CHƯƠNG 2
Lập trình hợp ngữ 8051

2.1 Bên trong 8051.

Trong phần này chúng ta nghiên cứu các thanh ghi chính của 8051 và trình bày cách sử dụng với các lệnh đơn giản MOV và ADD.

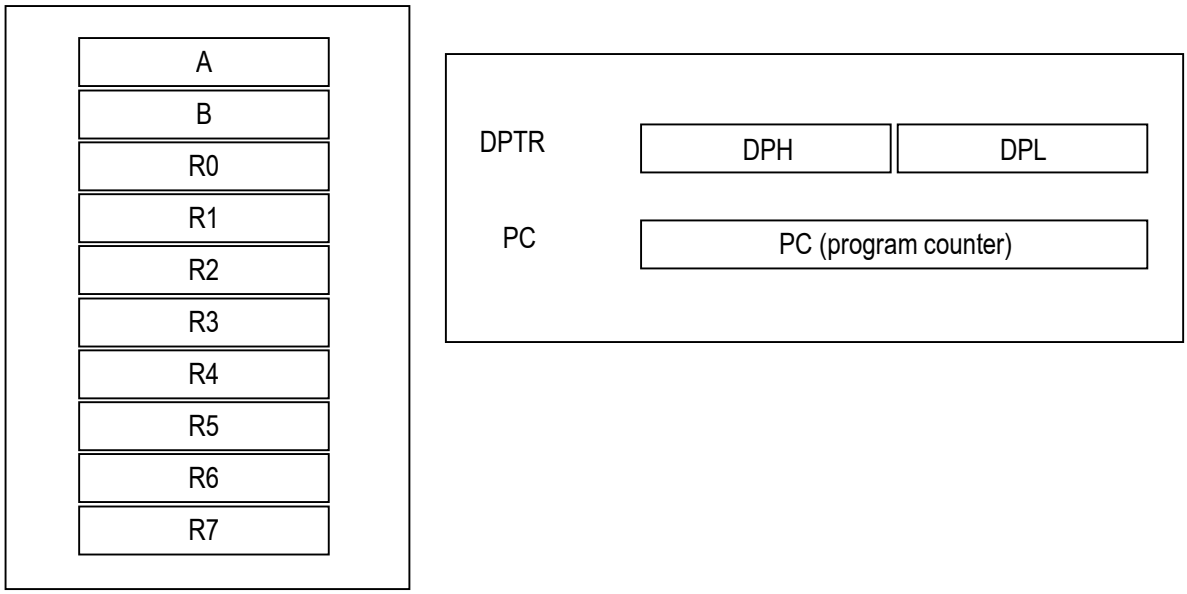
2.1.1 Các thanh ghi.

Trong CPU các thanh ghi được dùng để lưu cất thông tin tạm thời, những thông tin này có thể là một byte dữ liệu cần được sử lý hoặc là một địa chỉ đến dữ liệu cần được nạp. Phần lớn các thanh ghi của 8051 là các thanh ghi 8 bit. Trong 8051 chỉ có một kiểu dữ liệu: Loại 8 bit, 8 bit của một thanh ghi được trình bày như sau:

D7	D6	D5	D4	D3	D2	D1	D0

với MSB là bit có giá trị cao nhất D7 cho đến LSB là bit có giá trị thấp nhất D0. (MSB - Most Sigfican bit và LSB - Leart Significant Bit). Với một kiểu dữ liệu 8 bit thì bất kỳ dữ liệu nào lớn hơn 8 bit đều phải được chia thành các khúc 8 bit trước khi được xử lý. Vì có một số lượng lớn các thanh ghi trong 8051 ta sẽ tập trung vào một số thanh ghi công dụng chung đặc biệt trong các chương kế tiếp. Hãy tham khảo phụ lục Appendix A.3 để biết đầy đủ về các thanh ghi của 8051.

Hình 2.1: a) Một số thanh ghi 8 bit của 8051
b) Một số thanh ghi 16 bit của 8051



Các thanh ghi được sử dụng rộng rãi nhất của 8051 là A (thanh ghi tích luỹ), B, R0 - R7, DPTR (con trỏ dữ liệu) và PC (bộ đếm chương trình). Tất cả các dữ liệu trên đều là thanh g hi 8 bit trừ DPTR và PC là 16 bit. Thanh ghi tích luỹ A được sử dụng cho tất cả mọi phép toán số học và lô-gíc. Để hiểu sử dụng các thanh ghi này ta sẽ giới thiệu chúng trong các ví dụ với các lệnh đơn giản là ADD và MOV.

2.1.2 Lệnh chuyển MOV.

Nói một cách đơn giản, lệnh MOV sao chép dữ liệu từ một vị trí này đến một ví trí khác. Nó có cú pháp như sau:

```
MOV             ; Đích, nguồn; sao chép nguồn vào đích
```

Lệnh này nói CPU chuyển (trong thực tế là sao chép) toán hạng nguồn vào toán hạng đích. Ví dụ lệnh "MOV A, R0" sao chép nội dung thanh ghi R0 vào thanh ghi A. Sau khi lênh này được thực hiện thì thanh ghi A sẽ có giá trị giống như thanh ghi R0. Lệnh MOV không tác động toán hạng nguồn. Đoạn chương trình dưới đây đầu tiên là nạp thanh ghi A tới giá trị 55H 9là giá trị 55 ở dạng số Hex) và sau đó chuyển giá trị này qua các thanh ghi khác nhau bên trong CPU. Lưu ý rằng dấu "#" trong lệnh báo rằng đó là một giá trị. Tầm quan trọng của nó sẽ được trình bày ngay sau ví dụ này.

```
MOV   A, #55H;      ; Nạp trí trị 55H vào thanh ghi A (A = 55H)
MOV   R0, A         ; Sao chép nội dung A vào R0 (bây giờ R0=A)
MOV   R1, A         ; Sao chép nội dung A và R1 (bây giờ R1=R0=A)
MOV   R2, A         ; Sao chép nội dung A và R2 (bây giờ R2=R1=R0=A)
MOV   R3, #95H      ; Nạp giá trị 95H vào thanh ghi R3 (R3 = 95H)
MOV   A, R3         ; Sáo chép nội dung R3 vào A (bây giờ A = 95H)
```

Khi lập trình bộ vi điều khiển 8051 cần lưu ý các điểm sau:

1. Các giá trị có thể được nạp vào trực tiếp bất kỳ thanh ghi nào A, B, R0 - R7. Tuy nhiên, để thông báo đó là giá trị tức thời thì phải đặt trước nó một ký hiệu "#" như chỉ ra dưới đây.

```
MOV   A, #23H       ; Nạp giá trị 23H vào A (A = 23H)
MOV   R0, #12H      ; Nạp giá trị 12H vào R0 (R0 = 2BH)
MOV   R1, #1FH      ; Nạp giá trị 1FH vào R1 (R1 = 1FH)
MOV   R2, #2BH      ; Nạp giá trị 2BH vào R2 (R2 = 2BH)
MOV   B, # 3CH      ; Nạp giá trị 3CH vào B (B = 3CH)
MOV   R7, #9DH      ; Nạp giá trị 9DH vào R7 (R7 = 9DH)
MOV   R5, #0F9H     ; Nạp giá trị F9H vào R5 (R5 = F9H)
MOV   R6, #12       ;Nạp giá trị thập phân 12 = 0CH vào R6
                    (trong R6 có giá trị 0CH).
```

Để ý trong lệnh "MOV R5, #0F9H" thì phải có số 0 đứng trước F và sau dấu # báo rằng F là một số Hex chứ không phải là một ký tự. Hay nói cách khác "MOV R5, #F9H" sẽ gây ra lỗi.

2. Nếu các giá trị 0 đến F được chuyển vào một thanh ghi 8 bit thì các bit còn lại được coi là tất cả các số 0. Ví dụ, trong lệnh "MOV A,#5" kết quả là A=0.5, đó là A = 0000 0101 ở dạng nhị phân.
3. Việc chuyển một giá trị lớn hơn khả năng chứa của thanh ghi sẽ gây ra lỗi ví dụ:

```
MOV   A, #7F2H      ; Không hợp lệ vì 7F2H > FFH
MOV   R2, 456       ; Không hợp lệ vì 456 > 255 (FFH)
```

4. Để nạp một giá trị vào một thanh ghi thì phải gán dấu "#" trước giá trị đó. Nếu không có dấu thì nó hiểu rằng nạp từ một vị trí nhớ. Ví dụ "MOV A, 17H" có nghĩa là nạp giá trị trong ngăn nhớ có giá trị 17H vào thanh ghi A và tại địa chỉ đó dữ liệu có thể có bất kỳ giá trị nào từ 0 đến FFH. Còn để nạp giá trị là 17H vào thanh ghi A thì cần phải có dấu "#" trước 17H như thế này. "MOV A, #17H". Cần lưu ý rằng nếu thiếu dấu "#" trước một thì sẽ không gây lỗi vì hợp ngữ cho đó là một lệnh hợp

lệ. Tuy nhiên, kết quả sẽ không đúng như ý muốn của người lập trình. Đ ây sẽ là một lỗi thường hay gặp đối với lập trình viên mới.

2.1.3 Lệnh cộng ADD.

Lệnh cộng ADD có các phép như sau:

```
ADD a, nguồn          ; Cộng toán hạng nguồn vào thanh ghi A.
```

Lệnh cộng ADD nói CPU cộng byte nguồn vào thanh ghi A và đặt kết quả thanh ghi A. Để cộng hai số như 25H và 34H thì mỗi số có thể chuyển đến một thanh ghi và sau đó cộng lại với nhau như:

```
MOV   A, #25H       ; Nạp giá trị 25H vào A
MOV   R2, #34H      ; Nạp giá trị 34H vào R2
ADD   A, R2         ; Cộng R2 vào A và kết quả A = A + R2
```

Thực hiện chương trình trên ta được A = 59H (vì 25H + 34H = 59H) và R2 = 34H, chú ý là nội dụng R2 không thay đổi. Chương trình trên có thể viết theo nhiều cách phụ thuộc vào thanh ghi được sử dụng. Một trong cách viết khác có thể là:

```
MOV   R5, #25H      ; Nạp giá trị 25H vào thanh ghi R5
MOV   R7, #34H      ; Nạp giá trị 34H vào thanh ghi R7
MOV   A, #0         ; Xoá thanh ghi A (A = 0)
ADD   A, R5         ; Cộng nội dung R5 vào A (A = A + R5)
ADD   A, R7         ; Cộng nội dung R7 vào A (A = A + R7 = 25H + 34H)
```

Chương trình trên có kết quả trong A Là 59H, có rất nhiều cách để viết chương trình giống như vậy. Một câu hỏi có thể đặt ra sau khi xem đoạn chương trình trên là liệu có cần chuyển cả hai dữ liệu vào các thanh ghi trước khi cộng chúng với nhau không? Câu trả lời là không cần. Hãy xem đoạn chương trình dưới đây:

```
MOV   A, #25H       ; Nạp giá trị thứ nhất vào thanh ghi A (A = 25H)
ADD   A, #34H       ; Cộng giá trị thứ hai là 34H vào A (A = 59H)
```

Trong trường hợp trên đây, khi thanh ghi A đã chứa số thứ nhất thì giá trị thứ hai đi theo một toán hạng. Đây được gọi là toán hạng tức thời (trực tiếp).

Các ví dụ trước cho đến giờ thì lệnh ADD báo rằng toán hạng nguồn có thể hoặc là một thanh ghi hoặc là một dữ liệu trực tiếp (tức thời) nhưng thanh ghi đích luôn là thanh ghi A, thanh ghi tích luỹ. Hay nói cách khác là một lệnh như "ADD R2, #12H" là lệnh không hợp lệ vì mọi phép toán số học phải cần đến thanh ghi A và lệnh "ADD R4, A" cũng không hợp lệ vì A luôn là thanh ghi đích cho mọi phép số học. Nói một cách đơn giản là trong 8051 thì mọi phép toán số học đều cần đến thanh A với vai trò là toán hạng đích. Phần trình bày trên đây giải thích lý do vì sao thanh ghi A như là thanh thi tích luỹ. Cú pháp các lệnh hợp ngữ mô tả cách sử dụng chúng và liệt kê các kiểu toán hạng hợp lệ được cho trong phụ lục Appendix A.1.

Có hai thanh ghi 16 bit trong 8051 là bộ đếm chương trình PC và con trỏ dữ liệu APTR. Tầm quan trọng và cách sử dụng chúng được trình bày ở mục 2.3. Thanh ghi DPTR được sử dụng để truy cập dữ liệu và được làm kỹ ở chương 5 khi nói về các chế độ đánh địa chỉ.

2.2 Giới thiệu về lập trình hợp ngữ 8051.

Trong phần này chúng ta bàn về dạng thức của hợp ngữ và định nghĩa một số thuật ngữ sử dụng rộng rãi gắn liền với lập trình hợp ngữ.

CPU chỉ có thể làm việc với các số nhị phân và có thể chạy với tốc độ rất cao. Tuy nhiên, thật là ngán ngậm và chậm chạp đối với con người phải làm việc với các số 0 và 1 để lập trình cho máy tính. Một chương trình chứa các số 0 và 1 được gọi là ngôn ngữ máy.

Trong những ngày đầu của máy tính, các lập trình viên phải viết mã chương trình dưới dạng ngôn ngữ máy. Mặc dụ hệ thống thập lục phân (số Hex) đã được sử dụng như một cách hiệu quả hơn để biểu diễn các số nhị phân thì quá trình làm việc với mã máy vẫn còn là công việc cồng kềnh đối với con người. Cuối cùng, các nguồn ngữ hợp ngữ đã được phát, đã cung cấp các từ gợi nhớ cho các lệnh mã máy cộng với những đặc tính khác giúp cho việc lập trình nhanh hơn và ít mắc lỗi hơn. Thuật ngữ từ gợi nhớ (mnemonic) thường xuyên sử dụng trong tài liệu khoa học và kỹ thuật máy tính để tham chiếu cho các mã và từ rút gọn tương đối dễ nhớ, các chương trình hợp ngữ phải được dịch ra thanh mã máy bằng một chương trình được là trình hợp ngữ (hợp dịch). Hợp ngữ được coi như là một ngông ngữ bậc thấp vì nó giao tiếp trực tiếp với cấu trúc bên trong của CPU. Để lập trình trong hợp ngữ, lập trình viên phải biết tất cả các thanh ghi của CPU và kích thước của chúng cũng như các chi tiết khác.

Ngày nay, ta có thể sử dụng nhiều ngôn ngữ lập trình khác nhau, chẳng hạn như Basic, Pascal, C, C^{++}, Java và vô số ngôn ngữ khác. Các ngôn ngữ này được coi là nhưng ngôn ngữ bậc cao vì lập trình viên không cần phải tương tác với các chi tiết bên trong của CPU. Một trình hợp dịch được dùng để dịch chương trình hợp ngữ ra mã máy còn (còn đôi khi cũng còn được gọi mà đối tượng (Object Code) hay mã lệnh Opcode), còn các ngôn ngữ bậc cao được dịch thành các ngôn ngữ mã máy bằng một chương trình gọi là trình biên dịch. Ví dụ, để viết một chương trình trong C ta phải sử dụng một trình biên dịch C để dịch chương trình về dạng mã máy. Bây giờ ta xét dạng thức hợp ngữ của 8051 và sử dụng trình hợp dịch để tạo ra một chương trình sẵn sàng chạy ngay được.

2.2.1 Cấu trúc của hợp ngữ.

Một chương trình hợp ngữ bao gồm một chuỗi các dòng lệnh hợp ngữ. Một lệnh hợp ngữ có chứa một từ gợi nhớ (mnemonic) và tuy theo từng lệnh và sau nó có một hoặc hai toán hạng. Các toán hạng là các dữ liệu cần được thao tác và các từ gợi nhớ là các lệnh đối với CPU nói nó làm gì với các dữ liệu.

```
        ORG   0H          ; Bắt đầu (origin) tại ngăn nhớ 0
        MOV   R5, #25H    ; Nạp 25H vào R5
        MOV   R7, #34H    ; Nạp 34H vào R7
        MOV   A, #0       ; Nạp 0 vào thanh ghi A
        ADD   A, R5       ; Cộng nôi dụng R5 vào A (A = A + R5)
        ADD   A, R7       ; Cộng nội dung R7 vào A (A = A + R7)
        ADD   A, #121H    ; Cộng giá trị 12H vào A (A = A + 12H)
HERE: SJMP HERE           ; ở lại trong vòng lặp này
        END               ; Kết thúc tệp nguồn hợp ngữ
```

Chương trình 2.1: Ví dụ mẫu về một chương trình hợp ngữ.

Chương trình 2.1 cho trên đây là một chuỗi các câu lệnh hoặc các dòng lệnh được viết hoặc bằng các lệnh hợp ngữ như ADD và MOV hoặc bằng các câu lệnh được gọi là các chỉ dẫn. Trong khi các lệnh hợp ngữ thì nói CPU phải làm gì thì các chỉ lệnh

(hay còn gọi là giả lệnh) thì đưa ra các chỉ lệnh cho hợp ngữ. Ví dụ, trong chương trình 2.1 thì các lệnh ADD và MOV là các lệnh đến CPU, còn ORG và END là các chỉ lệnh đối với hợp ngữ. ORG nói hợp ngữ đặt mã lệnh tại ngăn nhớ 0 và END thì báo cho hợp ngữ biết kết thúc mã nguồn. Hay nói cách khác một chỉ lệnh để bắt đầu và chỉ lệnh thứ hai để kết thúc chương trình.

Cấu trúc của một lệnh hợp ngữ có 4 trường như sau:

[nhãn:] [từ gợi nhớ] [các toán hạng] [; chú giải]

Các trường trong dấu ngoặc vuông là tuỳ chọn và không phải dòng lệnh nào cũng có chúng. Các dấu ngoặc vuông không được viết vào. Với dạng thức trên đây cần lưu ý các điểm sau:

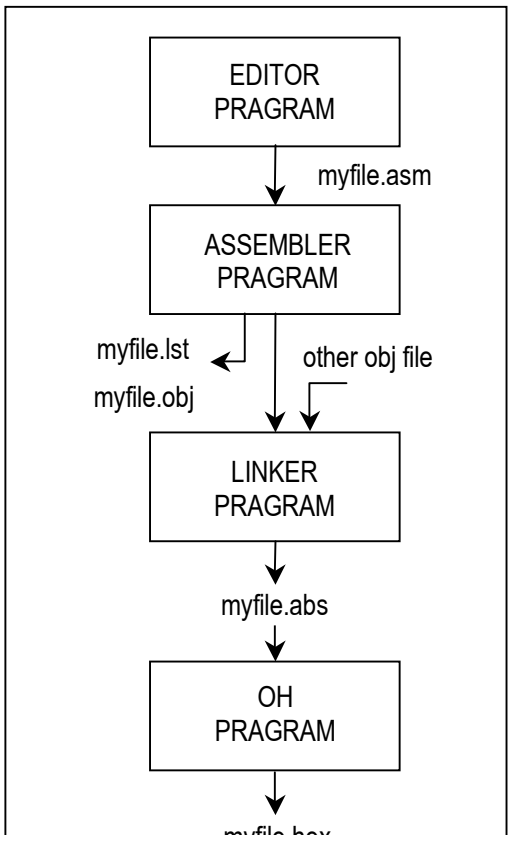
1. Trường nhãn cho phép chương trình tham chiếu đến một dòng lệnh bằng tên. Nó không được viết quá một số ký tự nhất định. Hãy kiểm tra quy định này của hợp ngữ mà ta sử dụng.
2. Từ gợi nhớ (lệnh) và các toán hạng là các trường kết hợp với nhau thực thi công việc thực tế của chương trình và hoàn thiện các nhiệm vụ mà chương trình được viết cho chúng. Trong hợp ngữ các câu lệnh như:

   " ADD A, B"
   "MOV A, #67H"

   thì ADD và MOV là những từ gợi nhớ tạo ra mã lệnh, còn "A, B" và "A, #67H" là những toán hạng thì hai trường có thể chứa các lệnh giả hoặc chỉ lệnh của hợp ngữ. Hãy nhớ rằng các chỉ lệnh không tạo ra mã lệnh nào (mã máy) và chúng chỉ dùng bởi hợp ngữ, ngược lại đối với các lệnh là chúng được dịch ra mã máy (mã lênh) cho CPU thực hiện. Trong chương trình 2.1 các lệnh ORG và END là các chỉ lệnh (một số hợp ngữ của 8051 sử dụng dạng .ORG và .END). Hãy đọc quy định cụ thể của hợp ngữ ta sử dụng.
3. Chương chú giải luôn phải bắt đầu bằng dấu chấm phẩy (;). Các chú giải có thể bắt đầu ở đầu dòng hoặc giữa dòng. Hợp ngữ bỏ qua (làm ngơ) các chú giải nhưng chúng lại rất cần thiết đối với lập trình viên. Mặc dù các chú giải là tuỳ chọn, không bắt buộc nhưng ta nên dùng chúng để mô tả chương trình để giúp cho người khác đọc và hiểu chương trình dễ dàng hơn.
4. Lưu ý đến nhãn HERE trong trường nhãn của chương trình 2.1. Một nhãn bất kỳ tham chiếu đến một lệnh phải có dấu hai chấm (:) đứng ở sau. Trong câu lệnh nhảy ngắn SJMP thì 8051 được ra lệnh ở lại trong vòng lặp này vô hạn. Nếu hệ thống của chúng ta có một chương trình giám sát thì takhông cần dòng lệnh này và nó có thể được xoá đi ra khỏi chương trình.

2.3 Hợp dịch và chạy một chương trình 8051.

Như vậy cấu trúc của một chương trình hợp ngữ ta đã được biết, câu hỏi đặt ra là chương



trình sẽ được tạo ra và hợp dịch như thế nào và làm thế nào để có thể chạy được? Các bước để tạo ra một chương trình hợp ngữ có thể chạy được là:

1. Trước hết ta sử dụng mộ trình soạn thảo để gỡ vào một chương trình giống như chương trình 2.1. Có nhiều trình soạn thảo tuyệt vời hoặc các bộ sử lý từ được sử dụng để tạo ra và/ hoặc để soạn thảo chương trình. Một trình soạn thảo được sử dụng rộng rãi là trình soạn thảo EDIT của MS-DOS (hoặc Noterad của Windows) đều chạy trên hệ điều hành Microsoft. Lưu ý rằng, trình soạn thảo phải có khả năng tạo ra tệp mã ASCII. Đối với nhiều trình hợp ngữ thì các tên tệp tuân theo các quy ước thường lệ củ DOS, nhưng phần mở rộng của các tệp nguồn phải là "asm" hay "src" tuỳ theo trình hợp ngữ mà ta sử dụng.
2. Tệp nguồn có phần mở rộng "asm" chứa mã chương trình được tạo ra ở bước 1 được nạp vào trình hợp dịch của 8051. Trình hợp dịch chuyển các lệnh ra mã máy. Trình hợp dịch sẽ tạo ra một tệp đối tượng và một tệp liệt kê với các thành phần mở rộng "obj" và "lst" tương ứng.
3. Các trình hợp dịch yêu cầu một bước thứ ba gọi là liên kết. Chương trình liên kết lấy một hoặc nhiều tệp đối tượng và tạo ra một tệp đối tượng tuyệt đối với thành phần mở rộng "abs". Tệp "abs" này được sử dụng bởi thùng chứa của 8051 có một chương trình giám sát.
4. Kế sau đó tệp "abs" được nạp vào một chương trình được gọi là "0H" (chuyển đối tượng object về dạng số Hex) để tạo ra một tệp với đuôi mở rộng "Hex" có thể nạp tốt vào trong ROM. Chương trình này có trong tất cả mọi trình hợp ngữ của 8051 các trình hợp ngữ dựa trên Windows hiện nay kết hợp các bước 2 đến 4 vào thành một bước.

   Hình 2.2: Các bước để tạo ra một chương trình.

2.3.1 Nói thêm về các tệp ".asm" và ".object".

Tệp ".asm" cũng được gọi là tệp nguồn và chính vì lý do này mà một số trình hợp ngữ đòi hỏi tệp này phải có một phần mở rộng "src" từ chữ "source" là nguồn. Hãy kiểm tra hợp ngữ 8051 mà ta sử dụng xem nó có đòi hỏi như vậy không? Như ta nói trước đây tệp này được tạo ra nhờ một trình biên tập chẳng hạn như Edit của DOS hoặc Notepad của Windows. Hợp ngữ của 8051 chuyển đổi các tệp hợp ngữ trong tệp .asm thành ngôn ngữ mã máy và cung cấp tệp đối tượng .object. Ngoài việc tạo ra tệp đối tượng trình hợp ngữ cũng cho ra tệp liệt kê "lst" (List file).

2.3.2 Tệp liệt kê ".lst".

Tệp liệt kê là một tuỳ chọn, nó rất hữu ích cho lập trình viên vì nó liệt kê tất cả mọi mã lệnh và địa chỉ cũng như tất cả các lỗi mà trình hợp ngữ phát hiện ra. Nhiều trình hợp ngữ giả thiết rằng, tệp liệt kê là không cần thiết trừ khi ta báo rằng ta muốn tạo ra nó. Tệp này có thể được truy cập bằng một trình biên dịch như Edit của DOS hoặc Notepad của Window và được hiển thị trên màn hình hoặc được gửi ra máy in. Lập trình viên sử dụng tệp liệt kê để tìm các lỗi cú pháp. Chỉ sau khi đã sửa hết các lỗi được đánh dấu trong tệp liệt kê thì tệp đối tượng mới sẵn sàng làm đầu vào cho chương trình liên kết.

```
1 0000              ORG   0H          ; Bắt đầu ở địa chỉ 0
2 0000  7D25        MOV   R5, #25H    ; Nạp giá trị 25H vào R5
3 0002 7F34         MOV   R7, #34H    ; Nạp giá trị 34H vào R7
4 0004  7400        MOV   A, #0       ; Nạp 0 vào A (xoá A)
5 0006 2D           ADD   A, R5       ; Cộng nội dung R5 vào A (A = A + R5)
6 0007 2F           ADD   A, R7       ; Cộng nội dung R7 vào A (A = A + R7)
7 0008 2412         ADD   A, #12H     ; Cộng giá trị 12H vào A (A = A + 12H)
```

```
8 00A  BCEF  HERE: SJMP HERE          ; ở lại vòng lặp này
9 000C             END                ; Kết thúc tệp .asm
```

Chương trình 2.2: Tệp liệt kê.

2.4 Bộ đếm chương trình và không gian ROM trong 8051.

2.4.1 Bộ đếm chương trình trong 8051.

Một thanh ghi quan trọng khác trong 8051 là bộ đếm chương trình . Bộ đếm chương trình chỉ đếm địa chỉ của lệnh kế tiếp cần được thực hiện. Khi CPU nạp mã lệnh từ bộ nhớ ROM chương trình thì bộ đếm chương trình tăng lên chỉ đếm lệnh kết tiếp. Bộ đếm chương trình trong 8051 có thể truy cập các địa chỉ chương trình trong 8051 rộng 16 bit. Điều này có nghĩa là 8051 có thể truy cập các địa chỉa chương trình từ 0000 đến FFFFH tổng cộng là 64k byte mã lệnh. Tuy nhiên, không phải tất cả mọi thành viên của 8051 đều có tất cả 64k byte ROM trên chíp được cài đặt. Vậy khi 8051 được bật nguồn thì nó đánh thức ở địa chỉa nào?

2.4.2 Địa chỉ bắt đầu khi 8051 được cấp nguồn.

Một câu hỏi mà ta phải hỏi về bộ vi điều khiển bất kỳ là thì nó được cấp nguồn thì nó bắt đầu từ địa chỉ nào? Mỗi bộ vi điều khiển đều khác nhau. Trong trường hợp họ 8051 thì mọi thành viên kể từ nhà sản xuất nào hay phiên bản nào thì bộ vi điều khiển đều bắt đầu từ địa chỉ 0000 khi nó được bật nguồn. Bật nguồn ở đây có nghĩa là ta cấp điện áp V_{cc} đến chân RESET như sẽ trình bày ở chương 4. Hay nói cách khác, khi 8051 được cấp nguồn thì bộ đếm chương trình có giá trị 0000. Điều này có nghĩa là nó chờ mã lệnh đầu tiên được lưu ở địa chỉa ROM 0000H. Vì lý do này mà trong vị trí nhớ 0000H của bộ nhở ROM chương trình vì đây là nơi mà nó tìm lệnh đầu tiên khi bật nguồn. Chúng ta đạt được điều này bằng câu lệnh ORG trong chương trình nguồn như đã trình bày trước đây. Dưới đây là hoạt động từng bước của bộ đếm chương trình trong qúa trình nạp và thực thi một chương trình mẫu.

2.4.3 Đặt mã vào ROM chương trình.

Để hiểu tốt hơn vai trò của bộ đếm chương trình trong quá trình nạp và thực thi một chương trình, ta khảo sát một hoạt động của bộ đếm chương trình khi mỗi lệnh được nạp và thực thi. Trước hết ta khảo sát một lần nữa tệp liệt kê của chương trình mẫu và cách đặt mã vào ROM chương trình 8051 như thế nào? Như ta có thể thấy, mã lệnh và toán hạng đối với mỗi lệnh được liệt kê ở bên trái của lệnh liệt kê.

Chương trình 2.1: Ví dụ mẫu về một chương trình hợp ngữ.

Chương trình 2.1 cho trên đây là một chuỗi các câu lệnh hoặc các dòng lệnh được viết hoặc bằng các lệnh hợp ngữ như ADD và MOV hoặc bằng các câu lệnh được gọi là các chỉ dẫn. Trong khi các lệnh hợp ngữ thì nói CPU phải làm gì thì các chỉ lệnh (hay còn gọi là giả lệnh) thì đưa ra các chỉ lệnh cho hợp ngữ. Ví dụ, trong chương trình 2.1 thì các lệnh ADD và MOV là các lệnh đến CPU, còn ORG và END là các chỉ lệnh đối với hợp ngữ. ORG nói hợp ngữ đặt mã lệnh tại ngăn nhớ 0 và END thì báo cho hợp ngữ biết kết thúc mã nguồn. Hay nói cách khác một chỉ lệnh để bắt đầu và chỉ lệnh thứ hai để kết thúc chương trình.

Cấu trúc của một lệnh hợp ngữ có 4 trường như sau:

[nhãn:] [từ gợi nhớ] [các toán hạng] [; chú giải]

Các trường trong dấu ngoặc vuông là tuỳ chọn và không phải dòng lệnh nào cũng có chúng. Các dấu ngoặc vuông không được viết vào. Với dạng thức trên đây cần lưu ý các điểm sau:

Trường nhãn cho phép chương trình tham chiếu đến một dòng lệnh bằng tên. Nó không được viết quá một số ký tự nhất định. Hãy kiểm tra quy định này của hợp ngữ mà ta sử dụng.
Từ gợi nhớ (lệnh) và các toán hạng là các trường kết hợp với nhau thực thi công việc thực tế của chương trình và hoàn thiện các nhiệm vụ mà chương trình được viết cho chúng. Trong hợp ngữ các câu lệnh như:

" ADD A, B"
"MOV A, #67H"

Thì ADD và MOV là những từ gợi nhớ tạo ra mã lệnh, còn "A, B" và "A, #67H" là những toán hạng thì hai trường có thể chứa các lệnh giả hoặc chỉ lệnh của hợp ngữ. Hãy nhớ rằng các chỉ lệnh không tạo ra mã lệnh nào (mã máy) và chúng chỉ dùng bởi hợp ngữ, ngược lại đối với các lệnh là chúng được dịch ra mã máy (mã lênh) cho CPU thực hiện. Trong chương trình 2.1 các lệnh ORG và END là các chỉ lệnh (một số hợp ngữ của 8051 sử dụng dạng .ORG và .END). Hãy đọc quy định cụ thể của hợp ngữ ta sử dụng.

Trương chú giải luôn phải bắt đầu bằng dấu chấm phẩy (;). Các chú giải có thể bắt đầu ở đầu dòng hoặc giữa dòng. Hợp ngữ bỏ qua (làm ngơ) các chú giải nhưng chúng lại rất cần thiết đối với lập trình viên. Mặc dù các chú giải là tuỳ chọn, không bắt buộc nhưng ta nên dùng chúng để mô tả chương trình để giúp cho người khác đọc và hiểu chương trình dễ dàng hơn.

Lưu ý đến nhãn HERE trong trường nhãn của chương trình 2.1. Một nhãn bất kỳ tham chiếu đến một lệnh phải có dấu hai chấm (:) đứng ở sau. Trong câu lệnh nhảy ngắn SJMP thì 8051 được ra lệnh ở lại trong vòng lặp này vô hạn. Nếu hệ thống của chúng ta có một chương trình giám sát thì takhông cần dòng lệnh này và nó có thể được xoá đi ra khỏi chương trình.

Địa chỉ	Mã lệnh
0000	7D
0001	25
0002	F7
0003	34
0004	74
0005	00
0006	2D
0007	2F
0008	24
0009	12
000A	80
000B	FE

Chương trình 2.1: Tệp liệt kê

Sau khi chương trình được đốt vào trong ROM của thành viên họ 8051 như 8751 hoặc AT 8951 hoặc DS 5000 thì mã lệnh và toán hạng được đưa vào các vị trí nhớ ROM bắt đầu từ địa chỉ 0000 như bảng liệt kê dưới đây.

Địa chỉ ROM	Ngôn ngữ máy	Hợp ngữ
0000	7D25	MOV R5, #25H
0002	7F34	MOV R7, #34H
0004	7400	MOV A, #0
0006	2D	ADD A, R5
0007	2F	ADD A, R7
0008	2412	ADD A, #12H
000A	80EF	HERE: SJMP HERE

Bảng nội dung ROM của chương trình 2.1.

Bảng liệt kê chỉ ra địa chỉ 0000 chứa mã 7D là mã lệnh để chuyển một giá trị vào thanh ghi R5 và địa chỉ 0001 chứa toán hạng (ở đây là giá trị 254) cần được chuyển vào R5. Do vậy, lệnh "MOV R5, #25H" có mã là "7D25" trong đó 7D là mã lệnh,

còng 25 là toán hạng. Tương tự như vậy, mã máy "7F34" được đặt trong các ngăn nhớ 0002 và 0003 và biểu diễn mã lệnh và toán hạng đối với lệnh "MOV R7, #34H". Theo cách như vậy, mã máy "7400" được đặt tại địa chỉ 0004 và 0005 và biểu diễn mã lệnh và toán hạng đối với lệnh "MOV A, #0". Ngăn nhớ 0006 có mã 2D là mã đối với lệnh "ADD A, R5" và ngăn nhớ 0007 có nội dung 2F là mã lệnh cho "ADD A, R7". Mã lệnh đối với lệnh "ADD A, #12H" được đặt ở ngăn nhớ 0008 và toán hạng 12H được đặt ở ngăn nhớ 0009. Ngăn nhớ 000A có mã lệnh của lệnh SJMP và địa chỉ đích của nó được đặt ở ngăn nhớ 000B. Lý do vì sao địa chỉ đích là FE được giải thích ở chương 3.

2.4.4 Thực hiện một chương trình theo từng byte.

Giả sử rằng chương trình trên được đốt vào ROM của chíp 8051 hoặc(8751, AT 8951 hoặc DS 5000) thì dưới đây là mô tả hoạt động theo từng bước của 8051 khi nó được cấp nguồn.

1. Khi 8051 được bật nguồn, bộ đếm chương trình PC có nội dung 0000 và bắt đầu nạp mã lệnh đầu tiên từ vị trí nhớ 0000 của ROM chương trình. Trong trường hợp của chương trình này là mã 7D để chuyển một toán hạng vào R5. Khi thực hiện mã lệnh CPU nạp giá trị 25 vào bộ đếm chương trình được tăng lên để chỉ đến 0002 (PC = 0002) có chứa mã lệnh 7F là mã của lệnh chuyển một toán hạng vào R7 "MOV R7, ...".
2. Khi thực hiện mã lệnh 7F thì giá trị 34H được chuyển vào R7 sau đó PC được tăng lên 0004.
3. Ngăn nhớ 0004 chứa mã lệnh của lệnh "MOV A, #0". Lệnh này được thực hiện và bây giờ PC = 0006. Lưu ý rằng tất cả các lệnh trên đều là những lệnh 2 byte, nghĩa là mỗi lệnh chiếm hai ngăn nhớ.
4. Bây giờ PC = 0006 chỉ đến lệnh kế tiếp là "ADD A, R5". Đây là lệnh một byte, sau khi thực hiện lệnh này PC = 0007.
5. Ngăn nhớ 0007 chứa mã 2F là mã lệnh của "ADD A, R7". Đây cũng là lệnh một byte, khi thực hiện lệnh này PC được tăng lên 0008. Qúa trình này cứ tiếp tục cho đến khi tất cả moi lệnh đều được nạp và thực hiện. Thực tế mà bộ đếm chương trình chỉ đến lệnh kế tiếp cần được thực hiện giải thích tại sao một số bộ vi xử lý (đáng nói là × 86) gọi bộ đếm là con trỏ lệnh (Instruction Pointer).

2.4.5 Bản đồ nhớ ROM trong họ 8051.

Như ta đã thấy ở chương trước, một số thành viên họ 8051 chỉ có 4k byte bộ nhớ ROM trên chíp (ví dụ 8751, AT 8951) và một số khác như AT 8951 có 8k byte ROM, DS 5000-32 của Dallas Semiconductor có 32k byte ROM trên chíp. Dallas Semiconductor cũng có motọ 8051 với ROM trên chíp là 64k byte. Điểm cần nhớ là không có thành viên nào của họ 8051 có thể truy cập được hơn 64k byte mã lệnh vì bộ đếm chương trình của 8051 là 16 bit (dải địa chỉ từ 0000 đến FFFFH). Cần phải ghi nhớ là lệnh đầu tiên của ROM chương trình đều đặt ở 0000, còn lệnh cuối cùng phụ thuộc vào dung lượng ROM trên chíp của mỗi thành viên họ 8051. Trong số các thành viên họ 8051 thì 8751 và AT 8951 có 4k byte ROM trên chíp. Bộ nhớ ROM trên chíp này có các địa chỉ từ 0000 đến 0FFFH. Do vậy, ngăn nhớ đầu tiên có địa chỉ 0000 và ngăn nhớ cuối cùng có địa chỉa 0FFFH. Hãy xét ví dụ 2.1.

VÝ dô 2.1:

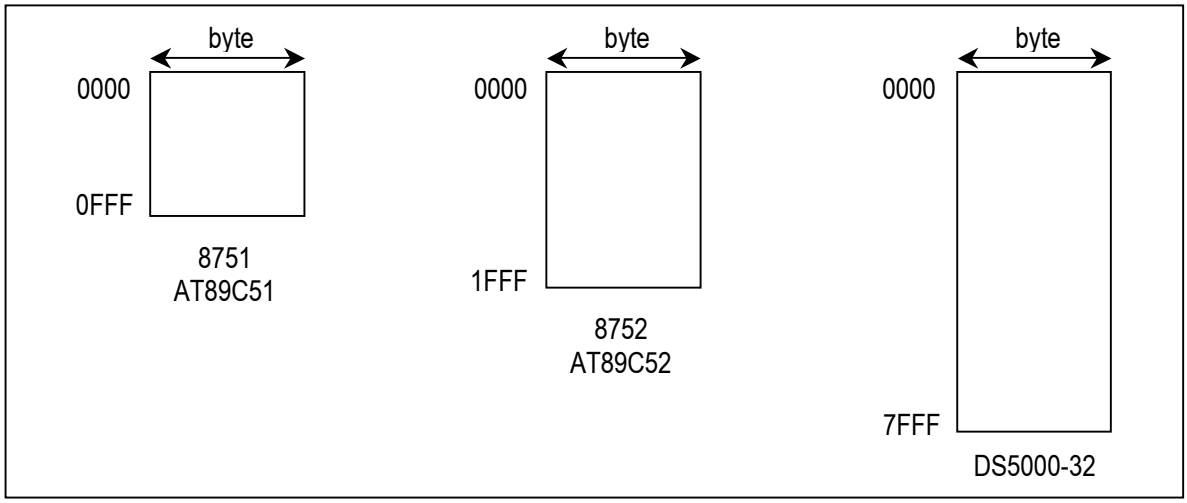
Tìm địa chỉ bộ nhớ ROM của mỗi thành viên họ 8051 sau đây.

a) AT 8951 (hoặc 8751) với 4k byte
b) DS 5000-32 với 32k byte

Lêi gi¶i:

a) Với 4k byte của không gian nhớ ROM trên chíp ta có 4096 byte bằng 1000H ở dạng Hex (4 × 1024 = 4096 hay 1000 ở dạng Hex). Bộ nhớ này được xắp xếp trong các ngăn nhớ từ 0000 đến 0FFFH. Lưu ý 0 luôn là ngăn nhớ đầu tiên.

b) Với 32k byte nhớ ta có 32.768 byte (32 × 1024). Chuyển đổi 32.768 về số Hex ta nhận được giá trị 8000H. Do vậy, không gian nhớ là dải từ 0000 đến 7FFFH.



Hình 2.3: Dải địa chỉ của ROM trên chíp một số thành viên họ 8051.

2.5 Các kiểu dữ liệu và các chỉ lệnh.

2.5.1 Kiểu dữ liệu và các chỉ lệnh của 8051.

Bộ vi điều khiển chỉ có một kiểu dữ liệu, nó là 8 bit và độ dài mỗi thanh ghi cũng là 8 bit. Công việc của lập trình viên là phân chia dữ liệu lớn hơn 8 bit ra thành từng khúc 8 bit (từ 00 đến FFH hay từ 0 đến 255) để CPU xử lý. Ví dụ về xử lý dữ liệu lớn hơn 8 bit được trình bày ở chương 6. Các dữ liệu được sử dụng bởi 8051 có thể là số âm hoặc số dương và về xử lý các số có dấu được bàn ở chương 6.

2.5.2 Chỉ lệnh DB (định nghĩa byte).

Chỉ lệnh DB là một chỉ lệnh dữ liệu được sử dụng rộng rãi nhất trong hợp ngữ. Nó được dùng để định nghĩa dữ liệu 8 bit. Khi DB được dùng để định nghĩa byte dữ liệu thì các số có thể ở dạng thập phân, nhị phân, Hex hoặc ở dạng thức ASII. Đối với dữ liệu thập phân thì cần đặt chữ "D" sau số thập phân, đối với số nhị phân thì đặt chữ "B" và đối với dữ liệu dạng Hex thì cần đặt chữ "H". Bất kể ta sử dụng số ở dạng thức nào thì hợp ngữ đều chuyển đổi chúng về thành dạng Hex. Để báo dạng thức ở dạng mã ASCII thì chỉ cần đơn giản đặt nó vào dấu nháy đơn 'như thế này'. Hợp ngữ sẽ gán mã ASCII cho các số hoặc các ký tự một cách tự động. Chỉ lệnh DB chỉ là chỉ lệnh mà có thể được sử dụng để định nghĩa các chuỗi ASCII lớn hơn 2 ký tự. Do vậy, nó có thể được sử dụng cho tất cả mọi định nghĩa dữ liệu ASCII. Dưới đây là một số ví dụ về DB:

```
        ORG     500H
DATA1:  DB      2B                  ; Số thập phân (1C ở dạng Hex)
DATA2:  DB      00110101B           ; Số nhị phân (35 ở dạng Hex)
DATA3:  DB      39H                 ; Số dạng Hex
                ORG     510H
DATA4:  DB      "2591"              ; Các số ASCII
                ORG     518H
DATA5   DB      "My name is Joe"    ; Các ký tự ASCII
```

Các chuỗi ASCII có thể sử dụng dấu nháy đơn 'như thế này' hoặc nháy kép "như thế này". Dùng dấu phẩy kép sẽ hữu ích hơn đối với trường hợp dấu nháy đơn được dùng sở hữu cách như thế này "Nhà O' Leary". Chỉ lệnh DB cũng được dùng để cấp phát bộ nhớ theo từng đoạn kích thước một byte.

2.5.3 Các chỉ lệnh của hợp ngữ.

1. Chỉ lệnh ORG: Chỉ lệnh ORG được dùng để báo bắt đầu của địa chỉ. Số đi sau ORG có kể ở dạng Hex hoặc thập phân. Nếu số này có kèm chữ H đằng sau thì là ở dạng Hex và nếu không có chữ H ở sau là số thập phân và hợp ngữ sẽ chuyển nó thành số Hex. Một số hợp ngữ sử dụng dấu chấm đứng trước "ORG" thay cho "ORG". Hãy đọc kỹ về trình hợp ngữ ta sử dụng.
2. Chỉ lệnh EQU: Được dùng để định nghĩa một hằng số mà không chiếm ngăn nhớ nào. Chỉ lệnh EQU không dành chỗ cất cho dữ liệu nhưng nó gắn một giá trị hằng số với nhãn dữ liệu sao cho khi nhãn xuất hiện trong chương trình giá trị hằng số của nó sẽ được thay thế đối với nhãn. Dưới đây sử dụng EQU cho hằng số bộ đếm và sau đó hằng số được dùng để nạp thanh ghi RS.

```
COUNT  EQU 25
MOV    R3, #count
```

   Khi thực hiện lện "MOV R3, #COUNT" thì thanh ghi R3 sẽ được nạp giá trị 25 (chú ý đến dấu #). Vậy ưu điểm của việc sử dụng EQU là gì? Giả sử có một hằng số (một giá trị cố định) được dùng trong nhiều chỗ khác nhau trong chương trình và lập trình viên muốn thay đổi giá trị của nó trong cả chương trình. Bằng việc sử dụng chỉ lệnh EQU ta có thể thay đổi một số lần và hợp ngữ sẽ thay đổi tất cả mọi lần xuất hiện của nó là tìm toàn bộ chương trình và gắng tìm mọi lần xuất hiện.
3. Chỉ lệnh END: Một lệnh quan trọng khác là chỉ lệnh END. Nó báo cho trình hợp ngữ kết thúc của tệp nguồn "asm" chỉ lệnh END là dòng cuối cùng của chương trình 8051 có nghĩa là trong mã nguồn thì mọi thứ sau chỉ lệnh END để bị trình hợp ngữ bỏ qua. Một số trình hợp ngữ sử dụng .END có dấu chấm đứng trước thay cho END.

2.5.4 Các quy định đố với nhãn trong hợp ngữ.

Bằng cách chọn các tên nhãn có nghĩa là một lập trình viên có thể làm cho chương trình dễ đọc và dễ bảo trì hơn, có một số quy định mà các tên nhãn phải tuân theo. Thứ nhất là mỗi tên nhãn phải thống nhất, các tên được sử dụng làm nhãn trong hợp ngữ gồm các chữ cái viết hoa và viết thường, các số từ 0 đến 9 và các dấu đặc biệt như: dấu hỏi (?), dấu (@), dấu gạch dưới (_), dấu đô là ($) và dấu chu kỳ (.). Ký tự đầu tiên của nhãn phải là một chữ cái. Hay nói cách khác là nó không thể là số Hex. Mỗi trình hợp ngữ có một số từ dự trữ là các từ gợi nhớ cho các lệnh mà không được dùng để làm nhãn trong chương trình. Ví dụ như "MOV" và "ADD". Bên cạnh các từ gợi nhớ còn có một số tự dự trữ khác, hãy kiểm tra bản liệt kê các từ dự phòng của hợp ngữ ta đang sử dụng.

2.6 Các bit cờ và thanh ghi đặc bệt PSW của 8051.

Cũng như các bộ vi xử lý khác, 8051 có một thanh ghi cờ để báo các điều kiện số học như bit nhớ. Thanh ghi cờ trong 8051 được gọi là thanh ghi từ trạng thái chương trình PSW. Trong phần này và đưa ra một số ví dụ về cách thay đổi chúng.

2.6.1 Thanh ghi từ trạng thái chương trình PSW.

Thanh ghi PSW là thanh ghi 8 bit. Nó cũng còn được coi như là thanh ghi cờ. Mặc dù thanh ghi PSW rộng 8 bit nhưng chỉ có 6 bit được 8051 sử dụng. Hai bit chưa dùng là các cờ ch người dùng định nghĩa. Bốn trong số các cờ được gọi là các cờ có điều kiện, có nghĩa là chúng báo một số điều kiện do kết quả của một lệnh vừa được thực hiện. Bốn cờ này là cờ nhớ CY (carry), cờ AC (auxiliary cary), cờ chẵn lẻ P (parity) và cờ tràn OV (overflow).

Như nhìn thấy từ hình 2.4 thì các bit PSW.3 và PSW.4 được gán như RS0 và RS1 và chúng được sử dụng để thay đổi các thanh ghi băng. Chúng sẽ được giải thích ở phần kế sau. Các bit PSW.5 và PSW.1 là các bit cờ trạng thái công dụng chung và lập trình viên có thể sử dụng cho bất kỳ mục đích nào.

CY	AC	F0	RS1	RS0	OV	–	P

CY PSW.7 ; Cờ nhớ
AC PSW.6 ; Cờ

- PSW.5 ; Dành cho người dùng sử dụng mục đích chung

RS1 PSW.4 ; Bit = 1 chọn băng thanh ghi
RS0 PSW.3 ; Bit = 0 chọn băng thanh ghi
OV PSW.2 ; Cờ bận

- PSW.1 ; Bit dành cho người dùng định nghĩa

P PSW.0 ; Cờ chẵn, lẻ. Thiết lập/ xoá bằng phần cứng mỗi chu kỳ lệnh báo tổng các số bit 1 trong thanh ghi A là chẵn/ lẻ.

RS1	RS0	Băng thanh ghi	Địa chỉ
0	0	0	00H - 07H
0	1	1	08H - 0FH
1	0	2	10H - 17H
1	1	3	18H - 1FH

H×nh 2.4: **C¸c bit cña thanh ghi PSW**

Dưới đây là giải thích ngắn gọn về 4 bit cờ của thanh ghi PSW.

1. Cờ nhớ CY: Cờ này được thiết lập mỗi khi có nhớ từ bit D7. Cờ này được tác động sau lệnh cộng hoặc trừ 8 bit. Nó cũng được thiết lập lên 1 hoặc xoá về 0 trực tiếp bằng lệnh "SETB C" và "CLR C" nghĩa là "thiết lập cờ nhớ" và "xoá cờ nhớ" tương ứng. Về các lệnh đánh địa chỉ theo bit được bàn kỹ ở chương 8.
2. Cờ AC: Cờ này báo có nhớ từ bit D3 sang D4 trong phép cộng ADD hoặc trừ SUB. Cờ này được dùng bởi các lệnh thực thi phép số học mã BCD (xem ở chương 6).
3. Cờ chẵn lẻ P: Cờ chẵn lẻ chỉ phản ánh số bit một trong thanh ghi A là chẵn hay lẻ. Nếu thanh ghi A chứa một số chẵn các bit một thì P = 0. Do vậy, P = 1 nếu A có một số lẻ các bit một.
4. Cờ chàn OV: Cờ này được thiết lập mỗi khi kết quả của một phép tính số có dấu quá lớn tạo ra bit bậc cao làm tràn bit dấu. Nhìn chung cờ nhớ được dùng để phát hiện lỗi trong các phép số học không dấu. Còn cờ tràn được dùng chỉ để phát hiện lỗi trong các phép số học có dấu và được bàn kỹ ở chương 6.

2.6.2 Lệnh ADD và PSW.

Bây giờ ta xét tác động của lệnh ADD lên các bit CY, AC và P của thanh ghi PSW. Một số ví dụ sẽ làm rõ trạng thái của chúng, mặc dù các bit cờ bị tác động bởi lệnh ADD là CY, P, AC và OV nhưng ta chỉ tập trung vào các cờ CY, AC và P, còn cờ OV sẽ được nói đến ở chương 6 vì nó liên quan đến phép tính số học số có dấu.

Các ví dụ 2.2 đến 2.4 sẽ phản ánh tác động của lệnh ADD lên các bit nói trên.

Bảng 2.1: Các lệnh tác động lên các bit cờ.

Instruction	CY	OV	AC
ADD	X	X	X
ADDC	X	X	X
SUBB	X	X	X
MUL	0	X	
DIV	0	X	
DA	X		
RRC	X		
RLC	X		
SETB C	1		
CLR C	0		
CPL C	X		
ANL C, bit	X		
ANL C,/ bit	X		
ORL C, bit	X		
ORL C,/bit	X		
MOV C, bit	X		
CJNE	X		

Ví dụ 2.2: Hãy trình bày trạng thái các bit cờ CY, AC và P sau lệnh cộng 38H với 2FH dưới đây:

```
MOV     A, #38H
ADD  A, #2FH              ; Sau khi cộng A = 67H, CY = 0
```

Lêi gi¶i:

```
     38        00111000
+    2F        00101111
-------------------------
     67        01100111
```

Cờ CY = 0 vì không có nhớ từ D7
Cờ AC = 1 vì có nhớ từ D3 sang D4
Cờ P = 1 vì thanh ghi A có 5 bit 1 (lẻ)

VÝ dô 2.3:

Hãy trình bày trạng thái các cờ CY, AC và P sau phép cộng 9CH với 64H.

Lêi gi¶i:

```
     9C        10011100
+    64        01100100
-------------------------
    100        00000000
```

Cờ CY = 1 vì có nhớ qua bit D7
Cờ AC = 1 vì có nhớ từ D3 sang D4
Cờ P = 0 vì thanh ghi A không có bit 1 nào (chẵn)

VÝ dô 2.4:

Hãy trình bày trạng thái các cờ CY, AC và P sau phép cộng 88H với 93H.

Lêi gi¶i:

```
     88        10001000
+    93        10010011
-------------------------
    11B        00011011
```

Cờ CY = 1 vì có nhớ từ bit D7
Cờ AC = 0 vì không có nhớ từ D3 sang D4
Cờ P = 0 vì số bit 1 trong A là 4 (chẵn)

2.7 Các băng thanh ghi và ngăn xếp của 8051.

Bộ vi điều khiển 8051 có tất cả 128 byte RAM. Trong mục này ta bàn vệ phân bố của 128 byte RAM này và khảo sát công dụng của chúng như các thanh ghi và ngăn xếp.

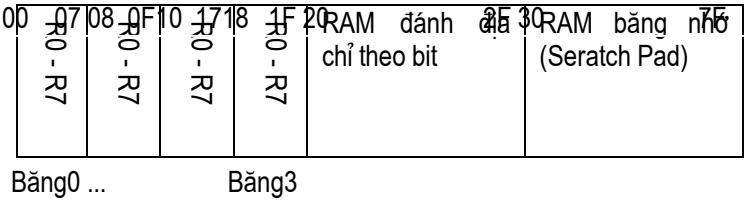
2.7.1 Phân bố không gian bộ nhớ RAM trong 8051.

Có 128 byte RAM trong 8051 (một số thành viên đang chú ý là 8052 có 256 byte RAM). 128 byte RAM bên trong 8051 được gán địa chỉ từ 00 đến 7FH. Như ta sẽ thấy ở chương 5, chúng có thể được truy cập trực tiếp như các ngăn nhớ 128 byte RAM này được phân chia thành từng nhóm như sau:

1. Tổng cộng 32 byte từ ngăn nhớ 00 đến 1FH được dành cho các thanh ghi và ngăn xếp.
2. Tổng cộng 16 byte từ ngăn nhớ 20H đến 2FH được dành cho bộ nhớ đọc/ ghi đánh địa chỉ được theo bit. Chương 8 sẽ bàn chi tiết về bộ nhớ và các lệnh đánh địa chỉ được theo bit.
3. Tổng cộng 80 byte từ ngăn nhớ 30H đến 7FH được dùng cho lưu đọc và ghi hay như vẫn thường gọi là bảng nháp (Serach pad). Những ngăn nhớ này (80 byte) của RAM được sử dụng rộng rãi cho mục đích lưu dữ liệu và tham số bởi các lập trình viên 8051. Chúng ta sẽ sử dụng chúng ở các chương sau để lưu dữ liệu nhận vào CPU qua các cổng vào-ra.

2.7.2 Các băng thanh ghi trong 8051.

Như đã nói ở trước, tổng cộng 32 byte RAM được dành riêng cho các băng thanh ghi và ngăn xếp. 32 byte này được chia ra thành 4 băng các thanh ghi trong đó mỗi băng có 8 thanh ghi từ R0 đến R7. Các ngăn nhớ RAM số 0, R1 là ngăn nhớ RAM số 1, R2 là ngăn nhớ RAM số 2 v.v... Băng thứ hai của các thanh ghi R0 đến R7 bắt đầu từ thanh nhớ RAM số 2 cho đến ngăn nhớ RAM số 0FH. Băng thứ ba bắt đầu từ ngăn nhớ 10H đến 17H và cuối cùng từ ngăn nhớ 18H đến 1FH là dùng cho băng các thanh ghi R0 đến R7 thứ tư.



Hình 2.5: Ngăn xếp các thanh nhớ RAM trong 8051.

Bank 0		Bank 1		Bank 2		Bank 3	
7	R7	F	R7	17	R7	1F	R7
6	R6	E	R6	16	R6	1E	R6
5	R5	D	R5	15	R5	1D	R5
4	R4	C	R4	14	R4	1C	R4
3	R3	B	R3	13	R3	1B	R3
2	R2	A	R2	12	R2	1A	R2
1	R1	9	R1	11	R1	19	R1
0	R0	8	R0	10	R0	18	R0

Hình 2.6: Các băng thanh ghi của 8051 và địa chỉ của chúng.

Như ta có thể nhìn thấy từ hình 2.5 băng 1 sử dụng cùng không gian RAM như ngăn xếp. Đây là một vấn đề chính trong lập trình 8051. Chúng ta phải hoặc là không sử dụng băng 1 hoặc là phải đánh một không gian khác của RAM cho ngăn xếp.

VÝ dô 2.5:

Hãy phát biểu các nội dung của các ngăn nhớ RAM sau đoạn chương trình sau:

```
MOV    R0, #99H          ; Nạp R0 giá trị 99H
MOV    R1, #85H          ; Nạp R1 giá trị 85H
MOV    R2, #3FH          ; Nạp R2 giá trị 3FH
MOV    R7, #63H          ; Nạp R7giá trị 63H
MOV    R5, #12H          ; Nạp R5 giá trị12H
```

Lêi gi¶i:

Sau khi thực hiện chương trình trên ta có:

Ngăn nhớ 0 của RAM có giá trị 99H

Ngăn nhớ 1 của RAM có giá trị 85H

Ngăn nhớ 2 của RAM có giá trị 3FH

Ngăn nhớ 7 của RAM có giá trị 63H

Ngăn nhớ 5 của RAM có giá trị 12H

2.6.3 Băng thanh ghi mặc định.

Nếu các ngăn nhớ 00 đến 1F được dành riêng cho bốn băng thanh ghi, vậy băng thanh ghi R0 đến R7 nào ta phải truy cập tới khi 8051 được cấp nguồn? Câu trả lời là các băng thanh ghi 0. Đó là các ngăn nhớ RAM số 0, 1, 2, 3, 4, 5, 6 và 7 được truy cập với tên R0, R1, R2, R3, R4, R5, R6 và R7 khi lập trình 8051. Nó dễ dàng hơn nhiều khi tham chiếu các ngăn nhớ RAM này ơví các tên R0, R1 v.v... hơn là số vị trí của các ngăn nhớ. Ví dụ 2.6 làm rõ khái niệm này.

VÝ dô 2.6:

Hãy viết lại chương trình ở ví dụ 2.5 sử dụng các địa chỉ RAM thay tên các thanh ghi.

Lêi gi¶i:

Đây được gọi là chế độ đánh địa chỉ trực tiếp và sử dụng địa chỉ các vị trí ngăn nhớ RAM đối với địa chỉ đích. Xem chi tiết ở chương 5 về chế độ đánh địa chỉ.

```
MOV    00, #99H          ; Nạp thanh ghi R0 giá trị 99H
MOV    01, #85H          ; Nạp thanh ghi R1 giá trị 85H
MOV    02, #3FH          ; Nạp thanh ghi R2 giá trị 3FH
MOV    07, #63H          ; Nạp thanh ghi R7giá trị 63H
MOV    05, #12H          ; Nạp thanh ghi R5 giá trị12H
```

2.6.4 Chuyển mạch các băng thanh ghi như thế nào?

Như đã nói ở trên, băng thanh ghi 0 là mặc định khi 8051 được cấp nguồn. Chúng ta có thể chuyển mạch sang các băng thanh ghi khác bằng cách sử dụng bit D3 và D4 của thanh ghi PSW như chỉ ra theo bảng 2.2.

Bảng 2.2: Bit lựa chọn các băng thanh ghi RS0 và RS1.

	RS1 (PSW.4)	RS0 (PSW.3)
Băng 0	0	0
Băng 1	0	1
Băng 2	1	0
Băng 3	1	1

Bit D3 và D4 của thanh ghi PSW thường được tham chiếu như là PSW.3 và PSW.4 vì chúng có thể được truy cập bằng các lệnh đánh địa chỉ theo bit như SETB và CLR. Ví dụ "SETB PSW.3" sẽ thiết lập PSW.3 và chọn băng thanh ghi 1. Xem ví dụ 2.7 dưới đây.

VÝ dô 2.7:

Hãy phát biểu nội dung các ngăn nhớ RAM sau đoạn chương trình dưới đây:

```
SETB    PSW.4           ; Chọn băng thanh ghi 4
MOV     R0, #99H        ; Nạp thanh ghi R0 giá trị 99H
MOV     R1, #85H        ; Nạp thanh ghi R1 giá trị 85H
MOV     R2, #3FH        ; Nạp thanh ghi R2 giá trị 3FH
MOV     R7, #63H        ; Nạp thanh ghi R7giá trị 63H
MOV     R5, #12H        ; Nạp thanh ghi R5 giá trị12H
```

Lêi gi¶i:

Theo mặc định PSW.3 = 0 và PSW.4 = 0. Do vậy, lệnh "SETB PSW.4" sẽ bật bit RS1 = 1 và RS0 = 0, bằng lệnh như vậy băng thanh ghi R0 đến R7 số 2 được chọn. Băng 2 sử dụng các ngăn nhớ từ 10H đến 17H. Nên sau khi thực hiện đoạn chương trình trên ta có nội dung các ngăn nhớ như sau:

Ngăn nhớ vị trí 10H có giá trị 99H
Ngăn nhớ vị trí 11H có giá trị 85H
Ngăn nhớ vị trí 12H có giá trị 3FH
Ngăn nhớ vị trí 17H có giá trị 63H
Ngăn nhớ vị trí 15H có giá trị 12H

2.6.5 Ngăn xếp trong 8051.

Ngăn xếp là một vùng bộ nhớ RAM được CPU sử dụng để lưu thông tin tạm thời. Thông tin này có thể là dự liệu, có thể là địa cỉ CPU cần không gian lưu trữ này vì số các thanh ghi bị hạn chế.

2.6.6 Cách truy cập các ngăn xếp trong 8051.

Nếu ngăn xếp là một vùng của bộ nhớ RAM thì phải có các thanh ghi trong CPU chỉ đến nó. Thanh được dùng để chỉ đến ngăn xếp được gọi là thanh ghi con trỏ ngăn xếp SP (Stack Pointer). Con trỏ ngăn xếp trong 8051 chỉ rộng 8 bit có nghĩa là nó chỉ có thể có thể được các địa chỉ từ 00 đến FFH.

Khi 8051 được cấp nguồn thì SP chứa giá trị 07 có nghĩa là ngăn nhớ 08 của RAM là ngăn nhớ đầu tiên được dùng cho ngăn xếp trong 8051. Việc lưu lại một thanh ghi

PCU trong ngăn xếp được gọi là một lần cất vào PUSH và việc nạp nội dung của ngăn xếp trở lại thanh ghi CPU được gọi là lấy ra POP. Hay nói cách khác là một thanh ghi được cất vào ngăn xếp để lưu cất và được lấy ra từ ngăn xếp để dùng tiếp công việc của SP là rất nghiêm ngặt mỗi khi thao tác cất vào (PUSH) và lấy ra (POP) được thực thi. Để biết ngăn xếp làm việc như thế nào hãy xét các lệnh PUSH và POP dưới đây.

2.6.7 Cất thanh ghi vào ngăn xếp.

Trong 8051 thì con trỏ ngăn xếp chỉ đến ngăn nhớ sử dụng cuối cùng của ngăn xếp. Khi ta cất dữ liệu vào ngăn xếp thì con trỏ ngăn xếp SP được tăng lên 1. Lưu ý rằng điều này đối với các bộ vi xử lý khác nhau là khác nhau, đáng chú ý là các bộ vi xử lý × 86 là SP giảm xuống khi cất dữ liệu vào ngăn xếp. Xét ví dụ 2.8 dưới đây, ta thấy rằng mỗi khi lệnh PUSH được thực hiện thì nội dung của thanh ghi được cất vào ngăn xếp và SP được tăng lên 1. Lưu ý là đối với mỗi byte của dữ liệu được cất vào ngăn xếp thì SP được tăng lên 1 lần. Cũng lưu ý rằng để cất các thanh ghi vào ngăn xếp ta phải sử dụng địa chỉ RAM của chúng. Ví dụ lệnh "PUSH 1" là cất thanh ghi R1 vào ngăn xếp.

VÝ dô 2.8:

Hãy biểu diễn ngăn xếp và con trỏ ngăn xếp đối với đoạn chương trình sau đây. Giả thiết vùng ngăn xếp là mặc định.

```
MOV     R6, #25H
MOV     R1, #12H
MOV     R4, #0F3H
PUSH    6
PUSH    1
PUSH    4
```

Lêi gi¶i:

	Sau PUSH 6	Sau PUSP 1	Sau PUSH 4
0B	0B	0B	0B
0A	0A	0A	0A F3
09	09	09 12	09 12
08	08 25	08 25	08 25
Bắt đầu SP = 07	SP = 08	SP = 09	SP = 0A

2.6.8 Lấy nôi dung thanh ghi ra từ ngăn xếp.

Việc lấy nội dung ra từ ngăn xếp trở lai thanh ghi đã cho là qúa trình ngược với các nội dung thanh ghi vào ngăn xếp. Với mỗi lần lấy ra thì byte trên đỉnh ngăn xếp được sao chép vào thanh ghi được xác định bởi lệnh và con trỏ ngăn xếp được giảm xuống 1. Ví dụ 2.9 minh hoạ lệnh lấy nội dung ra khỏi ngăn xếp.

VÝ dô 2.9:

Khảo sát ngăn xếp và hãy trình bày nội dung của các thanh ghi và SP sau khi thực hiện đoạn chương trình sau đây:

```
POP     3       ; Lấy ngăn xếp trở lại R3
POP     5       ; Lấy ngăn xếp trở lại R5
```

```
POP     2          ; Lấy ngăn xếp trở lại R2
```

Lêi gi¶i:

		Sau POP3		Sau POP 5		Sau POP 2	
0B	54	0B		0B		0B	
0A	F9	0A	F9	0A		0A	
09	76	09	76	09	76	09	
08	6C	08	6C	08	6C	08	6C
Bắt đầu SP = 0B		SP = 0A		SP = 09		SP = 08	

2.6.9 Giới hạn trên của ngăn xếp.

Như đã nói ở trên, các ngăn nhớ 08 đến 1FH của RAM trong 8051 có thể được dùng làm ngăn nhớ 20H đến 2FH của RAM được dự phòng cho bộ nhớ đánh địa chỉ được theo bit và không thể dùng trước cho ngăn xếp. Nếu trong một chương trình đã cho ta cần ngăn xếp nhiều hơn 24 byte (08 đến 1FH = 24 byte) thì ta có thể đổi SP chỉ đến các ngăn nhớ 30 đến 7FH. Điều này được thực hiển bởi lệnh "MOV SP, #XX".

2.6.10 Lệnh gọi CALL và ngăn xếp.

Ngoài việc sử dụng ngăn xếp để lưu cất các thanh ghi thì CPU cũng sử dụng ngăn xếp để lưu cất tam thời địa chỉ của lệnh đứng ngay dưới lệnh CALL. Điều này chính là để PCU biết chỗ nào để quay trở về thực hiện tiếp các lệnh sau khi chọn chương trình con. Chi tiết về lệnh gọi CALL được trình bỳ ở chương 3.

2.6.11 Xung đột ngăn xếp và băng thanh ghi số 1.

Như ta đa nói ở trên thì thanh ghi con trỏ ngăn xếp có thể chỉ đến vị trí RAM hiện thời dành cho ngăn xếp. Khi dữ liệu được lưu cất cào ngăn xếp thì SP được tăng lên và ngược lại khi dữ liệu được lấy ra từ ngăn xếp thì SP giảm xuống. Lý do là PS được tăng lên sau khi PUSH là phải biết lấy chắc chắn rằng ngăn xếp đang tăng lên đến vị trí ngăn nhớ 7FH của RAM từ địa chỉ thấp nhất đến địa chỉ cao nhất. Nếu con trỏ ngăn xếp đã được giảm sau các lệnh PUSH thì ta nên sử dụng các ngăn nhớ 7, 6, 5 v.v... của RAM thuộc các thanh ghi R7 đến R0 của băng 0, băng thanh ghi mặc định. Việc tăng này của con trỏ ngăn xếp đối với các lệnh PUSH cũng đảm bảo rằng ngăn xếp sẽ không với tới ngăn nhớ 0 của RAM (đáy của RAM) và do vậy sẽ nhảy ra khỏi không gian dành cho ngăn xếp. Tuy nhiên có vấn đề nảy sinh với thiết lập mặc định của ngăn xếp. Ví dụ SP = 07 khi 8051 được bật nguồn nên RAM và cũng thuộc về thanh ghi R0 củ băng thanh ghi số 1. Hay nói cách khác băng thanh ghi số 1 và ngăn xếp đang dùng chung một không gian của bộ nhớ RAM. Nếu chương trình đã cho cần sử dụng các băng thanh ghi số 1 và số 2 ta có thể đặt lại vùng nhớ RAM cho ngăn xếp. Ví dụ, ta có thể cấp vị trí ngăn nhớ 60H của RAM và cao hơn cho ngăn xếp trong ví dụ 2.10.

VÝ dô 2.10:

Biểu diễn ngăn xếp và con trỏ ngăn xếp đối với các lệnh sau:

```
MOV     SP, #5FH            ; Đặt ngăn nhớ từ 60H của RAM cho ngăn xếp
MOV     R2, #25H
MOV     R1, #12H
MOV     R4, #0F3H
PUSH    2
```

PUSH 1
PUSH 4

Lêi gi¶i:

	Sau PUSH 2	Sau PUSP 3	Sau PUSH 4
63	63	63	63
62	62	62	62 F3
61	61	61 12	61 12
60	60 25	60 25	60 25
Bắt đầuSP=5F	SP = 60	SP = 61	SP = 62