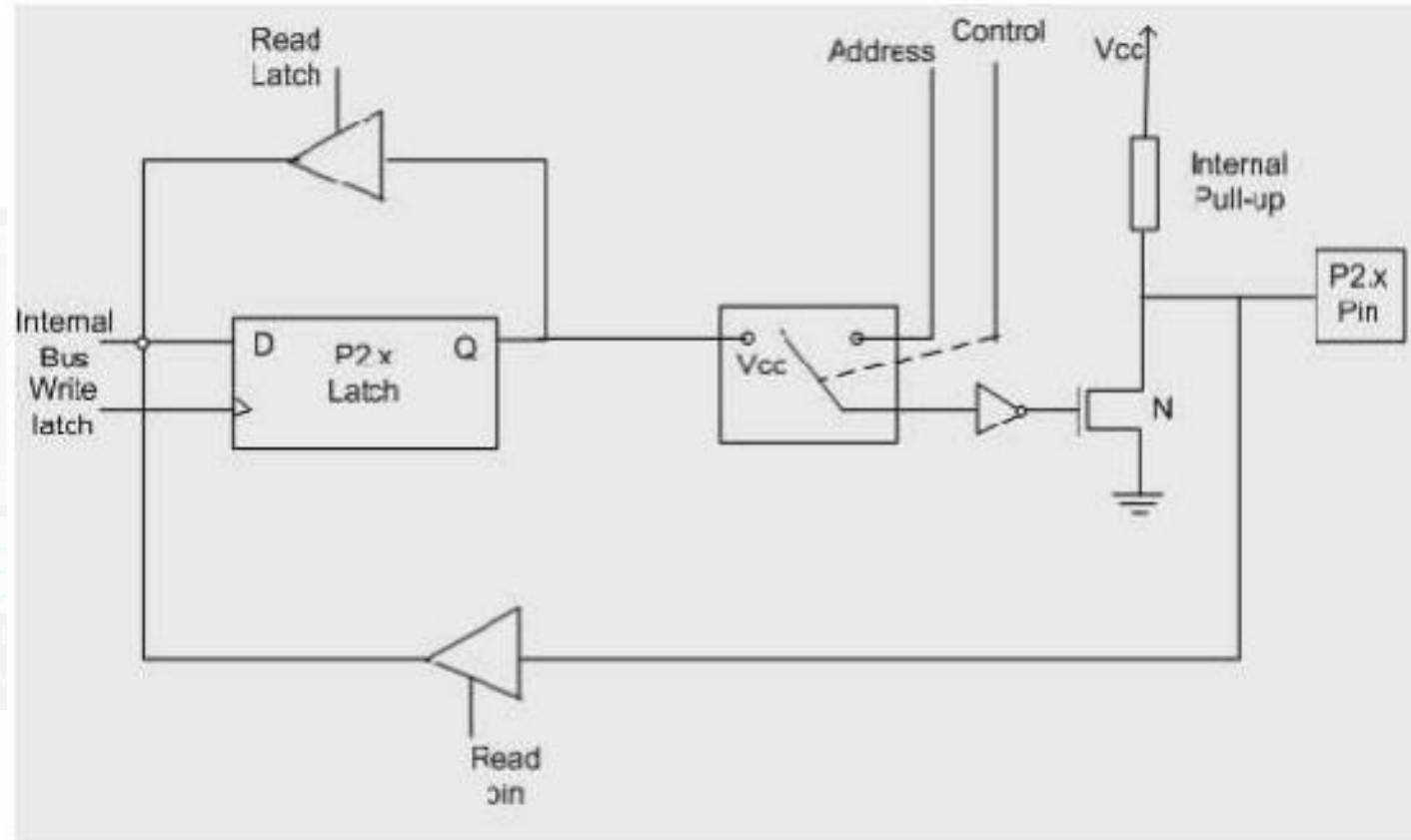
videoplayback.mp4



Port-0

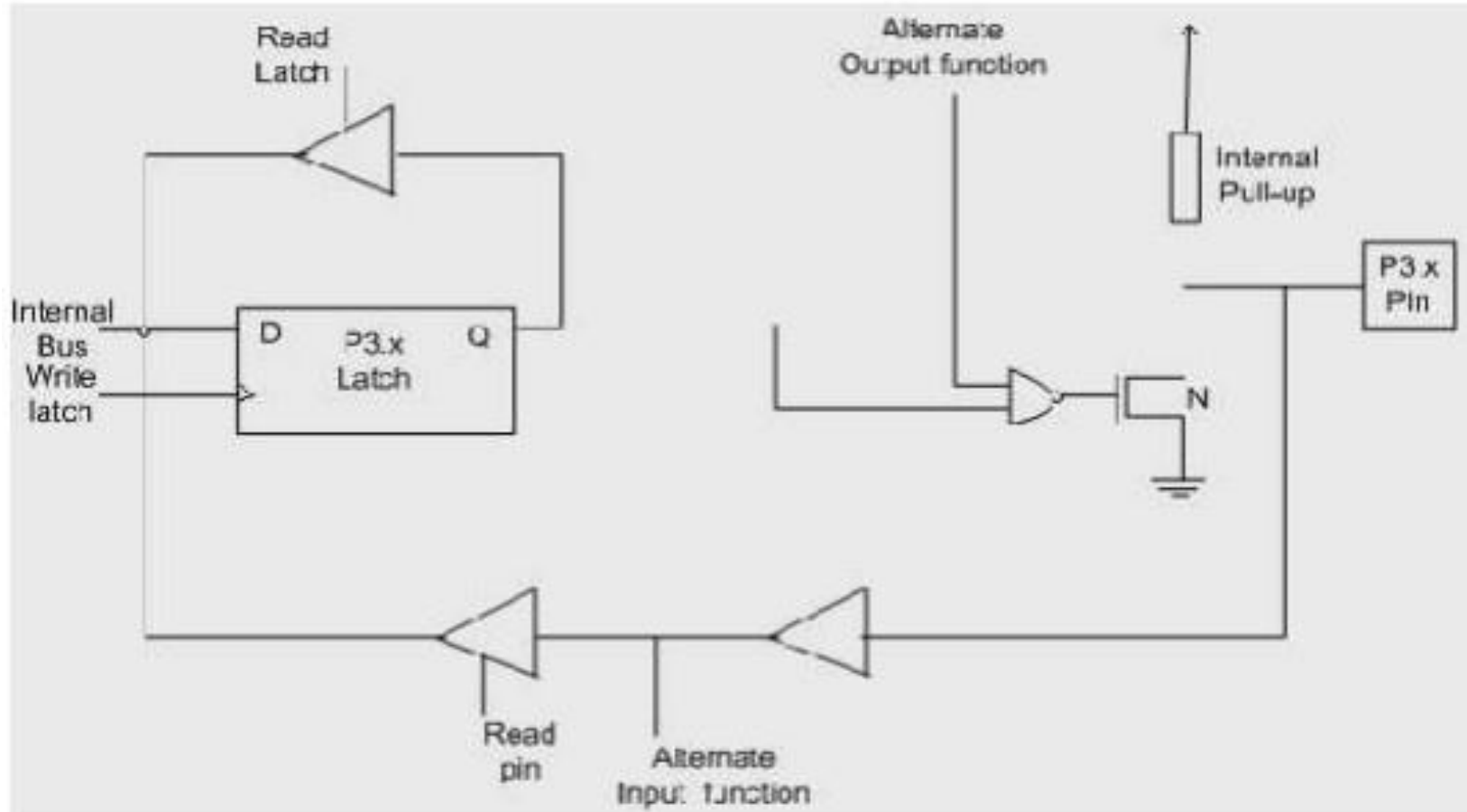
- Port -0 has 8 pins (P0.0-P0.7).The structure of a Port-0 pin is shown in Fig.
- **Port-0 can be configured as a normal bidirectional I/O port or it can be used for address/data interfacing for accessing external memory.**
- **CASE-1: When control is '1', the port is used for address/data interfacing.**
- **CASE-2: When the control is '0', the port can be used as a normal bidirectional I/O port.**





## Port-2

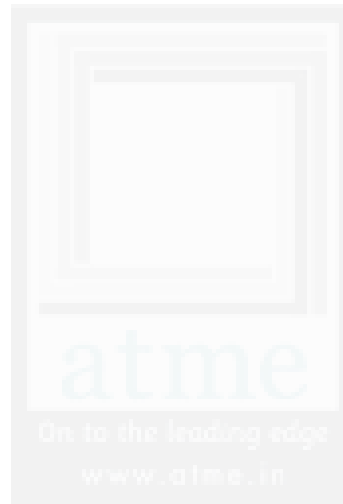
- Port-2 has 8-pins (P2.0-P2.7) . The structure of a port-2 pin is shown in Fig
- Port-2 is used for higher external address byte [A8 to A15] or a normal input/output port. The I/O operation is similar to Port-1.
- Port-2 latch remains stable when Port-2 pin are used for external memory access. Here again due to internal pull-up there is limited current driving capability.



## Port-3

- Each pin of Port-3 can be individually programmed for **I/O operation or for alternate function**. The alternate function can be activated only if the corresponding latch has been written to '1'.
- To use the port as **input port**, '1' should be written to the latch. This port also has internal pull-up and limited current driving capability.

# Alternate functions of Port-3 pins –



<b>P3.0</b>	<b>RxD</b>
<b>P3.1</b>	<b>TxD</b>
<b>P3.2</b>	<b>INT0</b>
<b>P3.3</b>	<b>INT1</b>
<b>P3.4</b>	<b>T0</b>
<b>P3.5</b>	<b>T1</b>
<b>P3.6</b>	<b>WR</b>
<b>P3.7</b>	<b>RD</b>



# QUIZ

1.8051 Microcontroller has how much External RAM

- a. 128Bytes
- b. 64KBytes
- c. 4KBytes
- d. 0Bytes

**2. The address of Stack is stack pointer and it uses instructions**

**QUIZ**

1. PRINT & POP
2. PAUSE & POP
3. PUSH & POP
4. PLAY & POP

# QUIZ

3. When control is 1, Port 0 functions as

- a. address/data bus controls
- b. Bidirectional I/O Port
- c. TxD
- d. RST



# Special Function Registers

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

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

Timer 0

<i>Name of the Register</i>	<i>Function</i>	<i>Internal RAM Address (HEX)</i>
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



# Ports



	1	1	1	1	1	1	1	(Value on RESET)
	P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0
P0	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
	Address = 80H							

	1	1	1	1	1	1	1	1	(Value on RESET)
<b>P1</b>	P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0	
	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Address = 90H

	1	1	1	1	1	1	1	1	(Value on RESET)
	P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0	
P2	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Address = A0H

	1	1	1	1	1	1	1	1	(Value on RESET)
P3	P3.7	P3.6	P3.5	P3.4	P3.3	P3.2	P3.1	P3.0	
	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Address = B0H

# Peripheral Control Registers

## PCON (Power Control)

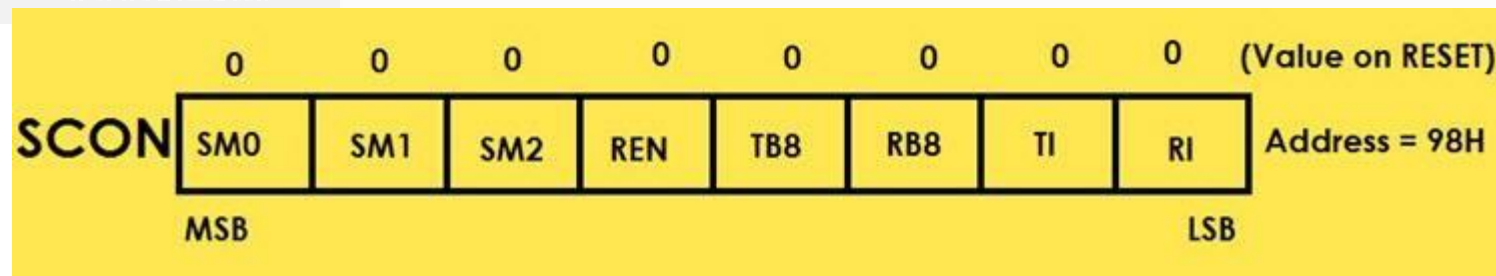
- The PCON or Power Control register, is used to control the 8051 Microcontroller's Power Modes and is located at 87H of the SFR Memory Space.
- Using two bits in the PCON Register, the microcontroller can be set to Idle Mode and Power Down Mode.
- During Idle Mode, the Microcontroller will stop the Clock Signal to the ALU (CPU) but it is given to other peripherals like Timer, Serial, Interrupts, etc. In order to terminate the Idle Mode, you have to use an Interrupt or Hardware Reset.
- In the Power Down Mode, the oscillator **will be stopped** and the power will be reduced to 2V. To terminate the Power Down Mode, you have to use the Hardware Reset.



PCON Register can also be used for few additional purposes. The **SMOD Bit** in the PCON Register is used to control the **Baud Rate of the Serial Port**.



- The **Serial Control** or **SCON SFR** is used to control the 8051 Microcontroller's **Serial Port**.
- It is located as an address of **98H**.
- Using SCON, you can control the Operation Modes of the **Serial Port**, **Baud Rate of the Serial Port** and **Send or Receive Data** using Serial Port.



# SCON (Serial Control)

## Serial Port Mode Control Bits

SM0	SM1	Mode	Description	Baud Rate
0	0	0	8-Bit Synchronous Shift Register Mode	Fixed Baud Rate ( Frequency of oscillator / 12)
0	1	1	8-bit Standard UART mode	Variable Baud Rate (Can be set by Timer 1)
1	0	2	9-bit Multiprocessor Comm. mode	Fixed Baud Rate ( Frequency of oscillator / 32 or Frequency of oscillator / 64
1	1	3	9-bit Multiprocessor Comm. mode	Variable Baud Rate (Can be set by Timer 1)

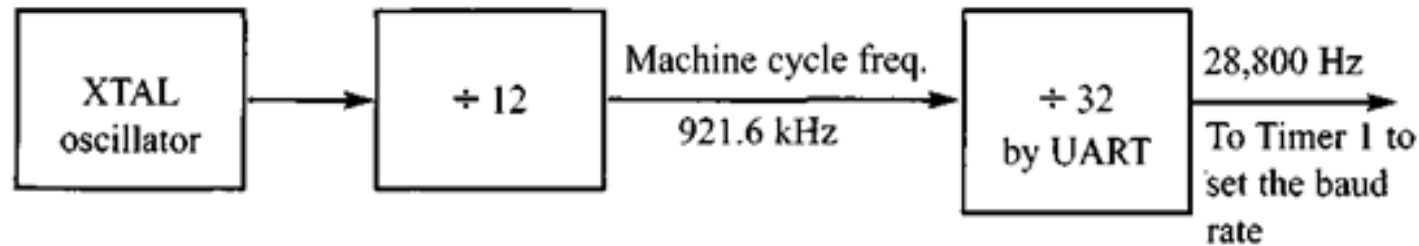


## Example

- **With XTAL = 11.0592 MHz**

The machine cycle frequency of the 8051 =  $11.0592 \text{ MHz} / 12 = 921.6 \text{ kHz}$ , and

$921.6 \text{ kHz} / 32 = 28,800 \text{ Hz}$  is the frequency provided by UART to Timer 1 to set baud rate.



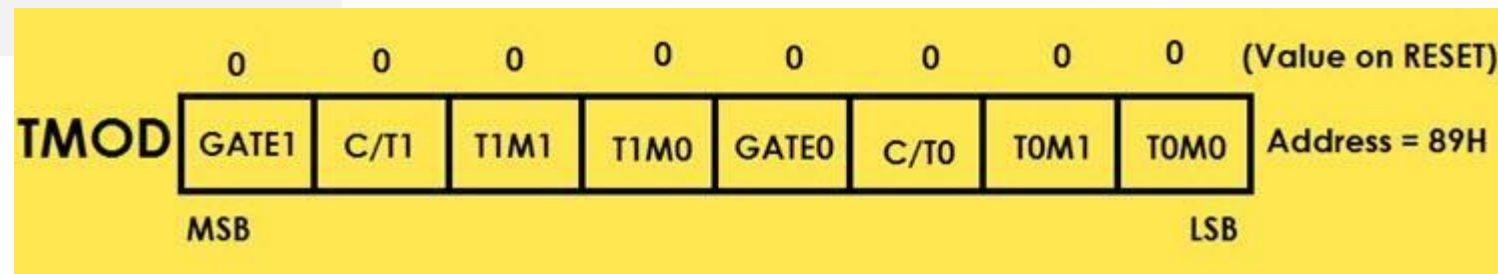
# TCON (Timer Control)

- Timer Control or TCON Register is used to start or stop the Timers of 8051 Microcontroller.
- It also contains bits to indicate if the Timers has overflowed.
- The TCON SFR also consists of Interrupt related bits.



# TMOD (Timer Mode)

- The TMOD or Timer Mode register or SFR is used to set the Operating Modes of the Timers T0 and T1.
- The lower four bits are used to configure Timer0 and the higher four bits are used to configure Timer1.



TxM <sub>0</sub>	TxM <sub>1</sub>	Mode	Description
0	0	0	13-bit Timer Mode (THx – 8-bit and TLx – 5-bit)
0	1	1	16-bit Timer Mode
1	0	2	8-bit Auto Reload Timer Mode
1	1	3	Two 8-bit Timer Mode or Split Timer Mode

# QUIZ

**1.How many SFRs are there in 8051 Microcontrollers**

- a. 22
- b. 40
- c. 21
- d. 8

# QUIZ

## 2. 8051 Microcontroller has oscillator frequency of:

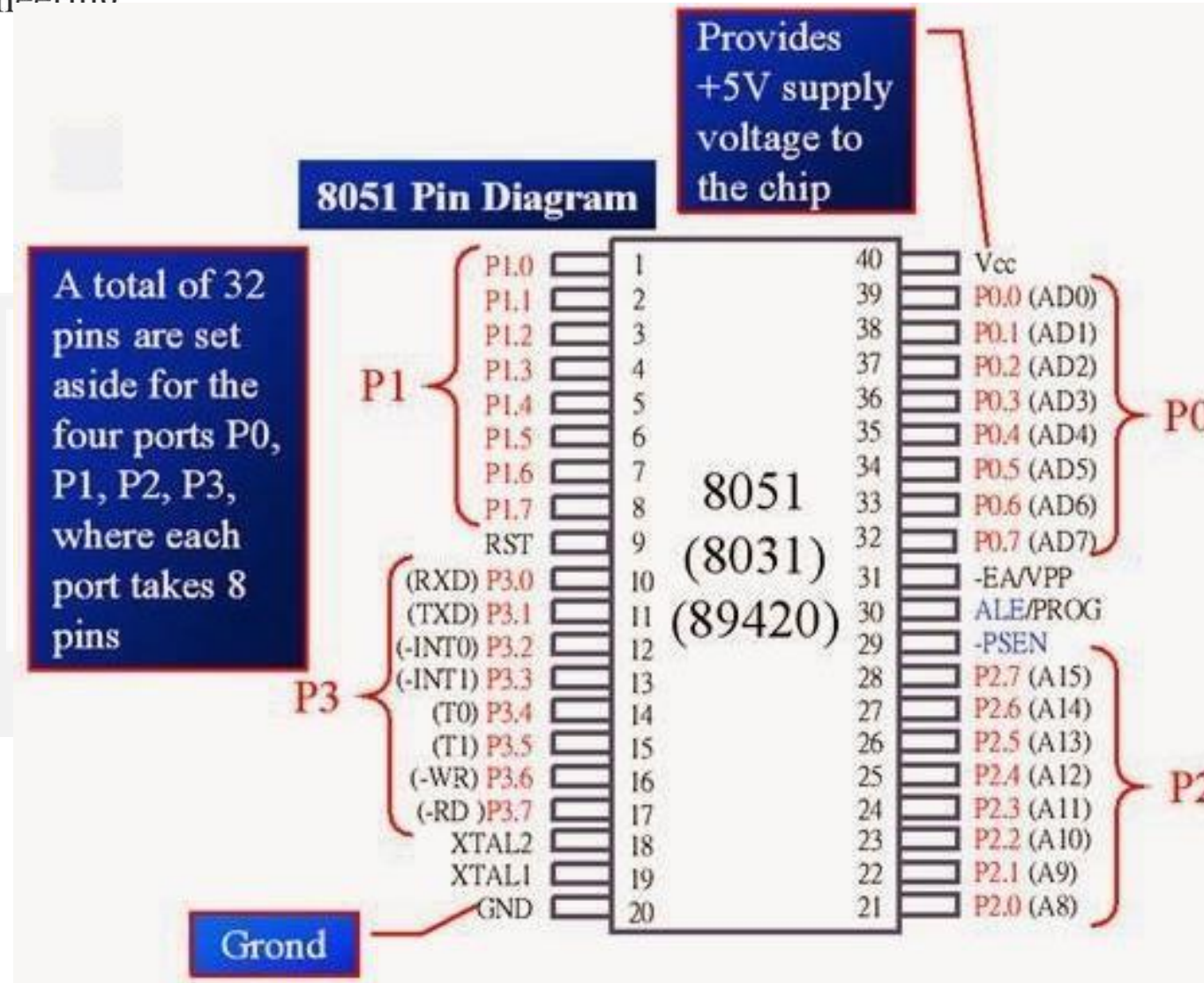
- a. 13MHz
- b. 11.0592MHz
- c. 10MHz
- d. 6MHz

# QUIZ

**3. Two bits in the PCON Register are:**

- a. Idle and Power Up
- b. Idle and Power down
- c. RST & Stop
- d. SCON & TCON

# 1.7 Pins Of 8051





<b>Pins 1-8</b>	<b>PORT 1.</b> Each of these pins can be configured as an <u>input or an output.</u>
<b>Pin 9</b>	<b>RESET.</b> <u>A logic one on this pin disables the microcontroller</u> and clears the contents of most registers. By applying logic zero to this pin, the program starts execution from the beginning.
<b>Pins 10-17</b>	<b>PORT 3.</b> Similar to port 1, each of these pins can serve as general <u>input or output. Besides, all of them have alternative functions</u>
<b>Pin 10</b>	<b>RXD.</b> Serial asynchronous communication input or Serial synchronous communication output.
<b>Pin 11</b>	<b>TXD.</b> Serial asynchronous communication output or Serial synchronous communication clock output.

Pin 12	<b>INT0</b> . External Interrupt 0 input
Pin 13	<b>INT1</b> . External Interrupt 1 input
Pin 14	<b>T0</b> . Counter 0 clock input
Pin 15	<b>T1</b> . Counter 1 clock input
Pin 16	<b>WR</b> . Write to external (additional) RAM
Pin 17	<b>RD</b> . Read from external RAM

XTAL2, XTAL1. Internal oscillator input and output. A quartz crystal which specifies operating frequency is usually connected to these pins.

Pin 20

GND. Ground.

Pin 21-28

Port 2. If there is no intention to use external memory then these port pins are configured as general inputs/outputs. In case external memory is used, the higher address byte, i.e. addresses A8-A15 will appear on this port.

Pin 29

PSEN. If external ROM is used for storing program then a logic zero (0) appears on it every time the microcontroller reads a byte from memory.

Pin 30

ALE. Prior to reading from external memory, the microcontroller puts the lower address byte (A0-A7) on P0 and activates the ALE output.

- After receiving signal from the ALE pin, the external latch latches the state of P0 and uses it as a memory chip address.
- Immediately after that, the ALE pin is returned its previous logic state and P0 is now used as a Data Bus.

Pin 31	<b>EA:</b> By applying <b>logic zero to this pin, P2 and P3 are used for data and address transmission</b> with no regard to whether there is internal memory or not. It means that even there is a program written to the microcontroller, it will not be executed. Instead, the program written to external ROM will be executed.
Pin 32-39	<b>PORT 0.</b> Similar to P2, if external memory is not used, these pins can be used as <b>general inputs/outputs</b> . Otherwise, <b>P0 is configured as address output (A0-A7) when the ALE pin is driven high (1) or as data output (Data Bus) when the ALE pin is driven low (0).</b>
Pin 40	VCC. +5V power supply

## Memory Address Decoding, 8031/51 Interfacing With External ROM and RAM

- **8031 chip** is a **ROM less version of the 8051**
- Before we discuss this topic, one might wonder why someone would want to use the 8031 when they could buy an **8751, 89C51, or DS5000**
- The reason is that all these chips have a limited amount of on-chip ROM.
- Therefore, in many systems where the on-chip ROM of the 8051 is not sufficient, the use of an 8031 is ideal since it allows the **program size to be as large as 64K bytes**

## Memory Address Decoding, 8031/51 Interfacing With External ROM and RAM

- The CPU provides the address of the data desired, but it is the job of the **decoding circuitry** to locate the selected memory block.
- Memory chips have one or more pins called **CS (chip select)**, which must be **activated for the memory's contents to be accessed.**

## Memory Address Decoding, 8031/51 Interfacing With External ROM and RAM

Sometimes the chip select is also referred to as chip enable (CE). In connecting a memory chip to the CPU, note the following points.

1. The **data bus of the CPU** is connected directly to the **data pins of the memory chip**.
2. Control signals **RD (read) and WR (memory write)** from the CPU are connected to the **OE (output enable) and WE (write enable) pins of the memory chip**, respectively.
3. In the case of the address buses, while the lower bits of the addresses from the CPU go directly to the memory chip address pins, the **upper ones are used to activate the CS pin of the memory chip**.
4. It is the **CS pin** that along with **RD/WR** allows the flow of data in or out of the memory chip.
5. No data can be written into or read from the memory chip **unless CS is activated**.



## Memory Address Decoding, 8031/51 Interfacing With External ROM and RAM

- We connect the **EA pin to  $V_{cc}$**  to indicate that the program code is stored in the microcontroller's **on-chip ROM**
- To indicate that the program code is stored in **external ROM**, this pin must be connected to GND.

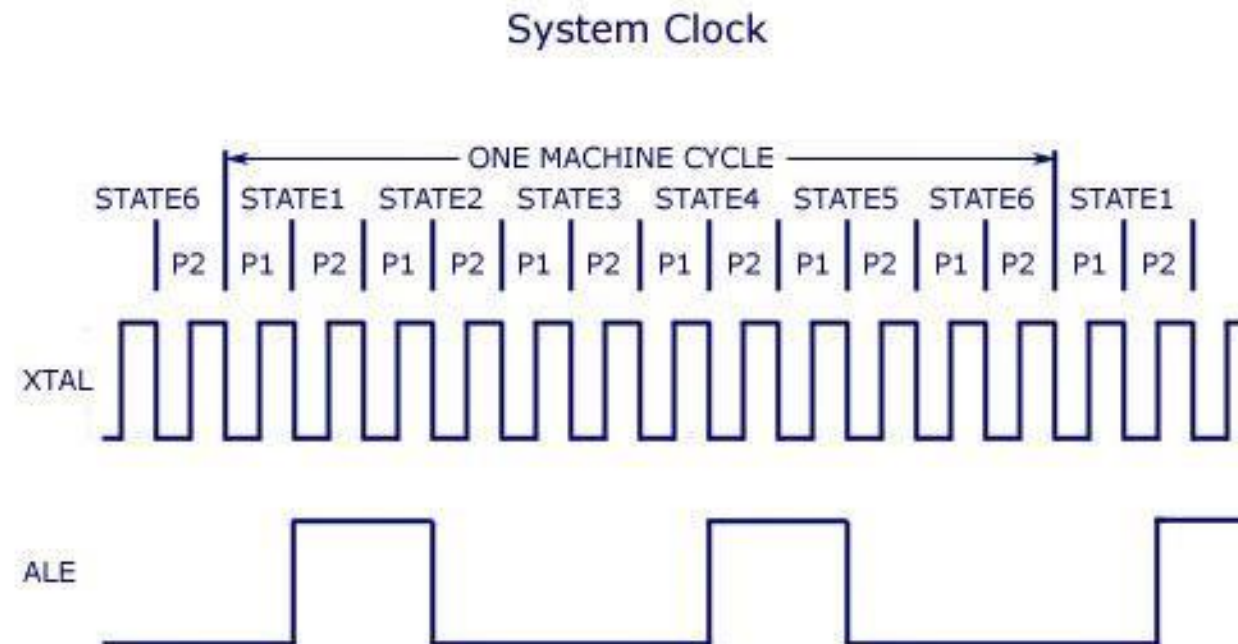


## Memory Address Decoding, 8031/51 Interfacing With External ROM and RAM

- In the 8031/51, port 0 and port 2 provide the 16-bit address to access external memory.
- **P0** provides the lower 8 bit addresses A0 – A7, and **P2** provides the upper 8 bit addresses A8 – A15
- **P0** is also used to provide the 8-bit data bus D<sub>0</sub> – D<sub>7</sub>.
- In other words, **pins P0.0 – P0.7 are used for both the address and data paths**. This is called **address/data multiplexing** in chip design

## One Machine Cycle

One complete oscillation of the clock source is called a **pulse**. **Two pulses forms a state** and **six states forms one machine cycle**. Also note that, two pulses of ALE are available for 1 machine cycle.



**Note :1 machine cycle = 12 clock pulses**

## 8031/51 Interfacing With External ROM and RAM

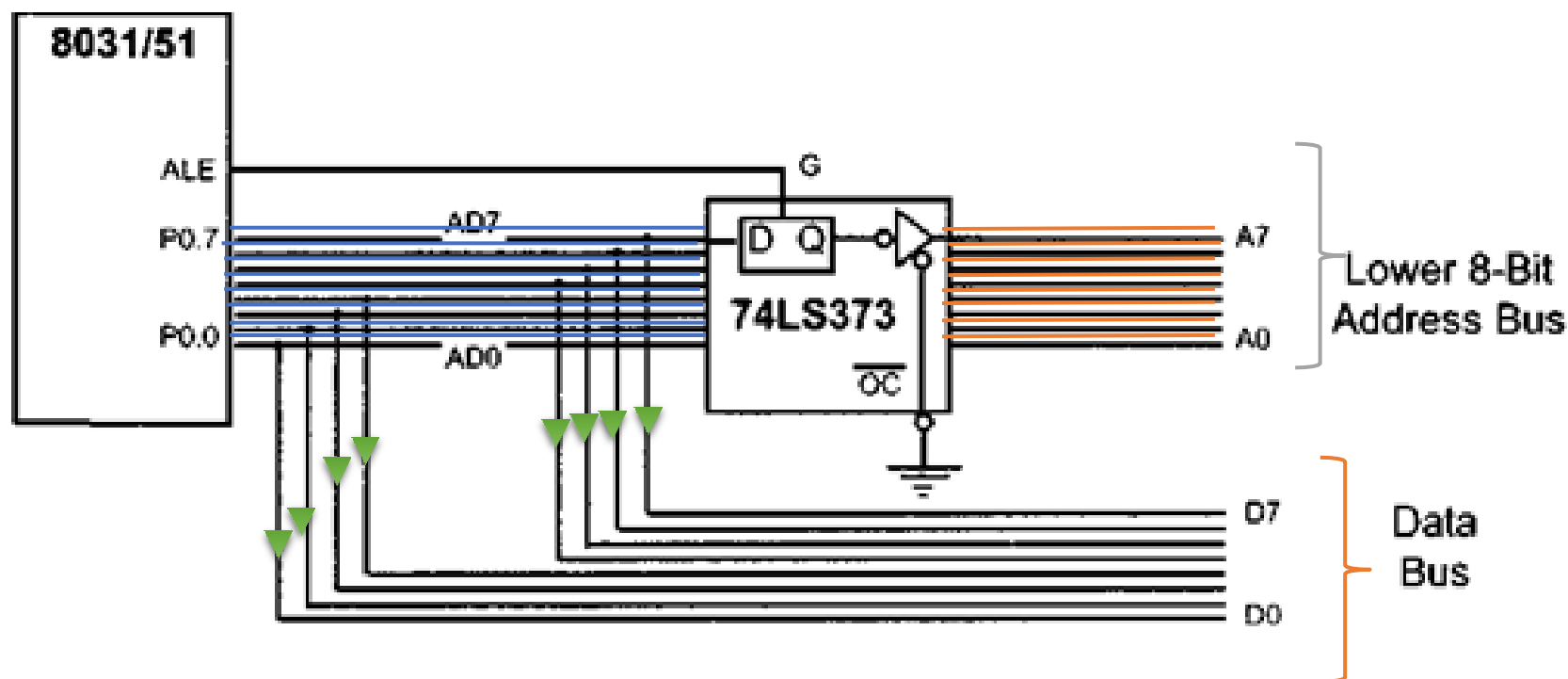
**How do we know when **P0** is used for the data path and when it is used for the address path?**

ALE (address latch enable) pin.

ALE is an output pin for the 8031/51 microcontroller.

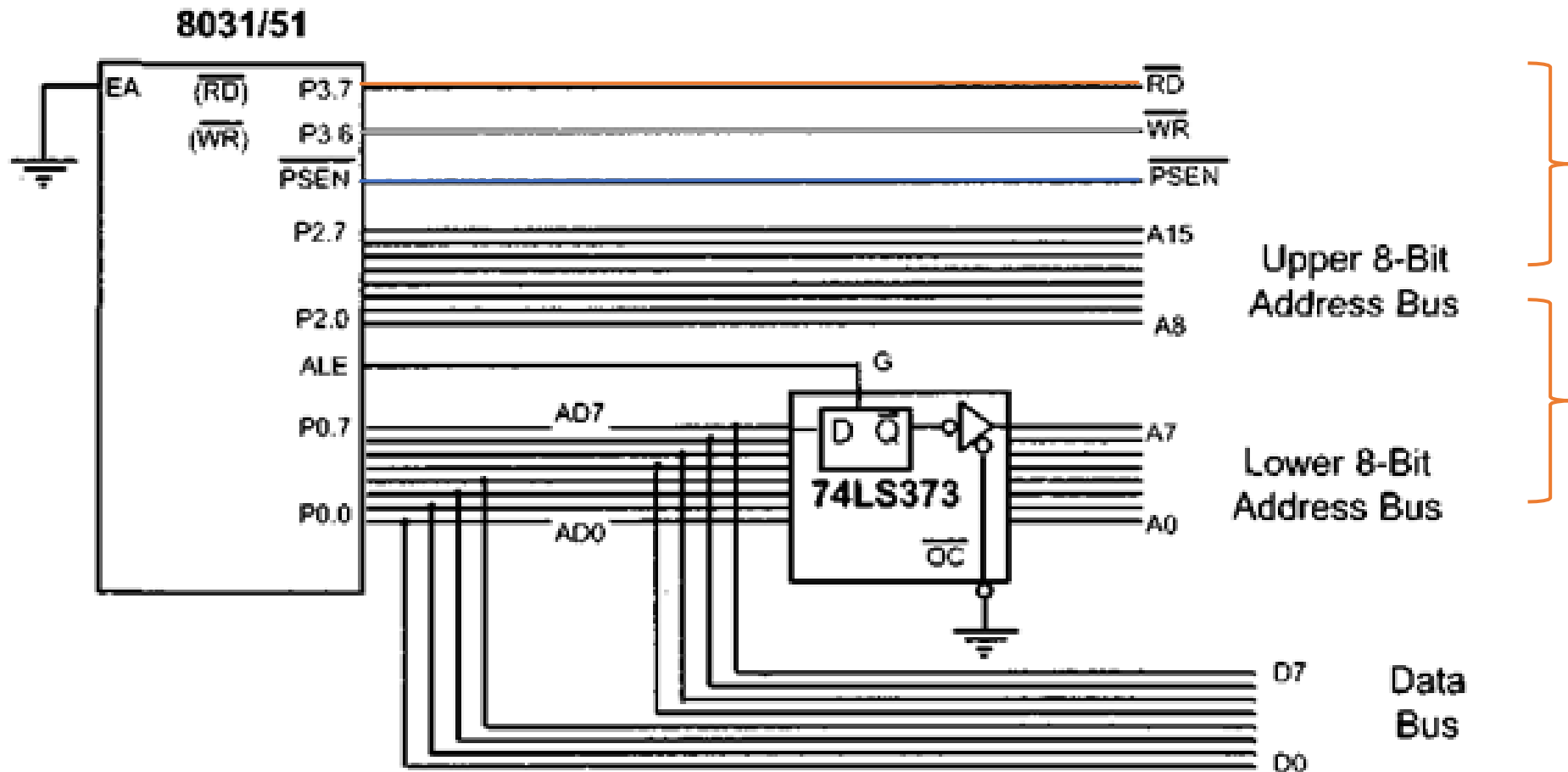
**CASE 1:** when **ALE = 0** the 8031/8051 uses P0 for the data path,  
**CASE 2:** **ALE = 1**, it uses it for the address path

**Step 1:** We connect  $P_0$  to a 74LS373 latch and use the **ALE pin** to latch the address



**Note that normally  $ALE = 0$ , and PO is used as a data bus**

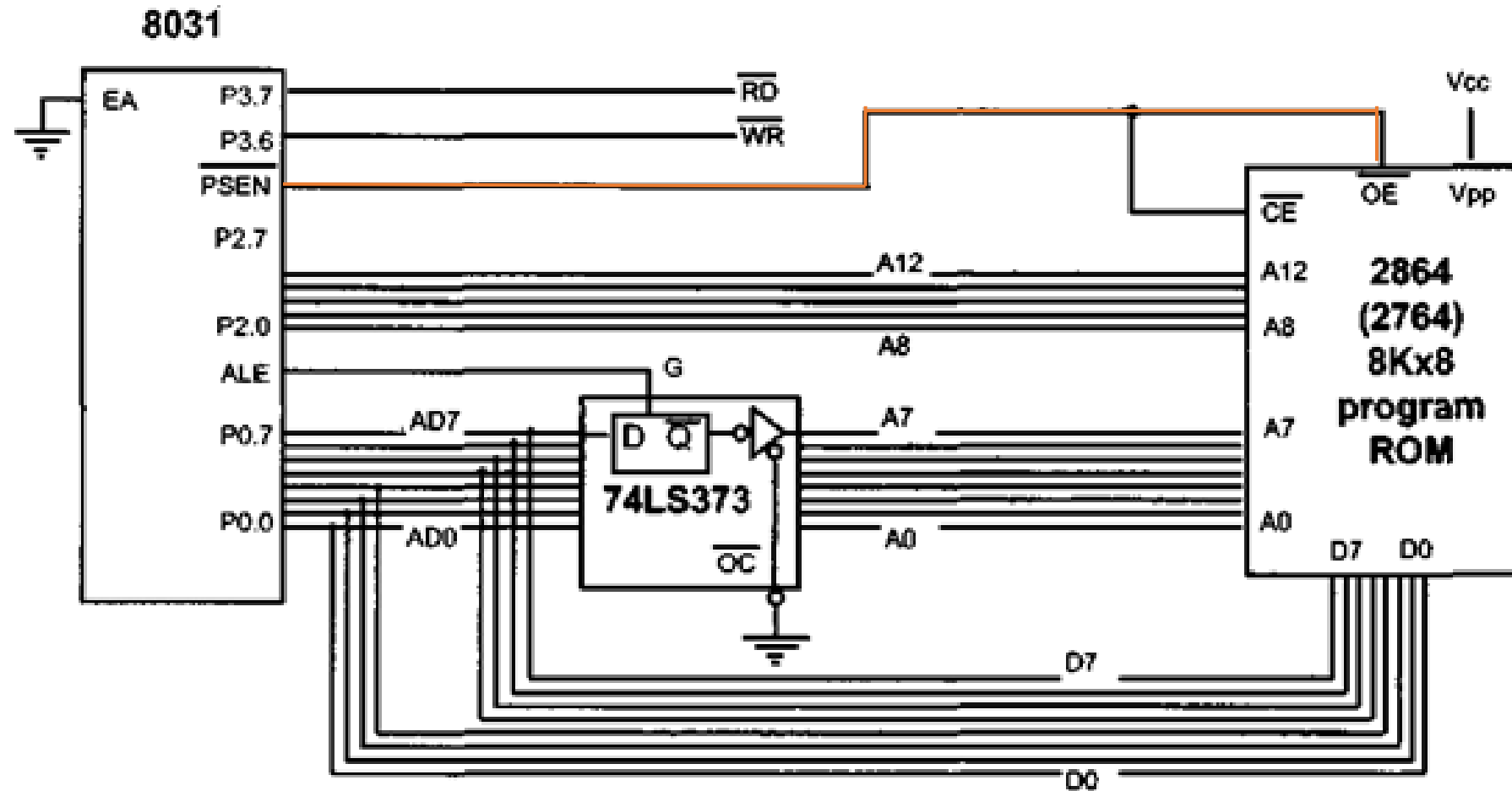
**Step 2:** Whenever the 8031/51 wants to use **P<sub>0</sub> as an address bus**, it puts the addresses A<sub>0</sub> – A<sub>7</sub> on the P<sub>0</sub> pins and activates **ALE = 1** to indicate that P<sub>0</sub> has the addresses.



# PSEN(program store enable))

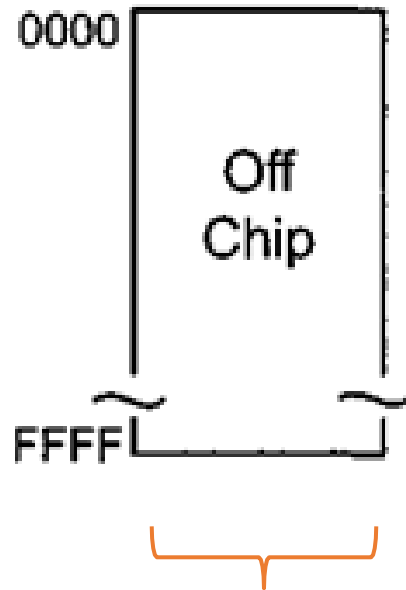
- **PSEN** is an output signal for the 8031/51 microcontroller and must be connected to the OE pin of a ROM containing the program code.
- To access external ROM containing program code, the 8031/51 uses the **PSEN signal**.
- When the **EA pin is connected to GND**, the 8031/51 fetches opcode from external ROM by using PSEN.

**STEP 3:** Notice in Figure the connection of the PSEN pin to the **OE pin** of ROM

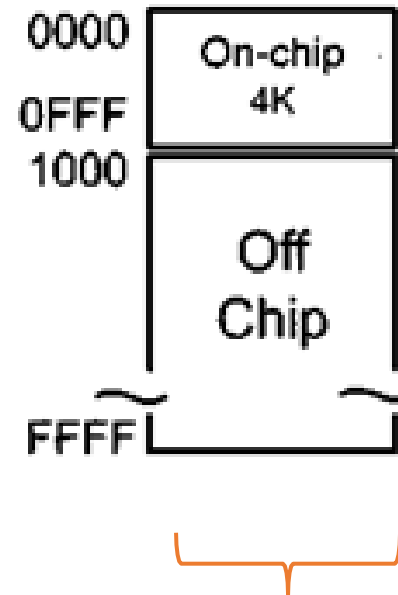


# On-chip and off-chip code ROM

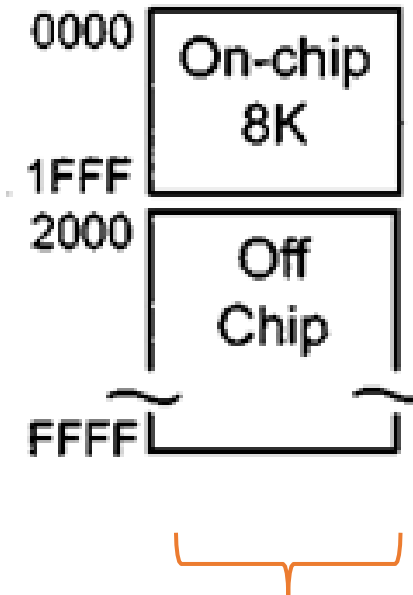
8031/51  
EA = GND



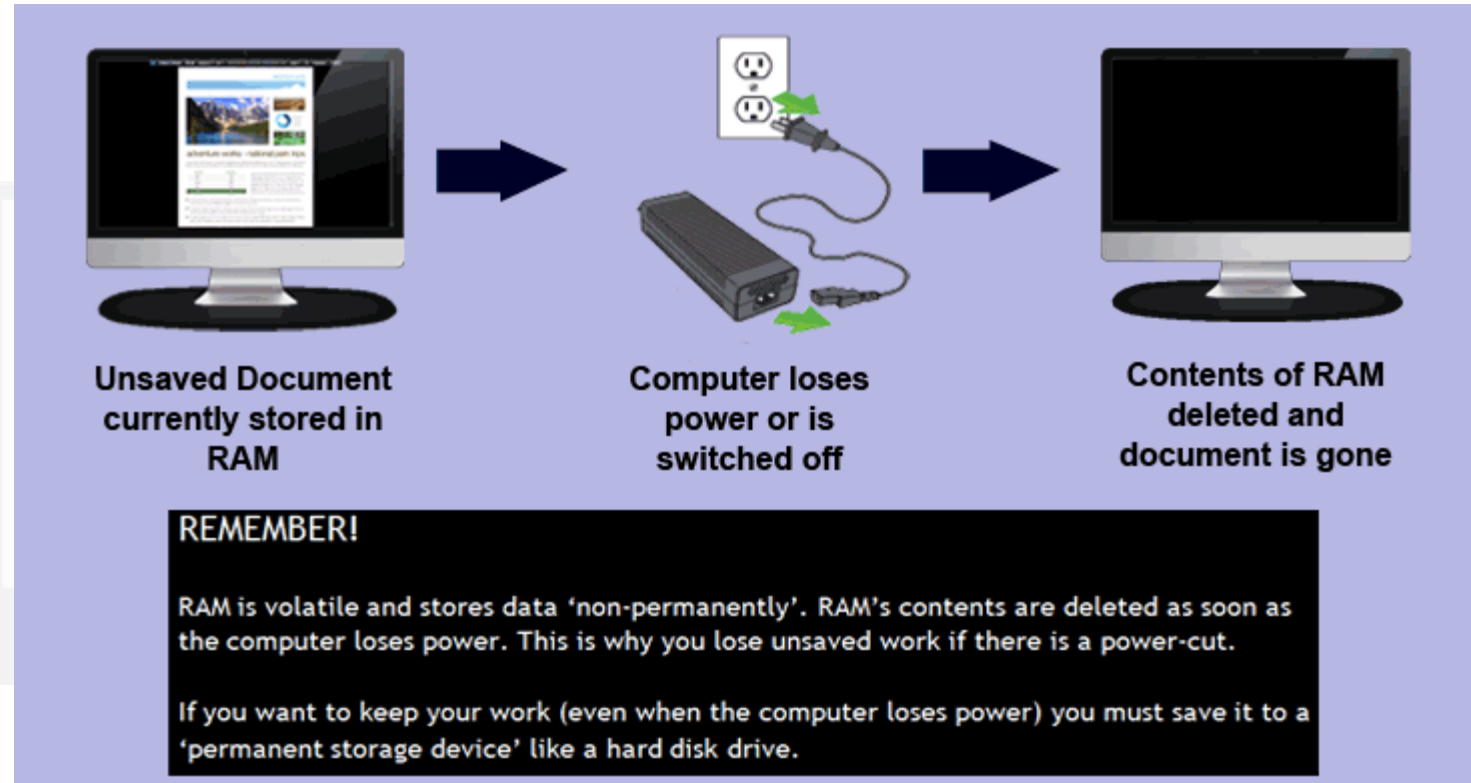
8051  
EA =  $V_{CC}$



8052  
EA =  $V_{CC}$







## Additional Info

## Examples of ROM



**BIOS Chip with instructions  
which boot computer and  
load operating system.**



**DVD / CD ROMs  
containing pre-loaded  
music and movie content**



**ROM Chip in printers  
which store pre-loaded  
font styles.**

### **Remember!**

**ROM is read only and cannot be altered by the user. This prevents accidental deletion of important commands which devices need to operate correctly.**

**READ  
ONLY  
MEMORY**

## Additional Info

# Addressing Modes

- 1. Immediate addressing.
- 2. Register addressing.
- 3. Direct addressing.
- 4. Indirect addressing.
- 5. Relative addressing.
- 6. Absolute addressing.
- 7. Long addressing.
- 8. Indexed addressing.
- 9. Bit inherent addressing.
- 10. Bit direct addressing.

# QUIZ

**1. How many pins are there in 8051 microcontroller**

- a. 22
- b. 40
- c. 21
- d. 8

**2. Pin No 9 is:**

QUIZ

- a. ALE
- b. RST
- c. Vcc
- d. TxD

3. **XTAL2, XTAL1** are:

## QUIZ

- a. Power sources
- b. Idle and Power down
- c. Crystal oscillators
- d. SCON & TCON

## 2. EA=GND indicates

### QUIZ

- a. On chip ROM
- b. Off chip ROM
- c. Both
- d. None of the above

# QUIZ

3. To access external ROM containing program code, the 8031/51 uses

- a. OLE
- b. EA
- c. PSEN
- d. ALE



# Reference

[http://ece-research.unm.edu/jimp/310/slides/8086\\_memory2.html](http://ece-research.unm.edu/jimp/310/slides/8086_memory2.html)



# 1. Immediate addressing mode

In this addressing mode the data is provided as a part of instruction itself.

In other words, data immediately follows the instruction.

Eg. MOV A, #30H ;  $A \rightarrow 30H$

ADD A, #83H ; Symbol # indicates the data

//is immediate  $A = 30H + 83H$

00110000

10000011

## 2. Register addressing mode

- In this addressing mode the register will hold the data.
- One of the eight general registers (R0 to R7) can be used and specified as the operand.
- Eg. MOV R0,#45H ; 45H→R0=45H
- MOV A,R0 ; R0=45H→A=45H
- MOV R6, #01H
- ADD A,R6 ; // A+R6 45H+01H=46H
- R0 – R7 will be selected from the current selection of register bank.  
The default register bank will be **bank 0**

## Internal RAM organization

R7	1F	BANK 3
R6	1E	
R5	1D	
R4	1C	
R3	1B	
R2	1A	
R1	19	
R0	18	BANK 2
R7	17	
R6	16	
R5	15	
R4	14	
R3	13	
R2	12	
R1	11	BANK 1
R0	10	
R7	0F	
R6	0E	
R5	0D	
R4	0C	
R3	0B	
R2	0A	BANK 0
R1	09	
R0	08	
R7	07	
R6	06	
R5	05	
R4	04	
R3	03	
R2	02	
R1	01	
R0	00	

## Working Registers

2F	7F						78
2E	77						70
2D	6F						68
2C	67						60
2B	5F						58
2A	57						50
29	4F						48
28	47						40
27	3F						38
26	37						30
25	2F						28
24	27						20
23	1F						18
22	17						10
21	0F						08
20	07						00

## Bit addressable memory

7F
7E
.
.
.
.
.
.
.
32
31
30

## General purpose memory

### 3. Direct addressing mode

- There are two ways to access the internal memory.
- Using direct address and indirect address.
- **Advantage:**
- Using direct addressing mode we can not only address the internal memory but SFRs also.

### 3. Direct addressing mode

- Move the contents from address 50H to 55h
- MOV R0,50H ; Initialised address
- MOV A, @R0 ; Move the contents @R0--→A=01H
- MOV R1,55H
- MOV @R1,A



## 4. Indirect addressing mode

- The **indirect addressing mode** uses a register to hold the actual address that will be used in data movement.
- Registers R<sub>0</sub> and R<sub>1</sub> and DPTR are the only registers that can be used as an alternate to **data pointers**.
- Indirect addressing cannot be used to refer to **SFR registers**.
- Both **R0** and **R1** can hold **8 bit address** and **DPTR** can hold **16 bit address**.
- Eg. MOV A,@R0      @R0--→A      @FFH--→05H  
      ADD A,@R1 ;      ADDD A+@R1      A= 05 + 02=07H

MOV DPTR,#5100H

MOV TMOD,@R0

MOVX A,@DPTR    ; @5100H-→A

## 5. Indexed addressing mode

- In indexed addressing, either the **program counter (PC)**, or the **data pointer (DTPR)**—is used to hold the base address, and the A is used to **hold the offset address**.
- Adding the value of the base address to the value of the offset address forms the effective address.
- Indexed addressing is used with **JMP or MOVC instructions**.



## 5. Indexed addressing mode

- Look up tables are easily implemented with the help of index addressing.
- Eg. `MOVC A, @A+DPTR` // copies the contents of memory location pointed by the sum of the accumulator **A** and the **DPTR** into accumulator A.
- `MOVC A, @A+PC` // copies the contents of memory location pointed by the sum of the accumulator A and the program counter into accumulator A

## 6. Relative Addressing mode

- Relative addressing is used only with **conditional jump instructions**.
- The relative address, (offset), is an **8 bit signed number**, which is automatically added to the PC to make the address of the next instruction.
- The 8 bit signed offset value gives an address range of **+127 to —128 locations**.

## 6. Relative Addressing mode

- The jump destination is usually specified using a label and the assembler calculates the jump offset accordingly.
- The advantage of relative addressing is that the **program code is easy to relocate and the address is relative to position in the memory.**
- Eg. SJMP LOOP1  
JC BACK

## 7. Absolute addressing mode

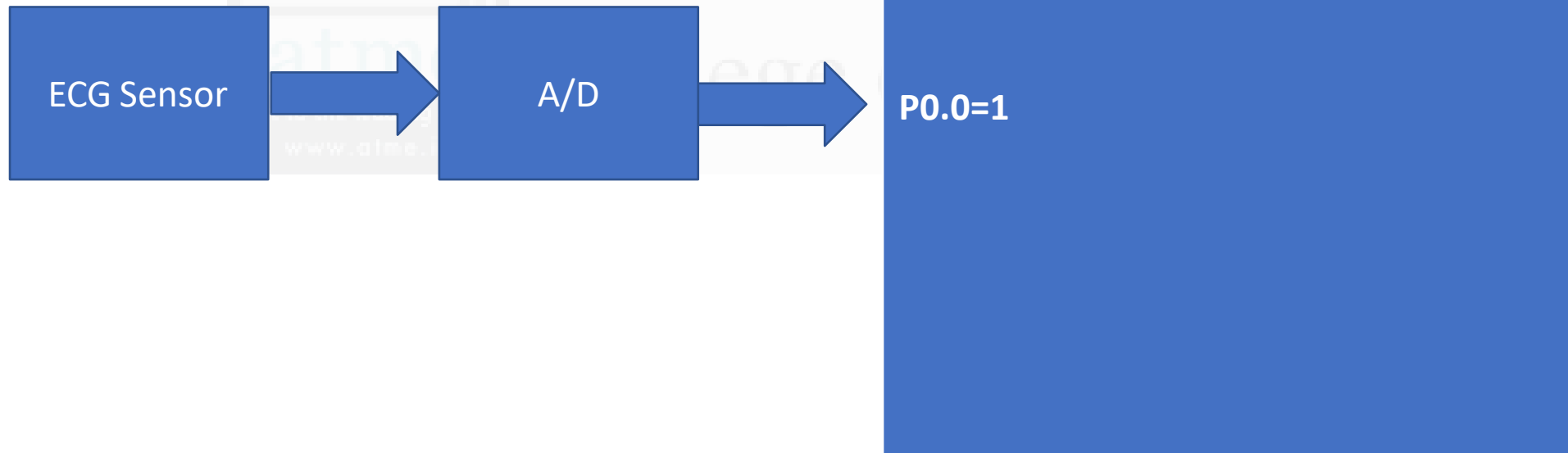
- Absolute addressing is used only by the **AJMP (Absolute Jump) and ACALL (Absolute Call) instructions.**
- These are **2 bytes instructions.**
- The absolute addressing mode specifies the lowest **11 bit** of the memory address as part of the instruction.
- The upper 5 bit of the destination address are the upper **5 bit** of the current program counter.
- Hence, absolute addressing allows branching only within the current **2 Kbyte page of the program memory.**
- **Eg. AJMP LOOP1**  
**ACALL LOOP2**

## 8. Long Addressing Mode

- The long addressing mode is used with the instructions **LJMP and LCALL**.
- These are **3 byte instructions**.
- The address specifies a full 16 bit destination address so that a jump or a call can be made to a location within a **64 Kbyte code memory space**.
- Eg. LJMP FINISH  
LCALL DELAY

## 9. Bit Inherent Addressing mode

- In this addressing, the address of the flag which contains the operand, is implied in the opcode of the instruction.
- Eg. CLR P0.0 ; P0.0=0
- SETB P0.0 ; P0.0=1



## 10. Bit Direct Addressing Mode

- In this addressing mode the direct address of the bit is specified in the instruction.
- The RAM space **20H to 2FH** and most of the special function registers are bit addressable. **Bit address values are between 00H to 7FH.**
- Eg. CLR 87h ; Clears the bit 7 of 20h RAM  
space PCON=1 IDL PD
- SETB 07H ; Sets the bit 7 of 20H RAM space.



A T M E

College of Engineering

# Programs



**Write a program to store data FFH into RAM memory locations 50H to 58H using direct addressing mode**

## Solution

ORG 0000H; Set program counter 0000H

MOV A, #0FFH; Load FFH into A

MOV 50H, A ; Store contents of A in location 50H

MOV 51H, A ; Store contents of A in location 51H

MOV 52H, A ; Store contents of A in location 52H

MOV 53H, A ; Store contents of A in location 53H

MOV 54H, A ; Store contents of A in location 54H

MOV 55H, A ; Store contents of A in location 55H

MOV 56H, A ; Store contents of A in location 56H

MOV 57H, A ; Store contents of A in location 57H

MOV 58H, A ; Store contents of A in location 58H



# Programs

- Write a program to store data FFH into RAM memory locations 50H to 58H using indirect addressing mode.

## Solution

ORG 0000H; Set program counter 0000H

MOV A, #0FFH ; Load FFH into A

MOV R0, #50H ; Load pointer, R0-50H

MOV R5, #08H ; Load counter, R5-08H

**Start:** MOV @R0, A ; Copy contents of A to RAM    A=FFH    ---→ @R0=**@52H=FFH**  
;pointed by R0

INC R0 ; Increment pointer **R0=52H**

DJNZ R5, **start** ; Repeat until R5 is zero    **R5=06**

END

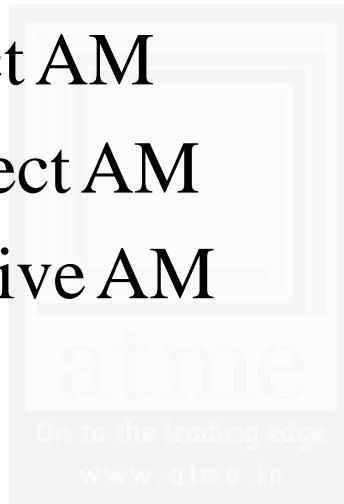
# QUIZ

1. Which addressing mode cannot be used to refer to **SFR registers**.
- a. Direct AM
  - b. Relative AM
  - c. Indirect AM
  - d. Indexed AM



## 2. MOVCA, @A+DPTR

- a. Indexed AM
- b. Direct AM
- c. Indirect AM
- d. Relative AM



### 3. What is the O/P in the Accumulator at the end of the execution:

MOV A,#45H

MOV B, A

MOV R0, 50H

MOV R1, 52H

MOV A, #0AH

# QUIZ

**4. What is the O/P at the end of the execution:**

MOV A,#45H

MOV B, A

MOV R<sub>0</sub>, #50H

MOV R<sub>1</sub>, #52H

MOV R<sub>1</sub>, R<sub>0</sub>



A T M E

College of Engineering

# Programs



**Write a program to store data FFH into RAM memory locations 50H to 58H using direct addressing mode**

## Solution

ORG 0000H; Set program counter 0000H

MOV A, #FFH; Load FFH into A

MOV 50H, A ; Store contents of A in location 50H

MOV 51H, A ; Store contents of A in location 51H

MOV 52H, A ; Store contents of A in location 52H

MOV 53H, A ; Store contents of A in location 53H

MOV 54H, A ; Store contents of A in location 54H

MOV 55H, A ; Store contents of A in location 55H

MOV 56H, A ; Store contents of A in location 56H

MOV 57H, A ; Store contents of A in location 57H

MOV 58H, A ; Store contents of A in location 58H

- Write a program to store data FFH into RAM memory locations 50H to 58H using indirect addressing mode.

## Solution

ORG 0000H; Set program counter 0000H

MOV A, #FFH ; Load FFH into A

MOV RO, #50H ; Load pointer, R0-50H

MOV R5, #08H ; Load counter, R5-08H

**Start:** MOV @RO, A ; Copy contents of A to RAM    A=FFH    ---→ @R0=**@52H=FFH**  
;pointed by R0

INC RO ; Increment pointer **R0=52H**

DJNZ R5, **start** ; Repeat until R5 is zero    **R5=06**

END

# More Information





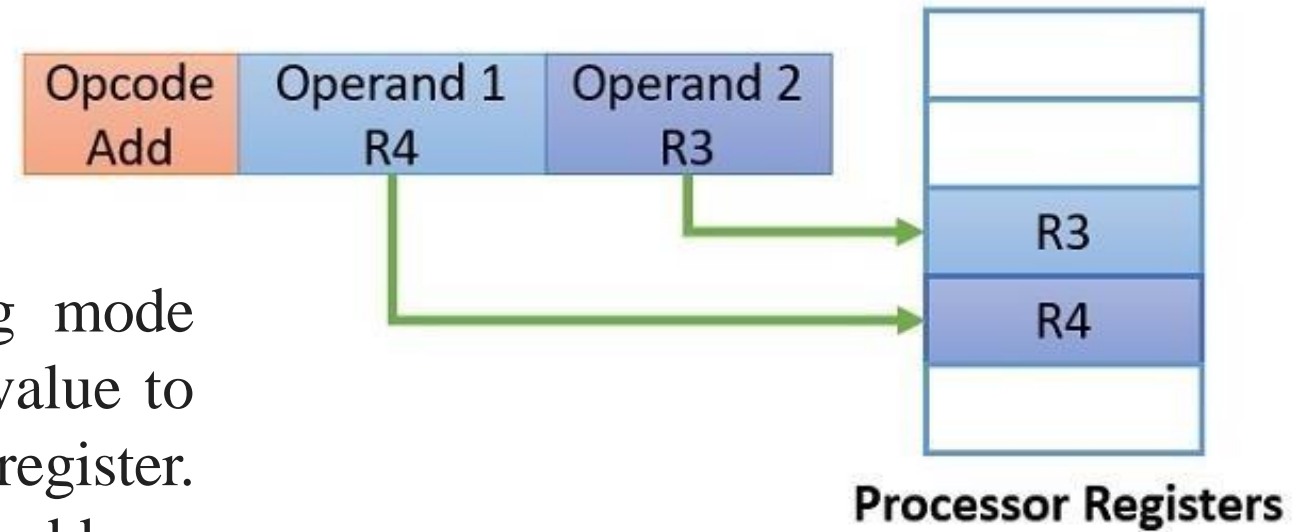
**Advantage:** In the register addressing mode there are no memory references as the value to be operated is present in the register.

**Disadvantage:** Registers have limited address space. So, it has a limit on the size of value that can be stored.

MOV R4, #05H

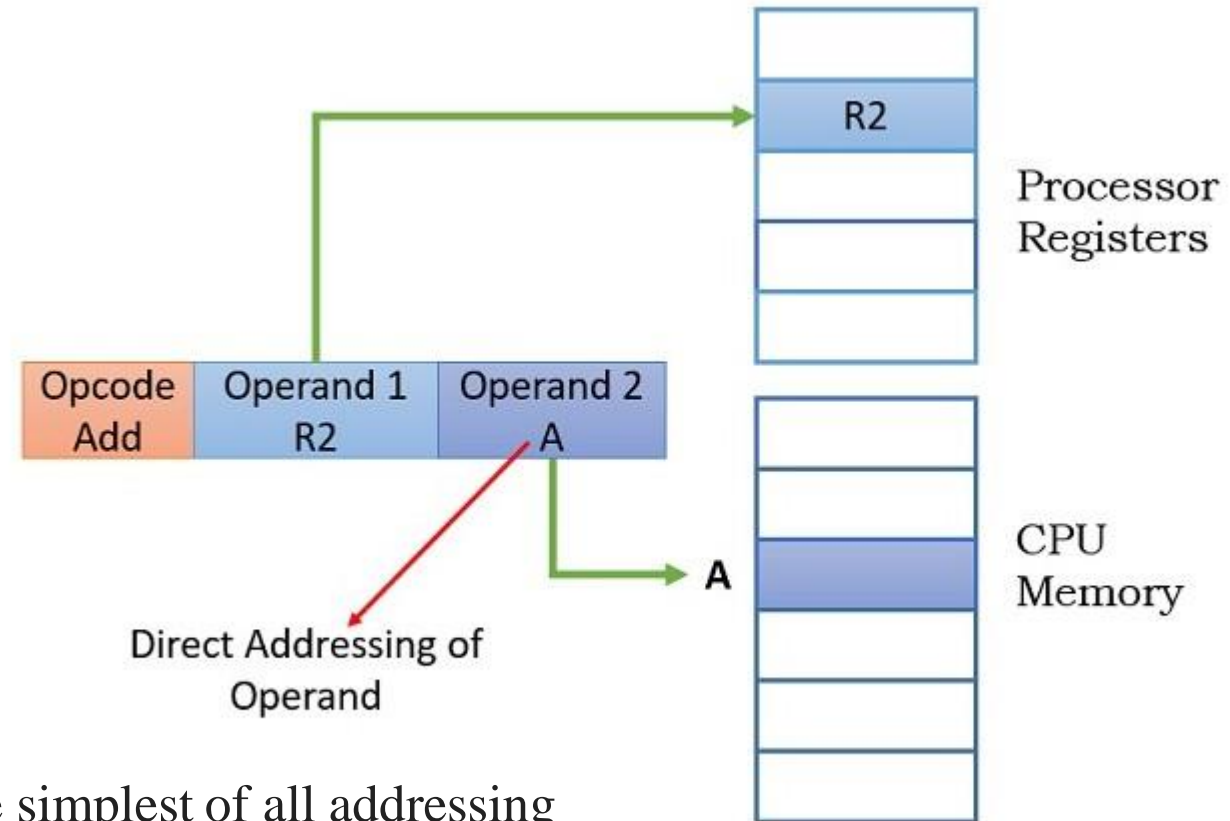
MOV R3, #05H

**ADD** R4, R3 ; 05+05 → 0AH



# Direct Addressing Mode

```
MOV R2, 50H  
MOV A, #05H  
ADD R2, A
```



**Advantage:** Direct addressing mode is the simplest of all addressing mode.

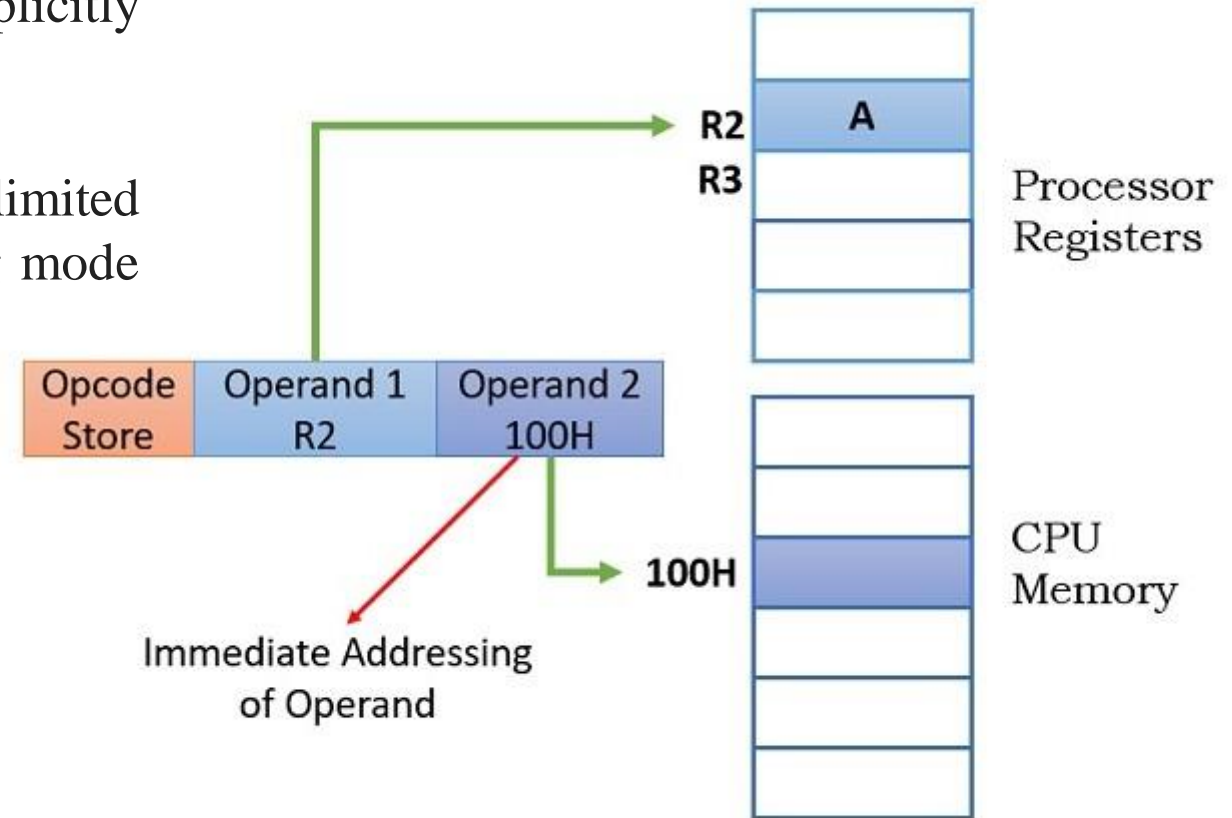
**Disadvantage:** Direct addressing mode provides a limited address space.

## Immediate Addressing Mode

**Advantage:** In the immediate addressing mode the memory reference is not required as the value is explicitly present in the instruction

**Disadvantage:** The instruction format provides a limited size for the operand. So, the immediate addressing mode has limited space for immediate value.

```
MOV R2, #05H  
ADD R2, #100H
```



# Programs

## 1.Put the numbers 34H in registers R5, R6 and R7

### Solution:

**Method 1:** Using immediate number and register addressing

MOV R5, # 34H ; MOVE THE NUMBER 34H-→ Source

//Register R5

MOV R6, #34H

MOV R7,#34H

- **Method 2:**

**Since the number is same move the number to A and move to each register**

MOV A, #34H

MOV R5, A

MOV R6, A

MOV R7, A

- **Method 3:**

**Copy one direct address to another**

MOV R5, #34H ; 34H--→R5=34H

MOV 06H, 05H ; R5=34H-→R6=34H

MOV 07H,06H ;R6=34H--→R7=34H

## 2. Put the numbers 4DH in RAM locations 30H to 34H

**4DH--→01001101**

- **Solution:**
- **Method 1:**
- **Use immediate addressing number to a direct address**

MOV 30H, #4DH ;

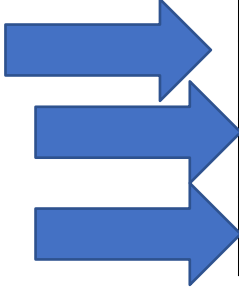
MOV 31H, #4DH

MOV 32H, #4DH

MOV 33H, #4DH

MOV 34H, #4DH

## • Method 2: PUSH Command



33H	
32H	4DH
31H	4DH
30H	4DH

- MOV 30H,#4DH
- MOV 81H,#30H :Set the SP to 30H
- PUSH 30H ; Push contents of 30H → to address 31H
- PUSH 30H ; Push contents of 30H → to address 32H
- PUSH 30H ; Push contents of 30H → to address 33H
- PUSH 30H ; Push contents of 30H → to address 34H



- **Method 3:**  
**Indirect addressing mode ( @ / Loop)**

**MOV R0,30H ; R0=30H (Address) pointer**

**MOV R1,#05H ; R1= 05H( Counter)**

**BACK:** **MOV R0,#4DH ; 4DH---→R0=31H(Address)=4DH(Value)**

**INC R0 ; R0= 34H Increment**

**DJNZ R1, BACK ; Decrement & Jump if not zero R1=00H**

**END**

<b>34H</b>	<b>4DH</b>	
33H	4DH	
32H	4DH	
31H	4DH	
30H	4DH	

3. What is the Output of the following instruction

- MOV A, #41H ; 41H  $\rightarrow$  A=41H
- ADD A, #55H ; 55H+A=41H= 96H
- MOV R0, A ; A=96H  $\rightarrow$  R0=96H
- MOV R1, #FFH ; R1=FFH
- MOV R0, R1 ; Illegal Operand
- **Output? Illegal Operand, Cannot Move contents from register to register**



*Thank You*

A T M E  
College of Engineering