BÀI 4: TẬP LỆNH TRONG VI ĐIỀU KHIỂN

Mã bài: MĐ ĐTCN 21-04

Giới thiệu:

Với kiến thức về cấu trúc bên trong của vi điều khiển và để điều khiển hoạt động khối mạch điện giao tiếp phức tạp của hệ thống. Các khối bao gồm bộ nhớ để chứa dữ liệu và chương trình thực hiện, các mạch điện giao tiếp ngoại vi để xuất nhập và điều khiển trở lại, các khối này cùng liên kết với vi điều khiển thì mới thực hiện được công việc. Để kết nối các khối này đòi hỏi người thiết kế phải hiểu biết tinh tường về các thành phần vi điều khiển, bộ nhớ, các thiết bị ngoại vi. Sau khi kết nối vi điều khiển với các thiết bị ngoại vi, ta phải tìm hiểu về các tập lệnh của MSC-51 để điều khiển hoạt động hệ thống.

Bài này giới thiệu cách thức lập trình trên MCS-51 cũng như giải thích hoạt động của các lệnh sử dụng cho họ MCS-51

Mục tiêu:

- Phân biệt được các kiểu định địa chỉ và dữ liệu;
- Trình bày được đặc tính và công dụng của từng lệnh trong 8051;
- Xác định được độ lớn và thời gian thực hiện chương trình;
- Kết hợp được các lệnh riêng lẻ để thực hiện thao tác cho trước đúng kỹ thuật;
- Chủ động, sáng tạo và an toàn trong quá trình học tập.

Nội dung chính:

1. Mở đầu

1.1.Cú pháp lệnh

Một lệnh trong chương trình hợp ngữ có dạng như sau:

Nhãn	Lệnh	Toán hạng	Chú thích
A:	MOV	A, #10h	; Đưa giá trị 10h vào thanh ghi
LED	EQU	30h	; Định nghĩa ô nhớ chứa
On_Led	BIT	00h	; Cờ trạng thái led

Trường nhãn định nghĩa các ký hiệu (có thể là địa chỉ trong chương trình, các hằng dữ liệu, tên đoạn hay các cấu trúc lập trình). Trường nhãn không bắt đầu bằng số và không trùng với các từ khóa có sẵn.

Trường lệnh chứa các từ gợi nhớ cho các lệnh của MCS-51 hay các lệnh giả dùng cho chương trình dịch.

Trường toán hạng chứa các thông số liên quan đến lệnh đang sử dụng.

Trường chú thích dùng để ghi chú trong chương trình hợp ngữ.

Trường này phải được bắt đầu bằng dấu; và chương trình dịch sẽ bỏ qua các từ đặt sau dấu ;.

Lưu ý rằng các chương trình dịch không phân biệt chữ hoa và chữ thường.

1.2.Khai báo dữ liệu

Khi khai báo hằng số, chữ h cuối cùng xác định hằng số là số thập lục phân; chữ b cuối cùng xác định số nhị phân và chữ d cuối (hay không có) xác định số thập phân. Lưu ý rằng đối với số thập lục phân, khi bắt đầu bằng chữ A → F thì phải thêm số 0 vào phía trước.

Ví dụ:

1010b; Số nhị phân

1010h; Số thập lục phân

1010; Số thập phân

0F0h ; Số thập lục phân nhưng bắt đầu bằng chữ F nên phải thêm vào phía trước số 0.

Khi dùng dấu # phía trước một con số, đó chính là dữ liệu tức thời cònnếu không dùng dấu # thì đó là địa chỉ của ô nhớ. Lưu ý rằng khi dùng RAM nội thì chỉ dùng địa chỉ từ 00 – 7Fh còn vùng địa chỉ từ 80h – 0FFh dùng cho các thanh ghi chức năng đặc biệt. Đối với họ 89x52, RAM nội có 256 byte thì các byte địa chỉ cao (từ 80h – 0FFh) không thể truy xuất trực tiếp mà phải truy xuất gián tiếp.

Ví dụ:

```
MOV A,30h  ; Chuyển nội dung ô nhớ  30h vào A
MOV A,#30h       ; Chuyển giá trị 30h vào A
MOV A,80h  ; Chuyển nội dung Port 0 vào A
MOV R0,#80h      ; Chuyển nội dung ô nhớ  80h vào A (chỉ
MOV A,@R0        ; dùng cho họ  89x52)
```

Để định nghĩa trước một vùng nhớ trong bộ nhớ chương trình, có thể dùng các chỉ dẫn DB (define byte – định nghĩa 1 byte) hay DW (define word – định nghĩa 2 byte).

Ví dụ: Định nghĩa trước dữ liệu cho led như sau:

```
Led:DB      01h,02h,04h,08h,10h,20h,40h,80h
```

Đoạn chương trình này xác định tại nhãn Led có chứa các giá trị lần lượt từ 01h đến 80h. Nếu nhãn Led đặt tại địa chỉ 100h thì giá trị tương ứng như sau:

Địa chỉ	Giá trị
100h	01h
101h	02h
102h	04h
103h	08h
104h	10h
105h	20h
106h	40h
107h	80h

Bảng 4.1 Mô tả dữ liệu tại vùng nhớ từ 100h đến 107h

Để dễ nhớ và dễ hiểu khi lập trình, các chương trình dịch cho phép dùng các ký tự thay thế cho các ô nhớ bằng các lệnh giả EQU, BIT.

Ví dụ:

```
LED    EQU   30h
ON_LED      BIT     00h
```

Giả sử chương trình hợp ngữ có các lệnh sau:

```
MOV  A,LED
SETBON_LED
```

Khi biên dịch, chương trình dịch sẽ tự động chuyển thành dạng lệnh sau:

```
MOV  A,30h
SETB 00h
```

Các ký hiệu cần chú ý:

Rn : các thanh ghi từ R0 – R7 (bank thanh ghi hiện hành).

Ri : các thanh ghi từ R0 – R1 (bank thanh ghi hiện hành).

@Rn : định địa chỉ gián tiếp 8 bit dùng thanh ghi Rn.

@DPTR : định địa chỉ gián tiếp 16 bit dùng thanh ghi DPTR.

direct : định địa chỉ trực tiếp RAM nội (00h – 7Fh) hay SFR (80h – FFh) (direct).

: nội dung của bộ nhớ tại địa chỉ direct

#data8 : giá trị tức thời 8 bit

#data16 : giá trị tức thời 16 bit

bit : địa chỉ bit của các ô nhớ có thể định địa chỉ bit (00h – 7Fh đối với địa chỉ bit và 20h – 2Fh đối với địa chỉ byte).

2. Các cách định địa chỉ

Các kiểu định địa chỉ cho phép định rõ nơi lấy dữ liệu hoặc nơi nhận dữ liệu tùy thuộc vào cách thức sử dụng lệnh của người lập trình. Vi điều khiển 8051 có 8 kiểu định địa chỉ như sau:

- Kiểu định địa chỉ dùng thanh ghi.
- Kiểu định địa chỉ trực tiếp
- Kiểu định địa chỉ gián tiếp.
- Kiểu định địa chỉ tức thời.
- Kiểu định địa chỉ tương đối.
- Kiểu định địa chỉ tuyệt đối.
- Kiểu định địa chỉ dài.
- Kiểu định địa chỉ chỉ số.

2.1. Định địa chỉ bằng thanh ghi

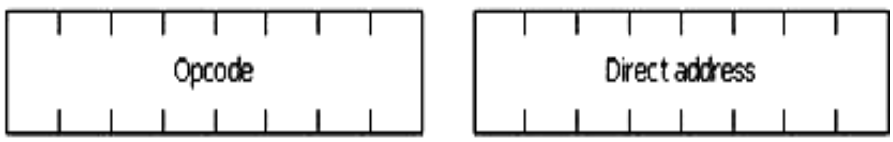
Hình 4.1. Định địa chỉ thanh ghi

Các thanh ghi từ R0 – R7 có thể truy xuất bằng cách định địa chỉ trực tiếp hay gián tiếp như trên. Ngoài ra, các thanh ghi này còn có thể truy xuất bằng cách dùng 3 bit trong mã lệnh để chọn 1 trong 8 thanh ghi (8 thanh ghi này có địa chỉ trực tiếp thay đổi tuỳ theo bank thanh ghi đang sử dụng).

Các lệnh sử dụng kiểu định địa chỉ thanh ghi được mã hóa bằng các dùng 3 bit thấp nhất của opcode (của lệnh) để chỉ ra 1 thanh ghi bên trong không gian địa chỉ logic này. Vậy : 1 mã chức năng + địa chỉ toán hạng → 1 lệnh ngắn 1 byte.

Kiểu này thường được dùng cho các lệnh xử lý dữ liệu mà dữ liệu luôn lưu trong các thanh ghi. Đối với vi điều khiển thì mã lệnh thuộc kiểu này chỉ có 1 byte.

2.2. Định địa chỉ trực tiếp



Hình 4.2. Định địa chỉ trực tiếp

Định địa chỉ trực tiếp (hình 3.2) chỉ dùng cho các thanh ghi chức năng đặc biệt và RAM nội của 8951. Giá trị địa chỉ trực tiếp 8 bit được thêm vào phía sau mã lệnh. Nếu địa chỉ trực tiếp từ 00h – 7Fh thì đó là RAM nội của 8951 (128 byte), còn địa chỉ từ 80h – FFh là địa chỉ các thanh ghi chức năng đặc biệt.

Các lệnh sau có kiểu định địa chỉ trực tiếp:

MOV A, P0

MOV A, 30h

Trong 8051 có 128 byte bộ nhớ RAM. Bộ nhớ RAM được gán địa chỉ từ 00H đến FFH và được phân chia như sau:

Các ngăn nhớ từ 00H đến 1FH được gán cho các băng thanh ghi và ngăn xếp.

Các ngăn nhớ từ 20H đến 2FH được dành cho không gian định địa chỉ bít để lưu dữ liệu theo từng bit.

Các ngăn nhớ từ 30H đến 7FH là không gian để lưu dữ liệu có kích thước 1 byte.

Chế độ định địa chỉ trực tiếp có thể truy cập toàn bộ không gian của bộ nhớ RAM. Tuy nhiên, chế độ này thường được dùng để truy cập các ngăn nhớ RAM từ 30H đến 7FH, vì thực tế đối với không gian nhớ danh cho băng thanh ghi thì đã được truy cập bằng tên thanh ghi như R0- R7. ở chế độ định địa chỉ trực tiếp, địa chỉ ngăn nhớ RAM chứa dữ liệu là toán hạng của lệnh.

Ví dụ:

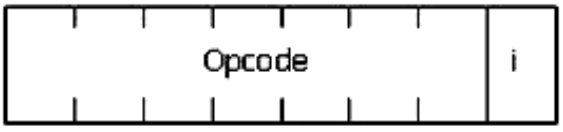
```
MOV  R0,  40      ; sao nội dung ngăn nhớ 40H của RAM vào R0
MOV  R4,  7FH     ; chuyển nội dung ngăn nhớ 7FH vào R4.
```

Một ứng dụng quan trọng của chế độ định địa chỉ trực tiếp là ngăn xếp. Trong họ 8051, chỉ có chế độ định địa chỉ trực tiếp là được phép cất và lấy dữ liệu từ ngăn xếp.

Lệnh đầu tiên chuyển nội dung từ Port 0 vào thanh ghi A. Khi biên dịch, chương trình sẽ thay thế từ gợi nhớ P0 bằng địa chỉ trực tiếp của Port 0 (80h) và đưa vào byte 2 của mã lệnh. Lệnh thứ hai chuyển nội dung của RAM nội có địa chỉ 30h vào thanh ghi A.

2.3. Định địa chỉ gián tiếp (Indirect Addressing)

Ở chế độ này, thanh ghi được dùng để trỏ đến dữ liệu có trong bộ nhớ.



Hình 4.3. Định địa chỉ gián tiếp

Nếu dữ liệu có trên chip CPU thì chỉ các thanh ghi R0 và R1 mới được sử dụng, và như vậy cũng có nghĩa là không thể dùng các thanh ghi R2-R7 để trỏ đến địa chỉ của toán hạng ở chế độ định địa chỉ này.

Ví dụ:

```
MOV   A,   @R0      ; chuyển ngăn nhớ RAM có địa chỉ ở R0 vào A
MOV   @R1,  B       ; chuyển B vào ngăn nhớ RAM có địa chỉ ở R1
```

Chú ý:

Kiểu định địa chỉ gián tiếp được tượng trưng bởi ký hiệu @,được đặt trước các thanh ghi R0, R1,SP cho địa chỉ 8 bit (không sử dụng các thanh ghi R2 – R7 trong chế độ địa chỉ này) hay DPTR cho địa chỉ 16 bit. R0 và R1 có thể hoạt động như một thanh ghi con trỏ, nội dung của nó cho biết địa chỉ của một ô nhớ trong RAM nội mà dữ liệu sẽ ghi hoặc sẽ đọc. Còn DPTR dùng để truy xuất ô nhớ ngoại. Các lệnh thuộc dạng này chỉ có 1 byte. Tuy nhiên R0 và R1 là các thanh ghi 8 bit, nên chúng chỉ được phép truy cập đến các ngăn nhớ RAM trong, từ địa chỉ 30H đến 7FH và các thanh ghi SFR. Trong thực tế, có nhiều trường hợp cần truy cập dữ liệu được cất ở RAM ngoài hoặc không gian ROM trên chip. Trong những trường hợp đó chúng ta cần sử dụng thanh ghi 16 bit DPTR.

Ví dụ:

```
MOV A, @R1          ;copy noi dung o nho co dia chi dat
                    ;trong thanh ghi R1 vao thanh ghi A
```

2.4. Định địa chỉ tức thời (Immediate Addressing)

Khi toán hạng là một hằng số thay vì là một biến, hằng số này có thể đưa vào lệnh và đây là byte dữ liệu tức thời.

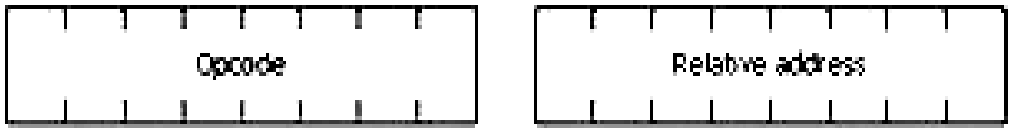
Trong hợp ngữ, các tồn hạng tức thời được nhận biết nhờ vào ký tự ‘#‘ đặt trước chúng. Toán hạng này có thể là một hằng số học, một biến hoặc một biểu thức số học sử dụng các hằng số, các ký hiệu và các toán tử. Trình dịch hợp ngữ tính giá trị và thay thế dữ liệu tức thời vào trong lệnh. Lệnh này thường dùng để nạp 1 giá trị là 1 hằng số ở byte thứ 2 (hoặc byte thứ 3) vào thanh ghi hoặc ô nhớ .

Ví dụ:

```
MOV  A,  #12  ;Nạp giá trị 12(OCH) vào thanh ghi A
MOV A, #30H         ;nap du lieu 30H vao thanh ghi A
```

Tất cả các lệnh sử dụng kiểu định địa chỉ tức thời đều sử dụng hằng dữ liệu 8 bit làm dữ liệu tức thời. Có một ngoại lệ khi ta khởi động con trỏ dữ liệu 16-bit DPTR, hằng địa chỉ 16 bit được cần đến.

2.5. Định địa chỉ tương đối



Hình 4.4. Định địa chỉ tương đối

Kiểu định địa chỉ tương đối chỉ sử dụng với những lệnh nhảy. Nơi nhảy đến có địa chỉ bằng địa chỉ đang lưu trong thanh ghi PC cộng với 1 giá trị 8 bit [còn gọi là giá trị lệch tương đối: relative offset] có giá trị từ – 128 đến +127 nên vi điều khiển có thể nhảy lùi [nếu số cộng với số âm] và nhảy tới [nếu số cộng với số dương]. Lệnh này có mã lệnh 2 byte, byte thứ 2 chính là giá trị lệch tương đối:

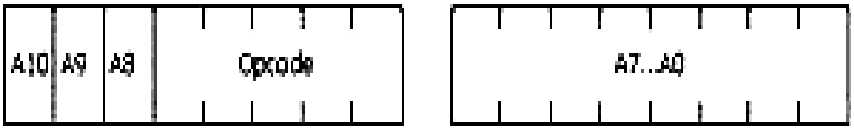
Nơi nhảy đến thường được xác định bởi nhãn (label) và trình biên dịch sẽ tính toán giá trị lệch.

Định vị tương đối có ưu điểm là mã lệnh cố định, nhưng khuyết điểm là chỉ nhảy ngắn trong phạm vi -128÷127 byte [256byte], nếu nơi nhảy đến xa hơn thì lệnh này không đáp ứng được sẽ có lỗi.

Ví dụ:

```
SJMP   X1     ;nhay den nhan co tên X1 nằm trong
               ;tam vuc 256 byte
```

2.6. Định địa chỉ tuyệt đối

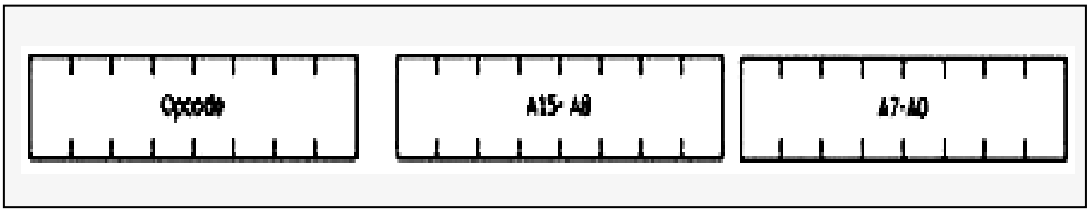


Hình 4.5. Định địa chỉ tuyệt đối

Kiểu định địa chỉ tuyệt đối (hình 4.5) được dùng với các lệnh ACALL và AJMP. Các lệnh này có mã lệnh 2 byte. Định địa chỉ tuyệt đối có ưu điểm là mã lệnh ngắn (2 byte), nhưng khuyết điểm là mã lệnh thay đổi và giới hạn phạm vi nơi nhảy đến, gọi đến không.

Ví dụ: AJMP X1 ;nhay den nhan co ten X1 nam trong tam vuc 2Kbyte

2.7. Định địa chỉ dài (Long Addressing)



Hình 4.6. Định địa chỉ dài

Kiểu định địa chỉ dài được dùng với lệnh LCALL và LJMP. Các lệnh này có mã lệnh 3 byte – trong đó có 2 byte (16bit) là địa chỉ của nơi đến. Cấu trúc mã lệnh là 3 byte chứa địa chỉ đích 16 bit. Định địa chỉ dài là có thể gọi 1 chương trình con hoặc có thể nhảy đến bất kỳ vùng nhớ nào vùng nhớ 64KB. Lợi ích của kiểu định địa chỉ này là sử dụng hết toàn bộ không gian nhớ chương trình 64K, nhưng lại có điểm bất lợi là lệnh dài đến 3-byte và phụ thuộc vào vị trí.

Ví dụ:

```
LJMP   X1     ;nhay den nhan co ten X1 nam trong
               ;tam vuc 64Kbyte
```

2.8. Định địa chỉ chỉ số (Index Addressing)

Chế độ định địa chỉ chỉ số được sử dụng rộng rãi khi truy cập các phần tử dữ liệu của bảng trong không gian ROM chương trình của 8051. Kiểu định địa chỉ chỉ số "dùng một thanh ghi cơ bản: là bộ đếm chương trình PC hoặc bộ đếm dữ liệu DPTR" kết hợp với "một giá trị lệch (offset) còn gọi là giá trị tương đối [thường lưu trong thanh ghi]" để tạo ra 1 địa chỉ của ô nhớ cần truy xuất hoặc là địa chỉ của nơi nhảy đến. Việc kết hợp được minh họa như sau:

Base Registr		Offset		Effective Address
PC (or PDTR)	+	A	=	

Ví dụ:

```
MOVC A, @A + DPTR      ;lay du lieu trong o nho
                       ;DPTR+A de nap vao thanh ghi A
```

Ở lệnh này, nội dung của A được cộng với nội dung thanh ghi 16- bit DPTR để tạo ra địa chỉ 1.

3. Các nhóm lệnh

Tùy thuộc vào cách và chức năng của mỗi lệnh, có thể chia ra thành 5 nhóm lệnh như sau:

- Nhóm lệnh số học
- Nhóm lệnh logic
- Nhóm lệnh vận truyền dữ liệu
- Nhóm lệnh Boolean (thao tác bit)
- Nhóm lệnh rẽ nhánh chương trình

Cấu trúc chung của mỗi lệnh:

Mã_lệnh Toán_hạng1, Toán_hạng2, Toán_hạng3

Trong đó:

Mã_lệnh: Tên gợi nhớ cho chức năng của lệnh. (VD như add cho addition)

- Toán_hạng1, Toán_hạng2, Toán_hạng3: Là các toán hạnh của lệnh, tùy thuộc vào mỗi lệnh số toán hạng có thể không có, có 1, 2 hoặc 3.
- Ví dụ:
- RET (Kết thúc chương trình con). Lệnh này không có toán hạng
- JZ TEMP (Chuyển con trỏ chương trình đến vị trí TEMP). Chỉ có 1 toán hạng.
- ADD A, R3 ; (A = A + R3) Có 2 toán hạng.
- CJNE A, #20, LOOP. (So sánh A với 20, nếu không bằng thì chuyển con trỏ chương trình đến nhãn LOOP). Có 3 toán hạng.

Cần nắm rõ phần cứng, đặc biệt là vùng nhớ Ram của vi điều khiển. Chú ý các thuật ngữ sau:

Các byte RAM 8 bit của vi điều khiển được gọi là "ô nhớ ", nếu các ô nhớ có chức năng đặc biệt thường được gọi là "thanh ghi", nếu là bit thì được gọi là "bit nhớ". Dữ liệu của một ô nhớ là trạng thái (0 hoặc 1) cần thiết lập cho các bit của ô nhớ (8 bit).

Ký hiệu	Mô tả
A:	Thanh ghi chứa (Accumulator).
B:	Thanh ghi B.
Ri:	Thanh ghi R0 hoặc R1 của bất kỳ băng thanh ghi nào trong 4 băng thanh ghi trong RAM.

Rn:	Rn: bất kỳ thanh ghi nào của bất kỳ băng thanh ghi nào trong 4 băng thanh ghi trong RAM.
Dptr:	Thanh ghi con trỏ dữ liệu (có độ rộng 16bit được kết hợp từ 2 thanh ghi 8 bit là DPH và DPL).
Direct:	Direct: là một biến 8 bit(hay chính là ô nhớ) bất kỳ trong RAM (trừ 32 thanh ghi Rn ở đầu RAM).
#data:	Một hằng số 8 bit bất kỳ.
#data16:	Một hằng số 16 bit bất kỳ.
<rel>:	Địa chỉ bất kỳ nằm trong khoảng [PC-128; PC+127]
<addr11>:	Địa chỉ bất kỳ nằm trong khoảng 0 – 2Kbyte tính từ địa chỉ của lệnh tiếp theo.
<addr16>:	Địa chỉ bất kỳ trong không gian 64K (áp dụng cho cả không gian nhớ chương trình và không gian nhớ dữ liệu).
<bit>:	Bit bất kỳ có thể đánh địa chỉ được (không dùng cho các bit không đánh được địa chỉ).

Bảng 4.2: Các ký hiệu sử dụng mô tả lệnh

3.1. Nhóm lệnh số học

Cờ nhớ C:

C=1 nếu phép toán cộng xảy ra tràn hoặc phép trừ có mượn.

C=0 nếu phép toán cộng không tràn hoặc phép trừ không có mượn.

Phép cộng xảy ra tràn là phép cộng mà kết quả lớn hơn 255 (hay FFH hay 11111111b), lúc này C=1

Ví dụ:

Phép cộng không tràn

Số cộng	38H	56	0 0 1 1 1 0 0 0 b
Số cộng	+3AH	58	0 0 1 1 1 0 1 0 b
Kết quả	72H	114	0 1 1 1 0 0 1 0 b
Cờ nhớ C	0		0

Phép cộng tràn

Số cộng	6CH	108	0 1 1 0 1 1 0 0 b
Số cộng	+9FH	159	1 0 0 1 1 1 1 1 b
Kết quả	10BH	267	1 0 0 0 0 1 0 1 1 b
Cờ nhớ C	1		1

Phần bên trái là 8 bit của thanh ghi A sau khi kết quả được thực hiện, phần màu đỏ trong kết quả là giá trị bị tràn, giá trị này không lưu ở thanh ghi A mà lưu ở thanh ghi PSW, tại cờ C.

Số bị trừ	9FH	159	1 0 0 1 1 1 1 1 b
Số trừ	- 6CH	108	0 1 1 0 1 1 0 0 b
Kết quả	33H	51	0 0 1 1 0 0 1 1 b
Cờ nhớ C 0		0	

Số bị trừ	6CH	108	0 1 1 0 1 1 0 0 b
Số trừ	- 9FH	159	1 0 0 1 1 1 1 1 b
Kết quả	CDH	-51	1 1 0 0 1 1 0 1 b
Cờ nhớ C 1		1	

→ phép trừ trên có số muợn

3.1.1. Lệnh cộng dữ liệu trên thanh ghi A với dữ liệu trên thanh ghi Rn:

Cú pháp: Add A,Rn

Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte.

Thời gian thực hiện: 1 chu kì máy.

Công dụng: Cộng giá trị dữ liệu trên thanh ghi A với giá trị dữ liệu trên thanh ghi Rn, sau khi thực hiện lệnh kết quả được lưu ở thanh ghi A. Lệnh này có ảnh hưởng đến thanh thanh trạng thái PSW.

A ← (A) + (Rn)

Ví dụ1:
```
Mov A,#20H
Mov R1,#08H
Add A,R1
```

Kết quả : A có giá trị là 28H

R1 vẫn giữ nguyên giá trị là 08H

Cờ C = 0

Ví dụ 2:

```
Mov     A,#0E9H
Mov     R6,#0BAH
Add     A,R6
```

Kết quả : A = #0A3h

R6 = #0BAh

Cờ C = 1

3.1.2. Lệnh cộng dữ liệu trên thanh ghi A với dữ liệu ở ô nhớ có địa chỉ direct:

Cú pháp: Add A,direct

Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte.

Thời gian thực hiện: 1 chu kì máy.

Công dụng: Cộng giá trị dữ liệu trên thanh ghi A với giá trị dữ liệu trên ô nhớ có địa chỉ direct, sau khi thực hiện lệnh kết quả được lưu ở thanh ghi A. Lệnh này có ảnh hưởng đến thanh thanh trạng thái PSW.

A ← (A) + (direct)

Ví dụ:

```
Mov     50h,#20H
Mov     A,#0E8H
Add     A,50H
```

Kết quả : A = #08H

50H = #20H

C = 1

3.1.3. Lệnh cộng dữ liệu trên thanh ghi A với dữ liệu của ô nhớ có địa chỉ gián tiếp:

Cú pháp: Add A,@Ri

Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte.

Thời gian thực hiện: 1 chu kì máy.

Công dụng: Cộng giá trị dữ liệu trên thanh ghi A với giá trị dữ liệu của ô nhớ có địa chỉ bằng giá trị của thanh ghi Ri, sau khi thực hiện lệnh kết quả được lưu ở thanh ghi A. Lệnh này có ảnh hưởng đến thanh thanh trạng thái PSW.

A ← (A) + (Ri)

Ví dụ:

AC = 1 ; cờ C đang mang giá trị 1

```
Mov     50H,#60H
Mov     R2,#50H
Mov     A,#01H
Add     A,@R2
```

Kết quả : A = #61H

R2 = #50H

C = 0 ; cờ C mang giá trị 0

3.1.4. Lệnh cộng dữ liệu trên thanh ghi A với dữ liệu xác định:

Cú pháp: Add A,#data

Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte.

Thời gian thực hiện: 1 chu kì máy.

Công dụng: Cộng giá trị dữ liệu trên thanh ghi A với một giá trị xác định, sau khi thực hiện lệnh kết quả được lưu ở thanh ghi A. Lệnh này có ảnh hưởng đến thanh thanh trạng thái PSW.

A ⟵ (A) + # data

Ví dụ:

```
Mov     A,#05h
Add     A,#06h
```

Kết quả : A = #0Bh

C = 0

3.1.5. Lệnh cộng dữ liệu trên thanh ghi A với dữ liệu trên thanh ghi Rn có số nhớ ở cờ C:

Cú pháp: AddC A,Rn

Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte.

Thời gian thực hiện: 1 chu kì máy.

Công dụng: Cộng giá trị dữ liệu trên thanh ghi A với giá trị dữ liệu trên thanh ghi Rn và cộng thêm giá trị của số nhớ trên cờ C, sau khi thực hiện lệnh kết quả được lưu ở thanh ghi A. Lệnh này có ảnh hưởng đến thanh thanh trạng thái PSW.

A ⟵ (A) + (C) + (Rn)

Ví dụ: C = 1

```
Mov     A,#08h
Mov     R1,#10h
Addc    A,R1
```

Kết quả : A = #19h ;cộng cả cờ C

R1 = #10h

C =0

3.1.6. Lệnh cộng dữ liệu trên thanh ghi A với dữ liệu ở ô nhớ có địa chỉ direct và giá trị số nhớ ở cờ C.

Cú pháp: AddC A,direct

Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte.

Thời gian thực hiện: 1 chu kì máy.

Công dụng: Cộng giá trị dữ liệu trên thanh ghi A với giá trị dữ liệu của ô nhớ có địa chỉ direct và cộng thêm giá trị của số nhớ trên cờ C, sau khi thực hiện lệnh kết quả được lưu ở thanh ghi A. Lệnh này có ảnh hưởng đến thanh thanh trạng thái PSW.

A ⟵ (A) + (C) + (Rn).

Ví dụ: C = 0

```
Mov   A,#0A5h
Mov   10h,#96h
Addc  A,10h
```

Kết quả : A = #3Bh

10h = #96h

C =1

3.1.7. Lệnh cộng dữ liệu trên thanh ghi A với dữ liệu của ô nhớ có địa chỉ gián tiếp và số nhớ ở cờ C.

Cú pháp: AddC A,@Ri

Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte.

Thời gian thực hiện: 1 chu kì máy.

Công dụng: Cộng giá trị dữ liệu trên thanh ghi A với giá trị dữ liệu của ô nhớ có địa chỉ bằng giá trị của thanh ghi Ri và cộng thêm giá trị của số nhớ trên cờ C, sau khi thực hiện lệnh kết quả được lưu ở thanh ghi A. Lệnh này có ảnh hưởng đến thanh thanh trạng thái PSW.

Ví dụ:

```
C = 1
Mov   A,#05h
Mov   50h,#10h
Mov   R2,#50h
Addc  a,@R2
```

Kết quả : A = #16h

C = 0

3.1.8. Lệnh cộng dữ liệu trên thanh ghi A với dữ liệu xác định và số nhớ ở cờ C.

Cú pháp: AddC A,#data

Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte.

Thời gian thực hiện: 1 chu kì máy.

Công dụng: Cộng giá trị dữ liệu trên thanh ghi A với giá trị xác định và cộng thêm giá trị của số nhớ trên cờ C, sau khi thực hiện lệnh kết quả được lưu ở thanh ghi A. Lệnh này có ảnh hưởng đến thanh thanh trạng thái PSW.

A ⟵ (A) + (C) + #data

Ví dụ: C = 1

```
Mov   A,#05h
Addc  A,#16h
```

Kết quả : A = #1Ch

C = 0

3.1.9. Lệnh trừ dữ liệu trên thanh ghi A với dữ liệu trên thanh ghi Rn và số nhớ ở cờ C.

Cú pháp: SUBB A,Rn

Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte.

Thời gian thực hiện: 1 chu kì máy.

Công dụng: Trừ giá trị dữ liệu trên thanh ghi A với giá trị dữ liệu trên thanh ghi Rn và trừ cho giá trị nhớ trên cờ C, sau khi thực hiện lệnh kết quả được lưu ở thanh ghi A. Lệnh này có ảnh hưởng đến thanh thanh trạng thái PSW.

A ⟵ (A) - (C) - (Rn)

Ví dụ: C = 1

Mov A,#0E5h

Mov R3,#9Fh

Subb A,R3

kết quả : A = 45h

C = 0

3.1.10. Lệnh trừ dữ liệu trên thanh ghi A với dữ liệu ở ô nhớ có địa chỉ direct và số nhớ ở cờ C.

Cú pháp: SUBB A,direct

Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte.

Thời gian thực hiện: 1 chu kì máy.

Công dụng: Trừ giá trị dữ liệu trên thanh ghi A với giá trị dữ liệu của ô nhớ có địa chỉ direct và trừ cho giá trị nhớ trên cờ C, sau khi thực hiện lệnh kết quả được lưu ở thanh ghi A. Lệnh này có ảnh hưởng đến thanh trạng thái PSW.

A ⟵ (A) - (C) - (direct)

Ví dụ:

C = 0

Mov A,#0E5h

Mov 05h,#9Fh

Subb A,05h

Kết quả : A = 46h

C = 0

3.1.11. Lệnh trừ dữ liệu trên thanh ghi A với dữ liệu của ô nhớ có địa chỉ gián tiếp và số nhớ ở cờ C.

Cú pháp: SUBB A,@Ri

Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte.

Thời gian thực hiện: 1 chu kì máy.

Công dụng: Trừ giá trị dữ liệu trên thanh ghi A với giá trị dữ liệu của ô nhớ có địa chỉ bằng giá trị của thanh ghi Ri và trừ cho giá trị nhớ trên cờ C, sau khi thực hiện lệnh kết quả được lưu ở thanh ghi A. Lệnh này có ảnh hưởng đến thanh thanh trạng thái PSW.

A ⟵ (A) - (C) – ((Ri))

Ví dụ:

C = 1

Mov A,#0E5h

Mov 4Fh,#50h

Mov R3,#4Fh

Subb A,@R3

kết quả : A = 94h

C = 0

3.1.12. Lệnh trừ dữ liệu trên thanh ghi A với dữ liệu xác định và số nhớ ở cờ C.

Cú pháp: SUBB A,#data

Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte.

Thời gian thực hiện: 1 chu kì máy.

Công dụng: Trừ giá trị dữ liệu trên thanh ghi A với giá trị xác định và trừ thêm giá trị nhớ trên cờ C, sau khi thực hiện lệnh kết quả được lưu ở thanh ghi A. Lệnh này có ảnh hưởng đến thanh thanh trạng thái PSW.

A ⟵ (A) - (C) - #data

Ví dụ:

C = 0

Mov A,#05h

Subb A,#4Fh

kết quả : A = 0B6h

C = 1

3.1.13.Lệnh tăng giá trị dữ liệu trên thanh ghi A lên 1 đơn vị.

Cú pháp: INC A

Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte.

Thời gian thực hiện: 1 chu kì máy.

Công dụng: Tăng giá trị dữ liệu lưu giữ trên thanh ghi A lên 1 đơn vị, không ảnh hưởng đến các cờ nhớ trên PSW.

A ⟵ (A) + 1

Ví dụ: Mov A,#05h

Inc A

Kết quả : A = #06h

3.1.14.Lệnh tăng giá trị dữ liệu trên thanh ghi Rn lên 1 đơn vị.

Cú pháp: INC Rn

Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte.

Thời gian thực hiện: 1 chu kì máy.

Công dụng: Tăng giá trị dữ liệu lưu giữ trên thanh ghi Rn lên 1 đơn vị, không ảnh hưởng đến các cờ nhớ trên PSW.

A ⟵ (Rn) + 1

Ví dụ:

```
Mov   R7,#0Fh
Inc   R7
```

Kết quả : R7 = #10h

3.1.15.Lệnh tăng giá trị dữ liệu ở ô nhớ có địa chỉ direct lên 1 đơn vị.

Cú pháp: INC direct

Lệnh này chiếm dung lượng bộ nhớ ROM là 2 Byte.

Thời gian thực hiện: 1 chu kì máy.

Công dụng: Tăng giá trị dữ liệu ở một ô nhớ có địa chỉ direct lên 1 đơn vị, không ảnh hưởng đến các cờ nhớ trên PSW.

Ví dụ:

```
Mov   50h,#0FFh
Inc   50h
```

Kết quả : 50h = #00

3.1.16.Lệnh tăng giá trị dữ liệu ở ô nhớ có địa chỉ gián tiếp lên 1 đơn vị.

Cú pháp: Inc @Ri

Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte.

Thời gian thực hiện: 1 chu kì máy.

Công dụng: Tăng giá trị dữ liệu ở ô nhớ có địa chỉ bằng giá trị dữ liệu trên R_i lên 1 đơn vị, không ảnh hưởng đến các cờ nhớ trên PSW.

Ví dụ:

```
Mov   0Fh,#05h
Mov   R0,#0Fh
Inc   @R0
```

Kết quả : R0 = #06h

0Fh = #05h

3.1.17.Lệnh tăng giá trị của con trỏ dữ liệu DPTR lên 1 đơn vị.

Cú pháp: Inc DPTR.

Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte.

Thời gian thực hiện: 2 chu kì máy.

Công dụng: Tăng giá trị dữ liệu của thanh ghi con trỏ dữ liệu DPTR lên 1 đơn vị, không ảnh hưởng đến các cờ nhớ trên PSW.

Ví dụ:

```
Mov   DPTR,#5Fh
Inc   DPTR
```

Kết quả : DPTR = #060h

3.1.18. Lệnh giảm giá trị dữ liệu trên thanh ghi A xuống 1 đơn vị.

Cú pháp: DEC A

Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte.

Thời gian thực hiện: 1 chu kì máy.

Công dụng: Giảm giá trị dữ liệu lưu giữ trên thanh ghi A xuống 1 đơn vị, không ảnh hưởng đến các cờ nhớ trên PSW.

Ví dụ:

```
Mov   A,#05h
Dec   A
```

Kết quả : A = #04h

3.1.19. Lệnh giảm giá trị dữ liệu trên thanh ghi Rn xuống 1 đơn vị

Cú pháp: Dec Rn

Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte.

Thời gian thực hiện: 1 chu kì máy.

Công dụng: Giảm giá trị dữ liệu lưu giữ trên thanh ghi Rn xuống 1 đơn vị, không ảnh hưởng đến các cờ nhớ trên PSW.

Ví dụ:

```
Mov   R6,#0Fh
Dec   R6
```

Kết quả : R6 = #0Eh

3.1.20.Lệnh giảm giá trị dữ liệu ở ô nhớ có địa chỉ direct xuống 1 đơn vị.

Cú pháp: Dec direct

Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte.

Thời gian thực hiện: 1 chu kì máy.

Công dụng: Giảm giá trị dữ liệu ở ô nhớ có địa chỉ direct xuống 1 đơn vị, không ảnh hưởng đến các cờ nhớ trên PSW.

Ví dụ:

Mov 7Fh,#0

Dec 7Fh

Kết quả : 7Fh = #0FFh

3.1.21.Lệnh giảm giá trị dữ liệu ở ô nhớ có địa chỉ gián tiếp xuống 1 đơn vị.

Cú pháp: Dec @Ri

Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte.

Thời gian thực hiện: 1 chu kì máy.

Công dụng: Giảm giá trị dữ liệu ở ô nhớ có địa chỉ bằng giá trị dữ liệu trên Ri xuống1 đơn vị, không ảnh hưởng đến các cờ nhớ trên PSW.

Ví dụ: Mov 60h,#05h

Mov R1,#60h

Dec @R1

Kết quả : R1 = #04h

60h = #05h

3.1.22.Lệnh nhân thanh ghi A với thanh ghi B.

Cú pháp: Mul AB

Lệnh này chiếm dung lượng bộ nhớ ROM là 1 Byte.

Thời gian thực hiện: 4 chu kì máy.

Công dụng: Nhân hai dữ liệu là số nguyên không dấu ở thanh ghi A với thanh ghi B, kết quả là một dữ liệu 16 bit. Byte thấp của kết quả lưu ở thanh ghi A và byte cao của kết quả lưu ở thanh ghi B. Nếu tích số lớn hơn 255(0FFH), cờ tràn 0V ở thanh trạng thái PSW được thiết lập lên 1, ngược lại nếu tích số nhỏ hơn 255(0FFH), cờ tràn 0V được thiết lấp về 0. Cờ nhớ C luôn ở giá trị 0.