

# Microcontroller 8051 Programming

---

Virendra Singh

Computer Architecture and Dependable Systems Lab

Department of Electrical Engineering  
Indian Institute of Technology Bombay

<http://www.ee.iitb.ac.in/~viren/>

E-mail: [viren@ee.iitb.ac.in](mailto:viren@ee.iitb.ac.in)

*EE-309: Microprocessors*



Lecture 5 (28 July 2015)

**CADSL**

# INSTRUCTION SET OF 8051



# Addressing Modes

- Eight modes of addressing are available
- The different addressing modes determine how the operand byte is selected

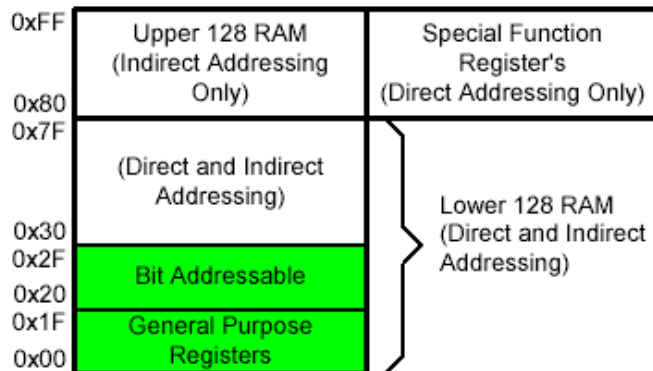
Addressing Modes	Instruction
Register	MOV A, B
Direct	MOV 30H, A
Indirect	ADD A, @R0
Immediate Constant	ADD A, #80H
Relative*	SJMP AHEAD
Absolute*	AJMP BACK
Long*	LJMP FAR_AHEAD
Indexed	MOVC A, @A+PC

\* Related to program branching instructions



# Data Transfer Instructions

- Data transfer instructions can be used to transfer data between an internal RAM location and an SFR location without going through the accumulator
- It is also possible to transfer data between the internal and external RAM by using indirect addressing
- The upper 128 bytes of data RAM are accessed only by indirect addressing and the SFRs are accessed only by direct addressing



Mnemonic	Description
MOV @Ri, direct	[@Ri] = [direct]
MOV @Ri, #data	[@Ri] = immediate data
MOV DPTR, #data 16	[DPTR] = immediate data
MOVC A, @A+DPTR	A = Code byte from [@A+DPTR]
MOVC A, @A+PC	A = Code byte from [@A+PC]
MOVX A, @Ri	A = Data byte from external ram [@Ri]
MOVX A, @DPTR	A = Data byte from external ram [@DPTR]
MOVX @Ri, A	External[@Ri] = A
MOVX @DPTR, A	External[@DPTR] = A
PUSH direct	Push into stack
POP direct	Pop from stack
XCH A, Rn	A = [Rn], [Rn] = A
XCH A, direct	A = [direct], [direct] = A
XCH A, @Ri	A = [@Rn], [@Rn] = A
XCHD A,@Ri	Exchange low order digits



# Arithmetic Operations

- The appropriate status bits in the PSW are set when specific conditions are met, which allows the user software to manage the different data formats

Mnemonic	Description
ADD A, Rn	$A = A + [Rn]$
ADD A, direct	$A = A + [\text{direct memory}]$
ADD A,@Ri	$A = A + [\text{memory pointed to by Ri}]$
ADD A,#data	$A = A + \text{immediate data}$
ADDC A,Rn	$A = A + [Rn] + CY$
ADDC A, direct	$A = A + [\text{direct memory}] + CY$
ADDC A,@Ri	$A = A + [\text{memory pointed to by Ri}] + CY$
ADDC A,#data	$A = A + \text{immediate data} + CY$
SUBB A,Rn	$A = A - [Rn] - CY$
SUBB A, direct	$A = A - [\text{direct memory}] - CY$
SUBB A,@Ri	$A = A - [@Ri] - CY$
SUBB A,#data	$A = A - \text{immediate data} - CY$
INC A	$A = A + 1$
INC Rn	$[Rn] = [Rn] + 1$
INC direct	$[\text{direct}] = [\text{direct}] + 1$
INC @Ri	$[@Ri] = [@Ri] + 1$
DEC A	$A = A - 1$
DEC Rn	$[Rn] = [Rn] - 1$
DEC direct	$[\text{direct}] = [\text{direct}] - 1$
DEC @Ri	$[@Ri] = [@Ri] - 1$
MUL AB	Multiply A & B
DIV AB	Divide A by B
DA A	Decimal adjust A

- $[@Ri]$  implies contents of memory location pointed to by R0 or R1
- Rn refers to registers R0-R7 of the currently selected register bank



# Logical Operations

- Logical instructions perform Boolean operations (AND, OR, XOR, and NOT) on data bytes on a *bit-by-bit* basis

- Examples:

**ANL A, #02H ;Mask bit 1**  
**ORL TCON, A ;TCON=TCON-**  
**OR-A**

Mnemonic	Description
ANL A, Rn	$A = A \& [Rn]$
ANL A, direct	$A = A \& [\text{direct memory}]$
ANL A, @Ri	$A = A \& [\text{memory pointed to by Ri}]$
ANL A, #data	$A = A \& \text{immediate data}$
ANL direct, A	$[\text{direct}] = [\text{direct}] \& A$
ANL direct, #data	$[\text{direct}] = [\text{direct}] \& \text{immediate data}$
ORL A, Rn	$A = A \text{ OR } [Rn]$
ORL A, direct	$A = A \text{ OR } [\text{direct}]$
ORL A, @Ri	$A = A \text{ OR } [@Ri]$
ORL A, #data	$A = A \text{ OR immediate data}$
ORL direct, A	$[\text{direct}] = [\text{direct}] \text{ OR } A$
ORL direct, #data	$[\text{direct}] = [\text{direct}] \text{ OR immediate data}$
XRL A, Rn	$A = A \text{ XOR } [Rn]$
XRL A, direct	$A = A \text{ XOR } [\text{direct memory}]$
XRL A, @Ri	$A = A \text{ XOR } [@Ri]$
XRL A, #data	$A = A \text{ XOR immediate data}$
XRL direct, A	$[\text{direct}] = [\text{direct}] \text{ XOR } A$
XRL direct, #data	$[\text{direct}] = [\text{direct}] \text{ XOR immediate data}$
CLR A	Clear A
CPL A	Complement A
RL A	Rotate A left
RLC A	Rotate A left (through C)
RR A	Rotate A right
RRC A	Rotate A right (through C)
SWAP A	Swap nibbles



# Boolean Variable Instructions

8051 can perform single bit operations

- The operations include *set*, *clear*, *and*, *or* and *complement* instructions
- Also included are bit-level moves or conditional jump instructions
- All bit accesses use direct addressing
- Examples:

**SETB TR0 ;Start Timer0.**

**POLL: JNB TR0, POLL ;Wait till timer overflows.**

Mnemonic	Description
CLR C	Clear C
CLR bit	Clear direct bit
SETB C	Set C
SETB bit	Set direct bit
CPL C	Complement c
CPL bit	Complement direct bit
ANL C,bit	AND bit with C
ANL C,/bit	AND NOT bit with C
ORL C,bit	OR bit with C
ORL C,/bit	OR NOT bit with C
MOV C,bit	MOV bit to C
MOV bit,C	MOV C to bit
JC rel	Jump if C set
JNC rel	Jump if C not set
JB bit,rel	Jump if specified bit set
JNB bit,rel	Jump if specified bit not set
JBC bit,rel	if specified bit set then clear it and jump



# Program Branching Instructions

- Program branching instructions are used to control the flow of program execution
- Some instructions provide decision making capabilities before transferring control to other parts of the program (conditional branches).

Mnemonic	Description
ACALL addr11	Absolute subroutine call
LCALL addr16	Long subroutine call
RET	Return from subroutine
RETI	Return from interrupt
AJMP addr11	Absolute jump
LJMP addr16	Long jump
SJMP rel	Short jump
JMP @A+DPTR	Jump indirect
JZ rel	Jump if A=0
JNZ rel	Jump if A NOT=0
CJNE A,direct,rel	Compare and Jump if Not Equal
CJNE A,#data,rel	
CJNE Rn,#data,rel	
CJNE @Ri,#data,rel	
DJNZ Rn,rel	Decrement and Jump if Not Zero
DJNZ direct,rel	
NOP	No Operation





# Example 1: Addition of 16 bit Data

---

Add 4A86H and 3895H

CLR C

MOV A, #86H

ADD A, #95H

MOV R5, A

MOV A, #4AH

ADDC A, #38H

MOV R6, A

STOP: SJMP STOP



# Example 1: ASCII to BCD

---

Convert ASCII character to BCD

```
MOV A, #'8'           ; (A) = 38 H
MOV R1, #'6'          ; (R1) = 36 H
ANL A, #0FH
ANL R1, #0FH
SWAP A
ORL A, R1
STOP: SJMP STOP
```



# Example 2: Conversion to 3 digit BCD

---

Conversion of one byte binary data to 3 digit BCD

```
MOV A, #'C'
```

```
MOV B, #100
```

```
DIV AB
```

```
MOV R1, A
```

```
XCH A, B
```

```
MOV B, #10
```

```
DIV AB
```

```
SWAP A
```

```
ADD A, B
```

```
MOV R1, A
```

```
STOP: SJMP STOP
```



# Example 3: Multibyte Addition

---

Addition of Multiple numbers stored at 50H

MOV R1, #50H

MOV R2, #5

CLR A

MOV R5, A

AGAIN: ADD A, @R1

JNC NEXT

INC R5

NEXT INC R1

DJNZ R2, AGAIN

HERE SJMP HERE



# Example 4: Checksum computation

---

Checksum computation of 50 numbers

MOV R1, @30H

MOV R2, #50

CLR A

**AGAIN:** ADD A, @R1

INC R1

DJNZ R2, **AGAIN**

MOV R3, A

**STOP:** SJMP **STOP**



# Example 5: Sequential Search

---

## Sequential Search

```
NUM      EQU 49H

        MOV R2, #50
        MOV R0, #40H

NEXT:    MOV A, @R0
        CJNE A, #NUM, FORWARD
        MOV R3, 0
        SJMP STOP

FORWARD: INC R0
        DJNZ R2, NEXT
        CLR A

STOP:    SJMP STOP
```



# Example 6: Bubble Sort

---

## Bubble Sort

	MOV R2, #50		MOV A, R6
LOOP2:	MOV R0, #40H		XCH A, @R0
	MOV R3, 2		DEC R0
LOOP1:	DEC R3		MOV @R0, A
	CLR C		INC R0
	MOV A, @R0	NEXT:	CJNE R3, #0, LOOP1
	INC R0		DEC R2
	MOV R6, A		CJNE R2, #0, LOOP2
	SUBB A, @R0	STOP:	SJMP STOP
	JC NEXT		



# Thank You

