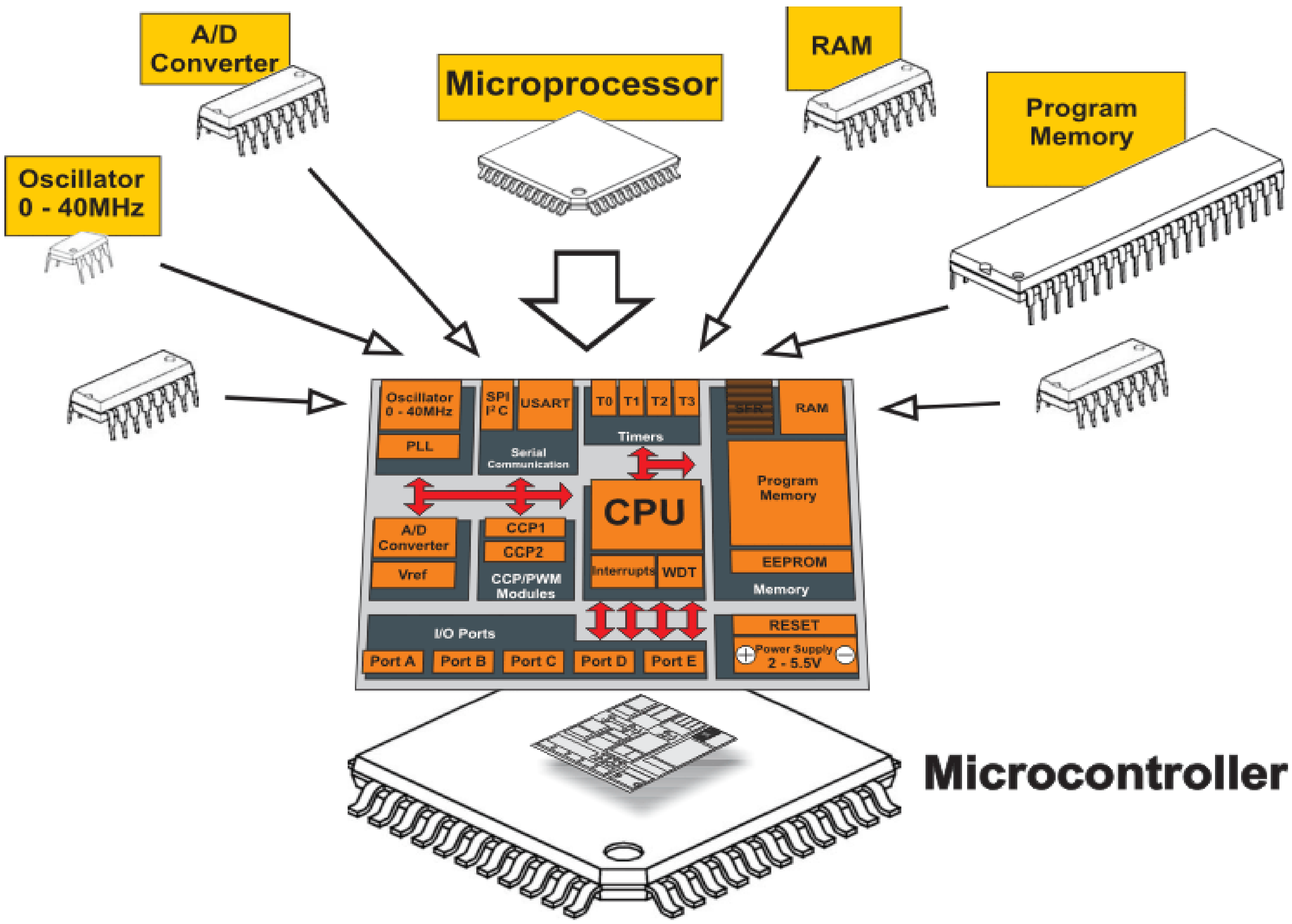
Microcontroller

Chapter 3

Microcontroller

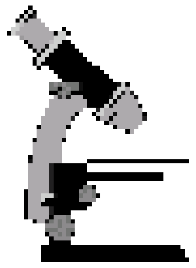
- Introduction 8051 architecture
- Programming model.
- Internal RAM and registers, I/O parts
- Interrupt system & Instruction sets



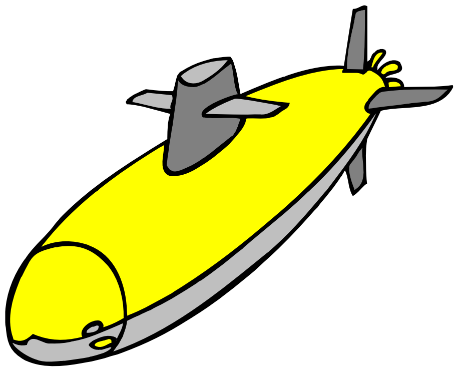
A/D Converter
Microprocessor
RAM
Program Memory
Oscillator 0 - 40MHz
Oscillator 0 - 40MHz
PLL
SPI I²C
USART
Serial Communication
T0
T1
T2
T3
Timers
SFR
RAM
Program Memory
EEPROM
Memory
CPU
Interrupts
WDT
A/D Converter
Vref
CCP1
CCP2
CCP/PWM Modules
I/O Ports
Port A
Port B
Port C
Port D
Port E
RESET
Power Supply 2 - 5.5V
Microcontroller

Application areas

- Medical systems



- Military applications

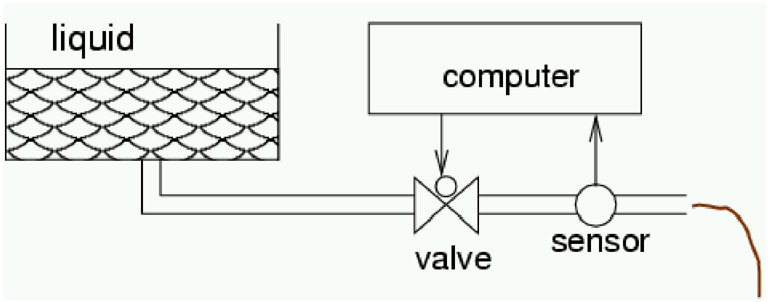


- Authentication

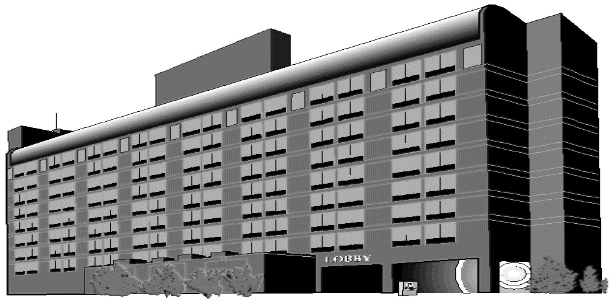
Application areas

- Consumer electronics

- Fabrication equipment



- Smart buildings



Essential Components

- Microprocessor / DSP
- Sensors
- Converters (A-D and D-A)
- Actuators
- Memory (On-chip and Off chip)
- Communication path with the interacting environment

Classification of Embedded System

1. **Small Scale** Embedded Systems: 8-16 bit microcontroller, little h/w and s/w complexities and involve board level design.

   Usually, ‘C’ is used for developing these systems. The software has to fit within the memory available in the system.

Contt..

2. **Medium Scale** Embedded Systems:

   - 16 or 32 bit Microcontroller.DSP or RISC

   - H/W and S/W complexities

3. **Sophisticated** Embedded Systems:

   - Enormous h/w and s/w complexities.

   - may need scalable processors, configurable processors.

Features of the Embedded System

1. Constituents of the embedded computer: h/w and s/w
2. Timeliness: The controller must be able to respond fast enough to keep its operation within a safe region.
3. System interconnection
4. Reliability

Why do we need to learn Microprocessors/controllers?

- The microprocessor is the core of computer systems.
- Nowadays 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.

Different aspects of a microprocessor/controller

- **Hardware** :Interface to the real world
- **Software** :order how to deal with inputs

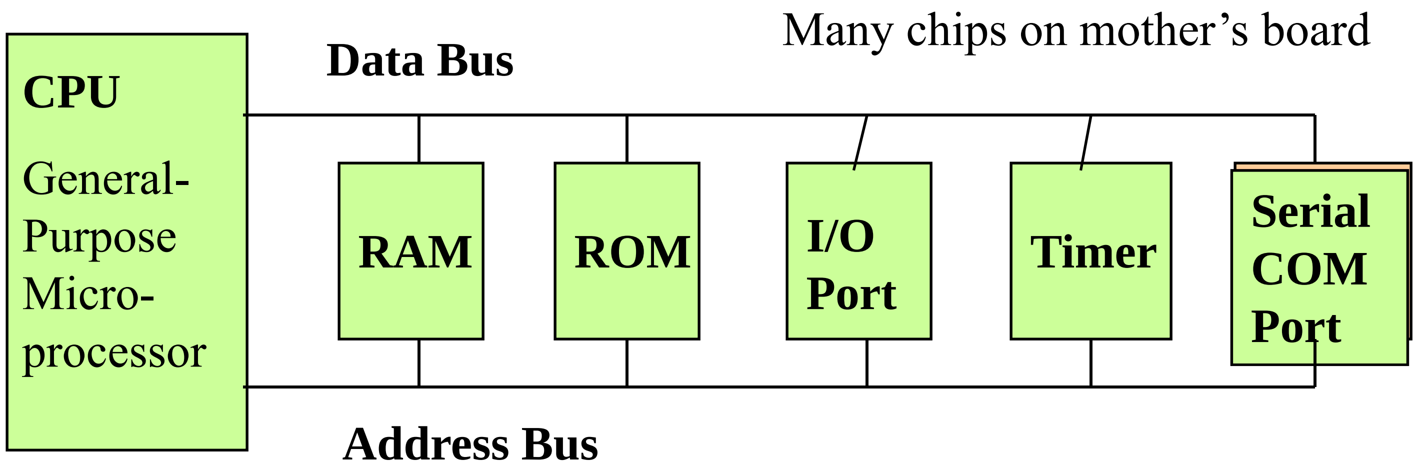
Tools for a microprocessor/controller

- **CPU**: Central Processing Unit
- **I/O**: Input /Output
- **Bus**: Address bus & Data bus
- **Memory**: RAM & ROM
- **Timer**
- **Interrupt**
- **Serial Port**
- **Parallel Port**

Microprocessors:

General-purpose microprocessor

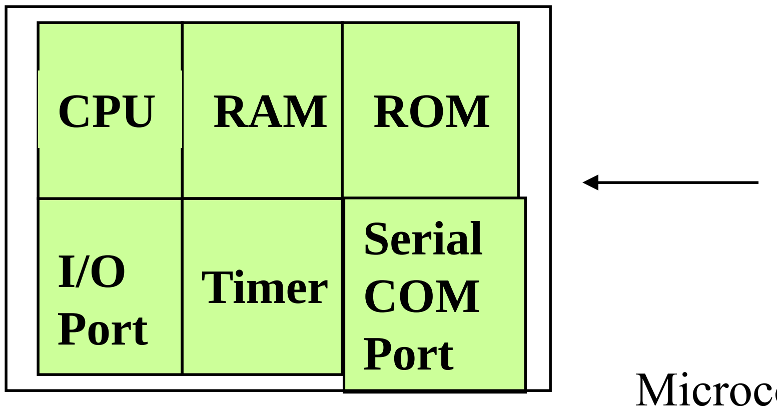
- CPU for Computers
- No RAM, ROM, I/O on CPU chip itself
- Example：Intel’s x86, Motorola’s 680x0



General-Purpose Microprocessor System

Microcontroller :

- A smaller computer
- 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.
- Different Ics for memory and I/O.
- Single memory map in which data & code will lie.
- 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
- Memory and I/O are inbuilt.
- Separate data and code memory.
- for applications in which cost, power and space are critical
- single-purpose.
- Compared to up, more numbers of pins are multifunctioned.

DIFFERENT TYPE OF MICROCONTROLLERS

- **EMBEDDED MICROCONTROLLERS**: When all the hardware required to run the application is provided on the chip, it is referred as an **embedded** Microcontrollers.

  – They are replacing devices like 555 timers because they are actually cheaper to use in applications

  – They are more precise

  – They are easier to control.

DIFFERENT TYPE OF MICROCONTROLLERS

- **EXTERNAL MEMORY** MICROCONTROLLERS: The microcontrollers which **allows the connection of external memory**.

- Higher-end microcontrollers (16 and 32 bit processor) have only external memory .
- These are different from microprocessor in the area of built-in peripheral features.

VARIOUS MICROCONTROLLER

- 8051
- 8052
- PIC
- ARM
- AVR
- HSM
- Various manufacturers are **Intel, Microchip, atmel, Dallas, Phillips, motorolla** etc.

Classification of Microcontrolles

- ***Processor Architecture***

i) Harvard architecture ii) Princeton architecture

- ***Memory***

i) DATA ii) PROGRAM

- ***Types of memory***

i) NONE ii) PROM iii) EPROM iv) EEPROM v) Flash

- ***Instructions***

i) CISC ii) RISC

PROCESSOR ARCHITECTURE

- HARVARD VERSUS PRINCETON: HARVARD UNIVERSITY AND PRINCETON UNIVERSITIES OF USA HAS GIVEN THE COMPUTER ARCHITECTURE .
- CISC OR RISC (Complex Instruction set Computers OR Reduced Instruct Set Computers)

HARVARD VERSUS PRINCETON

HARVARD

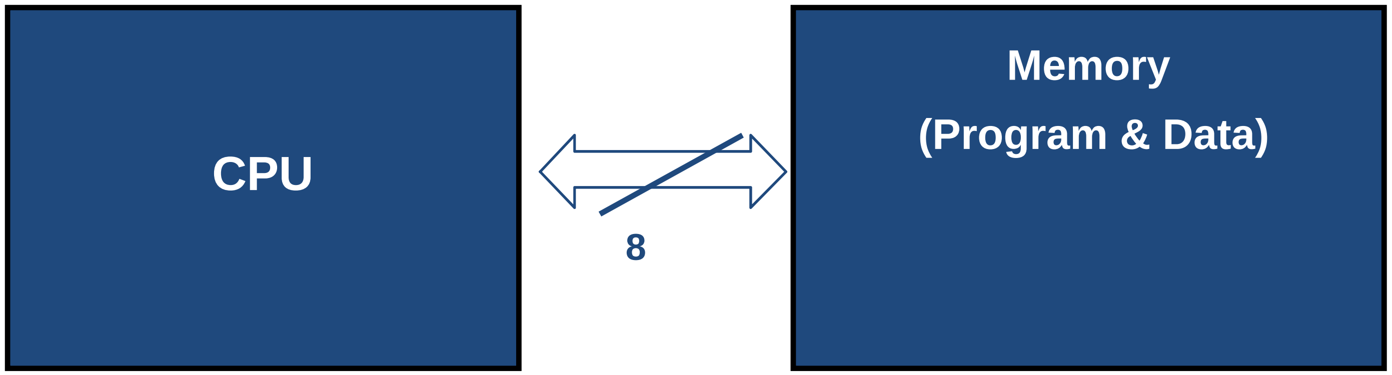
1. It uses separate memory banks for program storage, the processor stack and variable RAM.
2. Previously ignored till 1970 but later it is more popular.
3. Executes instructions in fewer instruction cycle due to Instruction Parallelism.
4. It can carryout the instruction while the next instruction is being fetched from memory and hence helps instructions to take the same no. of cycles for easier timing of loops and critical code.

PRINCETON

1. It uses the common memory for storing the control program as well as variables and other data structures.
2. It was more popular earlier because it simplifies the microcontroller chip design because only one memory is accessed.
3. Executes instructions in more cycles.
4. it requires separate cycles to execute and fetch the instruction.

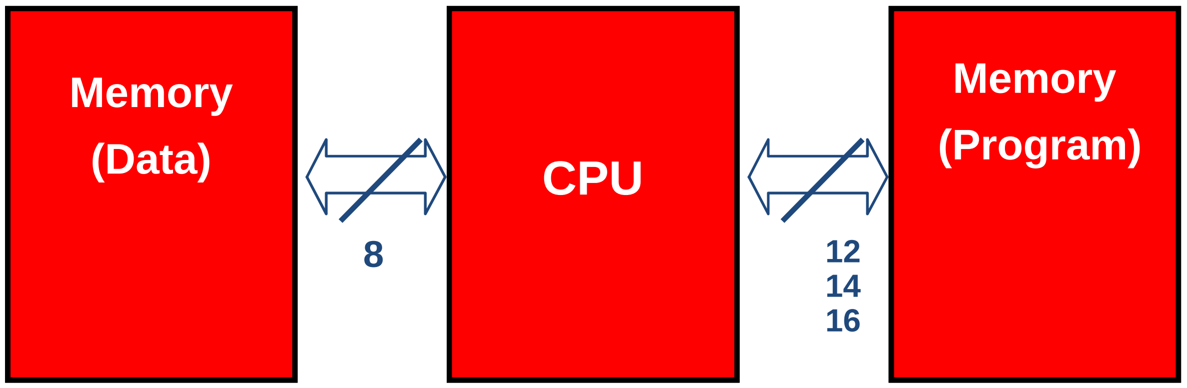
Von-Neuman/Princeton Architecture

- Used in: 80X86 (PCs), 8051, 68HC11, etc.)
- **Only one** bus between CPU and memory
- RAM and program memory share the same bus and the same memory, and so must have the same bit width
- **Bottleneck**: Getting instructions interferes with accessing RAM



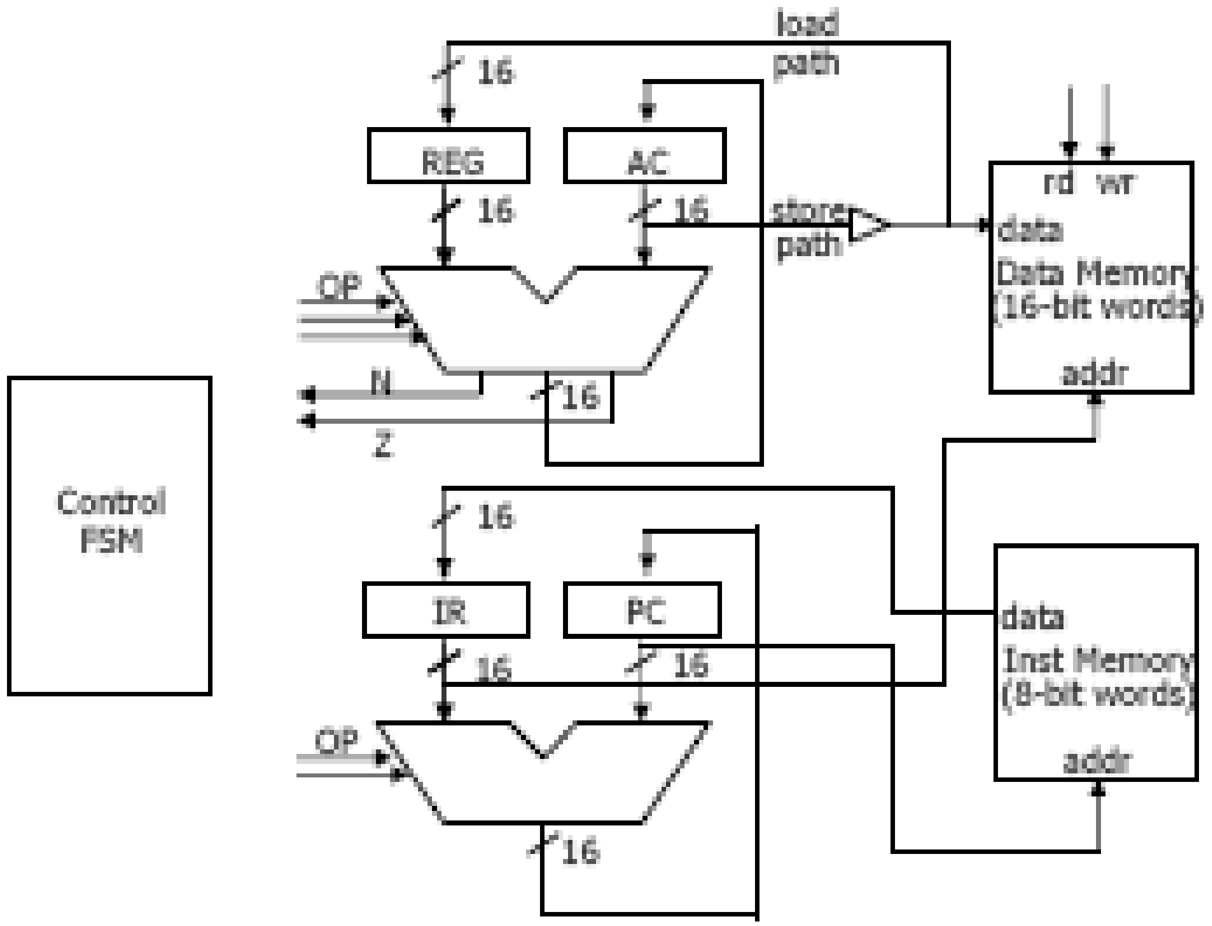
PICs use the Harvard Architecture

- Used mostly in RISC CPUs (we'll get there)
- **Separate** program bus and data bus: can be different widths!
- For example, PICs use:
  - Data memory (RAM): a small number of 8bit registers
  - Program memory (ROM): 12bit, 14bit or 16bit wide (in EPROM, FLASH, or ROM)



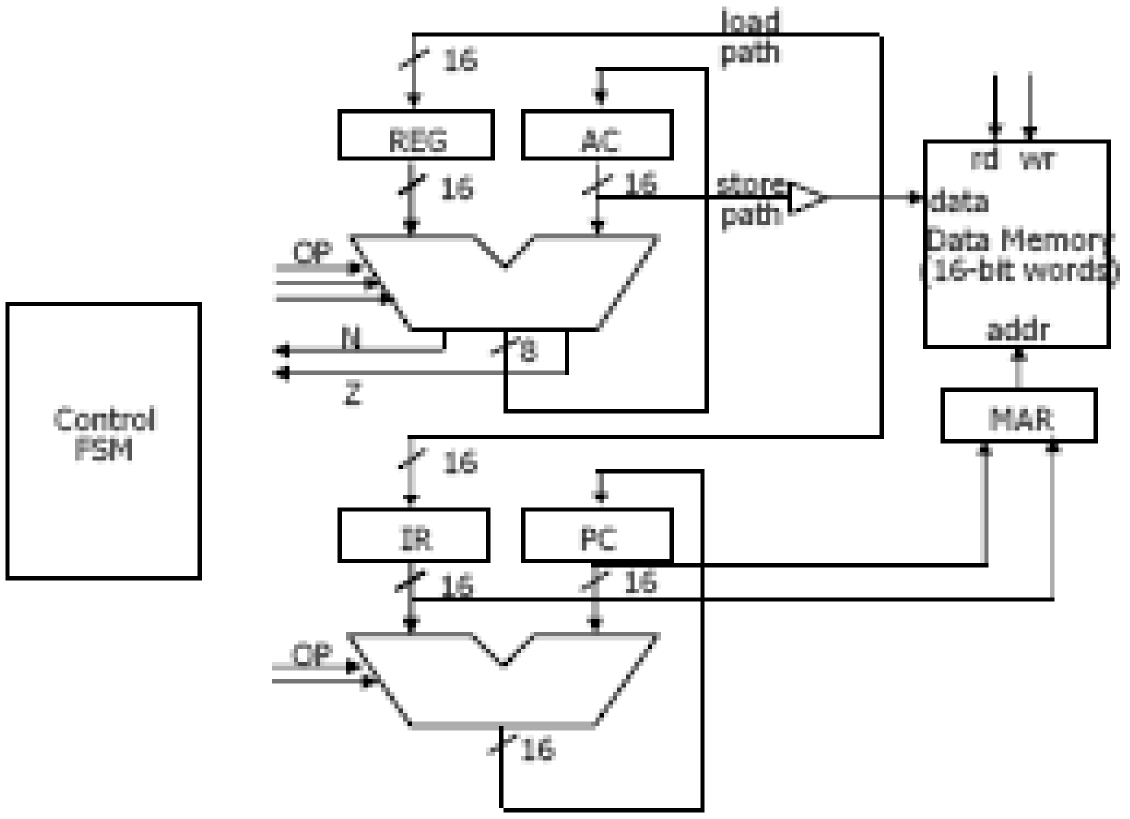
Block diagram of processor (Harvard)

- Register transfer view of Harvard architecture
    - Separate busses for instruction memory and data memory



Block diagram of processor (Princeton)

- Register transfer view of Princeton architecture
  - Single unified bus for instructions, data, and I/O



CISC VS RISC

- **Complex Instruction Set Computer (CISC)**
- Memory in those days was expensive
- bigger program->more storage->more money
- Hence needed to ***reduce the number of instructions per program***
- Number of instructions are reduced by having ***multiple operations*** within a single instruction
- Multiple operations lead to many different kinds of instructions that access memory
- In turn making instruction length variable and fetch-decode-execute time unpredictable – making it more complex
- Thus hardware handles the complexity

- Examples of CISC processors are the
  - System/360(excluding the 'scientific' Model 44),
  - VAX,
  - PDP-11,
  - Motorola 68000 family
  - Intel x86 architecture based processors.

RISC

- Original idea to reduce the ISA(inst. Set archit)
- Provide ***minimal set of instructions that could carry out all essential*** operations
- Instruction complexity is reduced by
- 1. Having ***few simple instructions*** *that are the same length*
- 2. Allowed memory access only ***with explicit load and store instructions***
- Hence each instruction performs less work but instruction execution time among different instructions is consistent
- The complexity that is removed from ISA is moved into the domain of the assembly programmer/compiler
- Examples: LC3, MIPS, PowerPC (IBM), SPARC (Sun)

- Apple iPods (custom ARM7TDMI SoC)
- Apple iPhone (Samsung ARM1176JZF)
- Palm and PocketPC PDAs and smartphones (Intel XScale family, Samsung SC32442 - ARM9)
- Nintendo Game Boy Advance (ARM7)
- Nintendo DS (ARM7, ARM9)
- Sony Network Walkman (Sony in-house ARM based chip)
- Some Nokia and Sony Ericsson mobile phones

- Consider the the program fragments:
- CISC:

```
mov ax, 10
mov bx, 5
mul bx, ax
```

- RISC:

```
mov ax, 0
mov bx, 10
mov cx, 5
Begin add ax, bx
loop Begin
```

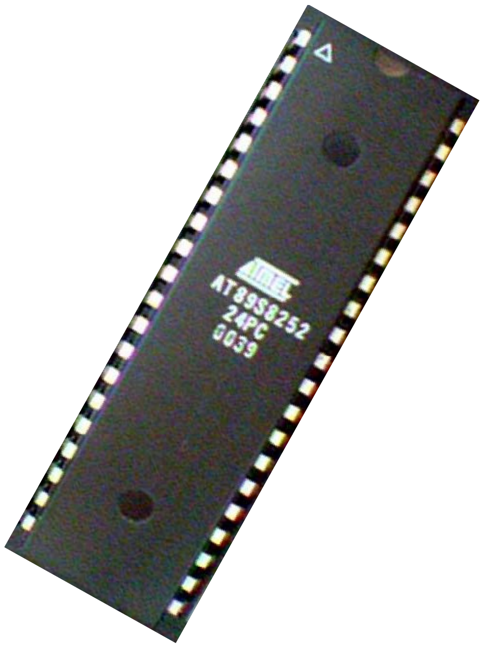
- The total clock cycles for the CISC version might be:
- **(2 movs × 1 cycle) + (1 mul × 30 cycles) = 32 cycles**
- While the clock cycles for the RISC version is:
- **(3 movs × 1 cycle) + (5 adds × 1 cycle) + (5 loops × 1 cycle) = 13**

Comparison

RISC

- • Simple instructions, few in number (128 or less)
- • Fixed length instructions
- All operations are performed on registers, no memory op
- Few addressing modes
- Complexity in compiler
- Only **LOAD/STORE** instructions access memory
- Pipelined inst. Execution
- One inst. Per clock cycle.
- Hardwired control unit design rather than microprogram

Microcontroller 8051



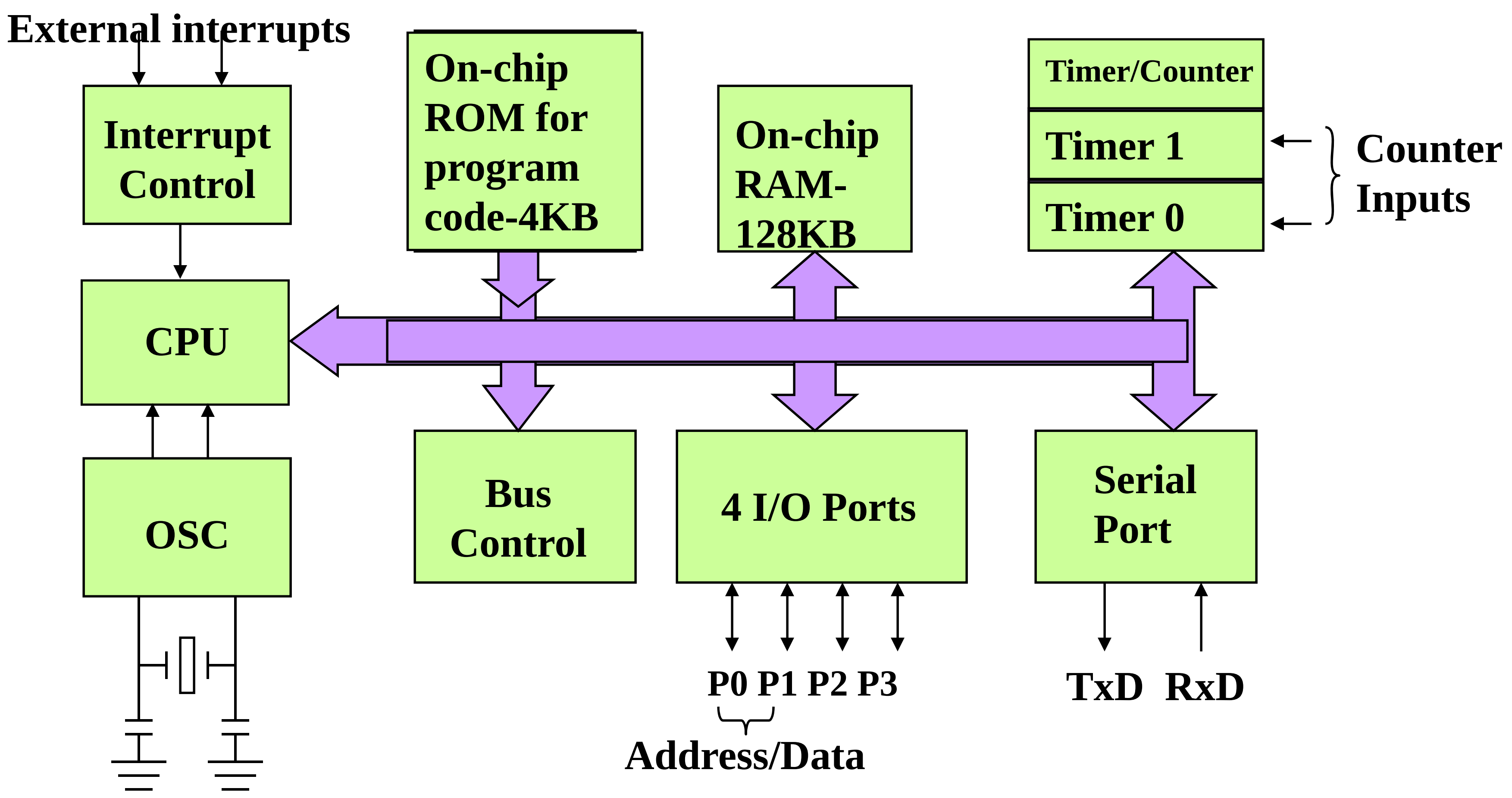
Microcontroller 8051

- Explain the feature of the 8051 mctr.
- Explain the architecture of the 8051 mctr.
- Explain the pin diagram of the 8051 mctr.
- Explain the memory organisation of the 8051 mctr.

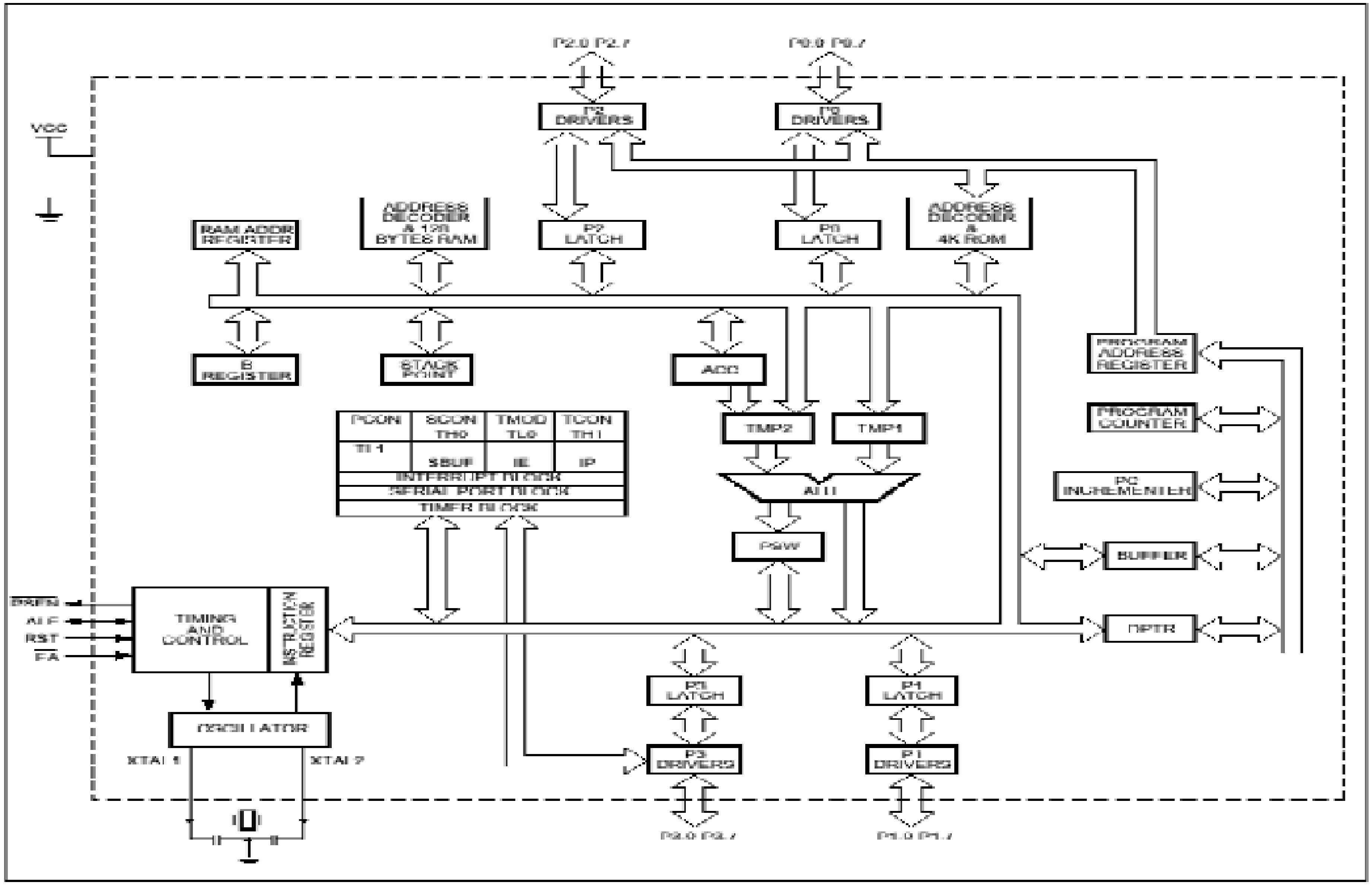
FEATURES OF 8051

- **8- bit microcontroller**
- Operating frequency is **12 MHz**.
- Separate program memory and data memory (**Harvard**).
- Separate 64K program and 64K data memory.
- **4k of on chip EPROM for program memory.**
- **128 bytes RAM (in built)**
- **32 bi-directional and individually addressable I/O lines.**
- **Two nos. of 16-bit timer/counter T0 and T1.**
- Full duplex **UART.**
- On-chip clock oscillator
- **Interrupts from six sources, 2 external and 4 internal.**
- **255** instructions.
- **Bit processing** capability.
- **16-bit** add bus multiplexed with P0 and P2, data bus mux P0

Block Diagram



ARCHITECTURE OF 8051

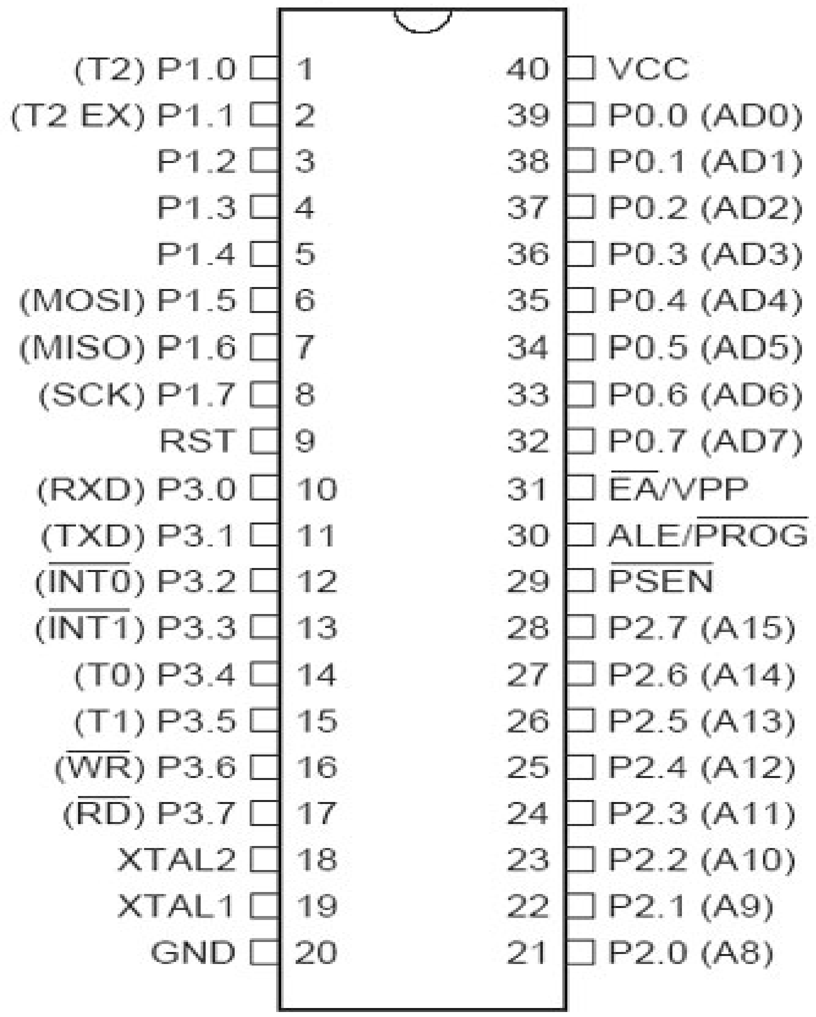


- 8051 Architecture
- 32 I/O pins arranged as four 8 bit ports (P0 – P3)
- 2 16-bit timer/counters: T0 and T1
- Full duplex serial data receiver/transmitter: SBUF
- Control registers: TCON, TMOD, SCON, PCON, IP
- and IE
- 2 external and 4 internal interrupt sources
- Oscillator and clock circuits

8051 PIN DESCRIPTION

- **8051 other family members:**
  - 8751 (has EPROM)
  - 8951 (has EEPROM)

8051 pin description



8051 PIN DESCRIPTION contd.

- Vcc → +5 v ,125 mA, max power diss 1W.
- Vss→ Gnd.
- XTAL2 →o/p of the cryt osc. Ckt is connected. 30pf disc capacitors, when 12MHz quartz cryt is used. In case of external clock , clock is connected to XTAL2.
- XTAL1 → i/p of cryt osc. Ckt is connected. In case of external clock, it is connected to Gnd.
- Port 0 → bidi, serve as low order address and data bus for external memory.
- Port 1 →bidi 8 bit I/O port.
- Port 2 → bidi 8 bit I/O port, and high order address bus.
- Port 3 → bidi 8 bit I/O port and serial i/p ,serial o/p, external int

8051 PIN DESCRIPTION contd.

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

8051 PIN DESCRIPTION contd.

P0	**P1**	**P2**	**P3**	**Port Bit**
P0.0	P1.0	P2.0	P3.0	D0
P0.1	P1.1	P2.1	P3.1	D1
P0.2	P1.2	P2.2	P3.2	D2
P0.3	P1.3	P2.3	P3.3	D3
P0.4	P1.4	P2.4	P3.4	D4
P0.5	P1.5	P2.5	P3.5	D5
P0.6	P1.6	P2.6	P3.6	D6
P0.7	P1.7	P2.7	P3.7	D7

Port Operations

- To read and write from port:

MOV A, P0 or

MOV A,80h

(This copies data from port 0 pins to register A).

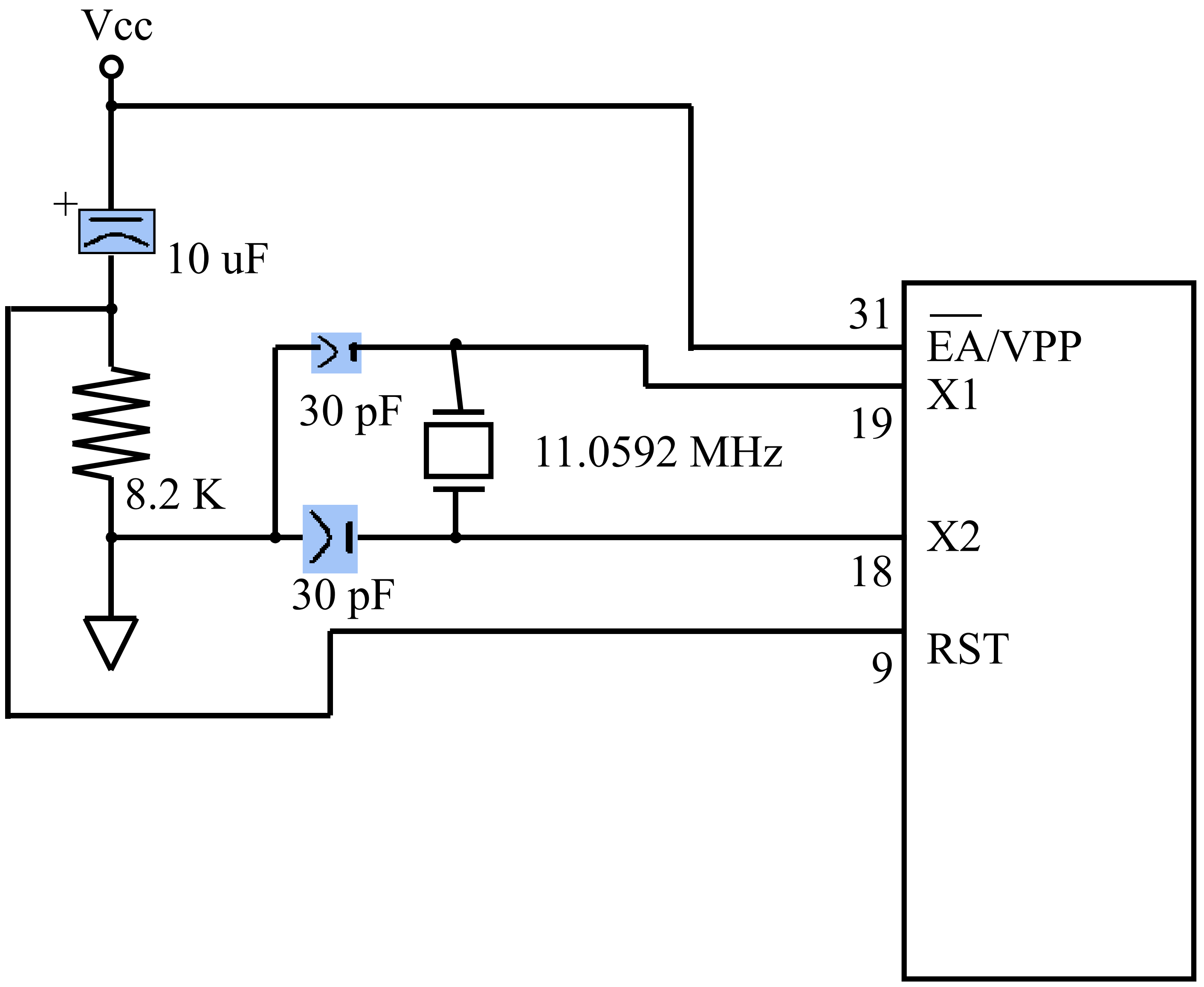
MOV P1, #0a5h or

MOV 90h,#0a5h

(This moves a constant number into port1).

- Moving data to a port changes the port latch,
- moving data from a port gets data from the port pins.

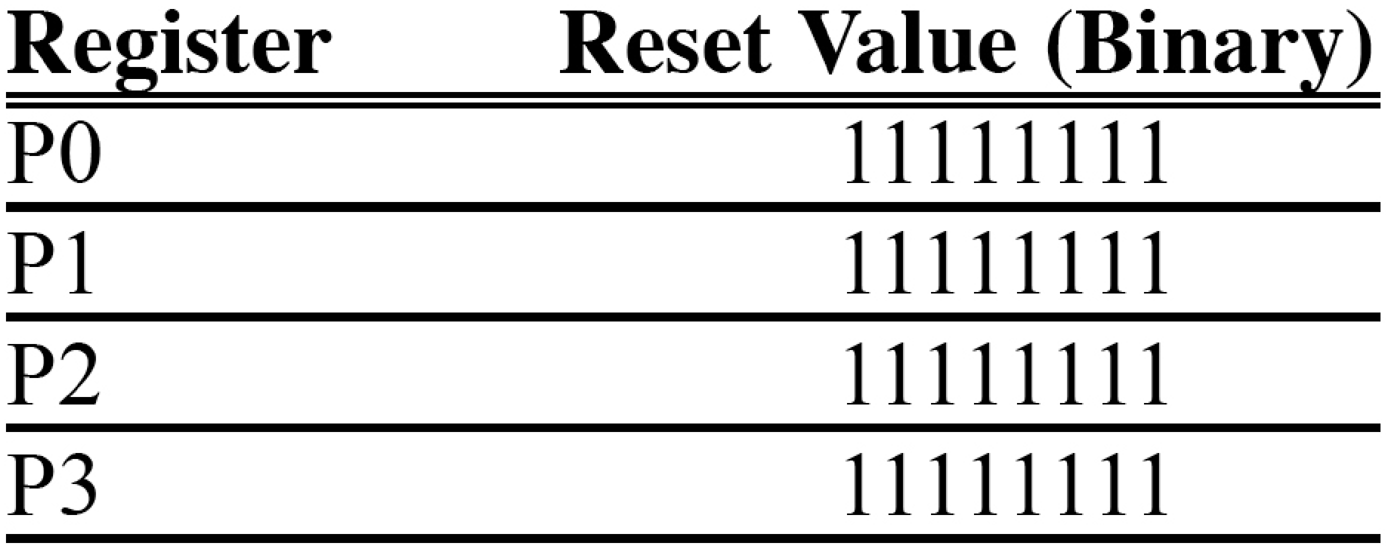
Figure (b). Power-On RESET Circuit



8051 PIN DESCRIPTION contd.

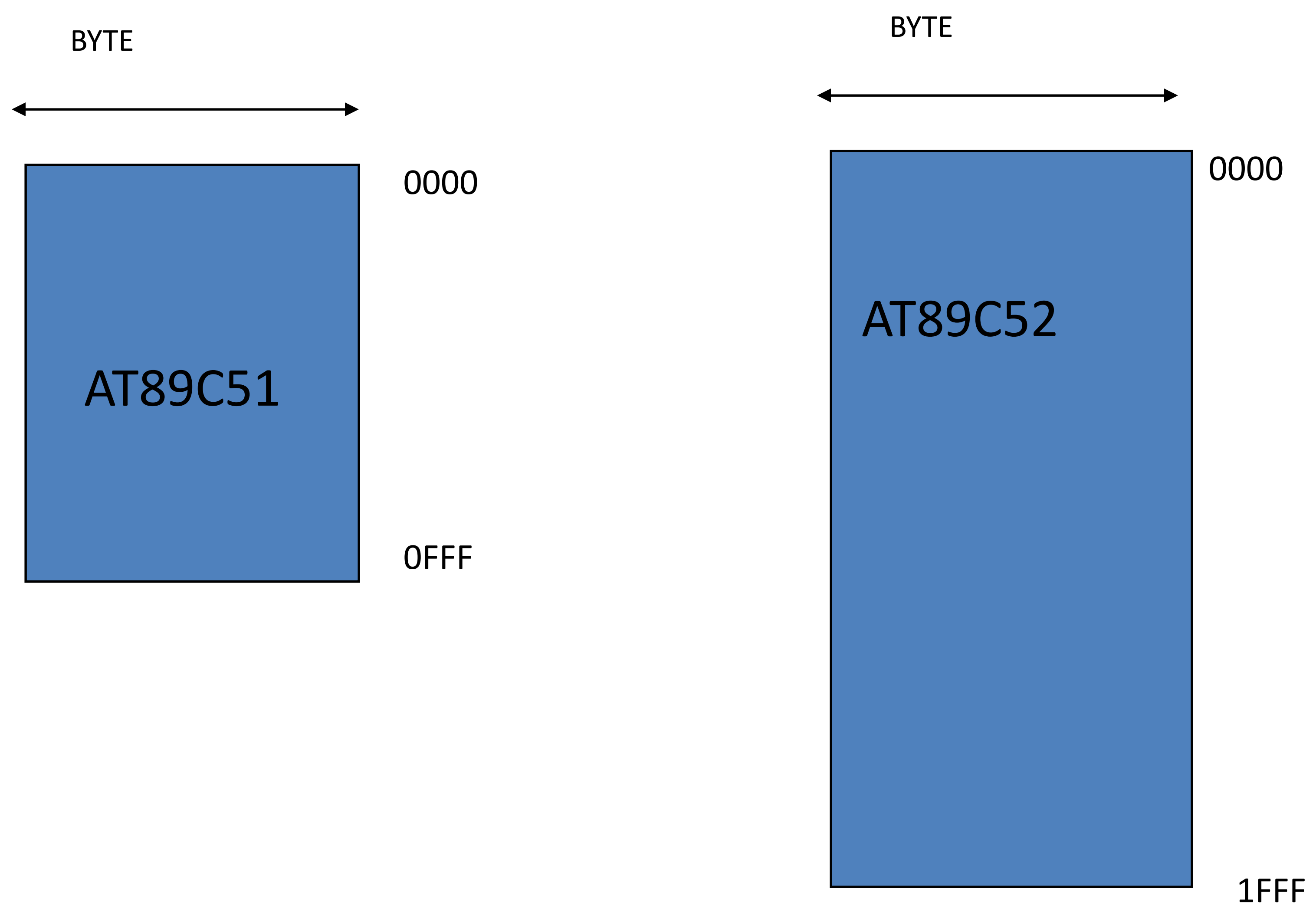
- RST → (reset ckt.) as shown in fig. for resetting 8051, RST pin is made high for 2 m/c. Table below lists SFR’s and reset values

PC	0000H
ACC,B,PSW	00H
SP	07



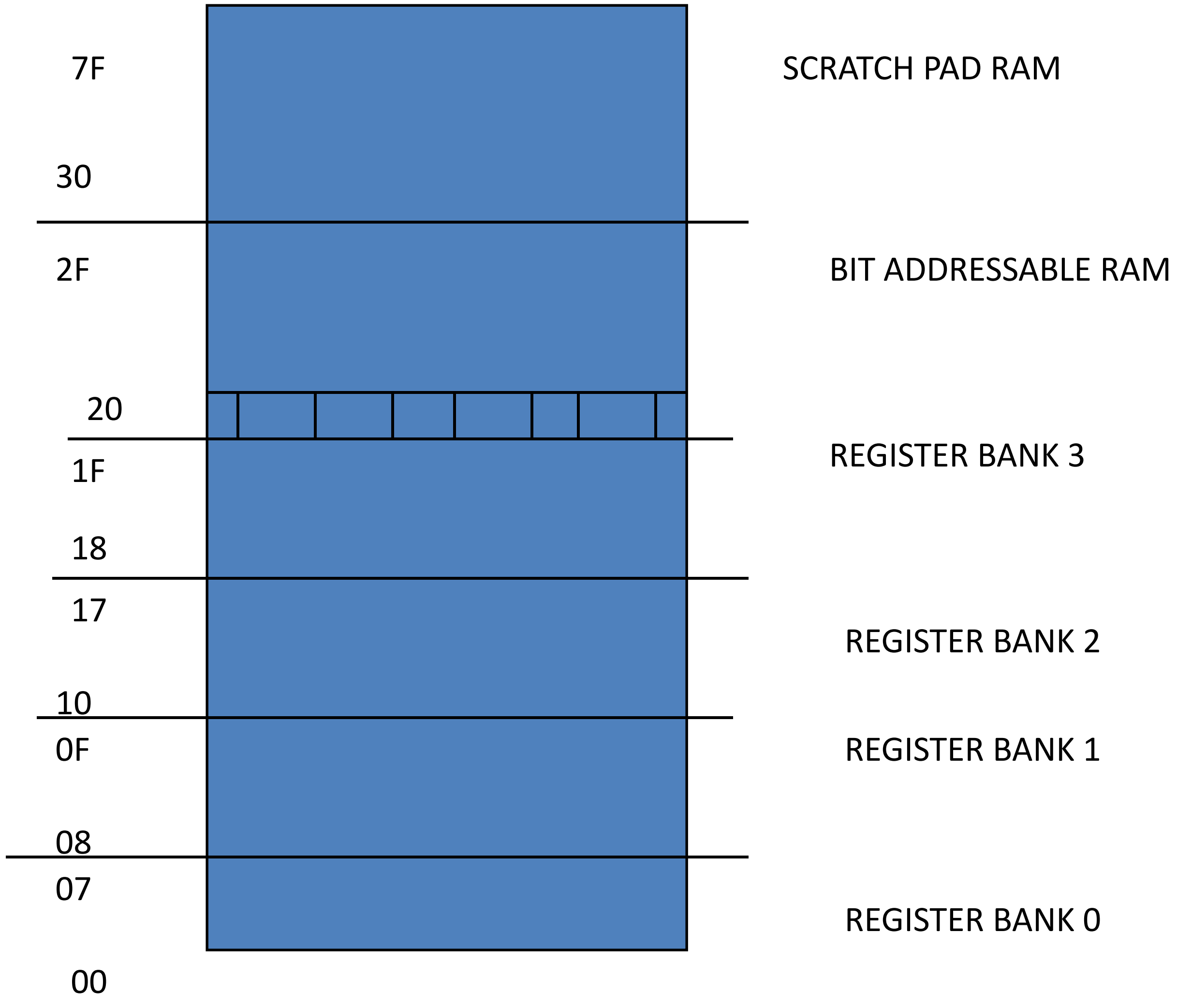
Register	Reset Value (Binary)
P0	11111111
P1	11111111
P2	11111111
P3	11111111

8051 MICROCONTROLLER ON CHIP ROM

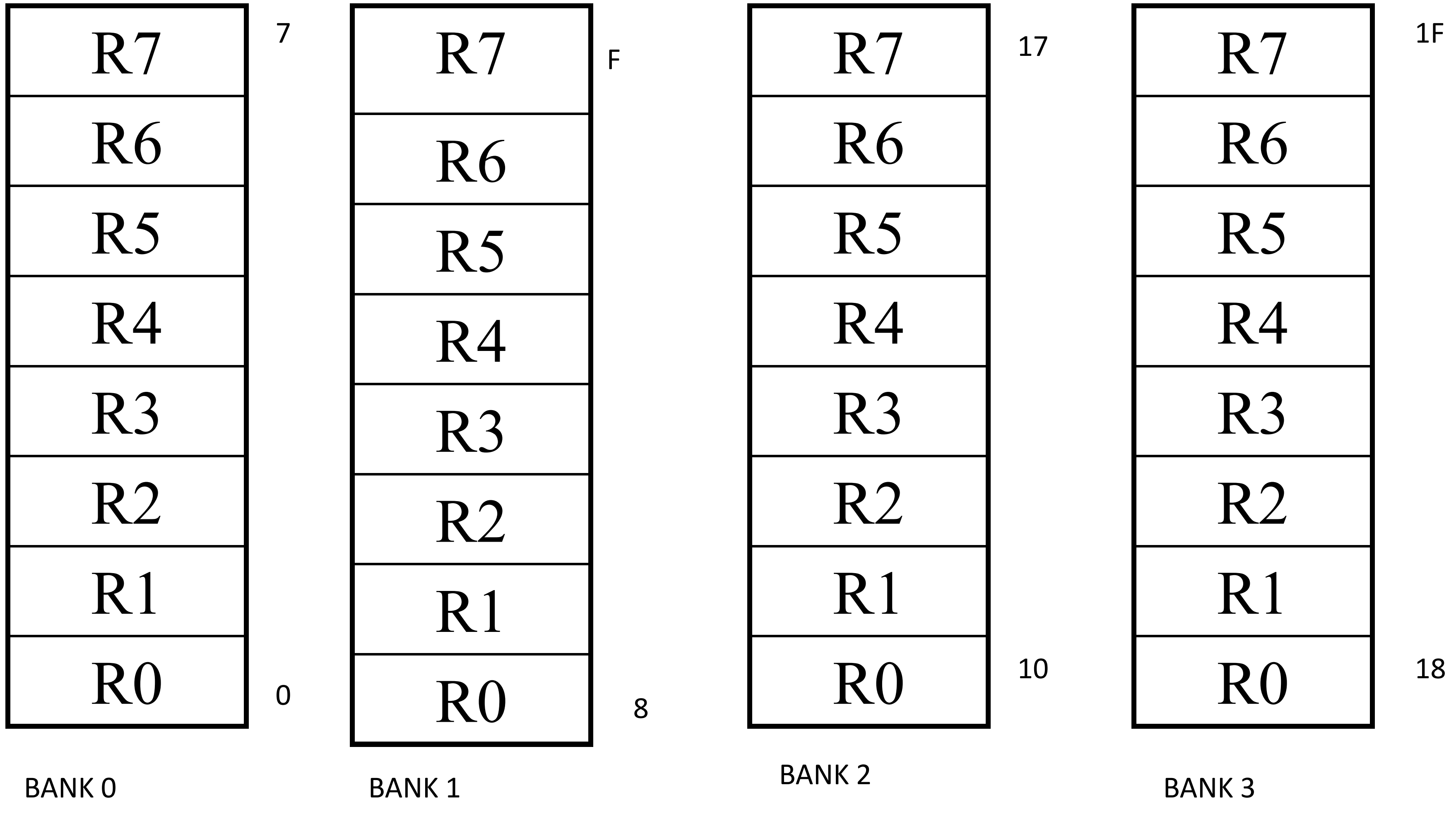


ON CHIP ROM ADDRESS RANGE

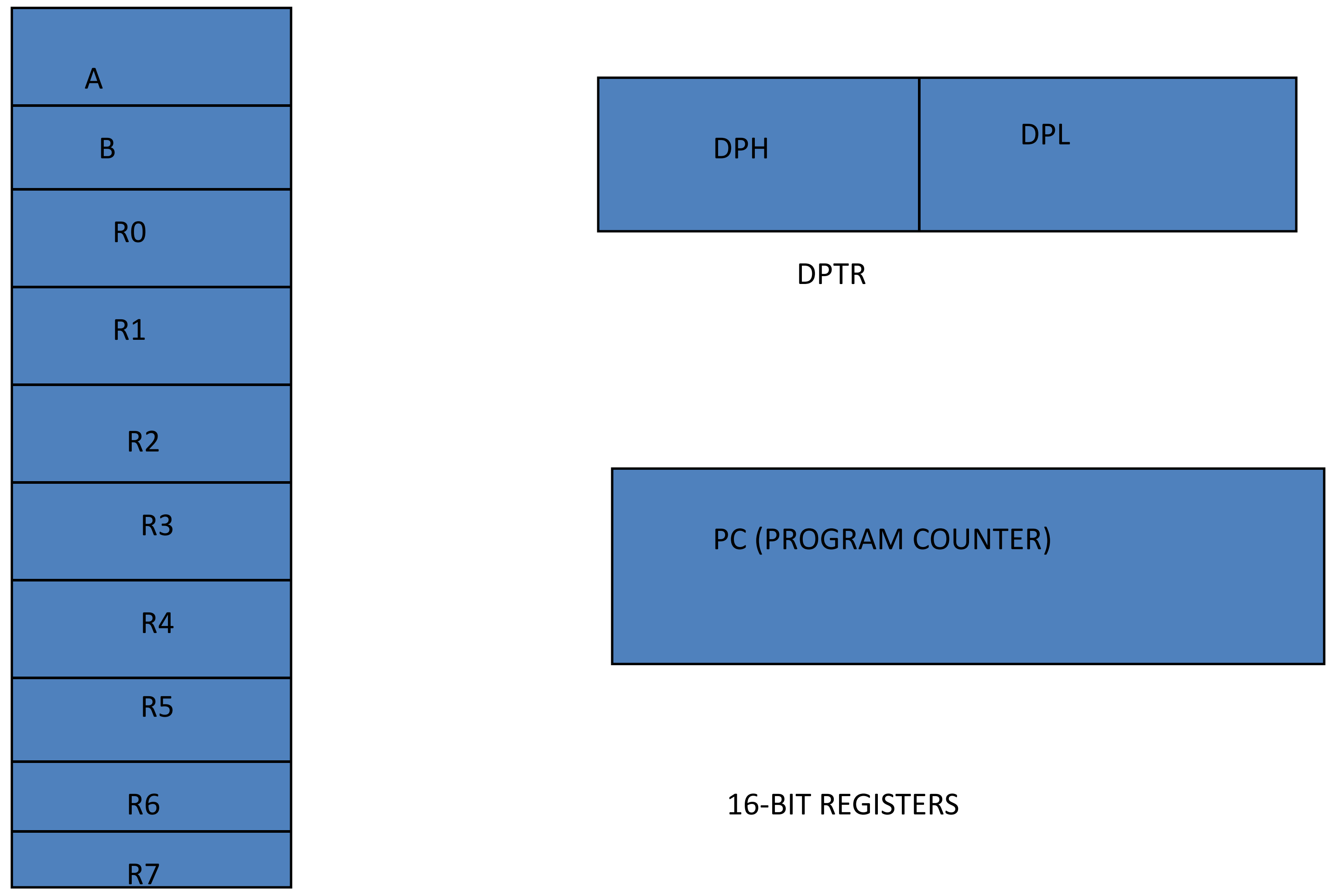
RAM ALLOCATION IN 8051



REGISTER BANKS

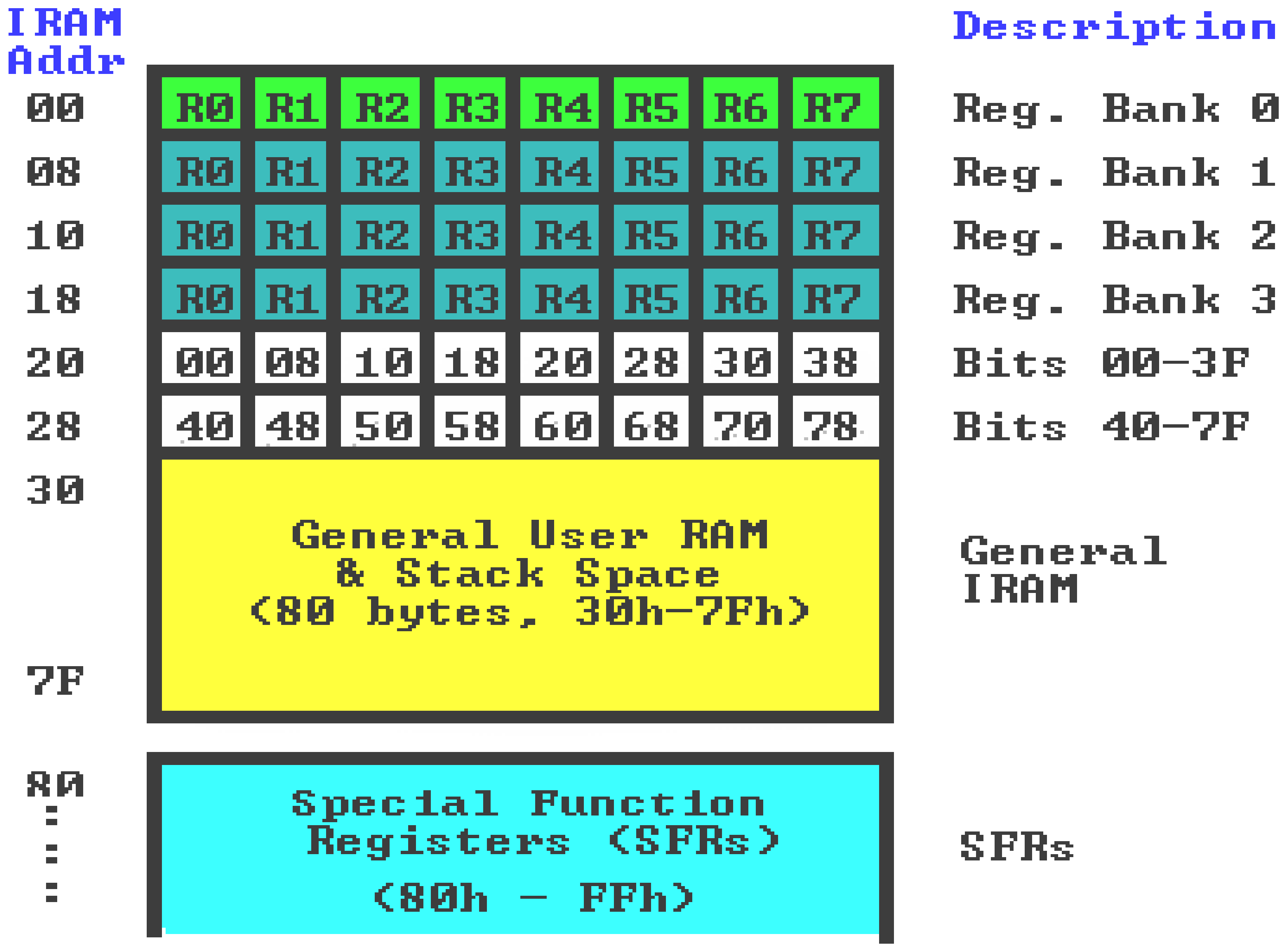


REGISTERS OF 8051

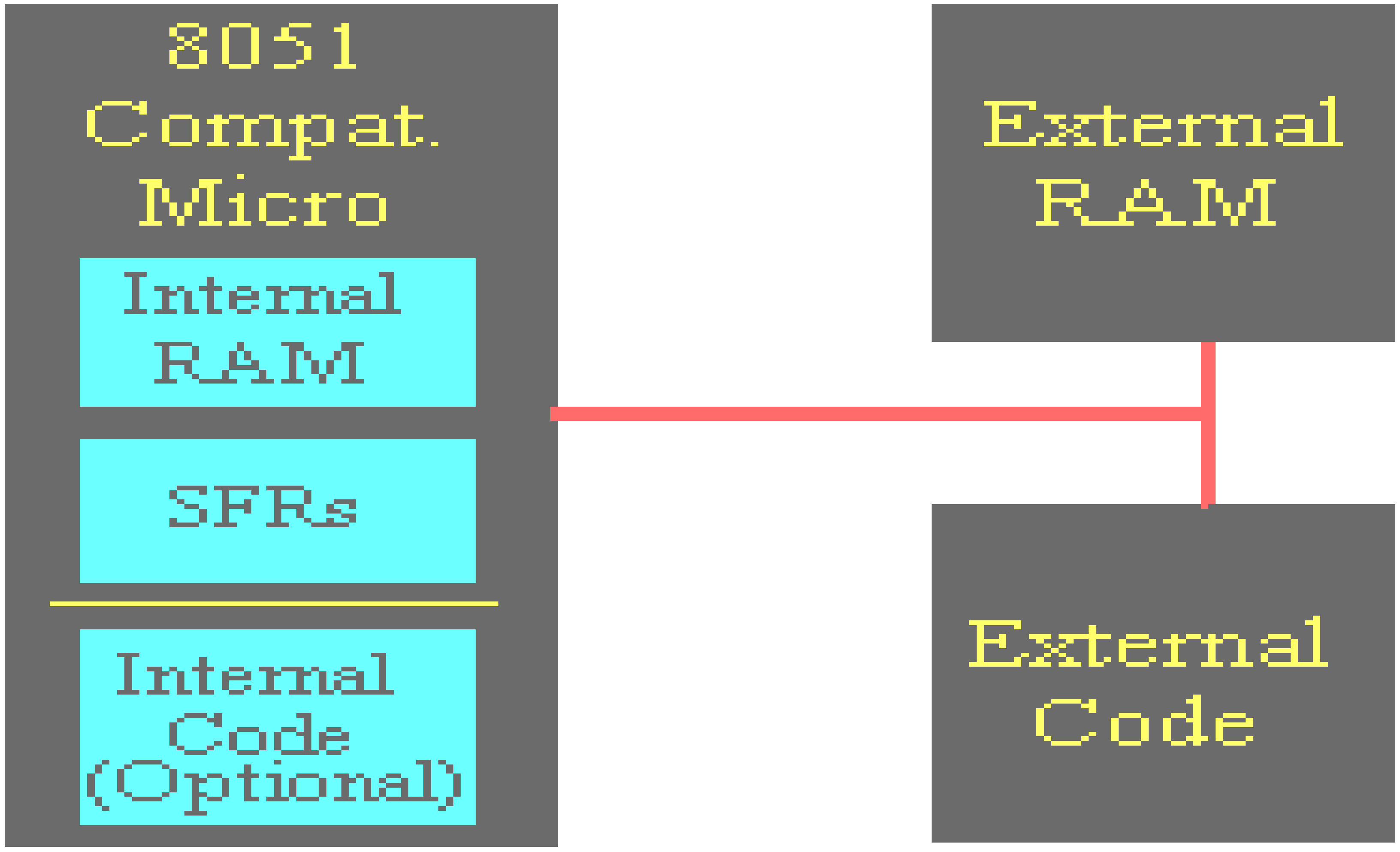


8-BIT REGISTERS

On-Chip Memory.



8051: Types of memory



8051: Types of memory

- **On-Chip Memory refers to any memory** (Code, RAM, or other) that physically exists on the microcontroller itself. On-chip memory can be of several types, but we'll get into that shortly.
- **External Code Memory is code (or program)** memory that resides off-chip. This is often in the form of an external EPROM.
- **External RAM is RAM memory that** resides off-chip. This is often in the form of standard static RAM or flash RAM.

CODE MEMORY

- Code memory is the memory that holds the actual 8051 program that is to be run.
- This memory is limited to 64K.
- It may be found *on-chip, either burned into the microcontroller as* ROM or EPROM. Code may also be stored completely *off-chip in an external ROM or, more* commonly, an external EPROM.
- it is possible to have 4K of code memory *on-chip and 64k of code* memory *off-chip in an EPROM.*

External RAM

- It is off-chip, it is not as flexible in terms of accessing, and is also slower as compared to internal RAM.
- While Internal RAM is limited to 128 bytes the 8051 supports External RAM up to 64K.

MOV and other inst.

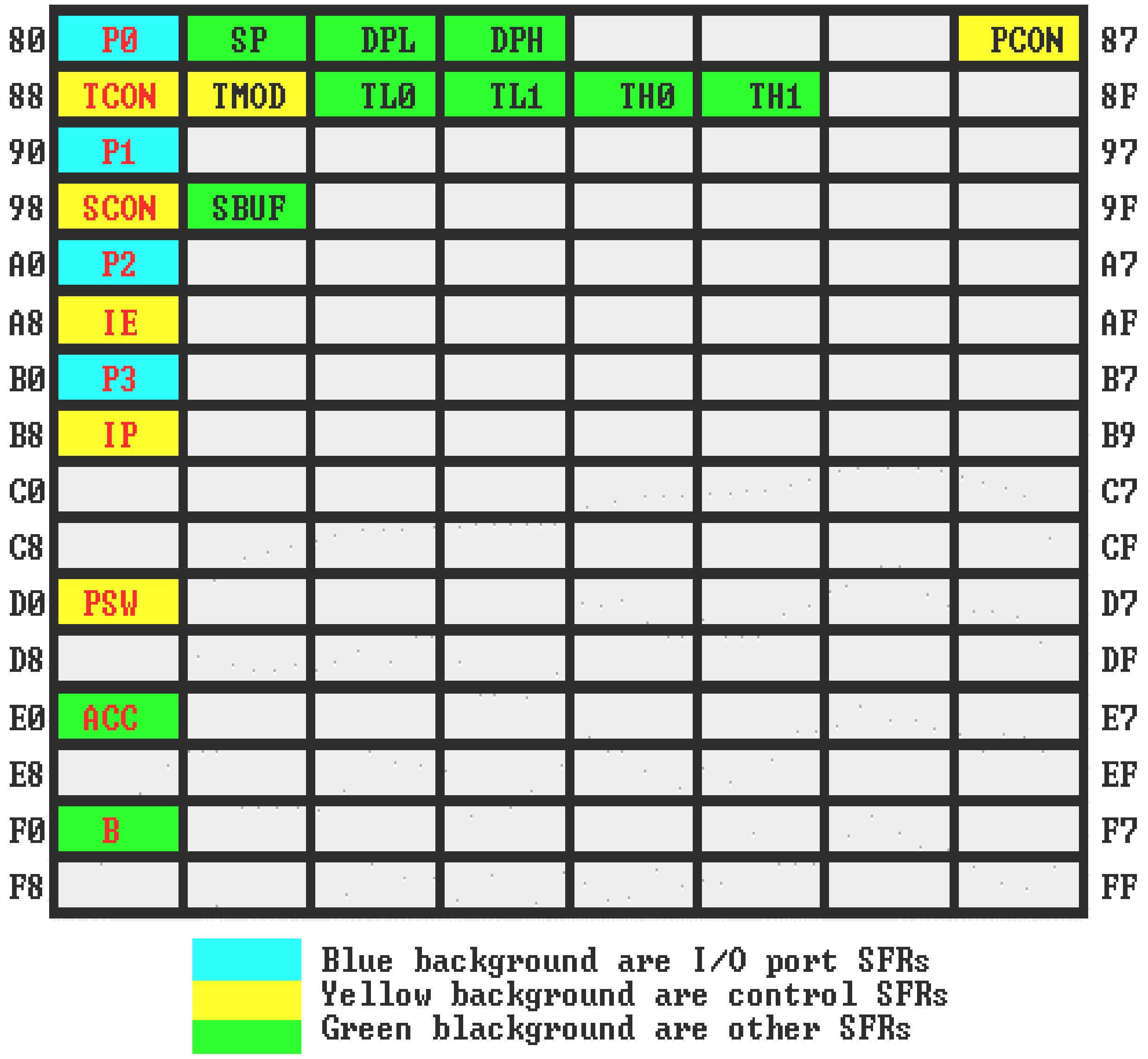
- **MOV A,R3** ;Move the value of R3 into the accumulator
- **MOV R5,A** ;Store the resulting value temporarily in R5
- **MOV A,R1** ;Move the value of R1 into the accumulator
- **MOV A, #55H**
- **(MOV A, #FFH)**
- **MOV A, #0FFH**

ADD and other inst.

- **ADD A, source**
- **ADD A,R2** ;Add the value of R2
- MOV A, #25H
- ADD A,#34H

SFRs(Special Function Registers)

Internal RAM is from address 00h through 7Fh whereas SFR registers exist in the address range of 80h through FFh.



80	P0	SP	DPL	DPH				PCON	87
88	TCON	TMOD	TL0	TL1	TH0	TH1			8F
90	P1								97
98	SCON	SBUF							9F
A0	P2								A7
A8	IE								AF
B0	P3								B7
B8	IP								B9
C0									C7
C8									CF
D0	PSW								D7
D8									DF
E0	ACC								E7
E8									EF
F0	B								F7
F8									FF

🟦	Blue background are I/O port SFRs
🟨	Yellow background are control SFRs
🟩	Green blackground are other SFRs

Special Function Registers

- SFRs contain memory locations that are used for special tasks.
- SFR occupies RAM from **0x80 to 0xFF**, (but some areas are empty!) They are 8 bits wide.
- Ex.

➢ **A register** or accumulator is used for most ALU operations & external moves

➢ **B** used for multiplication & division and can also be used for general purpose storage

➢ **PSW Program Status Word** is a bit addressable register.

SPECIAL FUNCTION REGISTERS

ACC	ACCUMULATOR	0E0H
B	B REGISTER	0F0H
SP	PROGRAM STATUS WORD	0D0H
PSW	STACK POINTER	81H
DPTR	DATA POINTER 2 BYTES	
DPL	LOW BYTE	82H
DPH	HIGH BYTE	83H
P0	PORT 0	80H
P1	PORT 1	90H
P2	PORT 2	0A0H
P3	PORT3	0B0H

CONTD …

IP	INTERRUPT PRIORITY CONTROL	0B8H
IE	INTRRUPT ENABLECONTROL	0A8H
TMOD	TIMER/COUNTER MODE CONTROL	89H
TCON	TIMER COUNTER CONTROL	88H
T2CON	TIMER/COUNTER 2 CONTROL	0C8H
TH0	TIMER/COUNTER 0 HIGH BYTE	8CH
TL0	TIMER/COUNTER 0 LOW BYTE	8AH
TH1	TIMER/COUNTER 1 HIGH BYTE	8DH
TL1	TIMER/COUNTER 1 LOW BYTE	8BH
TH2	TIMER/COUNTER 2 HIGH BYTE	0CDH
TL2	TIMER/COUNTER 2 LOW BYTE	0CCH

RCAP 2H	T/C 2 CAPTURE REG. HIGH BYTE	0CBH
RCAP2L	T/C 2 CAPTURE REG. LOW BYTE	0CAH
SCON	SERIAL CONTROL	98H
SBUF	SERIAL DATA BUFFER	99H
PCON	POWER CONTROL	87H

PC or program counter. This is not directly addressable, nor does it have a memory location.
It is not part of SFR.

DPTR or data pointer. DPL and DPH.

DPTR doesn’t have a single internal address.

This is used to furnish memory addresses for internal and external code access and external data access.

SFRs

SFRs which are also bit addressable

A, B, IP, IE, TCON, SCON, PSW, P0, P1, P2, P3

Other SFRs

TMOD, THO, TLO, TH1, TL1, SBUF, PCON, SP, DPTR

PROGRAM STATUS WORD- BIT ADDRESSABLE

CY	AC	F0*	RS1	RS0	OV	-*	P

- CY-CAARY FLAG
- AC AUXILLARY CARRY FLAG
- F0 –GENERAL PURPOSE FLAG
- RS1- REGISTER BANK SELECTOR BIT
- RS0- REGISTER BANK SELECTOR BIT
- OV- OVERFLOW FLAG
- - USER DEFINABLE FLAG
- P- PARITY FLAG

SELECTION OF REGISTER BANK

RS1	RS0	REGISTER BANK
0	0	0
0	1	1
1	0	2
1	1	3

Quiz

1. 8051 has------- Kbyte f program memory address space.(max memory attach)
2. 8051 has -----no. of interruts.
3. 8051 has---- no. of timers.
4. In divison operation,------ register is used to store quotient.
5. -----flag detects errors in signed arithmetic operations.
6. Stack pointer is ----- bit register.
7. Stack pointer is intialise to ---- after reset.
8. -------16 bit register contains the address of program and external data memory.
9. Port --- and ----- used for address bus.
10. The address of bit addressable RAM is ------ to -------.
11. After reset the value of P0-P3 will be------.
12. True or False. A mctr has fixed amount of RAM on the chip.
13. What is the largest hex value that can be moved into an 8 bit register.
14. Name some bit addressable regisers.
15. Name some SFRs.

Interrupts

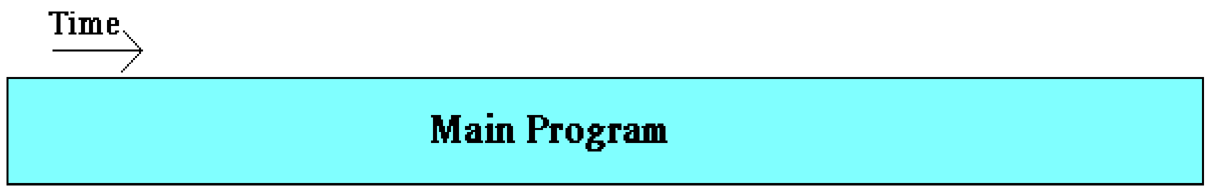
Interrupts are just *special subroutines that may (or may* not) be called explicitly.

If conditions are "right", when an interrupt occurs, then the processor will stop what it is doing, and jump to a specific place in memory (decided by the Intel 8051 designers) hooked by that particular interrupt.

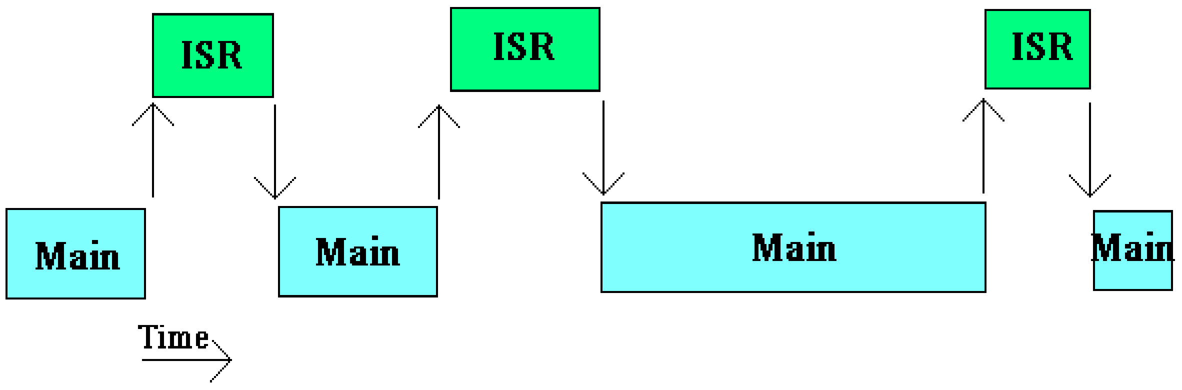
It is up to the programmer to make sure that you supply a sensible further course of action. This is called the interrupt handler routine or *interrupt service routine , ISR.*

Interrupt :

Program execution without intrrupts :



Program execution with intrrupts :



ISR : Intrrupt Service Routin

INTERRUPTS

All interrupt functions are under the control of the program. The programmer can alter the bits of IE, IP and TCON register.

Each interrupt forces the processor to jump at the interrupt location in the memory.

The interrupted program must resume operation at the instruction where the interrupt took place. Program resumption is done by storing the interrupted PC address on to stack. RETI instruction at the end of ISR will restore the PC address.

Types of Interrupts

Five interrupts are provided on 8051.

3 are generated by internal operations.

Generated by internal timer/counter

Timer flag 0 – TF0

Timer flag 1 – TF1

Indicates that a character has been received or the buffer is empty and a character can be transmitted

Serial port interrupt (RI or TI)

2 are triggered by external signals

INT0~

INT1~

IN 8052 ONE MORE INTERRUPT IS THERE FOR TIMER 2.

INTERRUPTS

INTERRUPT SOURCE	VECTOR ADDRESS
IE0	0003H
TFO	000BH
IE1	0013H
TF1	001BH
R1&T1(SERIAL INT.)	0023H
TF2 & EXF2	002BH

INTERRUPT ENABLE REGISTER

EA	-*	ET2	ES	ET1	EX1	ET0	EX0

- EA- ENABLE/DISABLES THE INTERRUPTS
- -*- RESERVE FOR FUTURE USE.
- ET2- ENABLE OR DISABLE THE TIMER 2 OVERFLOW.
- ES- ENABLE OR DISABLE THE SERIAL PORT INTERRUPT.
- ET1- ENABLE OR DISABLE THE TIMER 1 OVERFLOW
- EX1- ENABLE OR DISABLE EXTERNAL INTERRUPT 1.
- ET0- ENABLE OR DISABLE THE TIMER 0 OVERFLOW
- EX0- ENABLE OR DISABLE EXTERNAL INTERRUPT 0.

PCON- POWER CONTROL REGISTER- NOT BIT ADDRESSABLE.

SMOD	-*	*-*	-	GF1	GF0	PD	IDL

- SMOD- DOUBLE BAUD RATE BIT. IF TIMER 1 IS USED TO GENERATE BAUD RATE & SMOD =1, THE BAUD RATE IS DOUBLED WHEN THE SERIAL PORT IS USED IN MODES 1,2 OR 3.
- -* RESERVE FOR FUTURE USE.
- GF1- GENERAL PURPOSE FLAG BIT
- GF0- GENERAL PURPOSE FLAG BIT
- PD- POWER DOWN BIT.
- IDL- IDLE MOD BIT.

IF IDL & PD BOTH ARE SET PD WILL BE GIVEN PREFFERENCE.

TIMER 0

TH0	D15	D14	D13	D12	D11	D10	D9	D8

TL0	D7	D6	D5	D4	D3	D2	D1	D0

TIMER 0

- MOV TL0, # 4FH

- MOV R5, TH0

TIMER 1

D15	D14	D13	D12	D11	D10	D9	D8

TH1

D7	D6	D5	D4	D3	D2	D1	D0

TL1

TCON REGISTER

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

TF1= TIMER 1 OVERFLOW FLAG
TR1=TIMER 1 RUN CONTROL
TF0= TIMER 0 OVERFLOW FLAG
TR1= TIMER 0 RUN CONTROL
IE1= EXT INT. 1 EDGE FLAG
IT1= INT 1 TYPE CONTROL BIT
IE0= EXT. INT 0 EDGE FLAG
IT0- INT. 0 TYPE CONTROL BIT

T-MOD REGISTER

GATE	C/T	M1	M0	GATE	C/T	M1	MO
TIMER 1				TIMER 0			

GATE ------GATING CONTROL WHEN SET. T /C CAN BE ENABLED BY PUTTING INT PIN=1 & TRX =1 ,   WHEN O THE TIMER IS ENABLED WHEN THE TRX CONTROL PIN IS SET.

C/T : 0= TIMER (i/p from internal clock) , 1= COUNTER (i/p from TX pin.)
M1- MODE BIT 1
M0- MODE BIT 0

M1	M0	MODE	OPERATION
0	0	0	13 BIT TIMER MODE
0	1	1	16 BIT TIMER MODE
1	0	2	8 BIT AUTO RELOAD
1	1	3	SPLIT TIMER MODE

TIMER

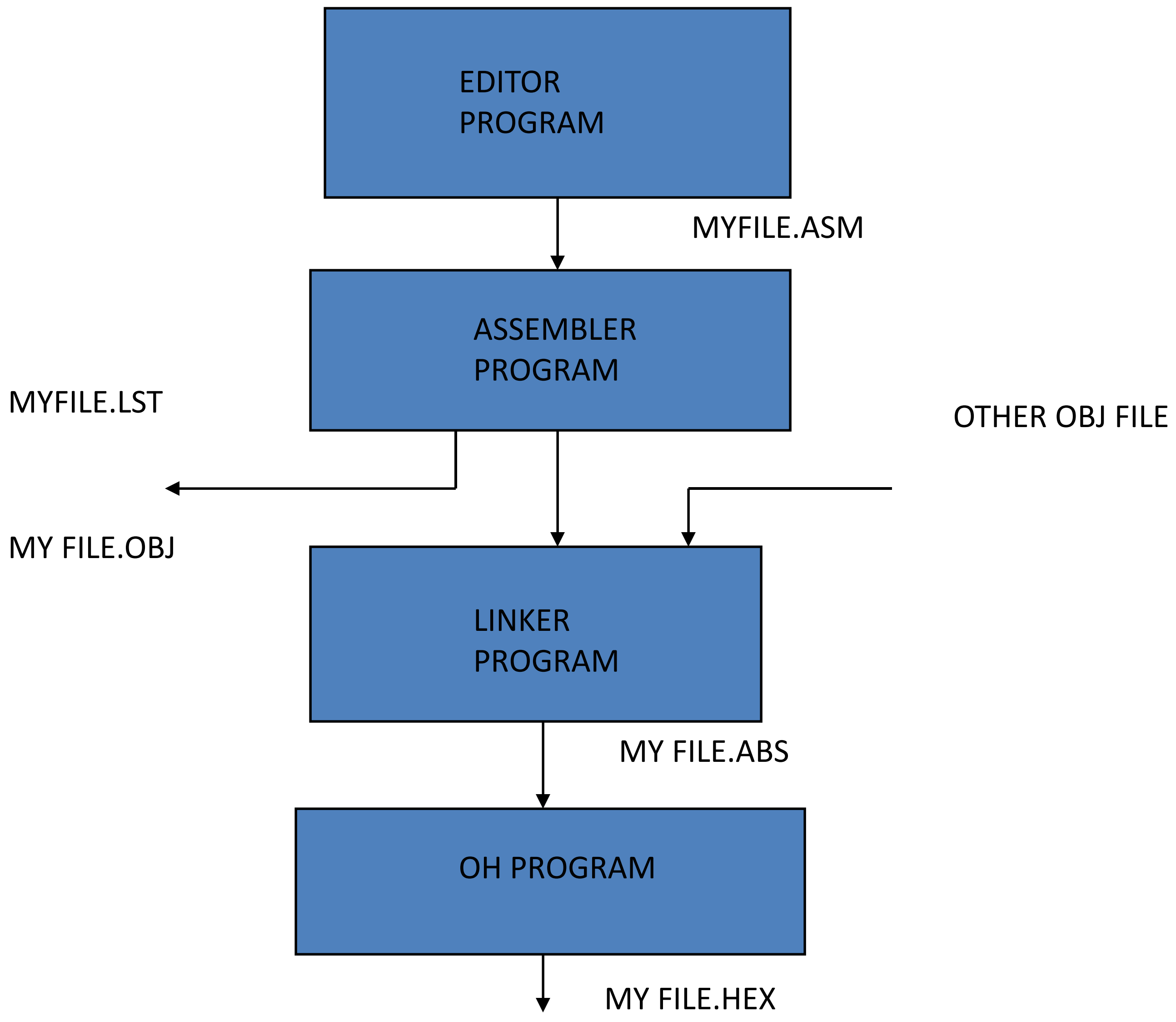
- Indicate which mode and which timer are selected for each of the following:

a) MOV TMOD, #01H

b) MOV TMOD, #20 h

c) MOV TMOD ,#12H

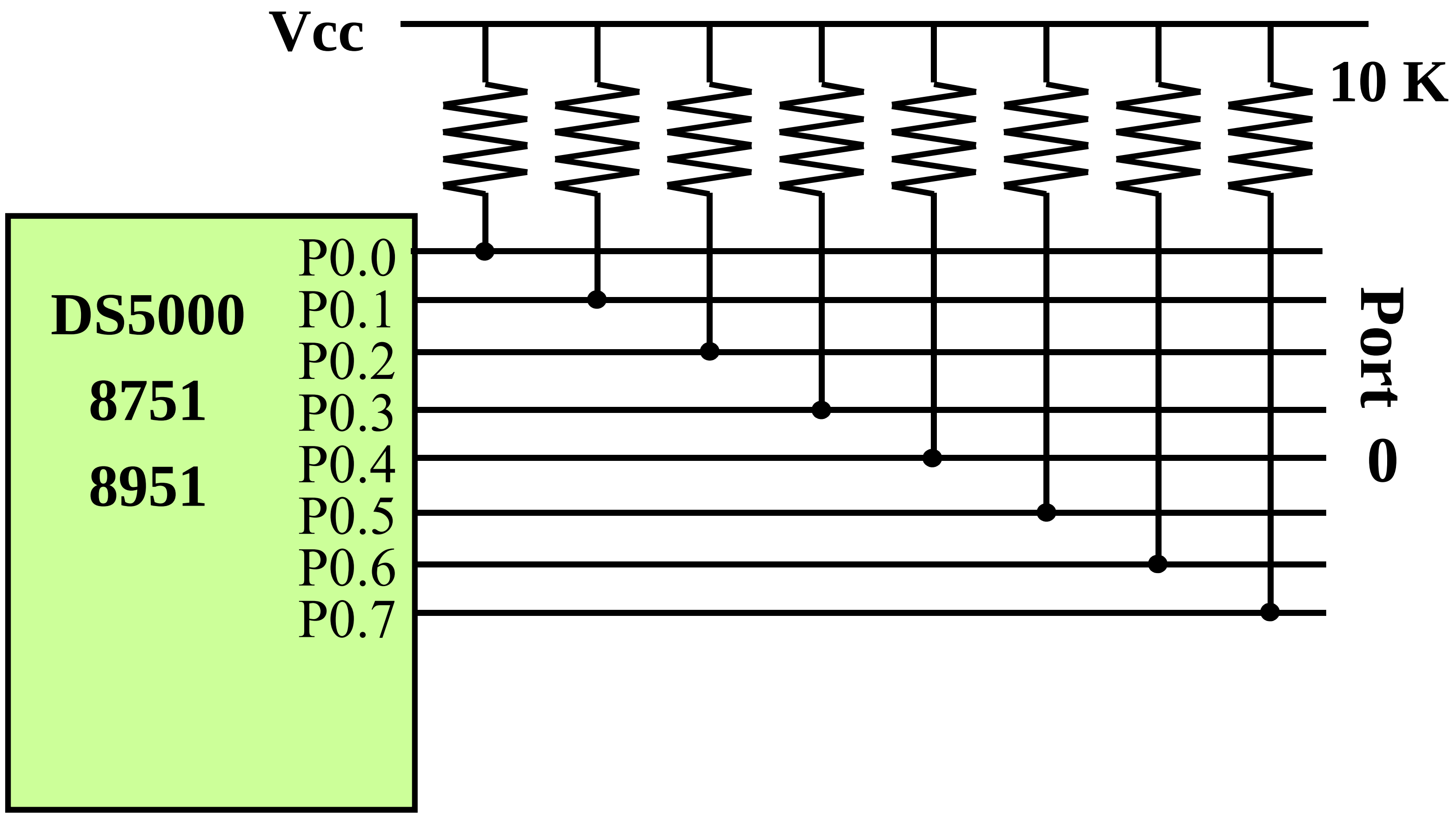
STEPS TO CREATE A PROGRAM



ASSEMBLER DIRECTIVES

- DB-(DEFINE BYTE)- IT IS USED TO DEFINE THE 8-BIT DATA. THE NUMBER CAN BE IN DECIMAL, BINARY, HEX OR ASCII FORMATS.
- ORG(ORIGIN)- IT IS USED TO INDICATE THE BEGINNING OF THE ADDRESS.
- EQU (EQUATE)–IT IS USED TO DEFINE A CONSTANT WITHOUT OCCUPYING A MEMORY LOCATION. IT ASSIGNS A CONSTANT VALUE TO A DATA LABEL.
- END- THIS INDICATES TO THE ASSEMBLER THE END OF THE SOURCE FILE.

Port 0 with Pull-Up Resistors



Port Operations

- Total 4 ports
- Port 0 may serve as inputs, outputs, or as a low order
- address and data bus for external memory.
- Port 1 may be used as input/output port.
- Port 2 may be used as input/output or high order address byte.
- Port 3 may be used as an input/output and for some alternate function.

PORT 3 ALTERNATE OPERATION

Pin Alternate use SFR

- P3.0-RXD Serial data input SBUF
- P3.1-TXD Serial data output SBUF
- P3.2-INT0~ Ext. Int. 0 TCON.1
- P3.3-INT1~ Ext. Int. 1 TCON.3
- P3.4-T0 Ext. Tim. 0 TMOD
- P3.4-T1 Ext. Tim. 1 TMOD
- P3.6-WR~ Ext. Mem write pulse
- P3.7-RD~ Ext. Mem Read pulse

Memory Mapped Port Operations

Store the address of the memory mapped port into the DPTR register, and indirectly address this through Acc.

Reading from a port

```
MOV DPTR, #0F012h; address to read digital
input
MOVX A, @DPTR ; read the values into Acc
```

Writing to a port

```
MOV DPTR, #0F011h; address of LED outputs
MOVX @DPTR, A ; write this value back out
again
```

Accessing external memory
Storage problems?
What happens if a program is larger than 4096 bytes of
code?
We have the ‘ROMless’ 8031 version, we must use the
movx
(move external) to access the data block, and movc to fetch
from code memory.
We have two separate read signals,
RD~, (read) and PSEN~, (program send enable).

Prototyping 8051 Design

For many small applications where we might want to
Change the ROM code, we would like to use *external*
EPROM , which means:
 We have lost the use of ports 0 and ports 2 since they
must be used to interface with external ROM and not
available for I/ O.
 We may need external RAM (since the 128 bytes may
not be enough for our application), which means that we
have lost port 3.6, WR~, and 3.7, RD~
 Any serial communication requires 3.0 (RXD) and
3.1(TXD)

Loss of ports leaves us with:

1. All of port #1, and
2. Port P3.2 – P3.5

for general purpose I/ O and external interrupts and timing inputs.

What is to be done for the lost ports?

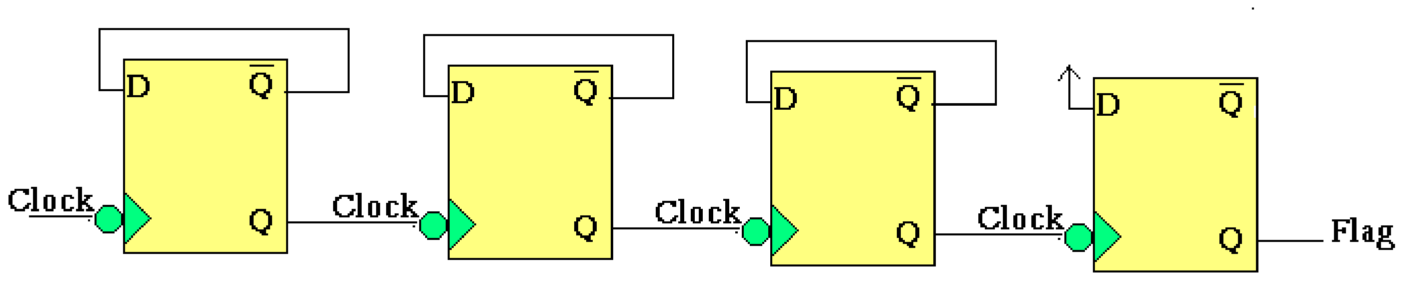
Timers and Counters

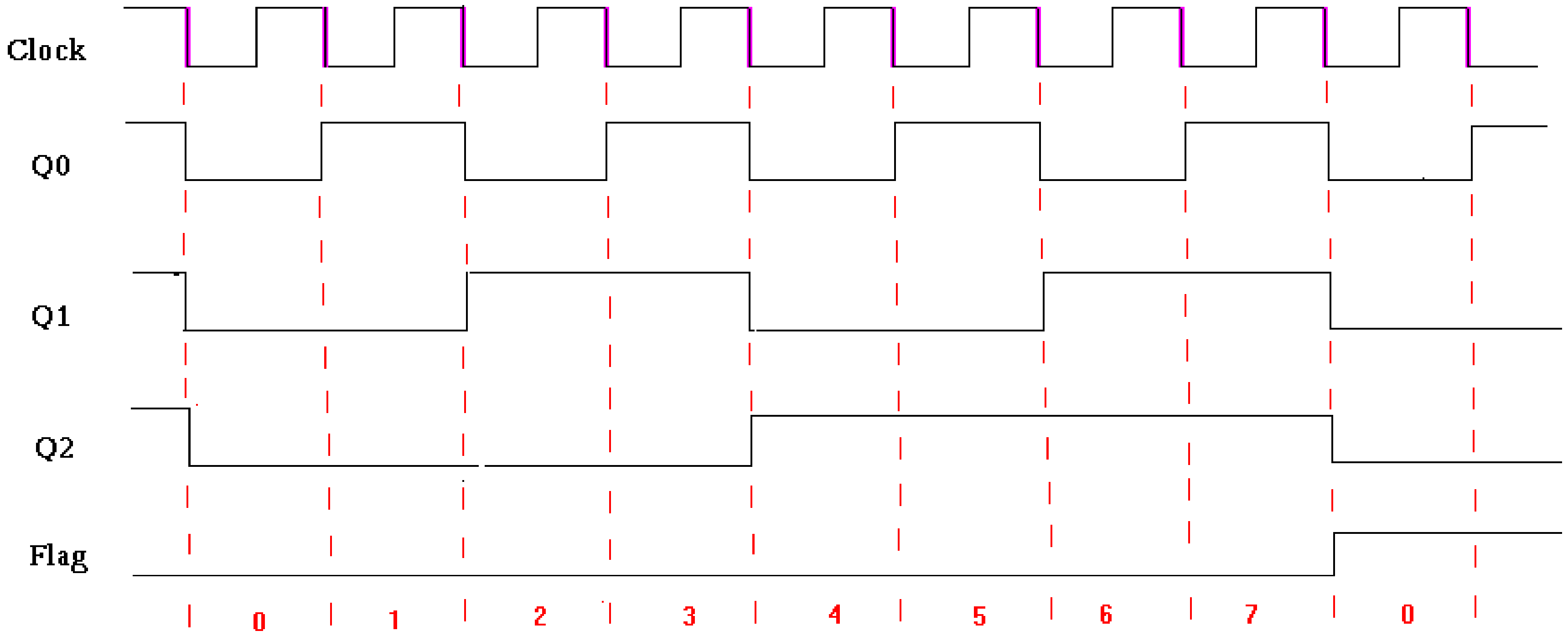
Two timers,   (set, read, and configured)

Functions:

1) Keeping time and/or calculating the amount of time between events,
2) Counting the events themselves, or
3) Generating baud rates for the serial port.

Timer:





Timers and Counters

Could use software techniques, but this keeps the processor occupied. Better to use interrupts & the two 16- bit *count- up timers. Can either be* programmed to:

1. count internal - acting as timer

2. count external - acting as counter

All counter action is controlled by the TMOD (timer mode register) and the TCON (timer/ counter control register).

Timers and Counters

TCON (Timer control SFR)contains timer 1& 2 overflow flags, external interrupt flags, timer control bits, falling edge/ Low level selector bit etc.

TMOD (timer mode SFR) is two four- bit registers
(timer #1, timer #0).
Select timer/ counter & various modes.)

Timers and Counters

Applications: (**counter**)

To count a specified number of events (clock pulses or external events), then

To configure as a counter:

1. **Store** the start number in the counter. (Value maxcountdesired count+ 1)
2. Counter automatically increments (in the background)
3. When it ***rolls over to zero****, it will set the* ***timer flag.***
4. **Test the flag** in the program, *or* ***generate an interrupt .***

Timers and Counters

Applications: (Timer)

Timing configures the **counter to count the internal clock frequency/ 12.**

To configure as a timer:

1. **Clear C/T~** bit in TMOD (Count internal frequency)
2. **Set TRx** in the TCON (timer run) *and the **gate bit** in the* **TMOD must be 0,** or the external pin INTx~ must be 1.
3. **Select** one of 4 modes.

Numerical Bases Used in Programming

- Hexadecimal
- Binary
- BCD

Hexadecimal Basis

- Hexadecimal Digits:

1 2 3 4 5 6 7 8 9 A B C D E F

A=10

B=11

C=12

D=13

E=14

F=15

Decimal, Binary, BCD, & Hexadecimal Numbers

$(43)_{10} =$

$(0100\ 0011)_{BCD} =$

$$\begin{array}{cccc}
(& \boxed{0010} & \boxed{1011} &)_2 = \\
 & \Downarrow & \Downarrow & \\
(& 2 & B &)_{16}
\end{array}$$

ADDRESSING MODES

- Immediate addressing
- Register addressing
- Direct addressing
- Register Indirect addressing
- Indexed addressing

Register Addressing Mode

MOV Rn, A ;n=0,..,7

ADD A, Rn

MOV DPL, R6

~~MOV DPTR, A~~

~~MOV Rm, Rn~~

Direct Addressing Mode

Although the entire of 128 bytes of RAM can be accessed using direct addressing mode, it is most often used to access RAM loc. 30 – 7FH.

```
MOV R0, 40H
MOV 56H, A
MOV A, 4            ; ≡ MOV A, R4
MOV 6, 2            ; copy R2 to R6
                    ; MOV  R6,R2 is invalid !
```

Immediate Addressing Mode

```
MOV A,#65H

MOV R6,#65H

MOV DPTR,#2343H

MOV P1,#65H
```

SETB bit ; bit=1
CLR bit ; bit=0

```
SETB    C           ; CY=1
SETB    P0.0        ;bit 0 from port 0 =1
SETB    P3.7        ;bit 7 from port 3 =1
SETB    ACC.2       ;bit 2 from ACCUMULATOR =1
SETB    05          ;set high D5 of RAM loc. 20h
```

Note:

CLR instruction is as same as SETB
i.e.:

```
    CLR         C       ;CY=0
```

But following instruction is only for CLR:

```
    CLR         A       ;A=0
```

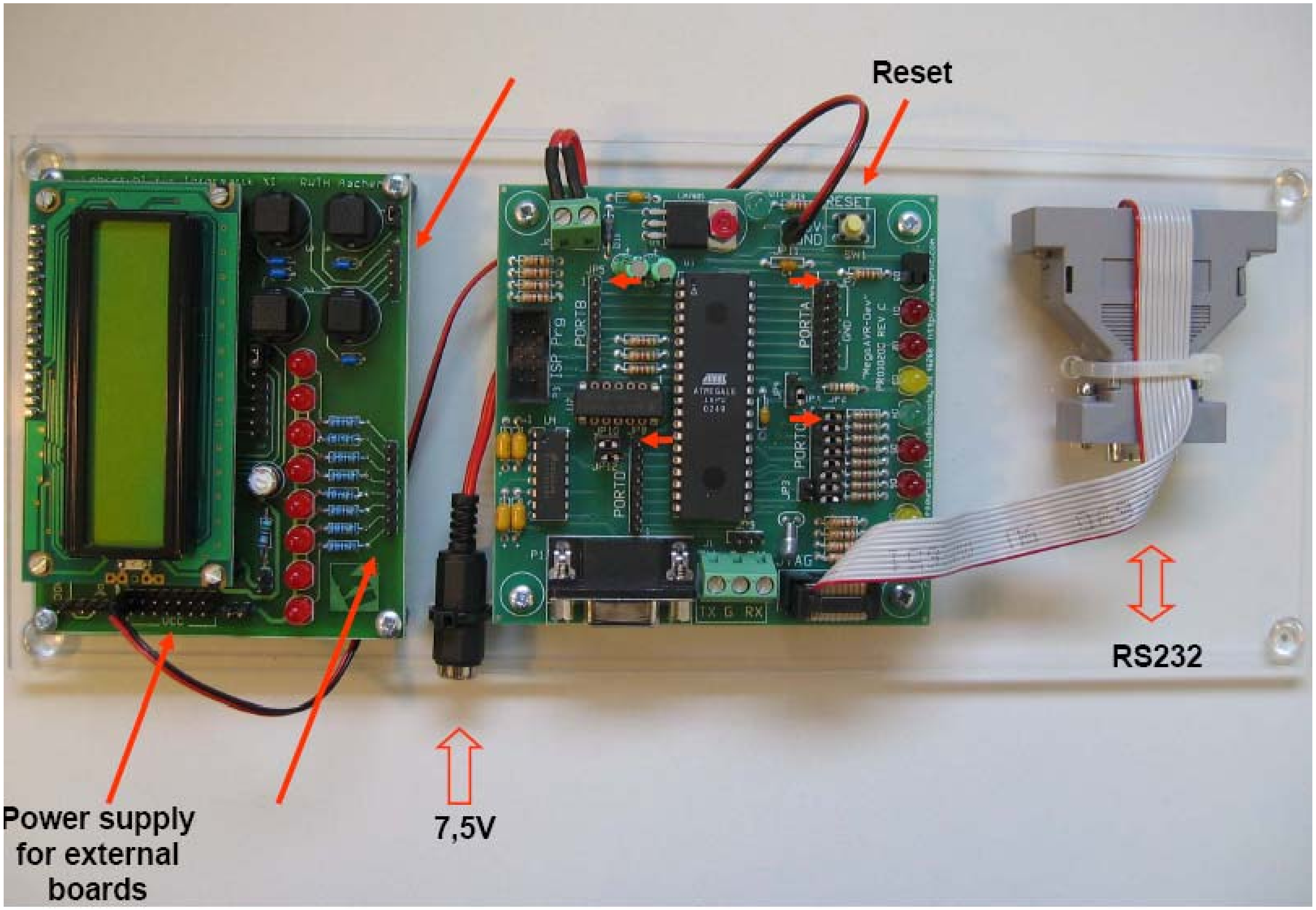
```
DEC     byte      ;byte=byte-1
INC     byte      ;byte=byte+1

INC     R7
DEC     A
DEC     40H       ; [40]=[40]-1
```

LOOP and JUMP Instructions

Conditional Jumps :

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


Reset
RS232
7,5V
Power supply for external boards