Interesting Question

- There is no P0MDIN!?

OSCXCN	0x8C	F	External Oscillator Control	page 176
P0	0x80	All Pages	Port 0 Latch	page 215
P0MDOUT	0xA4	F	Port 0 Output Mode Configuration	page 216
P1	0x90	All Pages	Port 1 Latch	page 216
P1MDIN	0xAD	F	Port 1 Input Mode Configuration	page 217
P1MDOUT	0xA5	F	Port 1 Output Mode Configuration	page 217
P2	0xA0	All Pages	Port 2 Latch	page 218
P2MDIN	0xAE	F	Port 2 Input Mode Configuration	page 218
P2MDOUT	0xA6	F	Port 2 Output Mode Configuration	page 219
P3	0xB0	All Pages	Port 3 Latch	page 219
P3MDIN	0xAF	F	Port 3 Input Mode Configuration	page 220
P3MDOUT	0xA7	F	Port 3 Output Mode Configuration	page 220

- Yes, there is only one input mode for P0

P0.0	62	55	D I/O	Port 0.0. See Port Input/Output section for complete description.
P0.1	61	54	D I/O	Port 0.1. See Port Input/Output section for complete description.

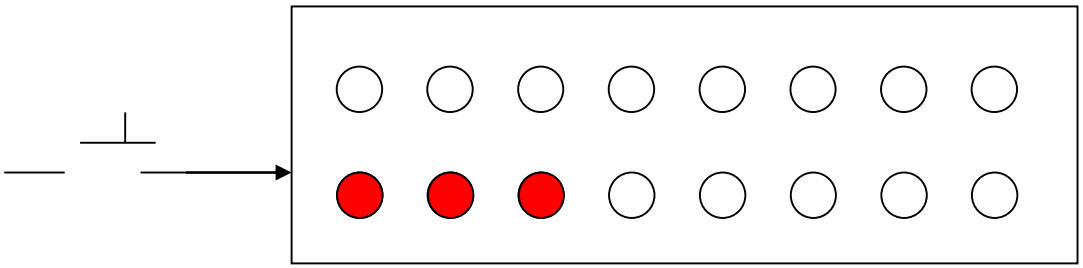
P1.0/AIN2.0/A8	36	29	A In D I/O	ADC1 Input Channel 0 (See ADC1 Specification for complete description). Bit 8 External Memory Address bus (Non-multiplexed mode) Port 1.0 See Port Input/Output section for complete description.
P1.1/AIN2.1/A9	35	28	A In D I/O	Port 1.1. See Port Input/Output section for complete description.

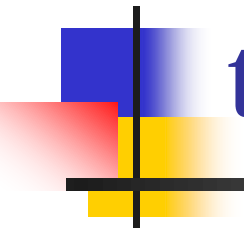
Lab 03

Timer and Interrupt Mechanism

Your work today

- program 8051 to show some LED pattern like last week
- but control using timer and interrupt mechanism

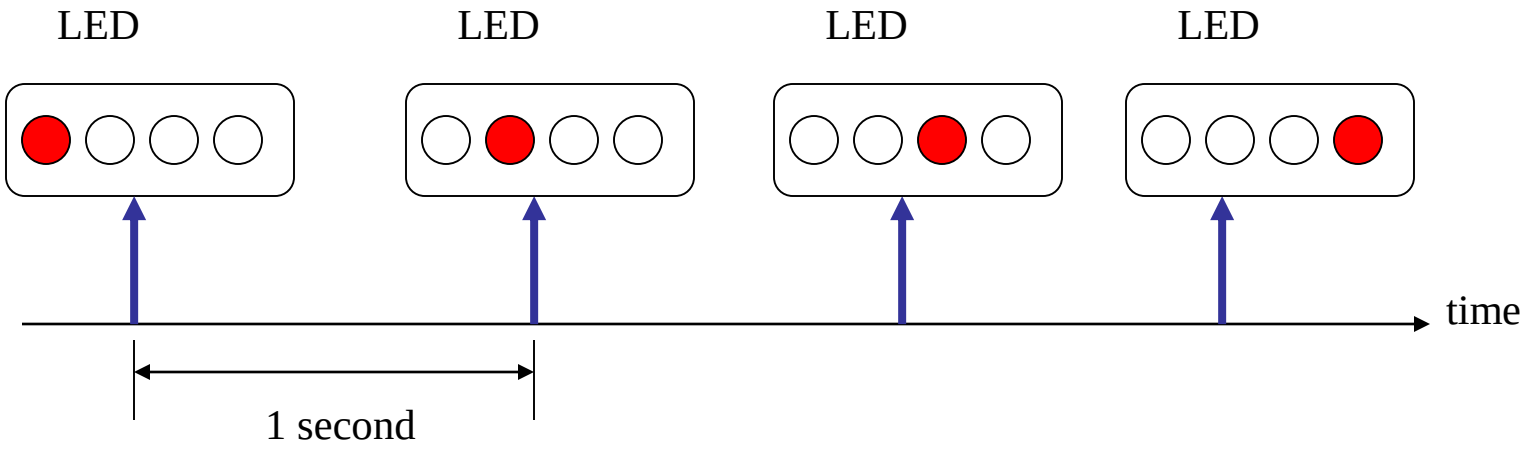




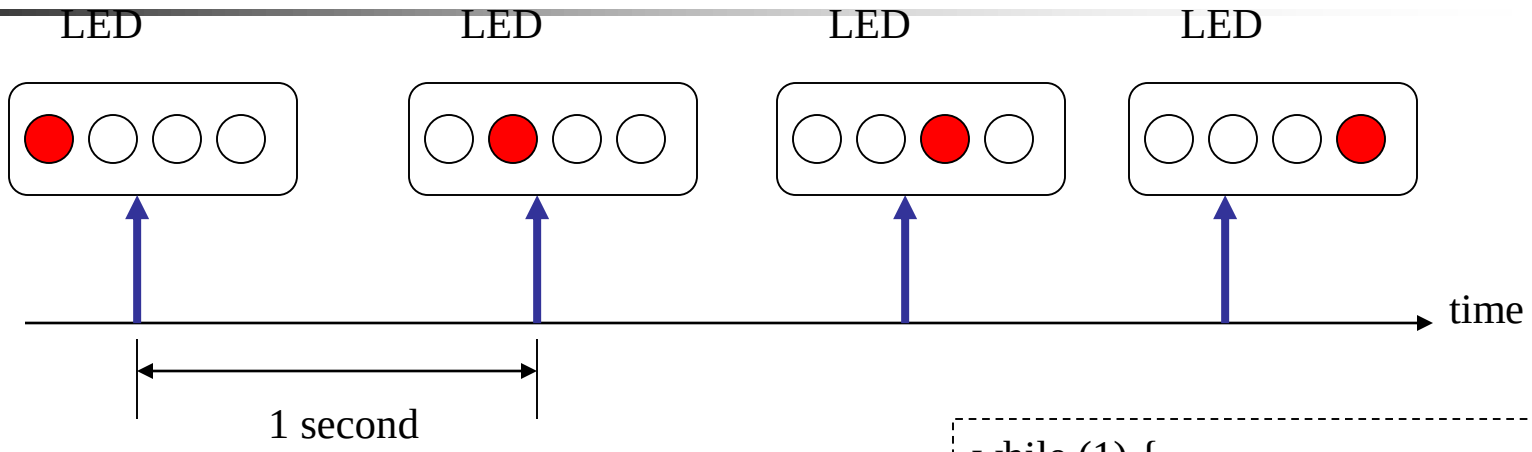
Overview: program control using timer and interrupt

Why use timer + interrupt

- a program to do precisely timed control
- Example: make LED switches precisely every 1 second



Will you like to do it in this way?

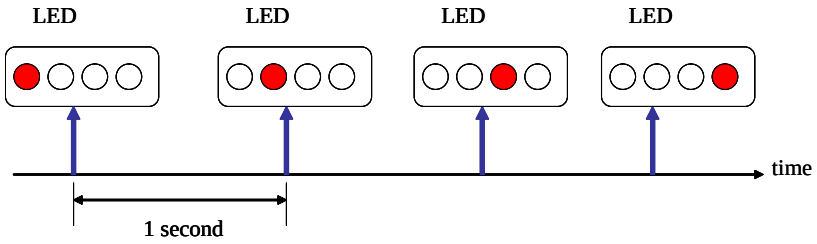


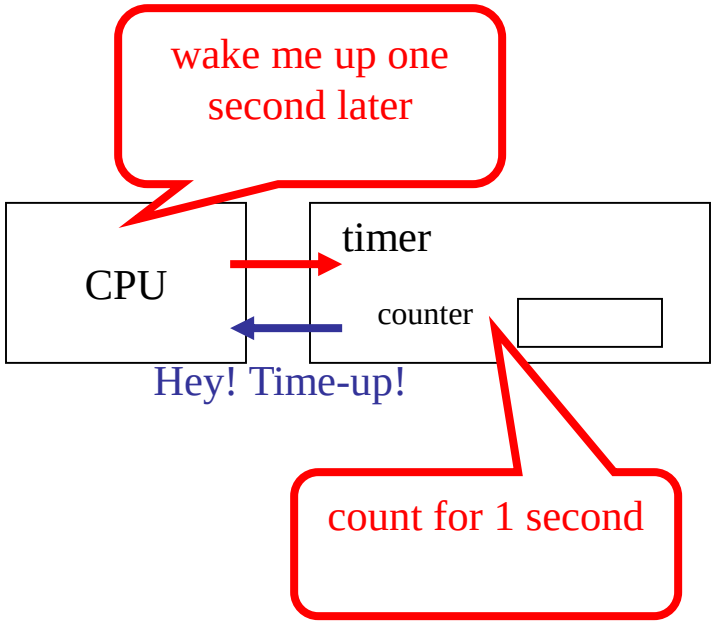
- How to set N?
  - you need precise cycle count for each assembly instruction

```
while (1) {
        A = RR(A);      //rotate right
        P0 = A;
        delay (N);
}

delay (int N)
{
        int i;
        for (i=0;i<N;i++);
}
```

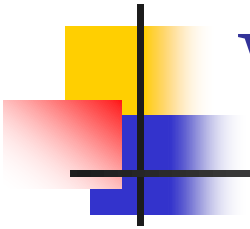
A better way to do timed control

- use timer + interrupt
- Example:





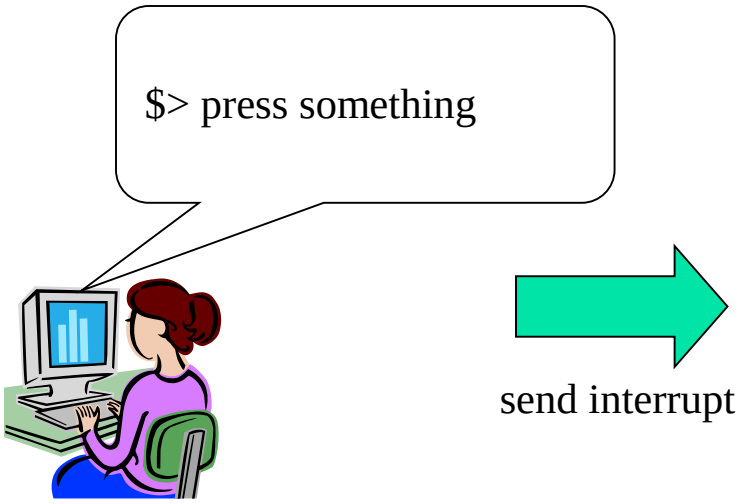
Basic concepts of interrupt mechanism



What is an interrupt?

- to “interrupt” the normal execution of a CPU
  - turn to do something exceptional and then back to normal execution
  - usually to serve external I/O devices

Interrupt normal execution and then return



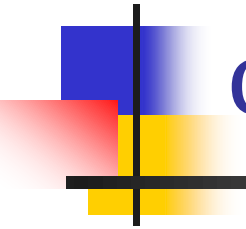
process ID. 1234

```
main()
{
   while (…) {
       …
       //normal execution
       …
   }
}

keyboard_intr_handler ()
{
   printf (“A”);
}
```

interrupt service routine (ISR)

→ PC

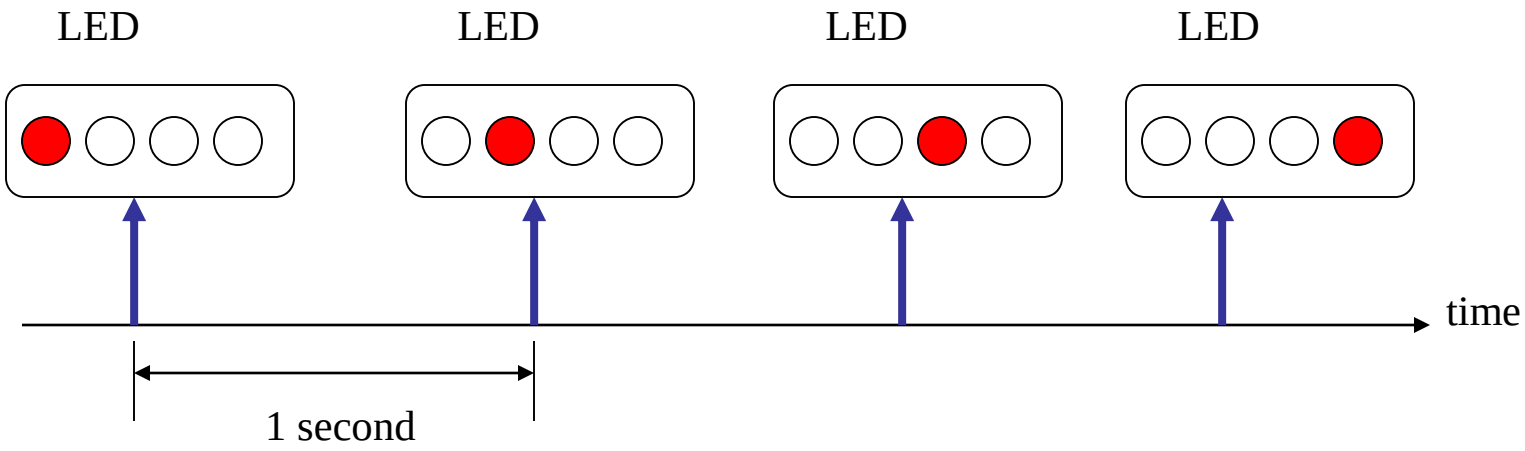
Timer + Interrupt for timed control



the conceptual idea

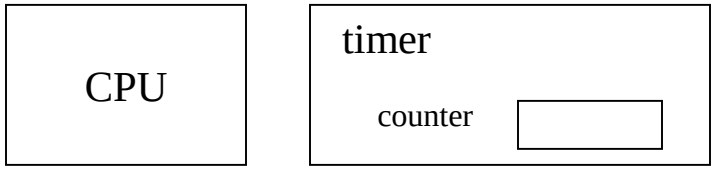
A better way to do timed control

- use timer + interrupt
- Example:

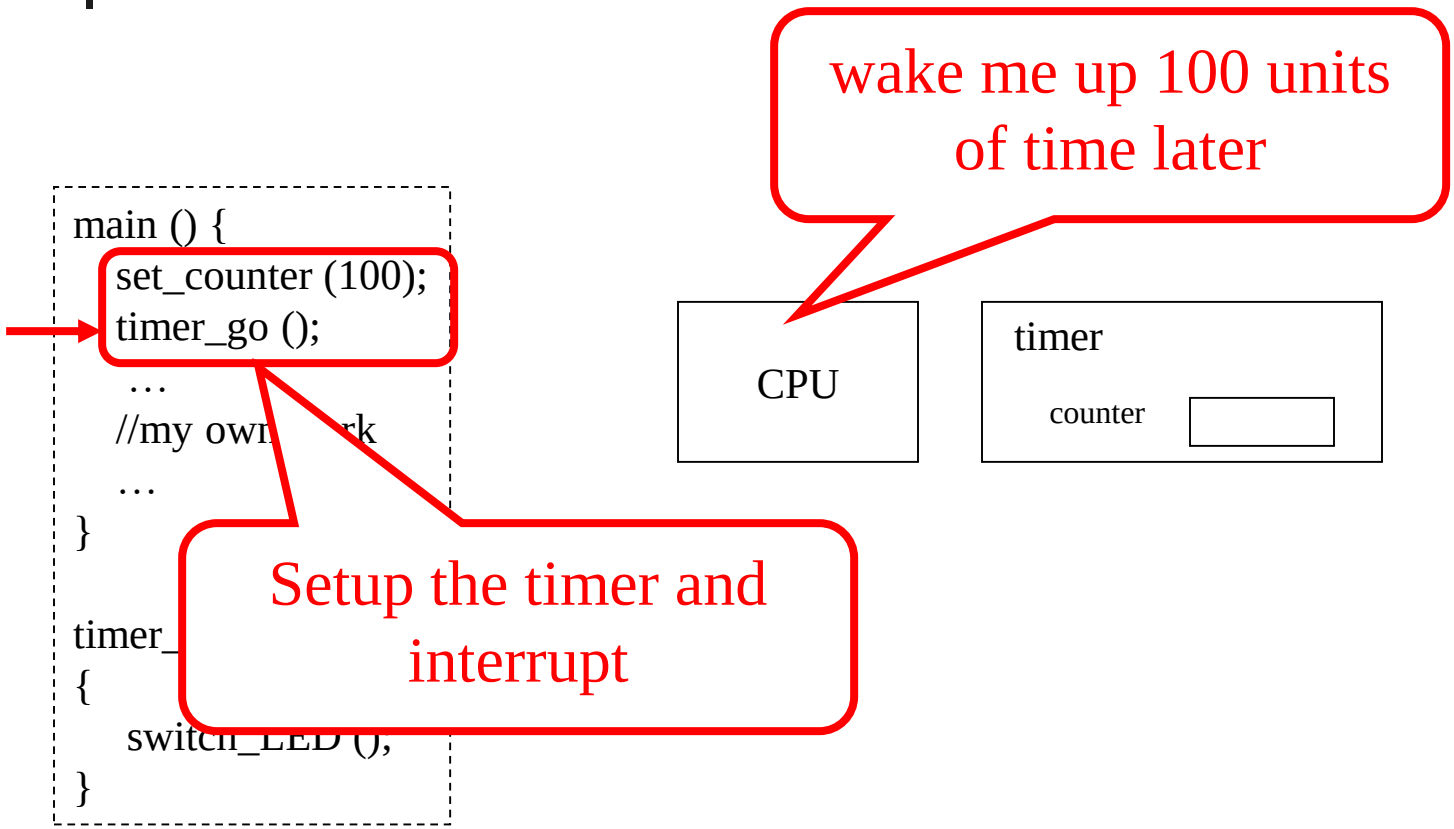


A better way: uses timer+interrupt

```
main () {
    set_counter (100);
    timer_go ();
     …
    //my own work
    …
}

timer_intr_service ()
{
    switch_LED ();
}
```



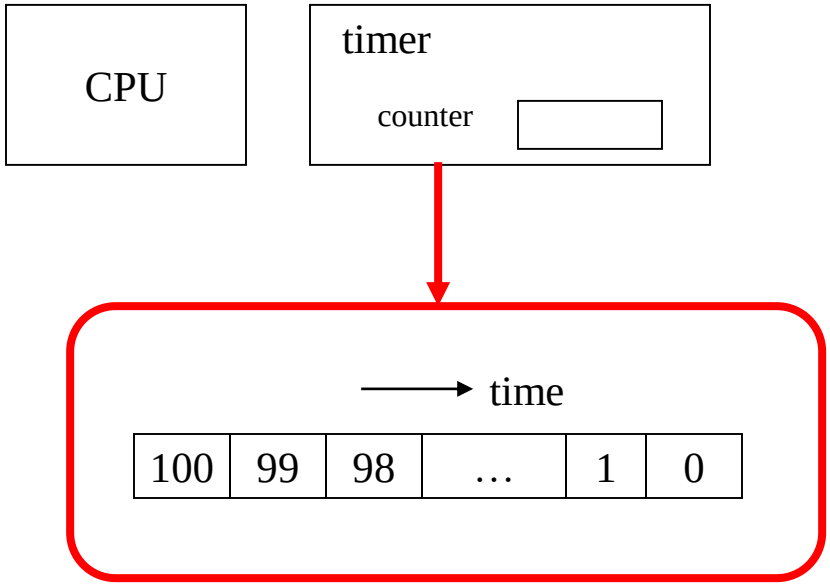
A better way: uses timer+interrupt



A better way: uses timer+interrupt

- CPU does its own work and the timer go counting

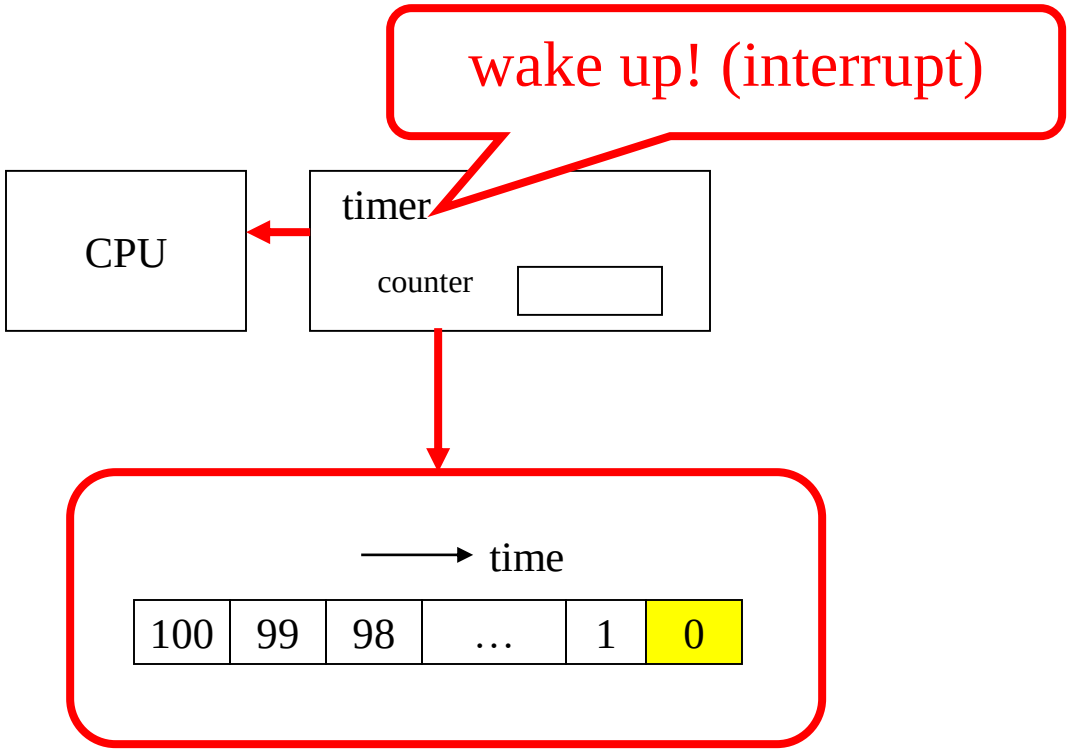
```
main () {
    set_counter (100);
    timer_go ();
     …
    //my own work
    …
}

timer_intr_service ()
{
     switch_LED ();
}
```



A better way: uses timer+interrupt

- the timer sends an interrupt to CPU when time-up

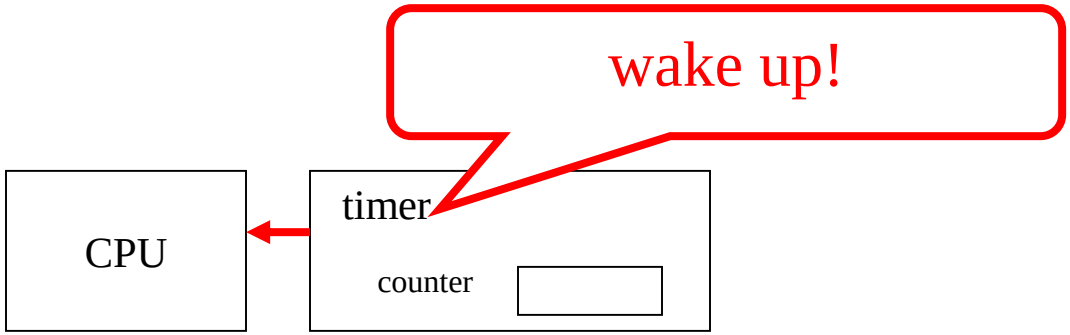
```
main () {
    set_counter (100);
    timer_go ();
     …
    //my own work
    …
}

timer_intr_service ()
{
     switch_LED ();
}
```

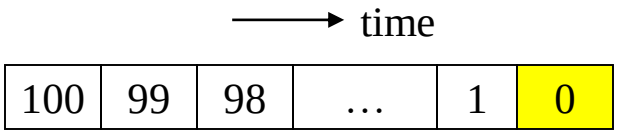


A better way: uses timer+interrupt

- CPU turn to the interrupt service routine

```
main () {
    set_counter (100);
    timer_go ();
     …
    //my own work
    …
}

timer_intr_service ()
{
    switch_LED ();
}
```

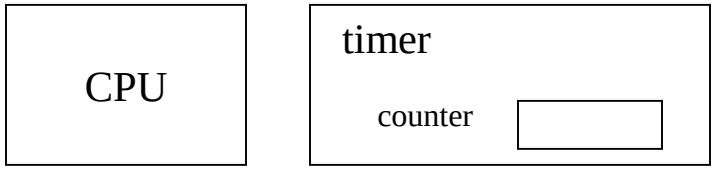


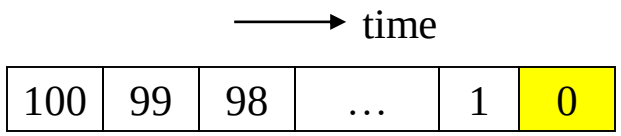


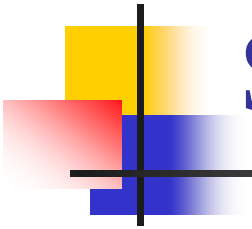
A better way: uses timer+interrupt

- then back to its normal execution

```
main () {
    set_counter (100);
    timer_go ();
     …
    //my own work
    …
}

timer_intr_service ()
{
      switch_LED ();
}
```





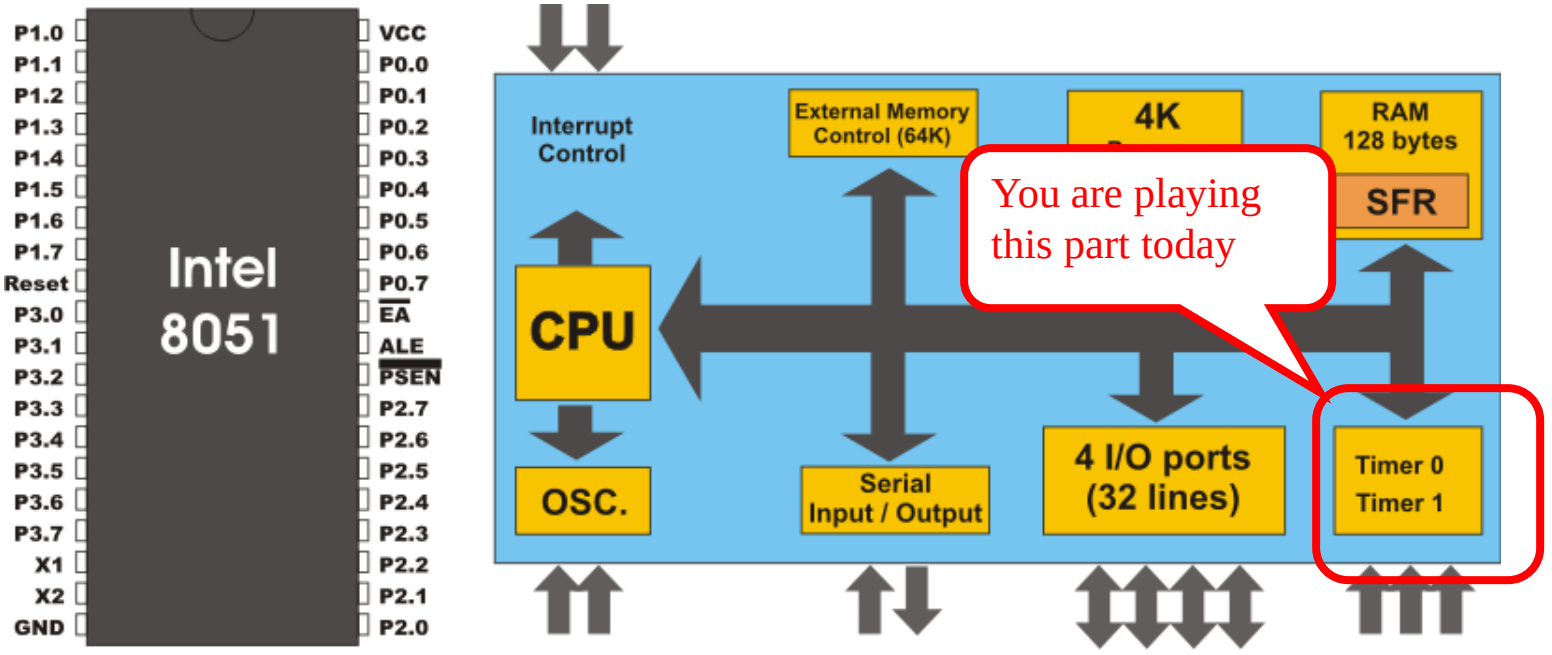


Summary

- Interrupt mechanism:
  - a hardware signal to inform CPU some event has happened
  - makes CPU change its execution path
    - turn to "Interrupt Service Routine" (ISR)
    - then return to its normal execution path and status
- Timer:
  - an external counter to count up for specified time
  - usually inform CPU with interrupt

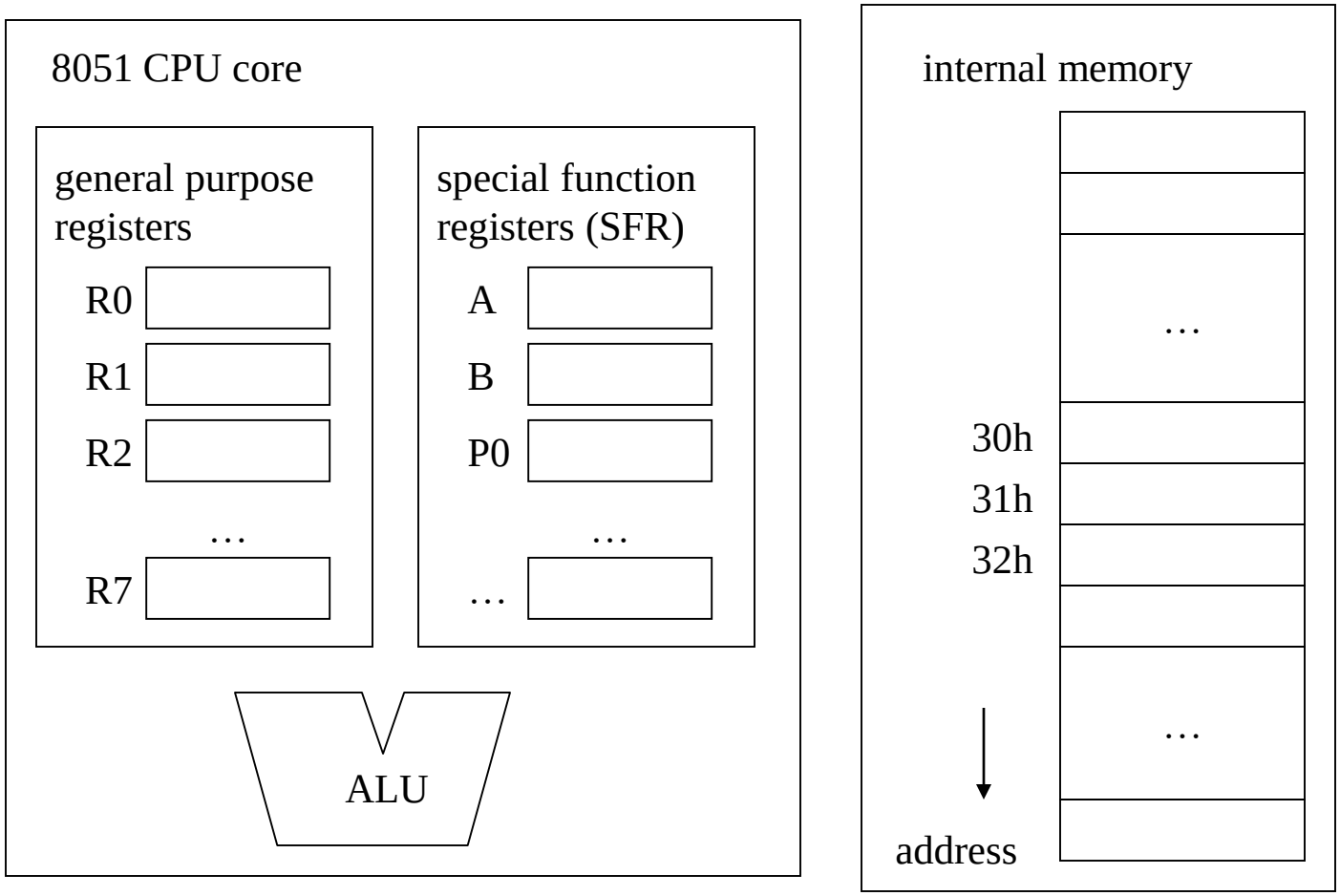
The 8051 part

The 8051 architecture



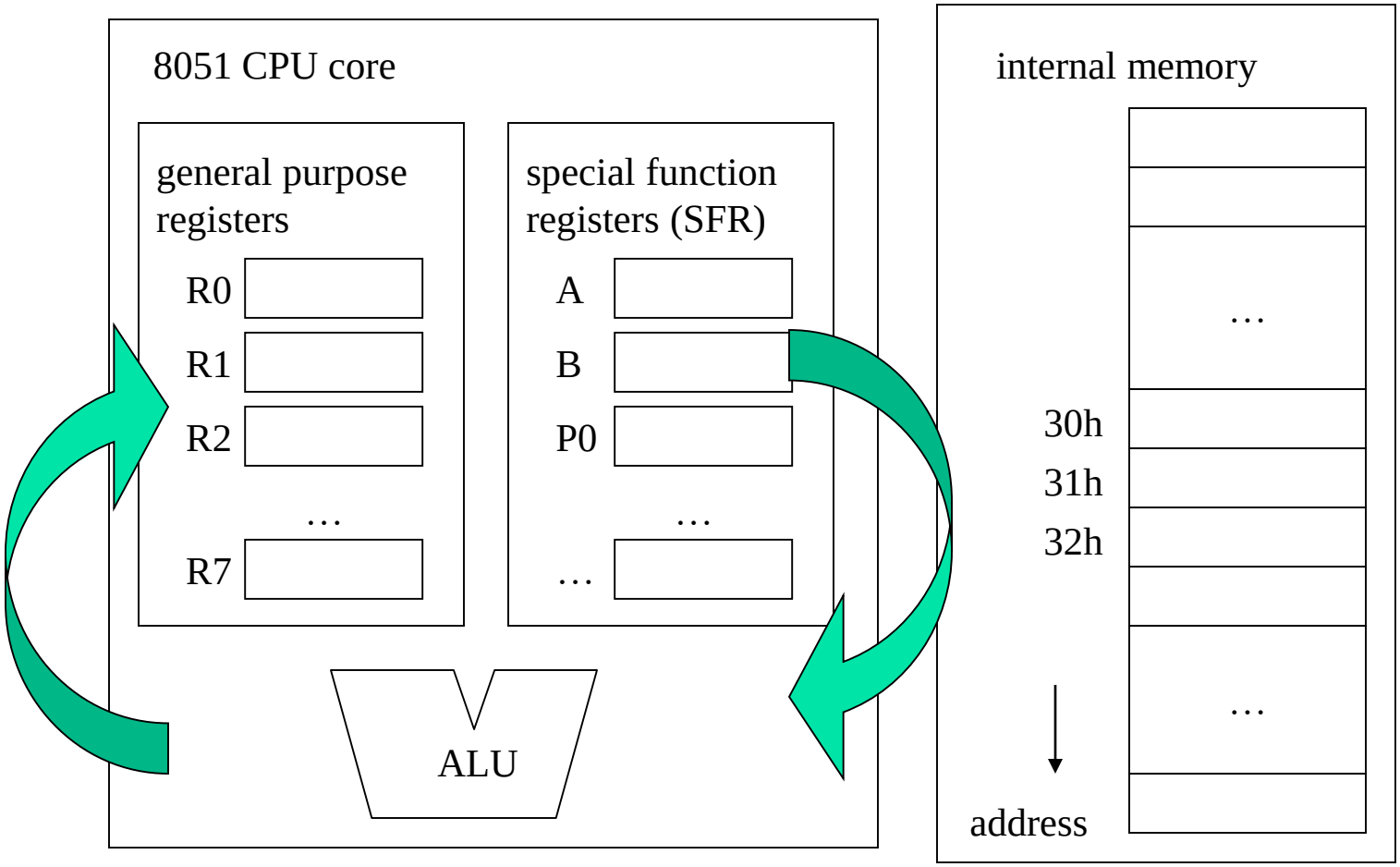
Imagination on 8051 architecture

- Imagine how data flow in the architecture!



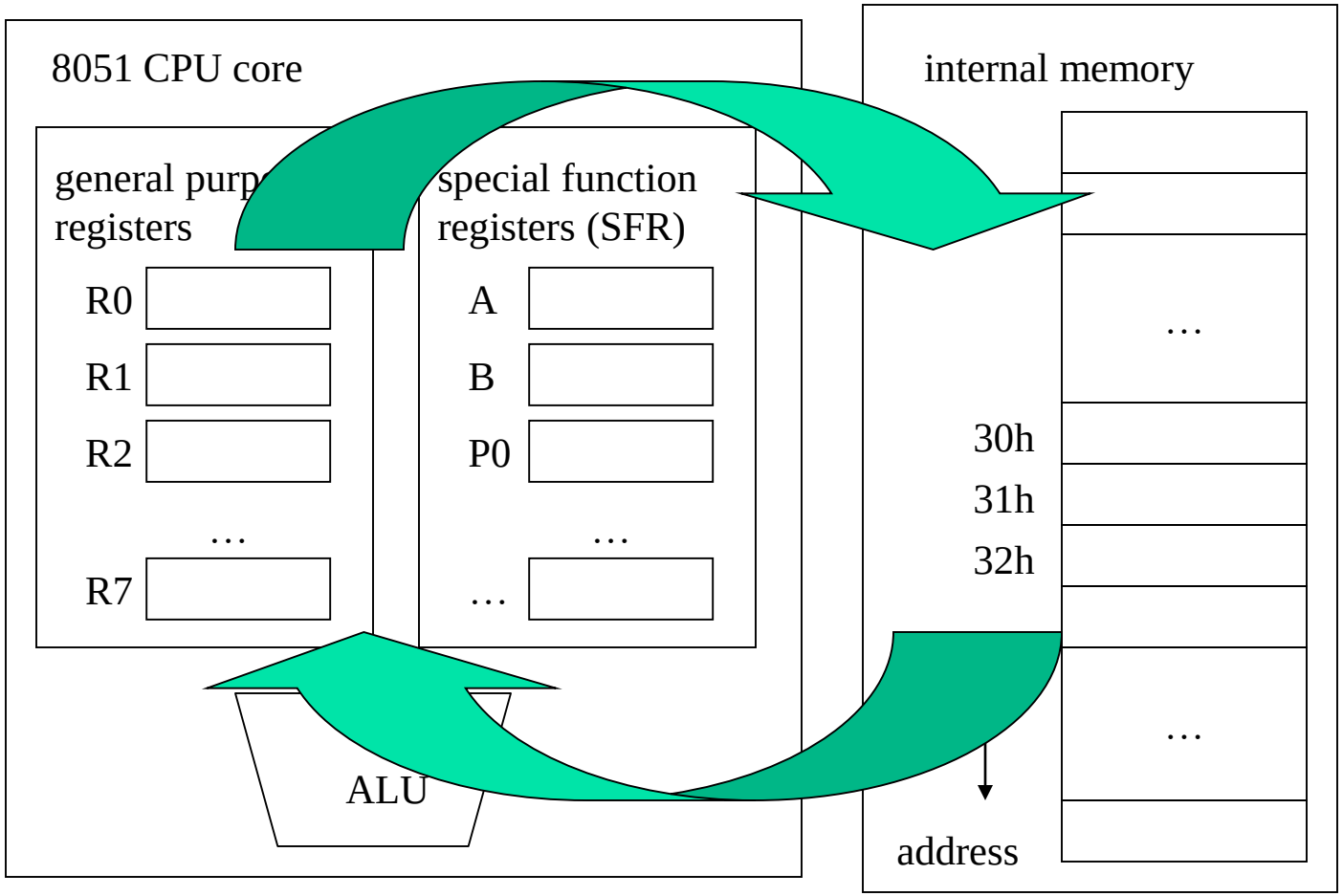
Imagination on 8051 architecture

- flow of an arithmetic instruction



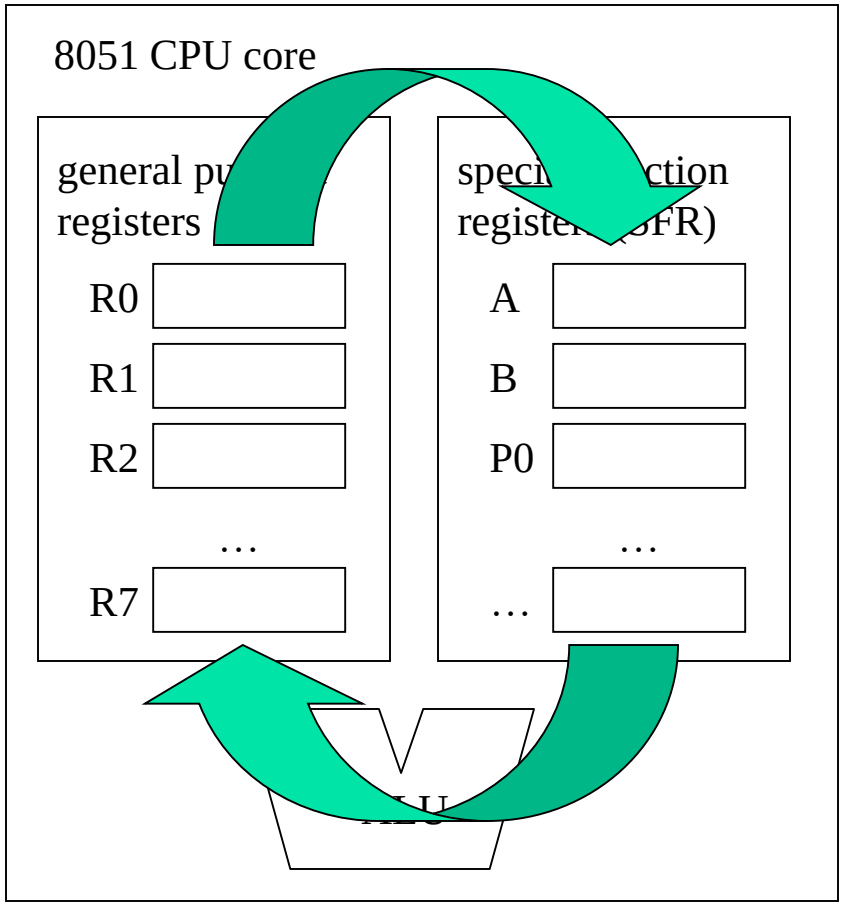
Imagination on 8051 architecture

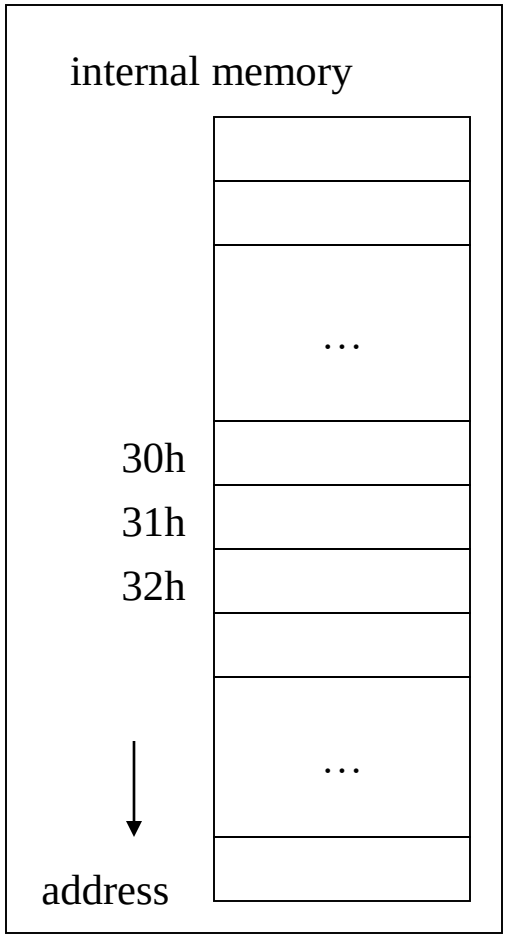
- data movement between memory and registers
- the MOV instruction



Imagination on 8051 architecture

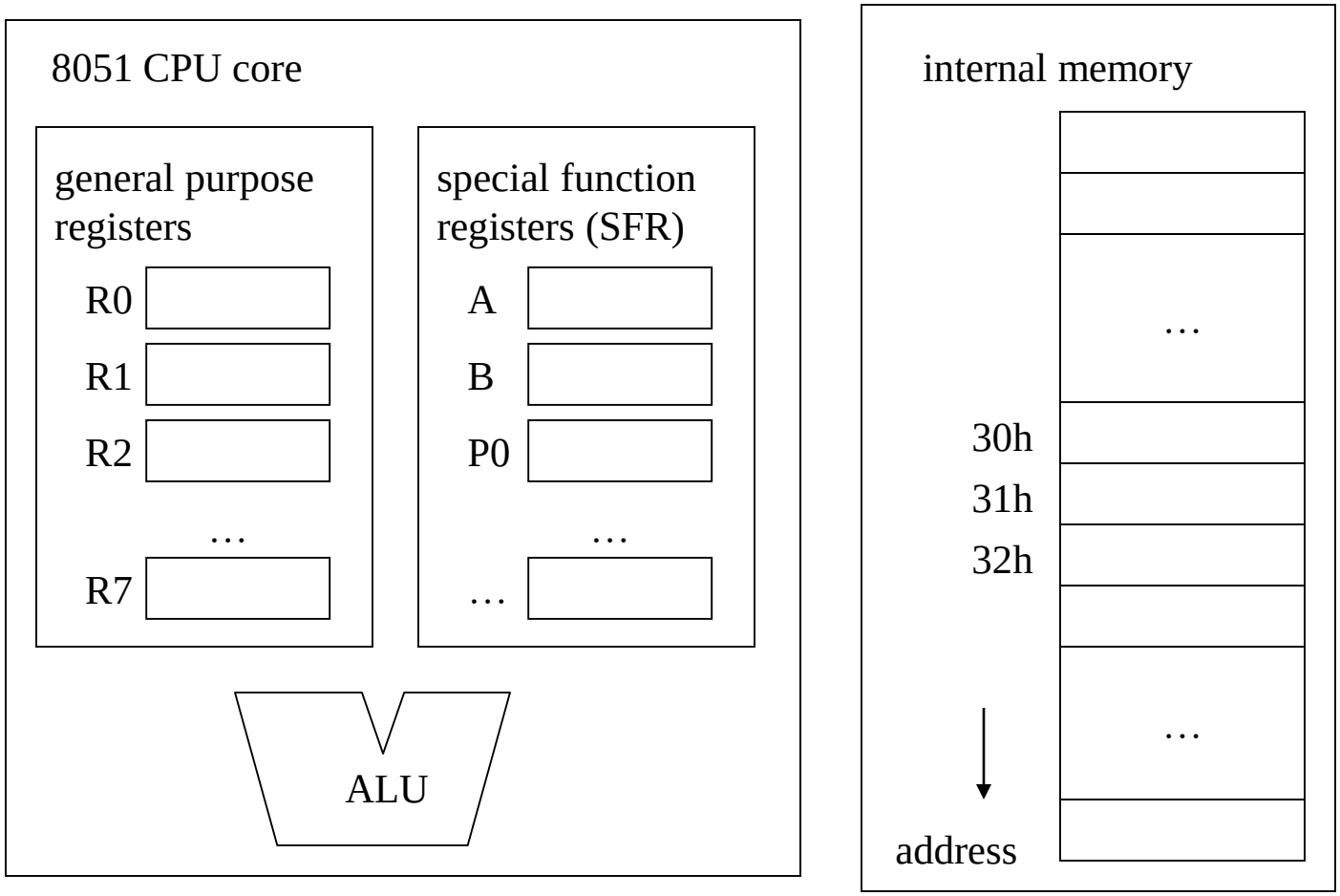
- the MOV also for registers

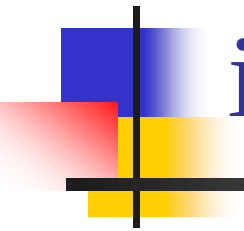




Imagination on 8051 architecture

- Imagine how data flow in the architecture!





How to program the timer and interrupt mechanism on 8051

Your program will look like this

```
main ()
{
   TMOD = ???
   TCON = ???
   TH0 = ???
   TL0 =  ???
   IE = ???

   while (1);  //infinite loop and do nothing
}

Timer_ISR ()
{
     //change LED pattern
}
```

Your program will look like this

```
main ()
{
   TMOD = ???
   TCON = ???
   TH0 = ???
   TL0 =  ???
   IE = ???

   while (1);  //infinite loop and do nothing
}

Timer_ISR ()
{
     //change LED pattern
}
```

Fill in SFR registers to setup the timer and the interrupt

Your program will look like this

```
main ()
{
   TMOD = ???
   TCON = ???
   TH0 = ???
   TL0 =  ???
   IE = ???

   while (1);  //infinite loop and do nothing
}

Timer_ISR ()
{
     //change LED pattern
}
```

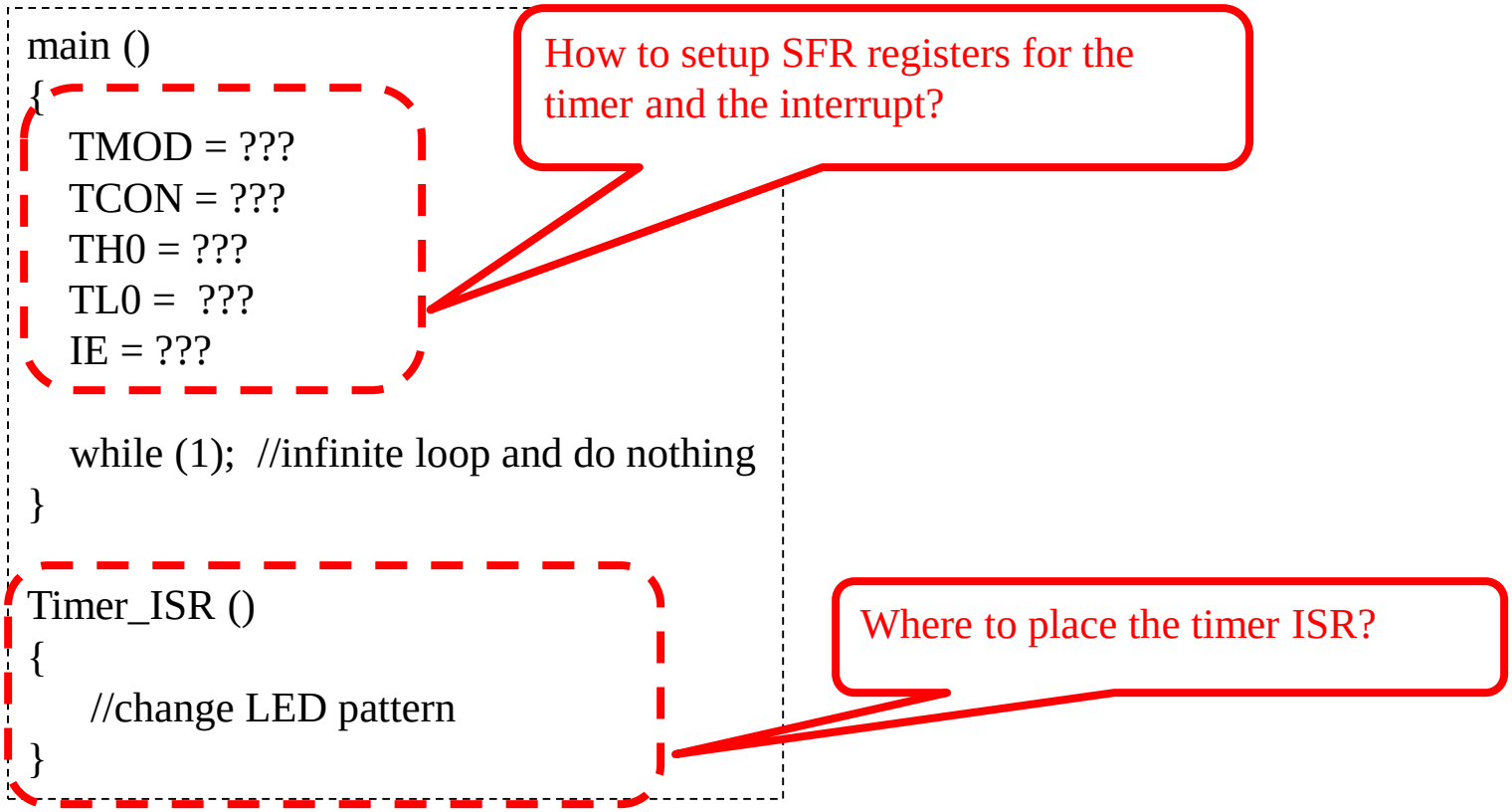
You don't need to branch to control the LED pattern

Your program will look like this

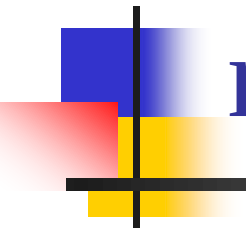
```
main ()
{
   TMOD = ???
   TCON = ???
   TH0 = ???
   TL0 =  ???
   IE = ???

   while (1);  //infinite loop
}

Timer_ISR ()
{
    //change LED pattern
}
```

The timer interrupt will be executed regularly once setup finished

Things you need to know



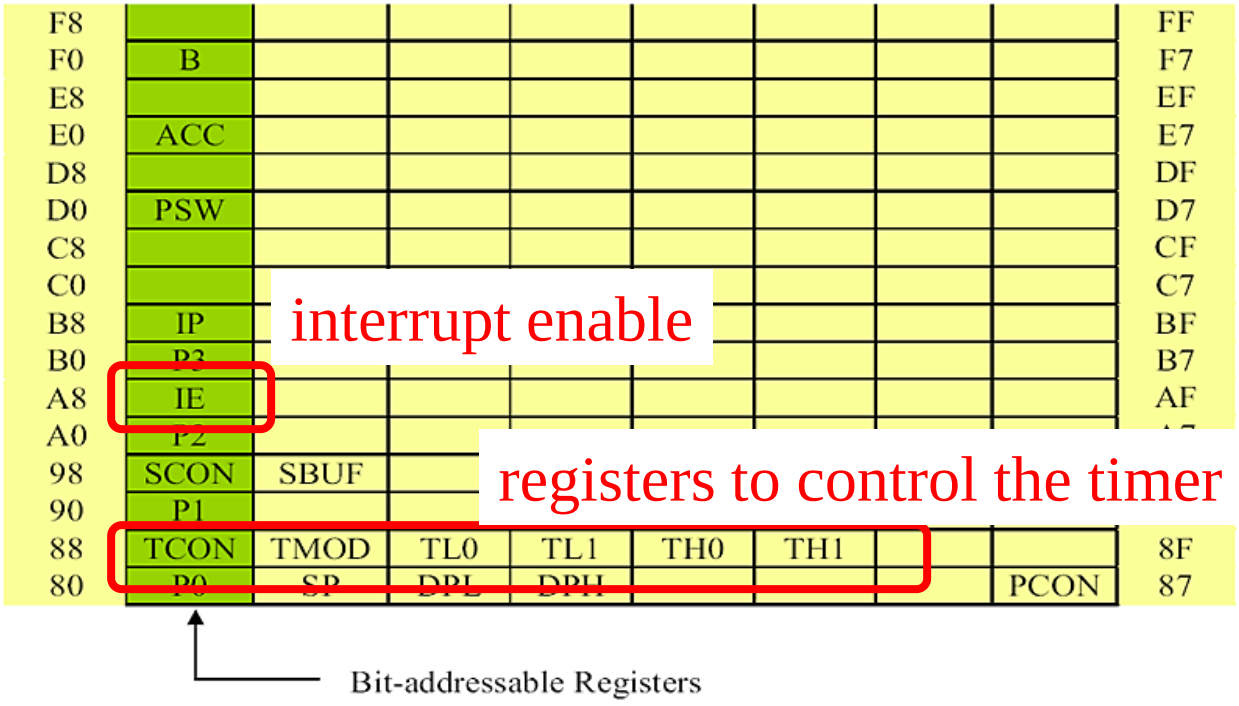
How to program 8051's interrupt mechanism



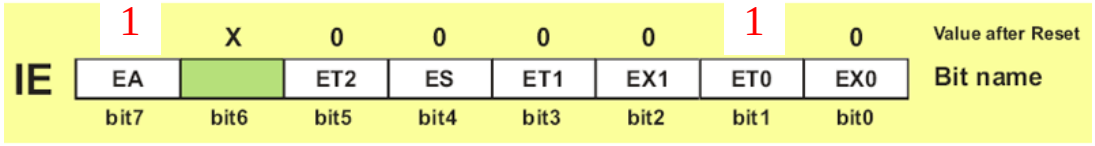
Things you need to know

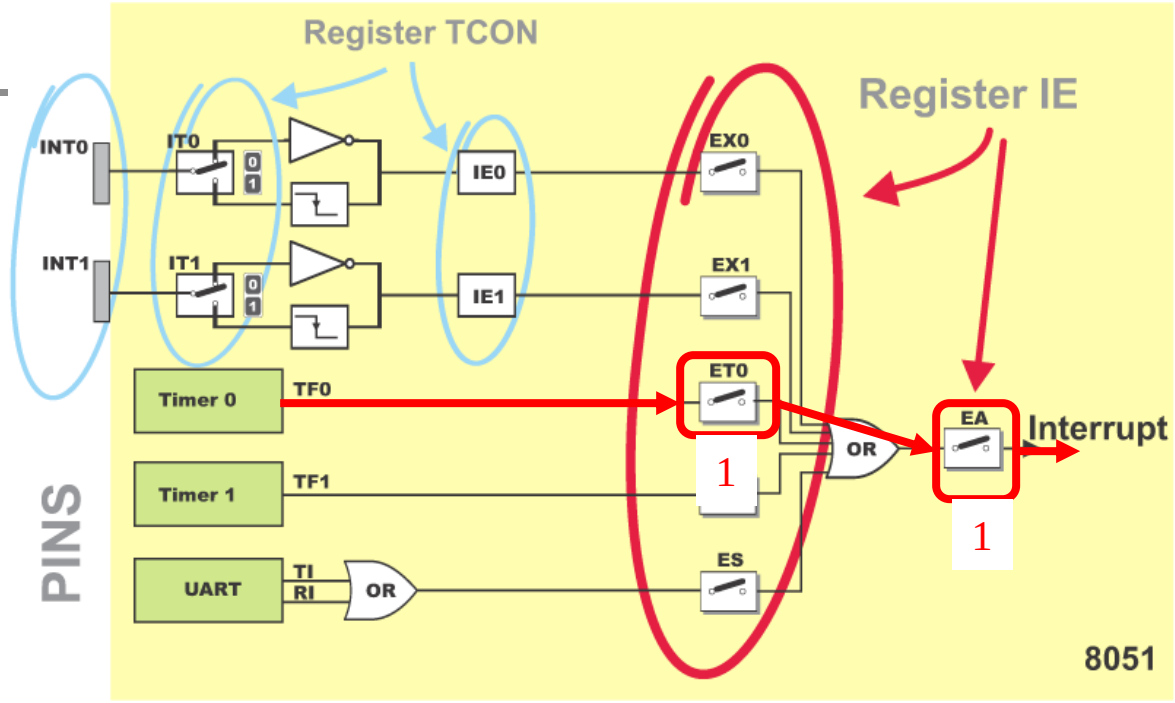
(1) How to set SFR registers

(2) Where to place interrupt service routine (ISR)?



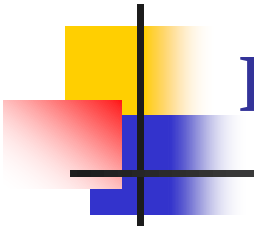
The IE register





- imagine a path in the figure
- set switches in the figure to enable the path

Where's the interrupt service routing?



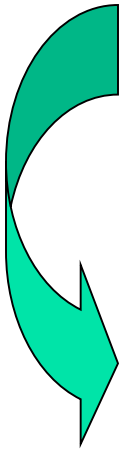
- IE0: 0x3 (external interrupt)
- TF0: 0xb (timer 0 overflow)
- TF1: 0x1b (timer 1 overflow)
- RI, TI: 0x23 (for UART)

So your program will looks like

```
        org 0H
        AJMP MAIN
        org 0Bh
        AJMP show_LED
        …

MAIN:
        …//your main program


show_LED:  //the tmer interrupt service routine
          MOV R0, #LED_pattern
          MOV P0, R0
          …
```

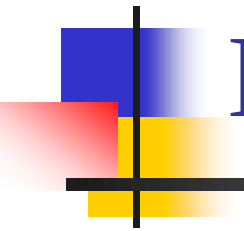
assembler directive: place my code from address 0x0b

So your program will looks like

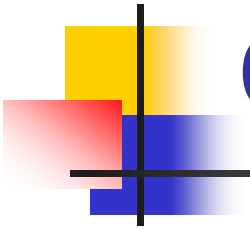


```
        org 0H
        AJMP MAIN
        org 0Bh
        AJMP show_LED
        …

MAIN:
        …//your main program


show_LED:  //the tmer interrupt service routine
          MOV R0, #LED_pattern
          MOV P0, R0
          …
```

timer 0 ISR starts from here



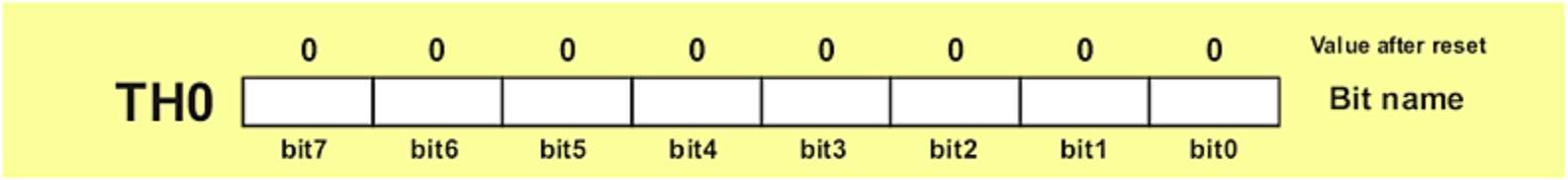
How to program 8051 timer

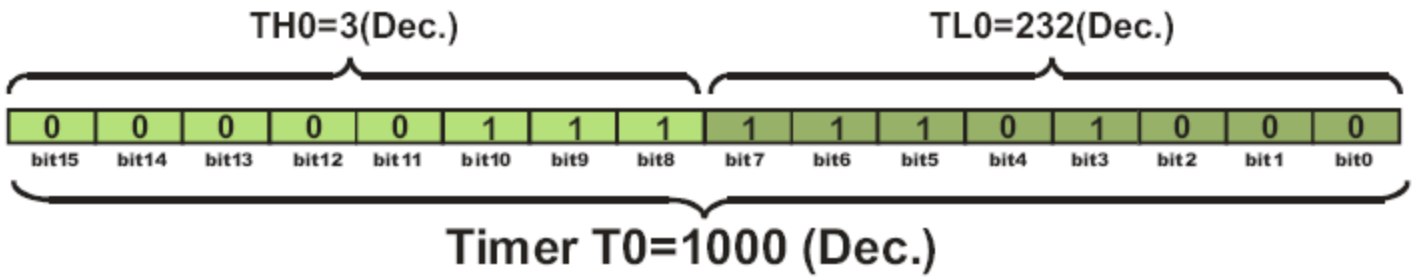


Overview of 8051 timer

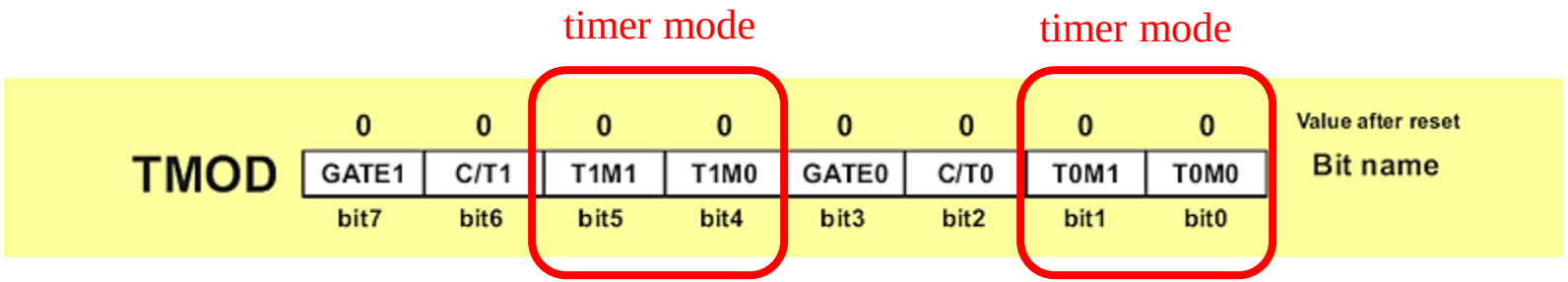
- two timers:
  - timer 0: {TH0, TL0}
  - timer 1: {TH1, TL1}
- four modes (set by TMOD register)
  - 0: 13-bit mode
  - 1: 16-bit mode
  - 2: auto reload mode
  - 3: split mode

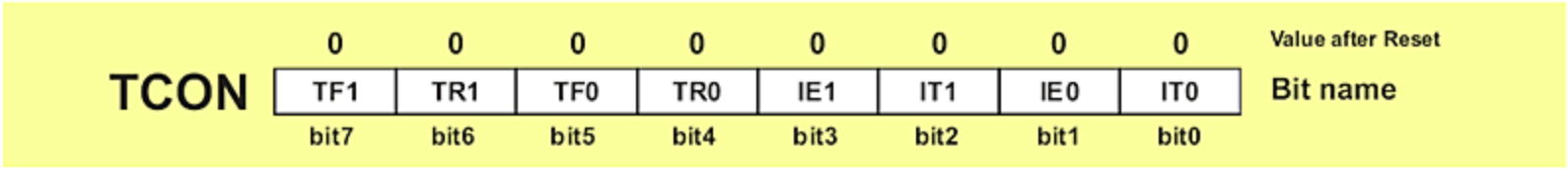
The SFRs for counting





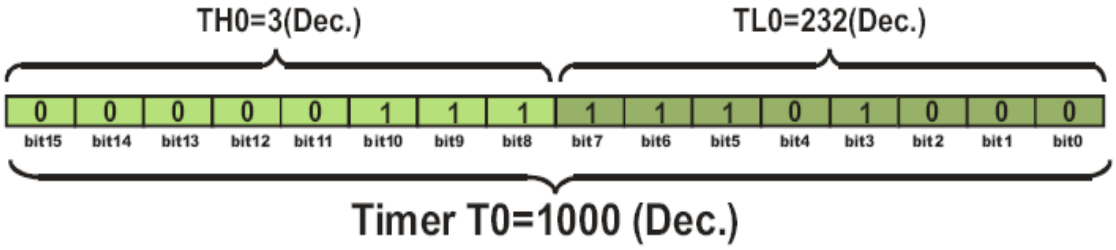
Registers to control the timer mode

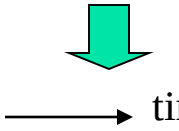




How 8051 timer works

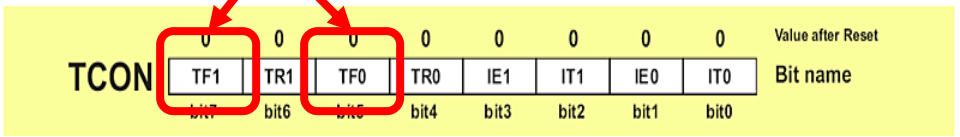
Step 1: set {TH, TL}=N
Step 2: enable counting by setup TMOD, TCON
Step 3: wait for timer overflow (check TCON)





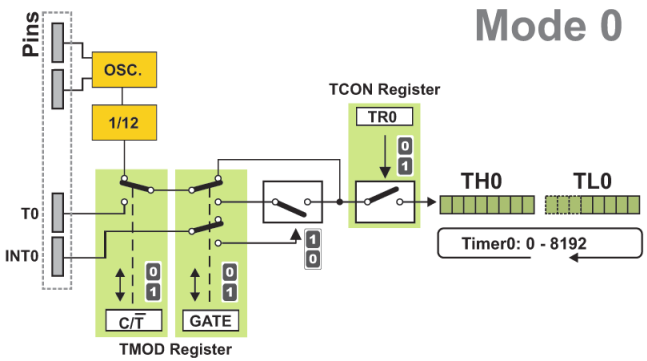
1000	1001	1002	…	0xffff	0

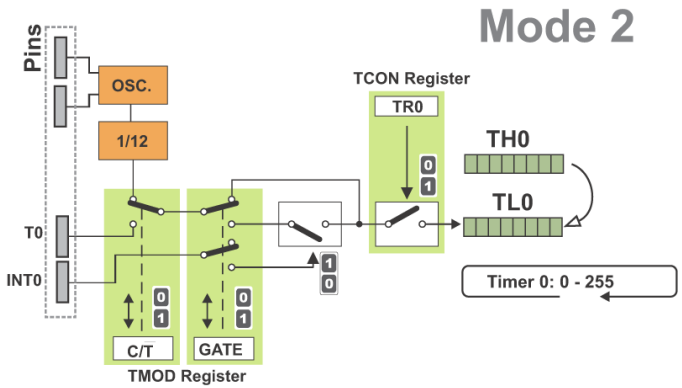
TF=1 to indicate timer overflow (0xffff reached)

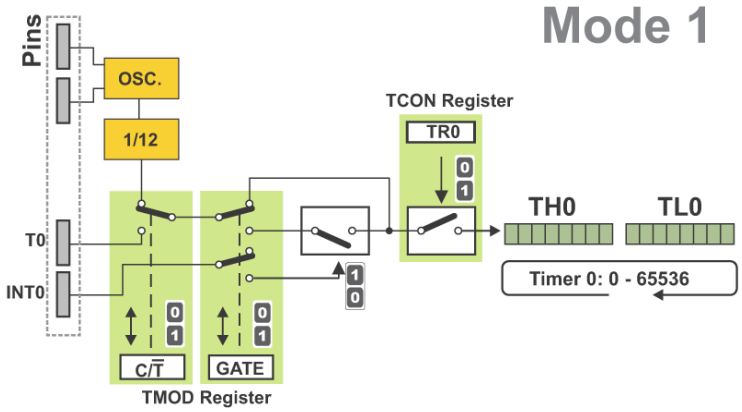


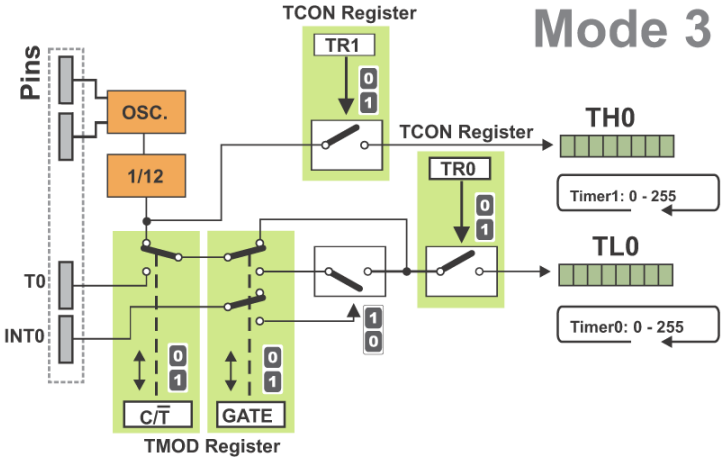
How to set lots of bits in TMOD, TCON, and IE

check these figures for the four timer modes

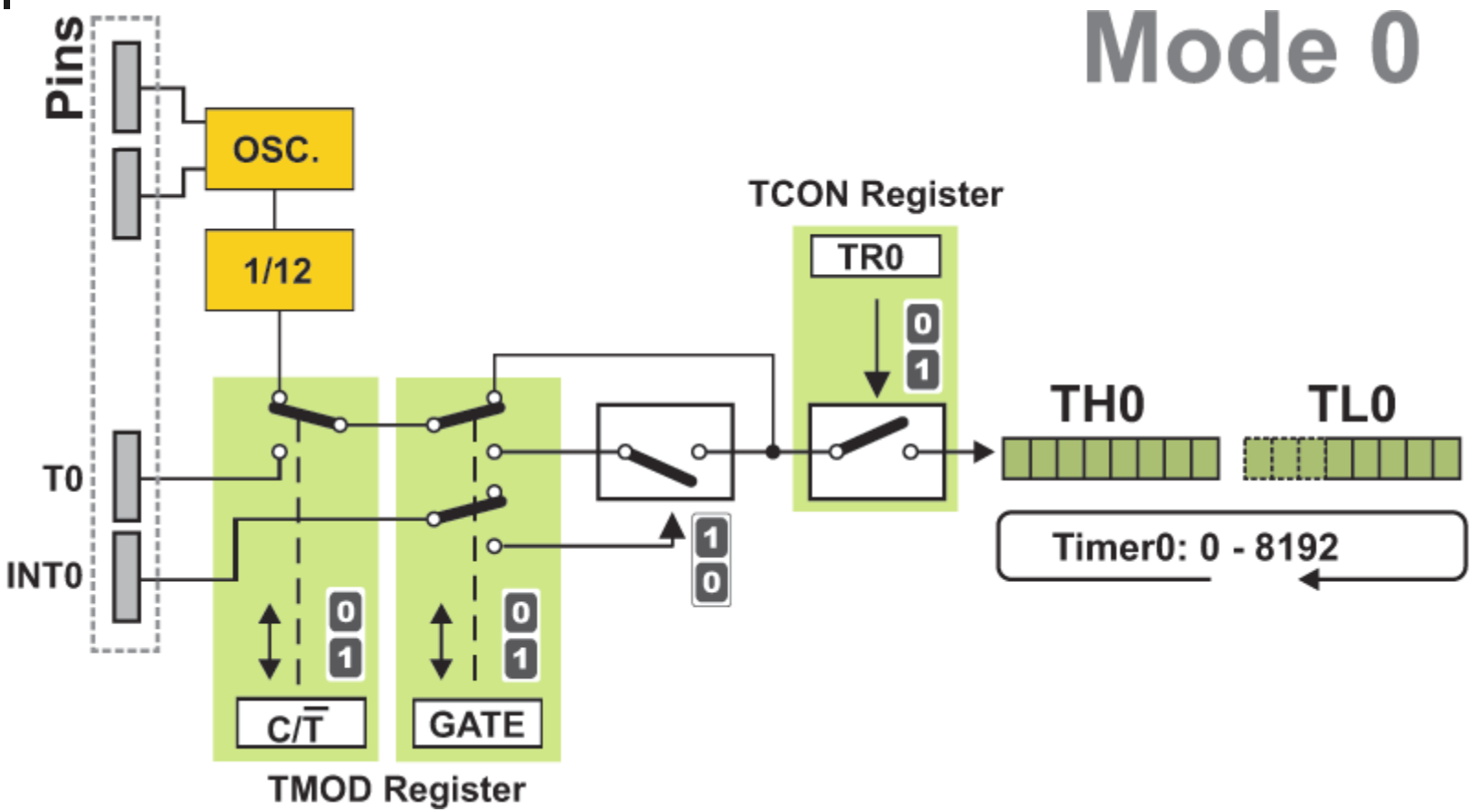




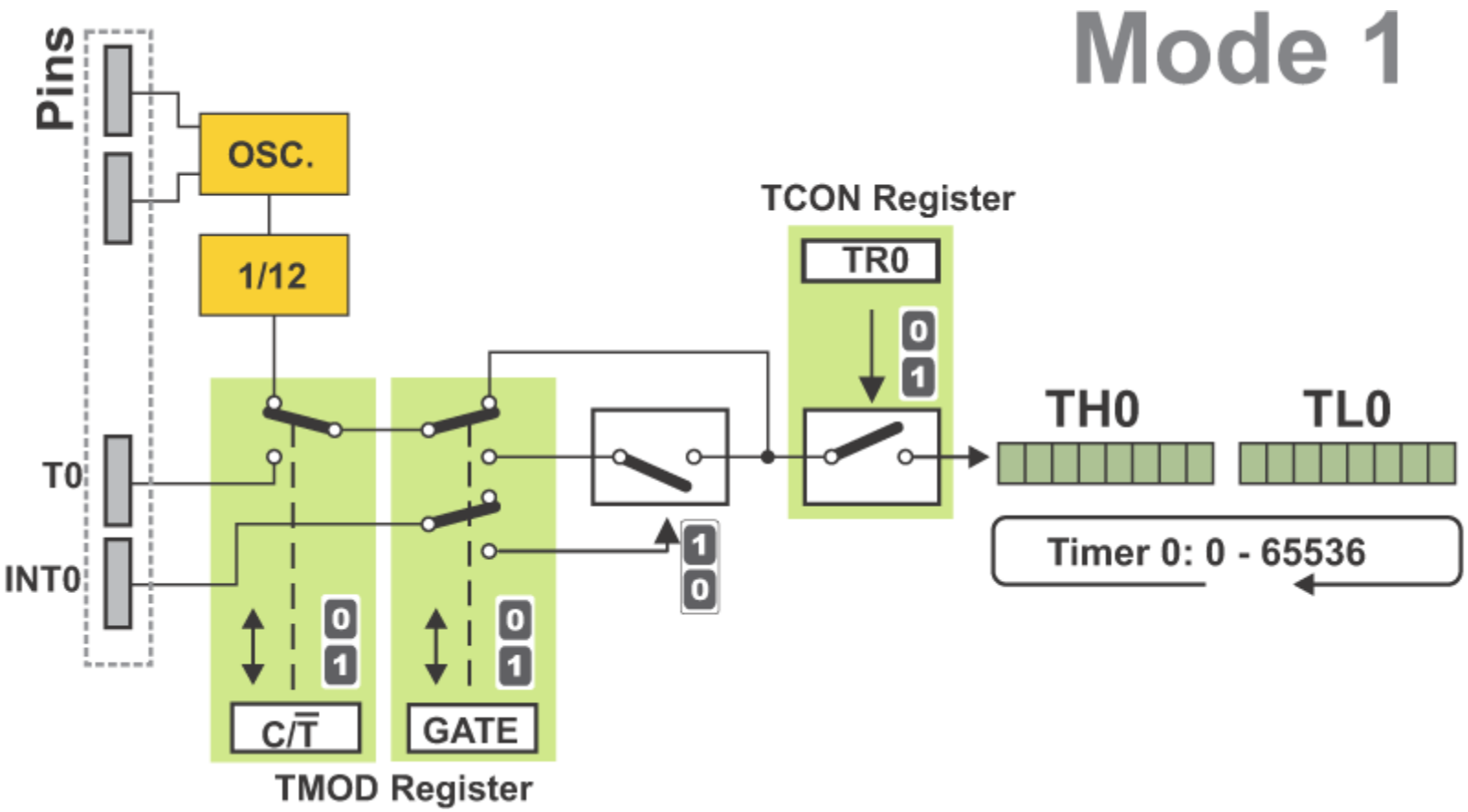




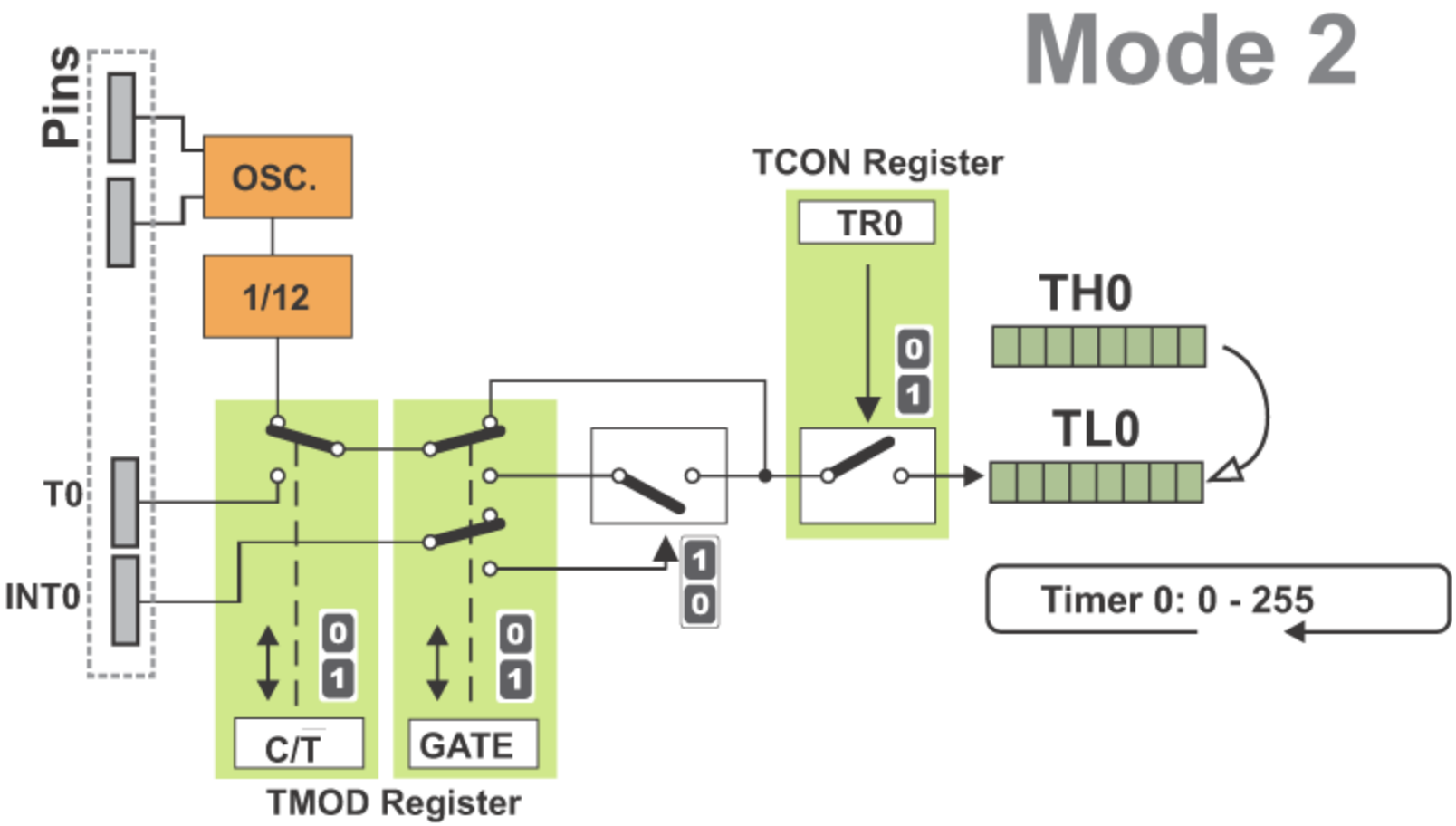
Timer Mode 0


Mode 0
Pins
OSC.
1/12
TCON Register
TR0
0
1
TH0
TL0
T0
INT0
1
0
Timer0: 0 - 8192
0
1
C/T
0
1
GATE
TMOD Register

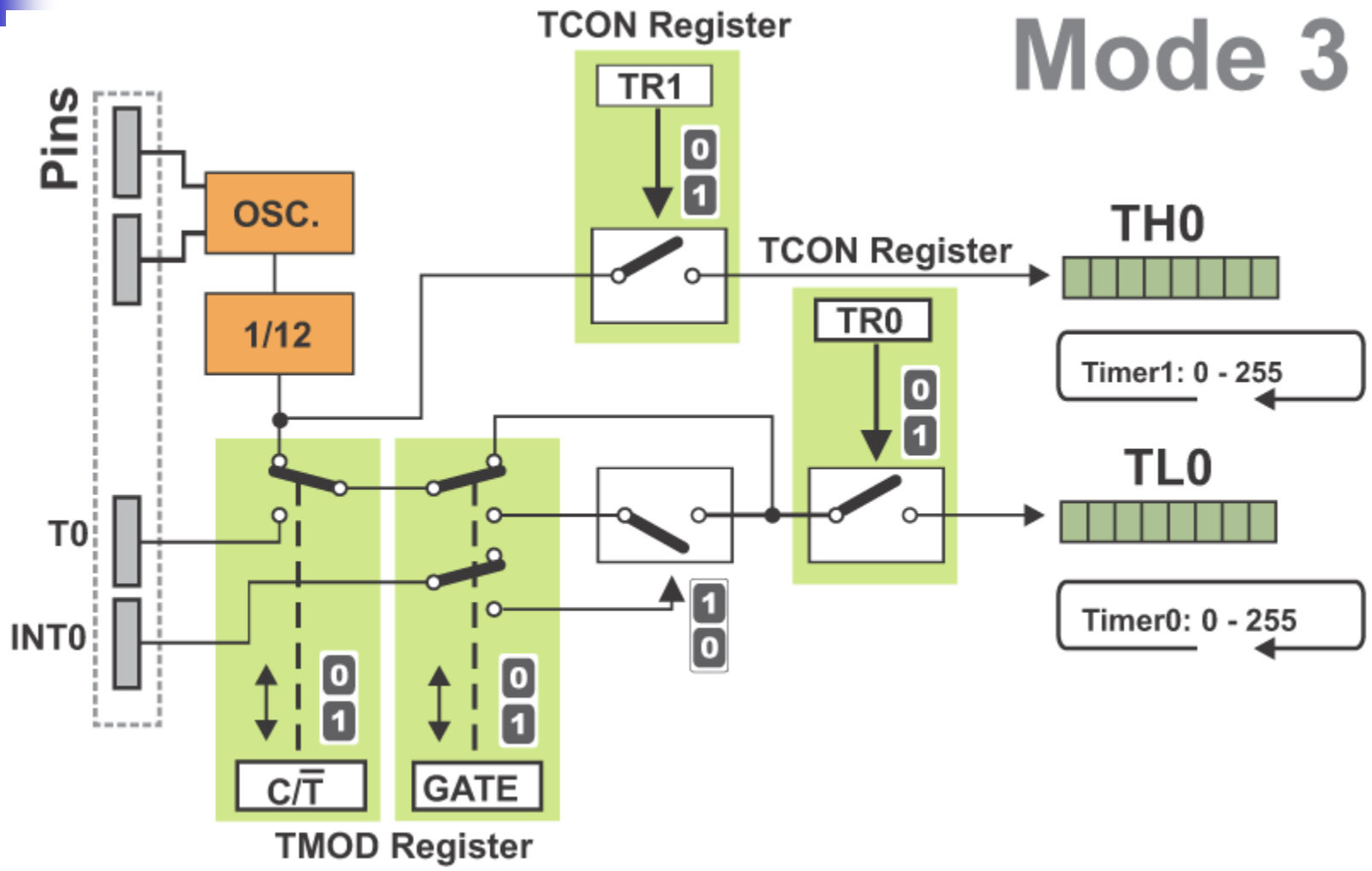
Timer Mode 1


Mode 1
Pins
OSC.
1/12
TCON Register
TR0
0
1
TH0
TL0
T0
INT0
1
0
Timer 0: 0 - 65536
0
1
0
1
C/T
GATE
TMOD Register

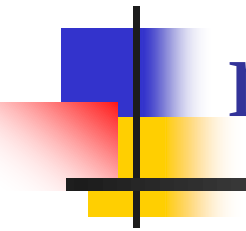
Timer Mode 2


Mode 2
Pins
OSC.
1/12
T0
INT0
C/T
GATE
TMOD Register
TCON Register
TR0
TH0
TL0
Timer 0: 0 - 255
0
1

Timer Mode 3

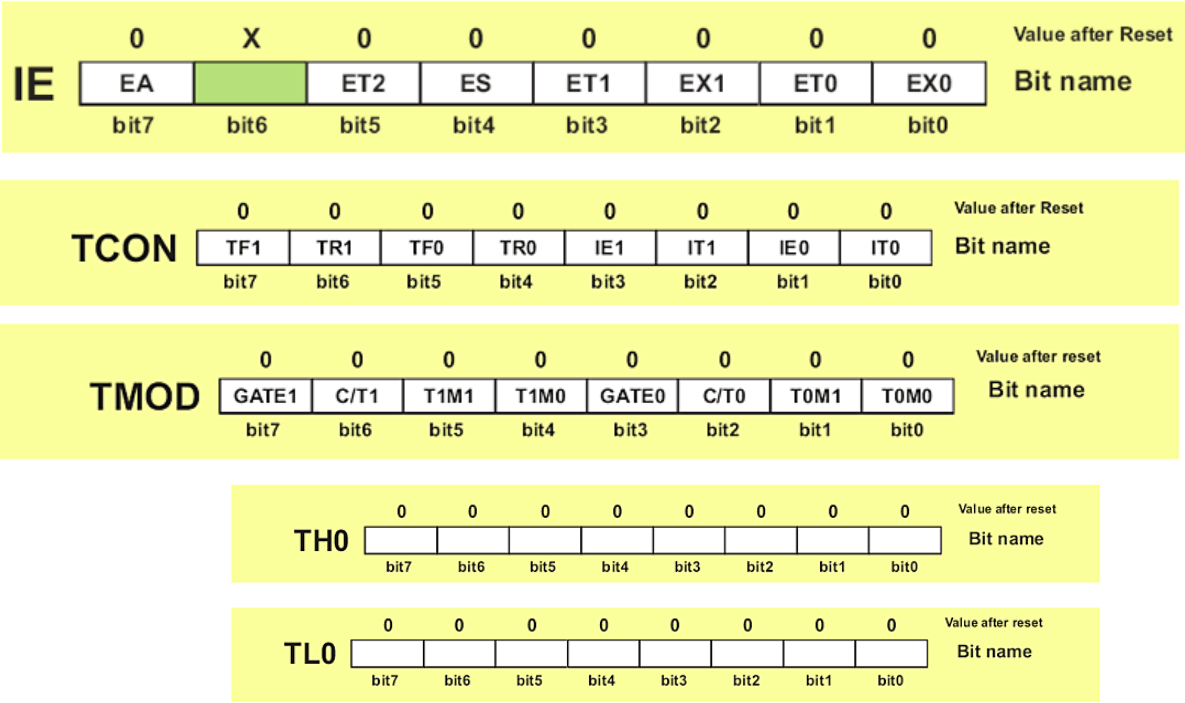

Mode 3
TCON Register
TR1
0
1
Pins
OSC.
1/12
TCON Register
TH0
TR0
Timer1: 0 - 255
T0
INT0
TL0
Timer0: 0 - 255
C/T
GATE
TMOD Register

Exercise: setup timer control registers



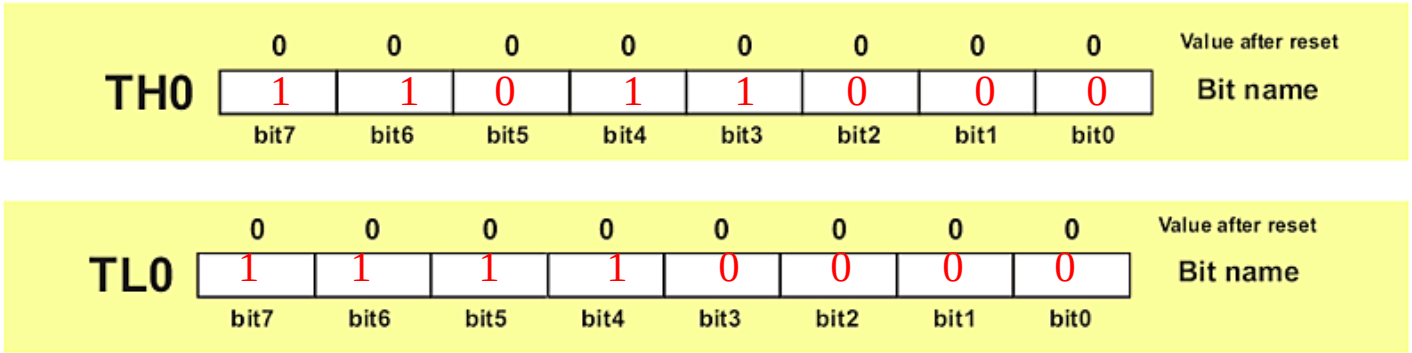
In-Class Exercise

- Suppose:
  - one cycle period of the timer counter is 0.1ms
- Q: How to program 8051 to send an interrupt every 1 second?



The SFR setup

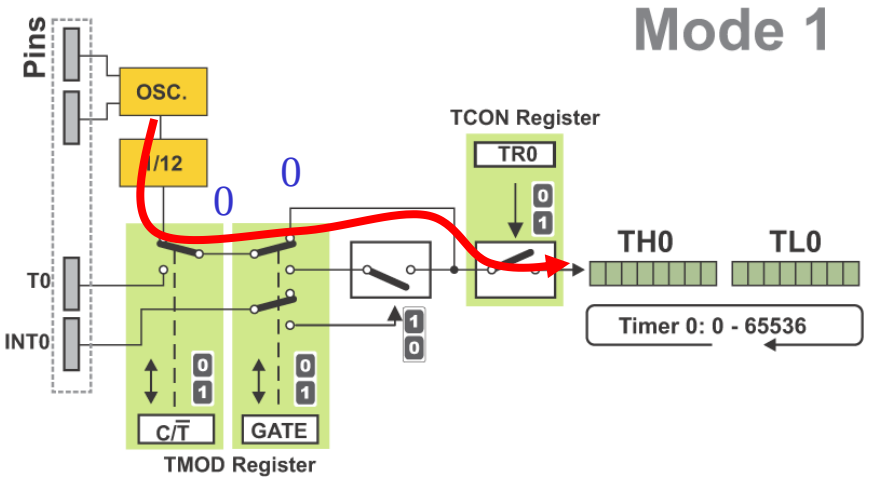
- setup the counter
  - count once every 0.1 ms
  - count 10000 times for 1 second
  - {TH0, TL0}=65536 – 10000 = 55536 = $(1101100011110000)_2$



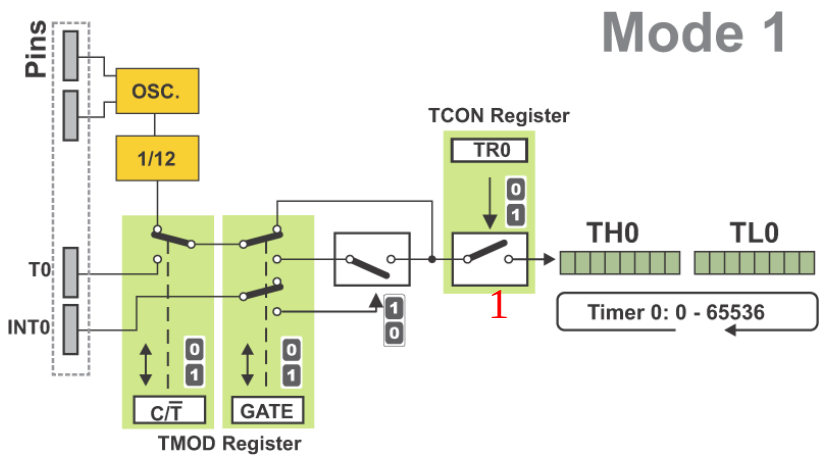
The SFR setup

- use timer 0 with mode 01
  - mode 01: 16-bit timer

TMOD	GATE1	C/T1	T1M1	T1M0	GATE0	C/T0	T0M1	T0M0	Value after reset
	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Bit name
	0	0	0	0	0	0	0	0	
					0	0	0	1	

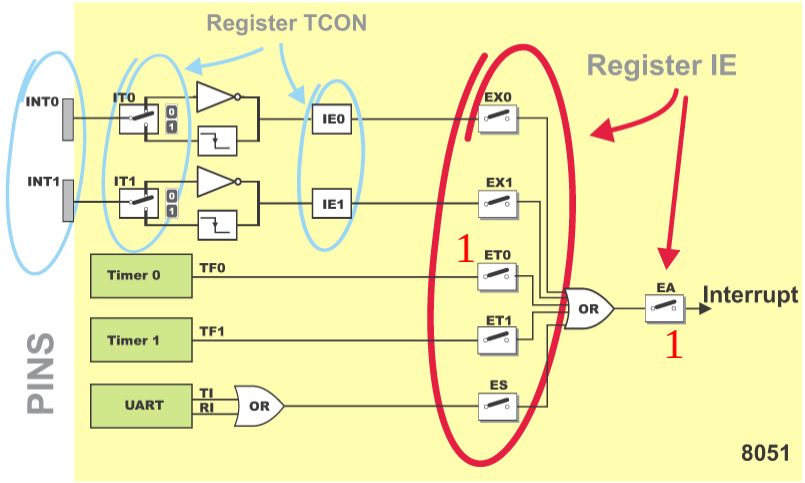


The SFR setup



	0	0	0	0	0	0	0	0	Value after Reset
TCON	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	Bit name
	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	

1



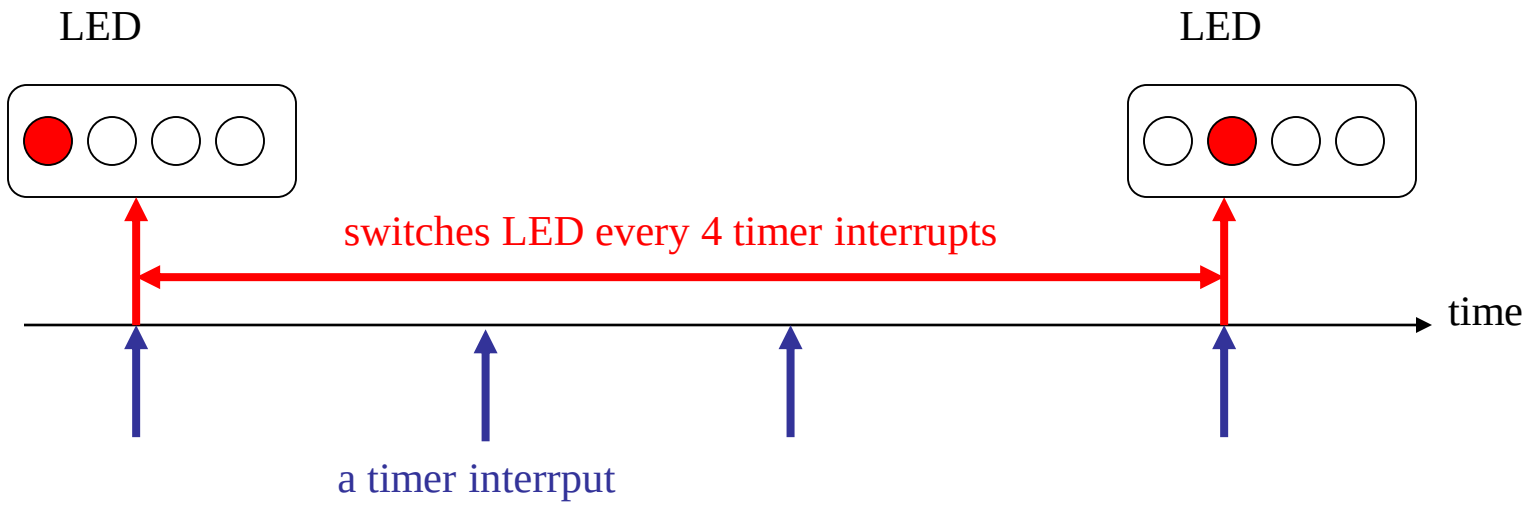
	0	X	0	0	0	0	0	0	Value after Reset
IE	EA		ET2	ES	ET1	EX1	ET0	EX0	Bit name
	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	

1 1

Demo: make LED run using the timer interrupt

Function of the demo

- switches LED every 4 timer interrupts



The demo program

- org to force program address
- and jump to actual ISR immediately

```
        org     0h
        ljmp    main

        org     0bh
        ljmp    Timer0_ISR

        org     0100h
main:
        lcall   Port_Config
        lcall   Timer_Config
        mov     R0, #4          ;the ISR entrance count
        mov     R1, #80h        ;the LED pattern to display
loop:
        mov     P2, R1
        sjmp    loop
```

The demo program

- Infinite loop to send control signals to LEDs

```
        org     0h
        ljmp    main

        org     0bh
        ljmp    Timer0_ISR

        org     0100h
main:
        lcall   Port_Config
        lcall   Timer_Config
        mov     R0, #4          ;the ISR entrance count
        mov     R1, #80h        ;the LED pattern to display
loop:
        mov     P2, R1
        sjmp    loop
```

The demo program

- The timer ISR
- Change LED pattern every 4 times the ISR is executed

```
Timer0_ISR:
        DJNZ    R0, reset_timer

        mov     R0, #4
        mov     A, R1
        RL      A
        mov     R1, A

reset_timer:
        mov     TL0, #0
        mov     TH0, #0
        reti
        end
```

The demo program

- Setup the timer interrupt
- Q: what is CKCON?

```
Timer_Config:
        mov     TMOD, #01h
        mov     TCON, #010h
        mov     CKCON, #010h
        mov     IE, #082h
        mov     TL0, #0
        mov     TH0, #0
        ret
```

The demo program

- Setup port configuration

```
Port_Config:
        ;turn-off the watch-dog timer
        mov     WDTCN, #0deh
        mov     WDTCN, #0adh

        ;setup port configuration
        mov     SFRPAGE, #CONFIG_PAGE
        mov     XBR2, #0c0h
        mov     P1MDIN, #0ffh
        mov     P2MDOUT, #0ffh
        mov     SFRPAGE, #LEGACY_PAGE
        ret
```

ONE PIECE

- 我也認真看過學長姊的版本了，其實都不是正確的……
- 範例程式中用：

  (4次timer interrupts) x (65536次計數)

  來代表一秒，其實是不精準的，這個時間事實上略長於一秒鐘。

ONE PIECE

- 所以各位加分的寶藏就在那裡了！只要在預報中寫道如何正確地做出一秒鐘的延遲，並在實驗中展現出來，實驗三的總成績加10分。
- 不要問我怎麼做，這是加分題。做完實驗後的下一周我會告訴你怎麼做。

ONE PIECE

- 給點提示：
  - 我的算式：

    65536 – (24.5Mhz / 8 / 12 / 跑4次) = 1734

    每次timer reset不該設成0，應是1734 (Dec.)
  - 關鍵字：System Clock, SYSCLK, OSCICN, CKCON
  - 看電子書，看技術手冊絕對有幫助
  - 別亂掰答案，亂掰的沒有加分

Lab03 Study Report

- File name: Bxxxxxxx-MCE-Lab3-Study
- File type: PDF only
- The requirements of report
  - Summarize the content of this slide set
  - Provide your plan for this lab exercise
  - No more than one A4 page
  - Grading: 80 ± 15
- Deadline: 2021/11/10 23:00 (不收遲交)
- Upload to e-learning system

Lab03 Lab Exercise Report

- File name: Bxxxxxxx-MCE-Lab3-Result
- File type: PDF only
- The requirements of report
  - Summarize the problems and results you have in this exercise
  - Some screen shots or some code explanation can be provided
  - No more than two A4 pages
  - Grading: 80 ± 15
- Deadline: 2021/11/17 23:00 (不收遲交)
- Upload to e-learning system