Chữa bài tập vi xử lí

Bùi Nhật Minh

Ngày 17 tháng 11 năm 2025

Bài 1: Trình bày kiến trúc và nguyên lí hoạt động của máy tính theo Von Neumann.
Lời giải bài 1:

Kiến trúc Von Neumann gồm các thành phần sau:

- Bộ xử lí trung tâm (CPU - Central Processing Unit), bên trong có đơn vị điều khiển (CU - Control Unit), đơn vị tính toán số học và lô-gíc (ALU - Arithmetic Logic Unit) và các thanh ghi;
- Bộ nhớ;
- Thiết bị vào/ra.

Các bộ phận này được kết nối với nhau bởi các đường truyền hệ thống (bus): dữ liệu, địa chỉ, và điều khiển.

Về mặt nguyên lí hoạt động, chương trình và dữ liệu được giữ chung ở cùng một bộ nhớ. Thông qua các đường truyền, lệnh và dữ liệu được chuyển vào và được thực hiện bởi bộ xử lí trung tâm. Trong trường hợp cần thiết, các bộ phận vào/ra có thể tương tác với bộ xử lí trung tâm hay trực tiếp với bộ nhớ, cũng qua các đường truyền.

Bài 2: Nêu vai trò của CPU và bộ nhớ trong máy tính theo hệ kiến trúc Von Neumann.
Lời giải bài 2:

Bộ xử lí trung tâm (CPU), theo đúng tên của nó, chịu trách nhiệm tiếp nhận và xử lí dữ liệu; điều khiển các hoạt động bên trong máy tính.

Bộ nhớ có vai trò ghi nhớ chương trình và dữ liệu phục vụ việc chạy và trong quá trình chạy chương trình.

Bài 3: Mô tả chi tiết hoạt động thực hiện một lệnh của CPU. Nêu cơ chế mà nhờ đó CPU có thể thực hiện một cách tuần tự và nguyên lí mà CPU có thể thực hiện rẽ nhánh có điều kiện.
Lời giải bài 3:

Nhờ sự kết hợp của đơn vị điều khiển và bộ đếm chương trình (PC - Program Counter), CPU có thể thực hiện lệnh một cách tuần tự như sau:

1. Tìm lệnh (Fetch): Qua bộ giải mã địa chỉ với PC làm địa chỉ, CPU lấy giá trị lưu trong ô nhớ có địa chỉ tương ứng. CPU cần phải nhận biết xem giá trị này là lệnh hay là dữ liệu. Nếu là lệnh, đưa giá trị này vào thanh ghi lệnh. Nếu là dữ liệu, đưa vào những thanh ghi nhớ phục vụ những lệnh sau.
2. Giải mã lệnh (Decode): Từ lệnh trong thanh ghi lệnh, cho nó đi qua một mạch giải mã để chuyển đổi từ mã lệnh thành lệnh.
3. Thực hiện lệnh (Execute): Từ kết quả của mạch giải mã, CPU thực hiện lệnh.
4. Lưu trữ (Store); Kết quả sau khi thực hiện lệnh sẽ được lưu ở trong bộ nhớ hoặc trong một thanh ghi. PC tự động tăng thêm 1 (nếu không phải là lệnh rẽ nhánh), qua đó đảm bảo tính tuần tự.

Tóm lại, CPU nhận lệnh và dữ liệu từ bộ nhớ với địa chỉ là PC thông qua mạch giải mã. Sau khi kết thúc lệnh, PC tự động tăng thêm 1, thể hiện tính tuần tự.

Như đã nhắc đến, trong trường hợp mạch giải mã lệnh ra lệnh rẽ nhánh thì CPU sẽ thay đổi giá trị PC đến vị trí của lệnh tiếp theo thay vì cho PC làm tuần tự như bình thường.

Bài 4: Nêu rõ vì sao có thể nối chung đường truyền (bus) số liệu trong hệ thống vi xử lí.
Lời giải bài 4:

Các bus số liệu trong hệ thống có thể nối chung với nhau **mà không bị xung đột (hai đơn vị dùng chung bus cùng một lúc, hoặc vừa dùng đường dữ liệu để truyền ra và nhập vào trên cùng một đơn vị)** vì bộ điều khiển phát xung tín hiệu điều khiển, chỉ cho phép những đơn vị cần thiết để thực hiện lệnh tương tác với đường dữ liệu và chỉ cho phép xuất hoặc nhập trong một thời điểm nhất định. Điều này làm tiết kiệm đường truyền dữ liệu và có thể tích hợp xuất/nhập trên cùng đường truyền.

Bài 5: Trình bày rõ vì sao CPU có thể truy nhập theo địa chỉ.
Lời giải bài 5:

CPU có thể truy cập theo địa chỉ bởi vì trong kiến trúc máy tính, mọi ô nhớ và cổng vào/ra **chỉ được gán một địa chỉ duy nhất**. Từ đó, CPU đặt các địa chỉ ô nhớ lên các đường truyền địa chỉ, và ô nhớ hay thiết bị vào/ra có đúng địa chỉ đó sẽ phản hồi và tương tác với CPU.

Bài 6: Anh/chị hiểu thế nào về câu: "CPU truy nhập cổng vào/ra như một ô nhớ và truy nhập ô nhớ nhờ cổng vào ra."?

Lời giải bài 6:

Trong một số kiến trúc, địa chỉ vào/ra được gán địa chỉ trong không gian bộ nhớ[1]. CPU nếu muốn đọc hay ghi vào thiết bị ngoại vi, chỉ cần truy cập vào vùng địa chỉ xác định trong bộ nhớ, giống như truy cập vào bộ nhớ thông thường.

Ngoài ra, trong một vài trường hợp khác, CPU không có khả năng truy cập trực tiếp vào bộ nhớ (hay xảy ra khi phải ghi và nhận vào dữ liệu từ bộ nhớ ngoài như trong cơ chế DMA[2] hay thay đổi VRAM). Khi này, CPU phải truyền lệnh qua các cổng vào/ra để thực hiện xuất và nhập dữ liệu cần thiết.

Bài 7: Nêu cơ chế ngắt trong hệ vi xử lí và mục đích của cơ chế này. Hệ vi xử lí thực hiện ngắt như thế nào? Chỉ rõ ưu, khuyết điểm của cơ chế ngắt.

Lời giải bài 7:

Trong hệ vi xử lí, ngắt là hoạt động tạm dừng chương trình bình thường để thực thi chương trình ngắt, với mục đích chính là để kịp thời phản ứng với sự kiện bên ngoài (chủ yếu là tác động vào/ra).

Chu trình ngắt cua hệ vi xử lí được thực hiện như sau:

1. Khi vi xử lí nhận và chấp nhận báo ngắt, hoàn thành nốt lệnh hiện tại.
2. Sau đó, lưu trạng thái hiện tại (còn được gọi là "ngữ cảnh", bao gồm bộ đếm chương trình, các thanh ghi đặc biệt, ghi cờ, v.v) vào ngăn xếp.
3. Vi xử lí sau khi lưu ngữ cảnh chuyển sang thực hiện chương trình phục vụ ngắt.
4. Hoàn thành chương trình phục vụ ngắt, vi xử lí khôi phục ngữ cảnh từ ngăn xếp, tiếp tục thực hiện chương trình bình thường.

Ưu điểm của cơ chế ngắt bao gồm:

•

Nhược điểm của cơ chế ngắt bao gồm:

•

Bài 8: Trình bày cách xác định nguồn báo ngắt bằng phần mềm, xác định bằng phần cứng.

Lời giải bài 8:

Bài 9: Trong tập lệnh của MCS-51 không có lệnh `ORG` và `END`. Tại sao trong một số chương trình, ta lại có thể thấy hai lệnh này?

Lời giải bài 9:

Đây không phải là lệnh dành cho vi xử lí mà là lệnh dành cho trình biên dịch.

Bài 10: Trong những thanh ghi có chức năng đặc biệt của 8051, có thanh ghi DPTR. Giải thích chức năng và cách hoạt động của thanh ghi này. Làm sao để biết DPTR đang truy cập đến vùng nhớ nào?

Lời giải bài 10:

DPTR (Data Pointer) là thanh ghi 16-bit dùng để chỉ đến byte dữ liệu trong bộ nhớ RAM ngoài hoặc/và bộ nhớ ROM. DPTR không có một địa chỉ đơn, nhưng người lập trình có thể viết vào DPTR như trong lệnh sau:

```
MOV DPTR, #1234H
```

Giá trị trong thanh ghi DPTR không xác định được bộ nhớ đang sử dụng là kiểu bộ nhớ nào. Để biết kiểu bộ nhớ đang dùng, cần phụ thuộc vào câu lệnh sử dụng thanh ghi DPTR. Ví dụ:

```
MOVX A,@DPTR                    ; viết dữ liệu từ RAM ngoài vào A
MOVC A,@DPTR                    ; viết dữ liệu từ ROM vào A
```

Bài 11: Ta có thể truy cập vào băng thanh ghi nào khi 8051 vừa được cấp nguồn, vì sao, và làm thế nào để thay đổi nó?

Lời giải bài 11:

Băng 0, do khi vừa cấp nguồn, cả hai giá trị bit RS0 và RS1 của thanh ghi PSW đều được khởi tạo bằng 0. Để thay đổi băng, ta cần thay đổi giá trị bit RS0 và RS1 (bit D3 và D4) của thanh ghi PSW như trong lệnh sau (thay đổi sang băng 1):

[1] Ví dụ, kiến trúc Von Neumann, khi tất cả bộ nhớ chương trình, bộ nhớ dữ liệu, và địa chỉ vào ra được dự trữ ở cùng một "bộ nhớ".

[2] Direct Memory Access. CPU ghi các thông số cần thiết và gửi lên các thanh ghi điều khiển của DMA. Sau đó, DMA tự động điều khiển bus để truyền dữ liệu.

```
CLR PSW.4
SETB PSW.3
```

Bài 12: Viết chương trình sử dụng 8051 để tạo xung vuông có tần số 100 kHz ở cổng P1.7. Cho biết rằng tần số dao động của xung tạo từ mạch thạch anh là 11,0592 MHz.

Lời giải bài 12:

Ta có chương trình như sau:

```
ORG 0
SJMP START
ORG 0040H
PULSE:
    SETB P1.7
    NOP
    NOP
    NOP
    NOP
    CLR P1.7
    NOP
    NOP
    AJMP PULSE
END
```

Thời gian P1.7 được kích ở mức lô-gíc 1 được tính như sau:

- `SETB P1.7`: 12 chu kì;
- Thời gian tạm nghỉ `NOP`: 12×4 chu kì.

Thời gian P1.7 được kích ở mức lô-gíc 1 là

$$12 + 12 \times 4 = 60$$

chu kì mạch dao động thạch anh. Thời gian P1.7 được kích ở mức lô-gíc 0 được tính như sau:

- `CLR P1.7`: 12 chu kì;
- Thời gian tạm nghỉ `NOP`: 12×2 chu kì;
- `AJMP PULSE`: 24 chu kì.

Thời gian P1.7 được kích ở mức lô-gíc 0 là $12 + 12 \times 2 + 24 = 60$ chu kì mạch dao động thạch anh.

Qua đó, chúng ta có chu kì của dao động là $60 + 60 = 120$ chu kì mạch dao động thạch anh. Tần số của dao động này bằng:

$$\frac{1}{120 \times \frac{1}{11{,}0592 \times 10^6 \text{ Hz}}} = 92160 \text{ Hz}.$$

Sở dĩ chúng ta không thể làm tốt hơn là do một lệnh của 8051 chạy với thời gian là bội số của 12 lần dao động. Nếu mỗi nửa chu kì một lệnh `NOP` thì tần số dao động sẽ là

$$\frac{1}{(120 - 12 \times 2) \times \frac{1}{11{,}0592 \times 10^6 \text{ Hz}}} = 115200 \text{ Hz}.$$

Bài 13: Khi người ta đề cập đến vùng bộ nhớ có thể truy cập theo bít của 8051, vùng bộ nhớ này bắt đầu từ địa chỉ byte nào và đến địa chỉ byte nào? Ngoài ra, còn ở những đâu có thể truy cập theo từng bít (trả lời ngắn gọn). Viết câu lệnh đặt bít thứ 5 (đếm bít từ vị trí thứ 0) của byte có địa chỉ $2BH$ thành giá trị 1.

Lời giải bài 13:

Vùng bộ nhớ có thể truy cập theo bít của 8051 bắt đầu từ địa chỉ byte $20H$ đến địa chỉ byte $2FH$. Bít truy cập theo địa chỉ được gán địa chỉ từ $00H$ đến $7FH$. Ngoài ra, còn có thể truy cập theo từng bít trong một số thanh ghi có chức năng đặc biệt.

Các bít trong vùng bộ nhớ có thể truy cập theo bít có địa chỉ riêng từ $00H$ đến $7FH$. Công thức để tính vị trí của bít là:

$$\text{Địa chỉ của bit} = (\text{Địa chỉ của byte} - 20H) \times 8 + \text{Vị trí của bit trong byte}.$$

Câu lệnh đặt bít thứ 5 (đếm bít từ vị trí thứ 0) của byte có địa chỉ $2BH$ thành giá trị 1 là:

```
SETB 5DH
```

Bài 14: Khi ngăn xếp không có dữ liệu, thanh ghi địa chỉ ngăn xếp (SP) chỉ vào vùng có địa chỉ bao nhiêu trong bộ nhớ? Viết lệnh để đặt giá trị $50H$ vào ngăn xếp. Với địa chỉ ban đầu đó, giá trị mới này được lưu vào đâu?

Lời giải bài 14:

Khi ngăn xếp không có dữ liệu, SP chỉ vào vùng có địa chỉ $07H$. Viết lệnh để đặt giá trị $50H$ vào ngăn xếp là:

```
MOV 50H, #50H
PUSH 50H
```

Với lệnh `PUSH`, giá trị của SP sẽ tăng lên 1 trước khi đưa dữ liệu vào ngăn xếp. Vậy giá trị của $50H$ được lưu trong thanh ghi $08H$.

Bài 15: Viết lệnh để đặt giá trị của A vào ngăn xếp sử dụng phương pháp truy địa chỉ trực tiếp. Tại sao trong một số trường hợp, có thể thay thế lệnh này bằng các lệnh như `PUSH ACC` hay `PUSH A`, mặc dù trong tập lệnh không có lệnh đó?

Lời giải bài 15:

Lệnh để đặt giá trị của A vào ngăn xếp là

```
PUSH 80H
```

Có thể thay thế được bằng các lệnh như đề bài là do trình biên dịch (assembler) đã hiểu rằng `ACC` hay `A` ám chỉ thanh ghi địa chỉ $80H$ và biên dịch hộ người lập trình.