8051 Addressing Mode and Instruction Set

Arithmetic Operations

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

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

SILICON LABS

ADD A,<source-byte> ADDC A,<source-byte>

- ADD adds the data byte specified by the source operand to the accumulator, leaving the result in the accumulator
- ADDC adds the data byte specified by the source operand, the carry flag and the accumulator contents, leaving the result in the accumulator
- Operation of both the instructions, ADD and ADDC, can affect the carry flag (CY), auxiliary carry flag (AC) and the overflow flag (OV)
  - CY=1 If there is a carryout from bit 7; cleared otherwise
  - AC =1 If there is a carryout from the lower 4-bit of A i.e. from bit 3; cleared otherwise
  - OV=1 If the signed result cannot be expressed within the number of bits in the destination operand; cleared otherwise

SUBB A,<source-byte>

- SUBB subtracts the specified data byte and the carry flag together from the accumulator, leaving the result in the accumulator

CY=1	If a borrow is needed for bit 7; cleared otherwise
AC =1	If a borrow is needed for bit 3, cleared otherwise
OV=1	If a borrow is needed into bit 6, but not into bit 7, or into bit 7, but not into bit 6.

- *Example*:

  The accumulator holds 0C1H (11000001B), Register1 holds 40H (01000000B) and the CY=1.The instruction,

```
SUBB    A, R1
```

  gives the value 70H (01110000B) in the accumulator, with the CY=0 and AC=0 but OV=1

INC <byte>

- Increments the data variable by 1. The instruction is used in register, direct or register direct addressing modes
- *Example*:

```
INC     6FH
```

  If the internal RAM location 6FH contains 30H, then the instruction increments this value, leaving 31H in location 6FH

- *Example*:

```
MOV     R1, #5E
INC     R1
INC     @R1
```

- If R1=5E (01011110) and internal RAM location 5FH contains 20H, the instructions will result in R1=5FH and internal RAM location 5FH to increment by one to 21H

DEC <byte>

- The data variable is decremented by 1
- The instruction is used in accumulator, register, direct or register direct addressing modes
- A data of value 00H underflows to FFH after the operation
- No flags are affected

INC DPTR

- Increments the 16-bit data pointer by 1
- DPTR is the only 16-bit register that can be incremented
- The instruction adds one to the contents of DPTR directly

MUL AB

- Multiplies A & B and the 16-bit result stored in [B15-8], [A7-0]
- Multiplies the unsigned 8-bit integers in the accumulator and the B register
- The **Low** order byte of the 16-bit product will go to the accumulator and the **High** order byte will go to the B register
- If the product is greater than 255 (FFH), the overflow flag is set; otherwise it is cleared. The carry flag is always cleared.
- If ACC=85 (55H) and B=23 (17H), the instruction gives the product 1955 (07A3H), so B is now 07H and the accumulator is A3H. The overflow flag is set and the carry flag is cleared.

SILICON LABS

DIV AB

- Divides A by B
- The integer part of the quotient is stored in A and the remainder goes to the B register
- If ACC=90 (5AH) and B=05(05H), the instruction leaves 18 (12H) in ACC and the value 00 (00H) in B, since 90/5 = 18 (quotient) and 00 (remainder)
- Carry and OV are both cleared
- *If B contains 00H before the division operation, then the values stored in ACC and B are undefined and an overflow flag is set. The carry flag is cleared.*

DA A

- This is a decimal adjust instruction
- It adjusts the 8-bit value in ACC resulting from operations like ADD or ADDC and produces two 4-bit digits (in packed Binary Coded Decimal (BCD) format)
- Effectively, this instruction performs the decimal conversion by adding 00H, 06H, 60H or 66H to the accumulator, depending on the initial value of ACC and PSW
- If ACC bits A3-0 are greater than 9 (xxxx1010-xxxx1111), or if AC=1, then a value 6 is added to the accumulator to produce a correct BCD digit in the lower order nibble
- If CY=1, because the high order bits A7-4 is now exceeding 9 (1010xxxx-1111xxxx), then these high order bits will be increased by 6 to produce a correct proper BCD in the high order nibble but not clear the carry

Logical Operations

Mnemonic	Description
ANL A, Rn	A = A & [Rn]
ANL A, direct	A = A & [direct memory]
ANL A,@Ri	A = A & [memory pointed to by Ri]
ANL A,#data	A= A & immediate data
ANL direct,A	[direct] = [direct] & A
ANL direct,#data	[direct] = [direct] & immediate data
ORL A, Rn	A = A OR [Rn]
ORL A, direct	A = A OR [direct]
ORL A,@Ri	A = A OR [@RI]
ORL A,#data	A = A OR immediate data
ORL direct,A	[direct] = [direct] OR A
ORL direct,#data	[direct] = [direct] OR immediate data
XRL A, Rn	A = A XOR [Rn]
XRL A, direct	A = A XOR [direct memory]
XRL A,@Ri	A = A XOR [@Ri]
XRL A,#data	A = A XOR immediate data
XRL direct,A	[direct] = [direct] XOR A
XRL direct,#data	[direct] = [direct] 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

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

ANL <dest-byte>,<source-byte>

- This instruction performs the logical AND operation on the source and destination operands and stores the result in the destination variable
- No flags are affected
- *Example*:

```
ANL     A,R2
```

  If ACC=D3H (11010011) and R2=75H (01110101), the result of the instruction is ACC=51H (01010001)

- The following instruction is also useful when there is a need to mask a byte
- *Example*:

```
ANL     P1,#10111001B
```

ORL <dest-byte>,<source-byte>

- This instruction performs the logical OR operation on the source and destination operands and stores the result in the destination variable
- No flags are affected
- *Example*:

```
ORL     A,R2
```

If ACC=D3H (11010011) and R2=75H (01110101), the result of the instruction is ACC=F7H (11110111)

- *Example*:

```
ORL     P1,#11000010B
```

This instruction sets bits 7, 6, and 1 of output Port 1

XRL <dest-byte>,<source-byte>

- This instruction performs the logical XOR (Exclusive OR) operation on the source and destination operands and stores the result in the destination variable

- No flags are affected

- *Example*:

```
XRL     A,R0
```

  If ACC=C3H (11000011) and R0=AAH (10101010), then the instruction results in ACC=69H (01101001)

- *Example*:

```
XRL     P1,#00110001
```

  This instruction complements bits 5, 4, and 0 of output Port 1

CLR A and CPL A

CLR A

- This instruction clears the accumulator (all bits set to 0)
- No flags are affected
- If ACC=C3H, then the instruction results in ACC=00H

CPL A

- This instruction logically complements each bit of the accumulator (one's complement)
- No flags are affected
- If ACC=C3H (11000011), then the instruction results in ACC=3CH (00111100)

RL A

- The 8 bits in the accumulator are rotated one bit to the left. Bit 7 is rotated into the bit 0 position.
- No flags are affected
- If ACC=C3H (11000011), then the instruction results in ACC=87H (10000111) with the carry unaffected

RLC A

- The instruction rotates the accumulator contents one bit to the left through the carry flag
- Bit 7 of the accumulator will move into carry flag and the original value of the carry flag will move into the Bit 0 position
- No other flags are affected
- If ACC=C3H (11000011), and the carry flag is 1, the instruction results in ACC=87H (10000111) with the carry flag set

RR A

- The 8 bits in the accumulator are rotated one bit to the right. Bit 0 is rotated into the bit 7 position.
- No flags are affected
- If ACC=C3H (11000011), then the instruction results in ACC=E1H (11100001) with the carry unaffected

RRC A

- The instruction rotates the accumulator contents one bit to the right through the carry flag
- The original value of carry flag will move into Bit 7 of the accumulator and Bit 0 rotated into carry flag
- No other flags are affected
- If ACC=C3H (11000011), and the carry flag is 0, the instruction results in ACC=61H (01100001) with the carry flag set

SWAP A

- This instruction interchanges the low order 4-bit nibbles (A3-0) with the high order 4-bit nibbles (A7-4) of the ACC
- The operation can also be thought of as a 4-bit rotate instruction
- No flags are affected
- If ACC=C3H (11000011), then the instruction leaves ACC=3CH (00111100)