

Call Instructions:

In 8051 microcontroller instruction set we have two call instructions, LCALL and ACALL. The call instructions are used to make program more structured and save memory space. For example, we have a set of code that is being used again and again so its better to write its subroutine in some suitable space and call it whenever needed instead of writing it again and again.

LCALL instruction:

Just like LJMP this instruction is called as “Long Call”. It is a 3 Bytes instruction with 1B for opcode and 2B to store the Address that could be anywhere in the 64kB of memory.

Syntax: LCALL Label

This instruction:

1. Push the address of the instruction next to the LCALL statement on to the stack.
2. Jump to the label location
3. Execute the subroutine at the label
4. Upon encountering the RET instruction POP the address Pushed previously from the stack
5. Resume the execution of the main program.

Example: Write a program that toggles the value of “0FH”and “F0H” at port 1?

Solution:

```

ORG 00H
BACK: MOV A, # 0FH
      MOV P1, A
      LCALL DELAY ; this instruction will push the address of
                  MOV A, # F0H on to stack and jump to label
                  Delay.
      MOV A, # 0F0H
      MOV P1, A
      LCALL DELAY ; this instruction will push the address of
                  SJMP BACK on to stack and jump to label
                  Delay.

      SJMP BACK
; Delay subroutine that is being instructed to be stored at location 302H

ORG 302H
DELAY: MOV R0, #0FFH
HERE:  DJNZ R0, HERE
      RET ; Upon on first encounter this will pop the address of 1st
         call and then of 2nd.

END ;Always write END as the last line of code

```

Use of PUSH and POP instructions in Subroutines:

With every CALL instruction the CPU pushes the address of the next instruction on to the stack. Now, if we are going to use the same registers in the subroutine and the main program then its our job to keep track of every PUSH and POP instruction. As a rule of thumb for every PUSH there should be a POP.

Example: Write a program that uses R5 and R7 to store two numbers in main program and in subroutine they are used as a counter for loop?

Solution:

```

ORG 00H
BACK: MOV R5, #57H
      MOV R7, #75H
      MOV A, #0FH
      MOV P1, A
      LCALL DELAY
      MOV A, #00H
      ADD A, R5,
      ADD A, R7
      MOV R4, A
      MOV A, #0F0H
      MOV P1, A
      MOV A, #00H
      ADD A, R5,
      ADD A, R7
      MOV R4, A
      SJMP BACK

      ORG 302H
DELAY:  PUSH 5      ; We want to save contents of R5 after this call
        PUSH 7      ; Same is here as we need the contents of R7 after call
        MOV R5, #10
AGAIN:  MOV R7, #70
HERE:   DJNZ R7, HERE
        DJNZ R5, AGAIN
        POP 7        ; Retrieve contents of stack at this location to R7
        POP 5        ; Retrieve contents of stack at this location to R5
        RET
      END

```

TABLE: Registers and Stack Status

Before LCALL		After LCALL		
SP	07	Stack Pointer	75	
R5	57H	0B	75	Contents of R7
R7	75H	0A	57	Contents of R5
		09	00	Address higher Byte
		08	0C	Address lower Byte

ACALL:

Absolute call is a 2 Bytes instruction with 1B for Opcode and 1B to store the address of the label or the subroutine called. It saves memory as it needs only 2B and normally in our programming the code lies between the 2B spaces.

Syntax: **ACALL Label**

This instruction:

1. Push the address of the instruction next to the ACALL statement on to the stack.
2. Jump to the label location
3. Execute the subroutine at the label
4. Upon encountering the RET instruction POP the address Pushed previously from the stack
5. Resume the execution of the main program.

Example: Re-write the program that toggles the value of “0FH” and “F0H” at port 1 using less memory space?

Solution:

```

ORG 00H
BACK: MOV A, # 0FH
      MOV P1, A
      ACALL DELAY ; this instruction will push the address of
                   MOV A, # F0H on to stack and jump to label
                   Delay.

      MOV A, # 0F0H
      MOV P1, A
      ACALL DELAY ; this instruction will push the address of
                   SJMP BACK on to stack and jump to label
                   Delay.

      SJMP BACK
; Delay subroutine

DELAY: MOV R0, #0FFH
HERE:  DJNZ R0, HERE
      RET ; Upon on first encounter this will pop the address of 1st
          call and then of 2nd.

```

END ;Always write END as the last line of code

Benefits of using CALLS:

Beside making the program more structured and avoiding the re-writing of same code again and again the major benefit comes in the shape of saving memory. The memory is saved by avoiding writing the same code again and again. Also by dividing complex programs in to smaller subroutines one can assign the task to write a specific subroutine to different departments or programmers of an organization.

By Engr. Mansoor Habib

How to Create Delay in Microcontroller Programming?

Whenever we are executing some lines of code repeatedly this introduces delay in the process of execution. The repeated execution can take place because of a loop or just to give some time before executing the next instruction. The lines of code written to do so are called as Delay or Delay Subroutine.

Machine Cycle:

Let's define it this way a clock cycle is the time between two adjacent pulses of the oscillator that sets the tempo of the computer processor. The number of these pulses per second is known as the clock speed, which is generally measured in MHz and these clock cycles in Microcontroller are called as *Machine Cycles*.

Each and every instruction takes 1 or more machine cycles to complete its execution. This requirement is normally given in the datasheets by the vendor of the microcontroller. Now the length or the time that a single machine cycles takes is defined by the frequency of the oscillator.

The frequency of the crystals that can be connected to 8051 microcontroller lies between 4MHz and 30MHz. Mostly 11.0592 MHz crystal is used because this is the compatible speed at which we can connect a microcontroller to the computer using serial port.

In 8051 microcontroller a machine cycle lasts for 12 oscillator periods so to calculate the machine cycle we use this formula:

$$\begin{aligned} \text{Crystal Frequency} / 12 &= F \\ \text{Time (T)} &= 1 / F \text{ (seconds)}. \end{aligned}$$

For example:

For an 11.0592MHz crystal the time to execute one MC (Machine Cycle) is:

$$11.0592\text{MHz} / 12 = 921.6\text{kHz}$$

$$T = 1/921.6\text{k} = 1.085\mu\text{S (microseconds)}.$$

Example: Find Delay in the following delay subroutine with a crystal of 11.0592MHz?

```

Org 00H
----
----
----
ACALL DELAY
----
----
----
DELAY: MOV R0, #100
HERE:  DJNZ R0, HERE
RET
End

```

Solution:

First of all calculate the time for one MC:

For an 11.0592MHz crystal the time to execute one MC (Machine Cycle) is:

$$11.0592\text{MHz} / 12 = 921.6\text{kHz}$$

$$T = 1/921.6\text{k} = 1.085\mu\text{s (microseconds).}$$

Now by looking at the tables we come to know that

MOV R0, #100 takes 1MC
 DJNZ R0, HERE takes 2MC and
 RET takes 2MC

So

Instruction	MC x Count x Time	Comments
MOV R0, #0FFH	1 x 1 x 1.085us	As this instruction will run only once
DJNZ R0, HERE	2 x 256 x 1.085us	This instruction will run 256 times because of the loop
RET	2 x 1 x 1.085us	As this instruction will run only once
Total Delay	558.775us	Add all of them

Example: Find the time delay introduced by the “DELAY” Subroutine in the following program?

```

ORG 00H
BACK: MOV A, # 0FH
      MOV P1, A
      ACALL DELAY
      MOV A, # 0F0H
      MOV P1, A
      ACALL DELAY
      SJMP BACK
; Delay subroutine

DELAY: MOV R1, #07
OUTER: MOV R0, #10
HERE:  DJNZ R0, HERE
      DJNZ R1, OUTER
      RET

```

END

; Always write END as the last line of code

Solution:

First of all find the counter in the subroutine:

R0 with “07” and R1 with “10”

Because of this counter the “DJNZ R0, HERE” will be forced to execute for “70” times and R1 will execute “DJNZ R1, OUTER” only 7 times

Instruction	MC x Count x Time	Comments
MOV R1, #07	1 x 1 x 1.085us	As this instruction will run only once
MOV R0, #10	1 x 7 x 1.085us	This instruction will run 7 times because of the outer loop
DJNZ R0, HERE	2 x (10 x 7) x 1.085us	This instruction will run 70 times because of the nested loop
DJNZ R1, OUTER	2 x 7 x 1.085us	This instruction will run 7 times because of the loop by R1
RET	2 x 1 x 1.085us	As this instruction will run only once
Total Delay	177.94us	Add all of them