

# Assembly Language Programming

**Wichit Sirichote**

Applied Physics Department

Faculty of Science

King Mongkut's Institute of Technology Ladkrabang

e-mail: [kswichit@kmitl.ac.th](mailto:kswichit@kmitl.ac.th)

URL: [www.kmitl.ac.th/~kswichit](http://www.kmitl.ac.th/~kswichit)

---

# What's Assembly Language?

---

; my first program

**\$MOD51**

LED equ P1.7

**CSEG AT 8000H**

**jmp 8100h**

**org 8100h**

PULSE: **CPL LED**

**NOP**

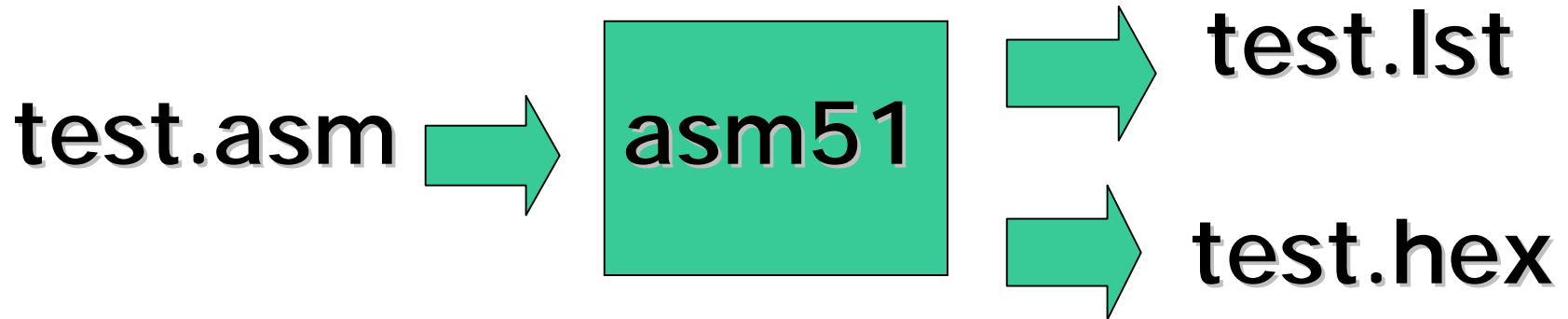
**jmp PULSE**

**END**

- **Assembler Controls**
- **Assembler Directives**
- **Instructions**
- **Mnemonics**
- **Operators**

# ASM51 Assembler

---



# Test.asm

---

```
CSEG AT 8000H  
jmp 8100h
```

```
org 8100h
```

```
PULSE: CPL P1.7  
NOP  
jmp PULSE
```

```
END
```

# Test.lst

---

```
8000          5          CSEG AT 8000H
8000 028100    6          jmp 8100h
              7
8100          8          org 8100h
              9
8100 B297     10        PULSE: CPL P1.7
8102 00       11          NOP
8103 80FB     12          jmp PULSE
              13
              14          END
```

# Test.hex

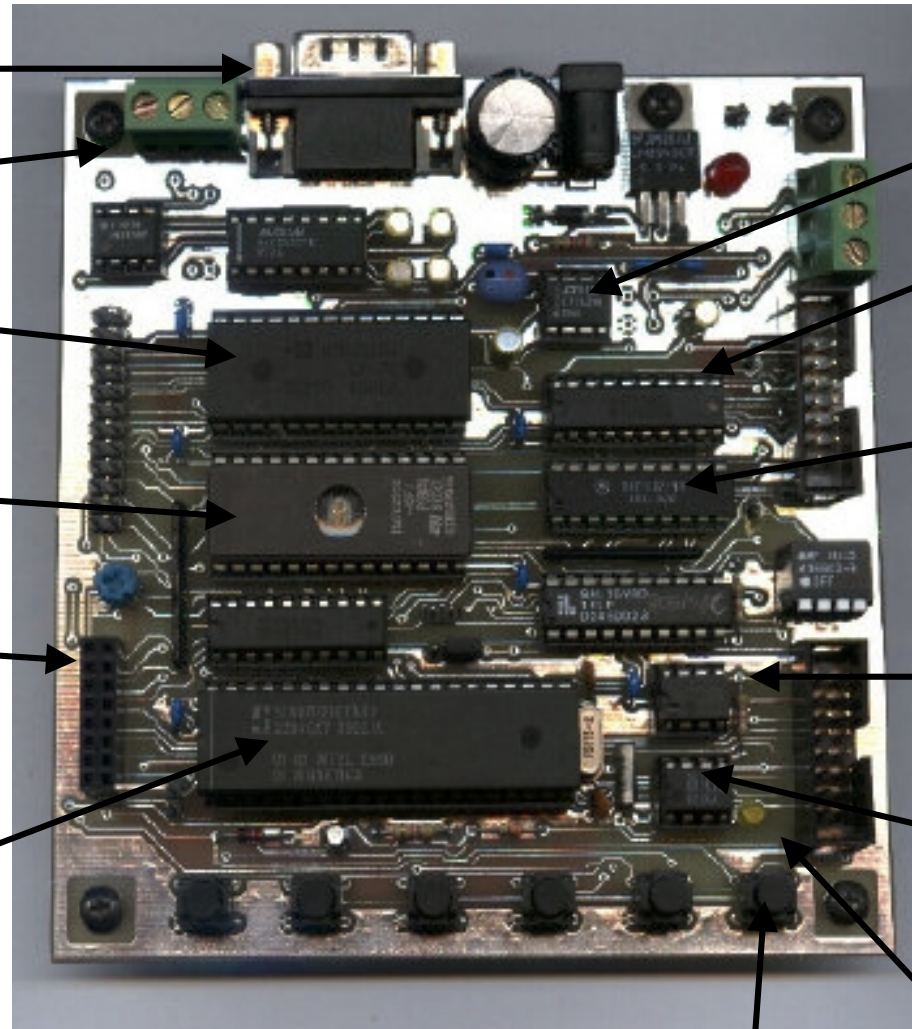
---

```
:03800000028100FA
:05810000B2970080FBB6
:00000001FF
:03 8000 00 02 81 00 FA
:05 8100 00 B2 97 00 80 FB B6
:00 0000 01 FF
```

FA byte check  
sum= two's  
complement  
of summing  
from 03 - 00

: begin of record  
03 number of byte  
8000 load address  
00 record type, 01 end of record

# 8051SBC Microprocessor Learning Board



RS232C

RS485

32kB SRAM

32kB EPROM

LCD connector

8051 CPU

12-bit ADC

8-bit input port

8-bit output port

32kB EEPROM

RTC chip

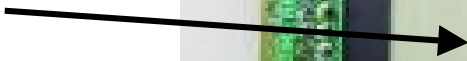
Reset

Debug LED 7

# 8051SBC Microprocessor Learning Board

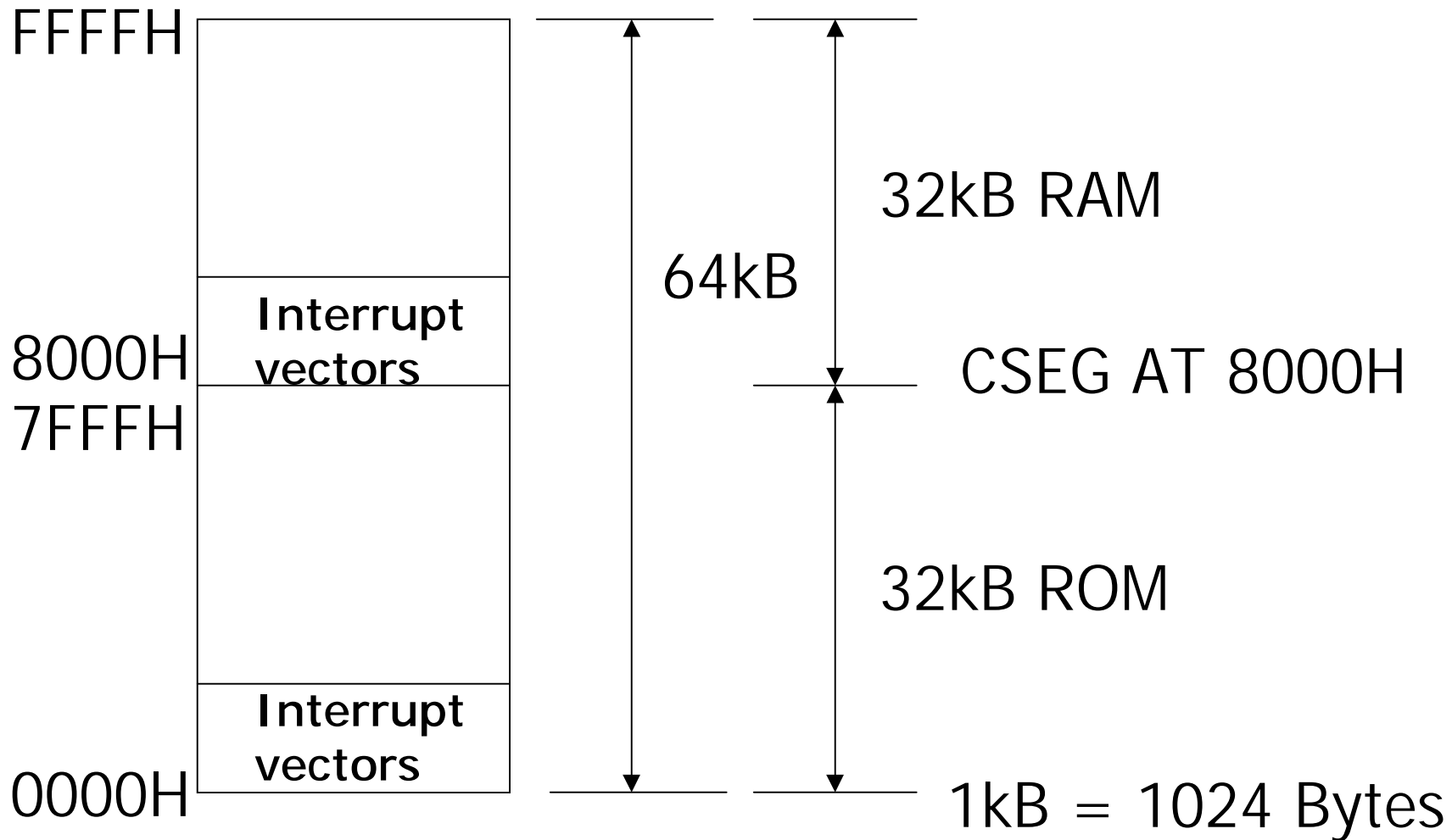
---

16x2  
LCD



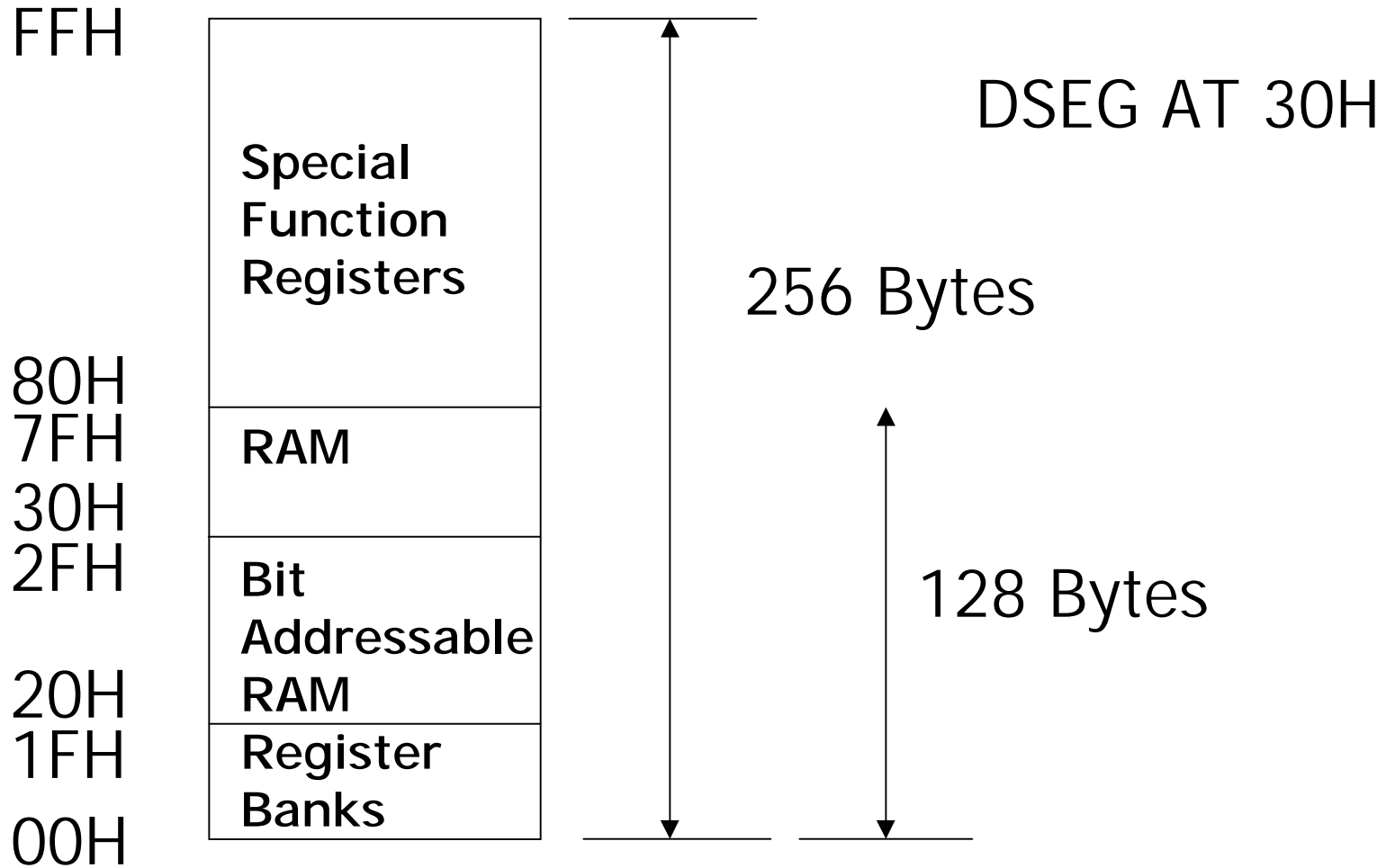
# 8051SBC: Program Memory

---



# 8051SBC: Data Memory

---



# 8051's programming registers

---

<b>A</b>	<b>Accumulator</b>
<b>B</b>	<b>register B</b>
<b>SP</b>	<b>Stack Pointer register</b>
<b>DPTR</b>	<b>Data Pointer register (16bit)</b>
<b>R0-R7</b>	<b>general registers</b>
<b>PSW</b>	<b>Program Status Word (Flag register)</b>

# 8051's programming registers

---

**P1**            **Port P1**

**P2**            **Port P2**

**P3**            **Port P3**

**TH0 TL0** **16-bit timer register**

**TH1 TL1** **16-bit timer register**

**SCON**        **Serial Control register**

**SBUF**         **Serial Buffer register**

# Addressing Modes:

---

**MOV A,#30H ;immediate mode**

**MOV A,30H ; direct mode**

**MOV A,R0 ; register mode**

**MOV R0,#30H ; immediate mode**

**MOV A,@R0 ; indirect mode**

**MOV DPTR,#9000H ;immediate mode**

**MOVB A,@DPTR ; indirect mode**

# Addressing Modes:

---

**MOV C, P3.2 ; bit addressing**

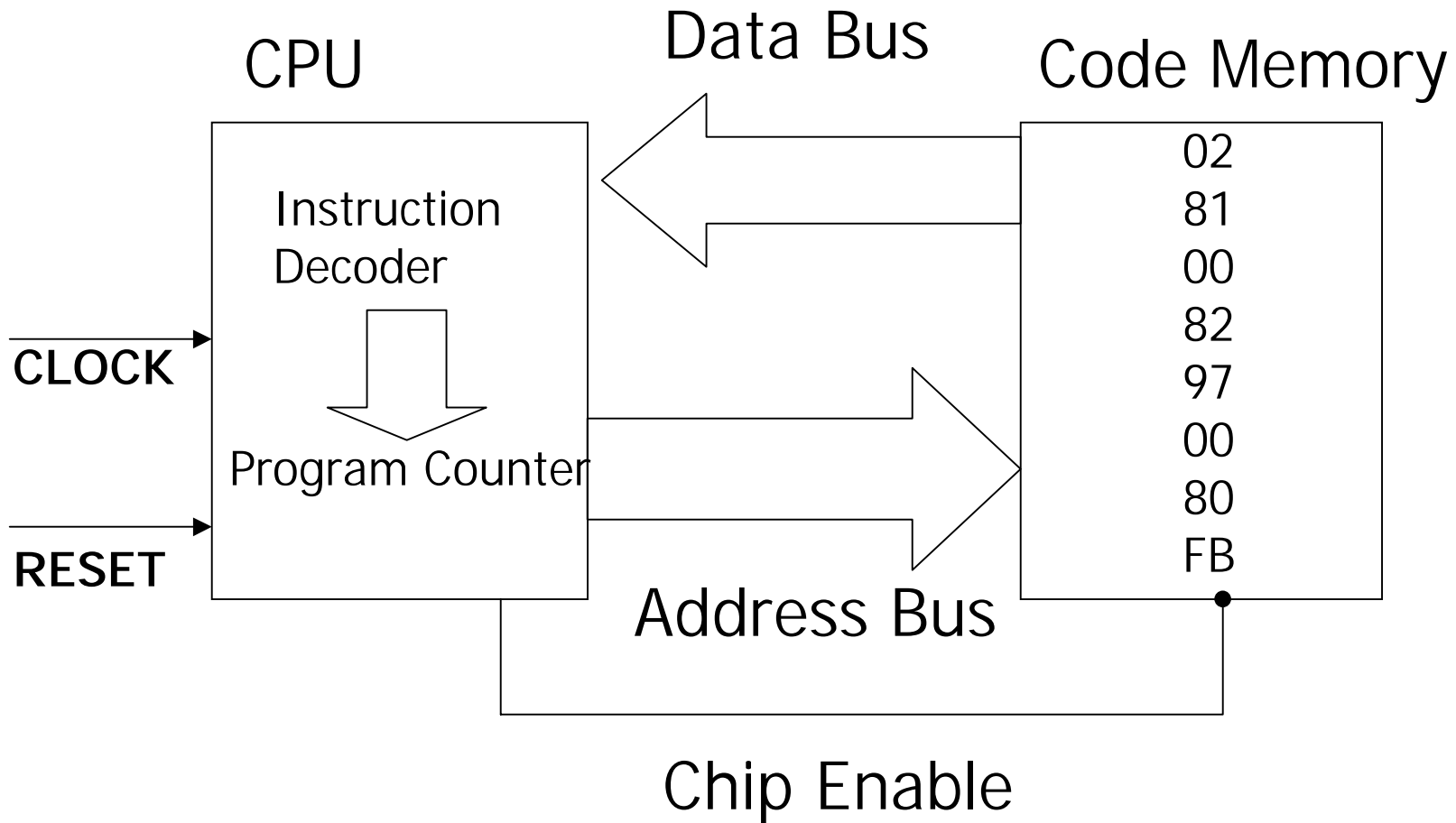
**SETB P1.7 ; bit mode**

**SJMP \$ ; relative addressing**

**MOVC A, @A+DPTR ; index mode**

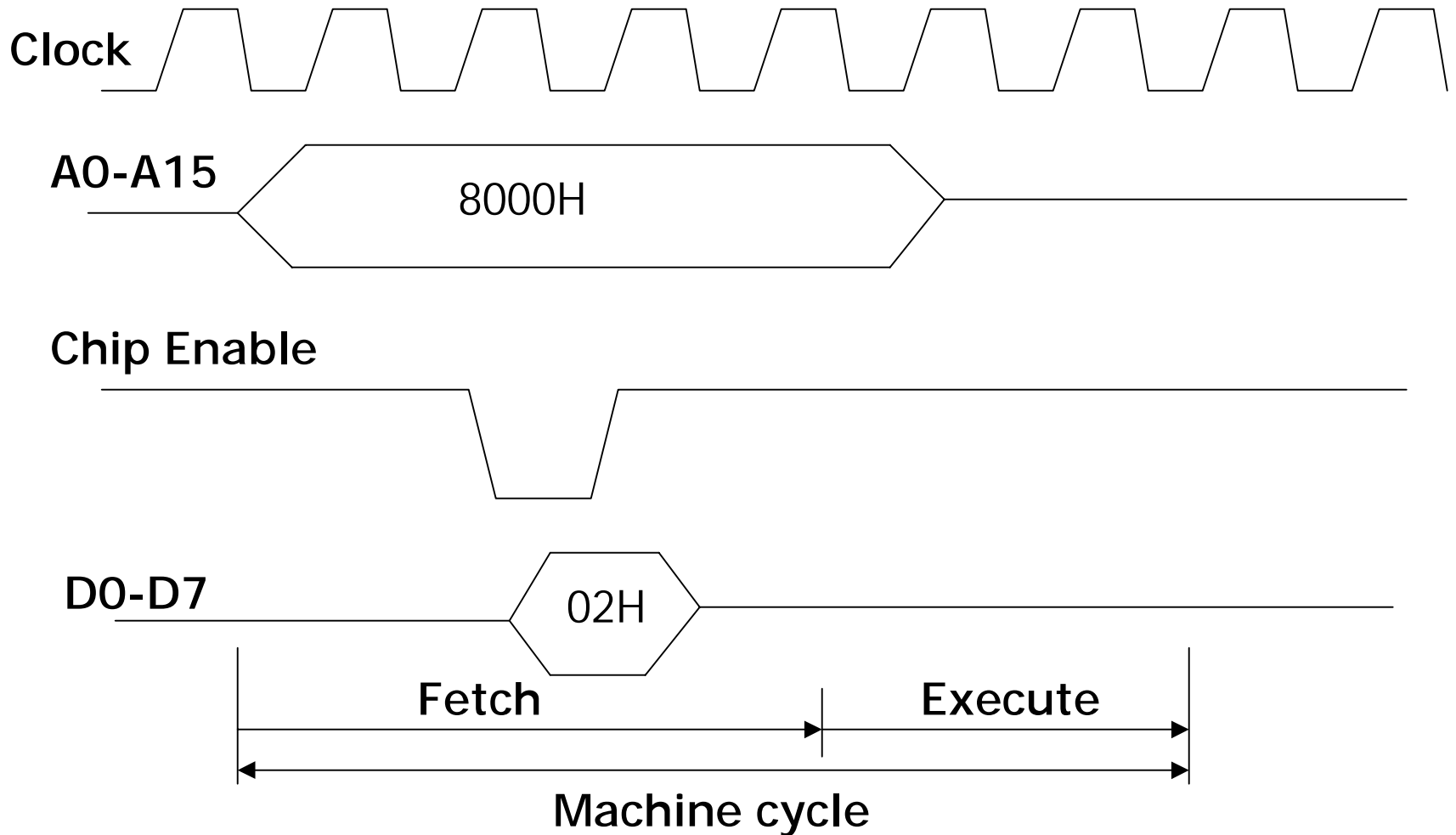
# Operation of Microprocessors:

---



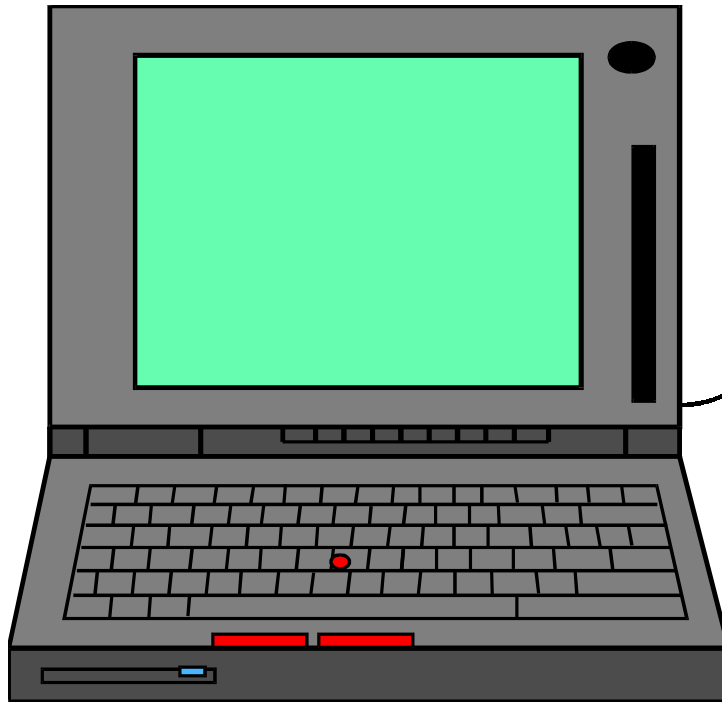
# Instruction Fetch & Execute Cycle:

---



# Using 8051SBC for program testing

---



PC

- Assembler
- Terminal emulator

RS232C Cable



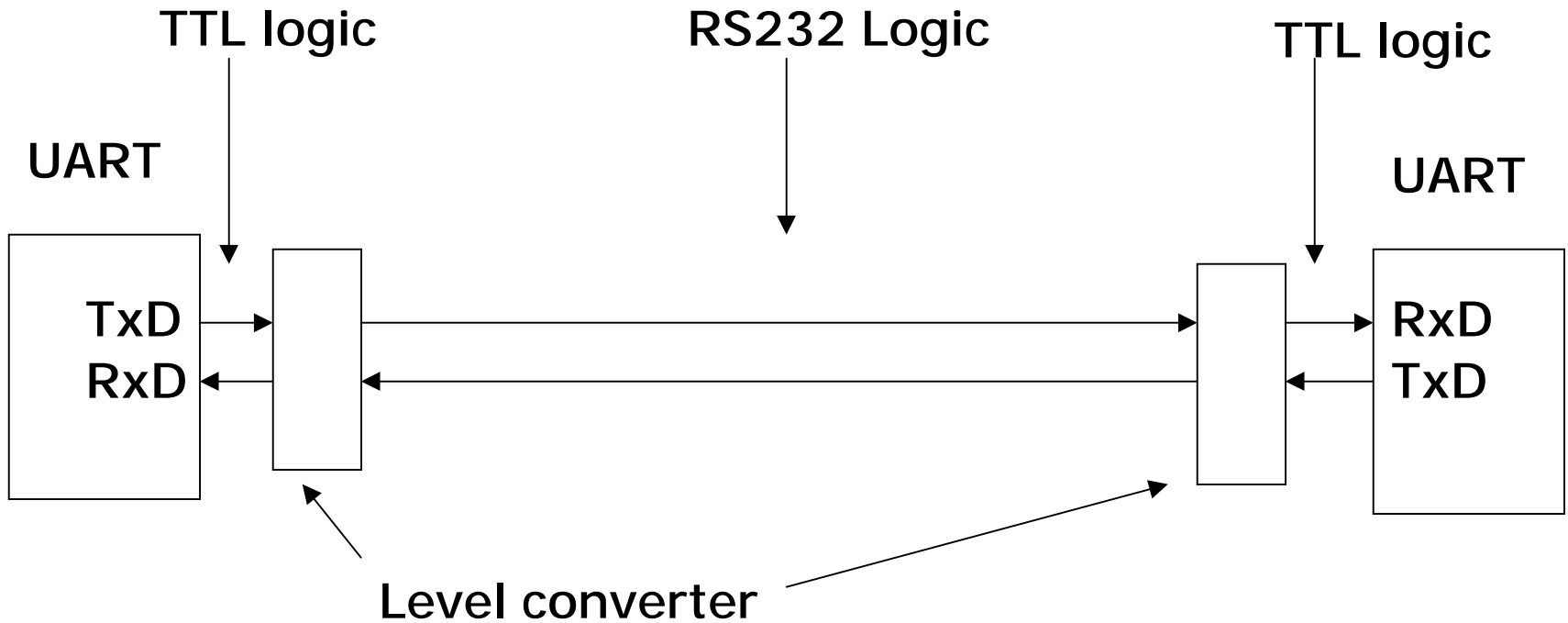
9600 8n1

8051SBC

9600 bit/sec data 8bits no parity  
one stop bit

# Serial Port: RS232C

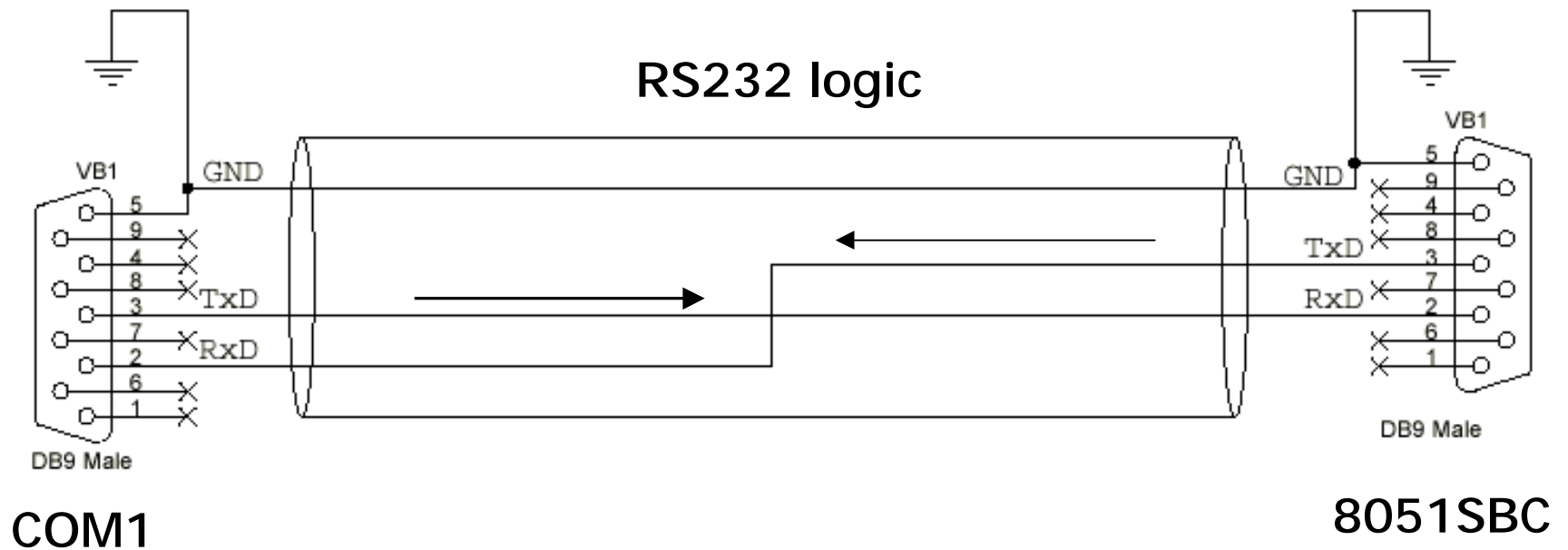
---



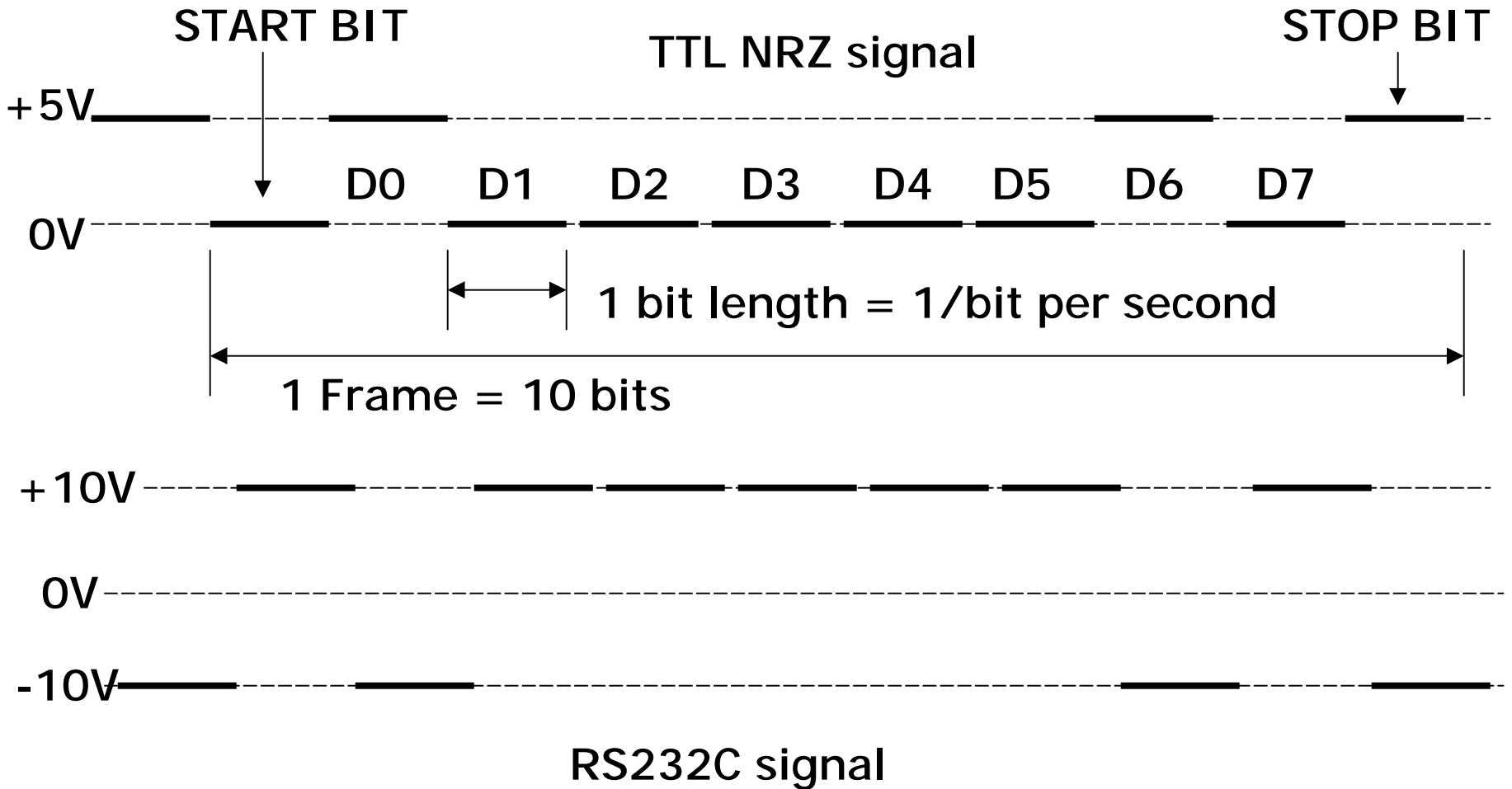
UART (Universal Asynchronous Receiver and Transmitter)

# Serial loading through RS232C

---



# NRZ Digital Baseband



# Sending letter 'A' to terminal

---

**LOOP:    MOV A,#'A'    ; 'A' = 41H**

**CALL putch**

**JMP LOOP**

**putch:    JNB TI, \$            ; TI transmitter**

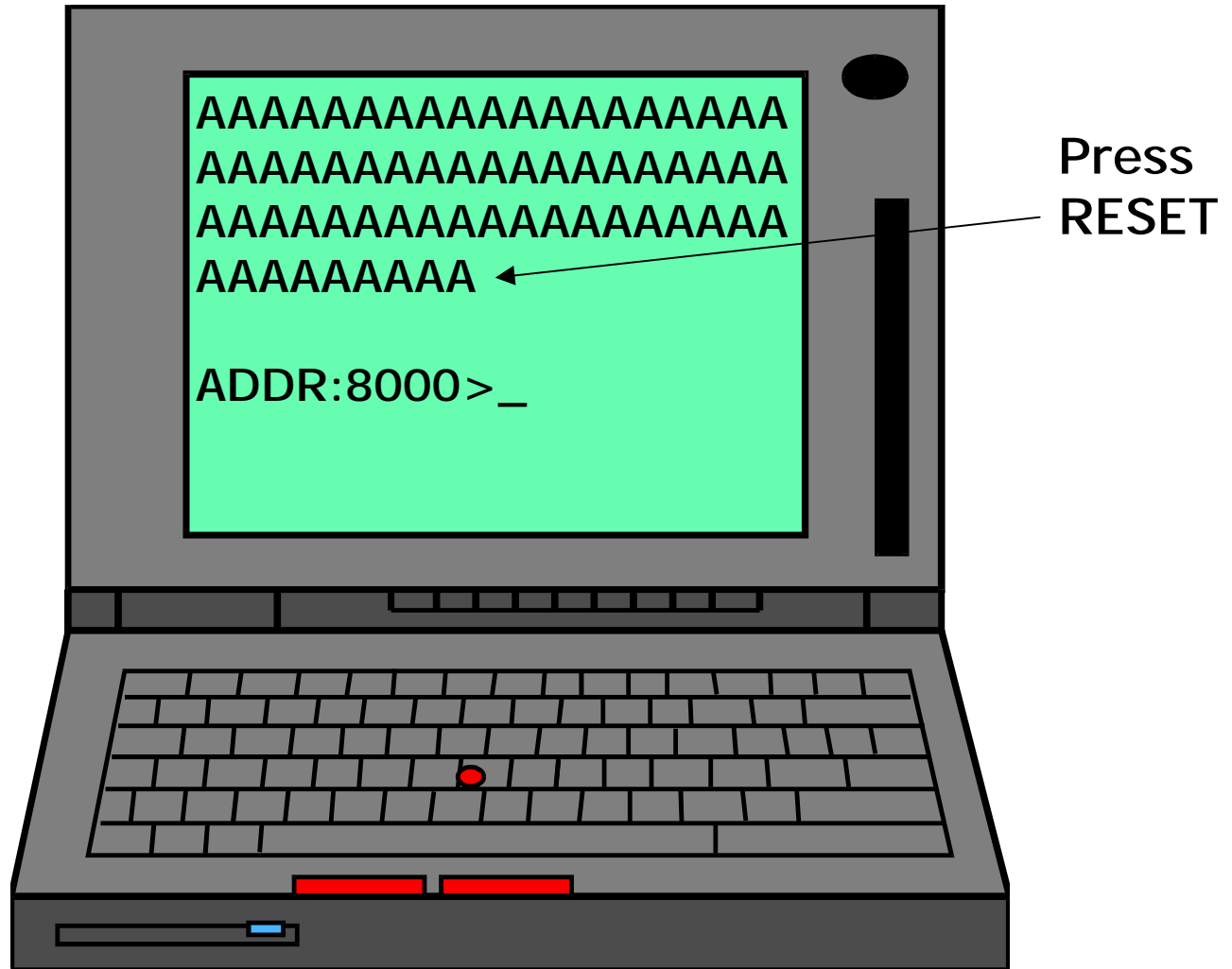
**CLR TI                ; empty flag**

**MOV SBUF,A          ;serial buffer reg**

**RET**

# Sending letter 'A' to terminal

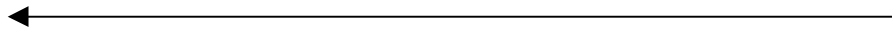
---



# Sending string 'Hello' to terminal

---

Hello



```
Loop: A = [MEM]
      if (A == eos) exit
      send A
      MEM++
      jump loop
```

H	[MEM]
e	
l	
l	
o	
eos	

# Subroutine putstring

---

putstring: CLR A

MOV C A, @A + DPTR

CJNE A, #eos, string1

RET

string1: CALL putch

INC DPTR ; next address

JMP putstring

eos equ 0 ; terminator

# Subroutine putstring

---

**Main:     MOV DPTR,#hello**

**CALL putstring**

**JMP Main**

**hello:     DB 'hello',13,10,eos**

**13     = 0dH is ASCII printing control code. It will move cursor back to the left most position! (Carriage Return)**

**10 = 0ah is Line Feed, LF. The cursor will move to new line!**

# Simple delay subroutine

---

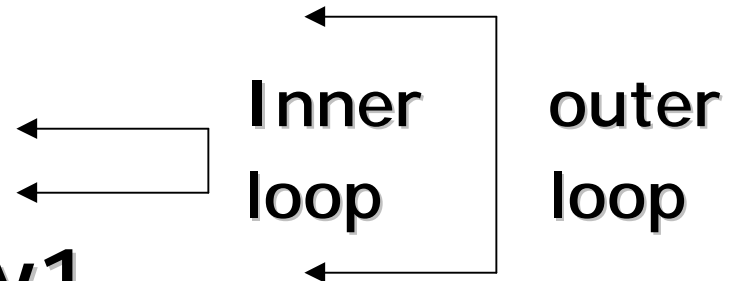
delay: MOV R6,#100

delay1: MOV R7,#0

DJNZ R7,\$

DJNZ R6,delay1

RET



; inner loop will be repeated 100 times!

; simple coding, but waste CPU time!

# Measuring delay period

---

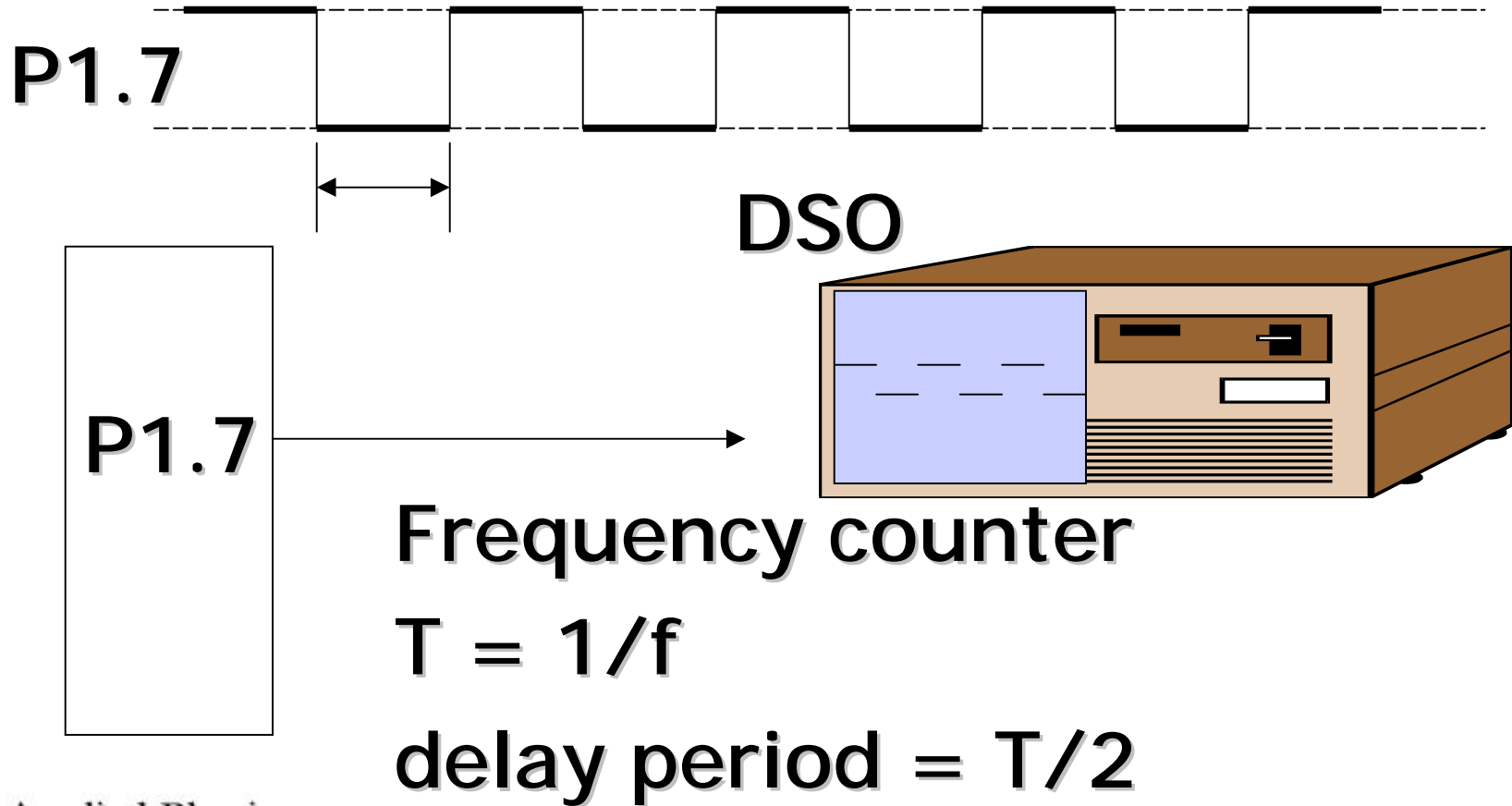
```
delay:  MOV R6,#100  
        DJNZ R6,$  
        RET
```

; we can determine delay period with  
simple code and a given output port!

```
main:   CPL P1.7  
        CALL DELAY  
        JMP main
```

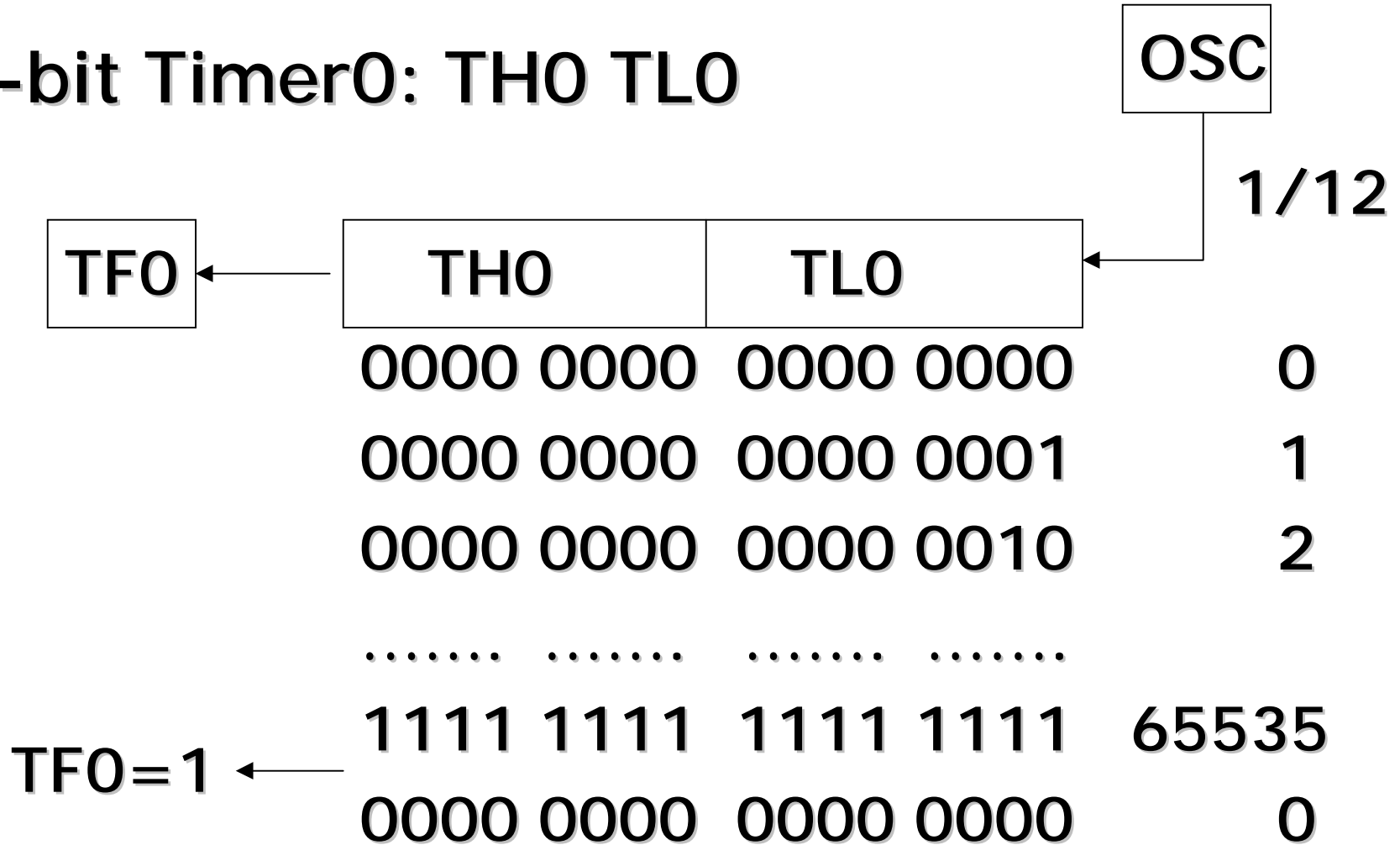
# Measuring delay period

---



# Using Hardware Timer

## 16-bit Timer0: TH0 TLO



# Using Hardware Timer

---

8051SBC has 11.0592MHz oscillator.

The clock frequency for timer is

$$11.0592\text{MHz}/12 = 921,600 \text{ Hz}$$

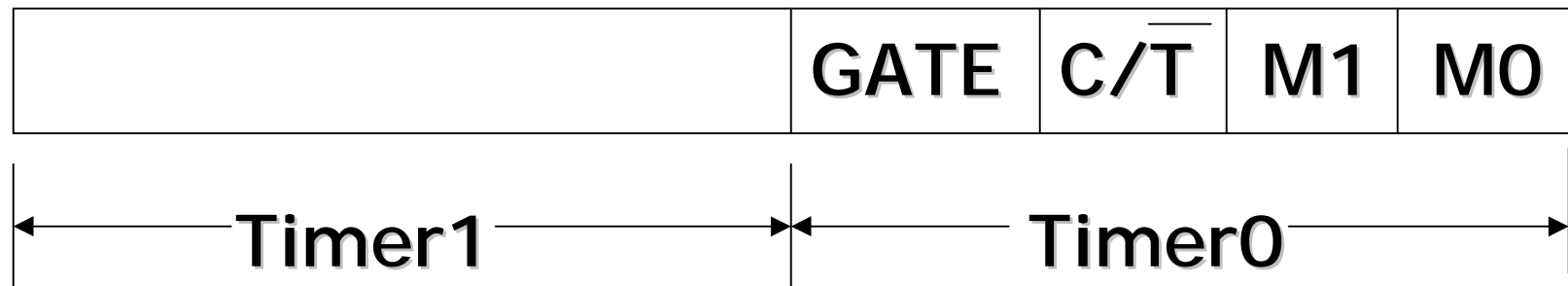
If we have timer0 with initial value = 0000h, TF0 will set when time equal  $[1/921,600\text{Hz}] \times 65,536 = 0.07111 \text{ s}$  has elapsed.

# How to set MODE for Timer0?

---

**TMOD (89H) : Mode setting for timer**

**TCON (88H): run/stop timer counting**



**MODE1 = 16-bit timer (M1=0,M0=1)**

**ORL TMOD,#1 ; logical OR TMOD with 1**

# How to run Timer0?

---

**TCON (88H): run/stop timer counting**



TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
-----	-----	-----	-----	-----	-----	-----	-----

**; run timer0 by setting TR0 to '1'**

**ORL TCON,#00010000B**

**; or with set bit instruction**

**SETB TCON.4**

# Sample code with timer0

---

**Main:**     **ORL TMOD,#1**  
              **SETB TCON.4**

**wait:**     **JNB TF0,\$**  
              **CLR TF0**  
              **CPL P1.7**  
              **JMP wait**