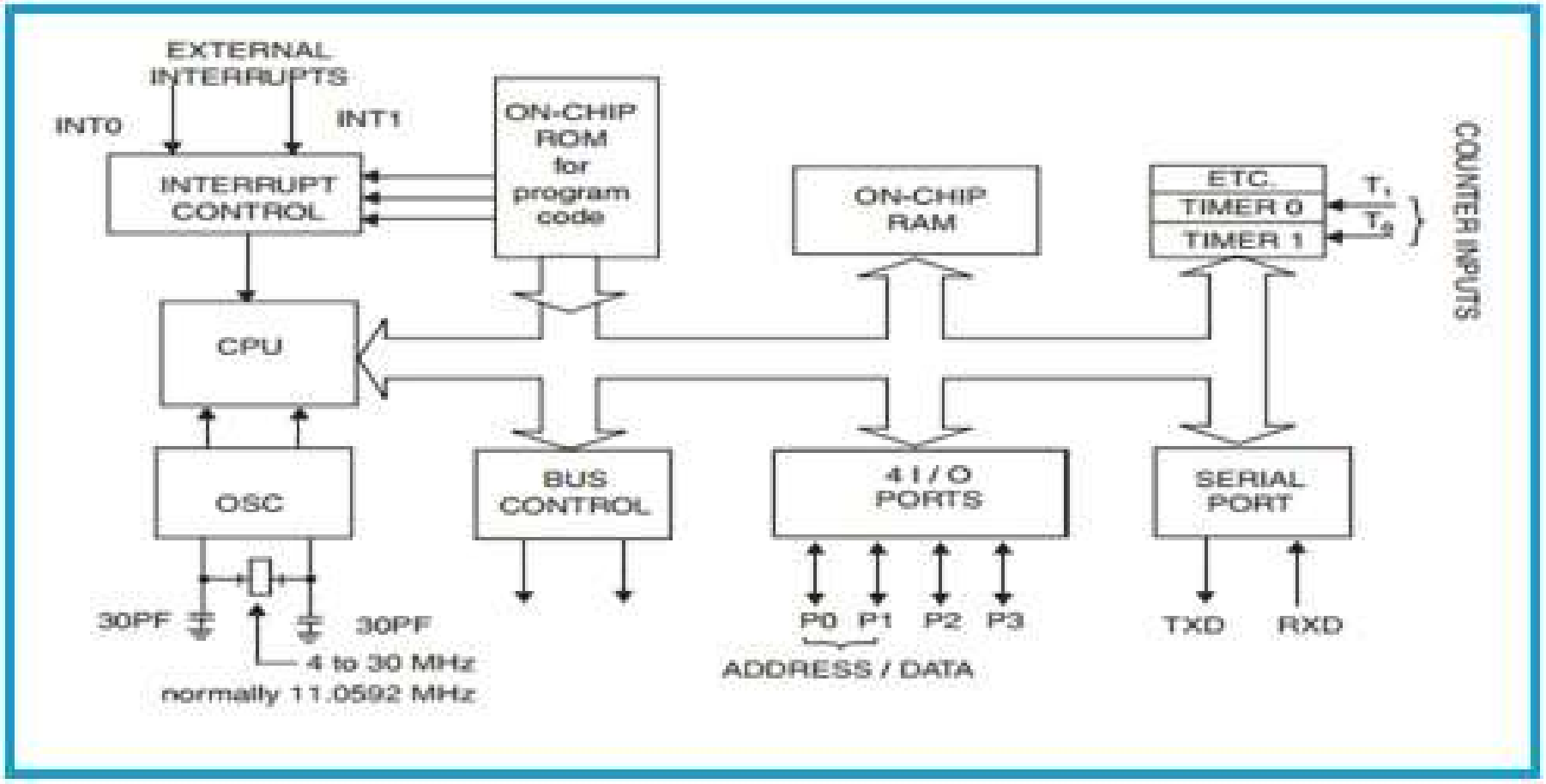
Micro-controllers: Moving on to 8051 from 8085

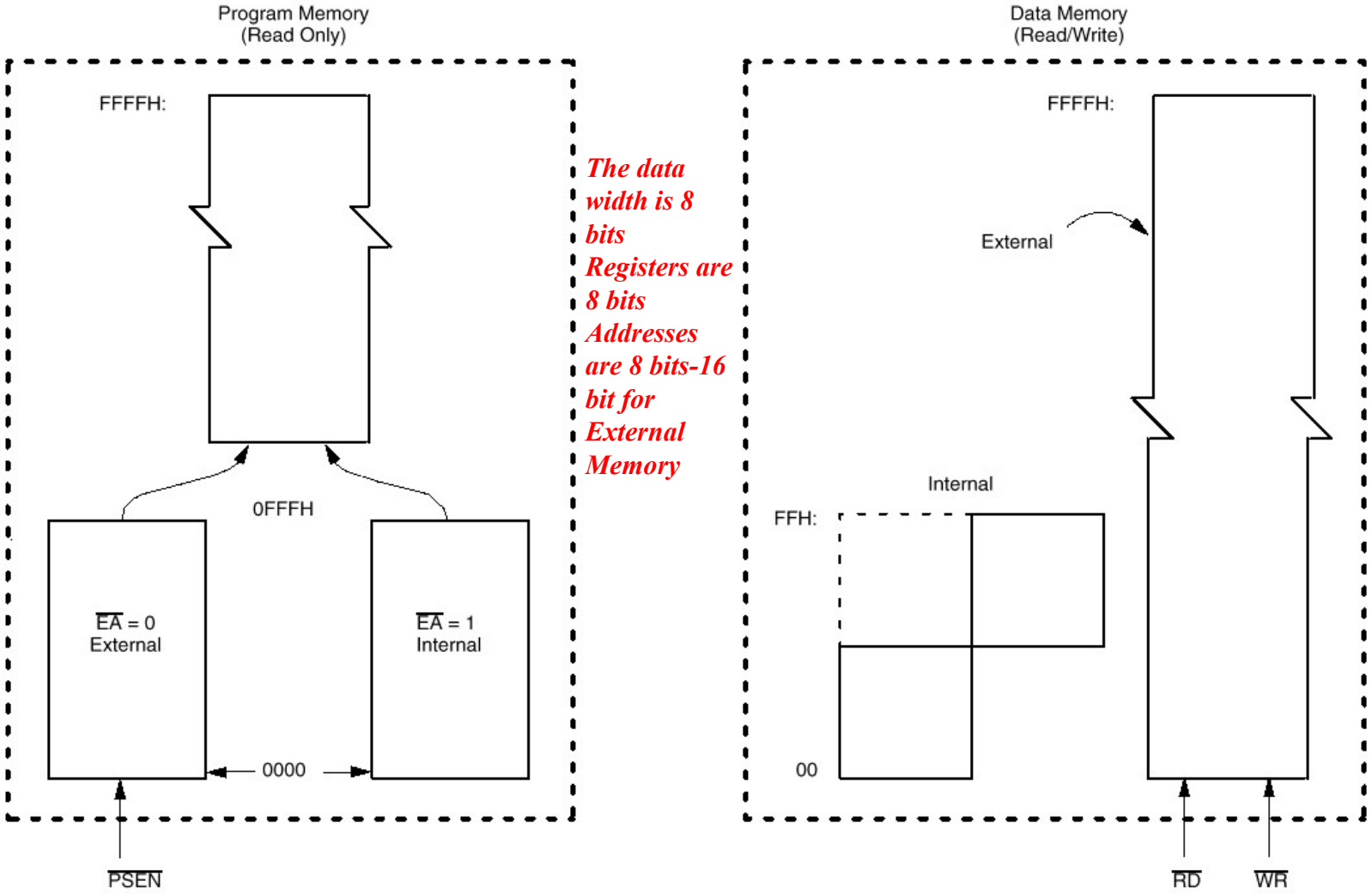
Specifications	8051	8085
Device Type	It is a microcontroller.	It is a microprocessor.
Timers/Counters	YES	No, Need to be connected externally.
Data bus width	8 lines	8 lines
Address Bus lines	16	16
DMA Access signals	NO	YES, Has HOLD and HLDA signals
Internal RAM, ROM	YES	No, need to be connected externally with 8085 chip, if data and code size requirement is more.

https://courses.cs.washington.edu/courses/cse477/00sp/ppt/MicrocontrollersII.ppt

8051 Architecture

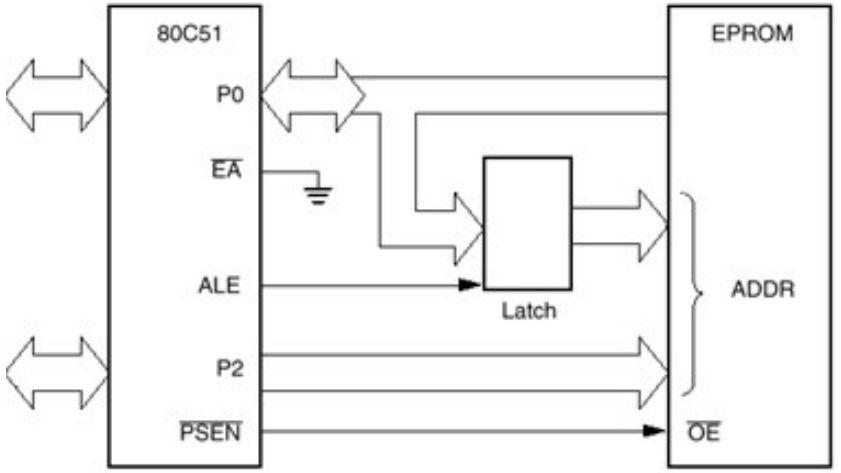


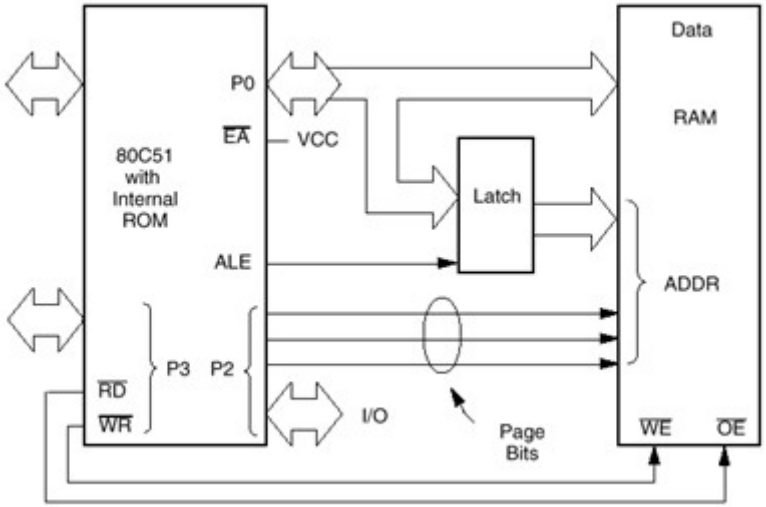
Memory Organization in 8051



The data width is 8 bits
Registers are 8 bits
Addresses are 8 bits-16 bit for External Memory

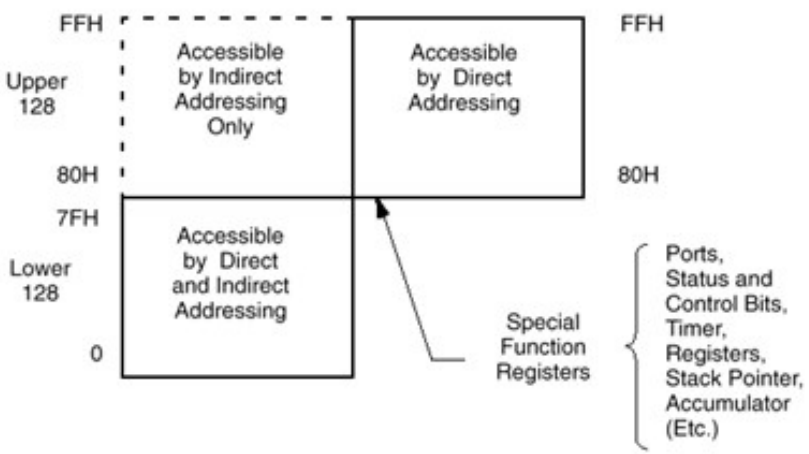
Accessing Internal and External Memory

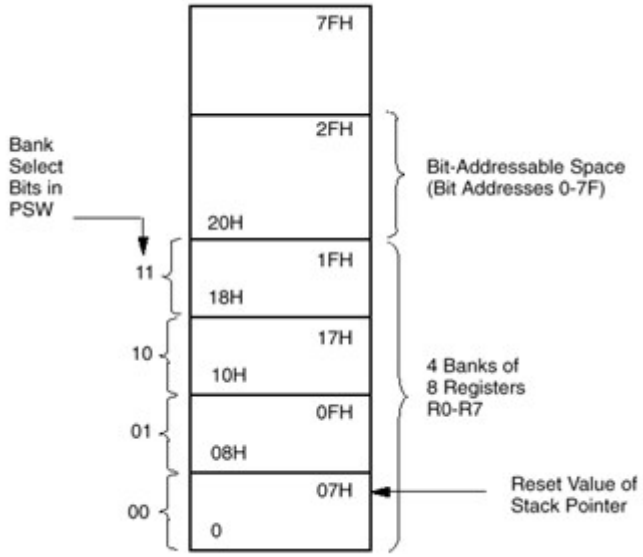




Program and Data memory are separate
Program Memory can be internal and/or external.
The ROM contains constant data and Instructions.

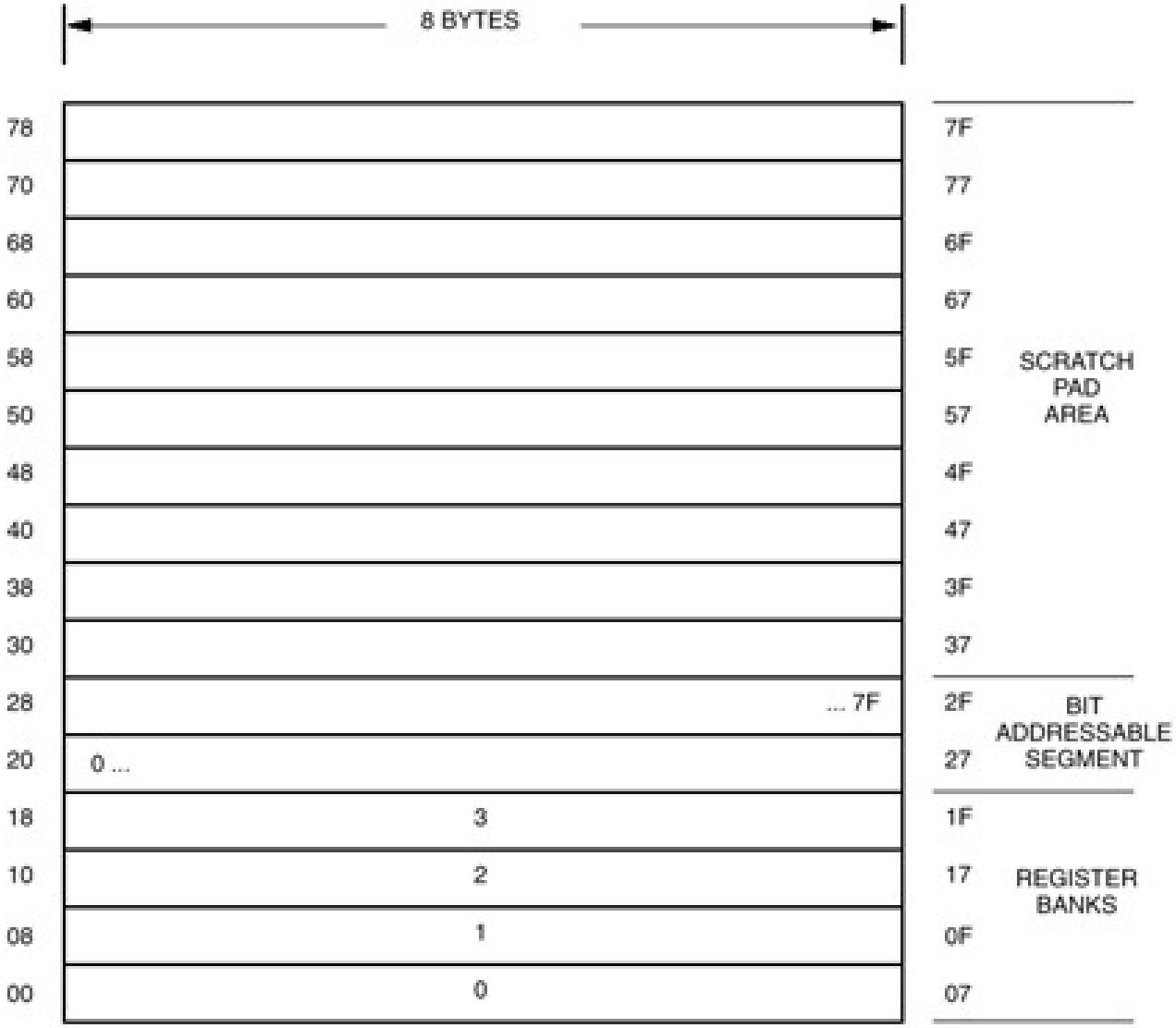
External Data - xdata
Resides off-chip
Accessed using the DPTR and MOVX instructions





(i)Lower 128 bytes: registers, general data
(ii)Upper 128 bytes:
a. indirectly addressed: 128 bytes, used for the stack (small!)
b.directly addressed: 128 bytes for "special" functions

Lower 128 Bytes



The 32 bytes (00 to 1Fare divided into 4 register banks in which each bank has 8 registers, R0–R7. RAM locations from 0 to 7 are set aside for bank 0 of R0–R7 where R0 is RAM location 0, R1 is RAM location 1, R2 is location 2, and so on, until the memory location 7, which belongs to R7 of bank 0.

Bitwise manipulation: flags

Upper 128 Bytes(SFRs)

SYMBOL	DESCRIPTION	DIRECT ADDRESS	BIT ADDRESS, SYMBOL, OR ALTERNATIVE PORT FUNCTION MSB							LSB	RESET VALUE
ACC*	Accumulator	E0H	E7	E6	E5	E4	E3	E2	E1	E0	00H
B*	B register	F0H	F7	F6	F5	F4	F3	F2	F1	F0	00H
DPTR	Data pointer (2 bytes)										
DPH	Data pointer high	83H									00H
DPL	Data pointer low	82H									00H
			AF	AE	AD	AC	AB	AA	A9	A8	
IE*	Interrupt enable	A8H	EA	–	–	ES	ET1	EX1	ET0	EX0	0x000000B
			BF	BE	BD	BC	BB	BA	B9	B8	
IP*	Interrupt priority	B8H	–	–	–	PS	PT1	PX1	PT0	PX0	xx000000B
			87	86	85	84	83	82	81	80	
P0*	Port 0	80H	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0	FFH
			97	96	95	94	93	92	91	90	
P1*	Port 1	90H	–	–	–	–	–	–	T2EX	T2	FFH
			A7	A6	A5	A4	A3	A2	A1	A0	
P2*	Port 2	A0H	A15	A14	A13	A12	A11	A10	A9	A8	FFH
			B7	B6	B5	B4	B3	B2	B1	B0	
P3*	Port 3	B0H	$\overline{RD}$	$\overline{WR}$	T1	T0	$\overline{INT1}$	$\overline{INT0}$	TxD	Rxd	FFH
PCON[1]	Power control	87H	SMOD	–	–	–	GF1	GF0	PD	IDL	0xxxxxxB
			D7	D6	D5	D4	D3	D2	D1	D0	
PSW*	Program status word	D0H	CY	AC	F0	RS1	RS0	OV	–	P	00H
SBUF	Serial data buffer	99H									xxxxxxxB
			9F	9E	9D	9C	9B	9A	99	98	
SCON*	Serial controller	98H	SM0	SM1	SM2	REN	TB8	RB8	TI	RI	00H
SP	Stack pointer	81H									07H
			8F	8E	8D	8C	8B	8A	89	88	
TCON*	Timer control	88H	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	
TH0	Timer high 0	8CH									00H
TH1	Timer high 1	8DH									00H
TL0	Timer low 0	8AH									00H
TL1	Timer low 1	8BH									00H
TMOD	Timer mode	89H	GATE	C/$\overline{T}$	M1	M0	GATE	C/$\overline{T}$	M1	M0	00H

The 8051 family of microcontrollers provides a distinct memory area for accessing Special Function Registers (SFRs). SFRs are used in a program to control timers, counters, serial I/Os, port I/Os, and peripherals..

The Processor Status Word(PSW) and Register Selection

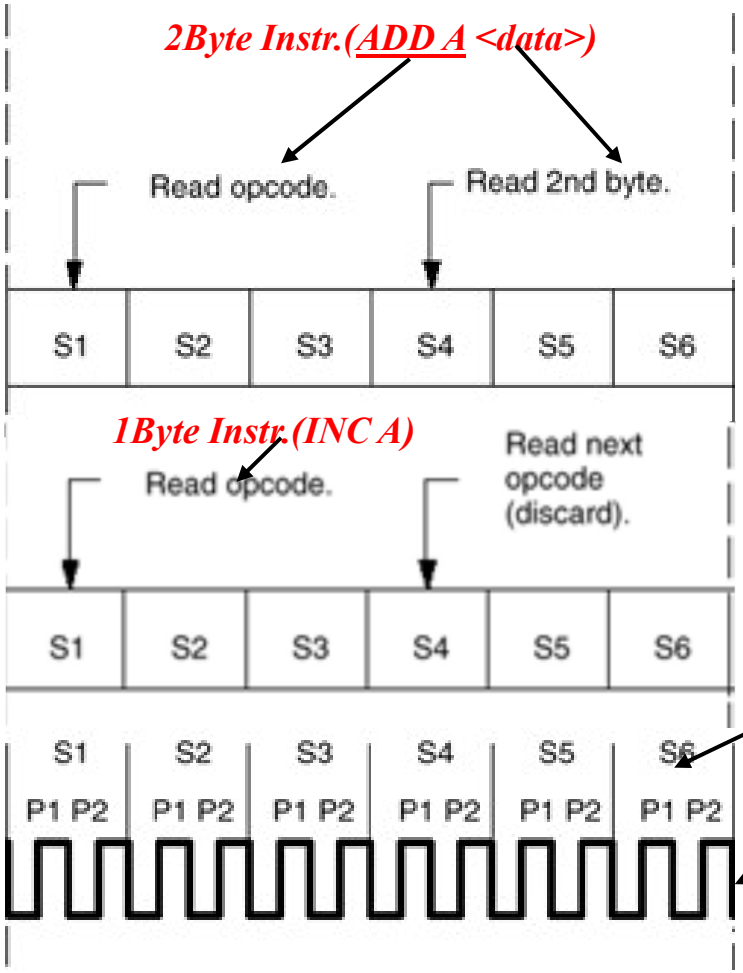
CY	AC	F0	RS1	RS0	OV	-	P

CY	PSW.7	Carry Flag
AC	PSW.6	Auxiliary Carry Flag
F0	PSW.5	Flag 0 available to user for general purpose.
RS1	PSW.4	Register Bank selector bit 1
RS0	PSW.3	Register Bank selector bit 0
OV	PSW.2	Overflow Flag
-	PSW.1	User definable FLAG
P	PSW.0	Parity FLAG. Set/ cleared by hardware during instruction cycle to indicate even/odd number of 1 bit in accumulator.

RS1	RS2	Register Bank	Address
0	0	0	00H-07H
0	1	1	08H-0FH
1	0	2	10H-17H
1	1	3	18H-1FH

addresses 0x80, 0x88, 0x90, . . . , 0xF8 are bit addresable

Instruction Timing



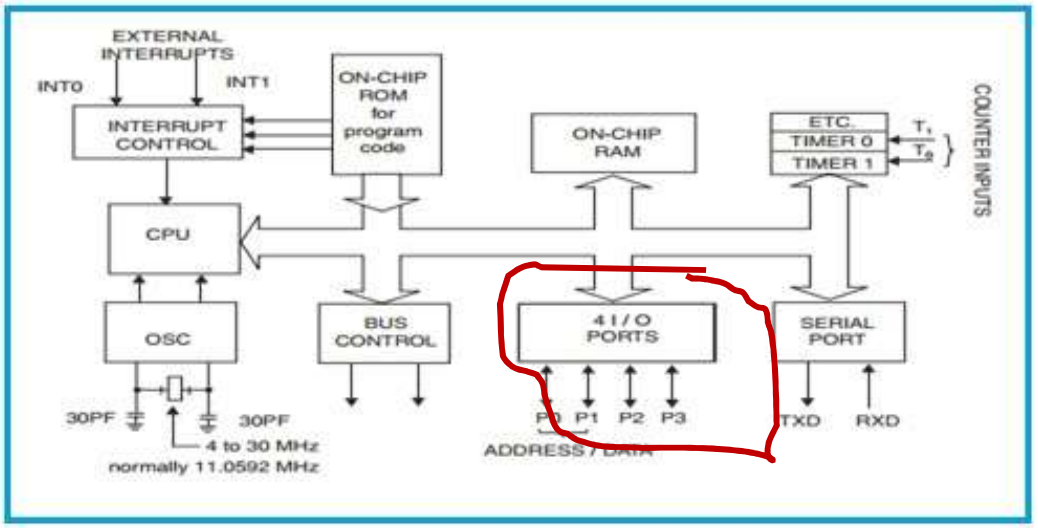
An Instruction can take 1-4 Machine Cycles(MCs)

Each MC consists of 6 states S1-s6

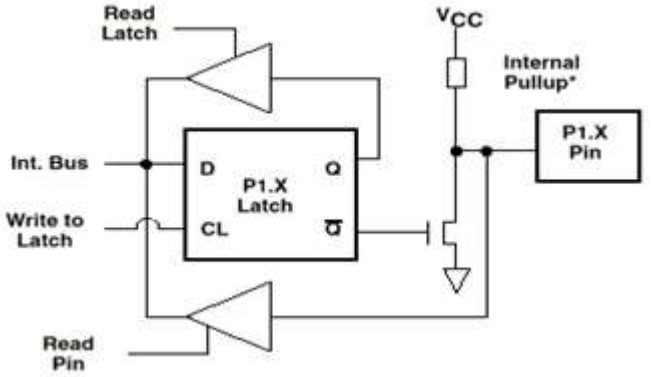
Each state consists of 2 clock cycles

Ports

Port 0 - external memory access
low address byte/data
Port 1 - general purpose I/O
pins 0, 1 for timer/counter 2
Port 2 - external memory access
high address byte
Port 3 - Special features
0 - RxD: serial input
1 - TxD: serial output
2 - INT0: external interrupt
3 - INT1: external interrupt
4 - T0: timer/counter 0 external input
5 - T1: timer/counter 1 external input
6 - WR: external data memory write strobe
7 - RD: external data memory read strobe



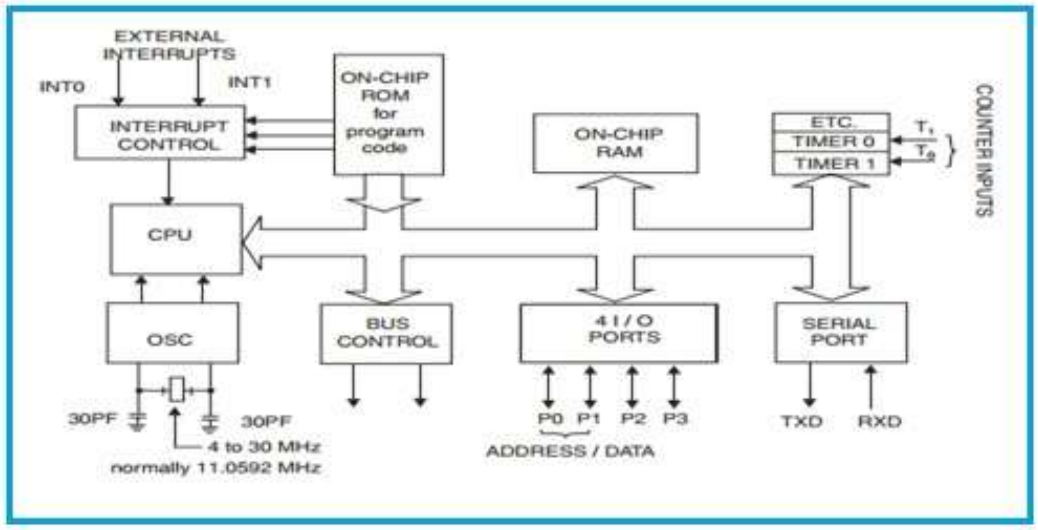
Port 0 - bi-directional
Port 1-3 - have internal pullups that will source current
Output pints: 0/1 to the bit/byte
Input pins Output latch must have a 1 (reset state)
Turns off the pulldown
pullup must be pulled down by external driver
bit/byte needs to be read



Refer slide on SFRs here

Timers

8051 has 2 timers: an extra Timer in some
Timers can be accessed ***directly***
They operate in two modes: ***Timer*** and *Counter.* ***TMOD*** *and* ***TCON*** *Registers in SFR are used*

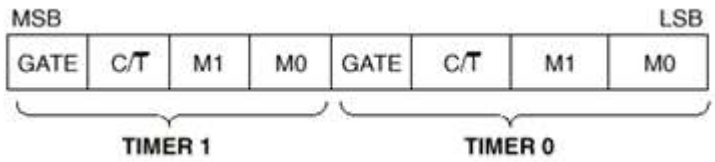


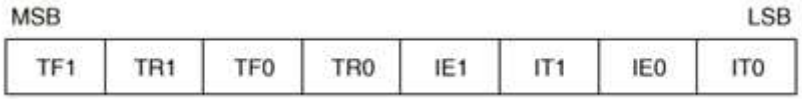
In the timer mode, the internal machine cycles are counted. This register is incremented in each machine cycle i.e. 12 clock cycles. For example,if the clock frequency is 12MHz, then the timer register is incremented in each millisecond. In this mode it ignores the external timer input pin

In the counter mode, the external events are counted. In this mode, the timer register is incremented for each 1 to 0 transition of the external input pin. This type of transitions is treated as events. The external input pins are sampled once in each machine cycle, and to determine the 1or 0 transitions, another machine cycle will be needed. So in this mode, at least two machine cycles are needed. When the frequency is12MHz, then the maximum count frequency will be 12MHz/24 = 500KHz. So for event counting the time duration is 2 µs.

Explain with a suitable example

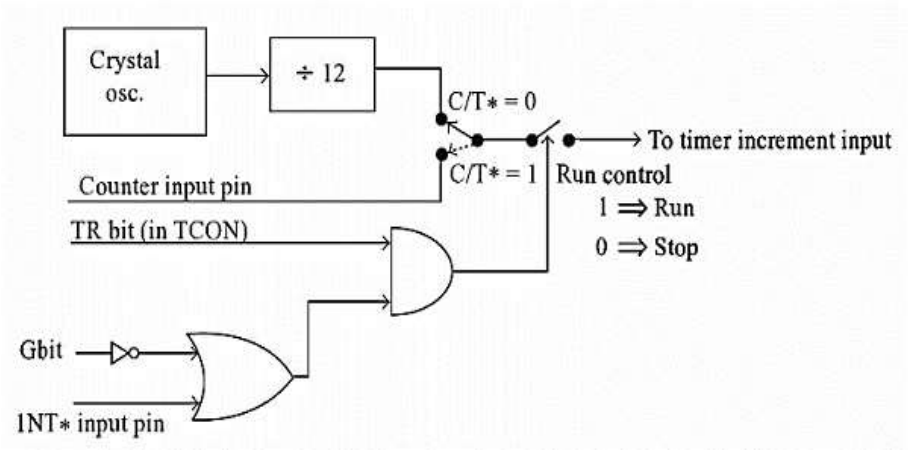
Timers: Using the TMOD& TCON





GATE — Gating control when set. Timer/Counter "x" is enabled only while "$\overline{INTx}$" pin is high and "TRx" control pin is set. when cleared Timer "x" is enabled whenever "TRx" control bit is set.

C/$\overline{T}$ — Timer or Counter Selector cleared for Timer operation (input from in=ternal system clock.) Set for Counter operation (input from "Tx" input pin).

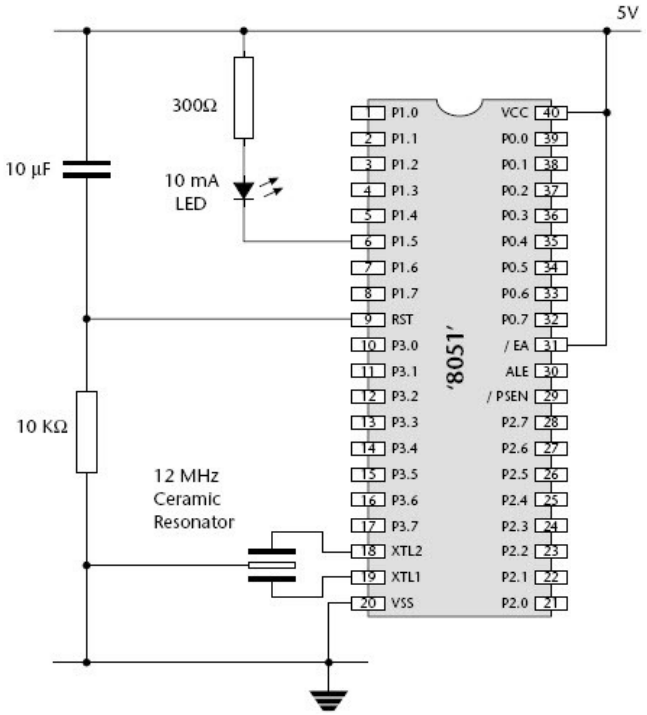


low value.

Bit Details	High Value(1)	Low Value(0)
C/$\overline{T}$	Configure for the Counter operations	Configure for the Timer operations
Gate (G)	Timer0 or Timer1 will be in RunMode when TRX bit of TCON register is high.	Timer0 or Timer1 will be in RunMode when TRX bit of TCON register is high and $\overline{INT0}$ or $\overline{INT1}$ is high.

Bit Details	**00**	**01**	**10**	**11**
M1 M0	This is for Mode 0. (8-bit timer/counter, with 5-bit pre-scaler)	This is Mode 1. (16-bit timer/counter)	This is Mode 3 (8-bit auto reload-timer/counter)	This is Mode 3 (The function depends on Timer0 or Timer1)

Programming The C interface (KEIL)



https://www.keil.com/c51/default.asp

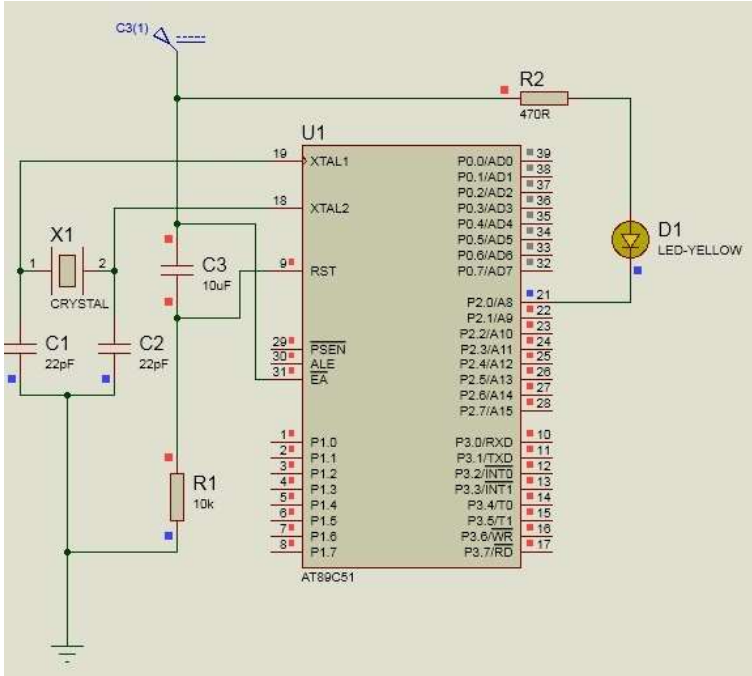
Usually the board comes with an IDE with an Application Programmers' Interface(API) for a version of C language called the Embedded C or its equivalent. The IDE can be downloaded on a PC/Laptop, the program can be written and debugged and then loaded for running on a micro-controller biard. This is the standard practice for most modern micro-controllers

Additional Embedded C Variable Types sbit,bit,SFR,volatile

Example:

```
SFR  P0=0x80 ; /*defines Port0 */
sbit L0= P0^1 ;/* Pin 1 of Port 0 is connected
                  an LED and this is variable
                  L0. We manipulate the pin
                  using this variable.
```

Programming with the C interface: Blinking LED



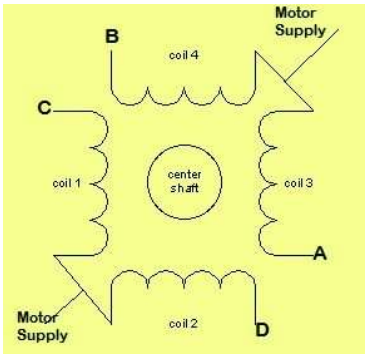

```
#include<reg51.h>
sbit L_pin = P2^0; /*set the LED pin as P2.0*/

void delay(int ms){
  unsigned int i, j;
  for(i = 0; i< ms; i++)
   {
    /* Outer for loop for given milliseconds value*/
    for(j = 0; j < 1275; j++)
    {
      /*execute in each millisecond*/
    }
  }
}
void main(){
  while(1){
    /*infinite loop for LED blinking*/
    L_pin = 0;
    delay(500); /*wait for 500 milliseconds*/
    L_pin = 1;
    delay(500); /*wait for 500 milliseconds*/
  }
}
```

LED connected to Pin 1 of Port 2

Q: Cany you guess the clock speed?

https://www.tutorialspoint.com/programming-8051-using-keil-software

Programming with the C interface: Stepper Motor Control



***Wave drive*: Only one coil is energized duringa time-slot(step). Low torque, low power**

Step	A	B	C	D
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1

***Full drive*: Only one coil is energized duringa time-slot(step). More torque, More power**

Step	A	B	C	D
1	1	1	0	0
2	0	1	1	0
3	0	0	1	1
4	1	0	0	1

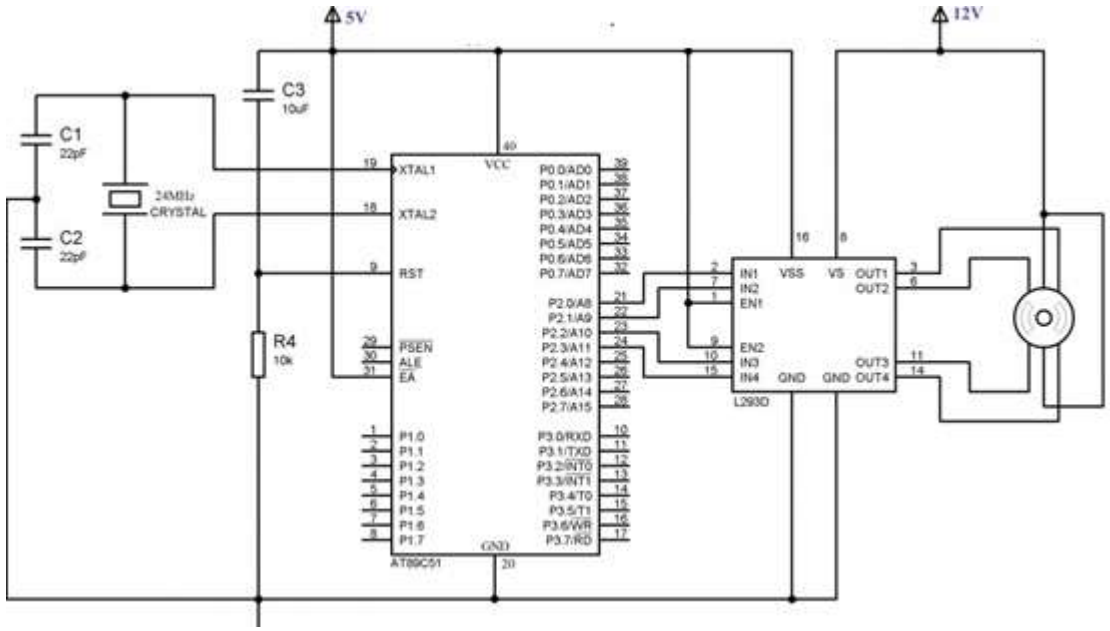
***Half drive*: Wave and Full in alternate steps. Smaller angular step-size**

Step	A	B	C	D
1	1	0	0	0
3	0	1	0	0
5	0	0	1	0
7	0	0	0	1

Step	A	B	C	D
2	1	1	0	0
4	0	1	1	0
6	0	0	1	1
8	1	0	0	1

https://electrosome.com/interfacing-stepper-motor-8051-keil-c-at89c51/

Programming with the C interface: Stepper Motor Control contd.



L293D is connected to pins P2.0, P2.1, P2.2, P2.3 of the microcontroller and two pairs of L293D are enabled by tieing EN1, EN2 to 5V. Logic Voltage (5V) is connected to Vss pin and Motor Supply (12V) is connected to the Vs pin of L293D. Each winding of the motor can be energized by making corresponding pin of L293D LOW.

The L293D is designed to provide bidirectional drive currents of up to 600-mA at voltages from 4.5 V to 36 V. It can drive relays, solenoids and in this case a stepper motor. Q: What is Vmax in this case?

https://electrosome.com/interfacing-stepper-motor-8051-keil-c-at89c51/

Programming with the C interface: Stepper Motor Control contd.

```
#include<reg52.h>
#include<stdio.h>
/*Code for Wave Control*/
void delay(int);
void main()
{
 while(1)
 {
  P2=0x01; /* Directly manipulating port */
  delay(1000);
  P2=0x02;
  delay(1000);
  P2=0x04;
  delay(1000);
  P2=0x08;
  delay(1000);
 }
}

void delay(int k)
{
 int i,j;
 for(i=0;i<k;i++)
 {
  for(j=0;j<100;j++)
  {}
 }
}
```

```
#include<reg52.h>
#include<stdio.h>
/*Code for Full
Drive*/
void delay(int);

void main()
{
 while(1)
 {
  P2 = 0x03;
  delay(1000);
  P2 = 0x06;
  delay(1000);
  P2 = 0x0C;
  delay(1000);
  P2 = 0x09;
  delay(1000);
 }
}
```

Half Drive

```
while()
 {
  P2=0x01;
  delay(1000);
  P2=0x03;
  delay(1000);
  P2=0x02;
  delay(1000);
  P2=0x06;
  delay(1000);
  P2=0x04;
  delay(1000);
  P2=0x0C;
  delay(1000);
  P2=0x08;
  delay(1000);
  P2=0x09;
  delay(1000);
 }
```

Can you deduce the speed for Half Drive?

https://electrosome.com/interfacing-stepper-motor-8051-keil-c-at89c51/