Applications of microcontroller 1

8051 Microcontroller

Introduction

- A microcontroller is **a complete computer system** built on a single chip
- It contains all components like
  - CPU
  - RAM
  - ROM
  - Serial Port
  - Parallel Port
  - Interrupt logic
  - Timer

  A microcontroller saves **cost**, saves **power consumption** and makes **circ uit compact**

Applications

Home

- Appliances, intercom, telephones, security systems, garage door openers, answering machines, fax machines, home computers, TVs, cable TV tuner, VCR, camcorder, remote controls, video games,cellular phones, musical instruments, sewing machines,lighting control, paging, camera, pinball machines, toys, exercise equipment

Office

- Telephones, computers, security systems, fax machines, microwave, copier, laser printer, color printer, paging

Applications

Auto

- Trip computer, engine control, air bag, ABS, instrumentation, security system, transmission control, entertainment, climate control, cellular phone, keyless entry

Overview of the 8051

- 8051 is an **8-bit Microcontroller** having
  - On chip ROM = 4KB (Program Memory)
  - On chip RAM = 128 Bytes (Data Memory)
  - Four 8-bit bidirectional I/O ports
  - Serial port
  - Two 16-bit up counters (Timers)
  - 40 pins in a dual in-line package (DIP) layout
  - Supports interrupts with two level priority
  - Power Saving Modes
- It is used in appliances such as washing machine, microwaves, mobile phones, MP3 players, etc

Overview of the 8051

- Made by Intel in 1981
- An 8-bit, single-chip microcontroller optimized for control applications
- 128 bytes RAM, 4096 bytes (4KB) ROM, 2 timers, 1 serial port, 4 I/O ports
- 40 pins in a dual in-line package (DIP) layout

General Physical Features

- 4KB ROM
- 128 bytes internal RAM
  - 4 register banks of 8 bytes each (R0-R7)
  - 16 bytes of bit-addressable area
  - 80 bytes of general purpose memory
- Four 8-bit I/O ports (P0-P3)
- Two 16-bit timers (Timer0 & Timer1)
- One serial receiver-transmitter interface
- Five interrupt sources (2 external & 3 internal)
- One oscillator (generates clock signal)

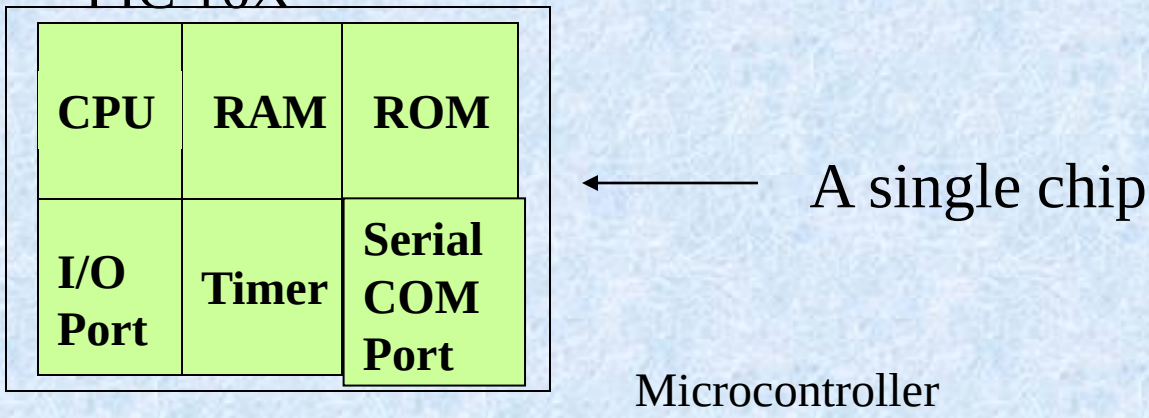
```
4 x 8 = 32
      16
      80
------------
     128
```

General Operational Features

- Memory of 8051 can be increased externally:
  - Increase memory space for codes (programs) by 64K
  - Increase memory space for data by 64K
- Boolean instructions work with 1 bit at a time
- Assume clock frequency = 12MHz, it takes about 4 μs (i.e. 4 x 10^{-6}s) to carry out a 8-bit multiplication instruction

Microcontroller :

- A smaller computer
- On-chip RAM, ROM, I/O ports...
- Example：Motorola’s 6811, Intel’s 8051, Zilog’s Z8 and PIC 16X



Microprocessor	Microcontroller
Just a processor, Memory and I/O components have to be connected externally	Microcontroller has an internal processor along with internal memory and I/O components
Circuit becomes large (as memory and I/O have to be connected externally)	**Circuit is small** (since memory and I/O components are present internally)
Cannot be used in Compact Systems	Can be used in Compact System
Cost of entire system **increases**	**Cost** of system **is low**
Since there are more external components, the total **power consumption is high**	Since external components are few total **power consumption is less** Hence ideal for appliances running on stored power
Most of initial Microprocessor **did not have Power Saving features**	Most microcontrollers **have Power Saving Mode** These help to reduce the power consumption even further

Microprocessor	Microcontroller
Since memory and I/O components are external, **each operation will be an external operation**, hence will be slower	Since components are internal, most **operations will be internal operations** hence will be faster
Less number of registers Hence more **operations are memory based**	Have more registers hence most **operations are register based** Hence programs are easier to write and faster
Based on **Von Neumann Model**, where programs and data are stored in the same memory module	Based on **Harvard Model**, where program memory and data memory are separate
Used mainly in PC	Used mainly in appliances such as washing machines, MP3 players, etc.

Register

Register are used to store information temporarily, while the information could be

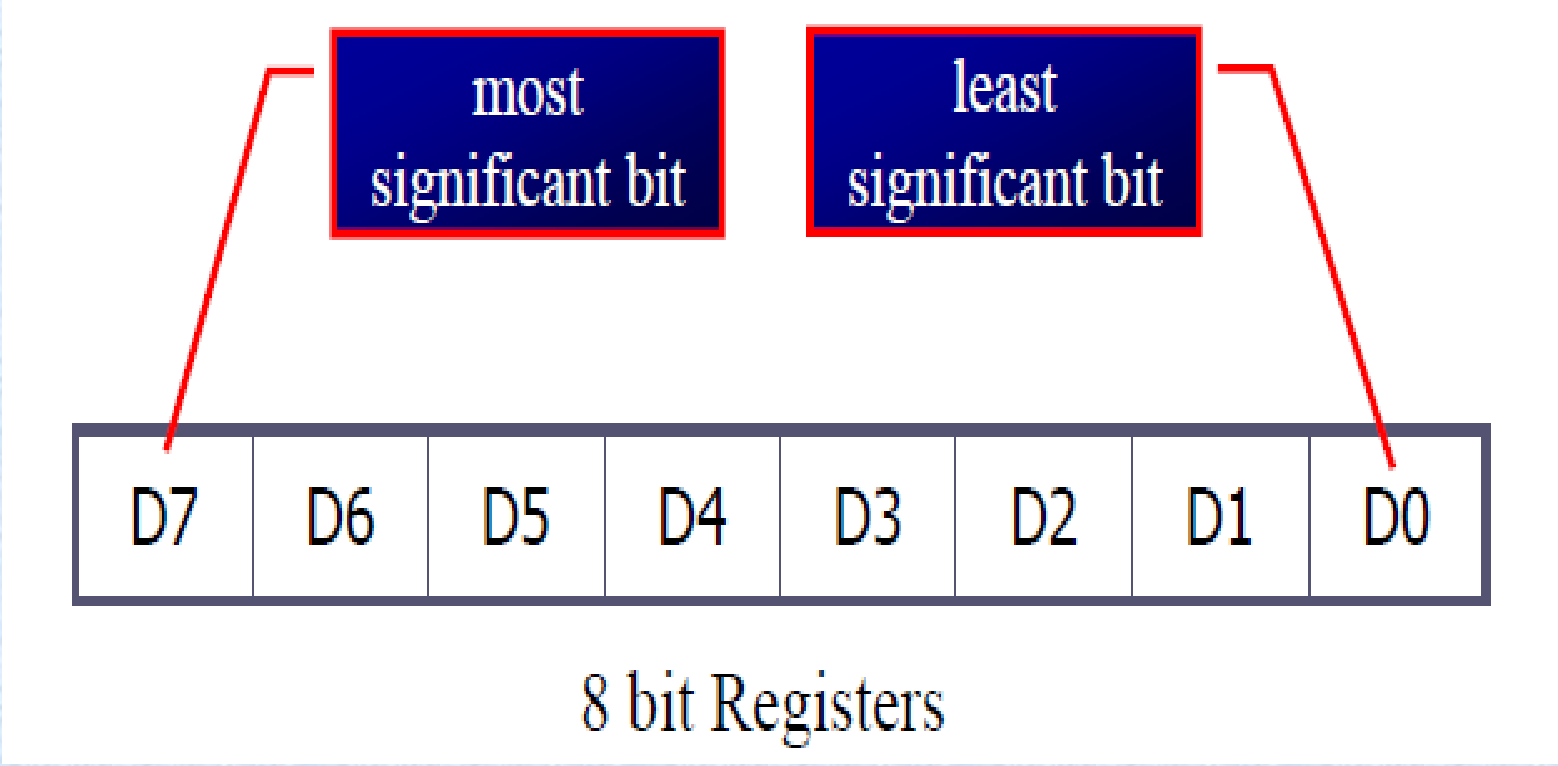
- a byte of data to be processed, or
- an address pointing to the data to be fetched

The vast majority of 8051 register are 8-bit registers

- There is only one data type, 8 bits

Register

- The 8 bits of a register are shown from MSB D7 to the LSB D0

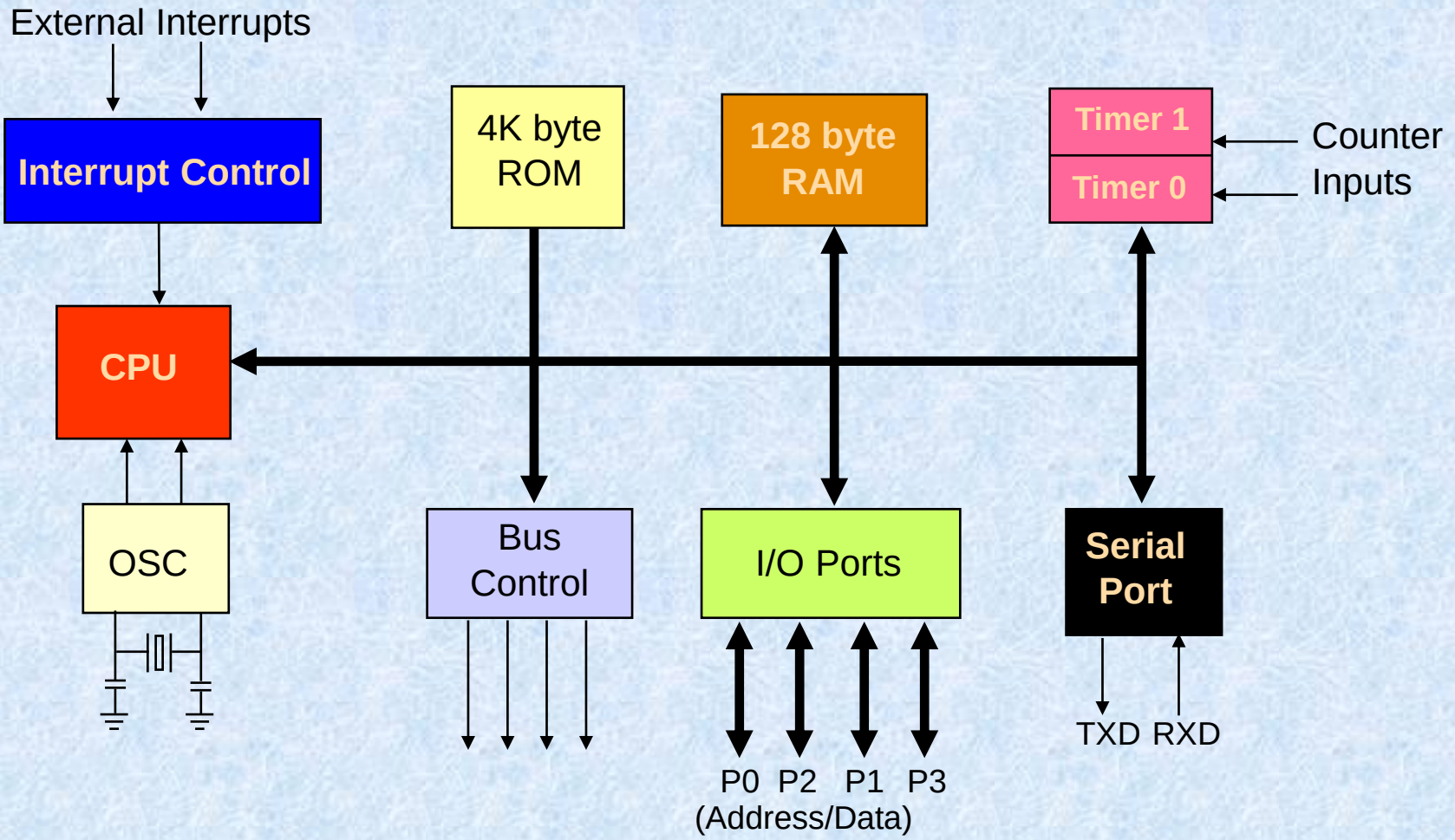


Comparison of 8051 Family Members

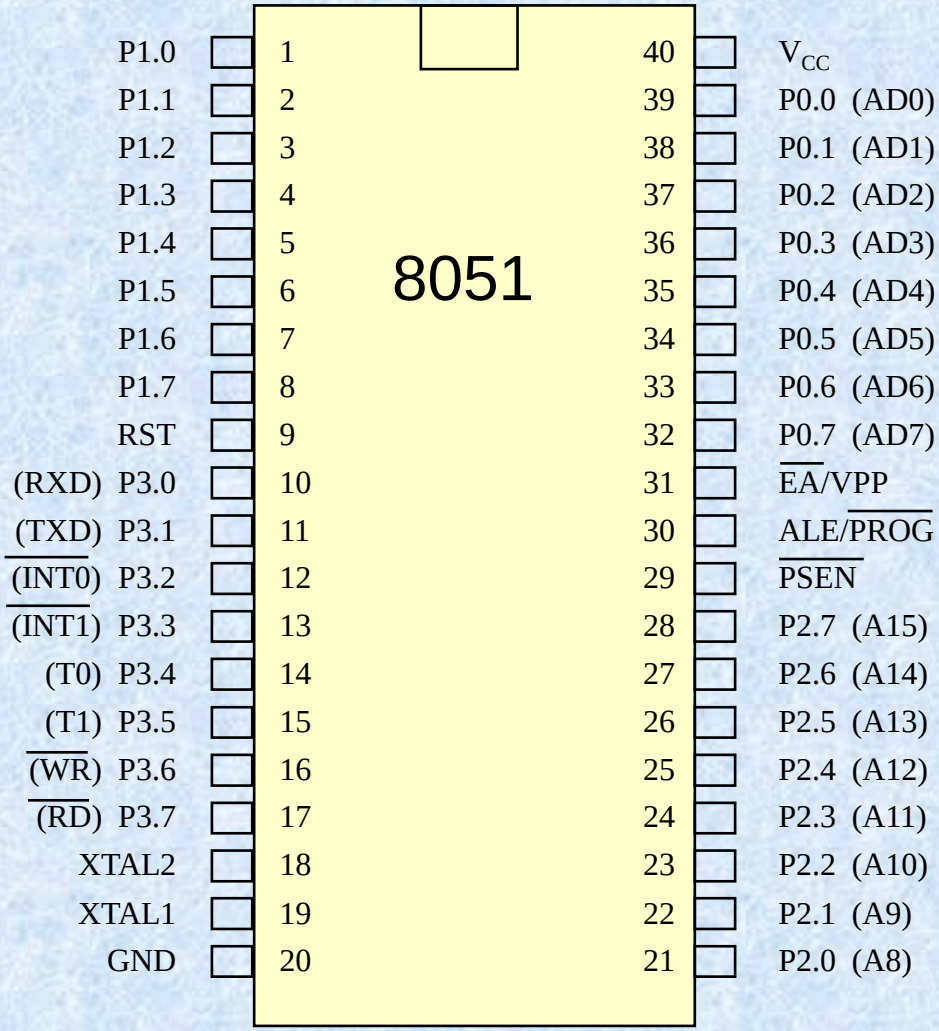
Feature	8051	8052	8031
ROM(on-chip program space in bytes)	4K	8K	0K
RAM	128	256	128
Timers	2	3	2
I/O pins	32	32	32
Serial port	1	1	1
Interrupt sources	6	8	6

Various 8051 Micro controllers

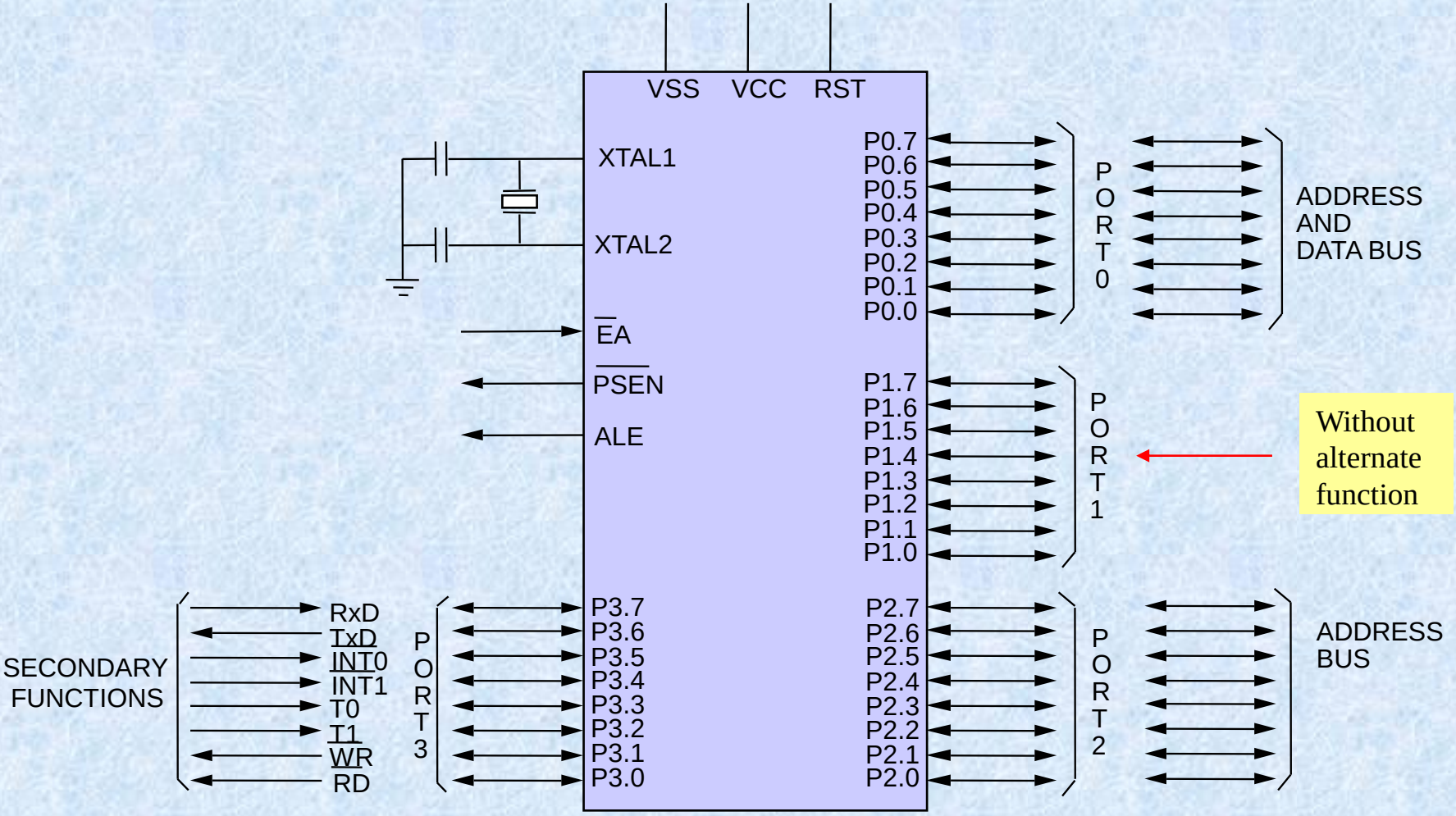
The 8051 Block Diagram



The 8051 Pin Assignments



The 8051 Logic Symbol



Hardware Description

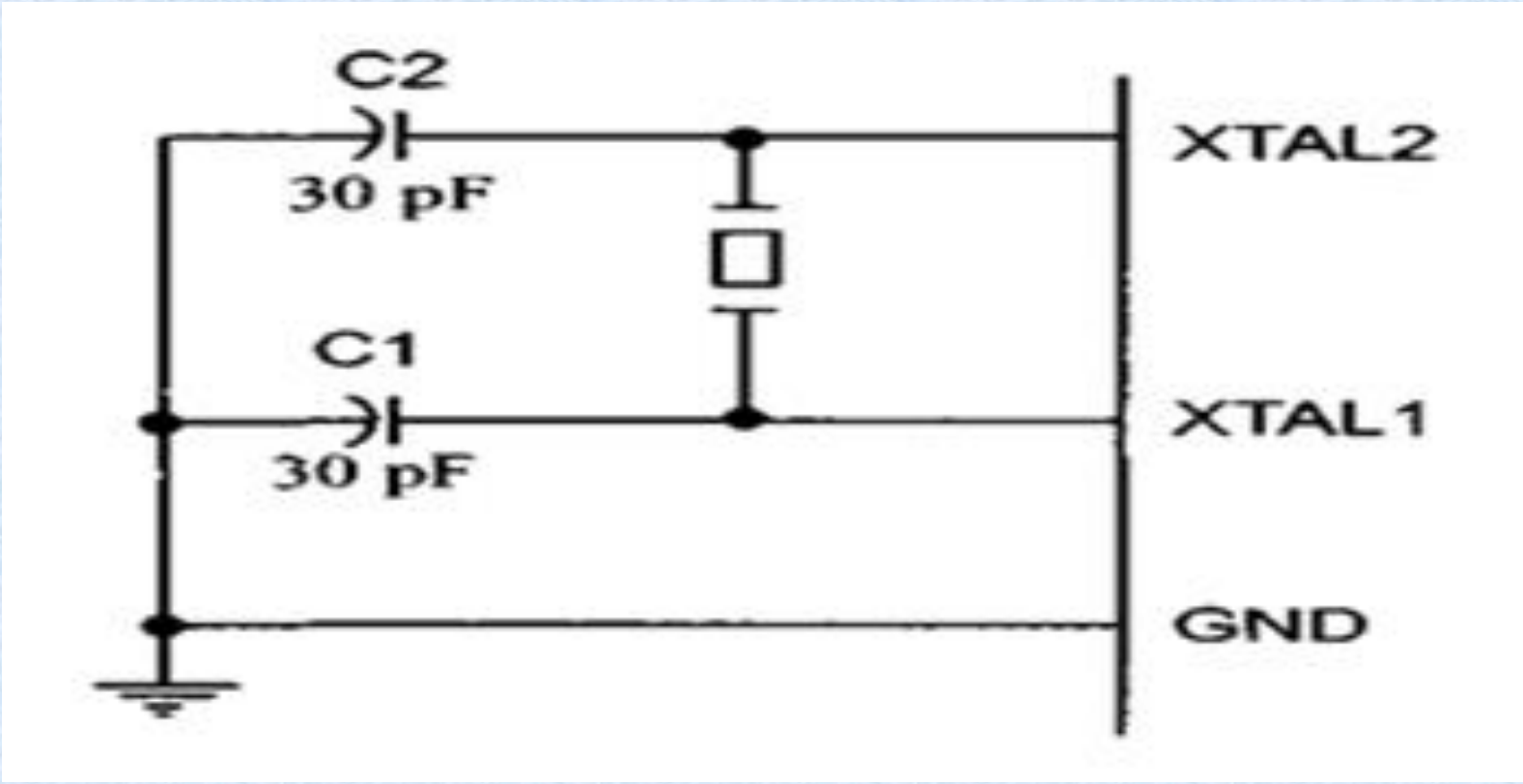
1. Oscillator circuit
2. Program counter (PC)
3. Data pointer (DPTR)
4. Accumulator (“A”) register
5. B register
6. Flags
7. Program status word (PSW)
8. Internal memory (ROM, RAM, additional memory)
9. Stack & stack pointer (SP)
10. Special function register (SFR)

Pin Description of 8051

- Vcc=pin 40 provides supply voltage to chip.
- Vtg src= +5 v
- GND :- Pin 20 is ground

XTAL 1 & XTAL 2:-

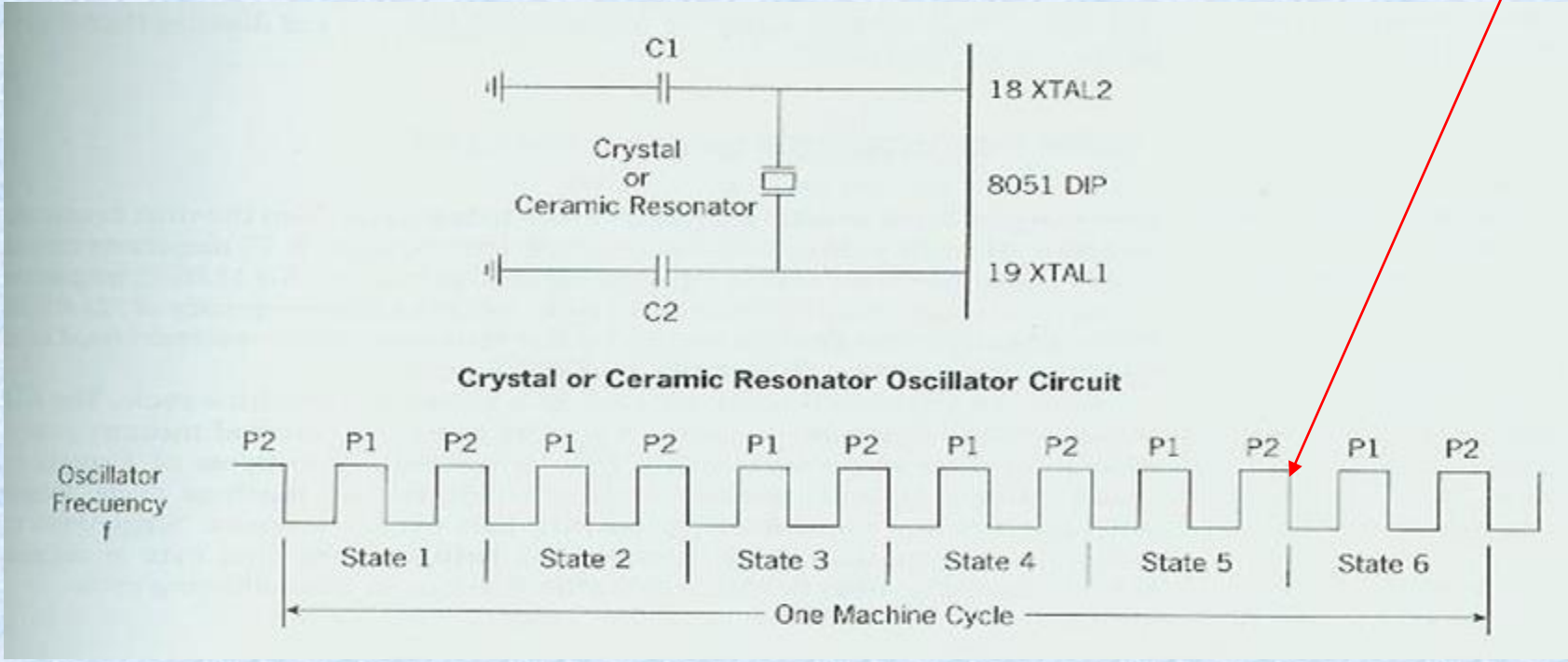
- 8051 has on-chip oscillator but requires external clock to run it.
- Quartz crystal oscillator connected to XTAL1(Pin 19) & XTAL2 (Pin 18).


C2
30 pF
C1
30 pF
XTAL2
XTAL1
GND

Oscillator Circuit

- The heart of the 8051
- Produces clock pulses
- Synchronize all 8051's internal operations

A single machine cycle consists of 12 crystal pulses !



Machine Cycle

- Machine cycle is the basic repetitive process that the CPU performs once it is powered on.
- A machine cycle consists of a fixed number of clock cycles (pulses).
- It is different for different kinds of CPU.
- The 8051 family needs 12 clock cycles for a machine cycle.
- The CPU takes one or more machine cycles to complete an instruction.
- More complex instructions require more number of machine cycles to complete the instruction.
- The number of machine cycles of the 8051 instructions are ranging from 1 to 4.

Example 1-1

Find the perid of the machine cycle for:

(a) XTAL = 11.0592 MHz

(b) XTAL = 16 MHz

(c) XTAL = 20 MHz

Solution:

(a) 11.0592 MHz / 12 = 921.6 kHz

Machine cycle = 1 / 921.6 kHz = 1.085 μs

(b) 16 MHz / 12 = 1.333 MHz

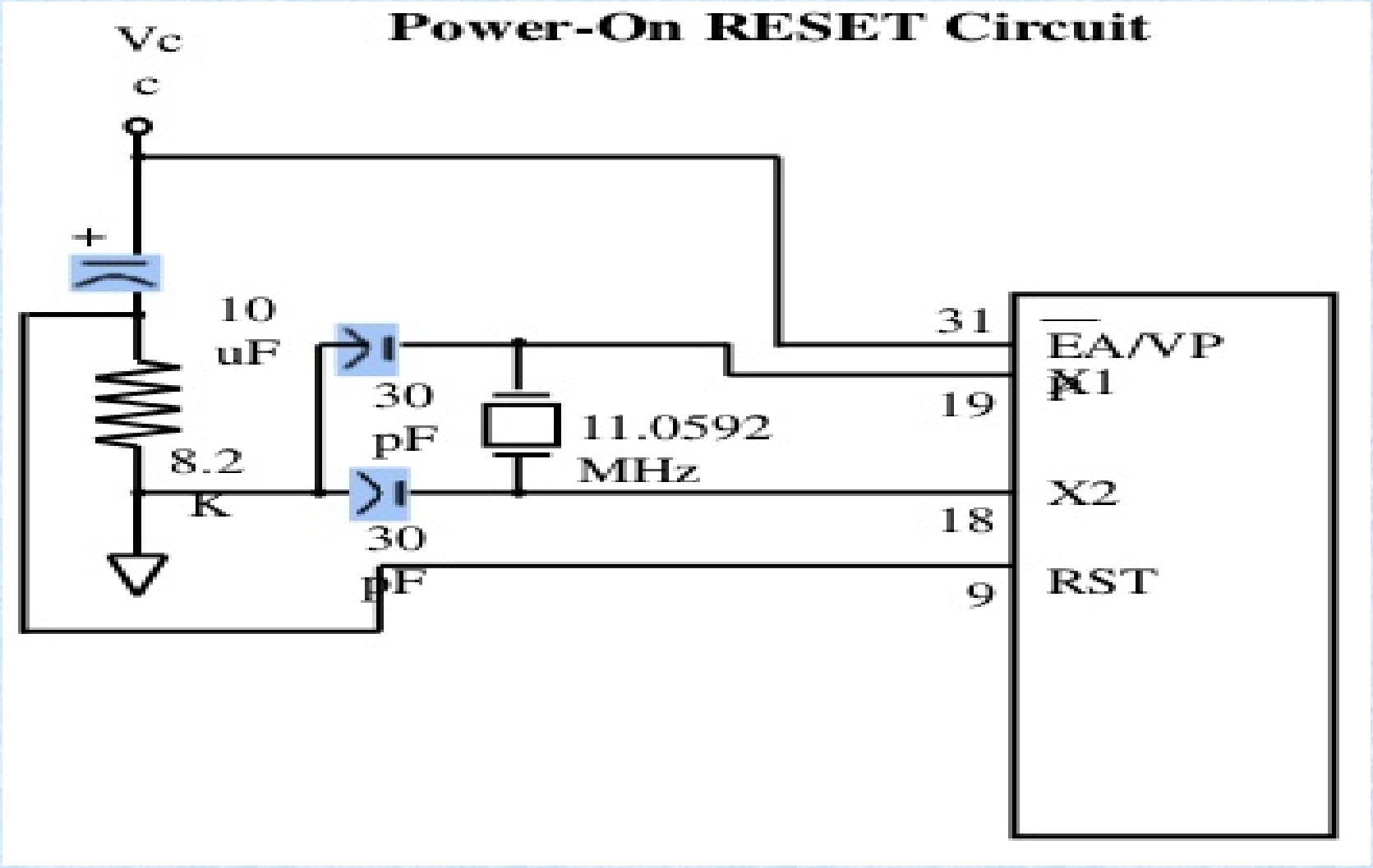
Machine cycle = 1 / 1.333 MHz = 0.75 μs

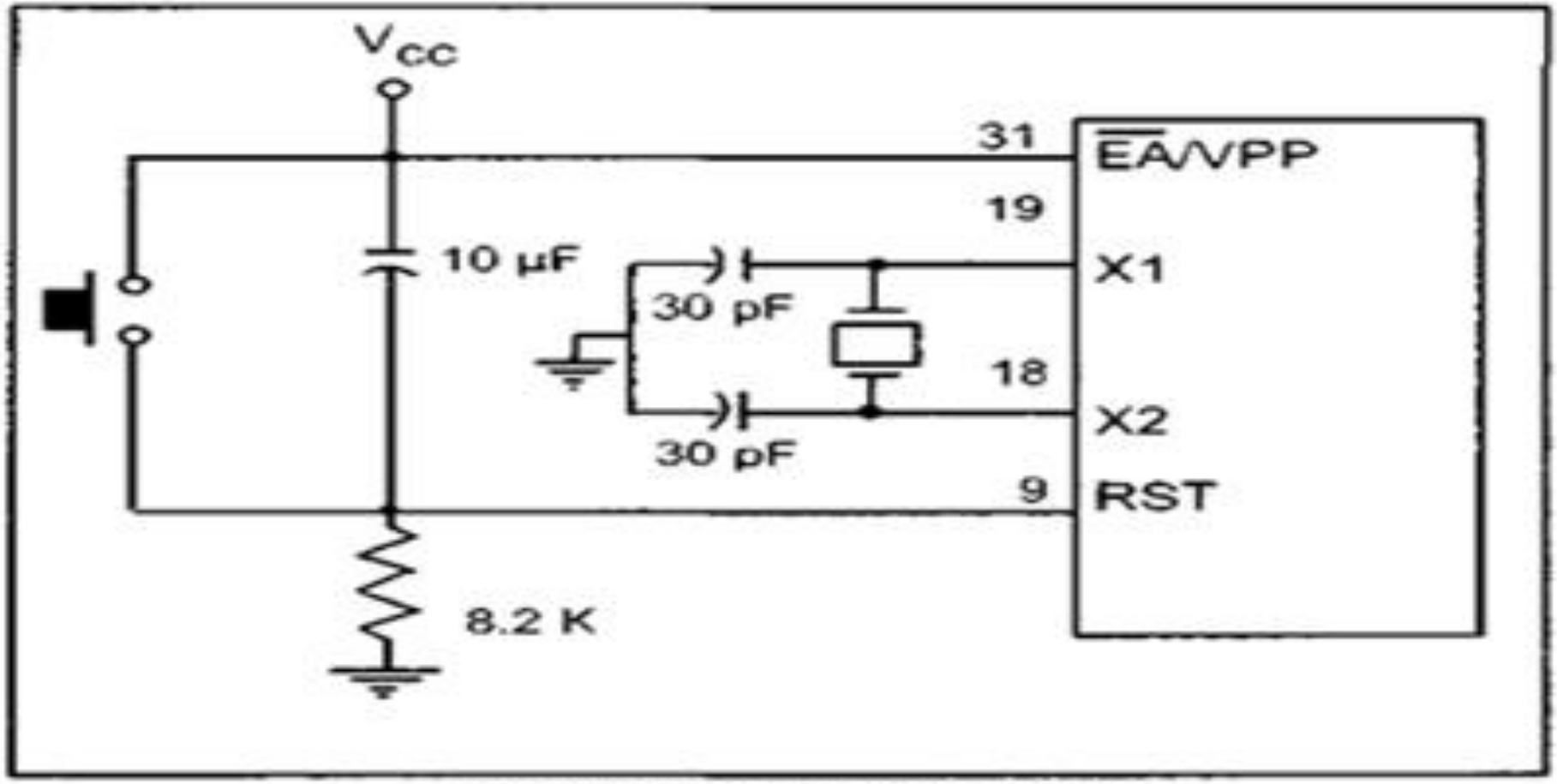
(c) 20 MHz / 12 = 1.667 MHz

Machine cycle = 1 / 1.667 MHz = 0.60 μs

RST (Pin 9 reset)

- It is i/p.Active high.
- Applying high pulse to this pin , microcontroller will re set & terminate all activities.
- Referred as Power on reset.
- RESET i/p effective,2 m/c cycle required.


Power-On RESET Circuit
Vc
c
+
10
uF
8.2
K
30
pF
30
pF
11.0592
MHz
31
EA/VP
19
X1
18
X2
9
RST



Power-On RESET with Momentary Switch

$\overline{EA}$（pin 31）：external access(i/p pin)

- There is no on-chip ROM in 8031 and 8032 .
- The $\overline{EA}$ pin is connected to GND to indicate the code is store d externally.
- $\overline{PSEN}$ & ALE are used for external ROM.
- For 8051, $\overline{EA}$ pin is connected to Vcc.
- “—” means active low.

$\overline{\text{PSEN}}$（pin 29）：program store enable

- This is an output pin and is connected to the OE pin of the ROM.

ALE（pin 30）：address latch enable

- It is an output pin and is active high.
- 8051 port 0 provides both address and data.
- The ALE pin is used for de-multiplexing the address and data by connecting to the G pin of the 74LS373 chip(3 state latch).

I/O port pins

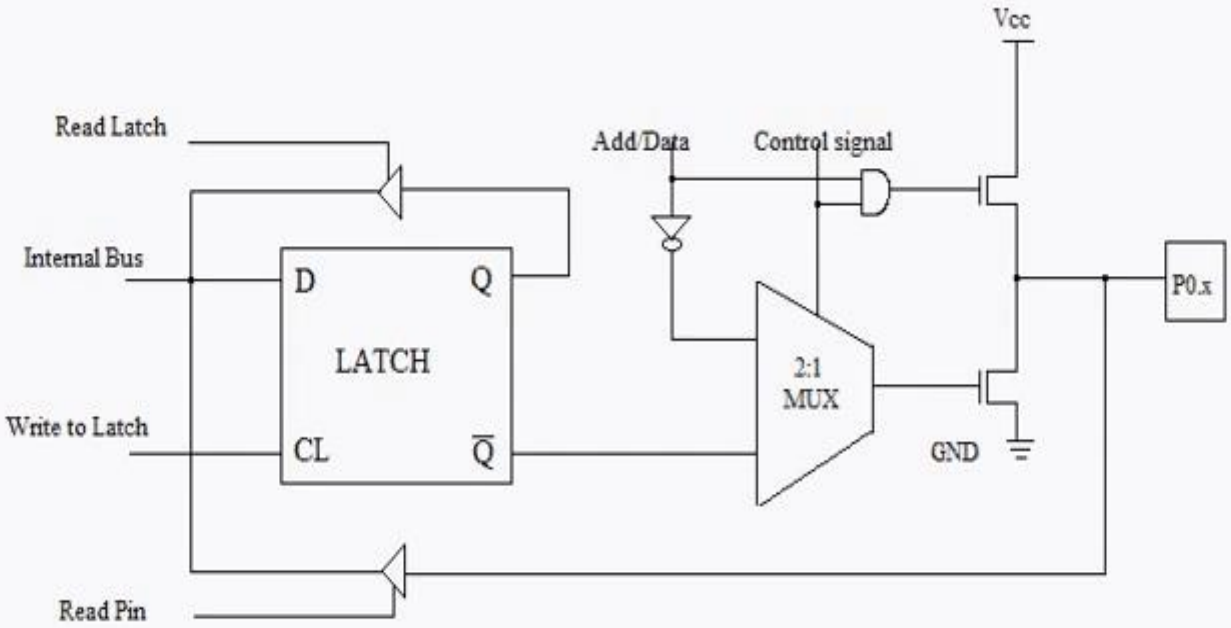
- The four ports P0, P1, P2, and P3.
- Each port uses 8 pins.
- All I/O pins are bi-directional.
- Port 0 （pins 32-39）：P0（P0.0～P0.7）
- Port 1（pins 1-8） ：P1（P1.0～P1.7）
- Port 2（pins 21-28）：P2（P2.0～P2.7）
- Port 3（pins 10-17）：P3（P3.0～P3.7）
- Named P0.X （X=0,1,...,7）, P1.X, P2.X, P3.X
- Ex：P0.0 is the bit 0（LSB）of P0
- Ex：P0.7 is the bit 7（MSB）of P0
- These 8 bits form a byte.

Port 0

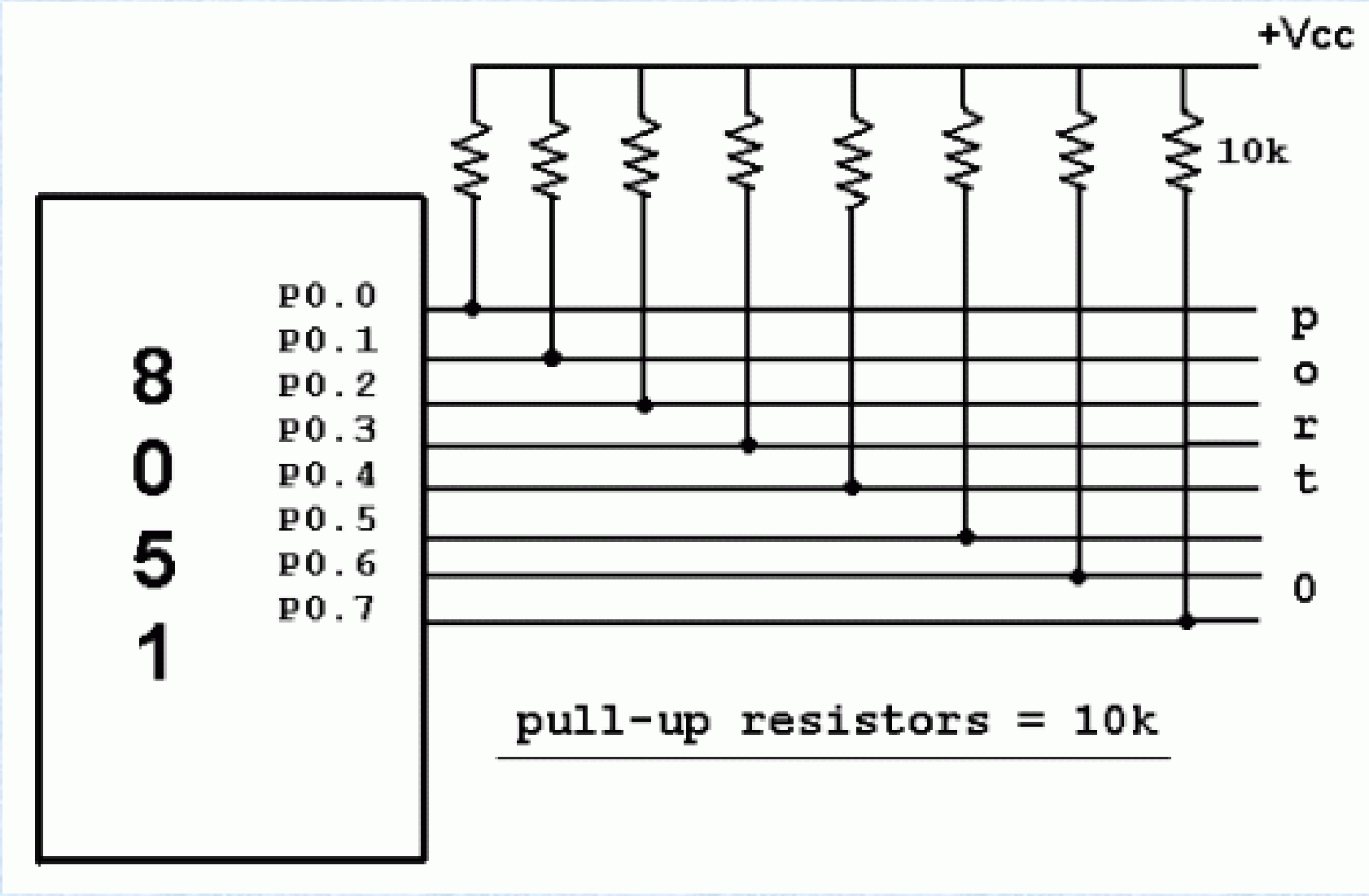
- Designed as AD0-AD7,allowing to use both address & d ata.
- When connecting 8051/31 to external memory , port 0 provides both address & data.
- When ALE =0,provides data D0-D7
- When ALE =1,provides address A0-A7
- In 8051 based system, there is no external memory co nnection, so pins of P0 connected externally to 10 k oh m pull-up resistor.

- 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.
- When **control is '1'**, the port is used for **address/data interfacing**.
- When the **control is '0'**, the port can be used as a normal **bidirectional I/O port.**
-

PORT 0


Vcc
Read Latch
Add/Data
Control signal
Internal Bus
D
Q
LATCH
P0.x
2:1
MUX
Write to Latch
CL
$\overline{Q}$
GND
Read Pin

- When control =0,it can used as bidirectional i/o port.
- Po=o/p,write 0 to pin ,Q=0 & Q =1,T1=1
- Po=i/p,write 1 to pin ,Q=1 & Q =0,T1=off which floats pin to high impedance state therefore ext pull up regis ters needed to provide logic1.
- T2 =off when P0 used as I/O port.


+Vcc
10k
8051
P0.0
P0.1
P0.2
P0.3
P0.4
P0.5
P0.6
P0.7
port 0
pull-up resistors = 10k

- Port 0 is open drain as compared to P1,P2 & P3.
- Open drain used for MOS chips in same way that open collector in TTL chips.
- When 0 written to port = it becomes o/p.
- When 1 written to port = it becomes i/p.

Port 0 as addr/data bus

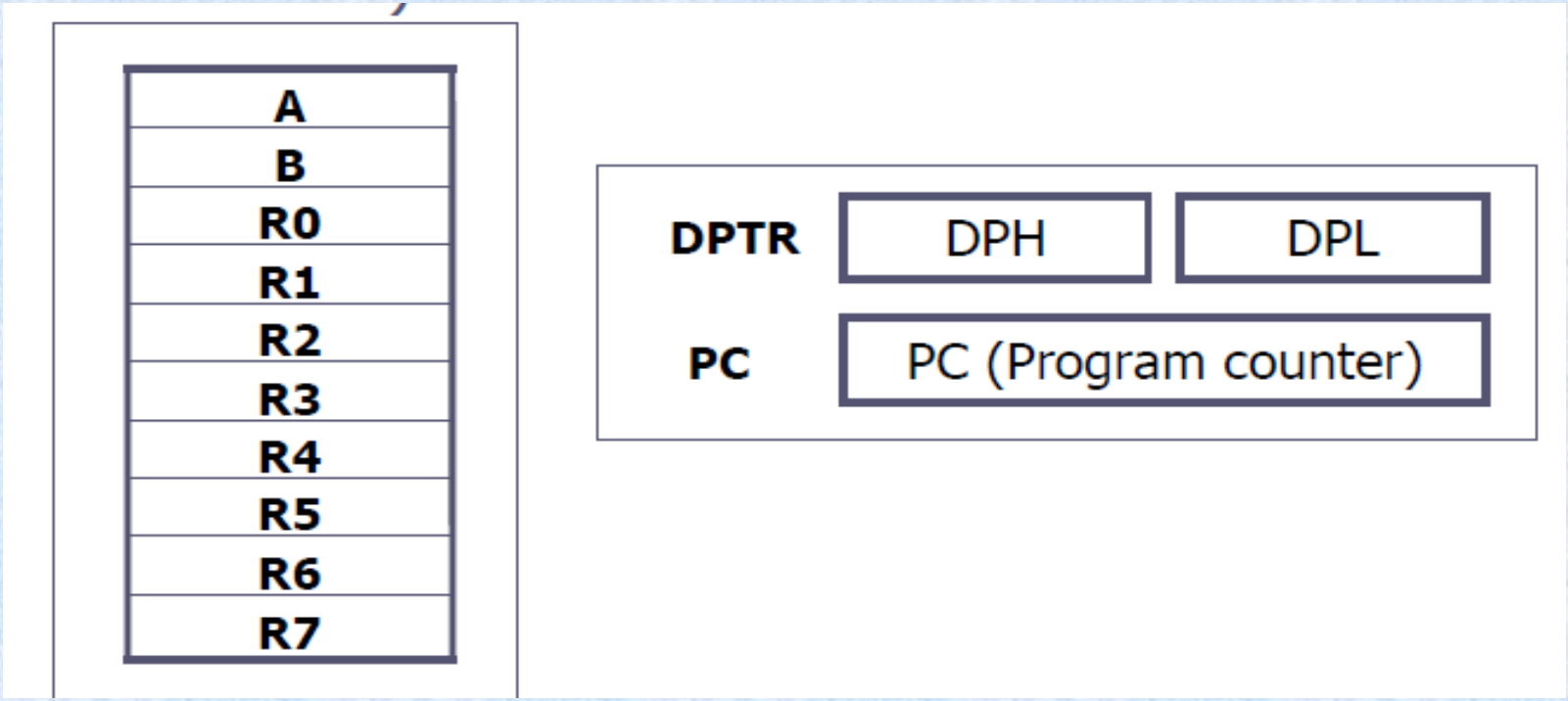
- When Control =1,Po used as address /data bus.

Port Pin	Alternate Function
P3.0	RXD (serial input port)
P3.1	TXD (serial output port)
P3.2	$\overline{\text{INT0}}$ (external interrupt)
P3.3	$\overline{\text{INT1}}$ (external interrupt)
P3.4	T0 (Timer/Counter 0 external input)
P3.5	T1 (Timer/Counter 1 external input)
P3.6	$\overline{\text{WR}}$ (external data memory write strobe)
P3.7	$\overline{\text{RD}}$ (external data memory read strobe)

The most widely used registers

1. A (Accumulator) For all arithmetic and logic instructions
2. B, R0, R1, R2, R3, R4, R5, R6, R7

- DPTR (data pointer), and PC (program counter)



Architecture contains following main components

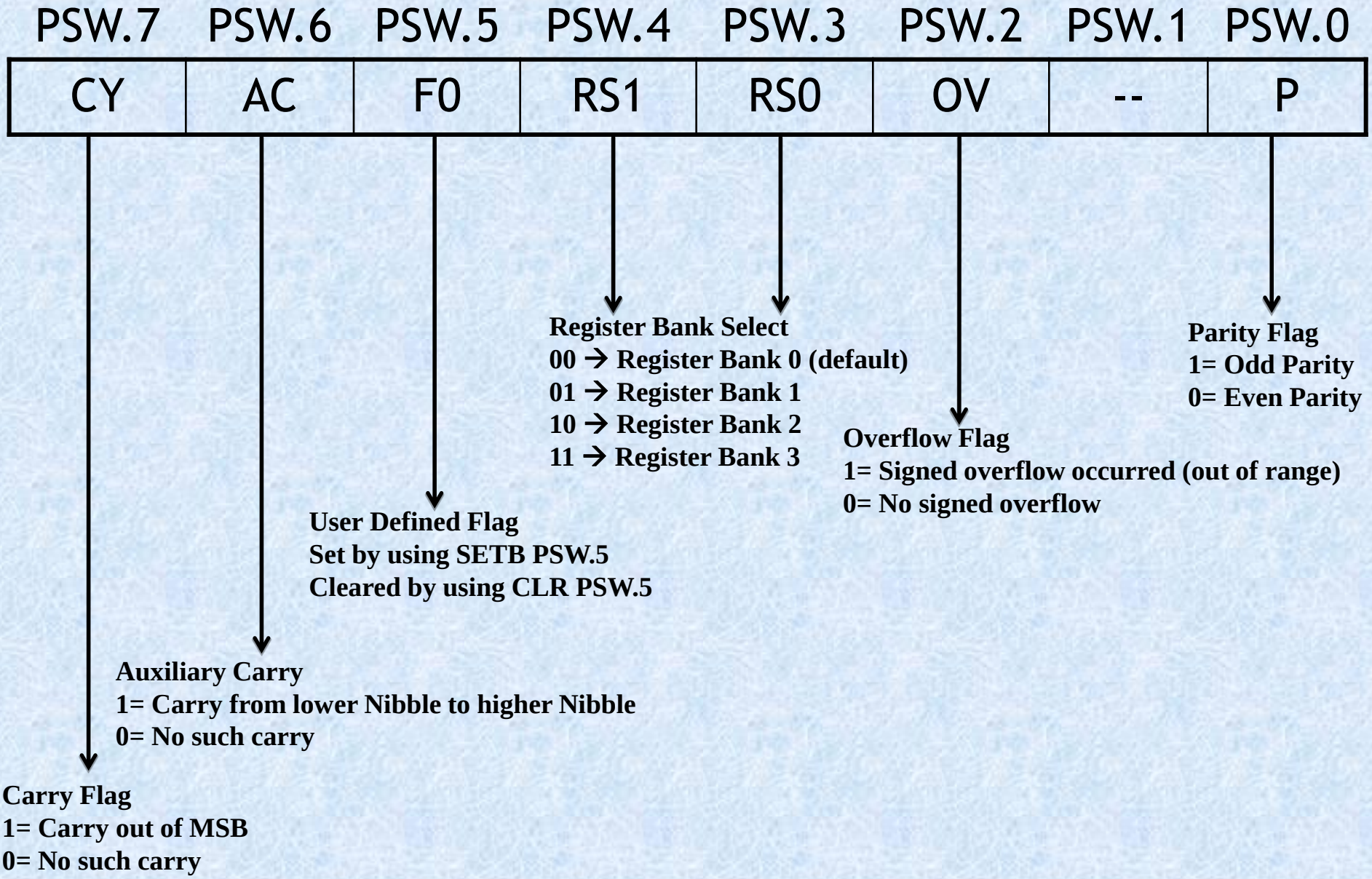
- **CPU contains**
  - ALU (8-bit)
    - It performs arithmetic & logic operations on 8-bit data
    - It also performs 1-bit operations
  - **Accumulator (8-bit)**
    - It is a Special Function Register (SFR) available to the progr ammer
    - It is the **only register which can interact with external m emory devices**
  - B Register **(8-bit)**
    - It is special 8-bit register used for multiplication and divisio n
    - It holds the operand along with the accumulator and also h olds the result
    - E.g. MUL A B ; B.A ← A x B

- PC (16-bit)
  - It holds the address of the program memory (ROM).
  - It is used to fetch instructions and provides ROM address du ring data transfer.
- DPTR (16-bit)
  - It is divided into two 8-bit registers DPH and DPL
  - It holds the memory address of RAM or ROM during data tra nsfer
- SP (8-bit)
  - It is an SFR
  - It holds the **address of the top of the stack**
  - The **stack can be placed** anywhere **in the internal RAM**
  - During PUSH the SP is Incremented by 1
  - During POP the SP is Decremented by 1
  - **When 8051 is initialized**, the SP contains the **value 07H**

Program Status Word

- PSW is 8 bit register.(Flag register)
- CV,AC,P,OV are conditional flags.

- PSW (8-bit)
  - PSW contains the flag bits as follows



CY(Carry flag):-

- This flag set when there is carry out from D7 bit.
- This flag bit is affected after 8-bit addition or subtraction.

AC

- Carry from D3 to D4 during ADD or SUB operation.

P

- No. Of 1’s in A register only.
- If A contain odd no.of 1’s then P =1
- If A contain even no.of 1’s then P =0

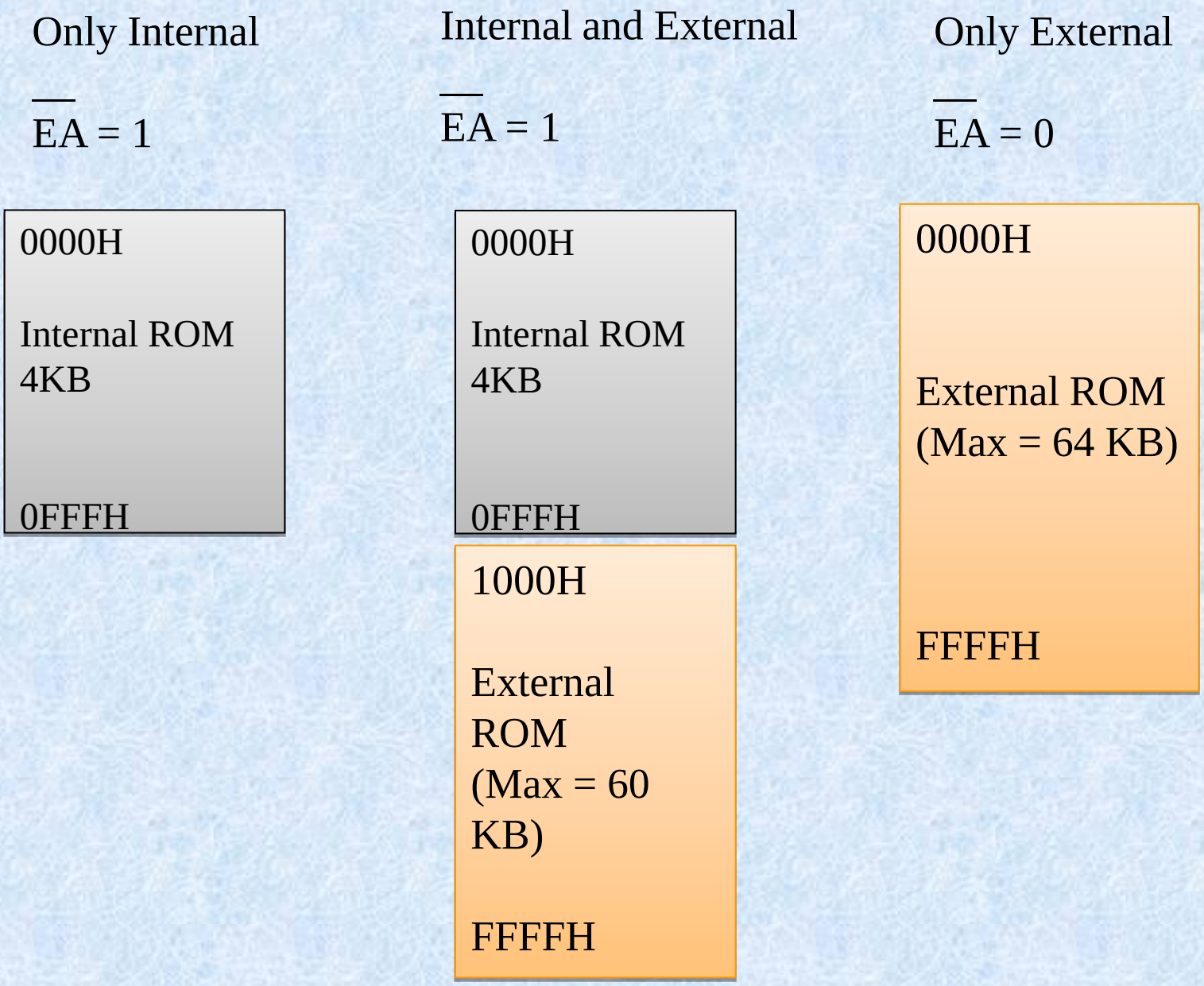
RS1	RS0	Register Bank	RAM Address	
0	0	Bank 0	00H … 07H	
0	1	Bank 1	08H … 0FH	
1	0	Bank 2	10H … 17H	
1	1	Bank 3	18H … 1FH	

Internal Memory

Internal Memory are of two types

- Internal ROM (Program Memory)
  - 8051 has **4KB internal ROM**
  - It mainly contains the executable programs and some data which may never change
  - The internal ROM address range is **0000H…0FFFH**
- Internal RAM (Data Memory)
  - 8051 has **128 Bytes internal RAM**
  - This contains Register Banks, read write data, Bit addressable registers and stack
  - The internal RAM address range is **00H … 07H**
- Additional Components are
  - Timers (2, 16-bit), I/O ports (4, 8-bit), Oscillator, Serial Port, Interrupt logic and Bus control logic

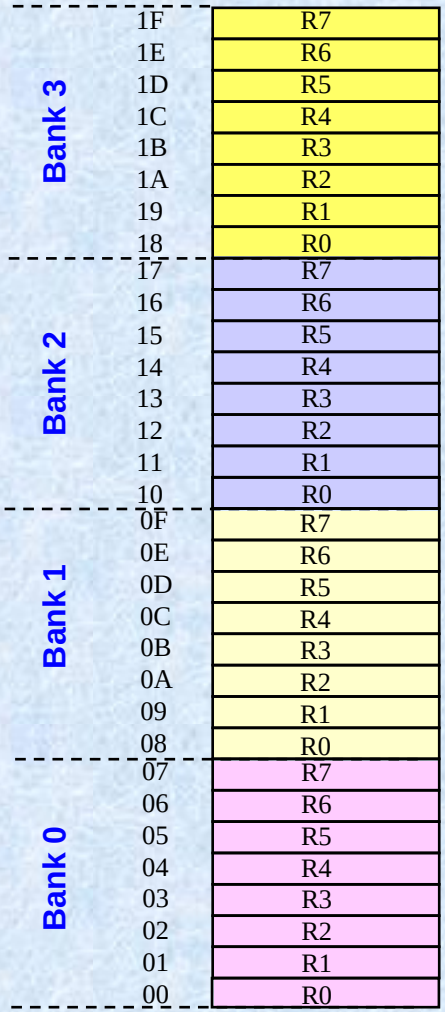
Program Memory Organization of 8051



- The $\overline{EA}$ pin of 8051 decides whether the internal ROM will be used or not
- If the internal ROM has to be used we must make $\overline{EA}$=1
  Now, 8051 will access the internal ROM for all address from 0000H to 0FFFH and will only access external ROM for address 1000H and beyond
- But if $\overline{EA}$=0 then the internal ROM is completely discar ded
  Now, 8051 will access the external ROM for all addresses from 0000H to FFFFH, hence discarding the internal ROM

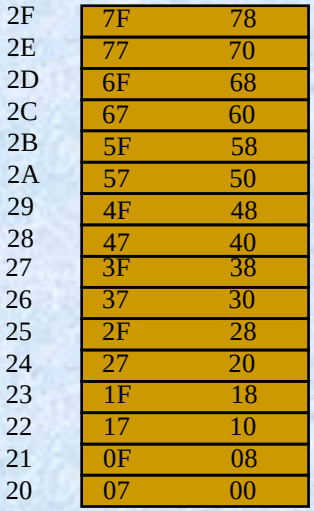
8051 Internal RAM Organisation

128 bytes internal RAM
- 4 register banks of 8 bytes each (R0-R7)
- 16 bytes of bit-addressable area
- 80 bytes of general purpose memory



Working Registers

32 Bytes



Bit Addressable

16 Bytes

General Purpose(Scratch Pad)

80 Bytes

- 8051 has 128x8 (128 Bytes) internal RAM
- The internal RAM has three parts
  - **Register Bank**
    - The first **32 locations** of the internal RAM **(00H … 1FH)** are divided into four banks
    - Each register bank has eight 8-bit registers R0 … R7
    - A **register can be addressed using its name** (e.g. R2)
    - Since **PSW is available** to the programmer, **any bank can be selected at run time**
    - **Bank 0** is selected by **default on reset**
  - **Bit Addressable Area**
    - The next **16 Bytes** of RAM address **20H … 2FH** is available as bit addressable area
    - As each location has 8-bits we have 16x8=128 addressable bits
    - These bits can be addressed using their individual address 00H … 7FH
    - Normal “Byte” operations can also be performed at the addresses: 20H … 2FH
  - **General Purpose Area**
    - The general purpose area ranges from location **30H .. 7FH**
    - This is an **80 Byte** area which can be used for general data storage

- Another important element of the Internal RAM is **STACK**
- **STACK can only be in Internal RAM**
- This is because SP which is an 8-bit register can only contain 8-bit address and External RAM has 16-bit address
- **On reset, SP gets the value 07H**
- Thereafter SP is changed by every PUSH or POP operation in the f ollowing manner

- PUSH:

SP ← SP + 1

[SP] ← New Data

Data ← [SP]

SP ← SP - 1

Special Function Registers (SFRs) of 8051

	Name	Function	Byte address	Bit address
Holding data & Status during Programming	A*	Accumulator	0E0H	0E7H … 0E0H
	B*	Arithmetic	0F0H	0F7H … 0F0H
	PSW*	Program Status Word	0D0H	0D7H … 0D0H
Used in instructions to point to memory	SP	Stack Pointer	81H	81H
	DPL	Address external memory	82H	82H
	DPH	Address external memory	83H	83H
Used by the respective I/O ports	P0*	I/O Port Latch	80H	87H … 80H
	P1*	I/O Port Latch	90H	97H … 90H
	P2*	I/O Port Latch	0A0H	0A7H … 0A0H
	P3*	I/O Port Latch	0B0H	0B7H … 0B0H
Used by Serial Port	SCON*	Serial Port Control	98H	9FH … 98H
	SBUF	Serial Port Data Buffer	99H	99H

* Bit addressable registers

	Name	Function	Byte address	Bit address
Used for Timer Control	TCON*	Timer / Counter Control	88H	8FH … 88H
	TMOD	Timer / Counter Mode Control	89H	8AH
	TL0	Timer 0 Low Byte	8AH	8BH
	TL1	Timer 1 Low Byte	8BH	8BH
	TH0	Timer 0 High Byte	8CH	8CH
	TH1	Timer 1 High Byte	8DH	8DH
Used for Interrupt Control	IE*	Interrupt Enable	0A8H	0AFH … 0A8H
	IP*	Interrupt Priority	0B8H	0BFH … 0B8H
Used for Power Control	PCON	Power Control	87H	87H

* Bit addressable registers

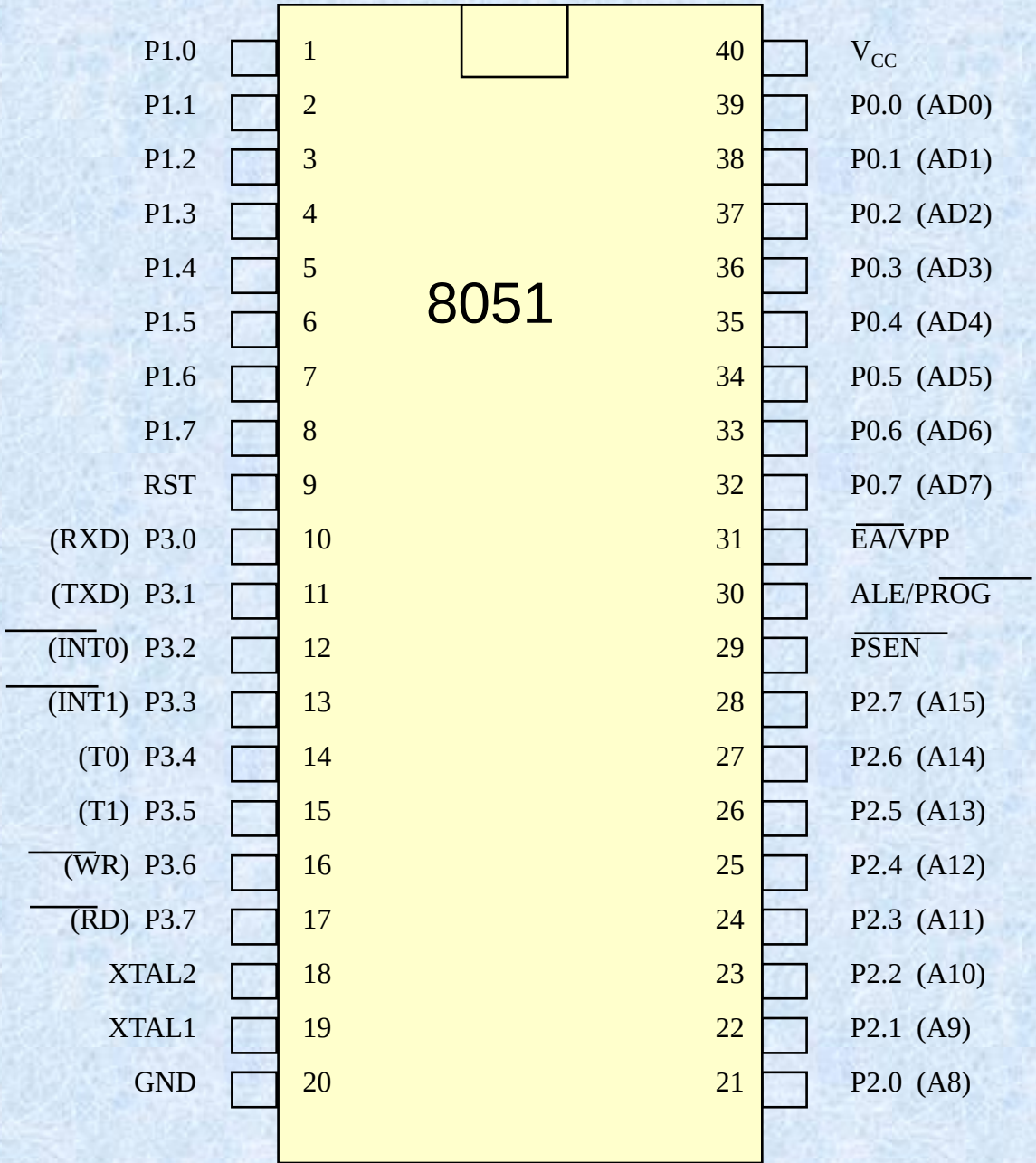
- SFRs are **8-bit registers** each having its own functions
- They are placed inside the microcontroller
- SFRs are allotted addresses **from 80H … FFH**
- If an **address begins with an character**, it is **preceded with a zero "0",** so that the assembler identifies it as an address and not a label

- Addressing the SFRs
- For **Byte wise operations**
  - SFRs are **not addressed by their name** like TLO or TCON etc **(except A)**
  - Instead we use respective addresses
  - Eg. To put the value 25H into B, we must use its address 0F0H as follows

    MOV 0F0H, #25H
  - However to put 25H into A, we can write

    MOV A, #25H
- For **Bit wise operations**
  - The individual bits of the Bit Addressable SFRs can be operated **using thei r bit address** as follows
  - Eg. The bits of P0 will be addressed as 80H … 87H (i.e. P0.0 … P0.7)
  - The bits of PSW will be addressed as 0D0H … 0D7H (i.e. PSW.0 … PSW.7)
  - The bits of TCON will be addressed as 88H … 8FH (i.e. TCON.0 … TCON.7) etc

The 8051 Pin Assignments

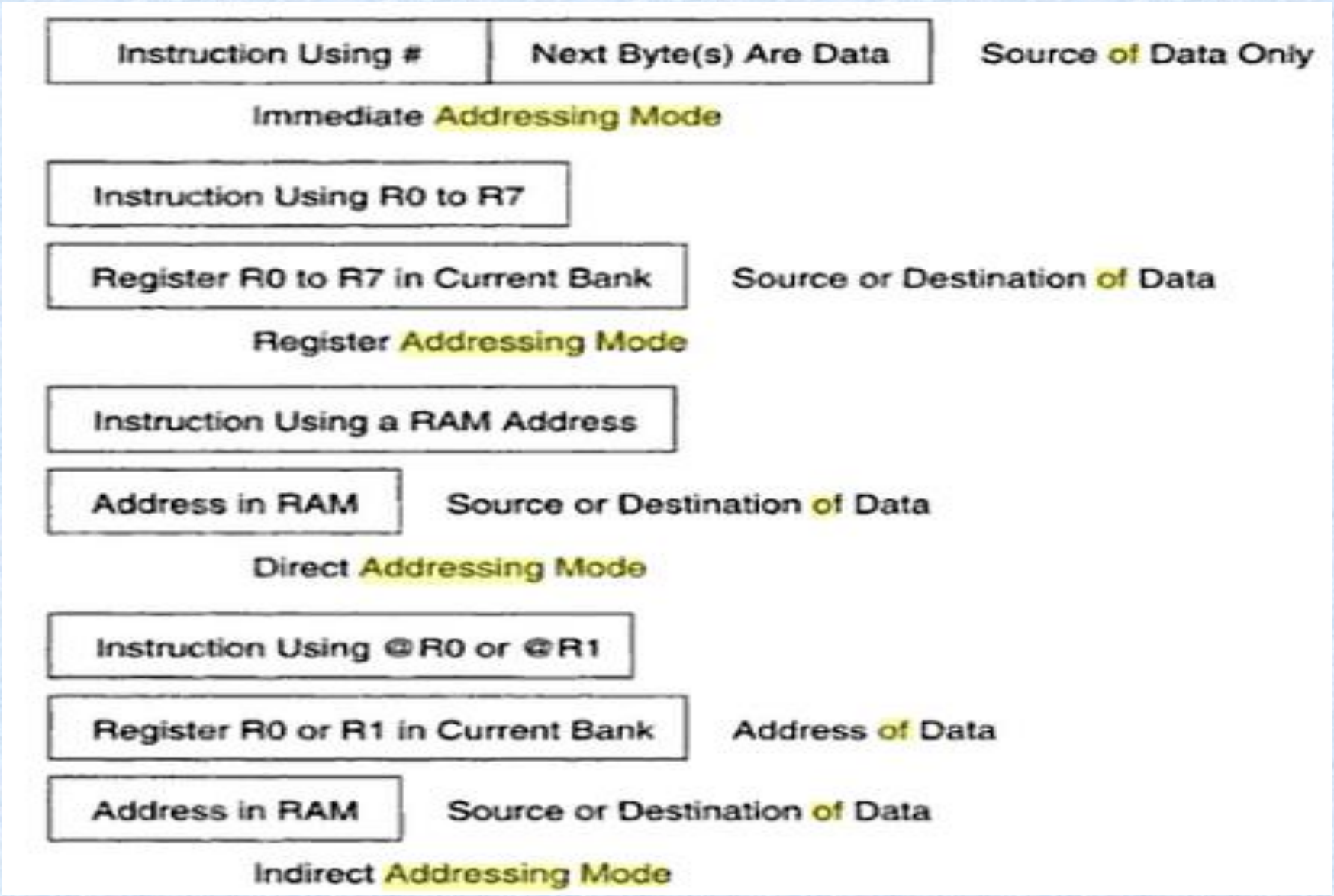


Addressing Modes

- It is the manner in which operands are given in the ins truction
- 8051 supports 5 Addressing Modes
  1. Immediate Addressing Mode
  2. Register Addressing Mode
  3. Direct Addressing Mode
  4. Indirect Addressing Mode
  5. Indexed Addressing Mode

Addressing Mode

- Data is stored at a source address & moved (data is co pied) to a destination address.
- The way by which these addresses are specified are ca lled addressing modes.


Instruction Using #
Next Byte(s) Are Data
Source of Data Only
Immediate Addressing Mode
Instruction Using R0 to R7
Register R0 to R7 in Current Bank
Source or Destination of Data
Register Addressing Mode
Instruction Using a RAM Address
Address in RAM
Source or Destination of Data
Direct Addressing Mode
Instruction Using @R0 or @R1
Register R0 or R1 in Current Bank
Address of Data
Address in RAM
Source or Destination of Data
Indirect Addressing Mode

Immediate Addressing Mode

- In this addressing mode, the **Data is specified in the instruction** itself
- It has a "**#**" symbol before the data in the instruction
- Eg.

```
MOV A, #35H          ; A ← 35H
MOV DPTR, #3000H     ; DPTR ← 3000H
```

Register Addressing Mode

- In this addressing mode, **Data is specified using registers** in the i nstruction

- The permitted registers are A, R7, … R0 of each memory bank

Note: Data transfer between two RAM registers is not allowed

- Eg. MOV A, R0 ; A ← R0

  MOV R5, A ; R5 ← A

  MOV Rx, Ry ; NOT ALLOWED

Direct Addressing Mode

- In this addressing mode, the **address of the operand is specified** in the instruction

- Only Internal RAM addresses (00H … 7FH) and SFR addresses (80H … FFH) allowed

- Eg. MOV A, 35H ; A ← Contents of RAM location 35H

  MOV A, 80H ; A ← Contents of RAM location 80H

  MOV 20H, 30H ; [20H] ← [30H]

Indirect Addressing Mode

- In this addressing mode, the **address of the operand is specified in the register**

- Internal RAM and External RAM can be accessed using t his mode

INTERNAL RAM: (8-bit address given by R0 or R1)

- **Only R0 or R1**, called as Data Pointer **can be used** to specify address (00 H … 7FH)

- An **"@" sign** is present before the register to indicate that the register is giving an address

- Eg. MOV A, @R0 ; A ← [R0]

  i.e. A ← contents of Internal RAM location whose address is given by R0

  MOV @R1, A ; [R1] ← A

  i.e. Internal RAM location pointed by R1 gets value of A

EXTERNAL RAM: (8-bit address given by R0 or R1)

- If **R0 or R1 is used** to give an address, then only the **first 256 locations** o f External RAM is available (0000H … 00FFH)

- This is because **R0 & R1 are 8-bit**

- An "**X**" is present in the instruction to indicate External RAM

- Eg. MOV**X** A, @R0 ; A ← [R0]^

  i.e. A will get the contents of External RAM location whose address is given by R0

MOV**X** @R1, A ; [R1]^ ← A

i.e. A is stored at the External RAM location whose address is given by R1

EXTERNAL RAM: (16-bit address given by DPTR)

- For External RAM, address is provided by **R1, R0 or by DPTR**

- **If DPTR is used** to give an address, then **full 64KB** range of External RAM from 0000H … FFFFH is available

- This is because **DPTR is 16-bit**

- An "**X**" is present in the instruction to indicate External RAM

- Eg. MOV**X** A, @DPTR ; A ← [DPTR]^

  i.e. A gets contents of External RAM location whose address is given by DPTR

  MOV**X** @DPTR, A ; [DPTR]^ ← A

  i.e. A is stored at the External RAM location whose address is given by DPTR

Indexed Addressing Mode

- This mode is used to **access data from Code Memory** (Internal ROM or External ROM)
- In this addressing mode, **address is indirectly specified as a "SUM"** of (A and DPTR) or (A and PC)
- This is very useful because **ROM contains permanent data which is stored in the form of look-up table**
- To access look-up tables, **address is given as a SUM of two registers**, where **one acts a s a base and the other acts as the index** within the table
- A "**C**" is present in such instructions to indicate Code memory

- Eg. MOVC A, @ A+DPTR ; A ← contents of ROM location pointed by A+DPTR

i.e. if DPTR = 0400H and A = 05H, then A gets the contents of ROM location whose address is 0405H

MOVC A,@ A+PC ; A ← Contents of ROM location pointed by A+PC

- The same instruction may operate on **Internal or External ROM**, depending upon the a ddress and the value of **EA pin of 8051**

Selecting Register Banks

- **1st Method**

  MOV PSW, #00001000B

Or MOV PSW, #08H

- **2nd Method**

  CLR PSW.4

  SETB PSW.3

The **second method is better** because it does not disturb other bits of the PSW registers

Storing contents of A and B registers in STACK and retrieving back

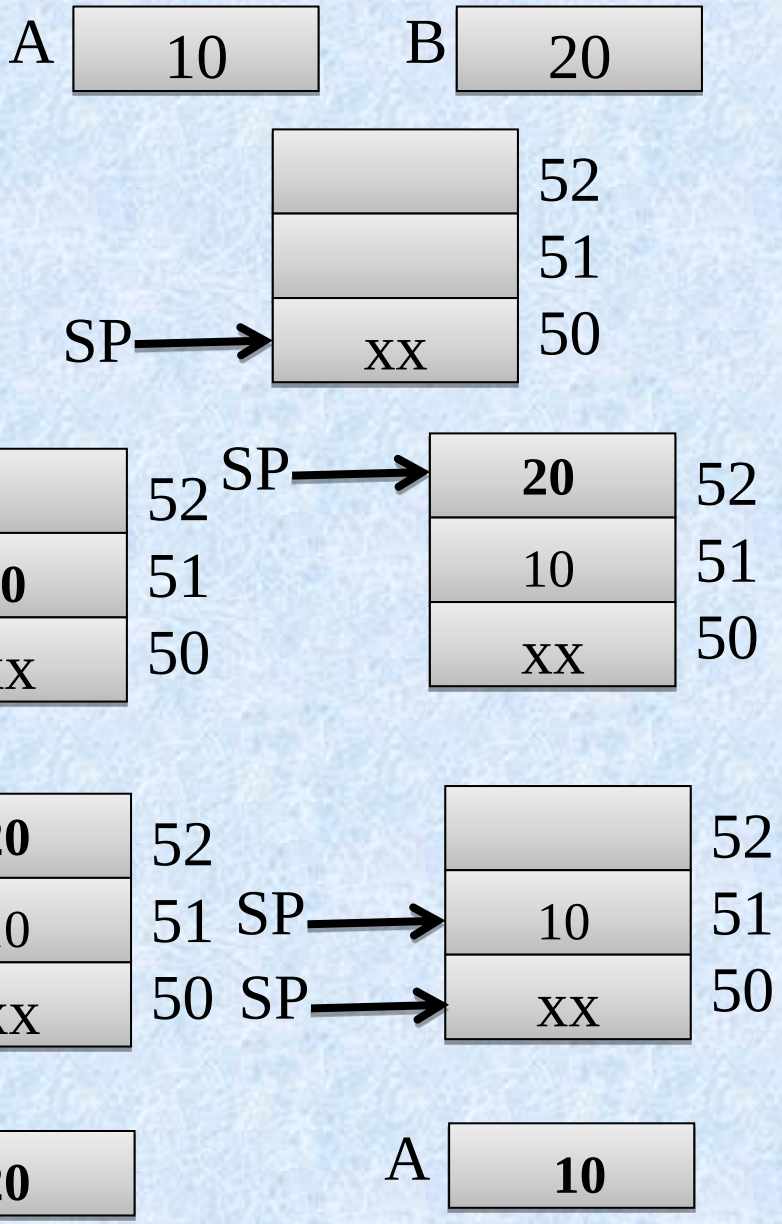
```
MOV A, #10H        ; A = 10H
MOV B, #20H        ; B = 20H
MOV SP, #50H       ; SP= 50H

PUSH 0E0H          ; SP=SP+1 (51H)
                     [51H] ← A

PUSH 0F0H          ; SP=SP+1 (52H)
                     [52H] ← B

POP 0F0H           ; B← [52H]
                     SP=SP-1 (51H)

POP 0E0H           ; A ← [51H]
                     SP=SP-1 (50H)
```



Example 1-2

State the contents of RAM locations after the following program:

```
MOV   R0, #99H
MOV   R1, #85H
MOV   R2, #3FH
MOV   R7, #63H
MOV   R5, #12H
```

Assume register bank 0 was selected !

After the execution of the above program we have the following:

RAM location 0 has value 99H RAM location 1 has value 85H
RAM location 2 has value 3FH RAM location 7 has value 63H
RAM location 5 has value 12H

Example 1-3

Repeat Example 1-2 using RAM addresses instead of register names.

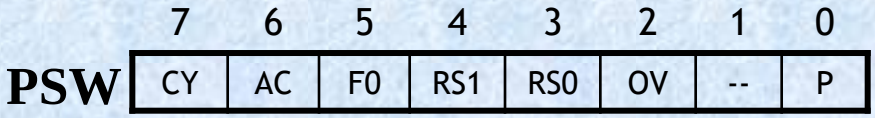
This is called direct addressing mode and uses the RAM address location for the destination address.

```
MOV  00, #99H
MOV  01, #85H
MOV  02, #3FH
MOV  07, #63H
MOV  05, #12H
```

Example 1-4

State the contents of RAM locations after the following program:

```
SETB    PSW.4
MOV     R0, #99H
MOV     R1, #85H
MOV     R2, #3FH
MOV     R7, #63H
MOV     R5, #12H
```



By default, PSW.3=0 and PSW.4=0;

therefore, the instruction "SETB PSW.4" sets RS1=1 and RS0=0,

thereby selecting register bank 2. Register bank 2 uses RAM locations 10H – 17H. After the execution of the above program we have the following RAM location 10 has value 99H; RAM location 11 has value 85H; RAM location 12 has value 3FH; RAM location 17 has value 63H; RAM location 15 has value 12H

Stack and Stack Pointer (SP)

- ***SP*** is a 8-bit register used to hold an internal RAM address that is called the "***top of the stack***"
- ***Stack*** refers to an area of internal RAM that is used in conjunction with certain opcodes to store and retrieve data quickly
- ***SP*** holds the internal RAM address where the last byte of data was stored by a stack operation
- When data is to be placed on the stack, the ***SP*** increments before storing data on the stack so that the stack ***grows up*** as data is stored
- As data is retrieved from the stack, the byte is read from the stack, and then the ***SP*** decrements to point to the next available byte of stored data
- ***SP*** = 07H after reset

Stack Operation

Storing Data on the Stack
(Increment then store)

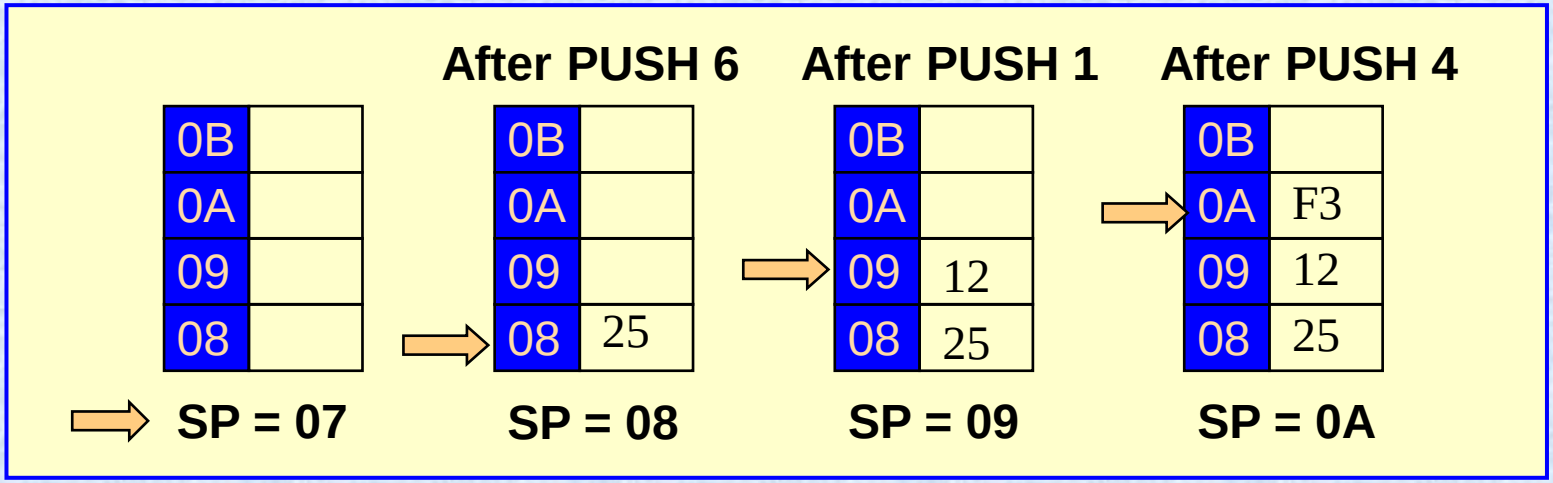
Internal RAM

Getting Data From the Stack
(Get then decrement)

Example 1-5

Show the stack and stack pointer for the following. Assume the default stack area.

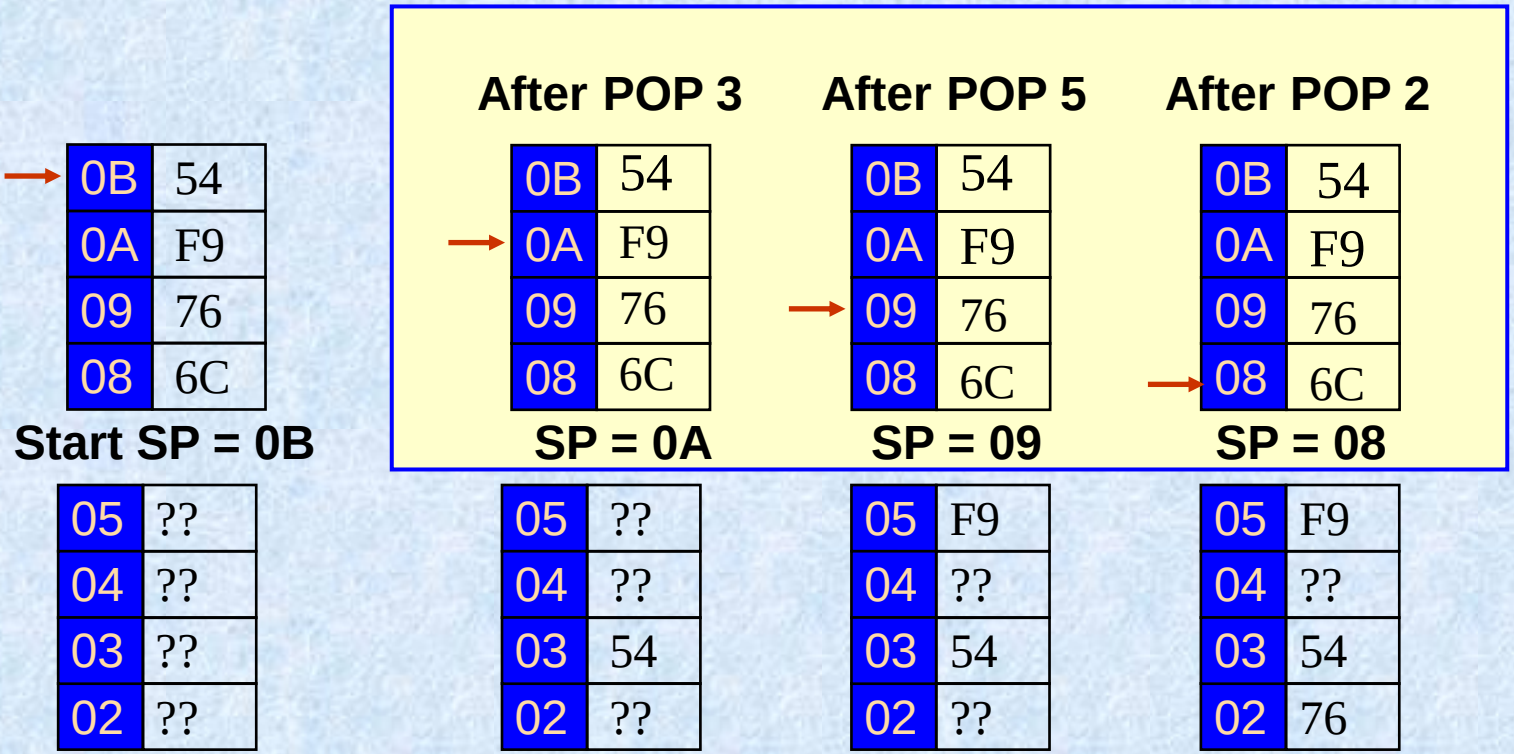
```
MOV   R6, #25H
MOV   R1, #12H
MOV   R4, #0F3H
PUSH  6
PUSH  1
PUSH  4
```



Example 1-6

Examine the stack, show the contents of the registers and ***SP*** after execution of the following instruction. All values are in hex.

```
POP     3       ;POP stack into R3
POP     5       ;POP stack into R5
POP     2       ;POP stack into R2
```



Example 1-7

Show the stack and stack pointer for the following.

```
MOV   SP, #5FH
MOV   R2, #25H
MOV   R1, #12H
MOV   R4, #0F3H
PUSH  2
PUSH  1
PUSH  4
```