National Forensic Sciences University
Knowledge | Wisdom | Fulfilment
An Institution of National Importance
(Ministry of Home Affairs, Government of India)

Microprocessor & Microcontrollers

(CTBTCSE SIV P3)

Dr. Ujjaval Patel

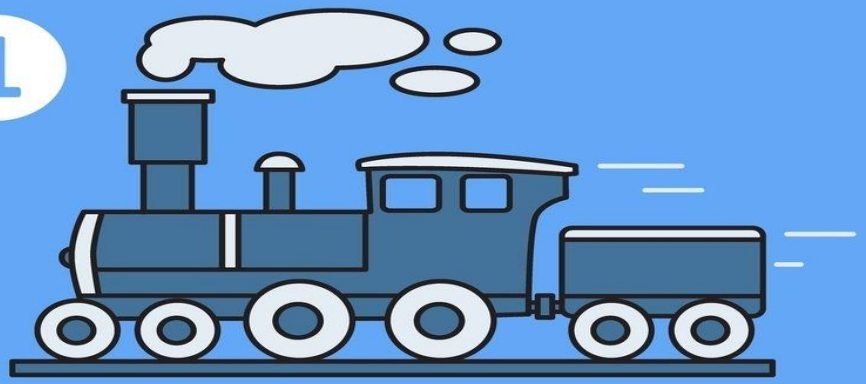
Assistant Professor

School of Cyber Security & Digital Forensics

(M: +91-987 987 97 46, E-mail: ujjaval.patel@nfsu.ac.in)

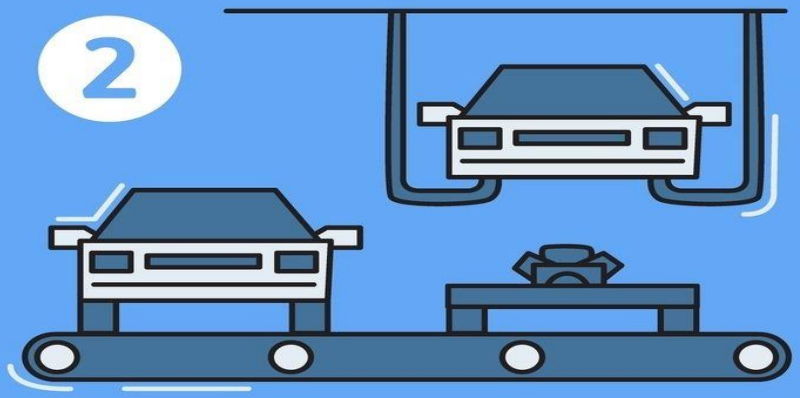
INDUSTRIAL REVOLUTIONS

1



The Industrial Revolution (~1760s-~1840s)
Manufactoring mechanization. Steam powered engines, machine tools.

2



The Second Industrial Revolution (Technological Revolution) (~1870s-~1930s)
Industrialization, electrification. Mass production assembly lines.

3

Third Industrial Revolution (Digital Revolution) (~1970s-nowadays)
Shift to digital electronic automation. Internet, mass comunication, digital social systems.

4

INDUSTRY 4.0 (Fourth Industrial Revolution) (nowadays -...)
Digital technologies interaction.
Cyber-phisical systems, internet of things, cloud networks.

Subject Outline:

- Microprocessors: Intel 8051
- Microprocessor: Intel 8086 Microprocessor
- Advanced Microcontroller: ARM Introduction

Subject Introduction:

- 8051 Microcontrollers
  - ✓Basic block diagram, Architecture
  - ✓Programming: Assembly & C language
  - ✓Timers in 8051
  - ✓Serial Communication
  - ✓Interrupts in 8051
- 8086 Microprocessors
  - ✓Basic block diagram, Architecture
  - ✓Programming: Assembly language
- ARM: Architecture and advanced features.

Pre-requisites: Digital Electronics.

Reference Books:

➢ Microcontrollers: 8051 family

The 8051 Microcontroller and Embedded Systems Using Assembly and C, 2/e by Muhammad Ali Mazidi, Janice Gillispie Mazidi and Rolin McKinlay (Second Edition , Pearson Education)

➢ Microprocessor: 8086

Microprocessors and Interfacing, 3rd Edition, Douglas V. Hall

➢ Advanced Microcontroller: ARM Introduction

ARM System Developer's Guide, Designing & Optimizing System Software, by Andrew Sloss, Dominic Symes, Chris Wright, Elsevier Publications.

Applications ??

- Invented in 1980s and 80% digital applications use 8051.
- Home Automation
- Building Automation
- Industry Automation
- Healthcare
- Domestic appliances
- Robotics
- Satellite communications

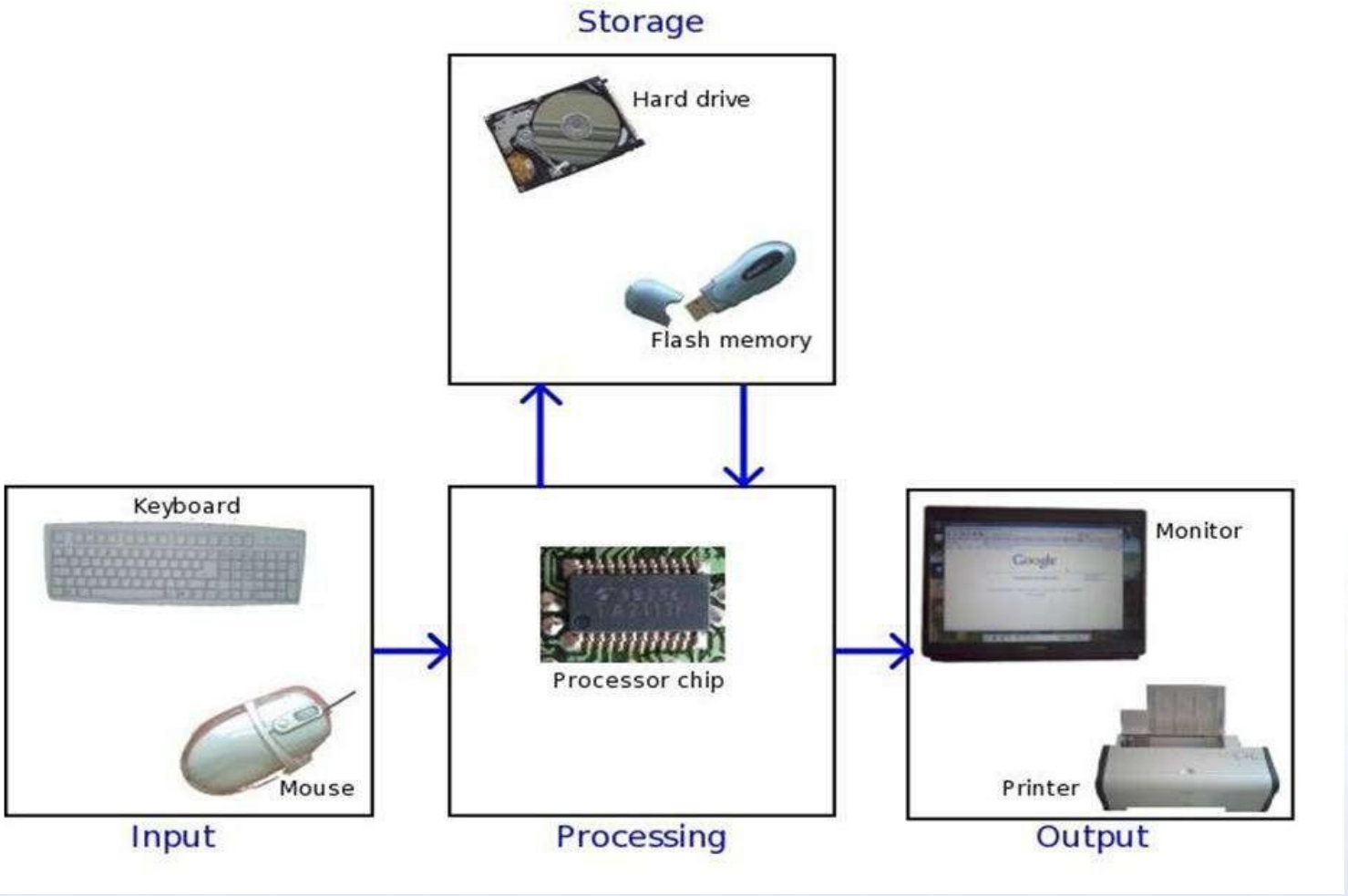
Unit 1: Microprocessor based systems

- Digital computers
- Microprocessors
- Microcontrollers
- Applications
- Evaluations of Microcontrollers & Vendors
- Van Neumann & Harward Architecture
- CISC & RISC processors

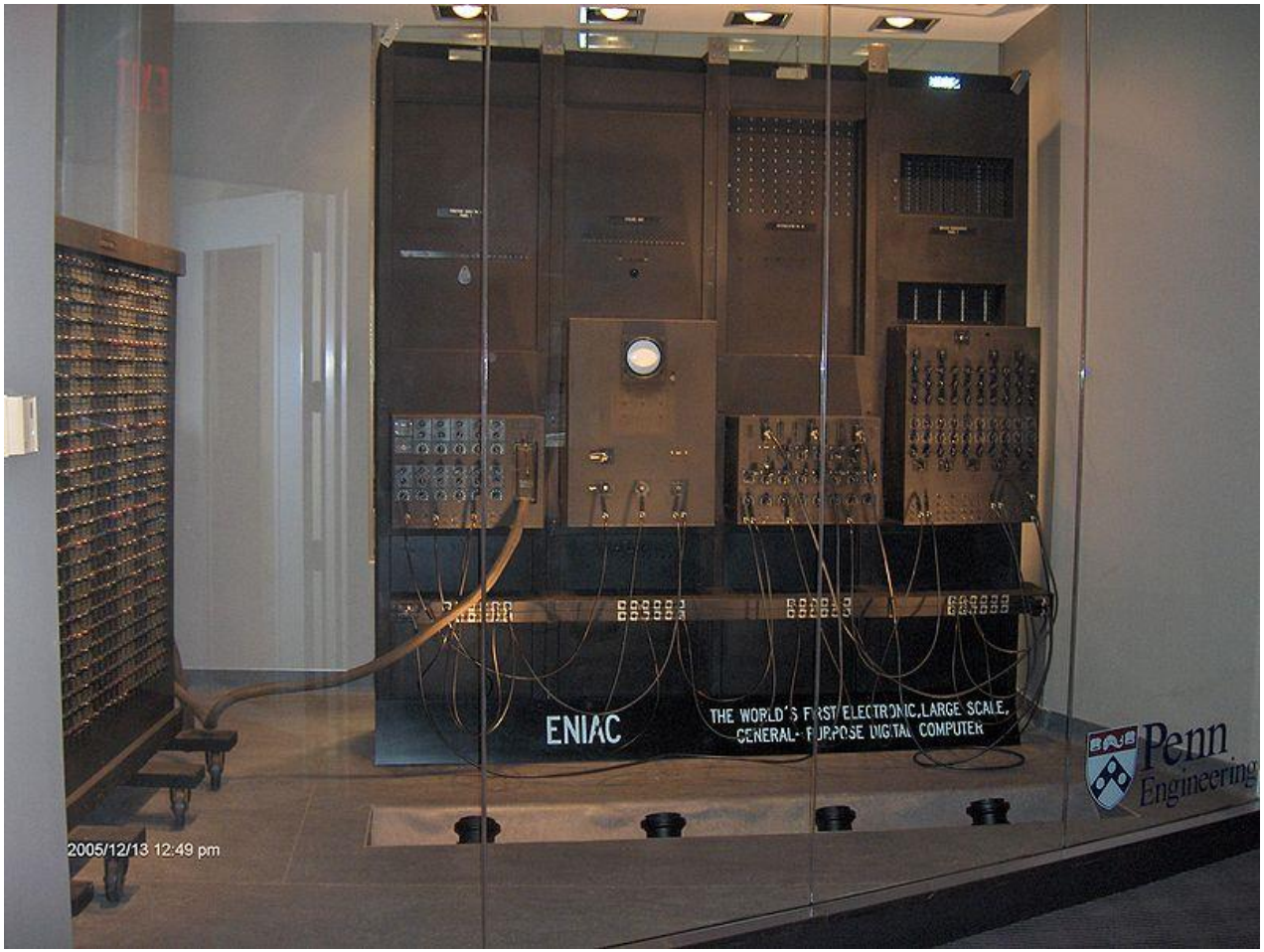
Digital computers

- Microcomputer
- Minicomputer
- Desktop Computer
- Personal Computer
- Portable Notebook Computer
- Workstation
- Mainframes
- Servers
- Super Computer

Digital Computers:



Electronic Numerical Integrator and Computer (ENIC)



Four ENIAC panels and one of its three function tables, on display at the School of Engineering and Applied Science at the University of Pennsylvania **February 15, 1946**

Supercomputer

The IBM Blue Gene/P supercomputer "Intrepid" at Argonne National Laboratory runs 164,000 processor cores using normal data center air conditioning, grouped in 40 racks/cabinets connected by a high-speed 3-D torus network.

Indian Super Computer PARAM YUVA

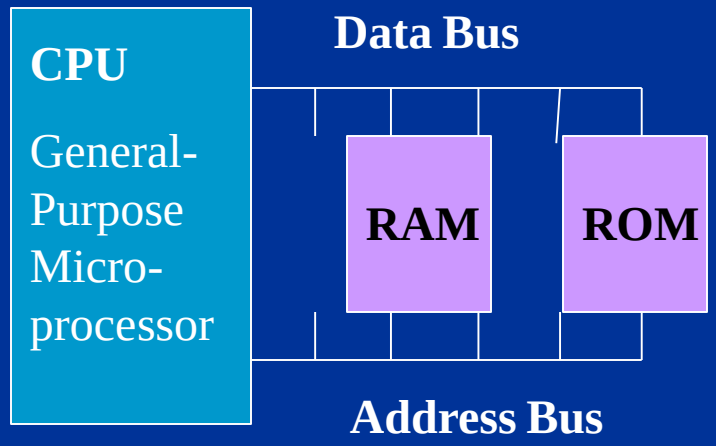
1991 by Centre for Development of Advanced Computing

What is a Microprocessor ??

- General-purpose microprocessor
- CPU for Computers
- No RAM, ROM, I/O on CPU chip itself Example : Intel's x86, Motorola's 680x0

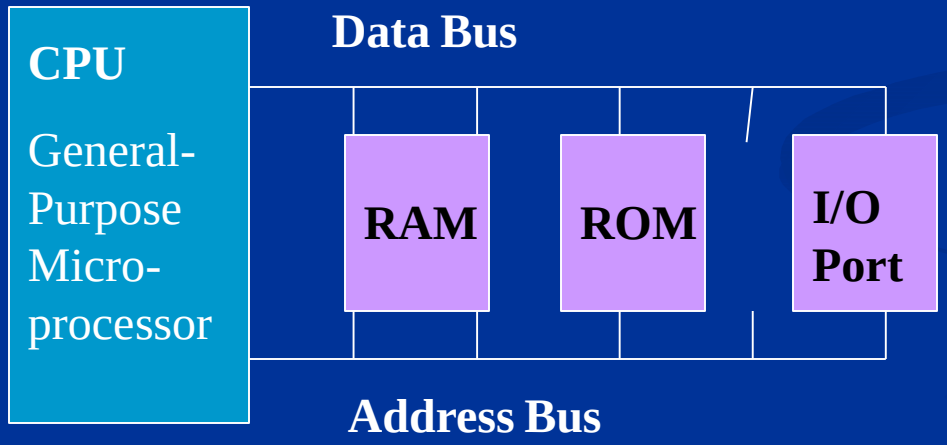
What is a Microprocessor ??

- General-purpose microprocessor
- CPU for Computers
- No RAM, ROM, I/O on CPU chip itself Example : Intel's x86, Motorola's 680x0



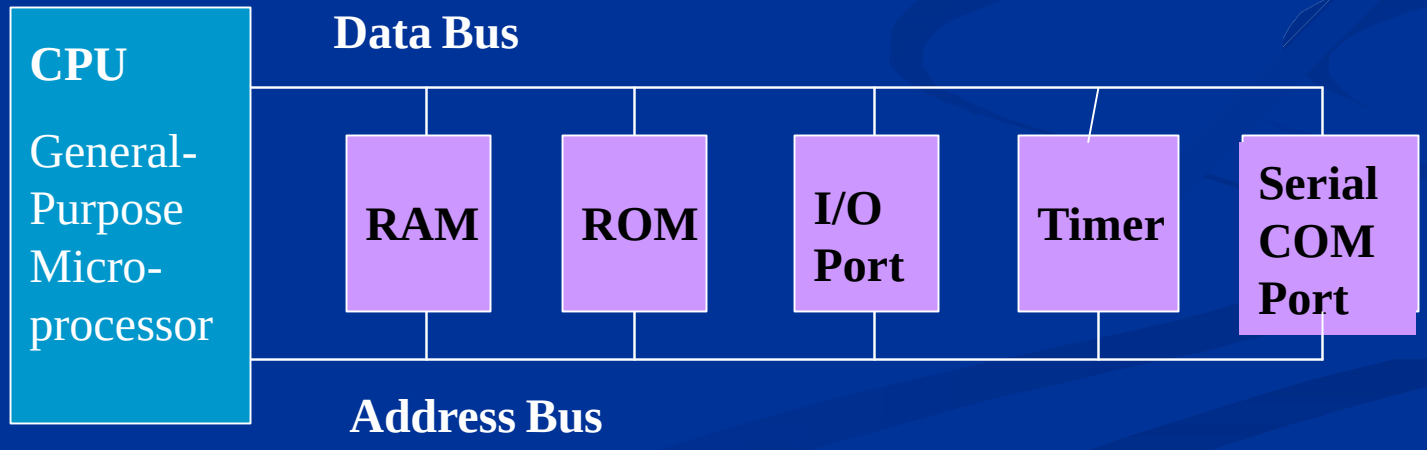
What is a Microprocessor ??

- General-purpose microprocessor
- CPU for Computers
- No RAM, ROM, I/O on CPU chip itself Example : Intel's x86, Motorola's 680x0



What is a Microprocessor ??

- General-purpose microprocessor
- CPU for Computers
- No RAM, ROM, I/O on CPU chip itself Example : Intel's x86, Motorola's 680x0



Evolution of Microprocessor

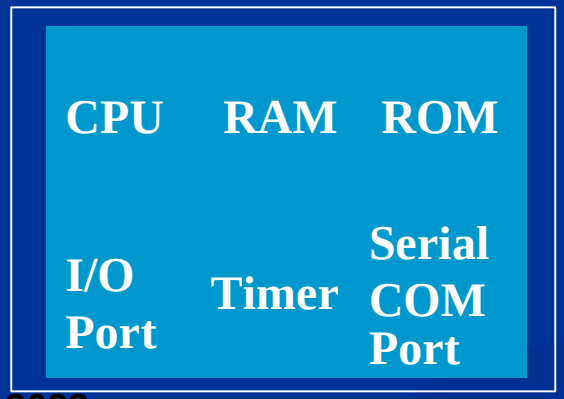
NAME	YEAR	TRANSISTORS	DATA WIDTH	CLOCK SPEED
4004	1971	2300	4 bits	108 kHz
8008	1972	3,500	8 bits	200 kHz
8080	1974	6,000	8 bits	2 MHz
8085	1976	6,500	8 bits	5 MHz
8086	1978	29,000	16 bits	5 MHz
8088	1979	29,000	8 bits	5 MHz
80286	1982	134,000	16 bits	6 MHz
80386	1985	275,000	32 bits	16 MHz
80486	1989	1,200,000	32 bits	25 MHz
PENTIUM	1993	3,100,000	32/64 bits	60 MHz
PENTIUM II	1997	7,500,000	64 bits	233 MHz
PENTIUM III	1999	9,500,000	64 bits	450 MHz
PENTIUM IV	2000	42,000,000	64 bits	1.5 GHz

What is a Microcontroller ?

- Small controller
- On-chip RAM, ROM, I/O ports...
- Example : Motorola's 6811, Intel's 8051, Zilog's Z8 and PIC 16X

What is a Microcontroller ?

- Small controller
- On-chip RAM, ROM, I/O ports...
- Example : Motorola's 6811, Intel's 8051, Zilog's Z8 and PIC 16X



A single chip Microcontroller

Microprocessor vs. Microcontroller

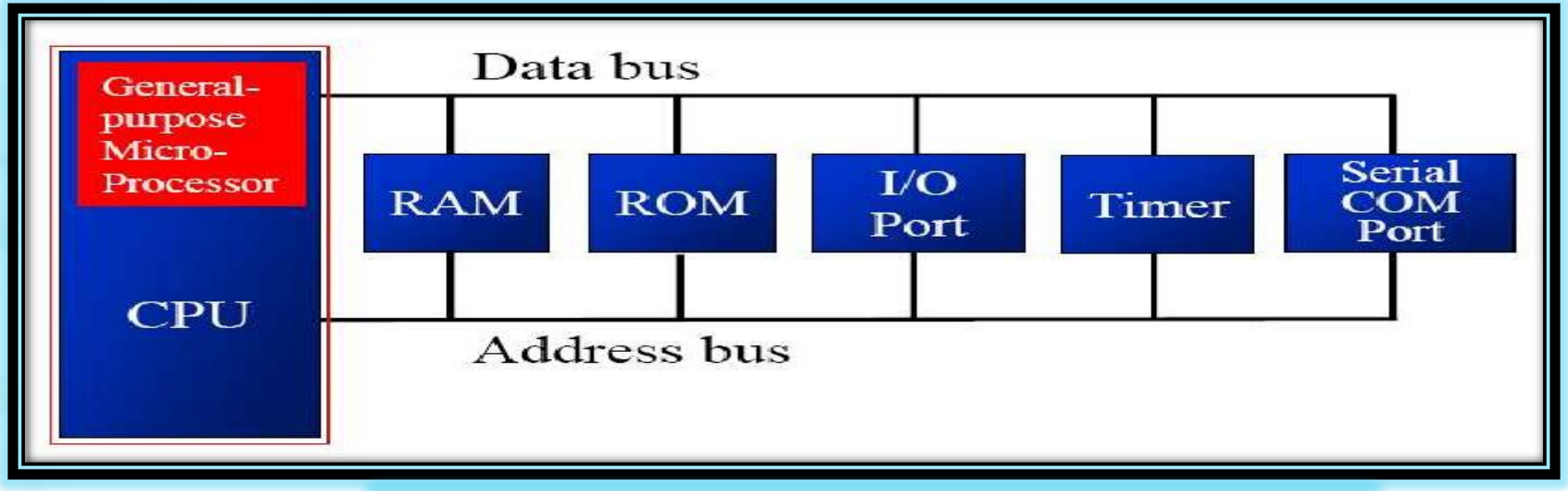
Microprocessor

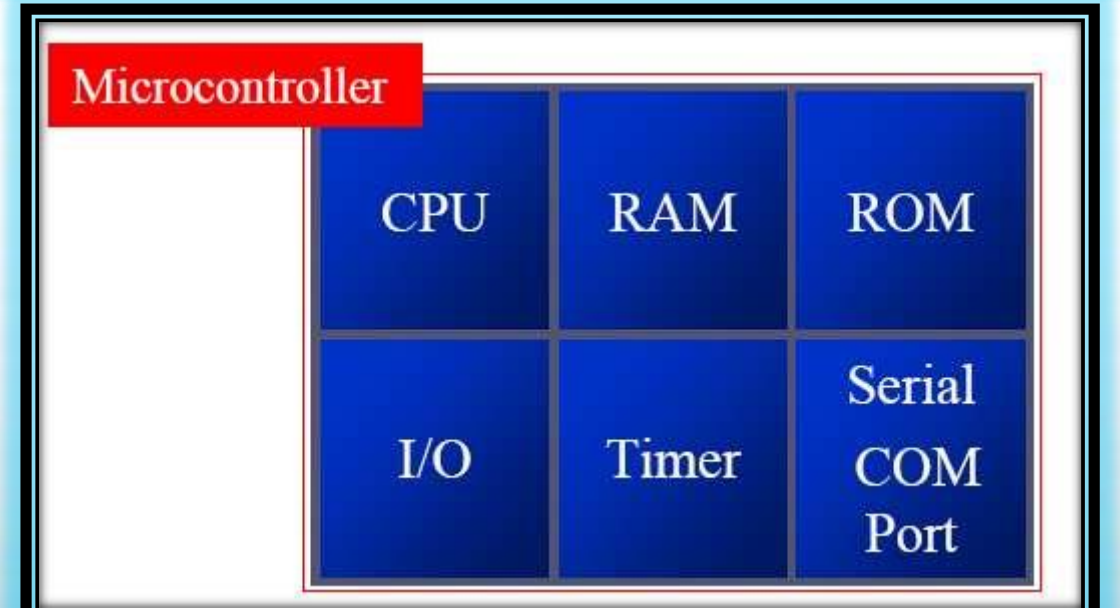
- CPU is stand-alone, RAM, ROM, I/O, timer are separate
- designer can decide on the amount of ROM, RAM and I/O ports.
- expansive
- versatility
- general-purpose

Microcontroller

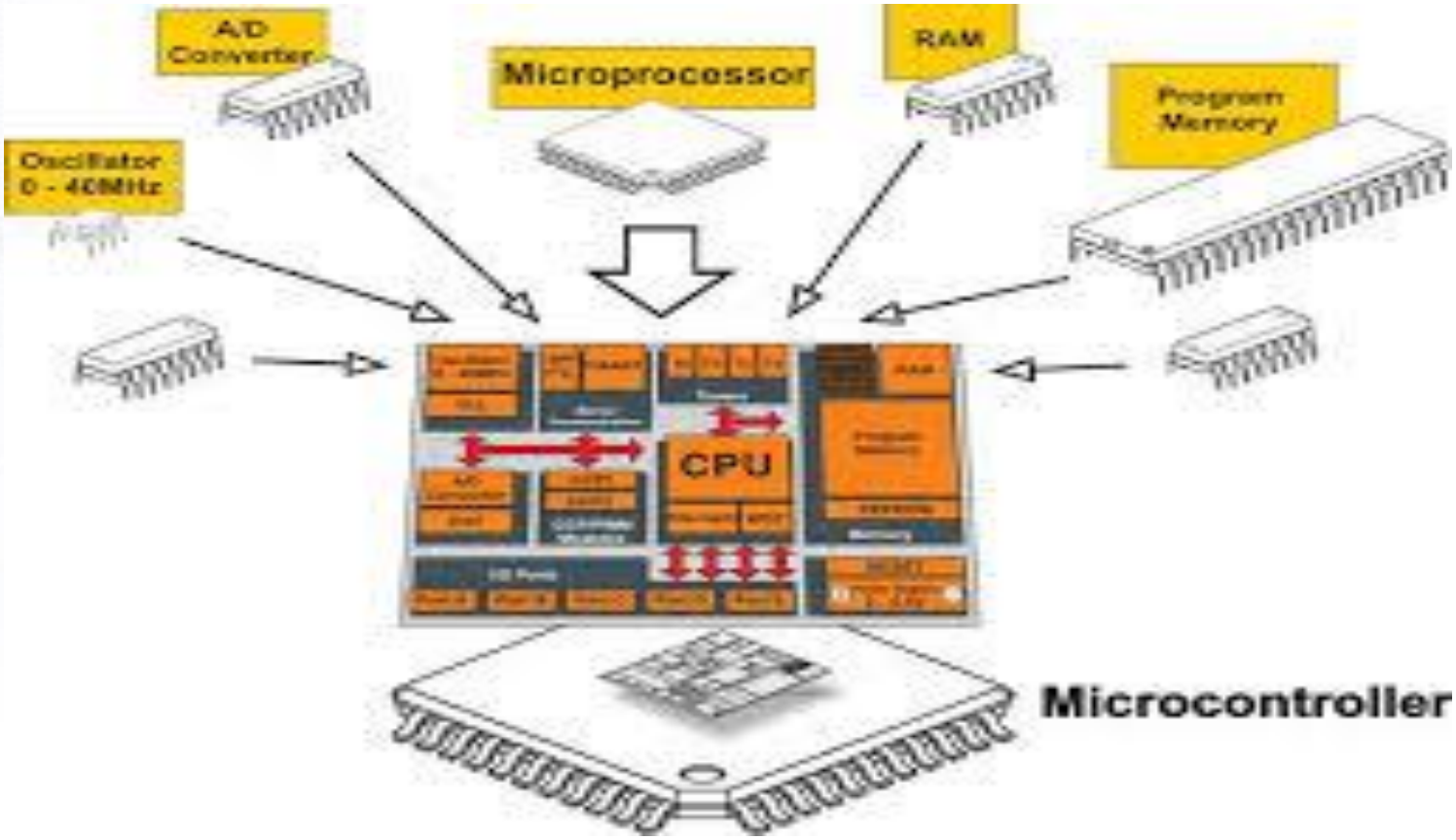
- CPU, RAM, ROM, I/O and timer are all on a single chip
- fix amount of on-chip ROM, RAM, I/O ports
- Highly bit addressable
- for applications in which cost, power and space are critical
- single-purpose

Microprocessor Vs Microcontroller:





Microprocessor vs. Microcontroller



Applications ??

- Personal information products: Cell phone, pager, watch, pocket recorder, calculator
- Laptop components: mouse, keyboard, modem, fax card, sound card, battery charger
- Home appliances: door lock, alarm clock, thermostat, air conditioner, TV remote, VCR, small refrigerator, exercise equipment, washer/dryer, microwave oven
- Industrial equipment: Temperature/pressure controllers, Counters, timers, RPM Controllers
- Toys: video games, cars, dolls, etc.

EVOLUTION

Flashback !!!!

1976: Motorola

1980: Intel 8080 & 8085

MICROCONTROLLER.

1981: Intel 8051

Evolution contd…

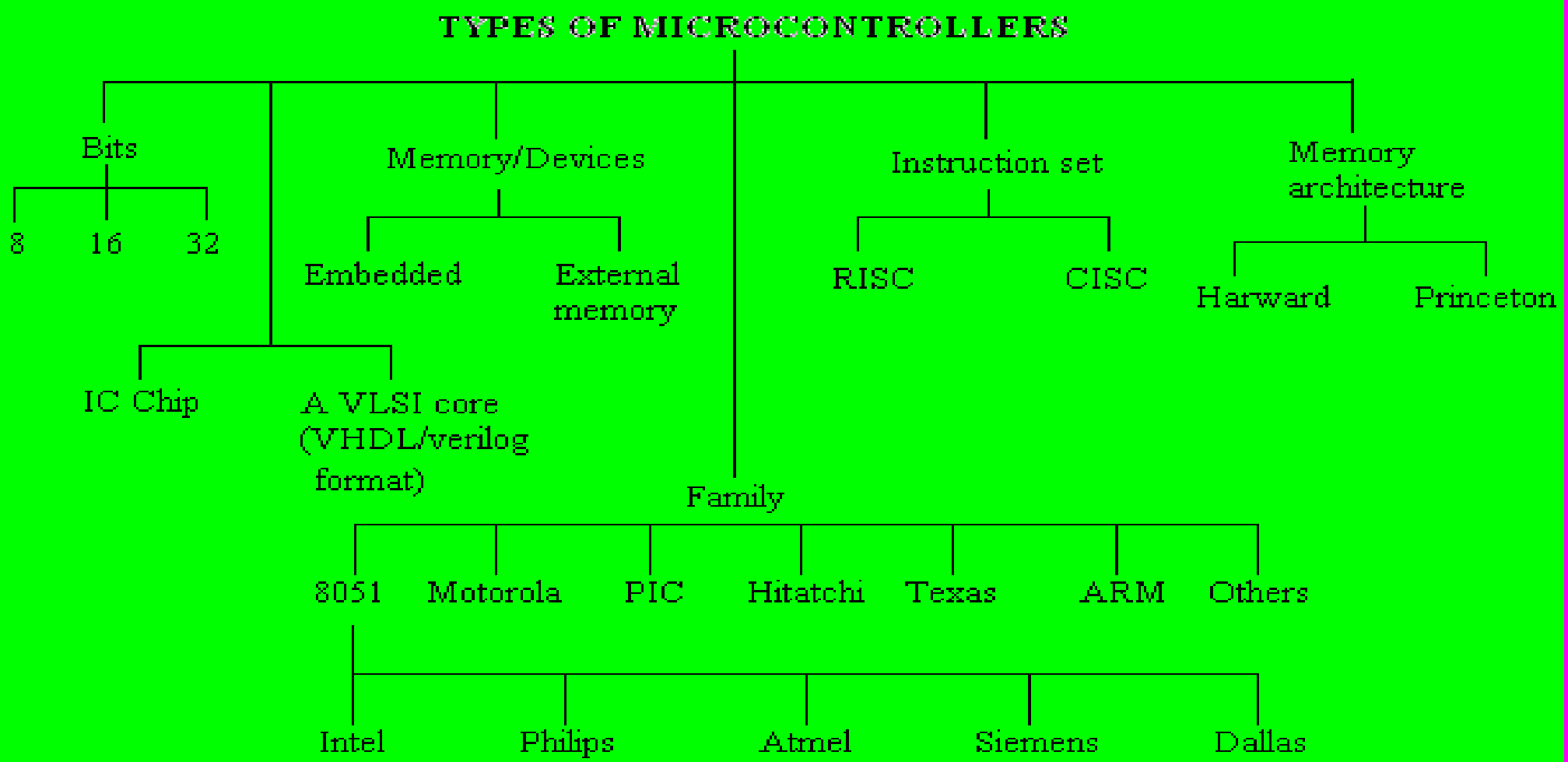
- Then followed the most popular controller 8051 in the year 1980 with 4K bytes of ROM,128 Bytes of RAM , a serial port, two 16-bit Timers , and 32 I/O pins.
- The same INTEL introduced a 16 bit controller 8096 in the year 1982
- Later INTEL introduced 80c196 series of 16-bit microcontrollers for mainly industrial applications
- Microchip, another company has introduced a microcontroller PIC 16C64 an 8-bit in the year 1985.
- 32-bit microcontrollers have been developed by IBM and Motorola-MPC 505 is a 32-bit RISC controller of Motorola
- The 403 GA is a 32 -bit RISC embedded controller of IBM

ARM Controllers

- In recent times ARM company (Advanced Risc machines) has developed and introduced 32 bit controllers which are highend application devices,especially communication devices like mobiles , ipods etc..

(Refer www.arm.com)

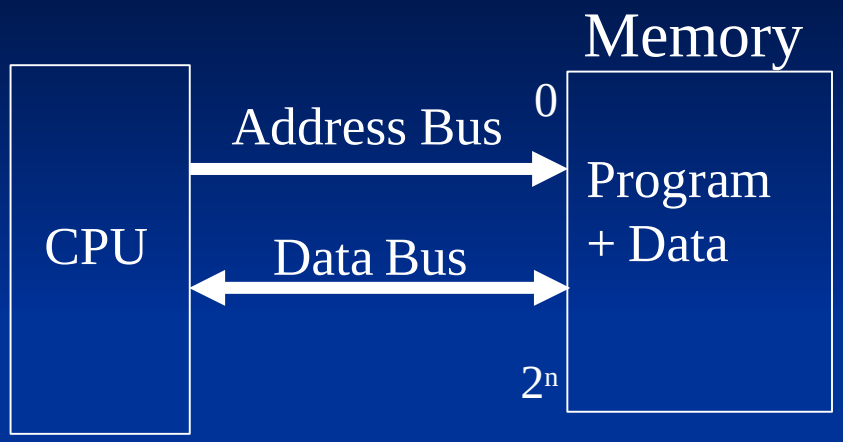
Classifications of Microcontrollers



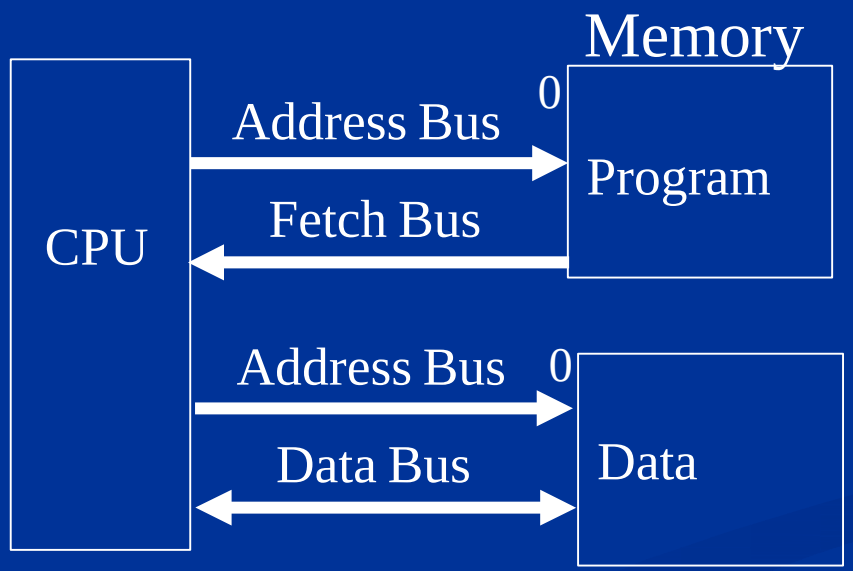
How to classify processors & controllers ?

- Categorized by memory organization
  1. Von-Neumann architecture
  2. Harvard architecture
- Categorized by instruction type
  1. CISC
  2. RISC

Microcontroller Architectures



Von Neumann Architecture

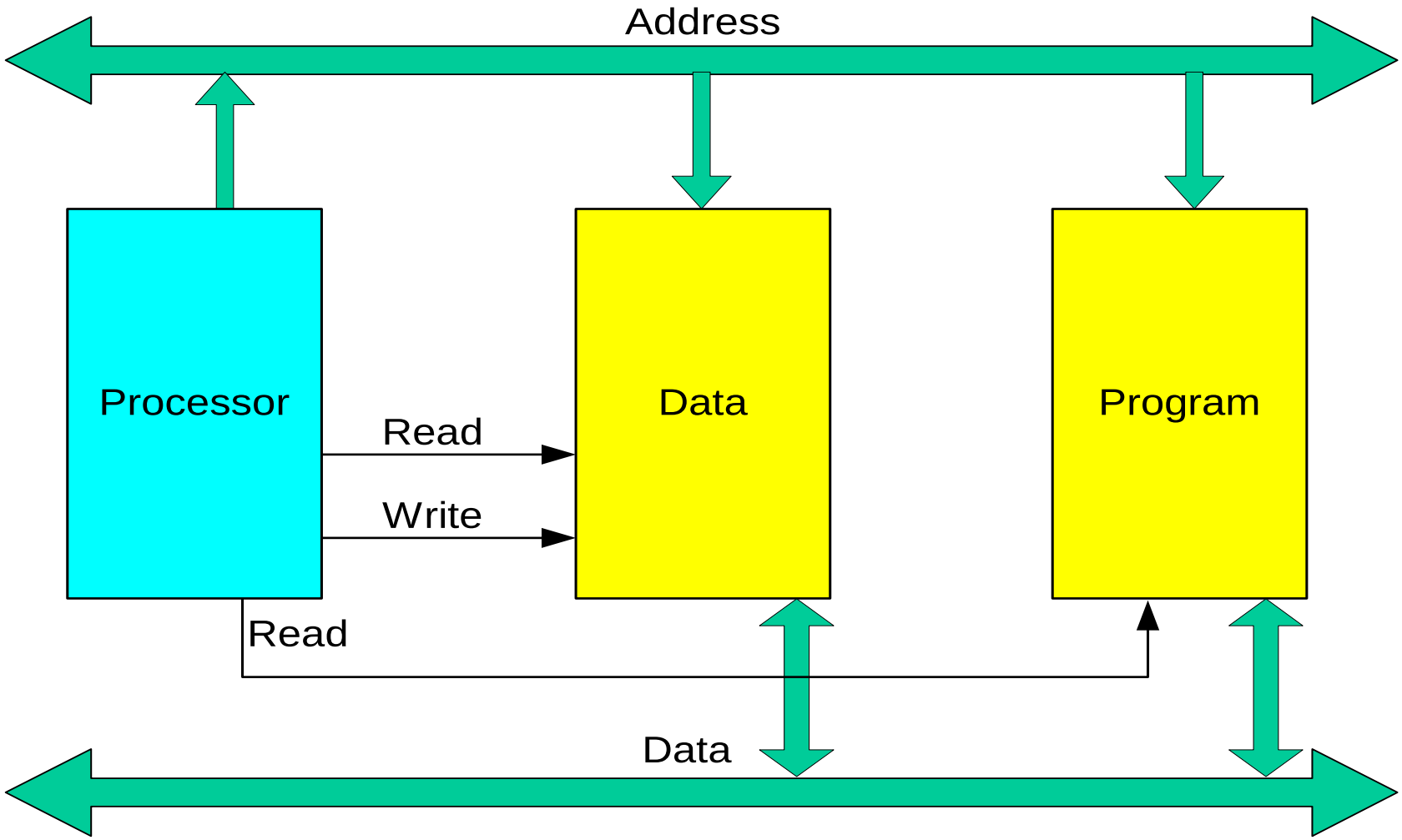


Harvard Architecture

Harvard architecture

- Separate memory into 2 types
  - Program memory
  - Data memory

Harvard architecture



Von-Neumann architecture

- Combine program and data in 1 chunk of memory
- Example : 80x86 architecture

CISC & RISC

Overview

The processor designs are broadly classified into two categories.

i. Complex Instruction Set Computers (CISC) &

ii.Reduced Instruction Set Computers (RISC).

- Earlier days Intel & Motorola processors were mainly based on CISC design. (For ex: 8085 to Pentium & Motorola MC68060)
- But the recent trend is to use the RISC designs. For example the widely used ARM processors are RISC.
- The ARM is a very popular processor in the market which is widely used in smartphones, Laptops etc.

Overview

- The five very popular RISC processors are MIPS, SPARC, PowerPC, Itanium, and ARM.
- Its clear from the above that Intel has also shifted to RISC architecture. For Ex: Itanium is from Intel.
- The RISC approach promises many advantages over CISC architectures including superior performance, design simplicity, rapid development time.

DIFFERENCES ::CISC &RISC

CISC	RISC
1.CISC stands for Complex Instruction Set Computer	RISC stands for Reduced Instruction Set Computer
2.Hardware centric design	Software centric design
3.Complex and variable length instructions	Simple and mostly fixed length instructions.
4.More number of addressing modes	Less number of addressing modes.
5. Instructions take more than one clock cycle for the execution.	Instructions require only one clock cycle
6.Less number of registers	More number of registers

Example of CPU Architectures

- Intel: 80x86
- Motorola: 680x0

CISC

- Sun : Sparc
- Silicon Graphics : MIPS
- HP : PA-RISC
- IBM: Power PC
- Compaq: Alpha

RISC

8051 Microcontroller

8051 microcontroller:

- Introduction: History, difference between Microprocessor & Microcontroller, Applications.
- Criteria for selection of microcontroller
- Features of 8051 microcontroller
- Pin Diagram
- Block diagram & Architecture
- Programming: Address modes & Instructions.
- Special features: Timers, Interrupt, SC

Criteria for Selection:

- Number of Digital & Analog Input-Output pins.
- Memory size: RAM & ROM
- Availability
- Cost
- Power Consumption
- Required skills
- Future expansion

8051 CPU Operation

1. Features
2. Pin Diagram
3. Block Diagram

8051 Microcontroller

- Intel introduced 8051, referred as MCS- 51, in 1981.
- The 8051 is an **8-bit processor**
  - ✓The CPU can work on only 8 bits of data at a time
  - ✓Size of data bus is 8 – bit.
- The 8051 became widely popular after allowing other manufactures to make and market any flavor of the 8051.

Important Features of 8051

- I/O ports: 4
- RAM: 128 bytes
- ROM: 4KB
- Timers: 2
- Serial Com. Port: 1
- Interrupts: 6

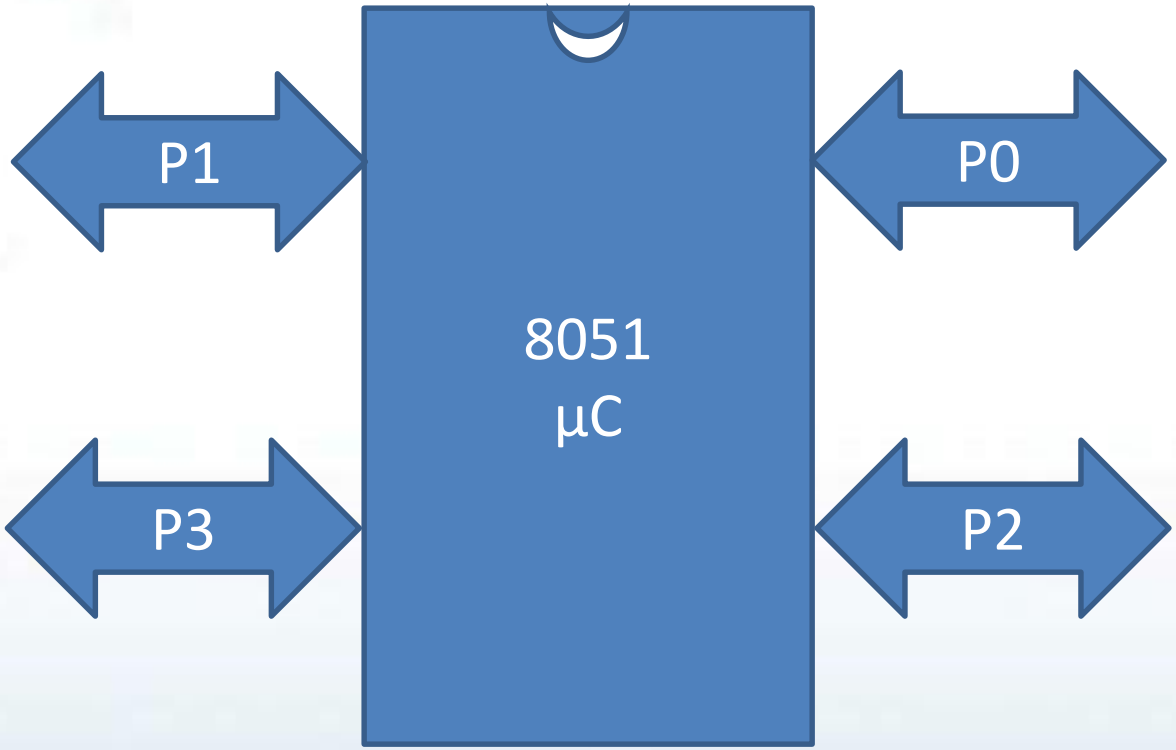
Microcontroller 8051 Family:

Feature	8051	8052	8031
ROM (on-chip program space in 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

Pin Description of the 8051

- 8051 family members (e.g., 8751, 89C51, 89C52, DS89C4x0)
  - Have **40 pins** dedicated for various functions such as I/O, RD, WR, address, data, and interrupts.
  - Come in different packages, such as
    - *DIP(dual in-line package),*
    - *QFP(quad flat package), and*
    - *LLC(leadless chip carrier)*
- Some companies provide a 20-pin version of the 8051 with a reduced number of I/O ports for less demanding applications

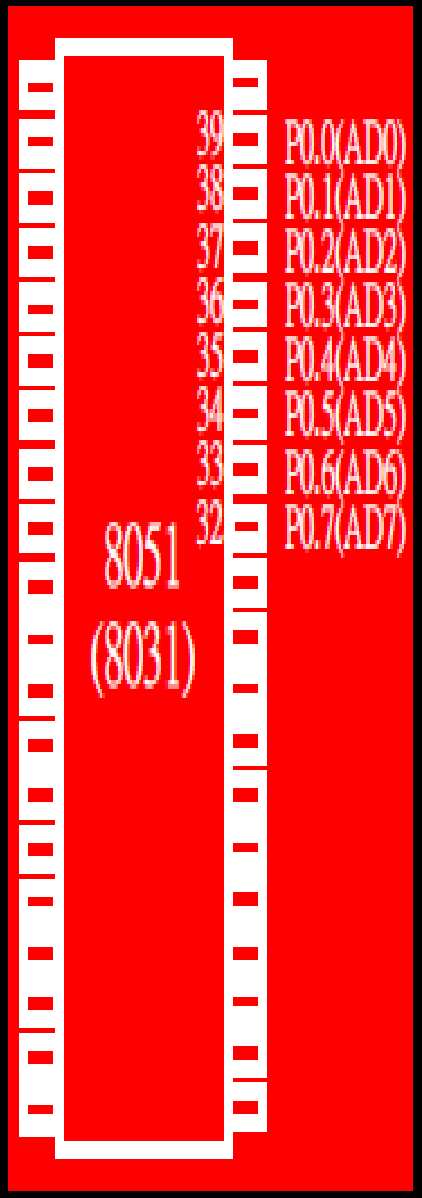
Pin Diagram-8051:



I/O Port Pins

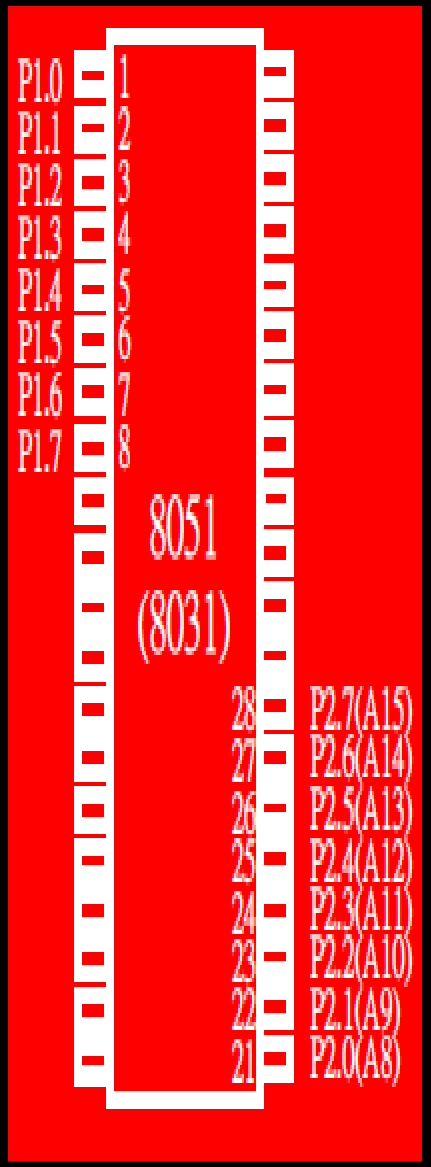
- The four 8-bit I/O ports **P0, P1, P2 and P3** each uses 8 pins.
- All the ports upon RESET are configured as **output**.

Port 0



- Port 0 is **also** designated as **AD0-AD7**.
- When connecting an 8051 to an external memory, port 0 provides both address and data.
- The 8051 multiplexes address and data through port 0 to save pins.
- **ALE** indicates if P0 has address or data.
  - *When ALE=0, it provides data D0-D7*
  - *When ALE=1, it has address A0-A7*

Port 1 and Port 2



- In 8051-based systems **with no external memory connection**:
  - ✓ Both P1 and P2 are used as simple I/O.
- In 8051-based systems **with external memory connections**:
  - ✓ Port 2 must be used along with P0 to provide the 16-bit address for the external memory.
  - ✓ P0 provides the lower 8 bits via A0 – A7.
  - ✓ P2 is used for the upper 8 bits of the 16-bit address, designated as A8 – A15, and it cannot be used for I/O.

Port 3 Alternate Functions

P3 Bit	Pin
P3.0	10
P3.1	11
P3.2	12
P3.3	13
P3.4	14
P3.5	15
P3.6	16
P3.7	17

Port 3 Alternate Functions

P3 Bit	Function	Pin
P3.0	RxD	10
P3.1	TxD	11
P3.2		12
P3.3		13
P3.4		14
P3.5		15
P3.6		16
P3.7		17

Port 3 Alternate Functions

P3 Bit	Function	Pin
P3.0	RxD	10
P3.1	TxD	11
P3.2	$\overline{\text{INT0}}$	12
P3.3	$\overline{\text{INT1}}$	13
P3.4		14
P3.5		15
P3.6		16
P3.7		17

Port 3 Alternate Functions

P3 Bit	Function	Pin
P3.0	RxD	10
P3.1	TxD	11
P3.2	$\overline{\text{INT0}}$	12
P3.3	$\overline{\text{INT1}}$	13
P3.4	T0	14
P3.5	T1	15
P3.6		16
P3.7		17

Port 3 Alternate Functions

P3 Bit	Function	Pin
P3.0	RxD	10
P3.1	TxD	11
P3.2	$\overline{\text{INT0}}$	12
P3.3	$\overline{\text{INT1}}$	13
P3.4	T0	14
P3.5	T1	15
P3.6	$\overline{\text{WR}}$	16
P3.7	$\overline{\text{RD}}$	17

Port 3 Alternate Functions

P3 Bit	Function	Pin
P3.0	RxD	10
P3.1	TxD	11
P3.2	$\overline{\text{INT0}}$	12
P3.3	$\overline{\text{INT1}}$	13
P3.4	T0	14
P3.5	T1	15
P3.6	$\overline{\text{WR}}$	16
P3.7	$\overline{\text{RD}}$	17

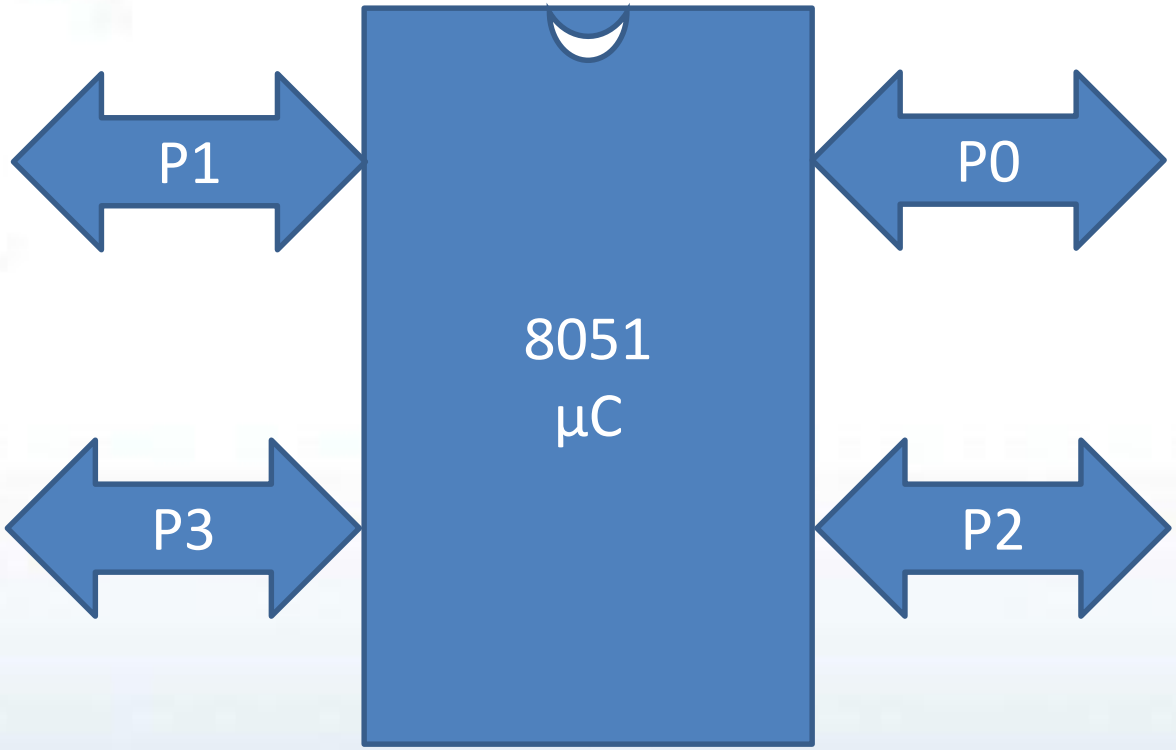
Serial communications

External interrupts

Timers

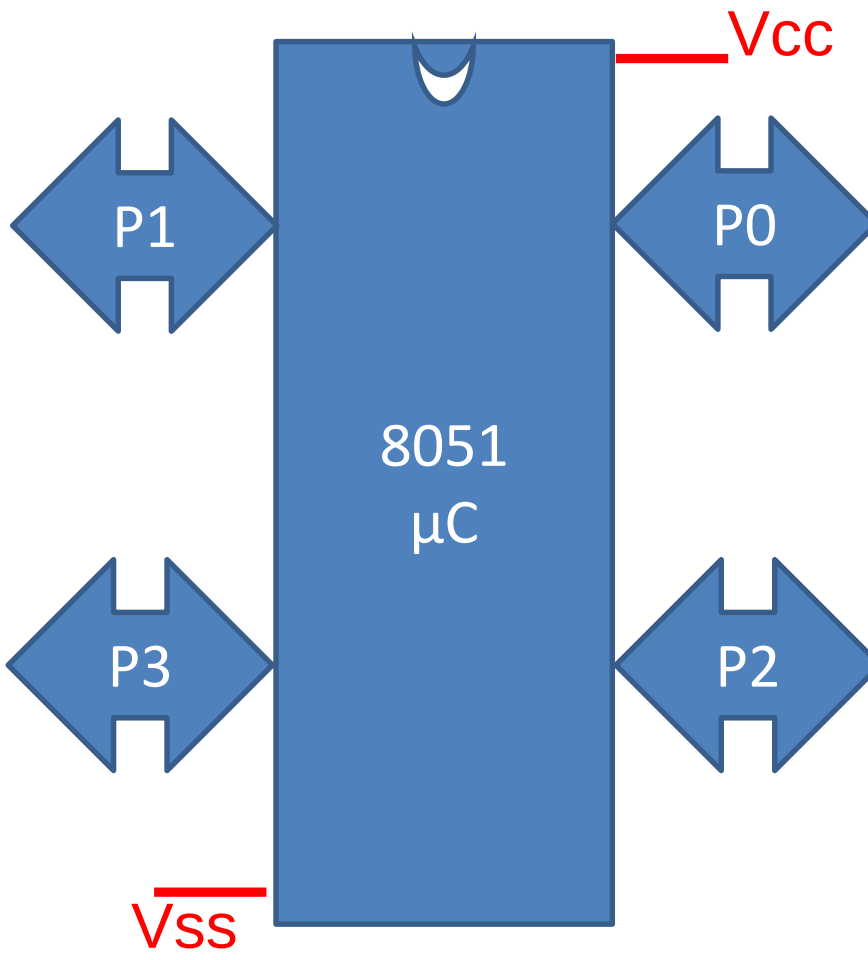
Read/Write signals of external memories

Pin Diagram-8051:



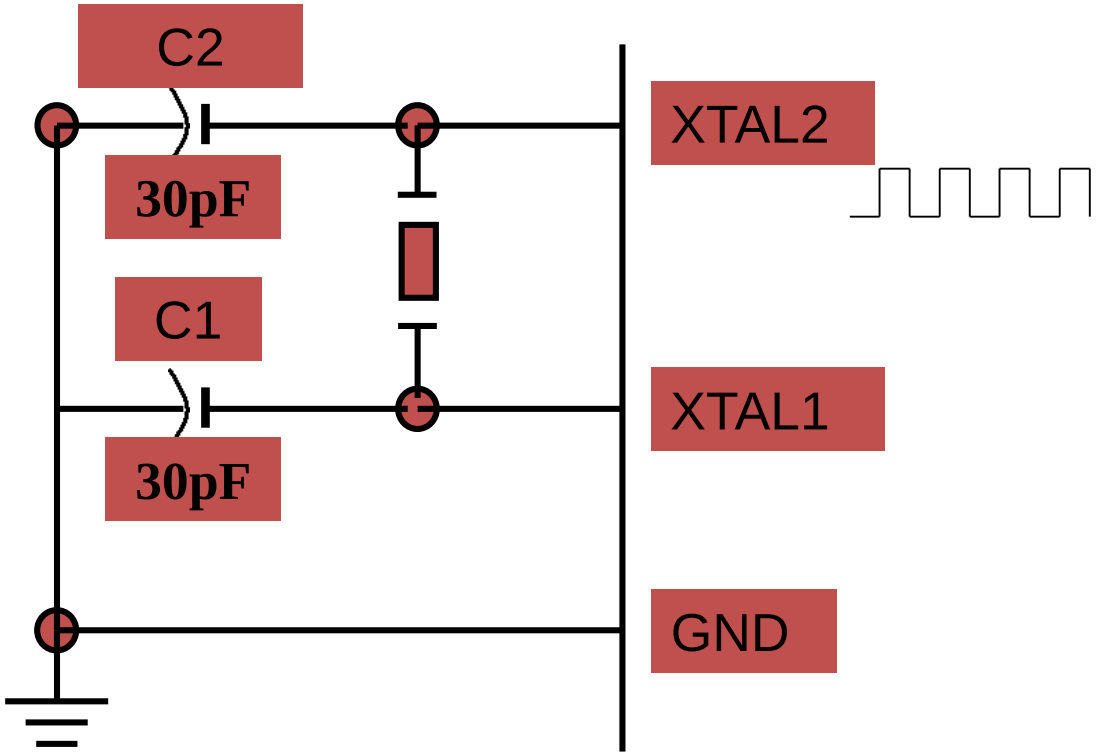
Pin Description of the 8051

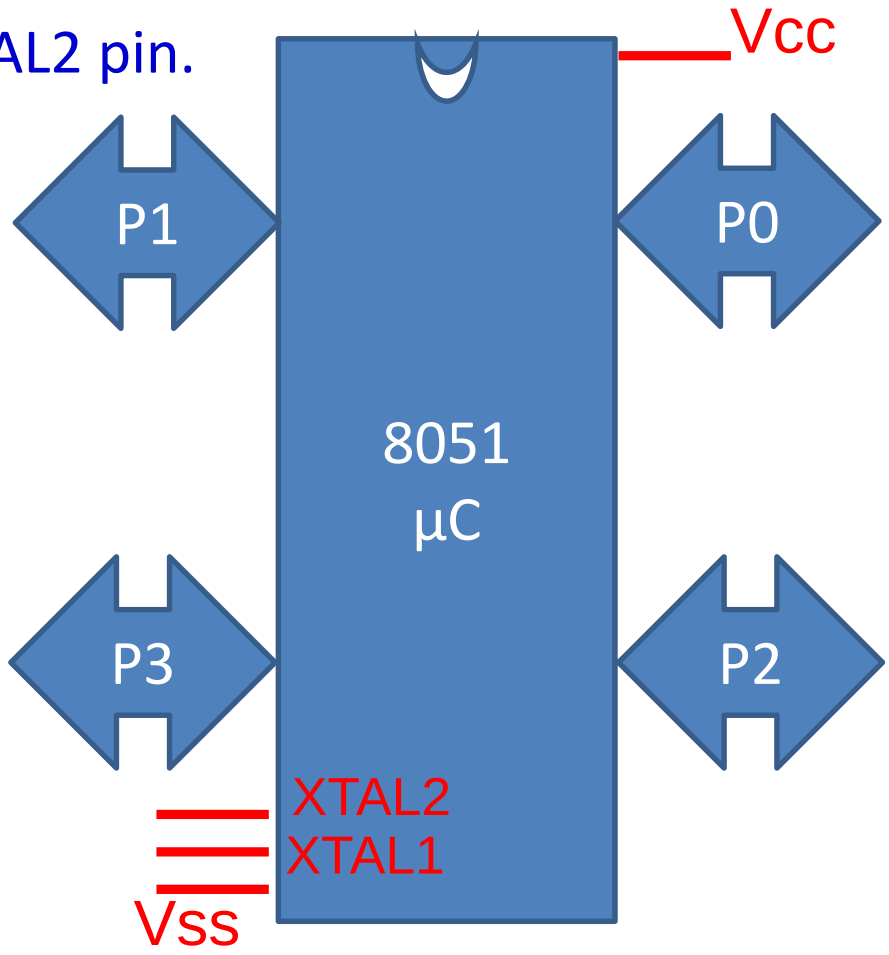
- Vcc （pin 40） :
  - Vcc provides supply voltage
  - The voltage source is +5V
- GND （pin 20） : ground



Pin Description of the 8051

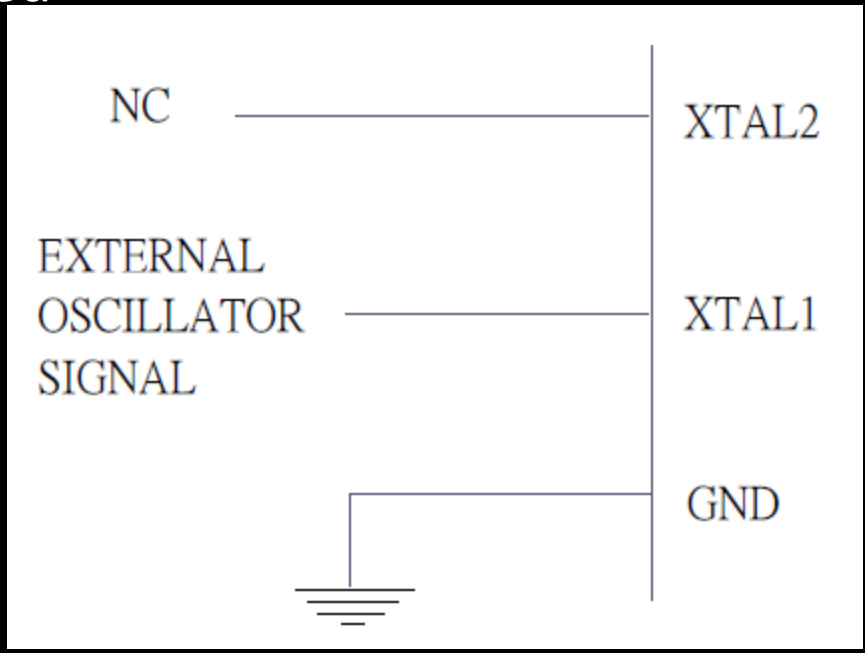
- Using a quartz crystal oscillator
- We can observe the frequency on the XTAL2 pin.





XTAL1 and XTAL2 .....

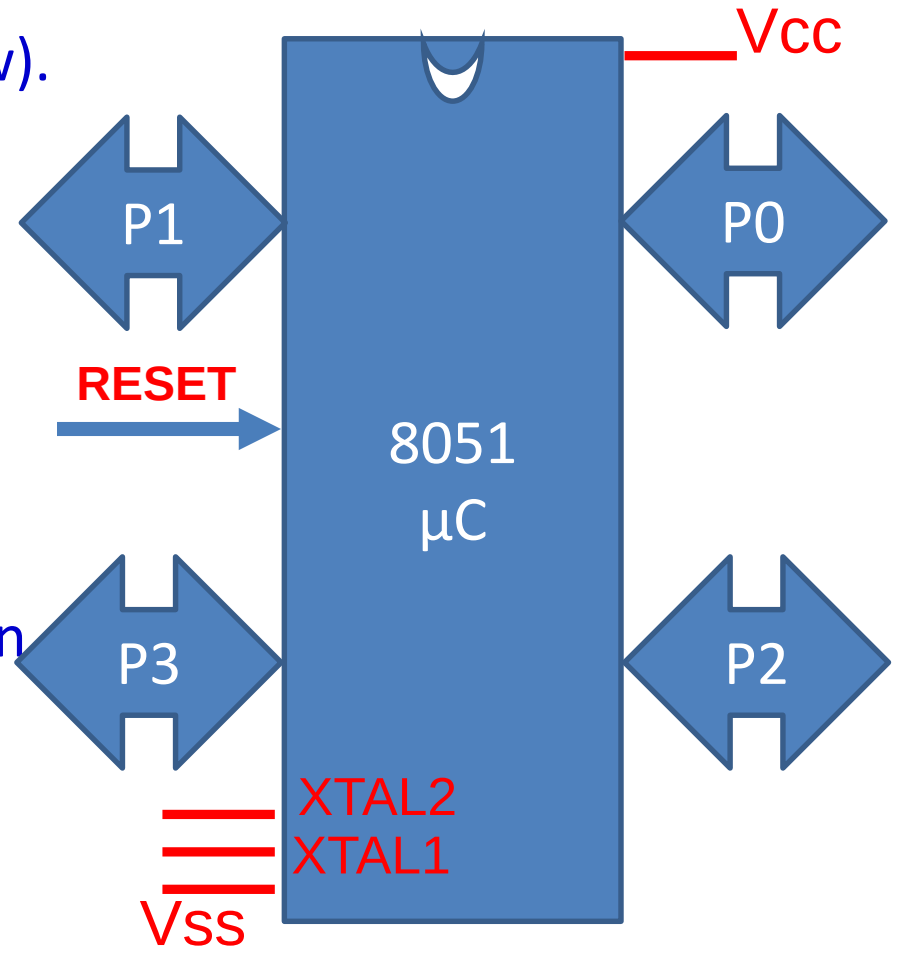
- If you use a frequency source other than a crystal oscillator, such as a TTL oscillator:
  - It will be connected to XTAL1
  - XTAL2 is left unconnected



Pin Description of the 8051

RST（pin 9）：reset

- Input pin and is active high (normally low).
- The high pulse must be high at least 2 machine cycles.
- It is a power-on reset.
- Upon applying a high pulse to RST, the microcontroller will reset and all values in registers will be lost.
- Reset values of some 8051 registers



RST

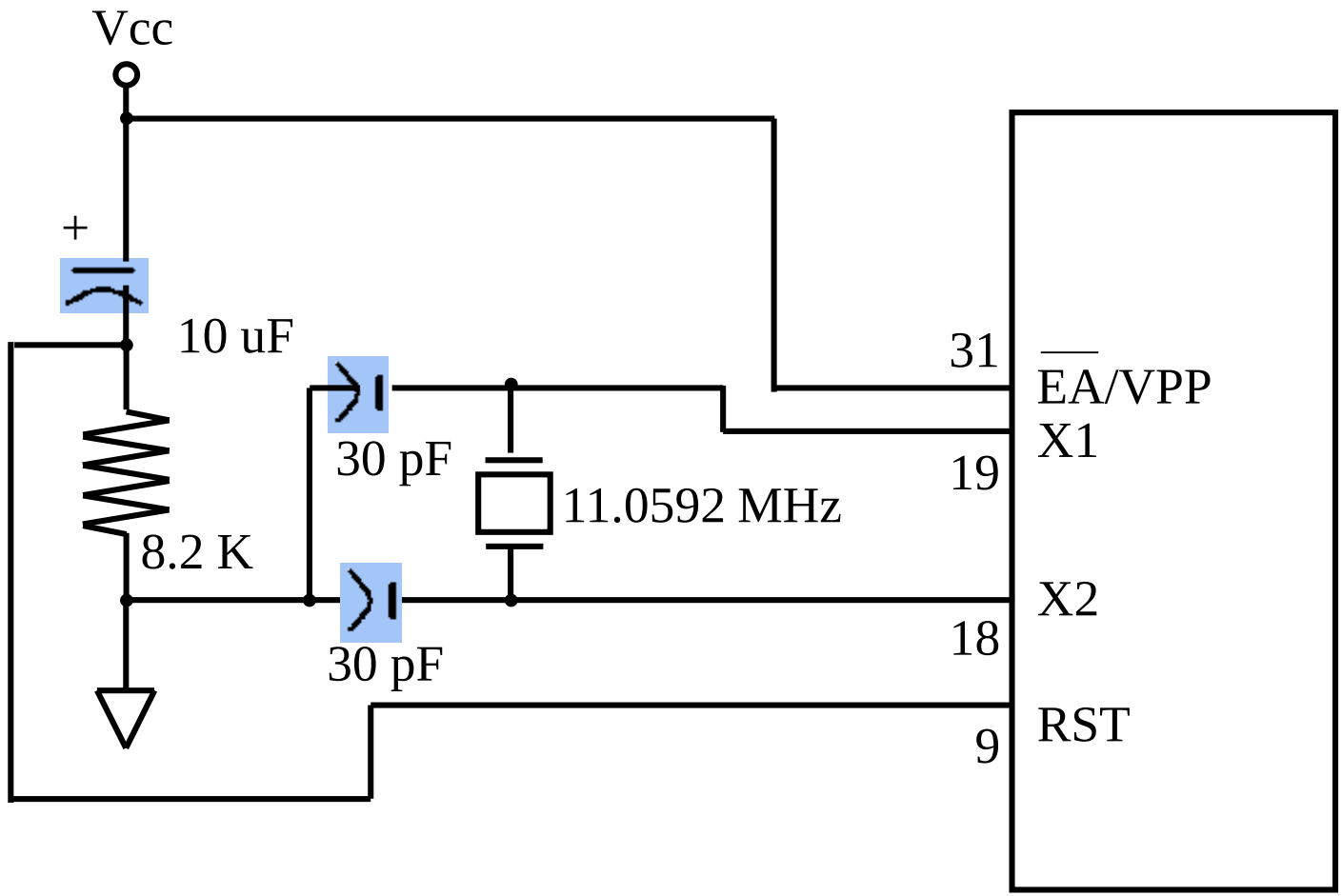
- Activating a power-on reset will cause all values in the registers to be lost

RESET value of some 8051 registers

we must place the first line of source code in ROM location 0

Register	Reset Value
PC	0000
DPTR	0000
ACC	00
PSW	00
SP	07
B	00
P0-P3	FF

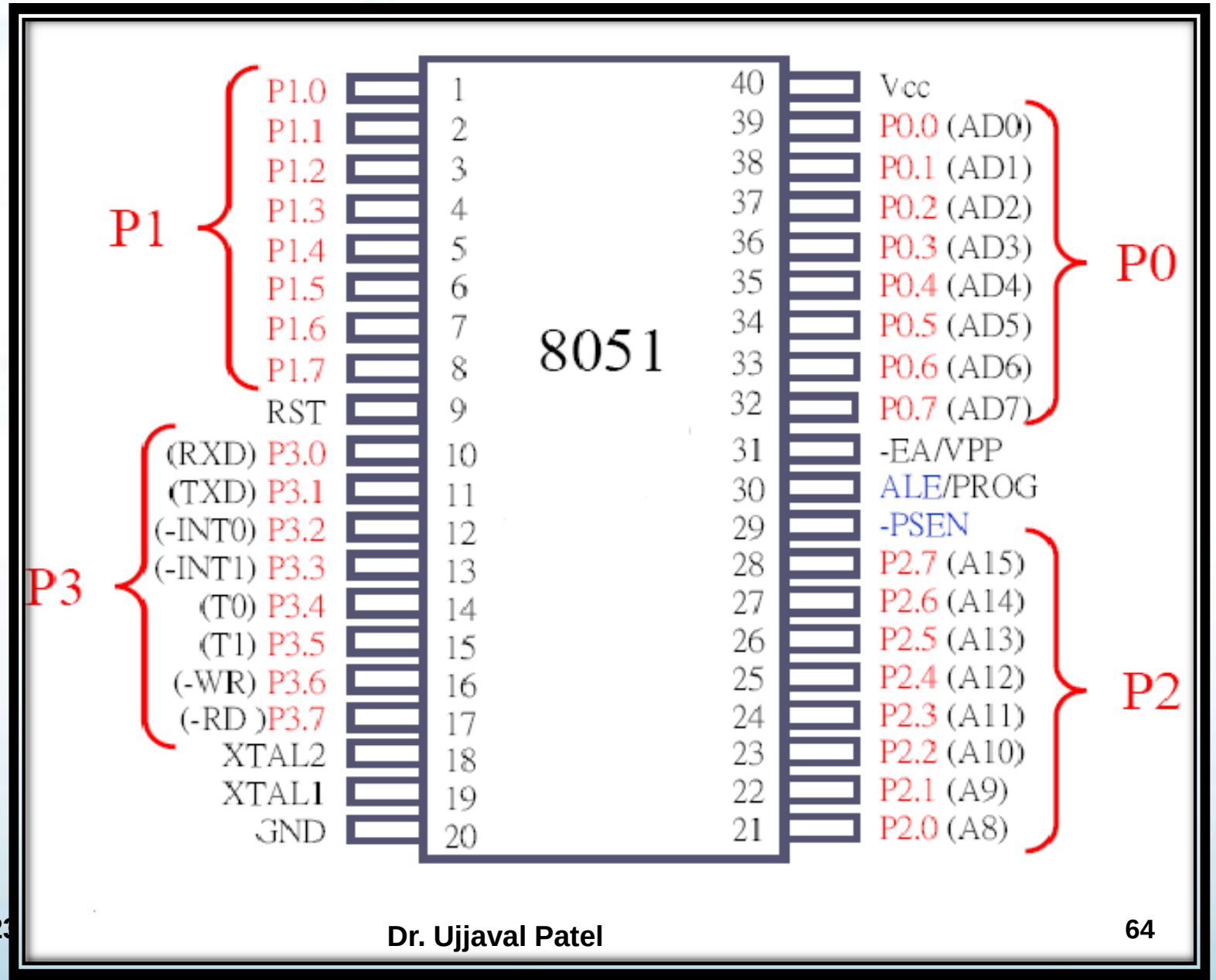
Pin Description of the 8051



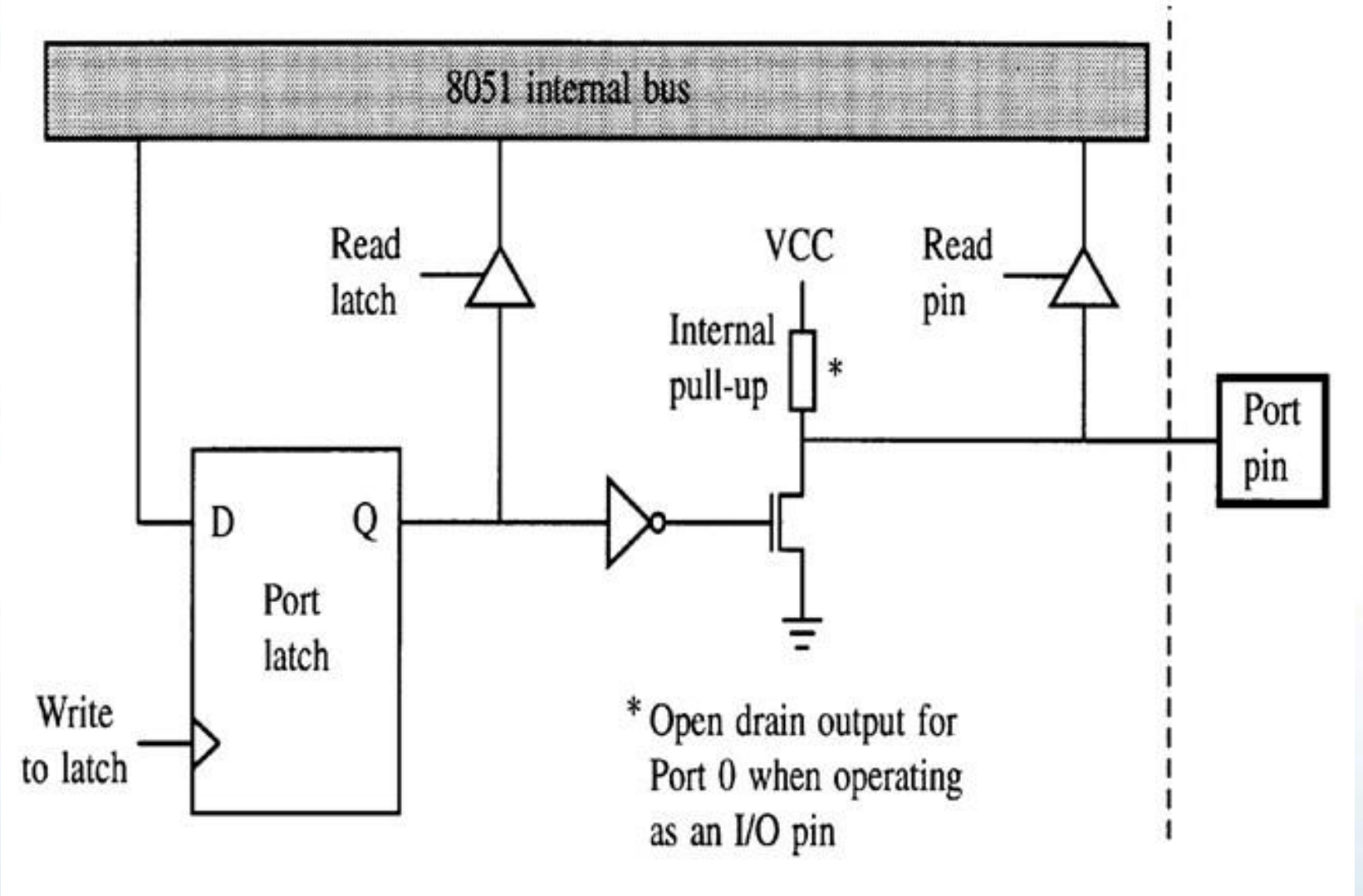
EA’

- EA’, **“external access”**, is an input pin and must be connected to Vcc or GND
- The 8051 family members all come with on-chip ROM to store programs and also have an external code and data memory.
- Normally EA pin is connected to Vcc (Internal Access)
- EA pin must be connected to GND to indicate that the code or data is stored externally.

Pin Diagram-8051:

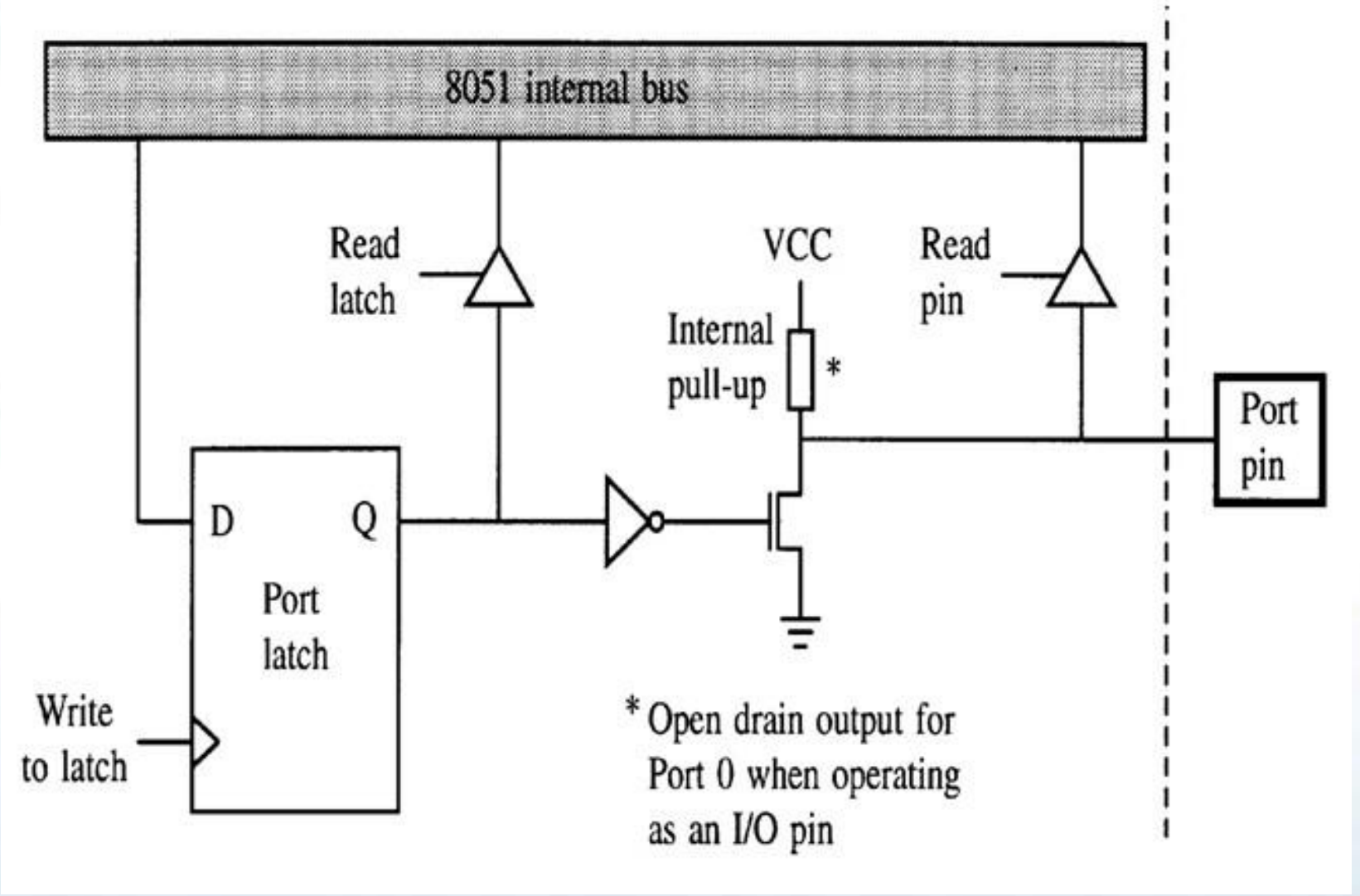


Port Pin configuration:



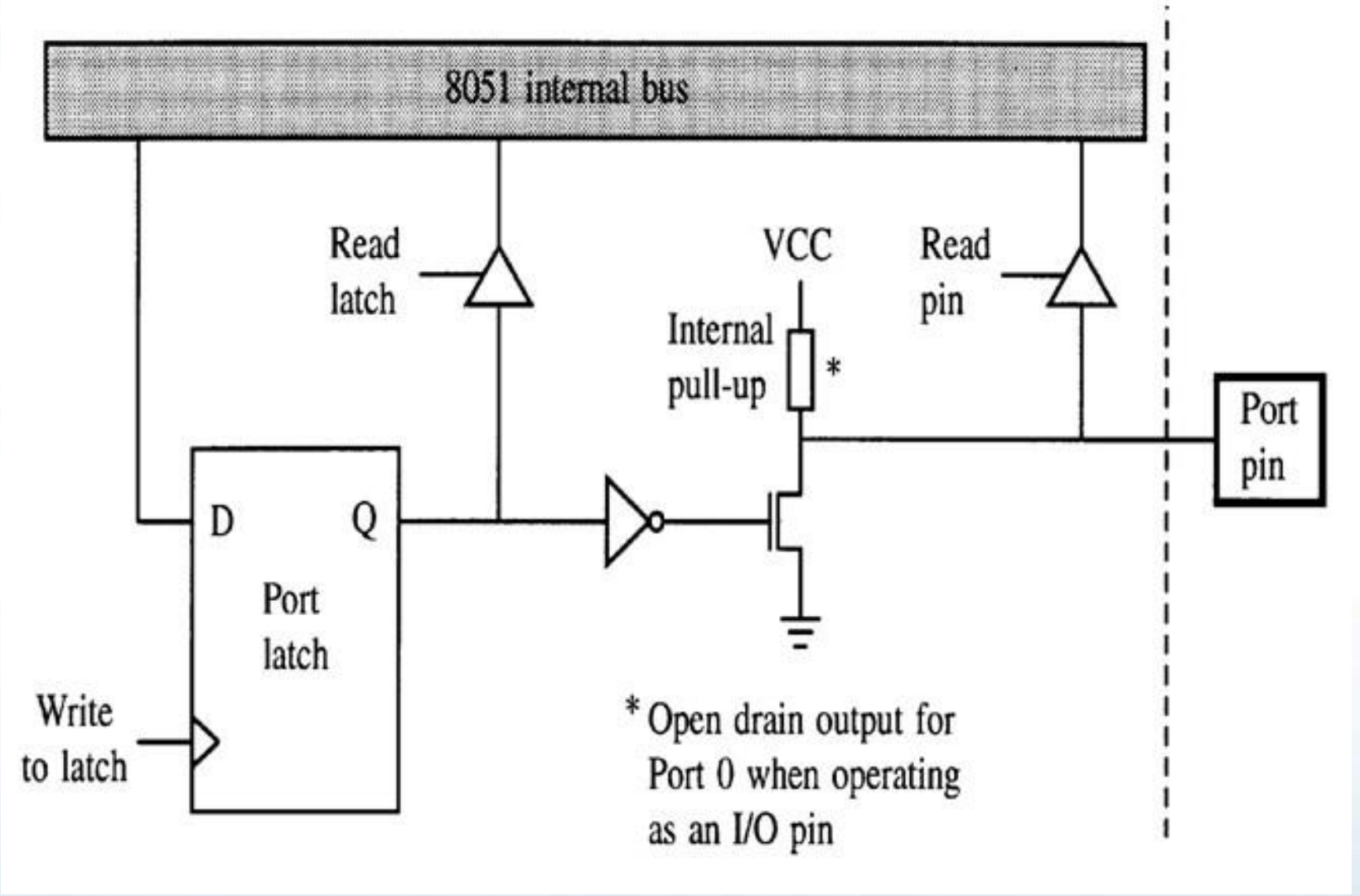
Output: SETB P1.0

Port Pin configuration:



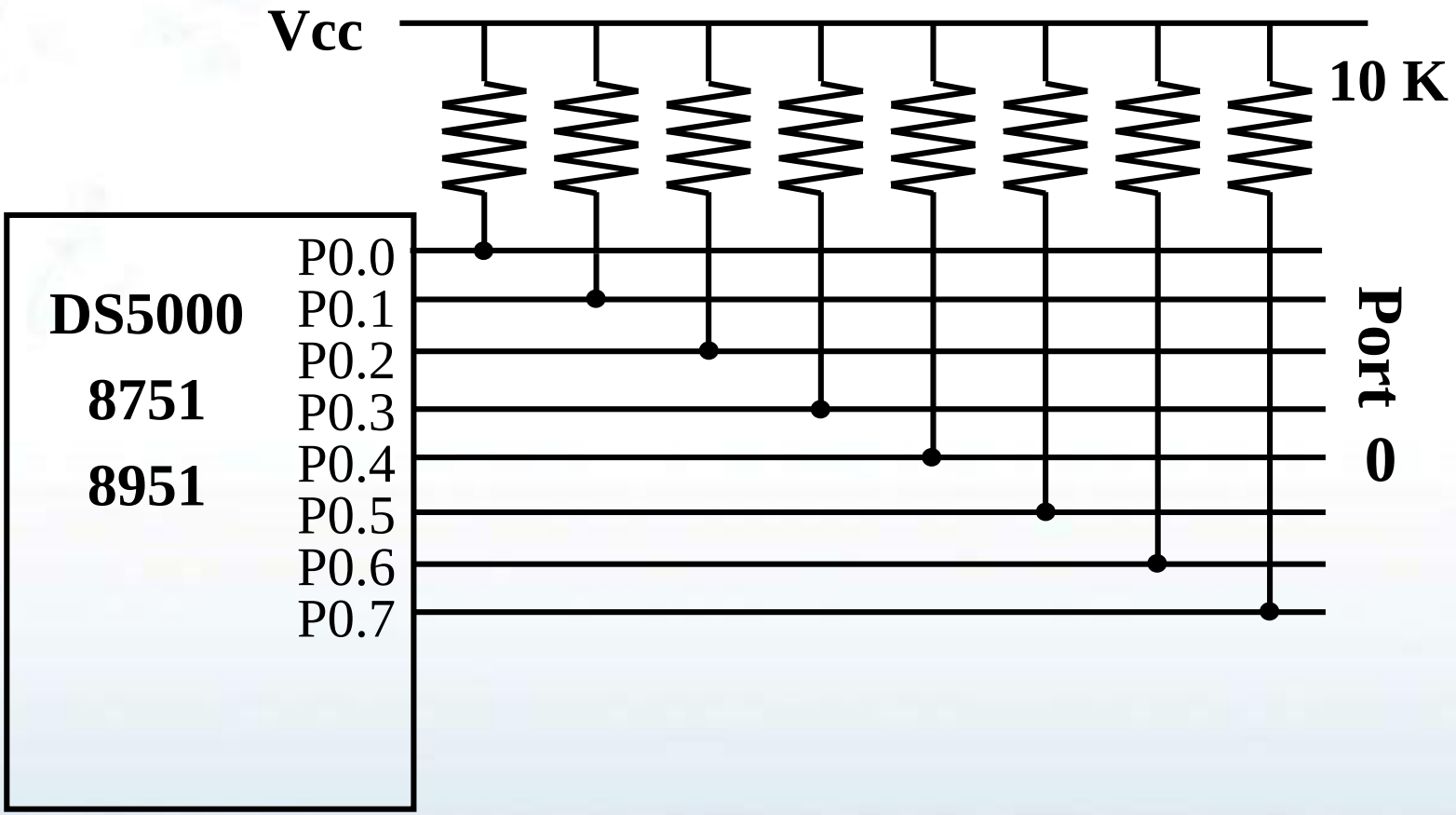
Output: CLR P1.0

Port Pin configuration:



Input: MOV C,P1.0

Port 0 with Pull-Up Resistors



PSEN' and ALE

- PSEN, **"program store enable"**, is an output pin
- This pin is connected to the OE pin of the external memory.
- For External Code Memory, PSEN' = 0
- For External Data Memory, PSEN' = 1
- ALE pin is used for demultiplexing the address and data.

Pin Description Summary

PIN	TYPE	NAME AND FUNCTION
Vss	**I**	**Ground:** 0 V reference.
Vcc	**I**	**Power Supply:** This is the power supply voltage for normal, idle, and power-down operation.
P0.0 - P0.7	**I/O**	**Port 0:** Port 0 is an open-drain, bi-directional I/O port. Port 0 is also the multiplexed low-order address and data bus during accesses to external program and data memory.
P1.0 - P1.7	**I/O**	**Port 1:** Port I is an 8-bit bi-directional I/O port.
P2.0 - P2.7	**I/O**	**Port 2:** Port 2 is an 8-bit bidirectional I/O. Port 2 emits the high order address byte during fetches from external program memory and during accesses to external data memory that use 16 bit addresses.
P3.0 - P3.7	**I/O**	**Port 3:** Port 3 is an 8 bit bidirectional I/O port. Port 3 also serves special features as explained.

Pin Description Summary

PIN	TYPE	NAME AND FUNCTION
RST	**I**	**Reset:** A high on this pin for two machine cycles while the oscillator is running, resets the device.
ALE	**O**	**Address Latch Enable:** Output pulse for latching the low byte of the address during an access to external memory.
PSEN*	**O**	**Program Store Enable:** The read strobe to external program memory. When executing code from the external program memory, PSEN* is activated twice each machine cycle, except that two PSEN* activations are skipped during each access to external data memory.
EA*/VPP	**I**	**External Access Enable/Programming Supply Voltage:** EA* must be externally held low to enable the device to fetch code from external program memory locations. If EA* Is held high, the device executes from internal program memory. This pin also receives the programming supply voltage Vpp during Flash programming. (applies for 89c5x MCU's)

National Forensic Sciences University
Knowledge | Wisdom | Fulfilment
An Institution of National Importance
(Ministry of Home Affairs, Government of India)

Microprocessor & Microcontrollers

(CTBTCSE SIV P3)

Dr. Ujjaval Patel

Assistant Professor

School of Cyber Security & Digital Forensics

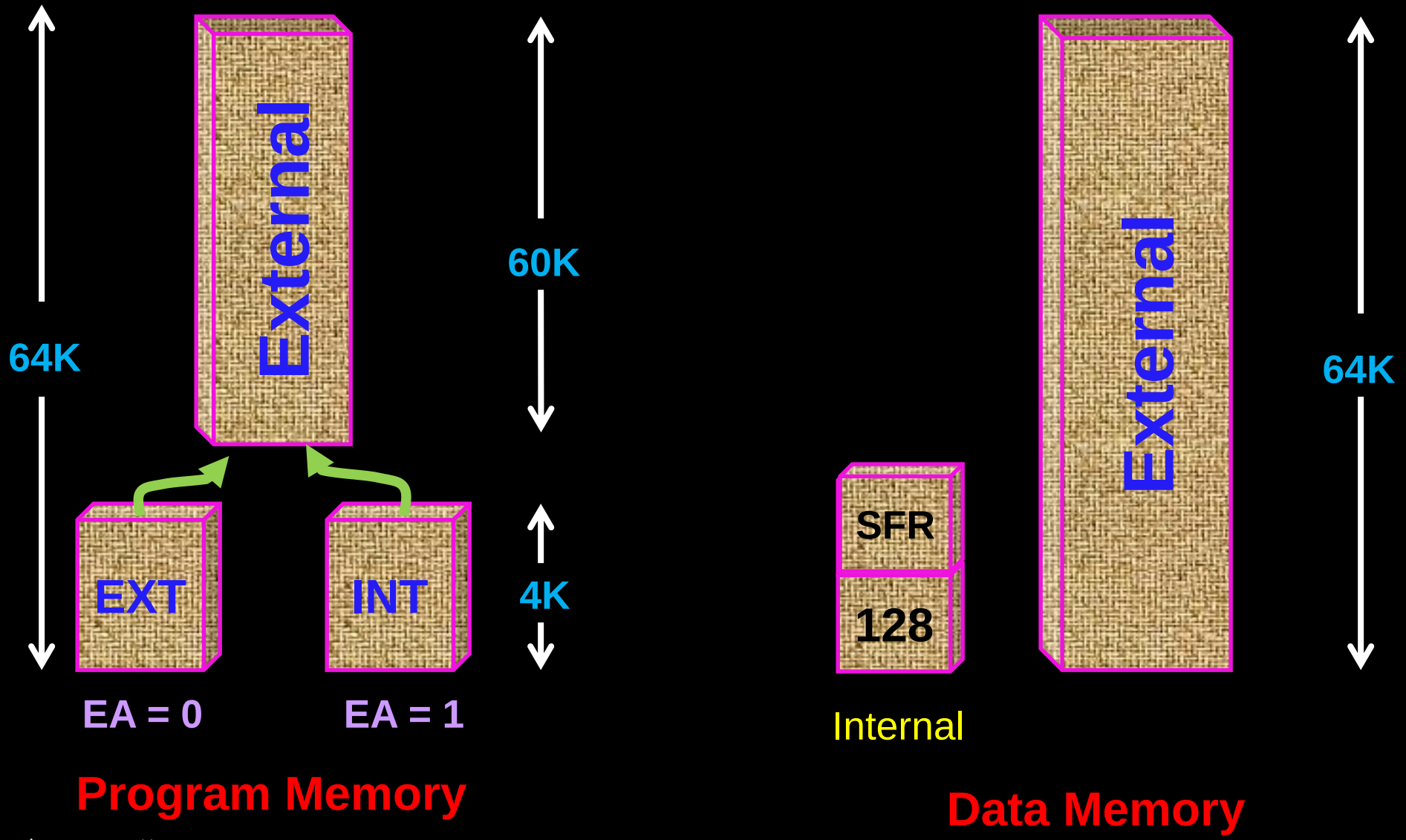
(M: +91-987 987 97 46, E-mail: ujjaval.patel@nfsu.ac.in)

Important Features of 8051

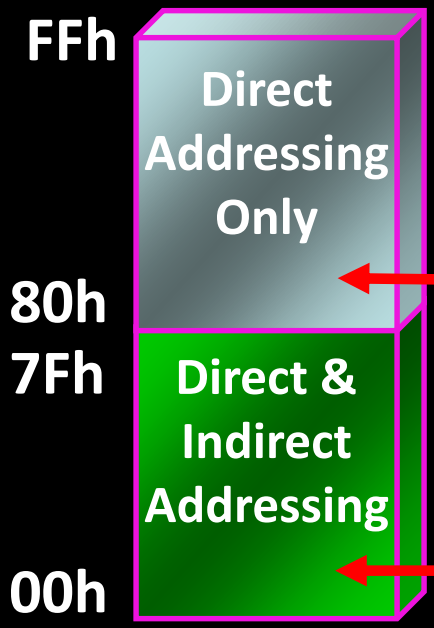
- I/O ports: 4
- RAM: 128 bytes
- ROM: 4KB
- Timers: 2
- Serial Com. Port: 1
- Interrupts: 6

8051 Memory Space

8051 Memory Structure



Internal RAM Structure



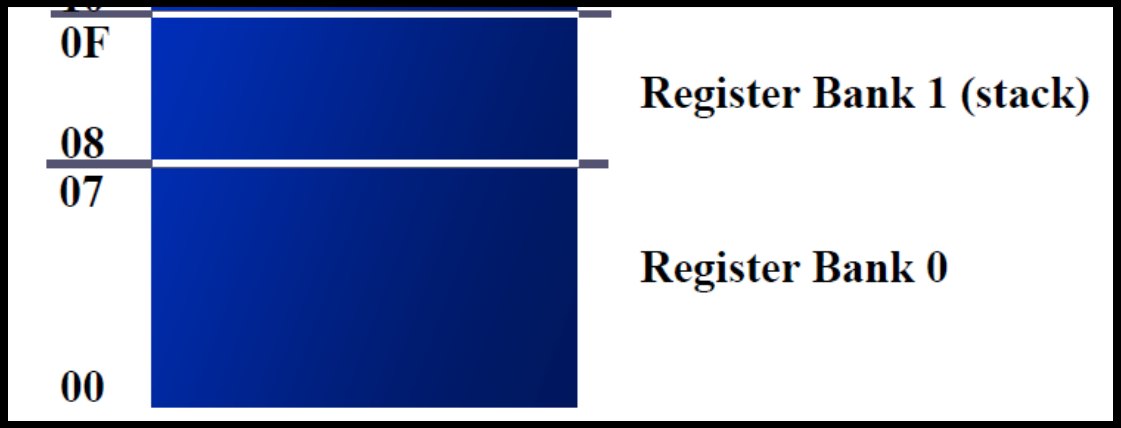
SFR [Special Function Registers]

128 Byte Internal RAM

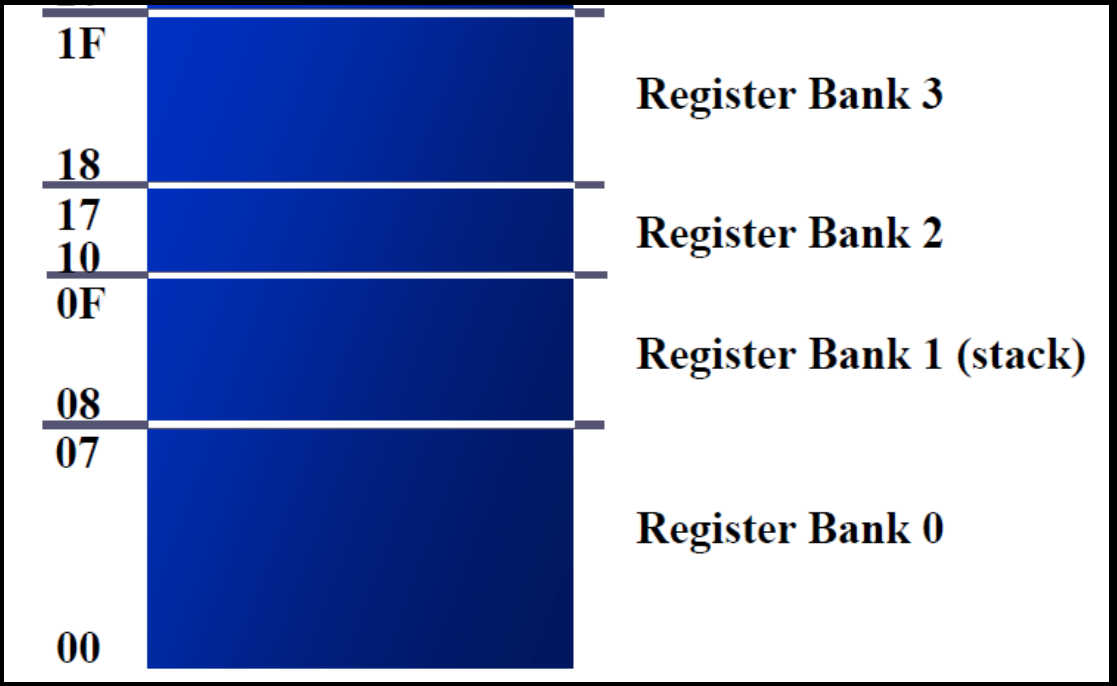
128 Byte RAM

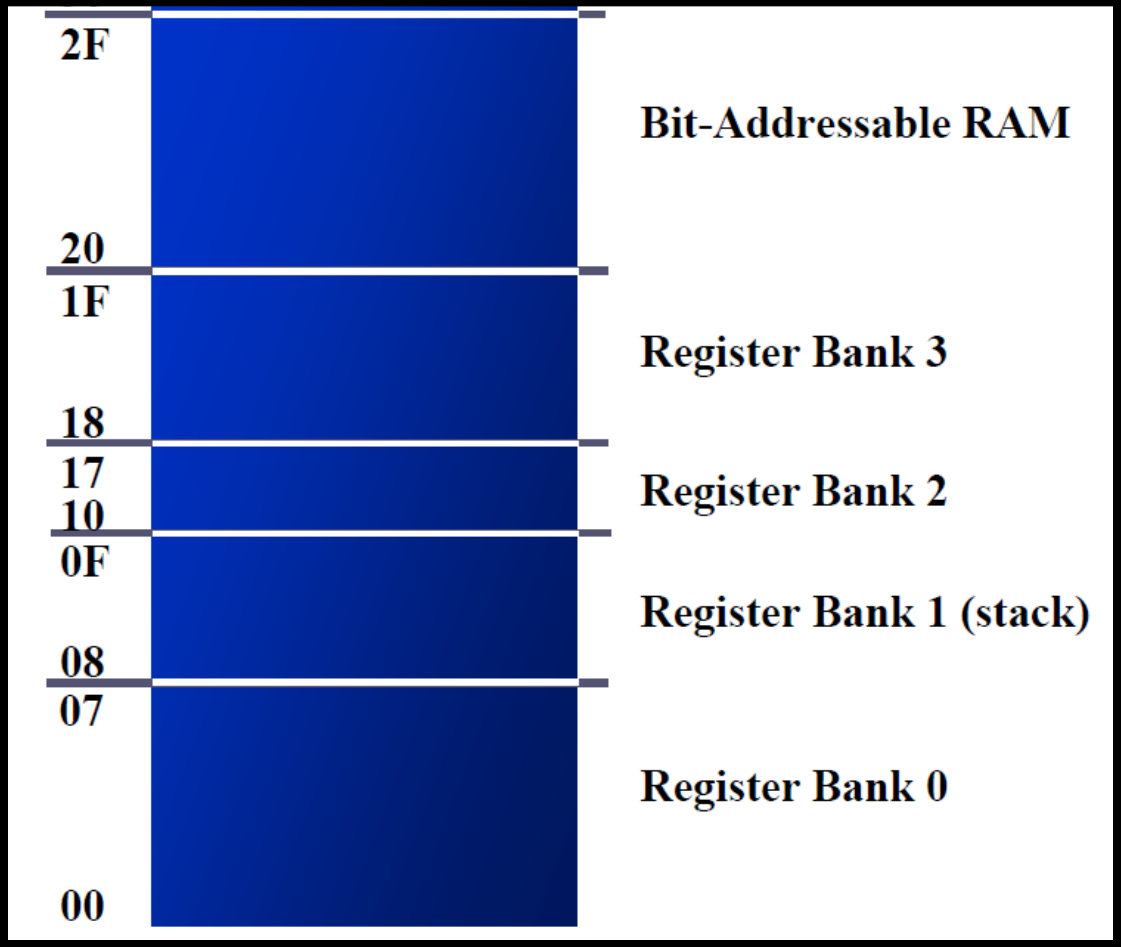
- There are 128 bytes of RAM in the 8051.
  - Assigned addresses 00 to 7FH
- The **128 bytes are divided into 3 different groups** as follows:
  1. A total of **32 bytes** from locations 00 to 1F hex are set aside for ***register banks*** and the ***stack.***
  2. A total of **16 bytes** from locations 20H to 2FH are set aside for ***bit-addressable*** read/write memory.
  3. A total of **80 bytes** from locations 30H to 7FH are used for read and write storage, called ***scratch pad.***


07
00
Register Bank 0

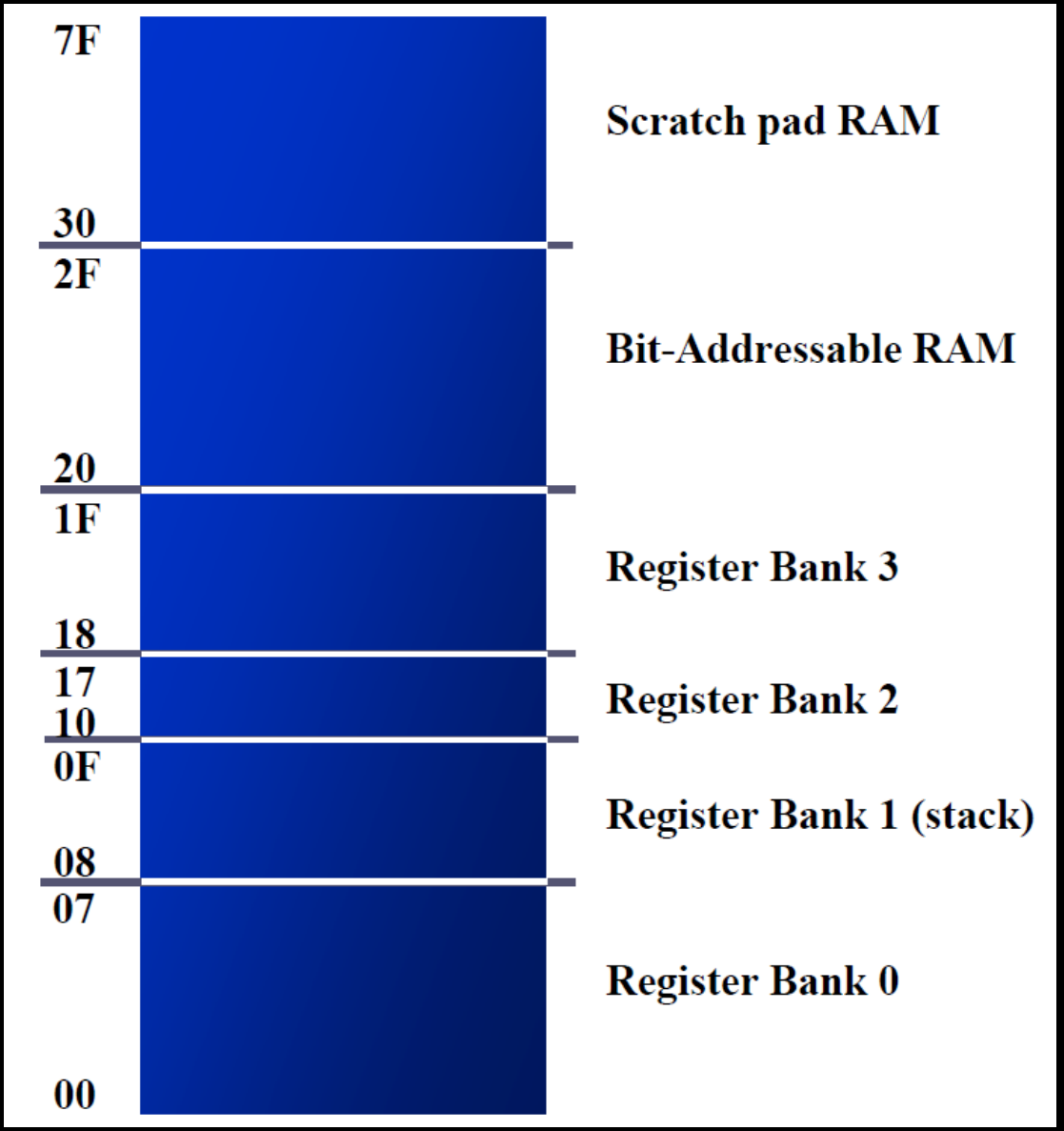

0F
08
07
00
Register Bank 1 (stack)
Register Bank 0


17
10
0F
08
07
00
Register Bank 2
Register Bank 1 (stack)
Register Bank 0

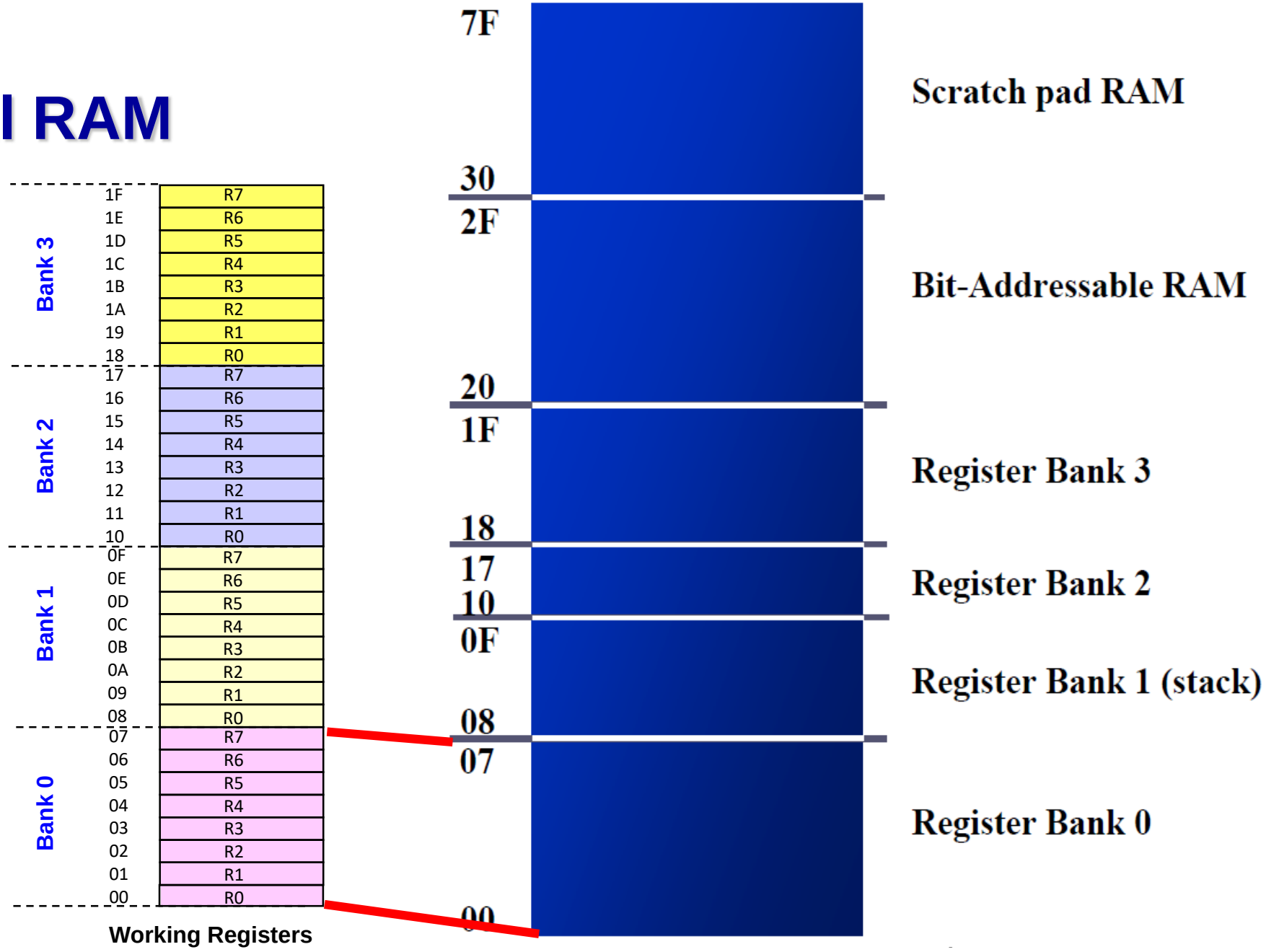

1F
18
17
10
0F
08
07
00
Register Bank 3
Register Bank 2
Register Bank 1 (stack)
Register Bank 0


2F
20
1F
18
17
10
0F
08
07
00
Bit-Addressable RAM
Register Bank 3
Register Bank 2
Register Bank 1 (stack)
Register Bank 0

8051 RAM with addresses

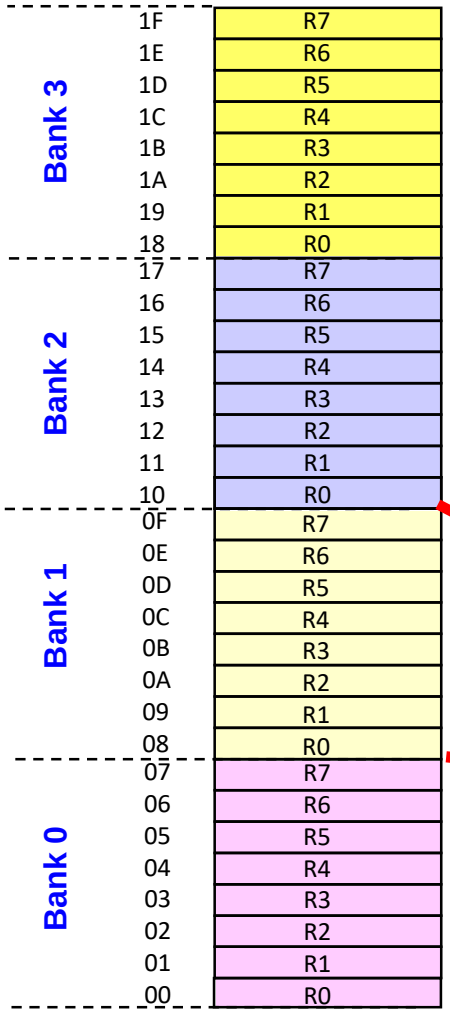


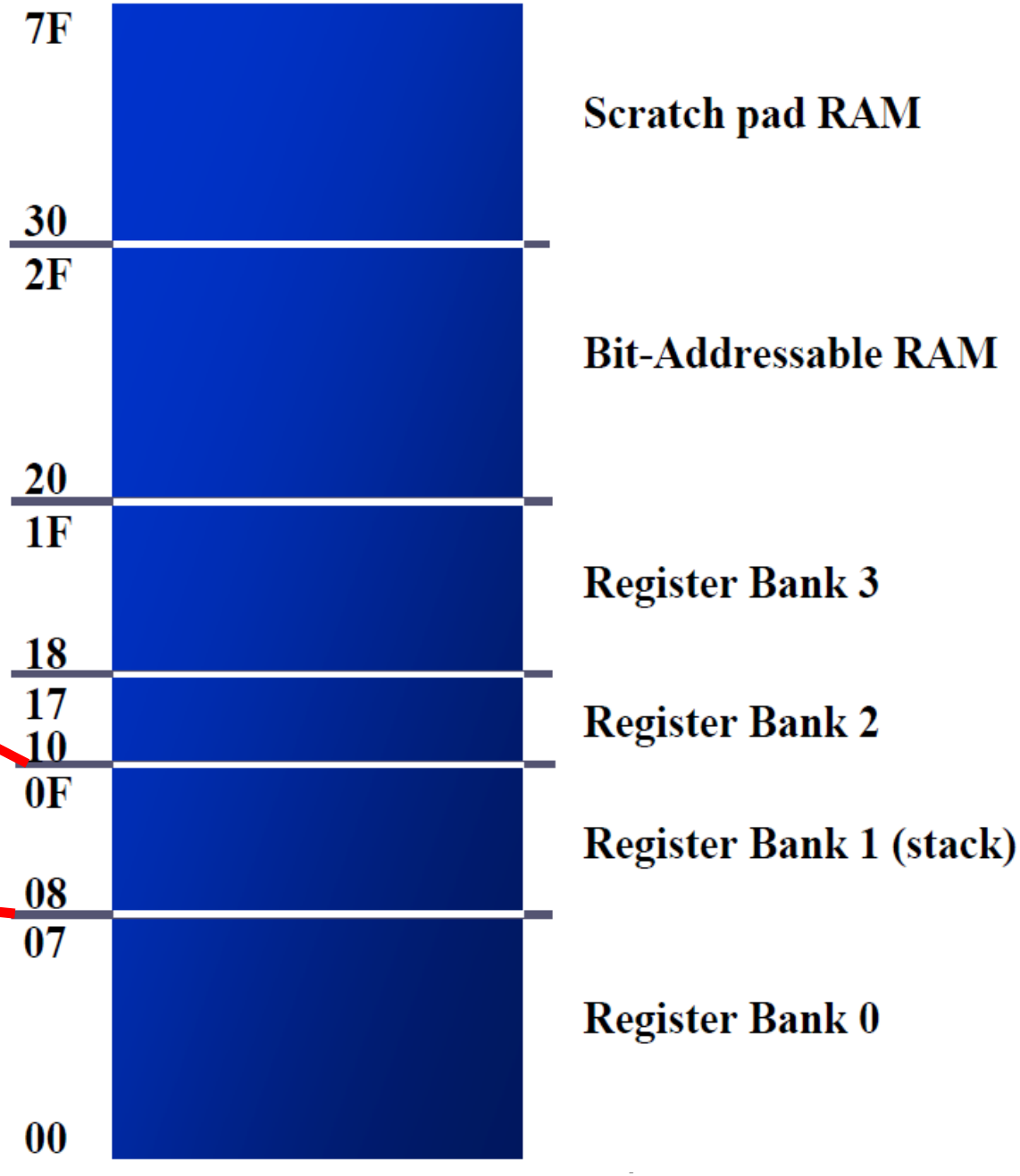
8051
Internal RAM



8051

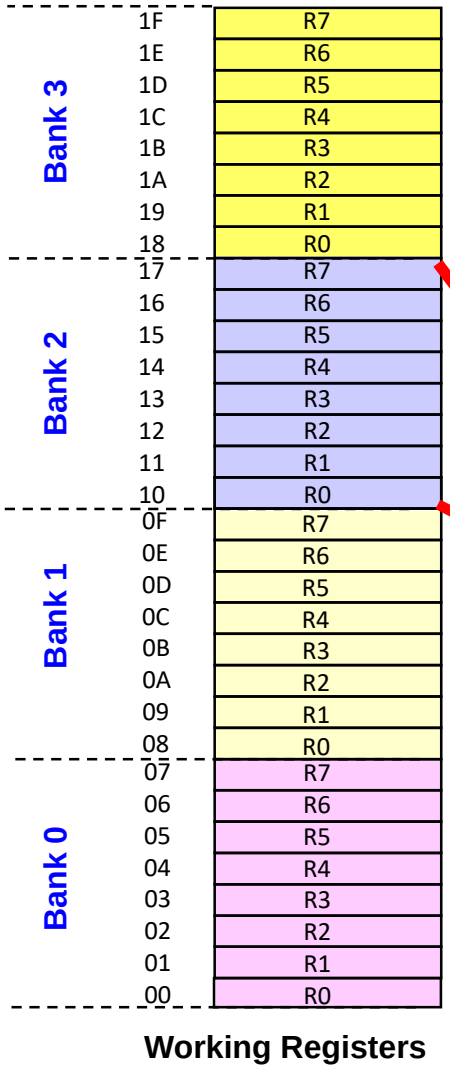
Internal RAM

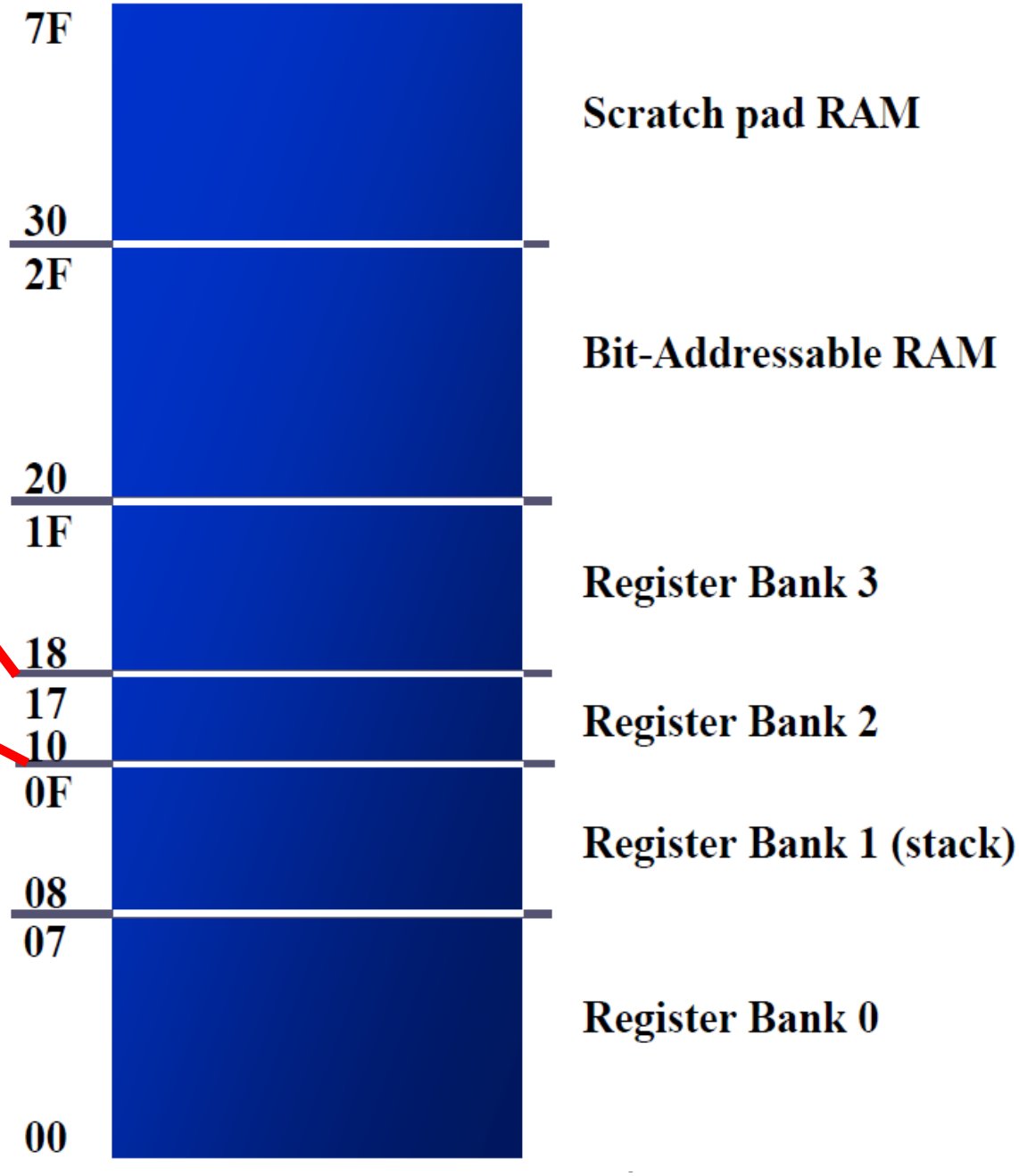




8051

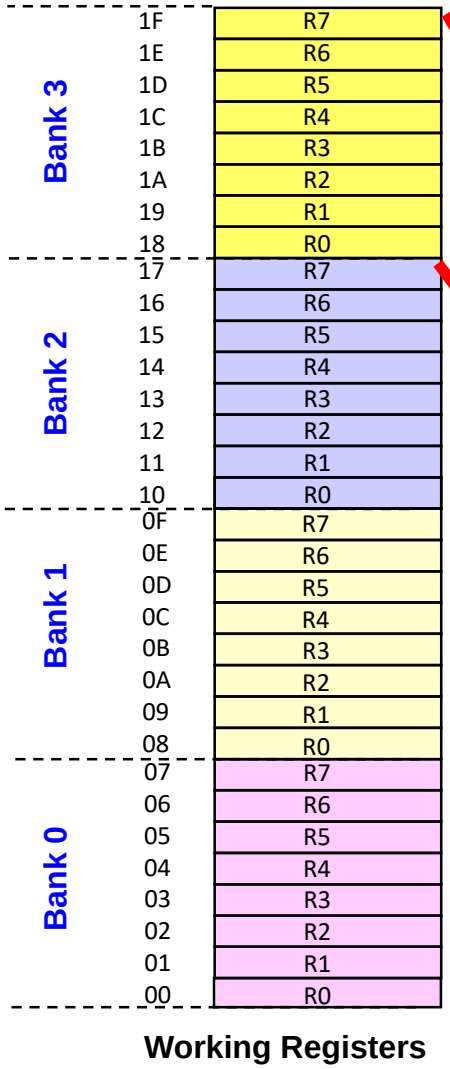
Internal RAM

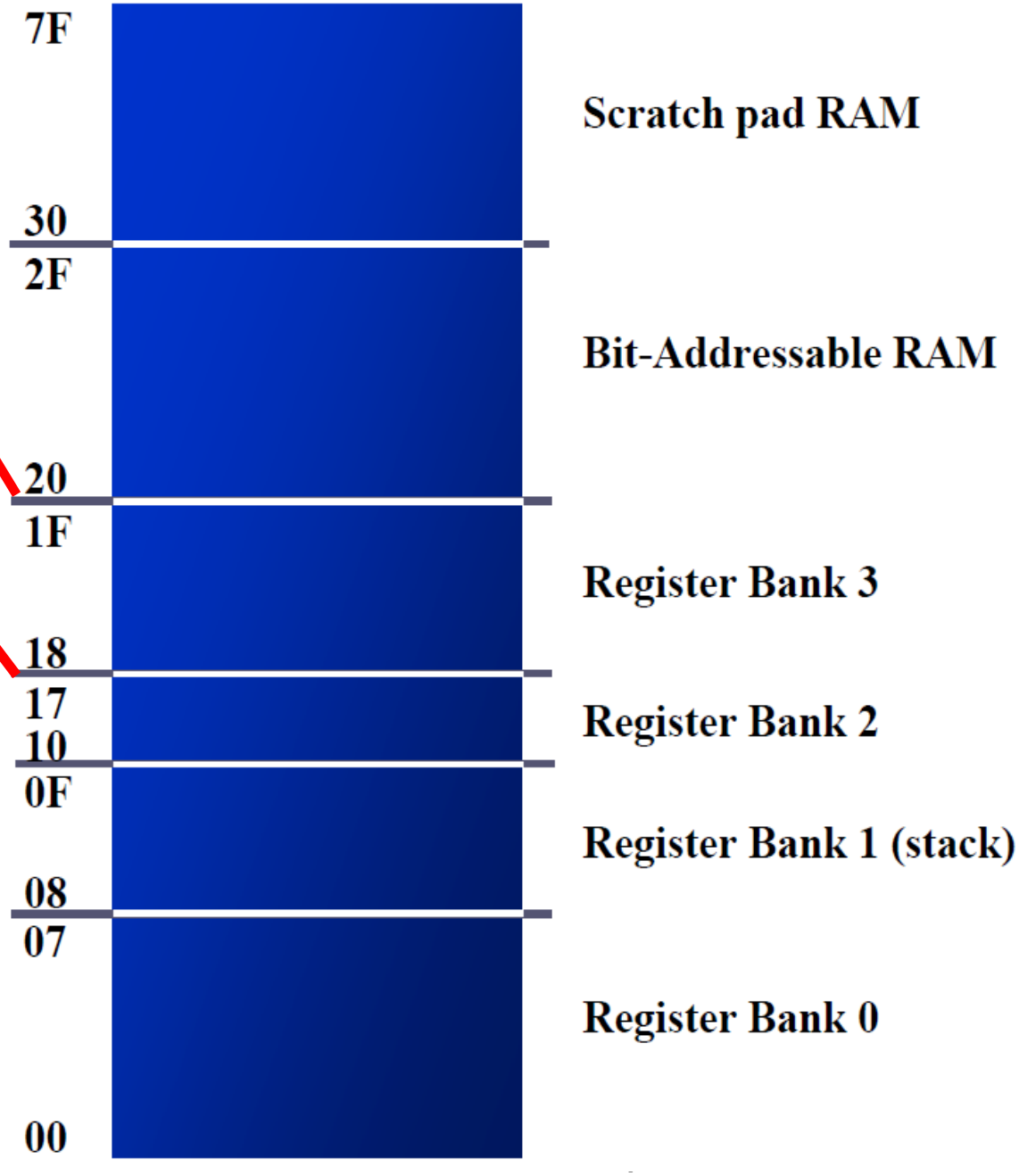




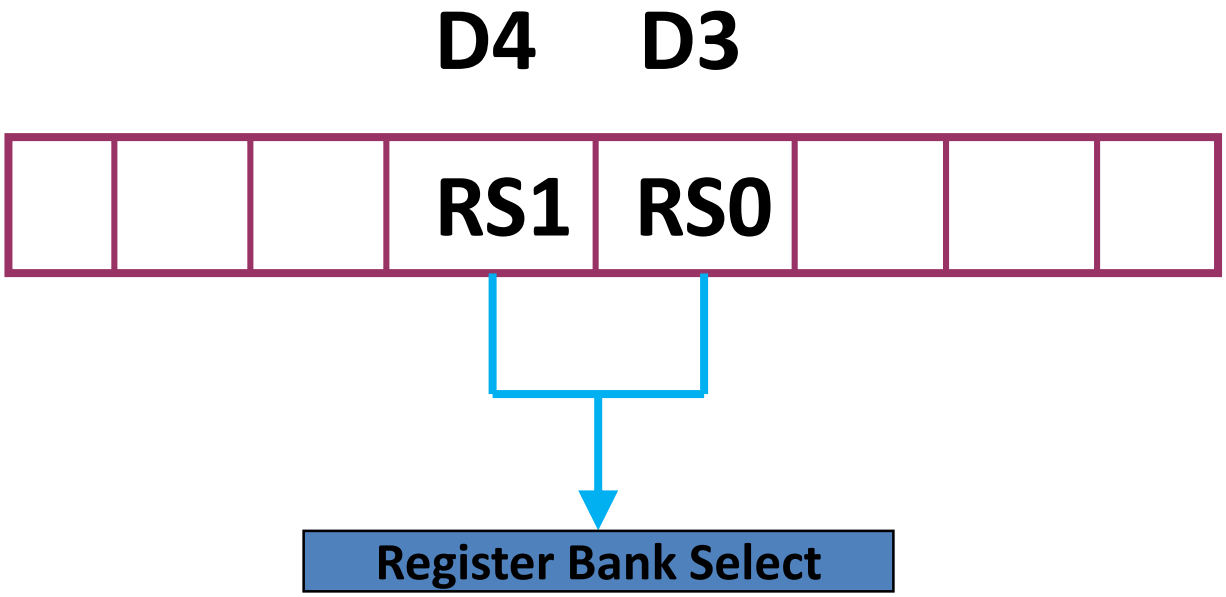
8051

Internal RAM



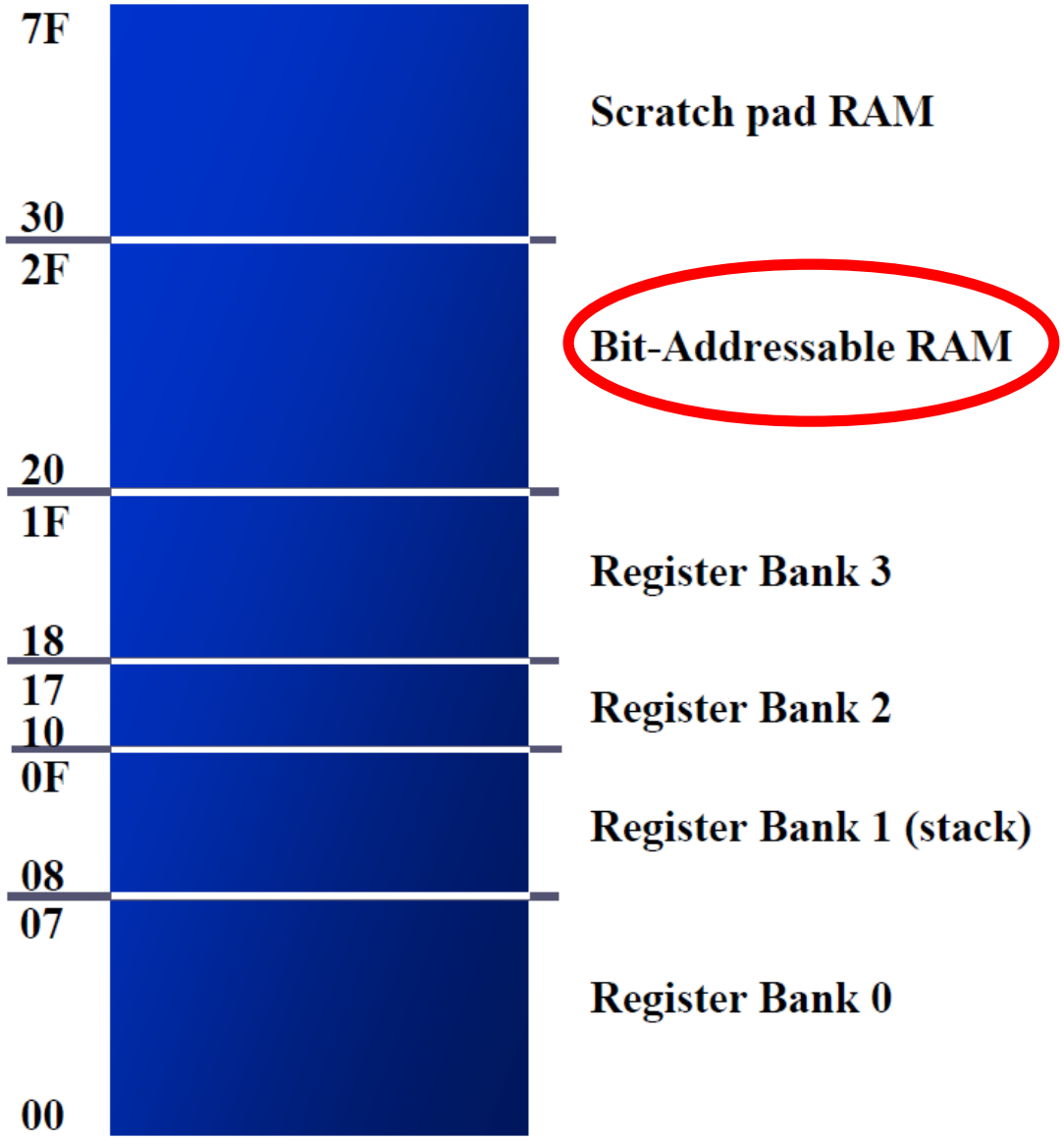


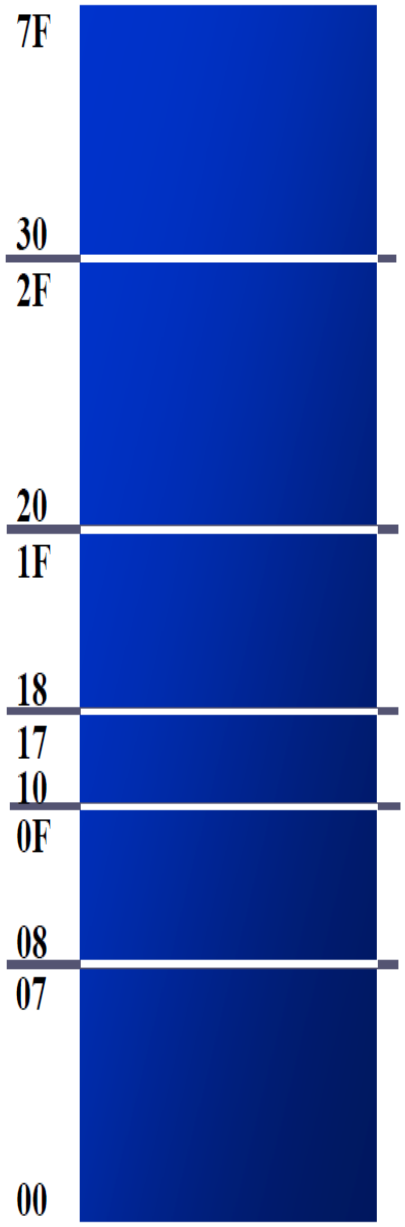
Program Status Word [PSW]



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

8051 RAM with addresses





Bit-addressable locations

Byte address

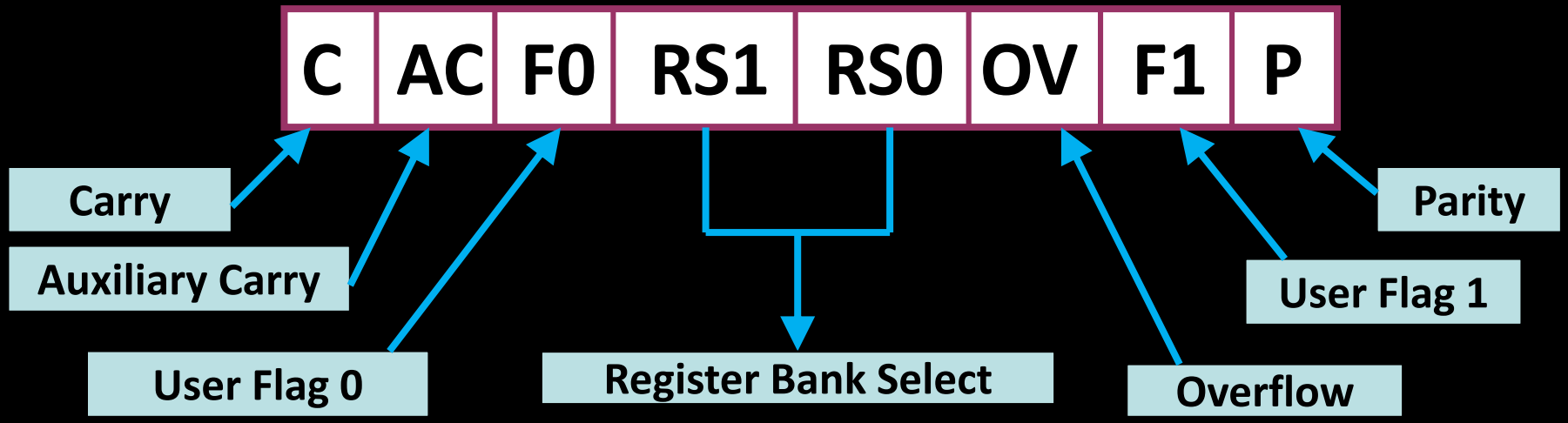
7F	General purpose RAM							
30								
2F	7F	7E	7D	7C	7B	7A	79	78
2E	77	76	75	74	73	72	71	70
2D	6F	6E	6D	6C	6B	6A	69	68
2C	67	66	65	64	63	62	61	60
2B	5F	5E	5D	5C	5B	5A	59	58
2A	57	56	55	54	53	52	51	50
29	4F	4E	4D	4C	4B	4A	49	48
28	47	46	45	44	43	42	41	40
27	3F	3E	3D	3C	3B	3A	39	38
26	37	36	35	34	33	32	31	30
25	2F	2E	2D	2C	2B	2A	29	28
24	27	26	25	24	23	22	21	20
23	1F	1E	1D	1C	1B	1A	19	18
22	17	16	15	14	13	12	11	10
21	0F	0E	0D	0C	0B	0A	09	08
20	07	06	05	04	03	02	01	00
1F 18	Bank 3							
17 10	Bank 2							
0F 08	Bank 1							
07 00	Default register bank for R0-R7							

Single bit Instructions

Single-Bit Instructions

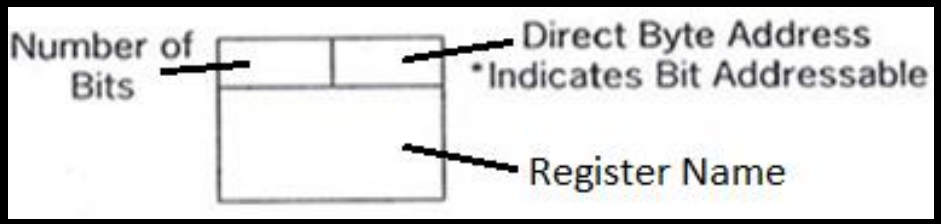
Instruction	Function
SETB bit	Set the bit (bit = 1)
CLR bit	Clear the bit (bit = 0)
CPL bit	Complement the bit (bit = NOT bit)
JB bit, target	Jump to target if bit = 1 (jump if bit)
JNB bit, target	Jump to target if bit = 0 (jump if no bit)
JBC bit, target	Jump to target if bit = 1, clear bit (jump if bit, then clear)

Program Status Word [PSW]



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

Special Function Registers [SFR]



8	E0*
A Register	

8	F0*
B Register	

Math Registers

8	89
TMOD Register	

8	88*
TCON Register	

Timer Control Registers

8	8C
TH0 Counter	

8	8A
TL0 Counter	

8	8D
TH1 Counter	

8	8B
TL1 Counter	

Timer/Counter Registers

8	D0*
PSW Register	

Flags

8	81
Stack Pointer	

8	B8*
IP Register	

8	A8*
IE Register	

Interrupt Registers

8	98*
SCON Register	

8	99
SBUF Register	

8	87
PCON Register	

Serial Data Registers

16	No Address
Program Counter	

8	80*
Port 0 Latch	

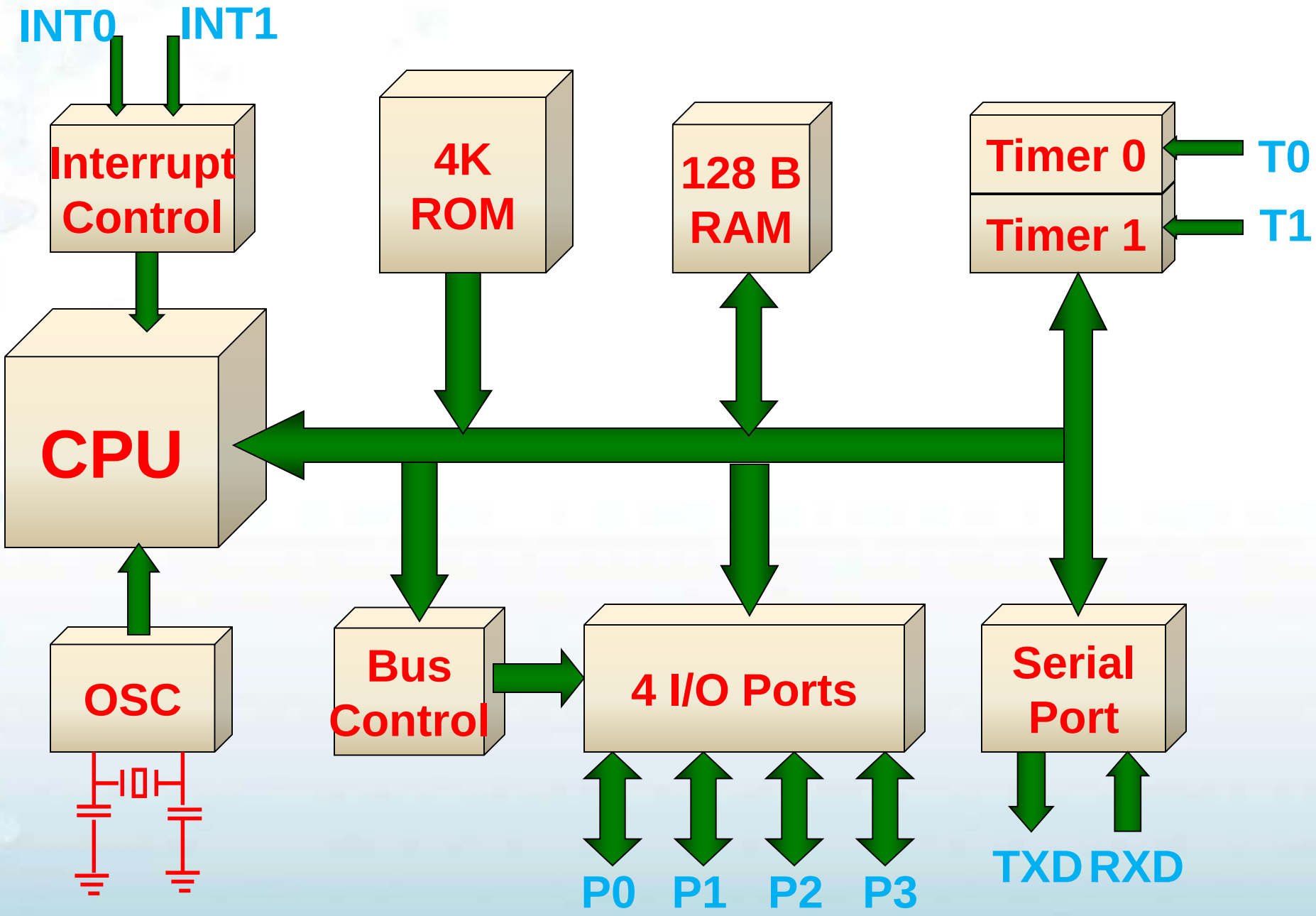
8	90*
Port 1 Latch	

8	A0*
Port 2 Latch	

8	B0*
Port 3 Latch	

8	83	8	82
Data Pointer			
DPH		DPL	

General Block Diagram of 8051



8051 Software Overview

1. Addressing Modes
2. Instruction Set
3. Programming

8051 Addressing Modes

- The CPU can access data in various ways, which are called addressing modes

1. **Immediate**
2. **Register**
3. **Direct**
4. **Register indirect**
5. **External Direct**
6. **Implicit addressing mode**

Addressing Modes (cont.):

Immediate Addressing Mode :

- MOV A, #64H
- MOV R1, #0FFH

Direct Addressing Mode:

- MOV A, 64H
- MOV A, 0FFH

Addressing Modes (cont.):

Register Addressing Mode

- MOV A, R0
- MOV R1, A

Register Indirect Addressing Mode

- MOV A,@R0
- MOV @R1,A

Immediate Addressing Mode

- The source operand is a **constant.**
- The immediate data must be preceded by the pound sign, **"#"**
- Can load information into **any registers**, including 16-bit DPTR register
  - DPTR can also be accessed as two 8-bit registers, the high byte DPH and low byte DPL

```
MOV A,#25H       ;load 25H into A
MOV R4,#62       ;load 62 into R4
MOV B,#40H       ;load 40H into B
MOV DPTR,#4521H ;DPTR=4512H
MOV DPL,#21H     ;This is the same
MOV DPH,#45H     ;as above

;illegal!! Value > 65535 (FFFFH)
MOV DPTR,#68975
```

Register Addressing Mode

- Use registers to hold the data to be manipulated.

```
MOV A,R0     ;copy contents of R0 into A
MOV R2,A     ;copy contents of A into R2
ADD A,R5     ;add contents of R5 to A
ADD A,R7     ;add contents of R7 to A
MOV R6,A     ;save accumulator in R6
```

- The source and destination registers must match in size. **MOV DPTR,A** will give an error

```
MOV DPTR,#25F5H
MOV R7,DPL
MOV R6,DPH
```

- The movement of data between Rn registers is not allowed **MOV R4,R7** is invalid

Direct Addressing Mode

- It is most often used the direct addressing mode to access RAM locations 30 – 7FH.
- The entire 128 bytes of RAM can be accessed.
- Contrast this with immediate addressing mode, there is no “#” sign in the operand.

```
MOV R0,40H  ;save content of 40H in R0
MOV 56H,A   ;save content of A in 56H
```

SFR Registers & their Addresses

```
MOV   0E0H,#55H     ;is the same as
MOV   A,#55H        ;which means load 55H into A (A=55H)

MOV   0F0H,#25H     ;is the same as
MOV    B,#25H       ;which means load 25H into B (B=25H)

MOV   0E0H,R2       ;is the same as
MOV   A,R2          ;which means copy R2 into A

MOV   0F0H,R0       ;is the same as
MOV   B,R0          ;which means copy R0 into B
```

Example

Example 5-1

Write code to send 55H to ports P1 and P2, using (a) their names (b) their addresses.

Solution:

(a)
```
MOV  A,#55H      ;A=55H
MOV  P1,A        ;P1=55H
MOV  P2,A        ;P2=55H
```

(b) From Table 5-1, P1 address = 80H; P2 address = A0H
```
MOV  A,#55H      ;A=55H
MOV  80H,A       ;P1=55H
MOV  0A0H,A      ;P2=55H
```

Stack and Direct Addressing Mode

Show the code to push R5 and A onto the stack and then pop them back them into R2 and B, where B = A and R2 = R5

Solution:

```
PUSH 05          ;push R5 onto stack
PUSH 0E0H        ;push register A onto stack
POP  0F0H        ;pop top of stack into B
                 ;now register B = register A
POP  02          ;pop top of stack into R2
                 ;now R2=R6
```

Register Indirect Addressing Mode

- A **register** is used as a pointer to the data.
- Only register **R0** and **R1** are used for this purpose.
- **R2 – R7** cannot be used to hold the address of an operand located in RAM.
- When **R0** and **R1** hold the addresses of RAM locations, they must be preceded by the **"@"** sign.

```
MOV A,@R0   ;move contents of RAM whose
            ;address is held by R0 into A
MOV @R1,B   ;move contents of B into RAM
            ;whose address is held by R1
```

Register Indirect Addressing Mode

- Write a program to copy the value 55H into RAM memory locations 40H to 41H using **(a) direct addressing mode**, **(b) register indirect addressing mode without a loop**, and **(c) with a loop.**

Solution:

```
(a)
    MOV A,#55H   ;load A with value 55H
    MOV 40H,A    ;copy A to RAM location 40H
    MOV 41H.A    ;copy A to RAM location 41H
(b)
    MOV A,#55H   ;load A with value 55H
    MOV R0,#40H  ;load the pointer. R0=40H
    MOV @R0,A    ;copy A to RAM R0 points to
    INC R0       ;increment pointer. Now R0=41h
    MOV @R0,A    ;copy A to RAM R0 points to
(c)
        MOV A,#55H     ;A=55H
        MOV R0,#40H    ;load pointer.R0=40H,
        MOV R2,#02     ;load counter, R2=3
AGAIN:  MOV @R0,A      ;copy 55 to RAM R0 points to
        INC R0         ;increment R0 pointer
        DJNZ R2,AGAIN  ;loop until counter = zero
```

External Indirect

- External Memory is accessed.

- There are only two commands that use External indirect addressing mode:
- MOV DPTR,#0300H
  - **MOVX A, @DPTR**
    MOVX @DPTR, A

- DPTR must first be loaded with the address of external memory.

Implicit Addressing Mode

- Data will there in A.
- Source and destination of data will A.
- CPL A
- RL A
- RLC A
- RR A
- RRC A
- SWAP A

8051 Instruction Set

- 8051 instructions have 8-bit opcode
- There are 256 possible instructions of which 255 are
- implemented

MOV Instruction

- **MOV destination, source** ; copy source to destination.

- ```
  MOV  A,#55H   ;load value 55H into reg. A
  MOV  R0,A     ;copy contents of A into R0
                  ;(now A=R0=55H)
  MOV  R1,A     ;copy contents of A into R1
                  ;(now A=R0=R1=55H)
  MOV  R2,A     ;copy contents of A into R2
                  ;(now A=R0=R1=R2=55H)
  MOV  R3,#95H  ;load value 95H into R3
                  ;(now R3=95H)
  MOV  A,R3     ;copy contents of R3 into A
                  ;now A=R3=95H
  ```
  ```

ADD Instruction

- **ADD A, source** ;ADD the source operand to the accumulator

- MOV A, #25H ;load 25H into A

  MOV R2,#34H ;load 34H into R2

- ;add R2 to accumulator

  ;(A = A + R2)

Structure of Assembly Language

```
        ORG   0H          ;start (origin) at location 0
        MOV  R5,#25H      ;load 25H into R5
        MOV  R7,#34H      ;load 34H into R7
        MOV  A,#0         ;load 0 into A
        ADD  A,R5         ;add contents of R5 to A
                          ;now A = A + R5
        ADD  A,R7         ;add contents of R7 to A
                          ;now A = A + R7
        ADD  A,#12H       ;add to A value 12H
                          ;now A = A + 12H
HERE: SJMP HERE           ;stay in this loop
         END              ;end of asm source file
```

Data Types & Directives

```
        ORG   500H
DATA1:  DB    28                ;DECIMAL (1C in Hex)
DATA2:  DB    00110101B         ;BINARY (35 in Hex)
DATA3:  DB    39H               ;HEX
        ORG   510H
DATA4:  DB    "2591"            ; ASCII NUMBERS
        ORG   518H
DATA6:  DB    "My name is Joe"  ;ASCII CHARACTERS
```

ADD Instruction and PSW

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	00101111
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).

ADD Instruction and PSW

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	01100100
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).

Multiplication of Unsigned Numbers

```
MUL   AB      ; A × B, place 16-bit result in B and A
```

```
MOV   A,#25H      ;load 25H to reg. A
MOV   B,#65H      ;load 65H in reg. B
MUL   AB          ;25H * 65H = E99 where B = 0EH and A = 99H
```

Table 6-1:Unsigned Multiplication Summary (MUL AB)

Multiplication	Operand 1	Operand 2	Result
byte byte	A	B	A=low byte, B=high byte

Division of Unsigned Numbers

DIV AB ; divide A by B

- MOV A,#95H ;load 95 into A
- MOV B,#10H ;load 10 into B
- DIV AB ;now A = 09 (quotient) and B = 05 (remainder)

Table 6-2:Unsigned Division Summary (DIV AB)

Division	Numerator	Denominator	Quotient	Remainder
byte / byte	A	B	A	B

Unconditional Jump Instructions

- All conditional jumps are short jumps
  - Target address within -128 to +127 of PC
- **LJMP** (long jump): 3-byte instruction
  - 2-byte target address: 0000 to FFFFH
  - Original 8051 has only 4KB on-chip ROM
- **SJMP** (short jump): 2-byte instruction
  - 1-byte relative address: -128 to +127

8051 Conditional Jump Instructions

Table 3-1: 8051 Conditional Jump Instructions

Instruction	Action
JZ	Jump if A = 0
JNZ	Jump if A ≠ 0
DJNZ	Decrement and jump if A ≠ 0
CJNE A,byte	Jump if A ≠ byte
CJNE reg,#data	Jump if byte ≠ #data
JC	Jump if CY = 1
JNC	Jump if CY = 0
JB	Jump if bit = 1
JNB	Jump if bit = 0
JBC	Jump if bit = 1 and clear bit

Checking an input bit

JNB (jump if no bit) ; JB (jump if bit = 1)

Example 8-3

Assume that bit P2.3 is an input and represents the condition of an oven. If it goes high, it means that the oven is hot. Monitor the bit continuously. Whenever it goes high, send a high-to-low pulse to port P1.5 to turn on a buzzer.

Solution:

```
HERE:JNB  P2.3,HERE      ;keep monitoring for high
     SETB P1.5           ;set bit P1.5=1
     CLR  P1.5           ;make high-to-low
```

Conditional Jump

- The 8051 offers a variety of conditional jump instructions
- JZ and JNZ tests the accumulator for a particular condition
- DJNZ (decrement and jump if not zero) is a useful instruction for building loops
- To execute a loop N times, load a register with N and terminate the loop with a DJNZ to the beginning of the loop
- CJNE (compare and jump if not equal) is another conditional jump instruction
- CJNE: two bytes in the operand field are taken as unsigned integers. If the first one is less than the second one, the carry is set
- Example: It is desired to jump to BIG if the value of the accumulator is greater than or equal to 20_H

  CJNE A,#20_H,$+3

  JNC BIG

  – $ is an assembler symbol representing the address of the current instruction
  – Since CJNE is a 3-byte instruction, $+3 is the address of next instruction JNC

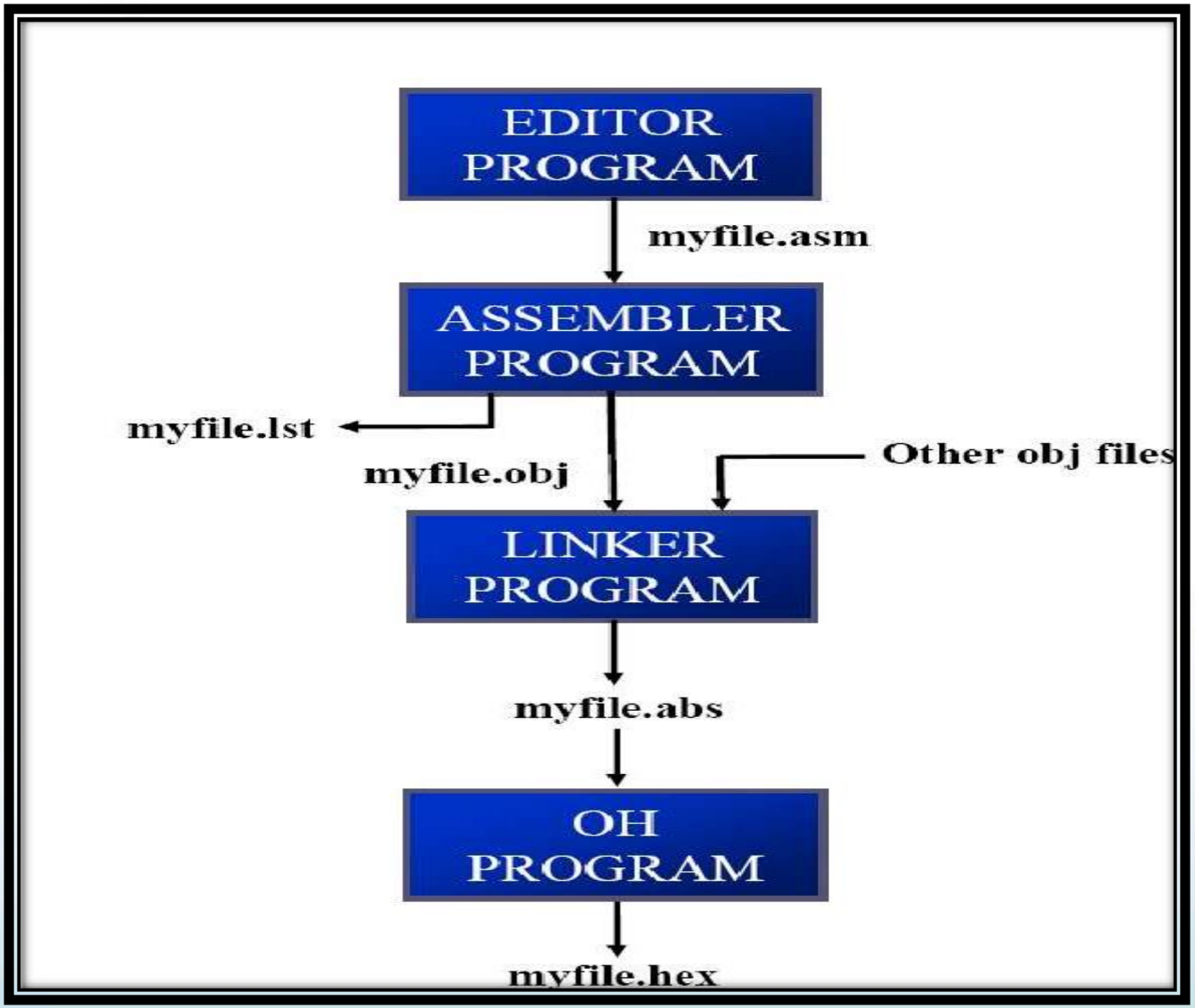
Call Instructions

- LCALL (long call): 3-byte instruction
  - 2-byte address
  - Target address within 64K-byte range
- ACALL (absolute call): 2-byte instruction
  - 11-bit address
  - Target address within 2K-byte range

C Programming concepts

1. Data types
2. looping
3. Logical operators

How to Write Program:



WHY PROGRAM 8051 IN C

- Compilers produce hex files that is downloaded to ROM of microcontroller
  - The size of hex file is the main concern
    - Microcontrollers have limited on-chip ROM
    - Code space for 8051 is limited to 64K bytes
- C programming is less time consuming, but has larger hex file size
- The reasons for writing programs in C
  - It is easier and less time consuming to write in C than Assembly
  - C is easier to modify and update
  - You can use code available in function libraries
  - C code is portable to other microcontroller with little of no modification

DATA TYPES

- A good understanding of C data types for 8051 can help programmers to create smaller hex files
  - Unsigned char
  - Signed char
  - Unsigned int
  - Signed int
  - Sbit (single bit)
  - Bit and sfr

Data types in Embedded C:

Data Type	Size in Bits	Data Range/Usage
unsigned char	8-bit	0 to 255
(signed) char	8-bit	-128 to +127
unsigned int	16-bit	0 to 65535
(signed) int	16-bit	-32768 to +32767
sbit	1-bit	SFR bit-addressable only
bit	1-bit	RAM bit-addressable only
sfr	8-bit	RAM addresses 80 – FFH only

DATA TYPES

Unsigned char

- The character data type is the most natural choice
  - 8051 is an 8-bit microcontroller
- Unsigned char is an 8-bit data type in the range of 0 – 255 (00 – FFH)
  - One of the most widely used data types for the 8051
    - Counter value
    - ASCII characters
- C compilers use the signed char as the default if we do not put the keyword *unsigned*

DATA TYPES

Unsigned char (cont')

Write an 8051 C program to send values 00 – FF to port P1.

Solution:

```
#include <reg51.h>
void main(void)
  {
    unsigned char z;
    for (z=0;z<=255;z++)
       P1=z;
  }
```

1. Pay careful attention to the size of the data
2. Try to use unsigned *char* instead of *int* if possible

Write an 8051 C program to send hex values for ASCII characters of 0, 1, 2, 3, 4, 5, A, B, C, and D to port P1.

Solution:

```
#include <reg51.h>
void main(void)
  {
    unsigned char mynum[]="012345ABCD";
    unsigned char z;
    for (z=0;z<=10;z++)
       P1=mynum[z];
  }
```

DATA TYPES

Unsigned char (cont')

Write an 8051 C program to toggle all the bits of P1 continuously.

Solution:

```
//Toggle P1 forever
#include <reg51.h>
void main(void)
  {
    for (;;)
      {
        p1=0x55;
        p1=0xAA;
       }
  }
```

DATA TYPES

Signed char

- The signed char is an 8-bit data type
  - Use the MSB D7 to represent – or +
  - Give us values from –128 to +127
- We should stick with the unsigned char unless the data needs to be represented as signed numbers
  - temperature

Write an 8051 C program to send values of –4 to +4 to port P1.

Solution:

```
//Singed numbers
#include <reg51.h>
void main(void)
  {
    char mynum[]={+1,-1,+2,-2,+3,-3,+4,-4};
    unsigned char z;
    for (z=0;z<=8;z++)
       P1=mynum[z];
  }
```

DATA TYPES

Unsigned and Signed int

- The unsigned int is a 16-bit data type
  - Takes a value in the range of 0 to 65535 (0000 – FFFFH)
  - Define 16-bit variables such as memory addresses
  - Set counter values of more than 256
  - Since registers and memory accesses are in 8-bit chunks, the misuse of int variables will result in a larger hex file
- Signed int is a 16-bit data type
  - Use the MSB D15 to represent – or +
  - We have 15 bits for the magnitude of the number from –32768 to +32767

DATA TYPES

Single Bit (cont')

Write an 8051 C program to toggle bit D0 of the port P1 (P1.0) 50,000 times.

Solution:

```
#include <reg51.h>
sbit MYBIT=P1^0;

void main(void)
  {
    unsigned int z;
    for (z=0;z<=50000;z++)
       {
         MYBIT=0;
         MYBIT=1;
       }
  }
```

sbit keyword allows access to the single bits of the SFR registers

DATA TYPES

Bit and sfr

- The bit data type allows access to single bits of bit-addressable memory spaces 20 – 2FH
- To access the byte-size SFR registers, we use the sfr data type

Data Type	Size in Bits	Data Range/Usage
unsigned char	8-bit	0 to 255
(signed) char	8-bit	-128 to +127
unsigned int	16-bit	0 to 65535
(signed) int	16-bit	-32768 to +32767
sbit	1-bit	SFR bit-addressable only
bit	1-bit	RAM bit-addressable only
sfr	8-bit	RAM addresses 80 – FFH only

TIME DELAY (cont')

Write an 8051 C program to toggle bits of P1 continuously forever with some delay.

Solution:

```
//Toggle P1 forever with some delay in between
//"on" and "off"
#include <reg51.h>
void main(void)
  {
    unsigned int x;
    for (;;)                    //repeat forever
      {
        p1=0x55;
        for (x=0;x<40000;x++); //delay size
                               //unknown
        p1=0xAA;
        for (x=0;x<40000;x++);
      }
  }
```

We must use the oscilloscope to measure the exact duration

TIME DELAY (cont')

Write an 8051 C program to toggle bits of P1 ports continuously with a 250 ms.

Solution:

```
#include <reg51.h>
void MSDelay(unsigned int);
void main(void)
  {
    while (1)                  //repeat forever
       {
         p1=0x55;
         MSDelay(250);
         p1=0xAA;
         MSDelay(250);
       }
  }

void MSDelay(unsigned int itime)
  {
    unsigned int i,j;
    for (i=0;i<itime;i++)
       for (j=0;j<1275;j++);
  }
```

I/O PROGRAMMING

Byte Size I/O

LEDs are connected to bits P1 and P2. Write an 8051 C program that shows the count from 0 to FFH (0000 0000 to 1111 1111 in binary) on the LEDs.

Solution:

```
#include <reg51.h>
#defind LED P2;

void main(void)
  {
    P1=00;              //clear P1
    LED=0;              //clear P2
    for (;;)            //repeat forever
       {
         P1++;          //increment P1
         LED++;         //increment P2
       }
  }
```

Ports P0 – P3 are byte-accessable and we use the P0 – P3 labels as defined in the 8051/52 header file.

I/O PROGRAMMING

Using bit Data Type for Bit-addressable RAM

Write an 8051 C program to get the status of bit P1.0, save it, and send it to P2.7 continuously.

Solution:

```
#include <reg51.h>
sbit inbit=P1^0;
sbit outbit=P2^7;
bit membit;              //use bit to declare
                         //bit- addressable memory

void main(void)
  {
    while (1)
       {
         membit=inbit;     //get a bit from P1.0
         outbit=membit;    //send it to P2.7
       }
  }
```

We use bit data type to access data in a bit-addressable section of the data RAM space 20 – 2FH

LOGIC OPERATIONS

Bit-wise Operators in C

- Logical operators
  - AND (&&), OR (||), and NOT (!)
- Bit-wise operators
  - AND (&), OR (|), EX-OR (^), Inverter (~), Shift Right (>>), and Shift Left (<<)
    - These operators are widely used in software engineering for embedded systems and control

Bit-wise Logic Operators for C

		AND	OR	EX-OR	Inverter
A	B	A&B	A|B	A^B	~B
0	0	0	0	0	1
0	1	0	1	1	0
1	0	0	1	1	
1	1	1	1	0	

LOGIC OPERATIONS

Bit-wise Operators in C (cont')

Run the following program on your simulator and examine the results.

Solution:

```
#include <reg51.h>

void main(void)
  {
    P0=0x35 & 0x0F;          //ANDing
    P1=0x04 | 0x68;          //ORing
    P2=0x54 ^ 0x78;          //XORing
    P0=~0x55;                //inversing
    P1=0x9A >> 3;            //shifting right 3
    P2=0x77 >> 4;            //shifting right 4
    P0=0x6 << 4;             //shifting left 4
  }
```

LOGIC OPERATIONS

Bit-wise Operators in C (cont')

Write an 8051 C program to toggle all the bits of P0 and P2 continuously with a 250 ms delay. Using the inverting and Ex-OR operators, respectively.

Solution:

```
#include <reg51.h>
void MSDelay(unsigned int);

void main(void)
  {
    P0=0x55;
    P2=0x55;
    while (1)
       {
         P0=~P0;
         P2=P2^0xFF;
         MSDelay(250);
       }
  }
```

Conclusion

- Speed with command is the secret of success.
- Be proactive and adaptive.
- Technological integration is the real education for any industry.
- Time & opportunity do not wait for anyone.