6. Các nhóm lệnh cơ bản của 8051

TS Nguyễn Hồng Quang

6.1 Lập trình có cấu trúc

- Lập trình tuần tự (sequential programming)
- Lập trình cấu trúc (structure programming)
- Lập trình hướng đối tượng (object oriented programming)

6.2 Các đặc trưng lập trình cấu trúc

- Dữ liệu + giải thuật = chương trình
- Chương trình
  - Chương trình con
  - Có 3 loại cú pháp cơ bản
    - Lệnh gán
    - Lệnh if .... then
    - Lệnh do while

6.3 Các nhóm lệnh 8051

- 6.3.1 Nhóm lệnh chuyển dữ liệu
- 6.3.2 Nhóm lệnh số học
- 6.3.3 Nhóm lệnh logic
- 6.3.4 Nhóm lệnh xử lý bit
- 6.3.5 Nhóm lệnh rẽ nhánh
- 6.3.6 Ví dụ lệnh vòng lặp

6.3.1 Nhóm lệnh chuyển dữ liệu

Mnemonic	Operation	Execution Time (μs) for 12MHz operation
MOV A, <src>	A = <src>	1
MOV <dest>,A	<dest> = A	1
MOV <dest>, <src>	<dest> = <src>	2
MOV DPTR, #data16	DPTR = 16-bit immediate constant	2
PUSH <src>	INC SP : MOV "@SP",<src>	2
POP <dest>	MOV <dest>, "@SP" : DEC SP	2
XCH A,<byte>	ACC and <byte> exchange data	1
XCHD a,@Ri	ACC and @Ri exchange low nibble	1

6.3.1 Truyền dữ liệu với RAM ngoài

Address width	Mnemonic	Operation	Execution Time (μs) for 12MHz operation
8 bits	MOV A,@Ri	Read RAM @Ri	2
8 bits	MOV @Ri,A	Write RAM @ Ri	2
16 bits	MOVX A, @DPTR	Read external RAM @DPTR	2
16 bits	MOVX @DPTR, A	write external RAM @DPTR	2

6.3.2 Lệnh xử lý lôgic

Mnemonic		Operation	Execution Time (μs) for 12MHz operation
ANL	A, <byte>	A = A .AND. <byte>	1
ANL	<byte>,A	<byte> = <byte> .AND. A	1
ANL	<byte>,#data	<byte> = <byte> .AND. #data	2
ORL	A, <byte>	A = A .OR. <byte>	1
ORL	<byte>,A	<byte> = <byte> .OR. A	1
ORL	<byte>,#data	<byte> = <byte> .OR. #data	2
XRL	A, <byte>	A = A .XOR. <byte>	1
XRL	<byte>,A	<byte> = <byte> .XOR. A	1
XRL	<byte>,#data	<byte> = <byte> .XOR. #data	2
CLR	A	A = 00H	1
CPL	A	A = .NOT. A	1

6.3.2 Ví dụ

```
      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
```

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

Lời giải:

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

6.3.2 Ví dụ XOR

```
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
```

6.3.2 Kiểm tra dùng XOR

Đọc và kiểm tra cổng P1 xem nó có chứa giá trị A5H không?
Nếu có gửi FFH đến cổng P2, nếu không xoá nó

```
MOV   P2, #00          ; Xóa P2
MOV   P1, #0FFH        ; Lấy P1 là cổng đầu vào
MOV   R3, #A5H         ; R3 = A5H
MOV   A, P1            ; Đọc P1
XRL   A, R3
JNZ   EXIT             ; Nhảy nếu A có giá trị khác 0
MOV   P2, #0FFH
```

6.3.2 Ví dụ về debounce XOR

```
;-------------------------------------------------
DEBOUNCE:
;-------------------------------------------------
                                ;CHECK CHANGE BY xrl
          mov   B,A             ;hold the input
          xrl   A,CharP         ;exclusive or
          jnz   DEBN1           ;<>0, so a change

                                ;KEEP DEBOUNCED
          mov   CharD,B         ;0=no change,debounc
DEBN1:    mov   CharP,B         ;not debounce
          ret
```

6.3.2 Lấy bù 2

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

Tìm giá trị bù 2 của 85H.
Lời giải:

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

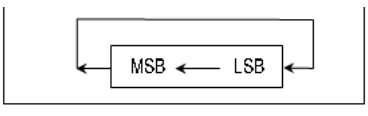
6.3.2 Lệnh xử lý lôgic (tiếp)

CLR A	A = 00H	1
CPL A	A = .NOT. A	1
RL A	Rotate ACC Left 1 bit	1
RLC A	Rotate Left through Carry	1
RR A	Rotate ACC Right 1 bit	1
RRC A	Rotate Right through Carry	1
SWAP A	Swap Nibbles in A	1

6.3.2 Ví dụ

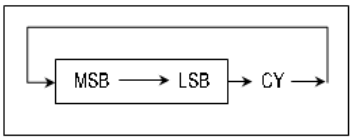
```
MOV   A, #36H     ; A = 0011 0110
RR    A           ; A = 0001 1011
RR    A           ; A = 1000 1101
RR    A           ; A = 1100 0110
RR    A           ; A = 0110 0011
```


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

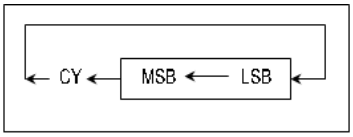


6.3.2 Quay có nhớ

```
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
```




```
SETB  C            ; Make CY = 1
MOV   A #15H       ; A = 0001 0101
RLC   A            ; A = 0010 1011  CY = 0
RLC   A            ; A = 0101 0110  CY = 0
```



6.3.2 Nhân và Chia cơ số 2

```
MOV B,#02h     ;Load B with 2
MUL AB         ;Multiply Accumulator by B (2)
```

```
CLR C          ;Make sure carry bit is initially clear
RLC A          ;Rotate left, multiplying by two
```

6.3.2 Ví dụ

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

```
        MOV   R1, #0          ; Chọn R1 giữ số các số 1
        MOV   R7, #8          ; Đặt bộ đếm = 8 để quay 8 lần
        MOV   A, #97H         ; Tìm 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


        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         ;
```

6.3.2 Ví dụ ASCII - BCD

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

```
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
MOV     R6, A           ; Lưu nó vào R6
MOV     A, R2
ANL     A, #0F0H
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,
MOV     R2, A           ; Lưu ký tự ASCII vào R2
```

6.3.3 Lệnh làm việc với bit

SETB C	C = 1	1
SETB bit	bit = 1	1
CPL C	C = .NOT. C	1
CPL bit	bit = .NOT. bit	1
JC rel	Jump if C = 1	2
JNC rel	Jump if C = 0	2
JB bit,rel	Jump if bit = 1	2
JNB bit,rel	Jump if bit = 0	2
JBC bit,rel	Jump if bit = 1; CLR bit	2

6.3.3 Lệnh làm việc với bit

Mnemonic	Operation	Execution Time (μs) for 12MHz operation
ANL C, bit	C = C .AND. bit	2
ANL C,/bit	C = C .AND. .NOT. bit	2
ORL C, bit	C = C .OR. bit	2
ORL C,/bit	C = C .OR. .NOT. bit	2
MOV C,bit	C = bit	1
MOV bit.C	bit = C	2
CLR C	C = 0	1
CLR bit	bit = 0	1

6.3.3 Ví dụ

```
BACK:
        SETB  P1.3          ;Thiết lập bít 3 cổng 1 lên 1.
        LCALL DELAY         ;Gọi chương trình con DELAY
        LCALL DELAY         ;Gọi chương trình con DELAY lần nữa.
        CLR   P1.3          ;Xoá bít 3 của cổng 1 và 0.
        LCALL DELAY         ;Gọi chương trình con DELAY
        SJMP BACK           ;Tiếp tục thực hiện nó.


HERE:
        CPL   P1.0          ;Bù bít 0 của cổng 1.
        LCALL DELAY         ;Gọi chương trình con giữ chậm DELAY
        SJMP  HERE          ;Tiếp tục thực hiện nó.
```

6.3.3 Ví dụ tiếp

Trạng thái của các bít P1.2 và P1.3 của cổng vào/ra P1 phải được lưu cất trước khi chúng được thay đổi. Hãy viết chương trình để lưu trạng thái của P1.2 vào vị trí bít 06 và trạng thái P1.3 vào vị trí bít 07

```
        CLR     06          ;Xoá địa chỉ bít 06
        CLR     07          ;Xoá địa chỉ bít 07
        JNB     P1.2, OVER  ;Kiểm tra bít P1.2 nhảy về OVER nếu P1.2 = 0
        SETB    06          ; Nếu P1.2 thì thiết lập vị trí bít 06 = 0
OVER:
        JNB     P1.3, NEXT   ;Kiểm tra bít P1.3 nhảy về NEXT nếu nó = 0
        SETB    07          ;Nếu P1.3 = 1thì thiết lập vị trí bít 07 = 1
NEXT:
```

6.3.3 Ví dụ với bit C

Hãy viết một chương trình để hiển thị ("New Message") trên màn hình LCD nếu bít 12H của RAM có giá trị cao. Nếu nó có giá trị thấp thì LCD hiển thị ("No New Message").

```
        MOV  C, 12H          ; Sao trạng thái bít 12H của RAM vào CY
        JNC   NO             ; Kiểm tra xem cờ CY có giá trị cao không.
        MOV  DPTR, # 400H    ; Nếu nó nạp địa chỉ của lời nhắn.
        LCAL DISPLAY         ; Hiển thị lời nhắn.
        SJMP  EXIT           ; Thoát
NO:     MOV  DPTR, #420H     ; Nạp địa chỉ không có lời nhắn.
        LCAL DISPLAY         ; Hiển thị nó.
EXIT:                        ;  Thoát

        ORG   400H
YES-MG: DB    "NEW Message"
        ORG   420H
NO-MG:  DB    "No New Message"
```


6.3.4 Lệnh nhảy có điều kiện

Mnemonic	Operation	Execution Time (μs) for 12MHz operation
JZ rel	Jump if A = o	2
JNZ rel	Jump if A ≠ 0	2
JC rel	Jump if C = 1	2
JNC rel	Jump if C = 0	2

6.3.4 Tính tổng

Hãy tìm tổng của các giá trị 79H, F5H và E2H. Đặt vào trong các thanh ghi R0 (byte thấp) và R5 (byte cao).

```
        MOV   A, #0       ; Xoá thanh ghi A = 0
        MOV   R5, A       ; Xoá R5
        ADD   A #79H      ; Cộng 79H vào A (A = 0 + 79H = 79H)
        JNC   N-1         ; Nếu không có nhớ cộng kế tiếp
        INC   R5          ; Nếu CY = 1, tăng R5

N-1:    ADD   A, #0F5H    ; Cộng F5H vào A (A = 79H + F5H = 6EH)
        JNC   N-2         ; Nhảy nếu CY = 0
        INC   R5          ; Nếu CY = 1 tăng R5 (R5 = 1)
N-2:    ADD   A, #0E2H    ; Cộng E2H vào A (A = 6E + E2 = 50)
        JNC   OVER        ; Nhảy nếu CY = 0
        INC   R5          ; Nếu CY = 1 tăng R5
 OVER:
        MOV   R0, A       ; Bây giờ R0 = 50H và R5 = 02
```

6.3.4 Chuỗi ký tự

```
          ORG       000
          MOV       DPTR, #MYDATA    ; Nạp  con trỏ ROM
          MOV       R0, #40          ; Nạp con trỏ   RAM
BACK:     CLR       A                ; Xoá thanh ghi A(A=0)
          MOVC      A, @A + DPTR     ; Chuyển dữ liệu
          JZ        HERE             ; Thoát ra nếu có ký tự Null
          INC       DPTR             ; Tăng con trỏ ROM
          MOV       @R0, A           ; Cất vào ngăn nhớ của RAM
          INC       R0               ; Tăng con trỏ RAM
          SJM       BACK             ; Lặp lại
HERE:     SJMP      HERE

MYDATA:   DB  "AMER1CA", 0           ;
```

6.3.4 Các lệnh nhảy không điều kiện

Mnemonic	Operation	Execution Time (μs) for 12MHz operation
JMP addr	Jump to addr	2
JMP @A+DPTR	Jump to A+DPTR	2
CALL addr	Call subroutine at addr	2
RET	Return from subroutine	2
RETI	Return from interrupt	2
NOP	No operation	1

6.3.4 Ví dụ JMP

```
            CJNE A,#00h,CHECK1  ;If it's not zero, jump to CHECK1
            AJMP SUB0           ;Go to SUB0 subroutine
CHECK1:     CJNE A,#01h,CHECK2  ;If it's not 1, jump to CHECK2
            AJMP SUB1           ;Go to SUB1 subroutine
CHECK2:
```

```
            RL A                    ;Rotate Accumulator left, multiply by 2
            MOV DPTR,#JUMP_TABLE    ;Load DPTR with address of jump table
            JMP  @A+DPTR            ;Jump to the corresponding address
JUMP_TABLE: AJMP SUB0               ;Jump table entry to SUB0
            AJMP SUB1
```

6.3.4 Lệnh làm việc với bảng dữ liệu

Mnemonic	Operation	Execution Time (μs) for 12MHz operation
MOVC A,@A+DPTR	Read Program memory at (A+DPTR)	2
MOVC A,@A+PC	Read Program memory at (A+PC)	2

MOVC, nghĩa là move constant

6.3.4 Ví dụ bảng tìm kiếm

```
        MOV A,#04h          ;Set Accumulator to offset into the
                            ;table we want to read
        LCALL SUB           ;Call subroutine to read 4th byte of the table
        …
SUB:    MOV DPTR,#2000h     ;Set DPTR to the beginning of the value table
        MOVC A,@A+DPTR      ;Read the 5th byte from the table
        RET                 ;Return from the subroutine
```

```
SUB:    INC A           ;Increment Acc to account for RET instruction
        MOVC A,@A+PC    ;Get the data from the table
        RET             ;Return from subroutine
        DB 01h,02h,03h,04h,05h     ;The actual data table
```

6.3.4 Lệnh nhảy với cờ

Mnemonic	Operation	Execution Time (μs) for 12MHz operation
DJNZ <byte>,rel	Decrement and jump if not zero	2
CJNE A,<byte>,rel	Jump if A ≠ <byte>	2
CJNE <byte>,#data,rel	Jump if <byte> ≠ #data	2

6.3.5 Lưu ý với lệnh CJNE

- Nếu tham số 1 < tham số 2, bit nhớ C được đặt lên 1
- Nếu tham số 1 > tham số 2, bit nhớ C được xóa về 0

6.3.5 Ví dụ

```
        MOV R0,#08h   ;Set number of loop cycles to 8
LOOP:   INC A         ;Increment acc (or do whatever the loop does)
        DJNZ R0,LOOP  ;Decrement R0, loop back to LOOP if R0 is not 0
        DEC A         ;Decrement accumulator (or whatever you want to do)
```

```
            CJNE A,#30h,CHECK2  ;If A is not 30h, jump to CHECK2 label
            LCALL PROC_A        ;If A is 30h, call the PROC_A subroutine
            SJMP CONTINUE       ;When we get back, we jump to CONTINUE label
CHECK2:     CJNE A,#42h,CHECK3  ;If A is not 42h, jump to CHECK3 label
            LCALL CHECK_LCD     ;If A is 42h, call the CHECK_LCD subroutine
            SJMP CONTINUE       ;When we get back, we jump to CONTINUE label
CHECK3:     CJNE A,#50h,CONTINUE;If A is not 50h, we jump to CONTINUE label
            LCALL DEBOUNCE_KEY  ;If A is 50h, call the DEBOUNCE_KEY subroutine
CONTINUE:   {Code continues here};The rest of the program continues here
```

6.3.5 Ví dụ lệnh if

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

```
        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 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:         ...
```

6.3.5 Phát biểu While/do

- while [condition] Do
- các lệnh chương trình
- do
- các lệnh chương trình
- while (...)

6.3.5 Ví dụ tính tổng

- Tính tổng dãy số
- Chiều dài của dãy số chứa trong thanh ghi R7
- Địa chỉ bắt đầu dãy số trong thanh ghi R0

6.3.5 Ví dụ

- [sum] = 0
- WHILE (length > 0) Do
  - [sum = sum + @pointer]
  - [pointer = pointer + 1]
  - [length = length – 1]
- End

6.3.5 Ví dụ

```
                    MOV R7, #05H
                    MOV DPTR, #TABLE

                    MOV R0, #00H
SUM:                CLR A
LOOP:
                    CJNE R7, #0, STATEMENT
                    JMP EXIT
STATEMENT:
                    MOV R1,A
                    MOV A,R0
                    MOVC A, @A+DPTR
                    ADD A, R1
                    INC R0
                    DEC R7
                    JMP LOOP
EXIT:
                    RET

TABLE:  DB 01, 02, 03, 04, 05
```

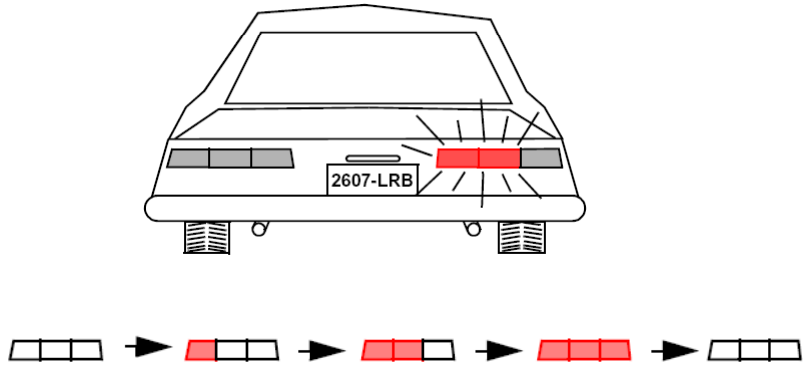
6.3.5 Phát biểu case

```
case [ expression] of
   0: do0
   1: do1
   .....
   n: don
   default:  do_default
End_case
```

6.3.5 Ví dụ case (tiếp)

```
            Call InChar
            CJNE A, #'0', Skip1
Act0:       .................
            JMP Exit
Skip1:      CJNE A, # '1', Skip2
Act1:       .................
            JMP   Exit
.....................................
Exit:
            DoN
```

6.3.5 Ví dụ T-bird light system



6.3.5 Yêu cầu

- Brake bật 6 LEDs.
- Dừng tắt 6 LEDs.
- Rẽ trái thì bật 3 LED trái tuần tự
- Rẽ phải thì bật 3 LED phải tuần tự

6.3.6 Lệnh số học

Mnemonic	Operation	Execution Time (µs) for 12MHz operation
ADD A, <byte>	A = A + <byte>	1
ADDC A, <byte>	A = A + <byte> + C	1
SUBB A, <byte>	A = A - <byte> - C	1
INC A	A = A + 1	1
INC <byte>	<byte> = <byte> + 1	1
INC DPTR	DPTR = DPTR + 1	2
DEC A	A = A -1	1
DEC <byte>	<byte> = <byte> - 1	1
MUL AB	B:A = B X A	4
DIV AB	A = Int [A/B] B = Mod [A/B]	4
DA A	Decimal adjust	1

6.3.6 Ví dụ lệnh cộng

Cộng 2 số 16-bit tại RAM 30h (high byte) và 31h (low byte), cộng với 1045h và lưu tại32h (high byte) and 33h (lowbyte)

```
MOV A,31h
ADD A,#45h
MOV 33h,A
MOV A,30h
ADDC A,#10h
MOV 32h,A
```

6.3.6 Sự khác nhau cơ bản giữa ADD và ADDC

- Lệnh ADD tương đương với lệnh
  - CLR C
  - ADDC
- Cờ OV xác định số có phải là số âm không
  - Ví dụ: 20h + 70h = 90 h (tương đương -10h)

6.3.6 Ví dụ về MUL, DIV

```
MOV A,#20h    ;Load Accumulator with 20h
MOV B,#75h    ;Load B with 75h
MUL AB        ;Multiply A by B
```

```
MOV A,#0F3h   ;Load Accumulator with F3h
MOV B,#13h    ;Load B with 13h
DIV AB        ;Divide A by B
```

6.3.6 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

```
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)
```

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