

# 8051 Instruction set

- Repeating a sequence of instructions a certain number of times is called a *loop*

- Loop action is performed by

DJNZ reg, Label

- The register is decremented
- If it is not zero, it jumps to the target address referred to by the label
- Prior to the start of loop the register is loaded with the counter for the number of repetitions
- Counter can be R0 – R7 or RAM location

A loop can be repeated a maximum of 255 times, if R2 is FFH

```
;This program adds value 3 to the ACC ten times
MOV  A,#0      ;A=0, clear ACC
MOV  R2,#10    ;load counter R2=10
AGAIN: ADD  A,#03 ;add 03 to ACC
      DJNZ R2,AGAIN ;repeat until R2=0,10 times
MOV  R5,A      ;save A in R5
```

## Looping and Jump

- ❑ If we want to repeat an action more times than 256, we use a loop inside a loop, which is called *nested loop*
  - We use multiple registers to hold the count

Write a program to (a) load the accumulator with the value 55H, and (b) complement the ACC 700 times

```
        MOV    A, #55H    ;A=55H
        MOV    R3, #10    ;R3=10, outer loop count
NEXT:   MOV    R2, #70    ;R2=70, inner loop count
AGAIN:  CPL    A         ;complement A register
        DJNZ  R2, AGAIN  ;repeat it 70 times
        DJNZ  R3, NEXT
```

## Nested Loop

## Jump only if a certain condition is met

**JZ label ;jump if A=0**

```
MOV  A,R0    ;A=R0
JZ   OVER    ;jump if A = 0
MOV  A,R1    ;A=R1
JZ   OVER    ;jump if A = 0
...
```

OVER:

Can be used only for register A,  
not any other register

Determine if R5 contains the value 0. If so, put 55H in it.

```
MOV  A,R5    ;copy R5 to A
JNZ  NEXT    ;jump if A is not zero
MOV  R5,#55H
```

NEXT: ...

# Conditional Jump

**JNC label ;jump if no carry, CY=0**

- If CY = 0, the CPU starts to fetch and execute instruction from the address of the label
- If CY = 1, it will not jump but will execute the next instruction below JNC

Find the sum of the values 79H, F5H, E2H. Put the sum in registers R0 (low byte) and R5 (high byte).

```
MOV A, #0 ;A=0
MOV R5, A ;clear R5
ADD A, #79H ;A=0+79H=79H
; JNC N_1 ;if CY=0, add next number
; INC R5 ;if CY=1, increment R5
N_1: ADD A, #0F5H ;A=79+F5=6E and CY=1
      JNC N_2 ;jump if CY=0
      INC R5 ;if CY=1, increment R5 (R5=1)
N_2: ADD A, #0E2H ;A=6E+E2=50 and CY=1
      JNC OVER ;jump if CY=0
      INC R5 ;if CY=1, increment 5
OVER: MOV R0, A ;now R0=50H, and R5=02
```

**MOV R5, #0**

## Conditional Jump

## 8051 conditional jump instructions

Instructions	Actions
JZ	Jump if A = 0
JNZ	Jump if A $\neq$ 0
DJNZ	Decrement and Jump if A $\neq$ 0
CJNE A,byte	Jump if A $\neq$ byte
CJNE reg,#data	Jump if byte $\neq$ #data
JC	Jump if CY = 1
JNC	Jump if CY = 0
JB	Jump if bit = 1
JNB	Jump if bit = 0
JBC	Jump if bit = 1 and clear bit

- ▣ All conditional jumps are short jumps
  - The address of the target must within -128 to +127 bytes of the contents of PC

## Conditional Jump Instruction Set

- The unconditional jump is a jump in which control is transferred unconditionally to the target location

### **LJMP** (long jump)

- 3-byte instruction
  - First byte is the opcode
  - Second and third bytes represent the 16-bit target address
    - Any memory location from 0000 to FFFFH

### **SJMP** (short jump)

- 2-byte instruction
  - First byte is the opcode
  - Second byte is the relative target address
    - 00 to FFH (forward +127 and backward -128 bytes from the current PC)

# Unconditional Jump

- ❑ To calculate the target address of a short jump (`SJMP`, `JNC`, `JZ`, `DJNZ`, etc.)
  - The second byte is added to the PC of the instruction immediately below the jump
- ❑ If the target address is more than -128 to +127 bytes from the address below the short jump instruction
  - The assembler will generate an error stating the jump is out of range

## Calculating Short Jump Address

<i>Line</i>	<i>PC</i>	<i>Opcode</i>	<i>Mnemonic</i>	<i>Operand</i>
01	0000		ORG	0000
02	0000	7800	MOV	R0, #0
03	0002	7455	MOV	A, #55H
04	0004	6003	JZ	NEXT
05	0006	08	INC	R0
06	0007	04	AGAIN:	INC A
07	0008	04		INC A
08	0009	2477	NEXT:	ADD A, #77H
09	000B	5005	JNC	OVER
10	000D	E4	CLR	A
11	000E	F8	MOV	R0, A
12	000F	F9	MOV	R1, A
13	0010	FA	MOV	R2, A
14	0011	FB	MOV	R3, A
15	0012	2B	OVER:	ADD A, R3
16	0013	50F2	JNC	AGAIN
17	0015	80FE	HERE:	SJMP HERE
18	0017			END

## Example to Calculate Address

- ❑ 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 Instruction

- ❑ 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

# LCALL

- ❑ 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

# ACALL

```

    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

```

## Comparison of CALL Instruction

- ❑ 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:**

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

$$\text{machine cycle is } 1/921.6 \text{ kHz} = 1.085 \mu\text{s}$$

## 8051 machine Cycle

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:**

	<i>Machine cycles</i>	<i>Time to execute</i>
(a)	1	$1 \times 1.085 \mu s = 1.085 \mu s$
(b)	1	$1 \times 1.085 \mu s = 1.085 \mu s$
(c)	2	$2 \times 1.085 \mu s = 2.17 \mu s$
(d)	2	$2 \times 1.085 \mu s = 2.17 \mu s$
(e)	2	$2 \times 1.085 \mu s = 2.17 \mu s$
(f)	1	$1 \times 1.085 \mu s = 1.085 \mu s$
(g)	4	$4 \times 1.085 \mu s = 4.34 \mu s$

**Time Delays for Various Instructions**

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:**

	<i>Machine cycle</i>
DELAY: MOV R3, #200	1
HERE: DJNZ R3, HERE	2
RET	2

Therefore,  $[(200 \times 2) + 1 + 2] \times 1.085 \mu s = 436.255 \mu s$ .

# Delay Calculation

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

	<i>Machine Cycle</i>
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

$$[250(1+1+1+1+2)] \times 1.085 \mu s = 1627.5 \mu s.$$

Adding the two instructions outside loop we

$$\text{have } 1627.5 \mu s + 3 \times 1.085 \mu s = 1630.755 \mu s$$

# Increasing Delay using NOP

- ❑ 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

## Other 8051 - Delay times

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

MC is  $1/921.6\text{kHz} = 1.085\ \mu\text{s} = 1085\text{ns}$

(b)  $11.0592\text{MHz}/6 = 1.8432\text{MHz};$

MC is  $1/1.8432\text{MHz} = 0.5425\ \mu\text{s} = 542\text{ns}$

(c)  $11.0592\text{MHz}/4 = 2.7648\text{MHz};$

MC is  $1/2.7648\text{MHz} = 0.36\ \mu\text{s} = 360\text{ns}$

(d)  $11.0592\text{MHz}/1 = 11.0592\text{MHz};$

MC is  $1/11.0592\text{MHz} = 0.0904\ \mu\text{s} = 90\text{ns}$

**Other 8051 - Time Delays**

**NEXT – ARITHMETIC AND LOGIC  
INSTRUCTIONS AND PROGRAMS**