## CALL INSTRUCTIONS

- Call instruction is used to call subroutine
  - Subroutines are often used to perform tasks that need to be performed frequently
  - This makes a program more structured in addition to saving memory space

**LCALL** (long call)

- 3-byte instruction
  - First byte is the opcode
  - Second and third bytes are used for address of target subroutine
    - Subroutine is located anywhere within 64K byte address space

**ACALL** (absolute call)

- 2-byte instruction
  - 11 bits are used for address within 2K-byte range

CALL INSTRUCTIONS

LCALL

- When a subroutine is called, control is transferred to that subroutine, the processor
  - Saves on the stack the the address of the instruction immediately below the LCALL
  - Begins to fetch instructions form the new location
- After finishing execution of the subroutine
  - The instruction RET transfers control back to the caller
    - Every subroutine needs RET as the last instruction

# CALL INSTRUCTIONS

## LCALL (cont')

```
        ORG    0
BACK:   MOV    A,#55H     ;load A with 55H
        MOV    P1,A       ;send 55H to port 1
        LCALL  DELAY      ;time delay
        MOV    A,#0AAH    ;load A with AA (in hex)
        MOV    P1,A       ;send AAH to port 1
        LCALL  DELAY
        SJMP   BACK       ;keep doing this indefinitely

;---------- this is delay subroutine ------------
        ORG    300H       ;put DELAY at address 300H
DELAY:  MOV    R5,#0FFH   ;R5=255 (FF in hex), counter
AGAIN:  DJNZ   R5,AGAIN   ;stay here until R5 become 0
        RET               ;return to caller (when R5 =0)
        END
```

Upon executing "LCALL DELAY", the address of instruction below it, "MOV A,#0AAH" is pushed onto stack, and the 8051 starts to execute at 300H.

The counter R5 is set to FFH; so loop is repeated 255 times.

When R5 becomes 0, control falls to the RET which pops the address from the stack into the PC and resumes executing the instructions after the CALL.

The amount of time delay depends on the frequency of the 8051

## CALL INSTRUCTIONS

### CALL Instruction and Stack

```
0000                      ORG   0
0000 7455   BACK:   MOV   A,#55H  ;load A with 55H
0002 F590           MOV   P1,A    ;send 55H to p1
0004 120300         LCALL DELAY   ;time delay
0007 74AA           MOV   A,#0AAH ;load A with AAH
0009 F590           MOV   P1,A    ;send AAH to p1
000B 120300         LCALL DELAY
000E 80F0           SJMP  BACK    ;keep doing this
0010
0010 ;-------this is the delay subroutine------
0300                ORG   300H
0300        DELAY:
0300 7DFF           MOV   R5,#0FFH ;R5=255
0302 DDFE   AGAIN:  DJNZ  R5,AGAIN ;stay here
0304 22             RET            ;return to caller
0305                END             ;end of asm file
```

**Stack frame after the first LCALL**

| | |
|---|---|
| 0A | |
| 09 | 00 |
| 08 | 07 |

SP = 09

Low byte goes first and high byte is last

# CALL INSTRUCTIONS

## Use PUSH/POP in Subroutine

```
0000                   ORG   0
0000 7455    BACK:     MOV   A,#55H   ;load A with 55H
0002 F590              MOV   P1,A     ;send 55H to p1
0004 7C99              MOV   R4,#99H
0006 7D67              MOV   R5,#67H
0008 120300            LCALL DELAY    ;time delay
000B 74AA              MOV   A,#0AAH  ;load A with AA
000D F590              MOV   P1,A     ;send AAH to p1
000F 120300            LCALL DELAY
0012 80EC              SJMP  BACK     ;keeping doing
   this
0014 ;-------this is the delay subroutine------
0300                   ORG   300H
0300 C004    DELAY:    PUSH  4        ;push R4
0302 C005              PUSH  5        ;push R5
0304 7CFF              MOV   R4,#0FFH;R4=FFH
0306 7DFF    NEXT:     MOV   R5,#0FFH;R5=FFH
0308 DDFE    AGAIN:    DJNZ  R5,AGAIN
030A DCFA              DJNZ  R4,NEXT
030C D005              POP   5        ;POP into R5
030E D004              POP   4        ;POP into R4
031                                              er
031                                              e
```

Normally, the number of PUSH and POP instructions must always match in any called subroutine

After first LCALL

| | | |
|---|---|---|
| 0B | | |
| 0A | | |
| 09 | 00 | PCH |
| 08 | 0B | PCL |

After PUSH 4

| | | |
|---|---|---|
| 0B | | |
| 0A | 99 | R4 |
| 09 | 00 | PCH |
| 08 | 0B | PCL |

After PUSH 5

| | | |
|---|---|---|
| 0B | 67 | R5 |
| 0A | 99 | R4 |
| 09 | 00 | PCH |
| 08 | 0B | PCL |

# CALL INSTRUCTIONS

## Calling Subroutines

```
;MAIN program calling subroutines
        ORG   0
MAIN:   LCALL          SUBR_1
        LCALL          SUBR_2
        LCALL          SUBR_3

HERE:   SJMP           HERE
;-----------end of MAIN

SUBR_1: ...
        ...
        RET
;-----------end of subroutine1

SUBR_2: ...
        ...
        RET
;-----------end of subroutine2

SUBR_3: ...
        ...
        RET
;-----------end of subroutine3
        END            ;end of the asm file
```

It is common to have one main program and many subroutines that are called from the main program

This allows you to make each subroutine into a separate module

- Each module can be tested separately and then brought together with main program
- In a large program, the module can be assigned to different programmers

CALL INSTRUCTIONS

ACALL

- The only difference between `ACALL` and `LCALL` is
  - The target address for `LCALL` can be anywhere within the 64K byte address
  - The target address of `ACALL` must be within a 2K-byte range
- The use of `ACALL` instead of `LCALL` can save a number of bytes of program ROM space

## CALL INSTRUCTIONS

## ACALL (cont')

```
        ORG   0
BACK:   MOV   A,#55H     ;load A with 55H
        MOV   P1,A       ;send 55H to port 1
        LCALL DELAY      ;time delay
        MOV   A,#0AAH    ;load A with AA (in hex)
        MOV   P1,A       ;send AAH to port 1
        LCALL DELAY
        SJMP  BACK       ;keep doing this indefinitely
         ...
        END              ;end of asm file
```

A rewritten program which is more efficiently

```
        ORG   0
        MOV   A,#55H     ;load A with 55H
BACK:   MOV   P1,A       ;send 55H to port 1
        ACALL DELAY      ;time delay
        CPL   A          ;complement reg A
        SJMP  BACK       ;keep doing this indefinitely
         ...
        END              ;end of asm file
```

TIME DELAY FOR VARIOUS 8051 CHIPS

- CPU executing an instruction takes a certain number of clock cycles
  - These are referred as to as *machine cycles*
- The length of machine cycle depends on the frequency of the crystal oscillator connected to 8051
- In original 8051, one machine cycle lasts 12 oscillator periods

Find the period of the machine cycle for 11.0592 MHz crystal frequency

**Solution:**

```
11.0592/12 = 921.6 kHz;
   machine cycle is 1/921.6 kHz = 1.085μs
```

# TIME DELAY FOR VARIOUS 8051 CHIPS (cont')

For 8051 system of 11.0592 MHz, find how long it takes to execute each instruction.

```
(a) MOV R3,#55  (b) DEC R3  (c) DJNZ R2 target
(d) LJMP  (e) SJMP  (f) NOP  (g) MUL AB
```

**Solution:**

| | ***Machine cycles*** | ***Time to execute*** |
|---|---|---|
| (a) | 1 | 1x1.085 μs = 1.085 μs |
| (b) | 1 | 1x1.085 μs = 1.085 μs |
| (c) | 2 | 2x1.085 μs = 2.17 μs |
| (d) | 2 | 2x1.085 μs = 2.17 μs |
| (e) | 2 | 2x1.085 μs = 2.17 μs |
| (f) | 1 | 1x1.085 μs = 1.085 μs |
| (g) | 4 | 4x1.085 μs = 4.34 μs |

TIME DELAY FOR VARIOUS 8051 CHIPS

Delay Calculation

Find the size of the delay in following program, if the crystal frequency is 11.0592MHz.

```
       MOV   A,#55H
AGAIN: MOV   P1,A
       ACALL DELAY
       CPL   A
       SJMP  AGAIN
;---time delay-------
DELAY: MOV   R3,#200
HERE:  DJNZ  R3,HERE
       RET
```

A simple way to short jump to itself in order to keep the microcontroller busy

```
HERE: SJMP HERE
```

We can use the following:

```
      SJMP $
```

**Solution:**

| | ***Machine cycle*** |
|---|---|
| `DELAY: MOV R3,#200` | 1 |
| `HERE: DJNZ R3,HERE` | 2 |
| `RET` | 2 |

Therefore, $[(200\text{x}2)+1+2]\text{x}1.085\mu s = 436.255\mu s$.

TIME DELAY FOR VARIOUS 8051 CHIPS

Increasing Delay Using NOP

Find the size of the delay in following program, if the crystal frequency is 11.0592MHz.

```
                                Machine Cycle
DELAY: MOV  R3,#250                   1
HERE:  NOP                            1
       NOP                            1
       NOP                            1
       NOP                            1
       DJNZ R3,HERE                   2
       RET                            2
```

**Solution:**

The time delay inside HERE loop is

[250(1+1+1+1+2)]x1.085μs = 1627.5μs.

Adding the two instructions outside loop we

have 1627.5μs + 3 x 1.085μs = 1630.755μs

TIME DELAY FOR VARIOUS 8051 CHIPS

Large Delay Using Nested Loop

Find the size of the delay in following program, if the crystal frequency is 11.0592MHz.

```
                                     Machine Cycle
DELAY: MOV   R2,#200                       1
AGAIN: MOV   R3,#250                       1
HERE:  NOP                                 1
       NOP                                 1
       DJNZ  R3,HERE                       2
       DJNZ  R2,AGAIN                      2
       RET                                 2
```

Notice in nested loop, as in all other time delay loops, the time is approximate since we have ignored the first and last instructions in the subroutine.

**Solution:**

For HERE loop, we have (4x250)x1.085 μs=1085 μs.
For AGAIN loop repeats HERE loop 200 times, so we have 200x1085 μs=217000 μs. But "MOV R3,#250" and "DJNZ R2,AGAIN" at the start and end of the AGAIN loop add (3x200x1.805)=651 μs.
As a result we have 217000+651=217651 μs.

- Two factors can affect the accuracy of the delay
  - Crystal frequency
    - The duration of the clock period of the machine cycle is a function of this crystal frequency
  - 8051 design
    - The original machine cycle duration was set at 12 clocks
    - Advances in both IC technology and CPU design in recent years have made the 1-clock machine cycle a common feature

Clocks per machine cycle for various 8051 versions

| Chip/Maker | Clocks per Machine Cycle |
|---|---|
| AT89C51 Atmel | 12 |
| P89C54X2 Philips | 6 |
| DS5000 Dallas Semi | 4 |
| DS89C420/30/40/50 Dallas Semi | 1 |

TIME DELAY FOR VARIOUS 8051 CHIPS

Delay Calculation for Other 8051 (cont')

Find the period of the machine cycle (MC) for various versions of 8051, if XTAL=11.0592 MHz.
(a) AT89C51 (b) P89C54X2 (c) DS5000 (d) DS89C4x0

```
Solution:
(a) 11.0592MHz/12 = 921.6kHz;
    MC is 1/921.6kHz = 1.085μs = 1085ns
(b) 11.0592MHz/6 = 1.8432MHz;
    MC is 1/1.8432MHz = 0.5425μs = 542ns
(c) 11.0592MHz/4 = 2.7648MHz ;
    MC is 1/2.7648MHz = 0.36μs = 360ns
(d) 11.0592MHz/1 = 11.0592MHz;
    MC is 1/11.0592MHz = 0.0904μs = 90ns
```

TIME DELAY FOR VARIOUS 8051 CHIPS

Delay Calculation for Other 8051 (cont')

| Instruction | 8051 | DSC89C4x0 |
|---|---|---|
| `MOV R3,#55` | 1 | 2 |
| `DEC R3` | 1 | 1 |
| `DJNZ R2 target` | 2 | 4 |
| `LJMP` | 2 | 3 |
| `SJMP` | 2 | 3 |
| `NOP` | 1 | 1 |
| `MUL AB` | 4 | 9 |

For an AT8051 and DSC89C4x0 system of 11.0592 MHz, find how long it takes to execute each instruction.

```
(a) MOV R3,#55 (b) DEC R3 (c) DJNZ R2 target
(d) LJMP (e) SJMP (f) NOP (g) MUL AB
```

**Solution:**

| | AT8051 | DS89C4x0 |
|---|---|---|
| (a) | 1×1085ns = 1085ns | 2×90ns = 180ns |
| (b) | 1×1085ns = 1085ns | 1×90ns = 90ns |
| (c) | 2×1085ns = 2170ns | 4×90ns = 360ns |
| (d) | 2×1085ns = 2170ns | 3×90ns = 270ns |
| (e) | 2×1085ns = 2170ns | 3×90ns = 270ns |
| (f) | 1×1085ns = 1085ns | 1×90ns = 90ns |
| (g) | 4×1085ns = 4340ns | 9×90ns = 810ns |