CHƯƠNG 6
Các lệnh số học và các chương trình

6.1 Phép cộng và trừ không dấu.

Các số không dấu được định nghĩa như những dữ liệu mà tất cả mọi bit của chúng đều được dùng để biểu diễn dữ liệu và khó có bit dành cho dấu âm hoặc dương. Điều này có nghĩa là toán hạng có thể nằm giữa 00 và FFH (0 đến 255 hệ thập phân) đối với dữ liệu 8 bit.

6.1.1 Phép cộng các số không dấu.

Trong 8051 để cộng các số với nhau thì thanh ghi tổng (A) phải được dùng đến. Dạng lệnh ADD là:

ADD A, nguồn; A = A + nguồn

Lệnh ADD được dùng để cộng hai toán hạng. Toán hạng đích luôn là thanh ghi A trong khi đó toán hạng nguồn có thể là một thanh ghi dữ liệu trực tiếp hoặc là ở trong bộ nhớ. Hãy nhớ rằng các phép toán số học từ bộ nhớ đến bộ nhớ không bao giờ được phép trong hợp ngữ. Lệnh này có thể thay đổi một trong các bit AF, CF hoặc PF của thanh ghi cờ phụ thuộc vào các toán hạng liên quan. Tác động của lệnh ADD lên cờ tràn sẽ được trình bày ở mục 6.3 vì nó chủ yếu được sử dụng trong các phép toán với số có dấu. Xét ví dụ 6.1 dưới đây:

Ví dụ 6.1:

Hãy biểu diễn xem cá lệnh dưới đây tác động đến thanh ghi cờ như thế nào?

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

Lời giải:

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

Sau phép cộng, thanh ghi A (đích) chứa 00 và các cờ sẽ như sau:

CY = 1 vì có phép nhớ từ D7

PF = 1 vì số các số 1 là 0 (một số chẵn) cờ PF được đặt lên 1.

AC = 1 vì có phép nhớ từ D3 sang D4

6.1.1.1 Phép cộng các byte riêng rẽ.

ở chương 2 đã trình bày một phép cộng 5 byte dữ liệu. Tổng số đã được cất theo chú ý nhỏ hơn FFH là giá trị cực đại một thanh ghi 8 bit có thể được giữ. Để tính tổng số của một số bất kỳ các toán hạng thì cờ nhớ phải được kiểm tra sau mỗi lần cộng một toán hạng. Ví dụ 6.2 dùng R7 để tích luỹ số lần nhớ mỗi khi các toán hạng được cộng vào A.

Ví dụ 6.2:

Giả sử các ngăn nhớ 40 - 44 của RAM có giá trị sau: 40 = (7D); 41 = (EB); 42 = (C5); 43 = (5B) và 44 = (30). Hãy viết một chương trình tính tổng của các giá trị trên. Cuối chương trình giá trị thanh ghi A chứa byte thấp và R7 chứa byte cao (các giá trị trên được cho ở dạng Hex).

Lời giải:

```
        MOV   R0, #40H        ; Nạp con trỏ
        MOV   R2, #5          ; Nạp bộ đệm
        CLR   A               ; Xoá thanh ghi A
        MOV   R7, A           ; Xoá thanh ghi R7
AGAIN:  ADD   A, @R0          ; Cộng byte con trỏ chỉ đến theo R0
        JNC   NEXT            ; Nếu CY = 0 không tích luỹ cờ nhớ
        INC   R7              ; Bám theo số lần nhớ
NEXT:   INC   R0              ; Tăng con trỏ
        DJNZ  R2, AGAIN       ; Lặp lại cho đến khi R0 = 0
```

Phân tích ví dụ 6.2:

Ba lần lặp lại của vòng lặp được chỉ ra dưới đây. Phần dò theo chương trình dành cho người đọc tự thực hiện.

Trong lần lặp lại đầu tiên của vòng lặp thì 7DH được cộng vào A với CY = 0 và R7 = 00 và bộ đếm R2 = 04.

Trong lần lặp lại thứ hai của vòng lặp thì EBH được cộng vào A và kết quả trong A là 68H với CY = 1. Vì cờ nhớ xuất hiện, R7 được tăng lên. Lúc này bộ đếm R2 = 03.

Trong lần lặp lại thứ ba thì C5H được cộng vào A nên A = 2DH và cờ nhớ lại bận. Do vậy R7 lại được tăng lên và bộ đệm R2 = 02.

Ở phần cuối khi vòng lặp kết thúc, tổng số được giữ bởi thanh ghi A và R7, trong đó A giữ byte thấp và R7 chứa byte cao.

6.1.1.2 Phép cộng vó nhớ và phép cộng các số 16 bit.

Khi cộng hai toán hạng dữ liệu 16 bit thì ta cần phải quan tâm đến phép truyền của cờ nhớ từ byte thấp đến byte cao. Lệnh ADDC (cộng có nhớ) được sử dụng trong những trường hợp như vậy. Ví dụ, xét phép cộng hai số sau: 3CE7H + 3B8DH.

```
    3C  E7
+   3B  8D
    78  74
    79
```

Khi byte thứ nhất được cộng (E7 + 8D = 74, CY = 1). Cờ nhớ được truyền lên byte cao tạo ra kết quả 3C + 3B + 1 = 78. Dưới đây là chương trình thực hiện các bước trên trong 8051.

Ví dụ 6.3:

Hãy viết chương trình cộng hai số 16 bit. Các số đó là 3CE7H và 3B8DH. Cất tổng số vào R7và R6 trong đó R6 chứa byte thấp.

Lời giải:

```
CLR                   ; Xoá cờ CY = 0
MOV     A, #0E7H      ; Nạp byte thấp vào A → A = E7H
ADD     A, #8DH       ; Cộng byte thấp vào A → a = 74H và CY = 1
MOV     R6, A         ; Lưu byte thấp của tổng vào R6
MOV     A, #3CH       ; Nạp byte cao vào A → A = 3CH
ADDC    A, #3BG       ; Cộng byte cao có nhớ vào A → A = 78H
                      ;
```

```
MOV     R7, A          ; Lưu byte cao của tổng vào R7
```

6.1.1.3 Hệ thống số BCD (số thập phân mã hoá theo nhị phân).

Số BCD là số thập phân được mã hoá theo nhị phân 9 mà không dùng số thập phân hay số thập lục (Hex). Biểu diễn nhị phân của các số từ 0 đến 9 được gọi là BCD (xem hình 6.1). Trong tài liệu máy tính ta thường gặp hai khái niệm đối với các số BCD là: BCD được đóng gói và BCD không đóng gói.

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

Hình 6.1: Mã BCD.

a- BCD không đóng gói.

Trong số BCD không đóng gói thì 4 bít thấp của số biểu diễn số BCD còn 4 bit còn lại là số 9. Ví dụ "00001001" và "0000 0101" là những số BCD không đóng gói của số 9 và số 5. Số BCD không đóng gói đòi hỏi một byte bộ nhớ hay một thanh ghi 8 bit để chứa nó.

b- BCD đóng gói.

Trong số BCD đóng gói thì một byte có 2 số BCD trong nó một trong 4 bit thấp và một trong 4 bit cao. Ví dụ "0101 1001" là số BCD đóng gói cho 59H. Chỉ mất 1 byte bộ nhớ để lưu các toán hạng BCD. Đây là lý do để dùng số BCD đóng gói vì nó hiệu quả gấp đôi trong lưu giữ liệu.

Có một vấn để khi cộng các số BCD mà cần phải được khắc phục. Vấn đề đó là sau khi cộng các số BCD đóng gói thì kết quả không còn là số BCD. Ví dụ:

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

Cộng hai số này cho kết quả là 0011 1111B (3FH) không còn là số BCD! Một số BCD chỉ nằm trong giải 0000 đến 1001 (từ số 0 đến số 9). Hay nói cách khác phép cộng hai số BCD phải cho kết quả là số BCD. Kết quả trên đáng lẽ phải là 17 + 28 = 45 (0100 0101). Để giải quyết vấn đề này lập trình viên phải cộng 6 (0110) vào số thấp 3F + 06 = 45H. Vấn đề tương tự cũng có thể xảy ra trong số cao (ví dụ khi cộng hai số 52H + 87H = D94). Để giải quyết vấn đề này ta lại phải cộng 6 vào số cao (D9H + 60H = 139). Vấn đề này phổ biến đến mức mọi bộ xử lý như 8051 đều có một lệnh để sử lý vấn đề này. Trong 8051 đó là lệnh "DA A" để giải quyết vấn đề cộng các số BCD.

6.1.1.4 Lệnh DA.

Lệnh DA (Decimal Adjust for addition điều chỉnh thập phân đối với phép cộng) trong 8051 để dùng hiệu chỉnh sự sai lệch đã nói trên đây liên quan đến phép cộng các số BCD. Lệnh giả "DA". Lệnh DA sẽ cộng 6 vào 4 bit thấp hoặc 4 bit cao nếu cần. Còn bình thường nó đê nguyên kết quả tìm được. Ví dụ sau sẽ làm rõ các điểm này.

```
MOV     A, #47H     ; A = 47H là toán hạng BCD đầu tiên
MOV     B, #25H     ; B = 25H  là toán hạng BCD thứ hai
ADD     A, B        ; Cộng các số hex (nhị phân) A = 6CH
DA      A           ; Điều chỉnh cho phép cộng BCD (A = 72H)
```

Sau khi chương trình được thực hiện thanh ghi A sẽ chứa 72h (47 + 25 = 72). Lệnh "DA" chỉ làm việc với thanh ghi A. Hay nói cách khác trong thanh ghi nguồn có thể là một toán hạng của chế độ đánh địa chỉ bất kỳ thì đích phải là thanh ghi A để DA có thể làm việc được. Cũng cần phải nhấn mạnh rằng lệnh DA phải được sử dụng sau phép cộng các toán hạng BCD và các toán hạng BCD không bao giờ có thể có số lớn hơn 9. Nói cách khác là không cho phép có các số A - F. Điều quan trọng cũng phải lưu ý là DA chỉ làm việc sau phép cộng ADD, nó sẽ không bao giờ làm việc theo lệnh tăng INC.

Tóm tắt về hoạt động của lệnh DA.

Hoạt động sau lệnh ADD hoặc ADDC.

1. Nếu 4 bit thấp lớn hơn 9 hoặc nếu AC = 1 thì nó cộng 0110 vào 4 bít thấp.
2. Nếu 4 bit cao lớn hơn 9 hoặc cờ CY = 1 thì nó cộng 0110 vào 4 bit cao.

Trong thực tế thì cờ AC chỉ để dùng phục vụ cho phép cộng các số BCD và hiệu chỉnh nó. Ví dụ, cộng 29H và 18H sẽ có kết quả là 41H sai với thực tế khi đó các số BCD và để sửa lại thì lệnh DA sẽ cộng 6 vào 4 bit thấp để có kết quả là đúng (vì AC = 1) ở dạng BCD.

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

Ví dụ 6.4:

Giả sử 5 dữ liệu BCD được lưu trong RAM tại địa chỉ bắt đầu từ 40H như sau: 40 = (71), 41 = (11), 42 = (65), 43 = (59) và 44 = (37). Hãy viết chương trình tính tổng của tất cả 5 số trên và kết quả phải là dạng BCD.

Lời giải:

```
        MOV     R0, #40H    ; Nạp con trỏ
        MOV     R2, #5      ; Nạp bộ đếm
        CLR     A           ; Xoá thanh ghi A
        MOV     R7, A       ; Xoá thanhg ghi R7
AGAIN:  ADD     A, @R0      ; Cộng byte con trỏ chỉ bởi R0
        DA      A           ; Điều chỉnh về dạng BCD đúng
        JNC     NEXT        ; Nếu CY = 0 không tích luỹ cờ nhớ
        JNC     R7          ; Tăng R7 bám theo số lần nhớ
NEXT:   INC     R0          ; Tăng R0 dịch con trỏ lên ô nhớ kế tiếp
        DJNZ    R2, AGAIN   ; Lặp lại cho đến khi R2 = 0
```

6.1.2 Phép trừ các số không dấu.

Cú pháp: SUBB A, nguồn; A = A - nguồn - CY.

Trong rất nhiều các bộ xử lý có hai lệnh khác nhau cho phép trừ đó là SUB và SUBB (trừ có mượn - Sub, tract with Borrow). Trong 8051 ta chỉ có một lệnh SUBB

duy nhất. Để thực hiện SUB từ SUBB, do vậy có hai trường hợp cho lệnh SUBB là: với CY = 0 và với CY = 1. Lưu ý rằng ở đây ta dùng cờ CY để mượn.

6.1.2.1 Lệnh SUBB với CY = 0.

Trong phép trừ thì các bộ vi xử lý 8051 (thực tế là tất cả mọi CPU hiện đại) đều sử dụng phương pháp bù 2. Mặc dù mỗi CPU đều có mạch cộng, nó có thể quá cồng kềnh (và cần nhiều bóng bán dẫn) để thiết kế mạch trừ riêng biệt. Vì ly do đó mà 8051 sử dụng mạch cộng để thực hiện lệnh trừ. Giả sử 8051 sử dụng mạch cộng để thực hiện lệnh trừ và rằng CY - 0 trước khi thực hiện lệnh thì ta có thể tóm tắt các bước mà phần cứng CPU thực hiện lệnh SUBB đối với các số không dấu như sau:

1. Thực hiện lấy bù 2 của số trừ (toán hạng nguồn)
2. Cộng nó vào số bị trừ (A)
3. Đảo nhớ

Đây là 3 bước thực hiện bởi phần cứng bên trong của CPU 8051 đối với mỗi lệnh trừ SUBB bất kể đến nguồn của các toán hạng được cấp có được hỗ trợ chế độ đánh địa chỉ hay không? Sau ba bước này thì kết quả có được và các cờ được bật. Ví dụ 6.5 minh hoạ 3 bước trên đây:

Ví dụ 6.5:

Trình bày các bước liên quan dưới đây:

```
CLR     C           ; Tạo CY = 0
MOV     A, #3FH     ; Nạp 3FH vào A (A = 3FH)
MOV     R3, #23H    ; Nạp 23H vào R3 (R3 = 23H)
SUBB    A, R3       ; Trừ A cho R3 đặt kết quả vào A
```

Lời giải:

```
      A =  3F     0011  1111        0011  1111
  -   R3 = 23     0010  0011    +   1101  1101   bù 2 của R3 (bước 1)
        1C                        1  0001  1100 - 1C (bước 2)
                                  0  CF = 0 (bước 3)
```

Các cờ sẽ được thiết lập như sau: CY = 0, AC = 0 và lập trình viên phải được nhìn đến cờ nhớ để xác định xem kết quả là âm hay dương.

Nếu sau khi thực hiện SUBB mà CY = 0 thì kết quả là dương. Nếu CY = 1 thì kết quả âm và đích có giá trị bù 2 của kết quả. Thông thường kết quả được để ở dạng bù 2 nhưng các lệnh bù CPL và tăng INC có thể được sử dụng để thay đổi nó. Lệnh CPL thực hiện bù 1 của toán hạng sau đó toán hạng được tăng lên 1 (INC) để trở thành dạng bù 2. Xem ví dụ 6.6.

Ví dụ 6.6:

Phân tích chương trình sau:

```
        CLR   C
        MOV   A, #4CH       ; Nạp A giá trị 4CH (A = 4CH)
        SUBB  A, #6EH       ; Trừ A cho 6EH
        JNC   NEXT          ; Nếu CY = 0 nhảy đến đích NEXT
        CPL   A             ; Nếu CY = 1 thực hiện bù 1
        INC   A             ; Tăng 1 để có bù 2
NEXT:   MOV   R1, A         ; Lưu A vào R1
```

Lời giải:

Các bước thực hiện lệnh "SUBB A, 6EH" như sau:

```
      4C      0100    1100                    0100    1100
  -   6E      0110    1110    → lấy bù 2      1001    0010    (bước 1)
    - 22                              0       1101    1110 = (bước 2)
                                              đảo CY = 1(bước 3)
```

Cờ CY = 1, kết quả âm ở dạng bù 2.

6.1.2.2 Lệnh SUBB khi CY = 1.

Lệnh này được dùng đối với các số nhiều byte và sẽ theo dõi việc mượn của toán hạng thấp. Nếu CY = 1 trước khi xem thực hiện SUBB thì nó cũng trừ 1 từ kết quả. Xem ví dụ 6.7.

Ví dụ 6.7:

Phân tích chương trình sau:

```
CLR     C           ; CY = 0
MOV     A, #62      ; 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
```

Lời giải:

Sau khi SUBB thì A = 62H - 96H = CCH và cờ nhớ được lập báo rằng có mượn. Vì CY = 1 nên khi SUBB được thực hiện lần thứ 2 thì a = 27H - 12H - 1 = 14H. Do vậy, ta có 2762H - 1296H = 14CCH.

6.2 Nhân và chia các số không dấu.

Khi nhân và chia hai số trong 8051 cần phải sử dụng hai thanh ghi A và B vì các lệnh nhân và chia chỉ hoạt động với những thanh ghi này.

6.2.1 Nhân hai số không dấu.

Bộ vi điều khiển chỉ hỗ trợ phép nhân byte với byte. Các byte được giả thiết là dữ liệu không dấu. Cấu trúc lệnh như sau:

```
MOV  AB     ; Là phép nhân A × B và kết quả 16 bit được đặt trong A và B.
```

Khi nhận byte với byte thì một trong các toán hạng phải trong thanh ghi A và toán hạng thứ hai phải ở trong thanh ghi B. Sau khi nhân kết quả ở trong các thanh ghi A và B. Phần tiếp thấp ở trong A, còn phần cao ở trong B. Ví dụ dưới đây trình bày phép nhân 25H với 65H. Kết quả là dữ liệu 16 bit được đặt trong A và B.

```
MOV     A, #25H         ; Nạp vào A giá trị 25H
MOV     B, 65H          ; Nạp vào B giá trị 65H
MUL     AB              ; 25H*65H = E99 với B = 0EH và A = 99H
```

Bảng 6.1: Tóm tắt phép nhân hai số không dấu (MULAB)

Nhân	Toán hạng 1	Toán hạng 2	Kết quả
Byte*Byte	A	B	A = byte thấp, B = byte cao

6.2.2 Chia hai số không dấu.

8051 cùng chỉ hỗ trợ phép chia hai số không dấu byte cho byte với cú pháp:

DIV AB ; Chia A cho B

Khi chia một byte cho một byte thì tử số (số bị chia) phải ở trong thanh ghi A và mẫu số (số chia) phải ở trong thanh ghi B. Sau khi lệnh chia DIV được thực hiện thì thương số được đặt trong A, còn số dư được đặt trong B. Xét ví dụ dưới đây:

```
MOV     A, #95      ; Nạp số bị chia vào A = 95
MOV     B, #10      ; Nạp số chia vào B = 10
DIV     AB          ; A = 09 (thương số); B = 05 (số dư)
```

Lưu ý các điểm sau khi thực hiện "DIV AB"

Lệnh này luôn bắt CY = 0 và OV = 0 nếu tử số không phải là số 0

Nếu tử số là số 0 (B = 0) thì OV =1 báo lỗi và CY = 0. Thực tế chuẩn trong tất cả mọi bộ vi xử lý khi chia một số cho 0 là bằng cách nào đó báo có kết quả không xác định. Trong 8051 thì cờ OV được thiết lập lên 1.

Bảng 6.2: Tóm tắt phép chia không dấu (DIV AB).

Phép chia	Tử số	Mẫu số	Thương số	Số dư
Byte cho Byte	A	B	A	B

6.2.3 Một ứng dụng cho các lệnh chia.

Có những thời điểm khi một bộ ADC được nối tới một cổng và ADC biểu diễn một số dư nhiệt độ hay áp suất. Bộ ADC cấp dữ liệu 8 bit ở dạng Hex trong dải 00 - FFH. Dữ liệu Hex này phải được chuyển đổi về dạng thập phân. Chúng ta thực hiện chia lặp nhiều lần cho 10 và lưu số dư vào như ở ví dụ 6.8.

Ví dụ 6.8:

a- Viết một chương trình để nhận dữ liệu dạng Hex trong phạm vi 00 - FFH từ cổng 1 và chuyển đổi nó về dạng thập phân. Lưu các số vào trong các thanh ghi R7, R6 và R5 trong đó số có nghĩa nhỏ nhất được cất trong R7.

b- Phân tích chương trình với giả thiết P1 có giá trị FDH cho dữ liệu.

Lời giải:

a)

```
MOV     A, #0FFH
MOV     P1, A       ; Tạo P1 là cổng đầu vào
MOV     A, P1       ; Đọc dữ liệu từ P1
MOV     B, #10      ; B = 0A Hex (10 thập phân)
DIV     AB          ; Chia cho 10
MOV     R7, B       ; Cất số thấp
MOV     B, #10      ;
DIV     AB          ; Chia 10 lần nữa
MOV     R6, B       ; Cất số tiếp theo
MOV     R5, A       ; Cất số cuối cùng
```

b) Để chuyển đổi số nhị phân hay Hex về số thập phân ta thực hiện chia lặp cho 10 liên tục cho đến khi thương số nhỏ hơn 10. Sau mỗi lần chia số dư được lưu cất.

Trong trường hợp một số nhị phân 8 bit như FDH chẳng hạn ta có 253 số thập phân như sau (tất cả trong dạng Hex)

	Thương số	Số dư	
FD/0A	19	3	(Số thấp - cuối)
19/0A	2	5	(Số giữa)
		2	(Số đầu)

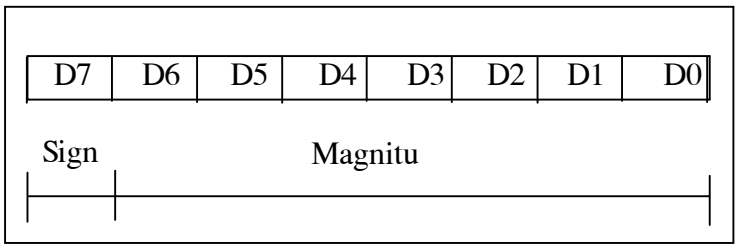
Do vậy, ta có FDH = 253. Để hiển thị dữ liệu này thì nó phải được chuyển đổi về ASCII mà sẽ được mô tả ở chương sau.

6.3 Các khái niệm về số có dấu và các phép tính số học.

Tất cả mọi dữ liệu từ trước đến giờ đều là các số không dấu, có nghĩa là toàn bộ toán hạng 8 bit đều được dùng cho bộ lớn. Có nhiều ứng dụng yêu cầu dữ liệu có dấu, phần này sẽ bàn về những lệnh liên quan đến các số có dấu.

6.3.1 Khái niệm về các số có dấu trong máy tính.

Trong cuộc sống hàng ngày các số được dùng có thể là số âm hoặc dương. Ví dụ 5 độ dưới 0^0C được biểu diễn là -5^0C và 20 độ trên 0^0C được biểu diễn là $+20^0$C. Các máy tính cũng phải có khả năng đáp ứng phù hợp với các số ấy. Để làm được điều ấy các nhà khoa học máy tính đã phát minh ra sự xắp xếp biểu diễn các số âm có dấu và số dương có dấu như sau: Bit cao nhất MSB được để dành cho bit dấu (+) hoặc (-), còn các bit còn lại được dùng biểu diễn độ lớn. Dấu được biểu diễn bởi 0 đối với các số dương và một số đối với các số âm (-). Biểu diễn của một byte có dấu được trình bày trên hình 6.2.



Hình 6.2: Các toán hạng 8 bit có dấu.

a- Các toán hạng 8 bit có dấu: Trong các toán hạng A byte có dấu thì bit cao nhất MSB là D7 được dùng để biểu diễn dấu, còn 7 bit còn lại từ D6 - D0 dùng để biểu diễn độ lớn của số đó. Nếu D7 = 0 thì đó là toán hạng dương và nếu D7 = 1 thì nó là toán hạng âm.

b- Các số dương: Dải của các số dương có thể được biểu diễn theo dạng cho trên hình 6.2 là từ 0 đến +127 thì phải sử dụng toán hạng 16 bit. Vì 8051 không hỗ trợ dữ liệu 16 bit nên ta không bàn luận đến.

c- Các số âm: Đối với các số âm thì D7 = 1, tuy nhiên độ lớn được biểu diễn ở dạng số bù 2 của nó. Mặc dù hợp ngữ thực hiện việc chuyển đổi song điều quan trọng là hiểu việc chuyển đổi diễn ra như thế nào. Để chuyển đổi về dạng biểu diễn số âm (bù 2) thì tiến hành theo các bước sau:

1. Viết độ lớn của số ở dạng nhị phân 8 bit (không dấu).
2. Đảo ngược tất cả các bit
3. Cộng 1 vào nó.

Ví dụ 6.9: Hãy trình bày cách 8051 biểu diễn số - 5.

Lời giải:

Hãy quan sát các bước sau:

0000	0101	Biểu diễn số 5 ở dạng 8 bit nhị phân
1111	1010	Đảo các bit
1111	1011	Cộng (thành số FB ở dạng Hex)

Do vậy, số FBH là biểu diễn số có dấu dạng bù 2 của số - 5.

Ví dụ 6.10: Trình bày cách 8051 biểu diễn - 34H.

Lời giải:

Hãy quan sát các bước sau:

0011	0200	Số 34 được cho ở dạng nhị phân
1100	1011	Đảo các bit
1100	1100	Cộng 1 (thành số CC ở dạng Hex)

Vậy số CCH là biểu diễn dạng bù 2 có dấu của - 34H.

Ví dụ 6.11: Trình bày cách 8051 biểu diễn - 128:

Lời giải:

Quan sát các bước sau:

1000	0000	Số 128 ở dạng nhị phân 28 bit
0111	1111	Đảo các bit
1000	0000	Cộng 1 (trở thành số 80 dạng Hex)

Vậy - 128 = 80H là biểu diễn số có dấu dạng bù 2 của - 128.

Từ các ví dụ trên đây ta thấy rõ ràng rằng dải của các số âm có dấu 8 bit là - 1 đến - 128. Dưới đây là liệt kê các số có dấu 8 bit:

Số thập phân	Số nhị phân	Số Hex
-128	1000 0000	80
-127	1000 0001	81
-126	1000 0010	82
...	.....	...
-2	1111 1110	FE
-1	1111 1111	FF
0	0000 0000	00
+1	0000 0001	01
+2	0000 0010	02
...	.....	...
-127	0111 1111	FE

6.3.2 Vấn đề tràn trong các phép toán với số có dấu.

Khi sử dụng các số có dấu xuất hiện một vấn đề rất nghiêm trọng mà phải được sử lý. Đó là vấn đề tràn, 8051 báo có lỗi bằng cách thiết lập cờ tràn OV nhưng trách nhiệm của lập trình viên là phải cẩn thận với kết quả sai. CPU chỉ hiểu 0 và 1 và nó làm ngơ với việc chuyển đổi số âm, số dương của con người. Vậy tràn số là gì? Nếu kết quả của một phép toán trên các số có dấu mà quá lớn đối với thanh ghi thì xuất hiện sự tràn số và lập trình viên phải được cảnh báo. Xét ví dụ 6.12 dưới đây.

Ví dụ 6.12:

Khảo sát đoạn mã sau và phân tích kết quả.

```
MOV     A, # + 96       ; A = 0110      0000 (A = 60H)
MOV     R1, # + 70      ; R1 = 0100     0110 (R1 = 46H)
ADD     A, R1           ; A = 1010      0110 = A6H = - 90
                          Sai !!!
```

Lời giải:

```
    + 96            0110    0000
+   + 70            0100    0110
  - 166             1010    0110        và OV = 1
```

Theo CPU kết quả là -90 và đó là kết quả sai nên CPU bật cờ OV = 1 để báo tràn số.

Trong ví dụ 6.12 thì + 96 được cộng với + 70 và kết quả theo CPU là - 90. Tại sao vậy? Lý do là kết quả của + 96 + 70 = 172 lớn hơn số mà thanh ghi A có thể chứa được. Cũng như tất cả mọi thanh ghi 8 bit khác, thanh ghi A chỉ chứa được đến số + 127. Các nhà thiết kế của PCU tạo ra cờ tràn OV phục vụ riêng cho mục đích báo cho lập trình viên rằng kết quả của phép toán số có dấu là sai.

6.3.3 Khi nào thì cờ tràn OV được thiết lập?

Trong các phép toán với số có dấu 8 bit thì cờ OV được bật lên 1 khi xuất hiện một trong hai điều kiện sau:

1. Cờ nhớ từ D6 sang D7 nhưng không có nhớ ra từ D7 (cờ CY = 0)
2. Có nhớ ra từ D7 (cờ CY = 1) nhưng không có nhớ từ D6 sang D7

Hay nói cách khác là cờ tràn OV được bật lên 1 nếu có nhớ từ D6 sang D7 hoặc từ D7 nhưng không đồng thời xảy ra cả hai. Điều này có nghĩa là nếu có nhớ cả từ D6 sang D7 và từ D7 ra thì cờ OV = 0. Trong ví dụ 6.12 vì chỉ có nhớ từ D7 ra nên cờ OV = 1. Trong ví dụ 6.13, ví dụ 6.14 và 6.15 có minh hoạ thêm về sử dụng cờ tràn trong các phép số học với số có dấu.

Ví dụ 6.13:

Hãy quan sát đoạn mã sau để ý đến vai trò của cờ OV.

```
MOV     A, # -128       ; A = 1000      0000 (A= 80H)
MOV     R4, # -2        ; R4 = 1111     (R4 = FEH)
ADD     A, R4           ; A = 0111      1110 (A = 7EH = +126, invalid)
```

Lời giải:

```
- 128           1000    0000
+ - 2           1111    1110
-130            0111    1110    và OV = 1
```

Theo CPU thì kết quả + 126 là kết quả sai, nên cờ OV = 1.

Ví dụ 6.14:

Hãy quan sát đoạn mã sau và lưu ý cờ OV.

```
MOV     A, # -2         ; A = 1111      1110 (A = FEH)
MOV     R1, # -5        ; R1 = 1111     1011    (R1 = FBH)
ADD     A, R1           ; A = 1111      1001    (A = F9H = -7, correct, OV = 0)
```

Lời giải:

```
  - 2       1111    1110
+ - 5       1111    1011
-   7       1111    1001    và OV = 0
```

Theo CPU thì kết quả - 7 là đúng nên cờ OV = 0.

Ví dụ 6.15:

Theo dõi đoạn mã sau, chú ý vai trò của cờ OV.

```
MOV     A, # +7         ; A =  0000     0111 (A = 07H)
MOV     R1, # +18       ; R1 = 0001     0010 (R1 = 12H)
ADD     A, R1           ; A = 1111      1001    (A = 19H = -25, correct, OV = 0)
```

Lời giải:

```
    7       0000    0111
- 18        0001    0010
  25        0001    1001    và OV = 0
```

Theo CPU thì kết quả - 25 là đúng nên cờ OV = 0.

Từ các ví dụ trên đây ta có thể kết luận rằng trọng bất kỳ phép cộng số có dấu nào, cờ OV đều báo kết quả là đúng hay sai. Nếu cờ OV = 1 thì kết quả là sai, còn nếu OV = 0 thì kết quả là đúng. Chúng ta có thể nhấn mạnh rằng, trong phép cộng các số không dấu ta phải hiển thị trạng thái của cờ CY (cờ nhớ) và trong phép cộng các số có dấu thì cờ tràn OV phải được theo dõi bởi lập trình viên. Trong 8051 thì các lệnh như JNC và JC cho phép chương trình rẽ nhánh ngay sau phép cộng các số không dấu như ở phần 6.1. Đối với cờ tràn OV thì không có như vậy. Tuy nhiên, điều này có thể đạt được bằng lệnh "JB PSW.2" hoặc "JNB PSW.2" vì PSW thanh ghi cờ có thể đánh địa chỉ theo bit.