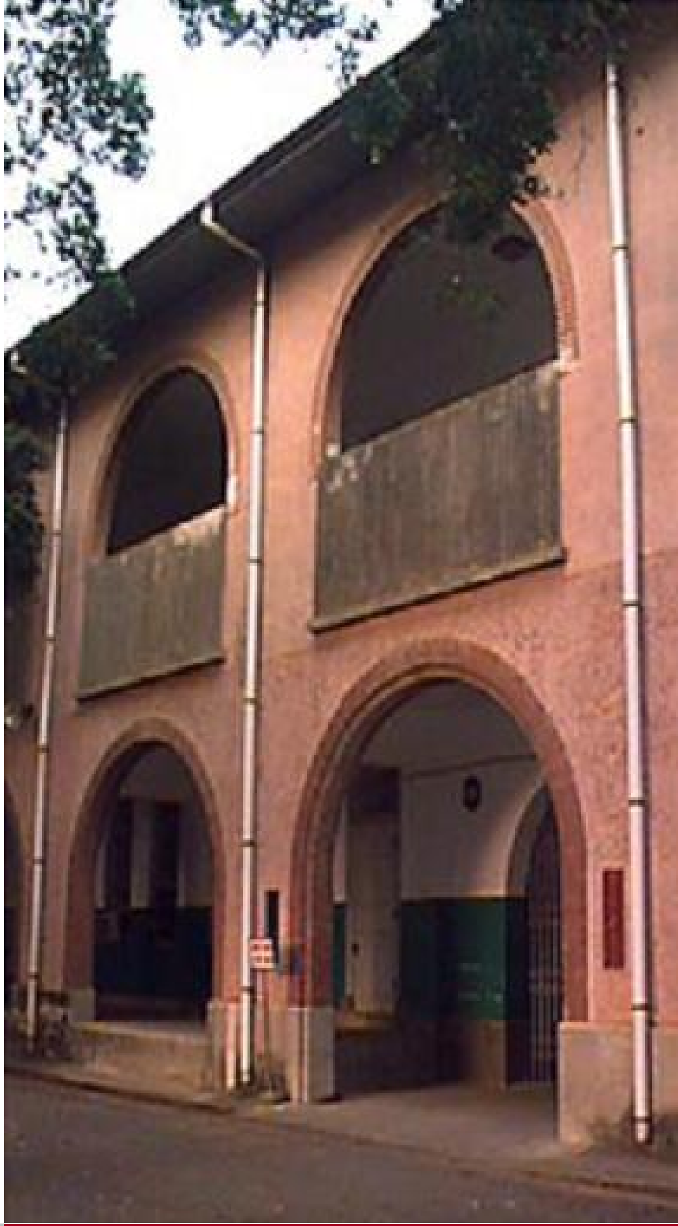
ARITHMETIC & LOGIC INSTRUCTIONS AND PROGRAMS

The 8051 Microcontroller and Embedded Systems: Using Assembly and C
Mazidi, Mazidi and McKinlay



Chung-Ping Young
楊中平

Home Automation, Networking, and Entertainment Lab

Dept. of Computer Science and Information Engineering
National Cheng Kung University, TAIWAN

ARITHMETIC INSTRUCTIONS

Addition of Unsigned Numbers

```
ADD  A,source  ;A = A + source
```

- The instruction ADD is used to add two operands
  - Destination operand is always in register A
  - Source operand can be a register, immediate data, or in memory
  - Memory-to-memory arithmetic operations are never allowed in 8051 Assembly language

Show how the flag register is affected by the following instruction.

```
MOV A,#0F5H ;A=F5 hex
ADD A,#0BH  ;A=F5+0B=00
```

Solution:

```
    F5H        1111 0101
+   0BH     +  0000 1011
   100H        0000 0000
```

CY =1, since there is a carry out from D7
PF =1, because the number of 1s is zero (an even number), PF is set to 1.
AC =1, since there is a carry from D3 to D4

Assume that RAM locations 40 – 44H have the following values. Write a program to find the sum of the values. At the end of the program, register A should contain the low byte and R7 the high byte.

40 = (7D)
41 = (EB)
42 = (C5)
43 = (5B)
44 = (30)

Solution:

```
        MOV R0,#40H   ;load pointer
        MOV R2,#5     ;load counter
        CLR A         ;A=0
        MOV R7,A      ;clear R7
AGAIN:  ADD A,@R0     ;add the byte ptr to by R0
        JNC NEXT      ;if CY=0 don't add carry
        INC R7        ;keep track of carry
NEXT:   INC R0        ;increment pointer
        DJNZ R2,AGAIN ;repeat until R2 is zero
```

ARITHMETIC INSTRUCTIONS

ADDC and Addition of 16-Bit Numbers

- When adding two 16-bit data operands, the propagation of a carry from lower byte to higher byte is concerned

```
       1
      3C E7
+     3B 8D
      -----
      78 74
```

When the first byte is added (E7+8D=74, CY=1).
The carry is propagated to the higher byte, which result in 3C + 3B + 1 =78 (all in hex)

Write a program to add two 16-bit numbers. Place the sum in R7 and R6; R6 should have the lower byte.

Solution:

```
CLR  C           ;make CY=0
MOV  A, #0E7H    ;load the low byte now A=E7H
ADD  A, #8DH     ;add the low byte
MOV  R6, A       ;save the low byte sum in R6
MOV  A, #3CH     ;load the high byte
ADDC A, #3BH     ;add with the carry
MOV  R7, A       ;save the high byte sum
```

- The binary representation of the digits 0 to 9 is called BCD (Binary Coded Decimal)
  - Unpacked BCD
    - In unpacked BCD, the lower 4 bits of the number represent the BCD number, and the rest of the bits are 0
    - Ex. 00001001 and 00000101 are unpacked BCD for 9 and 5
  - Packed BCD
    - In packed BCD, a single byte has two BCD number in it, one in the lower 4 bits, and one in the upper 4 bits
    - Ex. 0101 1001 is packed BCD for 59H

Digit	BCD
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

- Adding two BCD numbers must give a BCD result

```
MOV    A, #17H
ADD    A, #28H
```

Adding these two numbers gives 0011 1111B (3FH), Which is not BCD!

The result above should have been 17 + 28 = 45 (0100 0101). To correct this problem, the programmer must add 6 (0110) to the low digit: 3F + 06 = 45H.

ARITHMETIC INSTRUCTIONS

DA Instruction

```
DA A ;decimal adjust for addition
```

- The DA instruction is provided to correct the aforementioned problem associated with BCD addition
  - The DA instruction will add 6 to the lower nibble or higher nibble if need

Example:

```
MOV A,#47H      ;A=47H first BCD operand
MOV B,#25H      ;B=25H second BCD operand
ADD A,B         ;hex(binary) addition(A=6CH)
DA  A           ;adjust for BCD addition
                (A=72H)
```

6CH

72H

The "DA" instruction works only on A. In other word, while the source can be an operand of any addressing mode, the destination must be in register A in order for DA to work.

DA works only after an ADD, but not after INC

Summary of DA instruction

- After an ADD or ADDC instruction
  1. If the lower nibble (4 bits) is greater than 9, or if AC=1, add 0110 to the lower 4 bits
  2. If the upper nibble is greater than 9, or if CY=1, add 0110 to the upper 4 bits

Example:

	HEX		BCD	
	29		0010 1001	
+	18	+	0001 1000	
	41		0100 0001	AC=1
+	6	+	0110	
	47		0100 0111	

Since AC=1 after the addition, "DA A" will add 6 to the lower nibble.
The final result is in BCD format.

Assume that 5 BCD data items are stored in RAM locations starting at 40H, as shown below. Write a program to find the sum of all the numbers. The result must be in BCD.

40=(71)
41=(11)
42=(65)
43=(59)
44=(37)

Solution:

```
        MOV     R0,#40H    ;Load pointer
        MOV     R2,#5      ;Load counter
        CLR     A          ;A=0
        MOV     R7,A       ;Clear R7
AGAIN:  ADD     A,@R0      ;add the byte pointer
                           ;to by R0
        DA      A          ;adjust for BCD
        JNC     NEXT       ;if CY=0 don't
                           ;accumulate carry
        INC     R7         ;keep track of carries
NEXT:   INC     R0         ;increment pointer
        DJNZ    R2,AGAIN   ;repeat until R2 is 0
```

ARITHMETIC INSTRUCTIONS

Subtraction of Unsigned Numbers

- In many microprocessor there are two different instructions for subtraction: SUB and SUBB (subtract with borrow)
  - In the 8051 we have only SUBB
  - The 8051 uses adder circuitry to perform the subtraction

```
SUBB A,source ;A = A – source – CY
```

- To make SUB out of SUBB, we have to make CY=0 prior to the execution of the instruction
  - Notice that we use the CY flag for the borrow

ARITHMETIC INSTRUCTIONS

Subtraction of Unsigned Numbers (cont')

- SUBB when CY = 0
  1. Take the 2's complement of the subtrahend (source operand)
  2. Add it to the minuend (A)
  3. Invert the carry

```
        CLR    C
        MOV    A,#4C   ;load A with value 4CH
        SUBB   A,#6EH  ;subtract 6E from A
        JNC    NEXT    ;if CY=0 jump to NEXT
        CPL    A  ;if CY=1, take 1's complement
        INC    A  ;and increment to get 2's comp
NEXT:   MOV    R1,A    ;save A in R1
```

Solution:

```
    4C     0100 1100      0100 1100
 -  6E     0110 1110      1001 0010
   -22                   01101 1110
```

① 2's complement

② +

③ Invert carry

CY =1

CY=0, the result is positive;
CY=1, the result is negative and the destination has the 2's complement of the result

ARITHMETIC INSTRUCTIONS

Subtraction of Unsigned Numbers (cont')

- SUBB when CY = 1
  - This instruction is used for multi-byte numbers and will take care of the borrow of the lower operand

A = 62H – 96H – 0 = CCH
CY = 1

```
CLR     C
MOV     A,#62H  ;A=62H
SUBB    A,#96H  ;62H-96H=CCH with CY=1
MOV     R7,A    ;save the result
MOV     A,#27H  ;A=27H
SUBB    A,#12H  ;27H-12H-1=14H
MOV     R6,A    ;save the result
```

A = 27H - 12H - 1 = 14H
CY = 0

Solution:

We have 2762H - 1296H = 14CCH.

ARITHMETIC INSTRUCTIONS

Unsigned Multiplication

- The 8051 supports byte by byte multiplication only
  - The byte are assumed to be unsigned data

```
MUL AB  ;AxB, 16-bit result in B, A
```

```
        MOV     A,#25H      ;load 25H to reg.  A
        MOV     B,#65H      ;load 65H to reg.  B
        MUL     AB          ;25H * 65H = E99 where
                            ;B = OEH and A = 99H
```

Unsigned Multiplication Summary (MUL AB)

Multiplication	Operand1	Operand2	Result
Byte x byte	A	B	B = high byte A = low byte

ARITHMETIC INSTRUCTIONS

Unsigned Division

- The 8051 supports byte over byte division only
  - The byte are assumed to be unsigned data

```
DIV  AB   ;divide A by B, A/B
```

```
        MOV     A,#95   ;load 95 to reg.  A
        MOV     B,#10   ;load 10 to reg.  B
        MUL     AB      ;A = 09(quotient) and
                        ;B = 05(remainder)
```

Unsigned Division Summary (DIV AB)

Division	Numerator	Denominator	Quotient	Remainder
Byte / byte	A	B	A	B

CY is always 0
If B ≠ 0, OV = 0
If B = 0, OV = 1 indicates error

(a) Write a program to get hex data in the range of 00 – FFH from port 1 and convert it to decimal. Save it in R7, R6 and R5.
(b) Assuming that P1 has a value of FDH for data, analyze program.

Solution:

(a)

```
MOV  A,#0FFH
MOV  P1,A        ;make P1 an input port
MOV  A,P1        ;read data from P1
MOV  B,#10       ;B=0A hex
DIV  AB          ;divide by 10
MOV  R7,B        ;save lower digit
MOV  B,#10
DIV  AB          ;divide by 10 once more
MOV  R6,B        ;save the next digit
MOV  R5,A        ;save the last digit
```

(b) To convert a binary (hex) value to decimal, we divide it by 10 repeatedly until the quotient is less than 10. After each division the remainder is saves.

	Q	R
FD/0A =	19	3 (low digit)
19/0A =	2	5 (middle digit)
		2 (high digit)

Therefore, we have FDH=253.