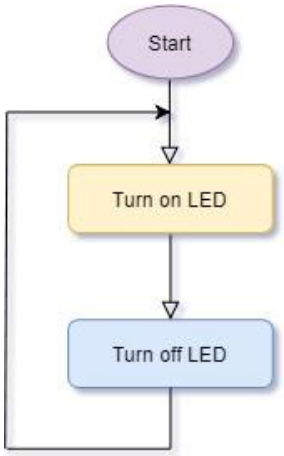
* Labta ortak bilgisayar kullanımı kesinlikle kabul edilmeyecektir.
* The use of shared computers in the lab will not be accepted.

EXPERIMENT 5: Time Delays

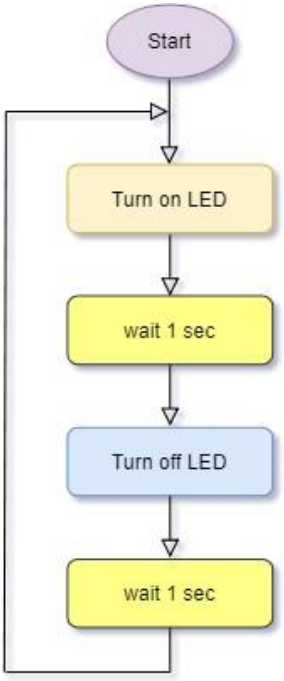
Why We Need Time Delays?

There are times when it is necessary to wait before executing the next part of a program. For this discussion we will take the example of flashing an LED.



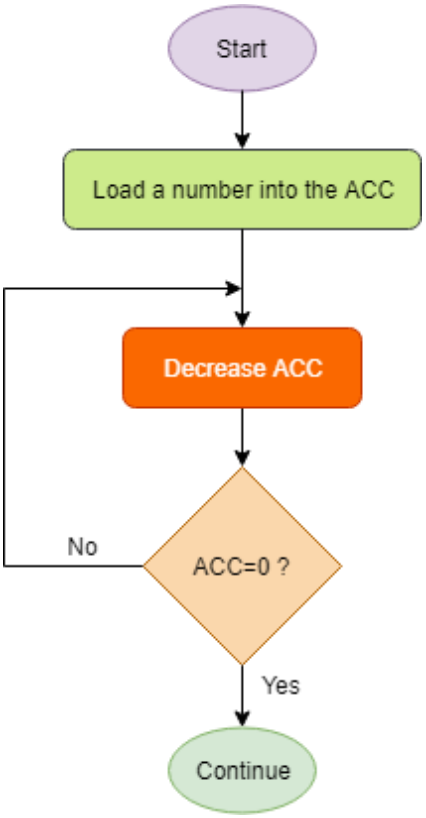
The above flowchart details program for flashing an LED. It works by turning on the LED, turning it off and then looping back to the start again. This program is in a never-ending loop.

However, while the above program may flash an LED, because of the speed at which microcontrollers operate, the frequency would be too fast for the human eye to see.



Above is a modification of the LED flashing program. It turns on the LED, waits a second, turns it off, waits another second and then loops back to the start again.

One method of forcing the microcontroller to wait is the time delay. The flow chart for a single- loop time delay is shown below.



A number is loaded into the accumulator. It is then decremented. Following this, a check on the contents of the accumulator is made. If the contents of the accumulator is not equal to zero a jump back to the decrement instruction is made.

If **0AH** is loaded into the accumulator, then the loop is executed ten times. To increase the length of the time delay we load a larger number into the accumulator. Therefore, the longest time delay (using this single loop time delay) is achieved by loading the accumulator with **FFH (**the loop would be executed 255 times) .

Now there is a question to be answered. One execution time is how many seconds?

Knowledge of the microcontroller clock cycle time is useful in defining timing events used in applications.

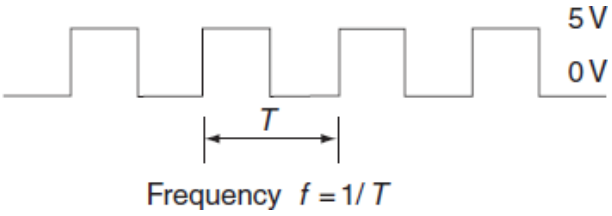
Some AT89 Instruction Set and Their Oscillator Periods

Mnemonic	Oscillator Period
MOV	12
ADD	12
SUBB	12
INC	12
DEC	12
MUL	48
DIV	48
ANL	12
ORL	12
XRL	12
JC	24
JNC	24
JB	24
JNB	24
JBC	24
ACALL	24
LCALL	24
RET	24
RETI	24
JMP	24
JZ	24
JNZ	24
DJNZ	24
NOP	12
CLR	12
SETB	12

Example 3.9 A AT89C52 microcontroller has a clock frequency of 11.0592 MHz. What is the time for each cycle?

Solution

$$\text{CycleTime} = \frac{1}{f} = \frac{1}{11.0592x10^{6}Hz} = 90.4225\ ns$$



Frequency $f = 1/T$

Figure 3.3 Square-wave signal at a frequency f Hz

Time Delays

The NOP instruction is a simple time delay but apart from this there are two methods of creating time delays:

- register decrement
- onboard timers

The use of onboard timers will be described in a later experiment; here the register decrement method will be described.

The basic single loop program lines are:

```
MOV R0, #number ; move a number into an 8-bit register R0
DJNZ R0, $ ; keep decrementing R0 until it is zero
```

MOV takes 12 clock cycles, DJNZ takes 24 clock cycles. The delay time is (12+(numberx24))xClock cycles.

Delay time = (12+(numberx24))xClock cycles

Example 3.10 A AT89C52 microcontroller has an 11.0592 MHz crystal-controlled clock oscillator. Write an assembly language program that will generate a 5 kHz squarewave signal on pin 7 of port 1.

Solution

Clock frequency = 11.0592 MHz

Thus period of clock cycle = (1/11.0592 MHz) =90.4225 ns Signal frequency = 5 kHz

Therefore period of signal cycle = (1/5 kHz) = 200 μs

The delay required is half of this value since the square wave has an equal logic 1/logic 0 time. See Figure 3.3.

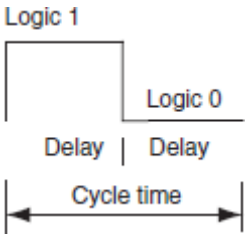


Figure 3.3 Delay period determination for a square-wave

signal Delay = 100 μs = (12+(numberx24))x90.4225ns

Hence, number = ((100 μs/90.4225ns)-12)/24=45.58 ≅ 46 decimal (to the nearest whole number).

```
        ORG 0               ; sets start address to 0
jump:   SETB P1^7           ; set pin 7 on port 1 to logic 1
        MOV R0, #46D        ; move 46 decimal number into an 8-bit register R0
        DJNZ R0, $          ; keep decrementing R0 until it is zero
        CLR P1^7            ; clear pin 7 on port 1 to logic 0
        MOV R0, #46D        ; move 46 decimal number into an 8-bit register R0
        DJNZ R0, $          ; keep decrementing R0 until it is zero
        JMP jump            ; it takes 24 clock cycles
        END                 ; no more assembly language
```

Exercise 3.5 Using the techniques above, assuming the clock frequency is 11.0592 MHz, write a program to generate a pulse of 20 kHz on pin 5 of port 2 of the microcontroller.

Delay with using subroutine

```
DELAY:  MOV R0,#34          ; move decimal 34 into register R0
TAKE:   DJNZ R0,TAKE        ; keep subtracting 1 from R0 until zero
        RET                 ; return from subroutine
```

MOV takes 12 clock cycles, DJNZ and RET each take 24 clock cycles. The delay is called up from the main program using ACALL, which takes 24 clock cycles. The delay time is $(24_{ACALL}+12_{MOV}+(numberx24)_{DJNZ}+24_{RET})$xClock cycles.

Delay time = (24+12+(numberx24)+24)xClock

cycles Delay time = (60+(numberx24))xClock

cycles

Example 3.11 A AT89C52 microcontroller has an 11.0592 MHz crystal-controlled clock oscillator. Write an assembly language program that will generate a 8 kHz squarewave signal on pin 3 of port 1.

Solution

Clock frequency = 11.0592 MHz

Thus period of clock cycle = (1/11.0592

MHz) =90.4225 ns Signal frequency = 8

kHz

Therefore period of signal cycle =

(1/8 kHz) = 125 μs Delay = 125μs/2

= (60+(numberx24))x90.4225ns

Hence number = ((125μs/(2x90.4225ns))-60)/24=26.3≅ 26 decimal (to the nearest whole number).

```
        ORG 0           ; sets start address to 0
START:  SETB P1^3       ; set pin 3 on port 1 to logic 1
        ACALL DELAY
        CLR P1^3        ; clear pin 3 on port 1 to logic 0
        ACALL DELAY
        JMP START       ; go to START
DELAY:  MOV R0, #26D    ; move a number into an 8-bit register R0
TAKE:   DJNZ R0, TAKE   ; keep decrementing R0 until it is zero
        RET             ; return from DELAY subroutine
        END             ; no more assembly language
```

Exercise 3.6 Using the techniques above, assuming the clock frequency is 11.0592 MHz, write a program to generate a pulse of 2.5 kHz on pin 2 of port 1 of the microcontroller.

Longer Delays

The longest single-loop delay is achieved by initialising the decrement register (ie; the register that will be decremented) to FFH. To achieve longer delays we can use multiple-loop time delays, as detailed in the flow chart below.