CHƯƠNG 7
Các lệnh lô - gíc và các chương trình

7.1 Các lệnh lô-gíc và so sánh.

7.1.1 Lệnh VÀ (AND).

Cú pháp: ANL đích, nguồn; đích = đích Và nguồn (kẻ bảng).

Lệnh này sẽ thực hiện một phép Và lô-gíc trên hai toán hạng đích và nguồn và đặt kết quả vào đích. Đích thường là thanh ghi tổng (tích luỹ). Toán hạngnguồn có thể là thanh ghi trong bộ nhớ hoặc giá trị cho sẵn. Hãy xem phụ lục Appendix A1 để biết thêm về các chế độ đánh địa chỉ dành cho lệnh này. Lệnh ANL đối với toán hạng theo byte không có tác động lên các cờ. Nó thường được dùng để che (đặt về 0) những bit nhất định của một toán hạng. Xem ví dụ 7.1.

Ví dụ:

Trình bày kết quả của các lệnh sau:

```
MOV     A, #35H         ; Gán A = 35H
ANL     A, #0FH         ; Thực hiện Và lô-gíc A và 0FH (Bây giờ A = 05)
```

Lời giải:

```
35H     0  0  1  1  0  1  0  1
0FH     0  0  0  0  1  1  1  1
05H     0  0  0  0  0  1  0  1          35H  và 0FH = 05H
```

7.1.2: Lệnh HOẶC (OR).

Cú pháp ORL đích = đích Hoặc nguồn (kẻ bảng)

Các toán hạng đích và nguồn được Hoặc với nhau và kết quả được đặt vào đích. Phép Hoặc có thể được dùng để thiết lập những bit nhất định của một toán hạng 1. Đích thường là thanh ghi tổng, toán hạng nguồn có thể là một thanh ghi trong bộ nhớ hoặc giá trị cho sẵn. Hãy tham khảo phụ lục Appendix A để biết thêm về các chế độ đánh địa chỉ được hỗ trợ bởi lệnh này. Lệnh ORL đối với các toán hạng đánh địa chỉ theo byte sẽ không có tác động đến bất kỳ cờ nào. Xem ví dụ 7.2.

Ví dụ 7.2: Trình bày kết quả của đoạn mã sau:

```
MOV     A, #04          ; A = 04
MOV     A, #68H ; A = 6C
```

Lời giải:

```
04H     0000  0100
68H     0110  1000
6CH     0110  1100          04  OR  68 = 6CH
```

7.1.3 Lệnh XOR (OR loại trừ?).

Cú pháp: XRL đích, nguồn; đích = đích Hoặc loại trừ nguồn (kẻ bảng).

Lệnh này sẽ thực hiện phép XOR trên hai toán hạng và đặt kết quả vào đích. Đích thường là thanh ghi tổng. Toán hạng nguồn có thể là một thanh ghi trong bộ nhớ hoặc giá trị cho sẵn. Xem phụ lục Appendix A.1 để biết thêm về chế độ đánh địa chỉ của lệnh này. Lệnh XRL đối với các toán hạng đánh địa chỉ theo byte sẽ không có tác động đến bất kỳ cờ nào. Xét ví dụ 7.3 và 7.4.

Ví dụ 7.3: Trình bày kết quả của đoạn mã sau:

```
MOV     A, #54H
XRL     A, #78H
```

Lời giải:

```
54H     0  1  0  1  0  1  0  0
78H     0  1  1  1  1  0  0  0
2CH     0  0  1  0  1  1  0  1          54H  XOR  78H = 2CH
```

Ví dụ 7.4:

Lệnh XRL có thể được dùng để xoá nội dung của một thanh ghi bằng cách XOR nó với chính nó. Trình bày lệnh "XRL A, A" xoá nội dung của A như thế nào? giả thiết AH = 45H.

Lời giải:

```
45H             01000101
45H             01000101
00              00000000          54H  XOR  78H = 2CH
```

Lệnh XRL cũng có thể được dùng để xem nếu hai thanh ghi có giá trị giống nhau không? Lệnh "XRL A, R1" sẽ hoặc loại trừ với thanh ghi R1 và đặt kết quả vào A. Nếu cả hai thanh ghi có cùng giá trị thì trong A sẽ là 00. Sau đó có thể dùng lệnh nhả JZ để thực hiện theo kết quả. Xét ví dụ 7.5.

Ví dụ 7.5:

Đọc và kiểm tra cổng P1 xem nó có chứa giá trị 45H không? Nếu có gửi 99H đến cổng P2, nếu không xoá nó.

Lời giải:

```
        MOV     P2, #00             ; Xóa P2
        MOV     P1, #0FFH           ; Lấy P1 là cổng đầu vào
        MOV     R3, #45H            ; R3 = 45H
        MOV     A, P1               ; Đọc P1
        XRL     A, R3
        JNZ     EXIT                ; Nhảy nếu A có giá trị khác 0
        MOV     P2, #99H
EXIT: ...
```

Trong chương trình của ví dụ 7.5 lưu ý việc sử dụng lệnh nhảy JNZ. Lệnh JNZ và JZ kiểm tra các nội dung chỉ của thanh ghi tổng. Hay nói cách khác là trong 8051không có cờ 0.

Một ứng dụng rộng rãi khác của bộ xử lý là chọn các bit của một toán hạng. Ví dụ để chọn 2 bit của thnh ghi A ta có thể sử dụng mã sau. Mã này ép bit D2 của thanh ghi A chuyển sang giá trị nghịch đảo, còn các bit khác không thay đổi.

```
XRL     A, #04H          ; Nghĩa hoặc loại trừ thanh ghi A với
                         ; Giá trị 0000 0100
```

7.1.4 Lệnh bù thanh ghi tổng CPL A.

Lệnh này bù nội dung của thanh ghi tổng A. Phép bù là phép biến đổi các số 0 thành các số 1 và đổi các số 1 sang số 0. Đây cũng còn được gọi là phép bù 1.

```
MOV          A, #55H
CPL          A           ; Bây giờ nội dung của thanh ghi A là AAH
                         ; Vì 0101 0101 (55H) → 1010 1010 (AAH)
```

Để nhận được kết quả bù 2 thì tất cả mọi việc ta cần phải làm là cộng 1 vào kết quả bù 1. Trong 8051 thì không có lệnh bù 2 nào cả. Lưu ý rằng trong khi bù một byte thì dữ liệu phải ở trong thanh ghi A. Lệnh CPL không hỗ trợ một chế độ đánh địa chỉ nào cả. Xem ví dụ 7.6 dưới đây.

Ví dụ 7.6: Tìm giá trị bù 2 của 85H.

Lời giải:

```
MOV          A, #85H     ; Nạp 85H vào A (85H = 1000 0101)
MOV          A           ; Lấy bù 1 của A (kết quả = 0111 1010)
ADD          A, #1       ; Cổng 1 vào A thành bù 2 A = 0111 1011 (7BH)
```

Ví dụ 7.1.5 Lệnh so sánh.

8051 có một lệnh cho phép so sánh. Nó có cú pháp như sau:

CJNE đích, nguồn, địa chỉ tương đối.

Trong 8051 thì phép so sánh và nhảy được kết hợp thành môt lệnh có tên là CJNE (so sánh và nhảy nếu kết quả không bằng nhau). Lệnh CJNE so sánh hai toán hạng nguồn và đích và nhảy đến địa chỉ tương đối nếu hai toán hạng không bằng nhau. Ngoài ra nó thay đổi cờ nhớ CY để báo nếu toán hạng đích lớn hơn hay nhỏ hơn. Điều quan trọng cần để là các toán hạng vẫn không giữ nguyên không thay thay đổi. Ví dụ, sau khi thực hiện lệnh "CJNE A, #67H, NEXT" thì thanh ghi A vẫn có giá trị ban đầu của nó (giá trị trước lệnh CJNE). Lệnh này so sánh nội dung thanh ghi A với giá trị 67H và nhảy đến giá trị đích NEXT chỉ khi thanh ghi A có giá trị khác 67H.

Ví dụ 7.7:

Xét đoạn mã dưới đây sau đó trả lời câu hỏi:

a) Nó sẽ nhảy đến NEXT không?

b) Trong A có giá trị bao nhiêu sau lệnh CJNE?

```
        MOV          A, #55H
        CJNE         A, #99H, NEXT
        ...
NEXT: ...
```

Lời giải:
a) Có vì 55H và 99H không bằng nhau
b) A = 55H đây là giá trị trước khi thực hiện CJNE.

Trong lệnh CJNE thì toán hạng đích có thể trong thanh ghi tổng hoặc trong một các thanh ghi Rn. Toán hạng nguồn có thể trong một thanh ghi, trong bộ nhớ hoặc giá trị cho sẵn. Hãy xem phụ lục Appendix A để biết thêm chi tiết về các chế độ đánh địa chỉ cho lệnh này. Lệnh này chỉ tác động cờ nhớ CY. Cờ này được thay đổi như chỉ ra trên bảng 7.1. Dưới đây trình bày phép so sánh hoạt động như thế nào đối với tất cả các điều kiện có thể:

```
                CJNE    R5, #80, NOT-EQUAL      ; Kiểm tra R5 có giá trị 80?
                ...                             ; R5 = 80
NOT-EQUAL:      JNC     NEXT                    ; Nhảy đến R5 > 80
                ...
NEXT:           ...
```

Bảng 7.1: Thiết kế cờ CY cho lệnh CJNE.

Compare	Carry Flag
Destinatio > Source	CY = 0
Destination < Source	CY = 1

Để ý rằng trong lệnh CJNE thì không có thanh ghi Rn nào có thể được so sánh với giá trị cho sẵn. Do vậy không cần phải nói đến thanh ghi A. Cũng cần lưu ý rằng cờ nhớ CY luôn được kiểm tra để xem lớn hơn hay nhỏ hơn, nhưng chỉ khi đã xác định là nó không bằng nhau. Xét ví dụ 7.8 và 7.9 dưới đây.

Ví dụ 7.8:

Hãy viết mã xác định xem thanh ghi A có chứa giá trị 99H không? Nếu có thì hãy tạo R1 = FFH còn nếu không tạo R1 = 0.

Lời giải:

```
                MOV     R1, #0          ; Xoá R1
                CJNE    A, #99H ; Nếu A không bằng 99H thì nhảy đến NEXT
                MOV     R1, #0FFH       ; Nếu chúng bằng nhau, gán R1 = 0FFH
NEXT: ...                               ; Nếu không bằng nhau, gán R1 = 0
OVER: ...
```

Ví dụ 7.9:

Giả sử P1 là một cổng đầu vào được nối tới một cảm biến nhiệt. Hãy viết chương trình đọc nhiệt độ và kiểm tra nó đối với giá trị 75. Theo kết quả kiểm tra hãy đặt giá trị nhiệt độ vào các thanh ghi được chỉ định như sau:

```
Nếu T = 75      thì A = 75
Nếu T < 75      thì R1 = T
Nếu T > 75      thì R2 = T
```

Lời giải:

```
        MOV     P1, 0FFH            ; Tạo P1 làm cổng đầu vào
        MOV     A, P1               ; Đọc cổng P1, nhiệt độ
        CJNE    A, #75, OVER        ; Nhảy đến OVER nếu A ≠ 75
        SJMP    EXIT                ; A = 75 thoát
OVER:   JNC     NEXT                ; Nếu CY = 0 thì A > 75 nhảy đến NEXT
        MOV     R1, A               ; Nếu CY = 1 thì A < 75 lưu vào R1
        SJMP    EXIT                ; Và thoát
NEXT:   MOV     R2, A               ; A > 75 lưu nó vào R2
EXIT:   ...
```

Lệnh so sánh thực sự là một phép trừ, ngoại trừ một điều là giá trị của các toán hạng không thay đổi. Các cờ được thay đổi tuỳ theo việc thực hiện lệnh trừ SUBB. Cần phải được nhấn mạnh lại rằng, trong lệnh CJNE các toán hạng không bị tác động bất kể kết quả so sánh là như thế nào. Chỉ có cờ CY là bị tác động, điều này bị chi phối bởi thực tế là lệnh CJNE sử dụng phép trừ để bật và xoá cờ CY.

Ví dụ 7.10:

Viết một chương trình để hiển thị liên tục cổng P1 đối với giá trị 63H. Nó chỉ mất hiển thị khi P1 = 63H.

Lời giải:

```
        MOV     P1, #0FFH           ; Chọn P1 làm cổng đầu vào
HERE:   MOV     A, P1               ; Lấy nội dung của P1
        CJNE    A, #63, HERE        ; Duy trì hiển thị trừ khi P1 = 63H
```

Ví dụ 7.11:

Giả sử các ngăn nhớ của RAM trong 40H - 44H chứa nhiệt độ hàng ngày của 5 ngày như được chỉ ra dưới đây. Hãy tìm để xem có giá trị nào bằng 65 không? Nếu giá trị 65 có trong bảng hãy đặt ngăn nhớ của nó vào R4 nếu không thì đặt R4 = 0.

40H = (76); 41H = (79); 42H = (69); 43H = (65); 44H = (64)

Lời giải:

```
        MOV     R4, #0              ; Xoá R4 = 0
        MOV     R0, #40H            ; Nạp con trỏ
        MOV     R2, #05             ; Nạp bộ đếm
        MOV     A, #65              ; Gán giá trị cần tìm vào A
BACK:   CJNE    A, @R0, NEXT        ; So sánh dữ liệu RAM với 65
        MOV     R4, R0              ; Nếu là 65, lưu địa chỉ vào R4
        SJMP    EXIT                ; Thoát
NEXT:   INC     R0                  ; Nếu không tăng bộ đếm
        DJNZ    R2, BACK            ; Tiếp tục kiểm tra cho đến khi bộ đếm bằng 0.
EXIT:   ...
```

7.2 Các lệnh quay vào trao đổi.

Trong rất nhiều ứng dụng cần phải thực hiện phép quay bit của một toán hạng. Các lệnh quay 8051 là R1, RR, RLC và RRC được thiết kế đặc biệt cho mục đích này. Chúng cho phép một chương trình quay thanh ghi tổng sang trái hoặc phải. Trong 8051 để quay một byte thì toán hạng phải ở trong thanh ghi tổng A. Có hai kiểu quay là: Quay đơn giản các bit của thanh ghi A và quay qua cờ nhớ (hay quay có nhớ).

7.2.1 Quay các bit của thanh ghi A sang trái hoặc phải.

a) Quay phải: RR A ; Quay các bit thanh ghi A sang phải.

Trong phép quay phải, 8 bit của thanh ghi tổng được quay sang phải một bit và bit D0 rời từ vị trí bit thấp nhất và chuyển sang bit cao nhất D7. Xem đoạn mã dưới đây.

```
MOV   A, #36H      ; A = 0011 0110
RR    A            ; A = 0001 1011
RR    A            ; A = 1000 1101
RR    A            ; A = 1100 0110
RR    A            ; A = 0110 0011
```

b) Quay trái:

Cú pháp: RL A ; Quay trái các bit của thanh ghi A (hình vẽ)

Trong phép quay trái thì 8 bit của thanh ghi A được quay sang trái 1 bit và bit D7 rời khỏi vị trí bit cao nhất chuyển sang vị trí bit thấp nhất D0. Xem biểu đồ mã dưới đây.

```
MOV   A, #72H      ; A = 0111 0010
RL    A            ; A = 1110 0100
RL    A            ; A = 1100 1001
```

Lưu ý rằng trong các lệnh RR và RL thì không có cờ nào bị tác động.

7.2.2 Quay có nhớ.

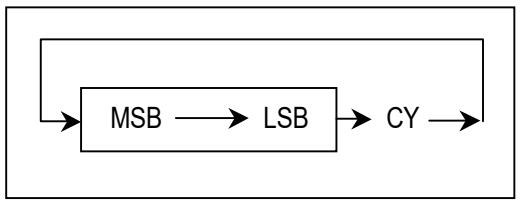
Trong 8051 còn có 2 kênh quay nữa là quay phải có nhớ và quay trái có nhớ.

Cú pháp: RRC A và RLC A

a) Quay phải có nhớ: RRC A

Trong quay phải có nhớ thì các bit của thanh ghi A được quay từ trái sang phải 1 bit và bit thấp nhất được đưa vào cờ nhớ CY và sau đó cờ CY được đưa vào vị trí bit cao nhất. Hay nói cách khác, trong phép RRC A thì LSB được chuyển vào CY và CY được chuyển vào MSB. Trong thực tế thì cờ nhớ CY tác động như là một bit bộ phận của thanh ghi A làm nó trở thành thanh ghi 9 bit.

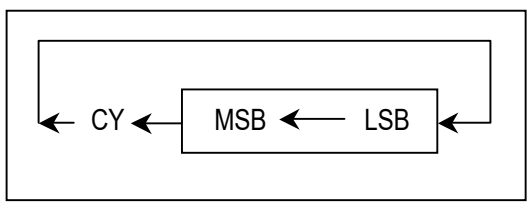
```
CLR   C            ; make CY = 0
MOV   A #26H       ; A = 0010 0110
RRC   A            ; A = 0001 0011  CY = 0
RRC   A            ; A = 0000 1001  CY = 1
RCC   A            ; A = 1000 0100  CY = 1
```



b) Quay trái có nhớ (hình vẽ): RLC A.

Trong RLC A thì các bit được dịch phải một bit và đẩy bit MSB vào cờ nhớ CY, sau đó CY được chuyển vào bit LSB. Hay nói cách khác, trong RLC thì bit MSB được chuyển vào CY và CY được chuyển vào LSB. Hãy xem đoạn mã sau.

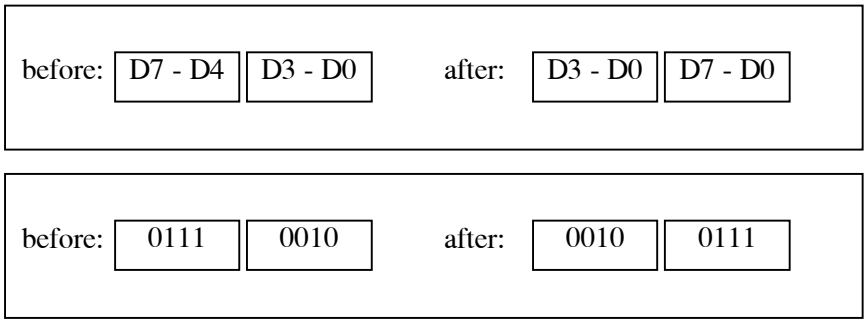
```
SETB  C            ; Make CY = 1
MOV   A #15H       ; A = 0001 0101
RRC   A            ; A = 0101 1011  CY = 0
RRC   A            ; A = 0101 0110  CY = 0
```



```
RCC     A               ; A = 1010 1100  CY = 0
RCC     A               ; A = 1000 1000  CY = 1
```

7.2.3 Lệnh trao đổi thanh ghi A: SWAP A

Một lệnh hữu ích khác nữa là lệnh trao đổi SWAP. Nó chỉ hoạt động trên thanh ghi A, nó trao đổi nửa phần cao của byte và nửa phần thấp của byte với nhau. Hay nói cách khác 4 bit cao được chuyển thành 4 bit thấp và 4 bit thấp thành 4 bit cao.



Ví dụ 7.12:

a) Hãy tìm nội dung của thanh ghi A ở đoạn mã sau.
b) Trong trường hợp không có lệnh SWAP thì cần phải làm như thế nào để trao đổi những bit này? Hãy viết một mã chương trình đơn giản về quá trình đó.

Lời giải:

a)

```
        MOV     A, #72H ; A = 72H
        SWAP    A               ; A = 27H
```

b)

```
        MOV     A, #72H ; A = 0111 0010
        RL      A               ; A = 1110 0100
        RL      A               ; A = 1100 1001
        RL      A               ; A = 0010 0111
```

Ví dụ 7.13:

Viết một chương trình để tìm số các số 1 trong một byte đã cho.

Lời giải:

```
                MOV     R1, #0          ; Chọn R1 giữ số các số 1
                MOV     R7, #8          ; Đặt bộ đếm  = 8 để quay 8 lần
                MOV     A, #97H         ; Tim các số 1 trong byte 97H
AGAIN:          RLC     A               ; Quay trái có nhớ một lần
                JNC     NEXT            ; Kiểm tra cờ CY
                INC     R1              ; Nếu CY = 1 thì cộng 1 vào bộ đếm
NEXT:           DJNZ    R7, AGAIN       ; Lặp lại quá trình 8 lần
```

Để truyền 1 byte dữ liệu nói tiếp thì dữ liệu có thể được chuyển đổi từ song song sang nối tiếp bằng các lệnh quay như sau:

```
RRC     A               ; Bít thứ nhất đưa vào cờ CY
MOV     P1.3, C         ; Xuất CY như một bit dữ liệu
RRC     A               ; Bit thứ hai đưa vào CY
```

```
MOV     P1.3, C         ; Xuất CY ra như một bit dữ liệu
RRC     A               ;
MOV     P1.3, C         ;
...
```

Đoạn mã trên đây là một phương pháp được sử dụng rộng rãi trong truyền dữ liệu tới các bộ nhớ nối tiếp như các EEPROM nối tiếp.

7.3 Các chương trình ứng dụng của mã BCD và ASCII.

Các số mã BCD đã được trình ở chương 6. Như đã nói ở đó rằng trong rất nhiều bộ vi điều khiển mới đều có một đồng hồ thời gian thực RTC (Real Time Clock) để giữ cho thời gian và cả lịch cho cả khi bị tắt nguồn. Các bộ vi điều khiển này cung cấp thời gian và lịch dưới dạng BCD. Tuy nhiên, để hiển thị chúng thì chúng phải được chuyển về mã ASCII. Trong phần này ta trình bày ứng dụng của các lệnh quay và các lệnh lô-gíc trong việc chuyển đổi mã BCD và ASCII.

Bảng 7.2: Mã ASCII cho các chữ số từ 0- 9.

Phím	Mã ASCII (Hex)	Mã ASCII nhị phân	Mã BCD (không đóng gói)
0	30	011 0000	0000 0000
1	31	011 0001	0000 0001
2	32	011 0010	0000 0010
3	33	011 0011	0000 0011
4	34	011 0100	0000 0100
5	35	011 0101	0000 0101
6	36	011 0110	0000 0110
7	37	011 0111	0000 0111
8	38	011 1000	0000 1000
9	39	011 1001	0000 1001

7.3.1 Các số mã ASCII.

Trên các bàn phím ASCII khi phím "0" được kích hoạt thì "011 0001" (30H) được cấp tới máy tính. Tương tự như vậy 31H (011 0001) được cấp cho phím "1" v.v... như cyhỉ ra trong bảng 7.2.

Cần phải ghi nhớ rằng mặc dù mã ASCII là chuẩn ở mỹ (và nhiều quốc gia khác) nhưng các số mã BCD là tổng quát. Vì bàn phìm, máy in và màn hình đều sử dụng mã ASCII nên cần phải thực hiện đổi chuyển giữa các số mã ASCII về số mã BCD và ngược lại.

7.3.2 Chuyển đổi mã BCD đóng gói về ASCII.

Các bộ vi điều khiển DS5000T đều có đồng bộ thời gian thực RTC. Nó cung cấp hiển thị liên tục thời gian trong ngày (giờ, phút và giây) và lịch (năm, tháng, ngày) mà không quan tâm đến nguồn tắt hay bật. Tuy nhiên dữ liệu này được cấp ở dạng mã BCD đóng gói. Để hiển thị dữ liệu này trên một LCD hoặc in ra trên máy in thì nó phải được chuyển về dạng mã ASCII.

Để chuyển đổi mã BCD đóng gói về mã ASCII thì trước hết nó phải được chuyển đổi thành mã BCD không đóng gói. Sau đó mã BCD chưa đóng gói được móc với 011 0000 (30H). Dưới đây minh hoạ việc chuyển đổi từ mã BCD đóng gói về mã ASCII. Xem ví dụ 7.14.

Mã BCD đóng gói	Mã BCD không đóng gói	Mã ASCII
29H	02H & 09H	32H & 39H
0010 1001	0000 0010 &	0011 0010 &
	0000 1001	0011 1001

7.3.3 Chuyển đổi mã ASCII về mã BCD đóng gói.

Để chuyển đổi mã ASCII về BCD đóng gói trước thì trước hết nó phải được chuyển về mã BCD không đóng gói (để có thêm 3 số) và sau đó được kết hợp để tạo ra mã SCD đóng gói. Ví dụ số 4 và số 7 thì bàn phím nhận được 34 và 37. Mục tiêu là tạo ra số 47H hay "0100 0111" là mã BCD đóng gói. Qúa trình này như sau:

Phím	Mã ASCII	Mã BCD không đóng gói	Mã BCD đóng gói
4	34	0000 0100	
7	37	0000 0111	0100 0111 hay 47H

```
MOV     A, # '4'        ; Gán A = 34H mã ASCII của số 4
MOV     R1, # '7'       ; Gán R1 = 37H mã ASCII của số 4
ANL     A, #0FH         ; Che nửa byte cao A (A = 04)
ANL     R1, #0FH        ; Che nửa byte cao của R1 (R1 = 07)
SWAP    A               ; A = 40H
ORL     A, R1           ; A = 47H, mã BCD đóng gói
```

Sau phép chuyển đổi này các số BCD đóng gói được xử lý và kết quả sẽ là dạng BCD đóng gói. Như ta đã biết ở chương 6 có một lệnh đặt biệt là "DA A" đòi hỏi dữ liệu phải ở dạng BCD đóng gói.

Ví dụ 7.14:

Giả sử thanh ghi A có số mã BCD đóng gói hãy viết một chương trình để chuyển đổi mã BCD về hai số ASCII và đặt chúng vào R2 và R6.

Lời giải:

```
MOV     A, #29H         ; Gán A = 29, mã BCD đóng gói
MOV     R2, A           ; Giữ một bản sao của BCD trong R2
ANL     A, #0FH         ; Che phần nửa cao của A (A = 09)
ORL     A, #30H         ; Tạo nó thành mã ASCII A = 39H (số 9)
MOV     R6, A           ; Lưu nó vào R6 (R6 = 39H ký tự của ASCII)
MOV     A, R2           ; Lấy lại giá trị ban đầu của A (A = 29H)
ANL     A, #0F0H        ; Che nửa byte phần thấp của A (A = 20)
RR      A               ; Quay phải
RR      A               ; Quay phải
RR      A               ; Quay phải
RR      A               ; Quay phải (A = 02)
ORL     A, #30H         ; Tạo nó thành mã ASCII (A = 32H, số 2)
MOV     R2, A           ; Lưu ký tự ASCII vào R2
```

Trong ví dụ trên tất nhiên là ta có thể thay 4 lệnh RR quay phải bằng một lệnh trao đổi WAPA.