

Microprocessor

Ch.3 Jump, Loop, and Call

OUTLINE

- **Loop and Jump instructions**
- **Call instructions**
- **Time delay**

LOOP: DJNZ

- DJNZ (decrease jump not zero)**

- DJNZ *reg, label*

- Decrement the reg by 1 ('D')
- If reg is **not zero** ('NZ), jump ('J') to the line specified by label; otherwise exit the loop.

- Example (Demo loop)

MOV A, #0

MOV R2, #4



AGAIN: ADD A, #03

DJNZ R2, AGAIN ; reg: R2, lable: AGAIN

MOV R5, A

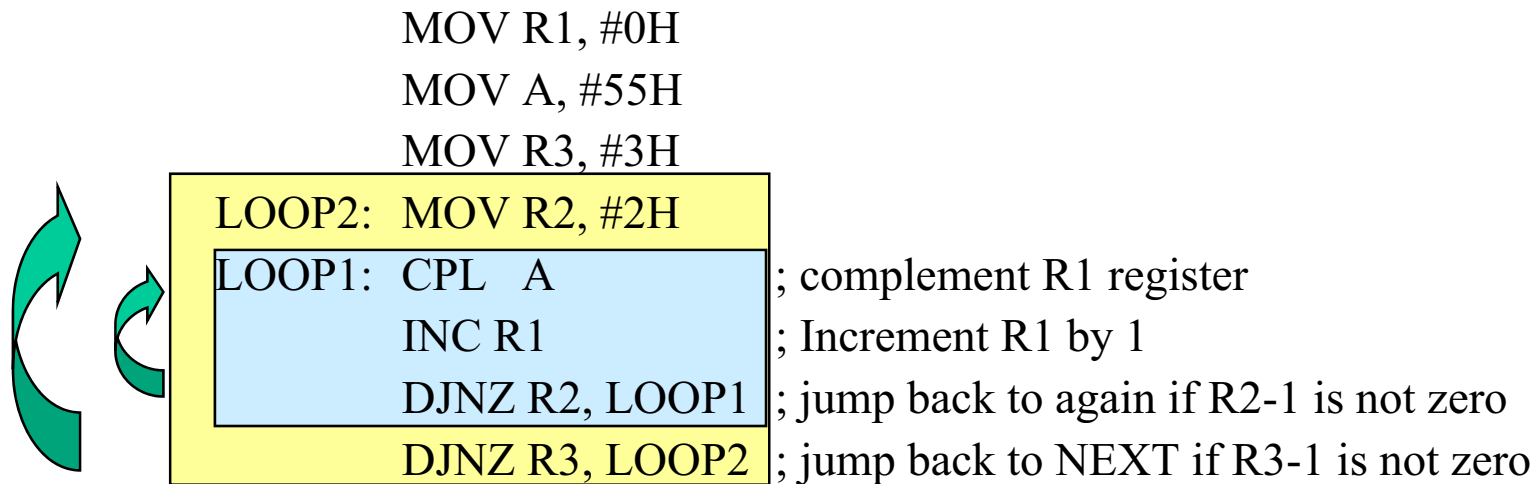
Loop		1	2	3	4
R2					
A					

- The maximum value that can be held in R2 is ; thus the loop can be repeated for a maximum of times.

LOOP: DJNZ

- **Loop inside a loop**

- More loops can be achieved by embedding one loop inside another loop
- Example



- Flow chart

- There are totally $2 \times 3 = 6$ loops

LOOP: CONDITIONAL JUMP INSTRUCTIONS

- **JZ (jump if A = 0)**

- *JZ label*
- Example:

```
MOV A, #0FEH
ADD A, #1H
JZ OVER
ADD A, #1H
JZ OVER
ADD A, #1H
JZ OVER
```

```
OVER:    MOV R0, #0H
```

- Note: JZ can only be used to test register A

- **JNZ (jump if A $\neq$ 0)**

- *JNZ label*
- Example: write a program to determine if R5 contains 0. If so, put 55H in it; otherwise do nothing (NOP)

```
MOV A, R5
JNZ NEXT
MOV R5, #55H
```

```
NEXT:    NOP      ; no operation
```

LOOP: CONDITIONAL JUMP INSTRUCTIONS

- **JNC (jump if no carry)**

- JNC *label*
- Jump to label if no carry (CY = 0)
- Example: find the sum of 79H, F5H, and E2H. Put the sum in registers R0 (low byte) and R5 (high byte)

```

MOV A, #0
MOV R0, A
MOV R5, A
MOV A, #79H      ; A = A + 79H = 79H
ADD A, #0F5H     ; A = 79H + F5H
JNC N2           ; if CY == 0, jump to the next summation
INC R5           ; if CY == 1, increment R5 to record the carry
N2: ADD A, #0E2H  ; A = A + E2H
JNC N3           ; if CY == 0, jump to the end
INC R5           ; if CY == 1, increment R5 to record the carry
N3: MOV R0, A

NOP

```

LOOP: SHORT JUMP

- All conditional jumps are short jump

- Short jump: the address of the target must be within -128 to +127 bytes of the current PC (program counter)

- E.g.

Addr	Opcode	
0000	7400	MOV A, #0
0002	6003	JZ TAGT
0004	FA	MOV R2, A
0005	740A	MOV A, #10
0007	7A00	MOV R2, #0
TAGT:		
;Target (0007) – PC at JZ (0004) = 7 – 4 = 3		

- Why -128 – 127?

- Opcode of JZ: 01100000 xxxxxxxx (offset)
- Offset = address of target – PC at JZ (PC+2, in the example, it is 4)
- The offset is limited to 8 bits (-128 – 127)
- The length of short jump instructions is 2 bytes
- If we want to jump further than -128 or 127, we need to use more bits to represent the jump offset.

LOOP: SHORT JUMP

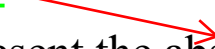
- **Example**

- Find the offset of the forward jump instructions

<i>Line</i>	<i>Addr</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	60 XX	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	50 XX	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	

LOOP: UNCONDITIONAL JUMP

- **LJMP (long jump)**

- LJMP *label*
- Jump to anywhere in the program
- Opcode (3 bytes)
 - 00000010 A15-A8 A7-A0 
 - The 2nd and 3rd bytes represent the absolute address in ROM
 - Review: PC has 16 bits → ROM address range is 0000 - FFFFH → 16 bits are enough to label any address in ROM
- Example

```
                                ORG 0H
                                LJMP FARAWAY      ; opcode 02F000H
                                ORG 0F000H
FARAWAY:                       MOV A, 55H
```

LOOP: UNCONDITIONAL JUMP

- **SJMP (short jump)**

- SJMP *label*
- Jump to an address within -128 – 127 of current PC
- Opcode (2 byte)
 - 10000000 xxxxxxxx (offset) 
- The calculation of offset is the same as conditional jumps (JZ, JNZ, JNC, ...)
- Example: find the offset of the SJMP instructions (xx and yy in the comments)

```

                                ORG 0H
                                SJMP TAGT1          ; opcode 80xxH
                                MOV A, #0            ; opcode 7400H
                                ORG 10
TAGT1:                          SJMP TAGT2          ; opcode 80yyH
                                MOV A, #0
                                ORG 35
TAGT2:                          MOV A, #55H

```

- What will happen if the target is out of the range of [-128, 127] of current PC?

OUTLINE

- Loop and Jump instructions
- **Call instructions**
- Time delay

CALL INSTRUCTIONS

- **Subroutine**
 - A section of code that can perform a specific task (e.g. introduce a certain amount of delay)
 - If a task needs to be performed frequently, it's better to structure the corresponding codes as a subroutine
 - Save memory space 
 - Better program structure
 - Subroutines are invoked by call instructions
- **There are two call instructions in 8051**
 - LCALL (3 byte instruction)
 - 16 bits (2 bytes) are used to represent target address
 - Long call, the subroutine can be placed anywhere in the ROM
 - ACALL (2 byte instruction)
 - Absolute call
 - Only 11 bits are used to represent target address
 - The target address must be within 2K bytes of ACALL

CALL: LCALL

- **LCALL**

- Long call, 3-byte instruction
- Opcode: 00010010 A15-A8 A7-A0 ; the last two bytes are used to represent target address
- Can be used to call subroutines located anywhere within the 64KB of the ROM.
- Example

```

                                ORG 0
BACK:                          MOV A, #55H
                                MOV P1, A           ; send 55H to port 1
                                LCALL DELAY          ; call the subroutine delay
                                MOV A, #0AAH
                                MOV P1, A           ; send AAH to port 1
                                LCALL DELAY
                                SJMP BACK

```

;-----begin of subroutine-----

```

                                ORG 300H
DELAY:                          MOV R5, #0FFH
AGAIN:                          DJNZ R5, AGAIN
                                RET                  ; return to caller

```

; -----end of subroutine-----

END

CALL: CALL AND STACK

• Call instructions and stack

- After 'LCALL' is executed, the PC is changed to the starting address of the subroutine
 - E.g. after LCALL, PC points to address 0300H
- After the subroutine is done ('RET' is executed), the PC goes back to the instruction that follows 'LCALL'
 - E.g. after RET, PC points back to address 0007H ('MOV A, #0AAH')
- How does the CPU know where the PC should point to after the subroutine? (DEMO)
 - **Before** loading the PC with the address of the subroutine (0300H), the CPU **automatically push** the address of the next instruction into **stack**.
 - **After** RET is executed, the CPU **automatically pop** the address back to PC.

```

001 0000          ORG 0
002 0000 7455 BACK: MOV A,#55H ;load A with 55H
003 0002 F590      MOV P1,A    ;send 55H to port 1
004 0004 120300    LCALL DELAY ;time delay
005 0007 74AA      MOV A,#0AAH ;load A with AAH
006 0009 F590      MOV P1,A    ;send AAH to port 1
007 000B 120300    LCALL DELAY
008 000E 80F0      SJMP BACK   ;keep doing this
009 0010
010 0010 ;-----this is the delay subroutine
011 0300          ORG 300H
012 0300 DELAY:
013 0300 7DFF      MOV R5,#0FFH ;R5=255
014 0302 DDFE AGAIN: DJNZ R5,AGAIN ;stay here
015 0304 22        RET          ;return to caller
016 0305          END          ;end of asm file

```

after LCALL at
04

0A

09 00

08 07

Will be
00@09,0E@08
later
after LCALL at
0B

CALL: CALL AND STACK

- **Call instructions and stack (Cont'd)**
 - Each address is 16 bits (recall: PC is a 16-bit register)
 - Each PUSH can put in 8 bit → two PUSH instructions are used
 - Similarly, two pop instructions are used to restore the address to PC.
 - If you use stack in a subroutine, you **MUST** use **EQUAL** number of **PUSH** and **POP**
 - Unequal number of PUSH and POP will result in a wrong value being restored (DEMO LCALL)
 - When you exit a subroutine, the SP should always point to the return address of the subroutine

2F
00
07

CALL: CALL AND STACK

- **Example**

- Analyze the contents of the stack and PC

addr	Opcode		
0000	7455	BACK:	MOV A, #55H
0002	F590		MOV P1, A
0004	7C99		MOV R4, #99H
0006	7D67		MOV R5, #67H
0008	120300		LCALL TEST
000B	74AA		MOV A, #0AAH
000D	80F1		SJMP BACK
			ORG 300H
0300	C004	TEST:	PUSH 4
0302	C005		PUSH 5
0304	D001		POP 1
0306	D002		POP2
0308	22		RET

CALL: ACALL

- **ACALL**

- Absolute call, 2-byte instruction
- 11-bits are used to represent address offset
 - The target address must be within 2K bytes of the address of ACALL
- The ONLY difference between ACALL and LCALL is the limit on target address
 - LCALL: 16 bits used to represent address → target can be anywhere within 64K bytes
 - ACALL: 11 bits used to represent address offset → target needs to be within 2K bytes of the address of ACALL
- Using ACALL will save 1 byte of memory space.

OUTLINE

- Loop and Jump instructions
- Call instructions
- **Time delay**

DELAY: CLOCK V.S. MACHINE CYCLE

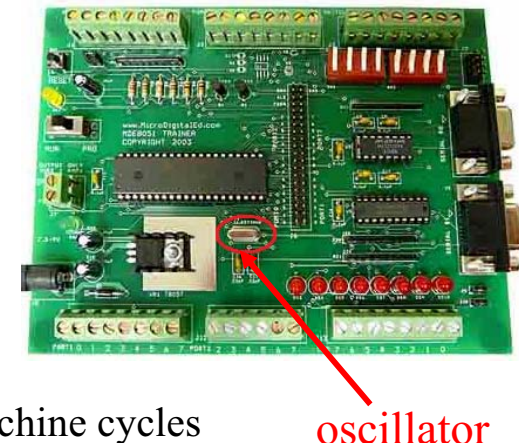
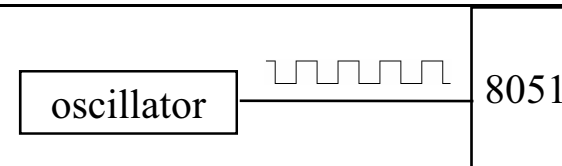
- **Terminology**

- Clock

- A crystal oscillator is connected to 8051 to provide clock source for 8051.
 - Typical clock frequency (f): 11.0592 MHz, 16 MHz, 20 MHz.
 - Oscillator period (T):

- Machine cycle

- A basic operation performed by CPU to execute an instruction.
 - Original 8051
 - 1 machine cycle = 12 oscillator periods
 - DS89C450
 - 1 machine cycle = 1 oscillator period
 - Different instructions require different number of machine cycles
 - E.g. original 8051
 - 1 machine cycle: ADD, MOV R3, A
 - 2 machine cycles: MOV 08, A
 - 4 machine cycles: MUL, DIV
 - Machine cycles can be found at Table A-1 in Appendix A (p.554).
 - It takes different amount of time to execute different instructions.



DELAY

- **Example**

- For an 8051 system with 1 machine cycle = 12 oscillator periods. If the clock frequency is 11.0592 MHz,
 - (1) What is the duration of 1 machine cycle?
 - (2) find how long it takes to execute each of the following instructions

- (a) MOV R3, #data
- (b) MOV P3, R1
- (c) NOP
- (d) DJNZ R2, AGAIN

Instruction	Machine Cycles
MOV Rn, A	1
MOV direct, Rn	2
NOP	1
DJNZ Rn, target	2

DELAY: LOOP

- **Example:**

- 1. For an 8051 system with 1 machine cycle = 12 oscillator periods. If the clock frequency is 11.0592 MHz. Find the delay incurred by the subroutine.

```
                ORG 300H
DELAY:          MOV R3, #200                ; 1 machine cycle
HERE:          DJNZ R3, HERE                ; 2 machine cycle
                RET                        ; 2 machine cycle
```

DELAY: DS89C450

- **DS89C450**

- 1 machine cycle = 1 oscillator clock period
- The machine cycles for all instructions can be found in the user guide of DS89C4x0

Instruction	Machine cycles	
	8051	DS89C4x0
MOV R3,#value	1	2
DEC R _x	1	1
DJNZ	2	4
LJMP	2	3
SJMP	2	3
NOP	1	1
MUL AB	4	9

- Example:
 - A 89C450 is connected to an oscillator with frequency 11.0592MHz. Find how long it takes to execute the following instruction
 - (a) MOV R3, #55
 - (b) DJNZ R2, target

DELAY: EMBEDDED LOOPS

- **Example**

- A DS89C450 is connected to a 11.0592 MHz XTAL. Find the time delay in the following subroutine

DELAY:		; machine cycles
	MOV R2, #200	; 2
AGAIN:	MOV R3, #250	; 2
HERE:	NOP	; 1
	NOP	; 1
	DJNZ R3, HERE	; 4
	DJNZ R2, AGAIN	; 4
	RET	; 3

DELAY:

- **Example**

- Write a program to toggle all the bits of P1 every 200 ms (55H → AAH → ...) with DS89C450 and 11.0592 MHz XTAL.

```
                MOV R1, #9
A1:             MOV R2, #242
A2:             MOV R3, #355
A3:             DJNZ R3, A3
                DJNZ R2, A2
                DJNZ R1, A1
```