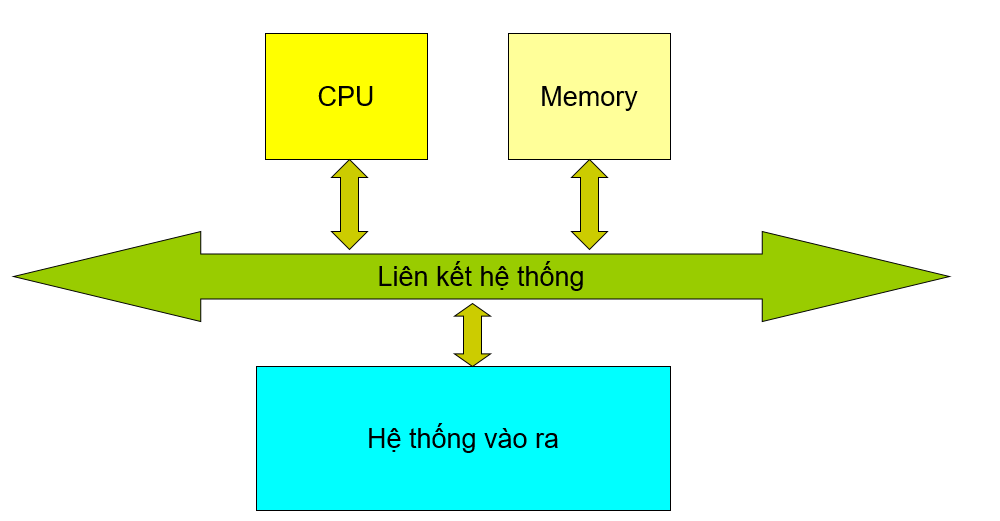
1. Tổng quan về kiến trúc máy tính
   1. Phân biệt kiến trúc máy tính Von Neumann và Non Von Neumann.

|  |  |  |
| --- | --- | --- |
|  | Von Neumann (cuối năm 1946) | Non Von Neumann.( trước 1946) |
| Hệ thống xử lý | Có | Có |
| Hệ thống nhập dữ liệu | Có | Có |
| Hệ thống xuất dư liệu | Có | Có |
| Hệ thống BUS | Có | Có |
| Hệ thống nhớ | Có | Không |

* 1. Vẽ sơ đồ khối cấu trúc chung của máy tính IBM-PC và trình bày tóm tắt chức năng các khối.

**Bài làm:**

**Chức năng CPU:**

* + điều khiển hoạt động của máy tính
  + xử lý dữ liệu

**Chức năng bộ nhớ trong:**

Chứa các thông tin mà CPU có thể trao đổi trực tiếp

Tốc độ rất nhanh

Dung lượng không lớn

Sử dụng bộ nhớ bán dẫn: ROM và RAM

**Chức năng hệ thống vào/ra:** Trao đổi thông tin giữa máy tính với thế giới bên ngoài.

**Chức năng bus liên kết hệ thống:** dùng trao đổi thông tin giữa CPU và bộ nhớ trong (thông qua cache)

* 1. Trình bày chức năng, đặc điểm của bus địa chỉ.

**Chức năng**: vận chuyển địa chỉ để xác định ngăn nhớ hay cổng vào/ra

**Độ rộng bus địa chỉ**: cho biết số lượng ngăn nhớ tối đa được đánh địa chỉ.

N bit: AN-1, AN-2, ... A2, A1, A0 → có thể đánh địa chỉ tối đa cho 2N ngăn nhớ (không gian địa chỉ bộ nhớ)

**Ví dụ**: Bộ xử lý Pentium có bus địa chỉ 32 bit có khả năng đánh địa chỉ cho 232 bytes nhớ (4GBytes) (ngăn nhớ tổ chức theo byte)

* 1. Cho biết ý nghĩa khi nói Bus địa chỉ có độ rộng 24 bit.

Bus gồm 24 đường dây dẫn, CPU có khả năng quản lý không gian nhớ là

2 mũ 24 = 16MB=4GB

* 1. Trình bày chức năng, đặc điểm của bus dữ liệu.

**Chức năng:**

* + vận chuyển lệnh từ bộ nhớ đến CPU
  + vận chuyển dữ liệu giữa CPU, module nhớ, module vào/ra với nhau

**Độ rộng bus dữ liệu**: Xác định số bit dữ liệu có thể được trao đổi đồng thời.

* + M bit: DM-1, DM-2, ... D2, D1, D0
  + M thường là 8, 16, 32, 64,128 bit.

**Ví dụ**: Các bộ xử lý Pentium có bus dữ liệu 64 bit

* 1. Cho biết ý nghĩa khi nói Bus dữ liệu có độ rộng 32 bit.

Bus gồm 32 đường dây dẫn, CPU có khả năng xử lý  toán hạng 32 bit trong 1 chu kỳ lệnh.

* 1. Trình bày chức năng, đặc điểm của bus điều khiển.

**Chức năng**: vận chuyển các tín hiệu điều khiển

**Các loại tín hiệu điều khiển bao gồm**:

-Các tín hiệu điều khiển đọc/ghi

-Các tín hiệu điều khiển ngắt

-Các tín hiệu điều khiển bus

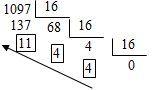
* 1. Trình bày hiểu biết của em về tín hiệu điều khiển đọc/ghi bộ nhớ.
  + *Memory Read* (MEMR): điều khiển đọc dữ liệu từ một ngăn nhớ có địa chỉ xác định lên bus dữ liệu.
  + *Memory Write* (MEMW): điều khiển ghi dữ liệu có sẵn trên bus dữ liệu đến một ngăn nhớ có địa chỉ xác định.
  1. Trình bày hiểu biết của em về tín hiệu điều khiển đọc/ghi thiết bị vào/ra.
  + *I/O Read* (IOR): điều khiển đọc dữ liệu từ một cổng vào-ra có địa chỉ xác định lên bus dữ liệu.
  + *I/O Write* (IOW): điều khiển ghi dữ liệu có sẵn trên bus dữ liệu ra một cổng có địa chỉ xác định.
  1. Trình bày 3 tín hiệu điều khiển ngắt cứng điển hình (NMI, INTR, INTA).
  + *Interrupt Request* (INTR): Tín hiệu từ bộ điều khiển vào-ra gửi đến yêu cầu ngắt CPU để trao đổi vào/ra.Tín hiệu INTR có thể bị che.
  + *Interrupt Acknowledge* (INTA): Tín hiệu phát ra từ CPU báo cho bộ điều khiển vào-ra biết CPU chấp nhận ngắt để trao đổi vào/ra.
  + *Non Maskable Interrupt* (NMI): tín hiệu ngắt không che được gửi đến ngắt CPU.

1. Kiến trúc tập lệnh
   1. Đổi số 1097.75 về hệ 2

***Bài làm:***

Đổi số 1097 về hệ 2.

* Đổi số 1097 về hệ 16:



Lấy kết quả theo chiều mũi tên. Ta được:

1097 = 44BH

* Đổi số 44BH về hệ 2 ta được: 010001001011B.
* Vậy 1097 = 10001001011B.

Đổi 0.75 về hệ 2.

0.75 x 2 = 1.5

0.5 x 2 = 1.0

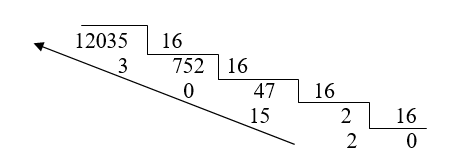
* Vậy 0.75 = 0.11B (kết quả: lấy phần nguyên của kết quả nhân từ theo thứ tự từ trên xuống dưới)

Ghép 2 phần, ta có: 1097.75 = 10001001011.11B

* 1. Đổi số 12035.125 về hệ 2

Đổi số 12035 về hệ 2.

Đổi số 10235 về hệ 16:



Lấy kết quả theo chiều mũi tên. Ta được: 10235= 2F03

Đổi số 2F03 về hệ 2 ta được:0010111100000011

Vậy 12035 =0010111100000011

Đổi 0.125 về hệ 2.

0.125 x 2 = 0.25

0.25 x 2 = 0.5

0.5 x 2 = 1.0

Vậy 0.125 = 0.001B

Ghép 2 phần, ta có:

12035.125 =0010111100000011.001B

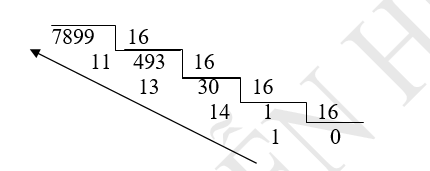
* 1. Đổi số 7899.8 về hệ 2

Mà 7899.8

Đổi số 7899.8125 về hệ 2

Đổi số 7899 về hệ 2.

Đổi số 10235 về hệ 16:



Lấy kết quả theo chiều mũi tên. Ta được: 7899= 1EDB

Đổi số 1EDC về hệ 2 ta được: B

Vậy 7899 = 0001111011011011B.

Đổi 0.8125về hệ 2.

0.8x 2 = 1.6

0.6 x 2 =1.2

0.2 x 2 = 0.4

0.4 x 2 =0.8

* 1. Đổi số 10110010.11B về hệ 10.

Gợi ý:

Đổi số Q hệ cơ số a về số P hệ cơ số 10: ta thực hiện theo công thức sau.

Q = mk-1 mk∙∙∙∙∙∙m1m0.m-1m-n+k

P = ak-1 × mk-1 + ak-2 × mk-2 + … + a1 × m1 + m0 + a-1 × m-1 + … + a-n+k × m-n+k (2.1)

Ta có thể thực hiện theo 2 cách:

*Cách 1: Đổi số đã cho qua số hệ 16, rồi đổi tiếp từ số hệ 16 về hệ 2*

* Đổi số 10110010.11B về hệ 16, ta có:

10110010.11B = 10110010.1100B = 0B2.BH

* Đổi số 0B2.BH về hệ 10 theo công thức (2.1), ta có:

11 \* 161 + 2 \* 160 + 11 \* 16-1*-1* = = 176 + 2 + 0.75 = 178.75

*Cách 2: Đổi số đã cho thẳng sang số hệ 10 (theo công thức 2.1)*

10110010.11B = 1\*27 + 0\*24 + 1\*25 + 1\*24 + 0\*23+ 0\*22 + 1\*21 + 2-1 + 2-2

= 128 + 32 + 16 + 2 + 0.5 + 0.25 = 178.75

* 1. Đổi số 1111011110110010.110001B về hệ 10.

(Thực hiện theo 2 cách như câu 2.4).

Cách 1: Đổi số đã cho qua số hệ 16, rồi đổi tiếp từ số hệ 16 về hệ 2

Đổi số 10110010.11B về hệ 16, ta có:

1111011110110010.110001=F7B2.C4

Đổi số E7B2.CB về hệ 10 theo công thức (2.1), ta có:

15 \* 163 + 7 \* 162 + 11 \* 161 *-1* + 2 \* 160 + 12 \* 16-1*-1* +11\*16-2*-1* =63410.75

Cách 2: Đổi số đã cho thẳng sang số hệ 10 (theo công thức 2.1)

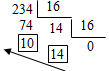
1111011110110010.110001 B= 1\*215 + 1\*214 + 1\*213 + 1\*212 + 0\*211+ 1\*210 + 1\*29 +1\*28 + 1\*27 + 0\*26 + 1\*25 + 1\*24+ 1\*21 + 2-1 + 2-2 =63410.75

* 1. Đổi số nguyên thập phân sau ra số hex 16 bit: 234.

Gợi ý

Số hệ hex 16 bit phải là số có đủ 4 chữ số hệ 16, 234 là số nguyên dương nên đổi về hệ 16 bình thường theo thuật toán chia 16.

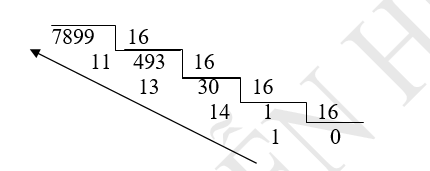
Bài làm



Vậy 234 = 00EAH

* 1. Đổi số nguyên thập phân sau ra số hex 16 bit: 7899.

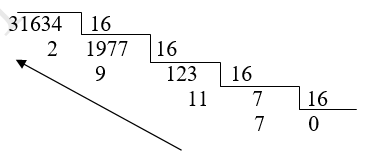
Bài làm



Vậy 7899 = 1EDB (H)

* 1. Đổi số nguyên thập phân sau ra số hex 16 bit: 31634.

Bài làm



Vậy 31634 = 7B92 (H)

* 1. Đổi số nguyên thập phân sau ra số hex 16 bit: -16.

Số -16 là số bù 2 của +16 và ngược lại, vậy ta tìm số bù 2 của +16.

+16 = 0000000000010000B

Số bù 1 (của +16) = 1111111111101111B

+ 1

Số bù 2 (của +16) = 1111111111110000B

Vậy -16 = 1111111111110000B = FFF0H

* 1. Đổi số nguyên thập phân sau ra số hex 16 bit: -116

Số -116 là số bù 2 của +116 và ngược lại, vậy ta tìm số bù 2 của +116.

+116 = 0000000001110100B

Số bù 1 (của +116) = 1111111110001011B

+ 1

Số bù 2 (của +116) = 1111111110001100B

Vậy -116 =1111111110001100 B = FF8C H

* 1. Biểu diễn số -67 ở dạng nhị phân 16 bit

Số -67 là số bù 2 của +67 và ngược lại, vậy ta tìm số bù 2 của +67.

+67 = 0000000000100011B

Số bù 1 (của +67) = 1111111111011100B

+ 1

Số bù 2 (của +67) = 1111111111011101B

Vậy -67 =1111111111011101B = FFDB H

* 1. Đổi -128 về dạng nhị phân.

Số -128 là số bù 2 của +128 và ngược lại, vậy ta tìm số bù 2 của +128.

+128 = 0000000010000000B

Số bù 1 (của +128) = 1111111101111111B

+ 1

Số bù 2 (của +128) = 1111111110000000B

Vậy -128 = 1111111110000000B = 10000000B (biểu diễn 8 bit, MSB = 1)

* 1. Đổi -132 về dạng nhị phân.

Số -132 là số bù 2 của +132 và ngược lại, vậy ta tìm số bù 2 của +132.

+132 = 0000000000000100B

Số bù 1 (của +132) = 1111111111111011B

+ 1

Số bù 2 (của +132) = 1111111111111100B

Vậy -132 =1111111111111100B = 11111100B (biểu diễn 8 bit, MSB = 1)

* 1. Đổi -92 về dạng nhị phân.

Bài làm

Số -92 là số bù 2 của +92 và ngược lại, vậy ta tìm số bù 2 của +92.

+92 = 0000000000011100B

Số bù 1 (của +92) = 1111111111100011B

+ 1

Số bù 2 (của +92) = 1111111111100100B

Vậy -92 = 1111111111100100B =11100100 B (biểu diễn 8 bit, MSB = 1)

* 1. Sau đây là tên và nội dung (dạng hex) của các thanh ghi 16 bit trong vi xử lý:

AX = 1234, BX = 900A, CX = FFFA, DX = 7FFF

Hãy tìm giá trị và dấu của giá trị trong chúng ở dạng hệ 10.

Bài làm

1. Với AX = 1234H

= 0001001000110100B (MSB = 0 là số dương)

= 212 + 29 + 25 + 24 + 22

= 4096 + 512 + 32 + 16 + 4 = 4660

1. Với BX = 900AH

= 1001000000001010B

Gọi B là giá trị chứa trong BX, ta thấy MSB = 1, nên giá trị B trong BX là số âm, ta xác định số bù 2 của B

B = 1001000000001010B

Số bù 1(B) = 0110111111110101B

+ 1

Số bù 2(B) = 0110111111110110B = 6FF6H

= 6 \* 163 + 15 \* 162 + 15 \* 16 + 6

= 28662 = |B|

Vậy giá trị trong AX = B = 28662

1. Với CX = FFFAH

Gọi C là giá trị chứa trong CX, ta thấy MSB = 1, nên giá trị C trong CX là số âm, ta xác định số bù 2 của C

C = 1111111111111010B

Số bù 1(C) = 0000000000000101B

+ 1

Số bù 2(C) = 000000000000110B = 0006H

= 0\* 163 + 0 \* 162 + 0 \* 16 + 6

= 6 = |C|

Vậy giá trị trong CX = C = 250

1. Với DX = 7FFFH

Với DX = 7FFFH

=0111111111111111 B(MSB = 0 là số dương)

= 214 + 213 + 212 + 211 + 210 + 29 + 28 + 27 + 26 +25 + 24 + 23 + 22 + 21 + 20=32767.

* 1. Biểu diễn số +1022,9 ở dạng số dấu chấm động trong máy tính theo chuẩn IEEE 32 bit.

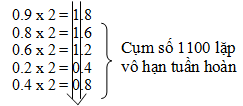
Bài làm

***Bước 1. Đổi số 1022.9 về hệ 2.***

1. Đổi phần nguyên hệ 10 về hệ 2 (sử dụng cách nhẩm nhanh)

1022 = (1024 -1) – 1 = (210 -1) -1 = 1111111111B – 1 = 1111111110B

1. Đổi phần lẻ thập phân về phần lẻ nhị phân



Vậy 0.9 = 0.11110111011101110…B

(Từng cụm 4 bit bắt đầu từ cụm gạch chân đầu tiên, thể hiện sự lặp vô hạn tuần hoàn, số bit ta lấy là bao nhiêu kể từ dấu chấm nhị phân phụ thuộc vào độ chính xác của giá trị cần lây.)

1. Ghép 2 phần

1022.9 = 1111111110.11100110011001100…B

***Bước 2. Chuẩn hóa theo định dạng số dấu chấm động 32 bit.***

1022.9 = 1111111110.11100110011001100…B

= (-1)0 x 1.11111111011100110011001B x 29

(Sau dấu chấm nhị phân lấy đủ 23 bit)

E – Bias = 9 ==> E = 9 + Bias = 9 + 127 = 128 + 8

E = 27 + 8 = 10000000B + 1000B = 10001000B

Vậy ta có :

S = 0.

E = 10001000B

M = 11111111011100110011001B

***Bước 3. Ghép vào khuôn dạng.***

1022.9 = 0100/0100/0111/1111/1011/1001/1001/1001B = 447FB999H

**0000 = 0 (h) = 0**

**0001 = 1 (h) = 1**

**0010 = 2 (h) = 2**

**0011 = 3 (h) = 3**

**0100 = 4(h) = 4**

**0101 = 5(h) = 5**

**0110 = 6(h) = 6**

**0111 = 7(h) = 7**

**1000 = 8(h) = 8**

**1001 = 9(h) = 9**

1010 = A (h) = 10

1011 = B(h) = 11

1100 = C (h) = 12

1101 = D(h) = 13

1110 = E (h) = 14

1111 = F(h) = 15

* 1. Biểu diễn số -1012,8 ở dạng số dấu chấm động trong máy tính theo chuẩn IEEE 32 bit.

Bài làm

1012,8 = 1111110000001100.1100110011001100…B

-1012.8= 1111110000001100.1100110011001100…B

= (-1)1 x 1.11111000000110011001100x 215

(Sau dấu chấm nhị phân lấy đủ 23 bit)

E – Bias = 15 ==> E =15 + Bias = 15 + 127 = 128 + 14

E = 27 + 14= 10001110B

Vậy ta có :

S = 1(vì là số âm).

E = 10001110B

M = 11111000000110011001100

==.> -1012.8 = 11000111011111000000110011001100=C77C0CCC H

* 1. Biểu diễn số +912,0625 ở dạng số dấu chấm động trong máy tính theo chuẩn IEEE 32 bit.

Bài làm

Bước 1. Đổi số 912,0625 về hệ 2.

Đổi phần nguyên hệ 10 về hệ 2

912 = (1024 -1) – (64 – 1) – (32 – 1) – (16 – 1)

= (210 -1) – (26 -1) - (25 -1) - (24 -1) = 1110010000B

Đổi phần lẻ thập phân về phần lẻ nhị phân

0,0625 x2= 0,125

0.125 x 2= 0,25

0,25 x 2 = 0,5

0,5 x 2 =1.0

Vậy 0.0625 = 0.0001B

Ghép 2 phần

912.0625 = 1110010000.0001B

Bước 2. Chuẩn hóa theo định dạng số dấu chấm động 32 bit.

912.0625 = 1110010000.0001 B = (-1)0 x 1.1100100000001 B x 2^9

(Sau dấu chấm nhị phân lấy đủ 23 bit)

E – Bias = 9 ==> E = 9 + Bias = 9 + 127 = 128 + 8

E = 27 + 8 = 10000000B + 1000B = 10001000B

Vậy ta có :

S = 0.

E = 10001000B

M = 11001000000010000000000 B

Bước 3. Ghép vào khuôn dạng.

919,0625 = 01000100011001000000010000000000 B = 4460400 H

* 1. Cho biết số dấu chấm động có giá trị 447FB999H bằng bao nhiêu ở hệ 10.

***Bước 1. Đổi số đã cho ra số nhị phân, xác định các giá trị S, E-Bias, M***

447FB999H = 0**10001000**11111111011100110011001B.

Theo khuôn dạng lưu trữ, ta tách được:

S = 0.

E = 10001000B = 28 + 23 = 128 +8 = 136

E – Bias = 136 – 127 = 9

M = 11111111011100110011001B

***Bước 2. Đưa ra số nhị phân dấu chấm động ở dạng chuẩn hóa theo công thức 2.2. và chuyển về số nhị phân dấu chấm tĩnh.***

Ta có: R = (-1)s x 1.M x 2E-Bias

= (-1)0 x 1. 11111111011100110011001B x 29

= 1111111110. 11100110011001B

***Bước 2. Chuyển giá trị nhị phân dấu chấm tĩnh về hệ 10.***

1. Đổi phần nguyên nhị phân về phần nguyên thập phân.

1111111110B = (210 – 1) – 1 = 1024 – 1 – 1 = 2=1022

(phương pháp nhẩm nhanh: 1111111111B = 210 - 1)

1. Đổi phần lẻ nhị phân về phần lẻ thập phân

0.11100110011001B = 0.1110011001100100B

= 0.E664H

= 14 \* 16-1 + 6 \* 16-2 + 6 \* 16-3 + 4 \* 16-4

= 0.875 + 0.0234375 + 0.001465 + 0.0000610

= 0.8999634

1. Ghép 2 phần nguyên và phần lẻ thập phân

Vậy 447FB999H = 1022. 8999634 ≈ 1022.9

* 1. Cho biết số dấu chấm động có giá trị 409CCCCCH bằng bao nhiêu ở hệ 10.

409CCCCC H = 01000000100111001100110011001100B.

Theo khuôn dạng lưu trữ, ta tách được:

S = 0.

E = 10000001B = 27 +1 = 128 +1 = 129

E – Bias = 129 – 127 = 2

M = 00111001100110011001100B

Ta có: R = (-1)s x 1.M x 2E-Bias

= (-1)0 x 1. 00111001100110011001100 B x 22 = 100.111001100110011001100 B

100B =4

0.111001100110011001100 B = 0. 11100110 B = 0.E6H = 14\* 16-1 + 6 \* 16-2 = 0.8984375

Ghép 2 phần nguyên và phần lẻ thập phân

Vậy 409CCCCC H = 4.8984375≈4.9

* 1. Cho biết số dấu chấm động có giá trị BF400000H bằng bao nhiêu ở hệ 10.

Bài làm

BF400000H = 10111111010000000000000000000000B.

Theo khuôn dạng lưu trữ, ta tách được:

S = 1.

E = 01111110B = 126

E – Bias = 126 – 127 = -1

M = 10000000000000000000000B

Ta có: R = (-1)s x 1.M x 2E-Bias = (-1)1 x 1. 10000000000000000000000 B x 2-1= - 0.11 B

-00000000B = -0

0.11B = 0.1100B= 0.C H= 12 \* 16-1 = 0.75

Ghép 2 phần nguyên và phần lẻ thập phân

Vậy BF400000 H = - 0. 75

* 1. Giả sử chuỗi “S12.75” đang lưu trong bộ nhớ bắt đầu tại địa chỉ 0, cho biết nội dung của các byte từ 0 đến 5 dưới dạng số hex.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Vị trí | 0 | 1 | 2 | 3 | 4 | 5 |
| Mã hexa | 53 | 31 | 32 | 2E | 37 | 35 |
| Kí Tự | S | 1 | 2 | . | 7 | 5 |

* 1. Hãy dịch thông điệp đã mã hóa dưới dạng mã ASCII (hệ hex) sau đây:

41 74 74 61 63 6B 20 61 77 6E

Mã hóa là : Attack awn

* 1. Giả sử một byte có mã ASCII của một chữ hoa, hỏi phải cộng thêm một số hex là bao nhiêu để đổi nó thành chữ thường.

một byte có mã ASCII của một chữ hoa, chúng ta phải cộng thêm một số hex là 20 H để đổi nó thành chữ thường.

* 1. Giả sử một byte có nội dung là mã ASCII mã hóa một số thập phân từ 0 đến 9, hỏi phải trừ đi một số hex là bao nhiêu để đổi nó thành chính số đó.

một byte có nội dung là mã ASCII mã hóa một số thập phân từ 0 đến 9, phải trừ đi một số hex là 30H để đổi nó thành chính số đó.

Thực hiện trừ 2 số nhị phân sau: 00001110B – 01111111B

Bài làm

* + Số bù 1 của 01111111 là 10000000B

+\_\_\_\_\_\_\_ 1

* + Số bù 2 của 01111111 = 10000001B

Vậy 00001110B – 01111111 = 00001110 + 10000001 = 10001111B

* 1. Cộng các số nhị phân sau: 01111011B, 11001111B, 10111111B, 00111101B, 01111111B, 01111110B

 Vậy = 1101000011B

* 1. Cộng các số nhị phân sau: 11101101B, 11011110B, 10111111B, 10101101B, 01111101B

 Vậy = 111010100B

* 1. Thực hiện nhân 2 số nhị phân 8 bit sau: 11101111B x 11111101B

** Vậy = 1110110001110011B

* 1. Nêu tên và công dụng của các thanh ghi trong hệ thống KTMT nói chung.

**Thanh ghi khả hiện** - Có hai loại. Thanh ghi dữ liệu và thanh ghi địa chỉ.

**Thanh ghi dữ liệu** được dùng để lưu [số nguyên](https://vi.wikipedia.org/wiki/S%E1%BB%91_nguy%C3%AAn_(khoa_h%E1%BB%8Dc_m%C3%A1y_t%C3%ADnh)) . Trong một số bộ CPU hiện nay và trước đây, có một thanh ghi đặc biệt là thanh ghi tích lũy [tích lũy](https://vi.wikipedia.org/w/index.php?title=Accumulator_(t%C3%ADnh_to%C3%A1n)&action=edit&redlink=1), được dùng cho nhiều tính toán.

**Thanh ghi địa chỉ** chứa địa chỉ bộ nhớ và được dùng để truy cập [bộ nhớ](https://vi.wikipedia.org/wiki/B%E1%BB%99_nh%E1%BB%9B). Trong một số CPU, có một thanh ghi địa chỉ đặc biệt [thanh ghi chỉ mục](https://vi.wikipedia.org/w/index.php?title=Thanh_ghi_ch%E1%BB%89_m%E1%BB%A5c&action=edit&redlink=1), dù chúng thường được dùng để sửa đổi địa chỉ hơn là chứa địa chỉ.

**Thanh ghi điều kiện** chứa giá trị thực thường dùng để quyết định hoật động thực thi lệnh

**Thanh ghi điều khiển và trạng thái** – Có ba loại: chương trình phản hồi, thanh ghi chỉ lệnh, chương trình trạng thái từ.

Thanh ghi liên quan đến tìm nạp thông tin từ bộ nhớ truy cập ngẫu nhiên (RAM), 1 tập hợp lưu giữ các thanh ghi được định vị trên các chip riêng biệt từ CPU (không giống đa số các loại trên, đây thông thường là những thanh ghi không có cấu trúc):

Thanh ghi bộ nhớ trung gian

Thanh ghi bộ nhớ địa chỉ

Thanh ghi bộ nhớ kiểu miền

* 1. Nêu tên và tóm tắt chức năng các cờ trong thanh ghi cờ của CPU 8086.

Cờ Zero (cờ rỗng): được thiết lập lên 1 khi kết quả của phép toán bằng 0.

Cờ Sign (cờ dấu): được thiết lập lên 1 khi kết quả phép toán nhỏ hơn 0

Cờ Carry (cờ nhớ): được thiết lập lên 1 nếu phép toán có nhớ ra ngoài bit cao nhất → cờ báo tràn với số không dấu.

Cờ Overflow (cờ tràn): được thiết lập lên 1 nếu cộng hai số nguyên cùng dấu mà kết quả có dấu ngược lại → cờ báo tràn với số có dấu.

* 1. Xác định giá trị của cờ tràn cho phép toán sau: 01110001B + 00100011B.

01110001B

+ 00100011B

10010100B

Ta thấy: CF = 0 (Không có nhớ từ MSB ra ngoài).

C6,7 = 1 (có nhớ từ bit 6 sang bit 7)

Vậy OF = CF ⊕ C6,7 = 0 ⊕ 1 = 1

Lưu ý: Khi OF = 1, hiện tượng tràn số đã xảy ra, cụ thể ta thấy kết quả phép toán trên là sai cho số có dâu (cộng 2 số dương – MSB = 0, cho kết quả là số âm – MSB = 1)

* 1. Xác định giá trị của cờ tràn cho phép toán sau: 11110001B + 01000010B.

01110001B

+ 00100011B

10010100B

Ta thấy: CF = 0 (Không có nhớ từ MSB ra ngoài).

C6,7 = 1 (có nhớ từ bit 6 sang bit 7)

Vậy OF = CF ⊕ C6,7 = 0 ⊕ 1 = 1

* 1. Xác định giá trị của cờ tràn cho phép toán sau: 10110001B + 11000010B.

10110001B

+ 11000010B

101110011B

Ta thấy: CF = 1 (Không có nhớ từ MSB ra ngoài).

C6,7 = 0 (có nhớ từ bit 6 sang bit 7)

Vậy OF = CF ⊕ C6,7 = 1 ⊕ 0 = 1

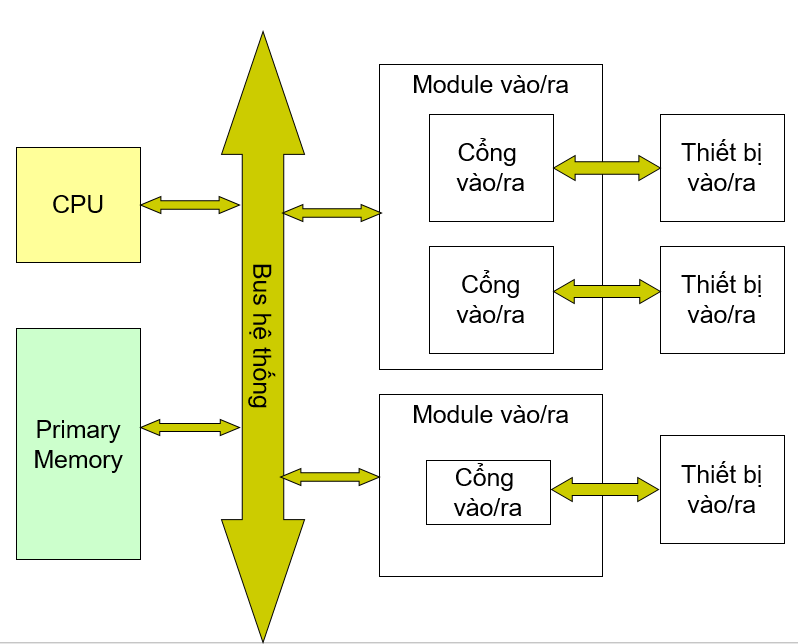
1. CPU, Đường truyền và hệ thống vào/ra
   1. Cho biết ý nghĩa khi nói Bus địa chỉ có độ rộng 32 bit.

Bus gồm 32 đường dây dẫn, CPU có khả năng quản lý không gian nhớ là 2 mũ 32 =4GB

* 1. Cho biết ý nghĩa khi nói Bus dữ liệu có độ rộng 64 bit.

Bus gồm 64 đường dây dẫn, CPU có khả năng xử lý  toán hạng 64 bit trong 1 chu kỳ lệnh.

* 1. Vẽ sơ đồ khối chung của hệ thống xử lý vào/ra trong máy tính và trình bày tóm tắt chức năng của từng thành phần trong sơ đồ.

Chức năng CPU:

* + điều khiển hoạt động của máy tính
  + xử lý dữ liệu

**Chức năng bộ nhớ trong:**

Chứa các thông tin mà CPU có thể trao đổi trực tiếp

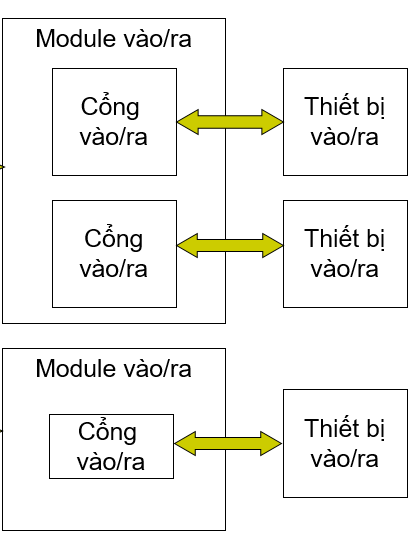
Tốc độ rất nhanh

Dung lượng không lớn

Sử dụng bộ nhớ bán dẫn: ROM và RAM

**Chức năng hệ thống vào/ra:** Trao đổi thông tin giữa máy tính với thế giới bên ngoài.

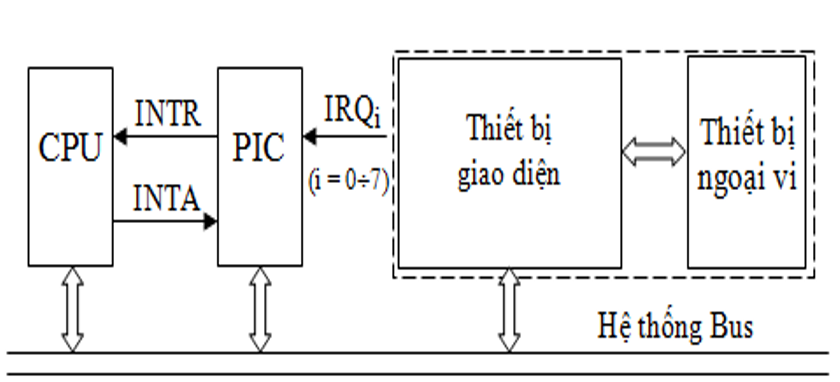
**Chức năng bus liên kết hệ thống:** dùng trao đổi thông tin giữa CPU và bộ nhớ trong (thông qua cache)

* 1. Vẽ sơ đồ khối chung của hệ thống vào/ra dữ liệu và trình bày tóm tắt chức năng của từng thành phần trong sơ đồ.
* Chức năng của mô-đun vào-ra:
* Điều khiển và định thời
* Trao đổi thông tin với CPU
* Trao đổi thông tin với thiết bị ngoại vi
* Đệm giữa bên trong máy tính với thiết bị ngoại vi
* Phát hiện lỗi của thiết bị ngoại vi

Chức năng thiết bị ngoại vi: chuyển đổi dữ liệu giữa bên trong và bên ngoài máy tính

* 1. Trình bày cấu trúc chung của modul vào/ra dữ liệu.

Thiết bị giao diện được chế tạo theo từng module, gọi là module vào/ra hoặc còn gọi là control card (thẻ mạch điều khiển thiết bị). Mỗi module có 3 loại thanh ghi:

* + Các thanh ghi điều khiển nhận và chứa các từ điều khiển xác lập chế độ làm việc của thiết bị.
  + Các thanh ghi trạng thái chứa thông tin phản ánh trạng thái làm việc của thiết bị ngoại vi.
  + Các thanh ghi dữ liệu thực hiện chức năng bộ đếm, nơi trung chuyển dữ liệu ra/vào.
  1. Trình bày 2 phương pháp vào ra dữ liệu do CPU chủ động.
* ***phương pháp vào ra dữ liệu  theo định trình***
* Đây là phương pháp mà quá trình vào/ra được thực hiện tức thời nhờ các lệnh vào/ra và CPU không cần quan tâm đến trạng thái của thiết bị vào/ra **(bao gồm thiết bị giao diện và thiết bị ngoại vi)**.
* Nhược: độ tin cậy trong truyền, nhận dữ liệu không cao, dữ liệu truyền nhận dễ bị chồng lên nhau gây mất dữ liệu nếu bên nhận xử lý không kịp.
* Phương pháp này thích hợp với những quá trình vào/ra có chu kỳ cố định và có thể xác định trước.
* ***phương pháp vào ra dữ liệu kiểu thăm dò***
* CPU kiểm tra trạng thái sẵn sàng làm việc của thiết bị trước khi việc vào/ra dữ liệu được thực hiện. Khi thiết bị chưa sẵn sàng làm việc thì CPU lại phải tiếp tục thăm dò, việc thăm dò được lặp đi lặp lại cho đến khi thiết bị sẵn sàng trao đổi dữ liệu với CPU thì quá trình nhận dữ liệu được tiếp hành.
* Ưu: quá trình trao đổi dữ liệu có độ tin cậy rất cao vì việc truyền nhận dữ liệu chỉ xảy ra khi hai bên truyền và nhận đều sẵn sàng.
* Nhược: chiếm dụng nhiều thời gian CPU cho việc thăm dò nên hiệu quả hoạt động của hệ thống không cao.
  1. Trình bày cấu trúc của hệ thống vào/ra theo ngắt cứng 
* Hệ thống ngắt cứng gồm thiết bị điều khiển ngắt PIC, BUS dữ liệu, các thiết bị vào ra.
* PIC nhận các yêu cầu ngắt IRQi (I = 0 ÷ 7), xử lí ưu tiên ngắt và cung cấp số hiệu ngắt có ưu tiên cao nhất cho CPU qua BUS dữ liệu. CPU căn cứ vào số hiệu này thực hiện quá trình vào ra dữ liệu với chương trình được chọn.
  1. Trình bày quá trình vào/ra dữ liệu theo phương pháp vào/ra dữ liệu theo ngắt cứng.
* Quá trình xảy ra cụ thể như sau:

-   CPU đang thực hiên tiến trình A.

-   Các thiết bị vào/ra có yêu cầu phục vụ, phát ra tín hiệu IRQi (I = 0 ÷ 7) tới PIC. Mỗi thiết bị vào/ra đã được ấn định sẵn một số hiệu ngắt cứng nhất định.

-   Thiết bị PIC lựa chọn số hiệu ngắt có mức ưu tiên cao nhất, phát tín hiệu yêu cầu tới CPU yêu cầu CPU phục vụ.

-   Nếu CPU chấp nhận ngắt, CPU hoàn thành nốt lệnh đang thực hiện, tiến hành lưu trạng thái của tiến trình đang thực hiện và trạng thái hiện thời của CPU.

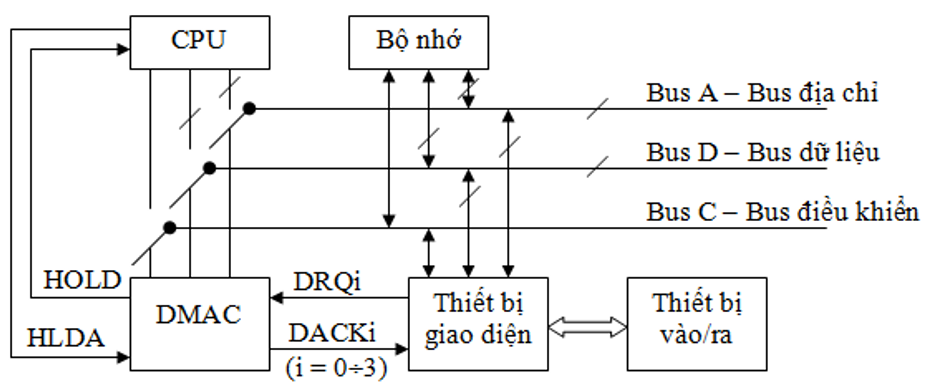
-   CPU phát tín hiệu sẵn sàng phục vụ yêu cầu ngắt tới PIC.

-   PIC phát số hiệu ngắt được chọn tới CPU qua Bus dữ liệu.

-   Dựa vào số liệu này CPU xác định địa chỉ chương trình con phục vụ ngắt, kích hoạt và thực hiện vào/ra dữ liệu.

-   Kết thúc, CPU lấy lại trạng thái cũ của CPU và tiến trình A vừa bị ngắt để tiếp tục thi hành.

* Ưu: quá trình trao đổi dữ liệu có độ tin cậy rất cao vì việc truyền nhận dữ liệu chỉ xảy ra khi hai bên truyền và nhận đều sẵn sàng.
* Nhược: chiếm dụng nhiều thời gian CPU cho việc thăm dò nên hiệu quả hoạt động của hệ thống không cao
  1. Trình bày cấu trúc của hệ thống vào/ra dữ liệu theo kiểu DMA.



* CPU:
* DMAC:
* Bộ nhớ:
* Hệ thống vào/ra:
* Bus:
  1. Trình bày khái niệm quá trình DMA và quá trình vào/ra dữ liệu kiểu DMA
* Khái niệm:
* DMA – Direct Memory Access (truy nhập bộ nhớ trực tiếp)
* Quá trình DMA là quá trình vào/ra dữ liệu giữa bộ nhớ và thiết bị ngoại vi mà không thông qua CPU.
* Quá trình DMA diễn ra như sau:
* CPU đang hoạt động bình thường (CPU quản lý hệ thống BUS gồm bus A – bus địa chỉ, bus D – bus dữ liệu, bus C – bus điều khiển).
* DMAC được xác lập chế độ làm việc, nhận thông tin về địa chỉ đầu khối nhớ chưa dữ liệu và kích thước khối dữ liệu cần truyền.
* Các thiết bị vào/ra phát tín hiệu DRQi cho DMAC, DMAC chọn thiết bị có mức ưu tiên cao nhất.
* DMAC phát tín hiệu BRQ/HOLD = 1 cho CPU, yêu cầu CPU chuyển nhượng BUS.
* Nếu CPU chấp nhận, CPU thực hiện nốt chu kỳ máy, CPU phát tín hiệu BGT/HLDA chấp nhận chuyển nhượng BUS.
* CPU tự tách ra khỏi hệ thông BUS, quyền điều khiển BUS thuộc về DMAC.
* DMAC phát tín hiêu DACKi báo cho thiết bị yêu cầu được chọn vào/ra dữ liệu. DMAC phát đia chỉ ô nhớ đầu tiên của khối dữ liệu cần truyền lên bus A, quá trình truyền dữ liệu bắt đầu. Trong quá trình truyền, DMAC giảm bộ đếm và tăng nội dung của con trỏ chứa địa chỉ cho đến khi nội dung bộ đếm bằng 0 thì khối dữ liệu đã truyền xong.
* DMAC kết thúc quá trình DMA, phát tín hiệu BRQ/HOLD = 0 đến CPU, trả lại quyền điều khiển BUS cho CPU. CPU tiếp tục làm việc bình thường.
  1. Trình bày các kiểu trao đổi dữ liệu theo phương pháp vào/ra dữ liệu kiểu DMA.
* **Có 3 kiểu trao đổi dữ liệu DMA:**

-         Treo CPU một khoảng thời gian để trao đổi cả mảng dữ liệu.

-         Treo CPU để trao đổi tưng byte.

-         Tận dụng thời gian CPU không dùng BUS để trao đổi dữ liệu.

* **Các chế độ ưu tiên:**

-Ưu tiên cố định: kênh 0 luôn có mức ưu tiên cao nhất, kênh 3 có mức ưu tiên thấp nhấp

-Ưu tiên vòng: Kênh nào vừa phục vụ sẽ có mức ưu tiên thấp nhất, lúc mới thiết lập kênh 0 sẽ có mức ưu tiên cao nhất.

* 1. Nhận biết các mạch cổng sau (ghi rõ tên mạch cổng ở bên dưới hình)



XOR NOT NOR AND OR NAND

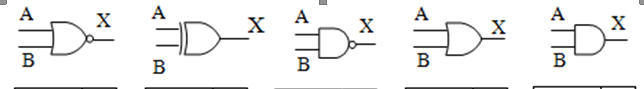
* 1. Cho các mạch cổng 2 đầu vào, ghi tên cổng, hoàn thiện các bảng chân lý tương

NOR XOR NAND OR AND

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| IN | | OUT |  | IN | | OUT |  | IN | | OUT |
| A | B | C | A | B | C |  | A | B | C |
| 0 | 0 | 1 | 0 | 0 | 0 |  | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 | 1 |  | 0 | 1 | 0 |
| 1 | 0 | 1 | 1 | 0 | 1 |  | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 | 1 |  | 1 | 1 | 1 |

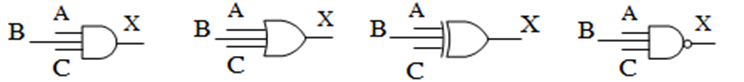
|  |  |  |
| --- | --- | --- |
| IN | | OUT |
| A | B | C |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

|  |  |  |
| --- | --- | --- |
| IN | | OUT |
| A | B | C |
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |



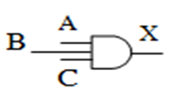
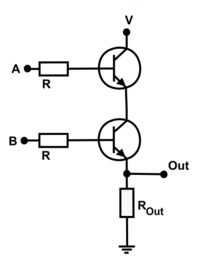
* 1. Cho các mạch cổng 3 đầu vào, ghi tên cổng, hoàn thiện các bảng chân lý tương

AND OR XOR NAND

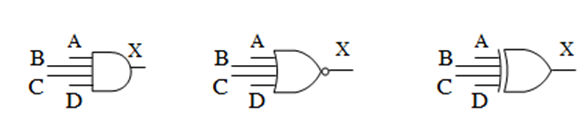


|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| IN | | | OUT |  | IN | | | OUT |  | IN | | | OUT |  |  | IN | | | OUT |
| A | B | C | X | A | B | C | X |  | A | B | C | X |  |  | A | B | C | C |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |  | 0 | 0 | 0 | 0 |  |  | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |  | 0 | 0 | 1 | 1 |  |  | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |  | 0 | 1 | 0 | 1 |  |  | 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 |  | 0 | 1 | 1 | 0 |  |  | 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |  | 1 | 0 | 0 | 1 |  |  | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 |  | 1 | 0 | 1 | 0 |  |  | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 |  | 1 | 1 | 0 | 0 |  |  | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |  | 1 | 1 | 1 | 1 |  |  | 1 | 1 | 1 | 1 |

* 1. Vẽ sơ đồ mạch điện của cổng AND 3 đầu vào từ các transistor

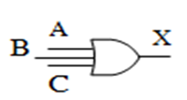
 

* 1. Cho các mạch cổng 4 đầu vào sau, ghi tên cổng, hoàn thiện các bảng chân lý tương ứng

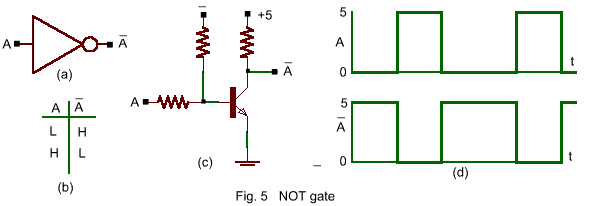
AND NOR OR 

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| IN | | | | OUT |  | IN | | | | OUT |  | IN | | | | OUT |
| A | B | C | D | X |  | A | B | C | D | X |  | A | B | C | D | X |
| 0 | 0 | 0 | 0 | 0 |  | 0 | 0 | 0 | 0 | 1 |  | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |  | 0 | 0 | 0 | 1 | 0 |  | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 0 |  | 0 | 0 | 1 | 0 | 0 |  | 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 |  | 0 | 0 | 1 | 1 | 0 |  | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 |  | 0 | 1 | 0 | 0 | 0 |  | 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 |  | 0 | 1 | 0 | 1 | 0 |  | 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 |  | 0 | 1 | 1 | 0 | 0 |  | 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |  | 0 | 1 | 1 | 1 | 0 |  | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 |  | 1 | 0 | 0 | 0 | 0 |  | 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |  | 1 | 0 | 0 | 1 | 0 |  | 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 0 |  | 1 | 0 | 1 | 0 | 0 |  | 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |  | 1 | 0 | 1 | 1 | 0 |  | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 |  | 1 | 1 | 0 | 0 | 0 |  | 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 |  | 1 | 1 | 0 | 1 | 0 |  | 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 |  | 1 | 1 | 1 | 0 | 0 |  | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |  | 1 | 1 | 1 | 1 | 0 |  | 1 | 1 | 1 | 1 | 1 |

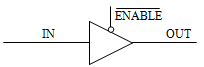
* 1. Vẽ sơ đồ mạch điện của cổng OR 3 đầu vào từ các transistor



* 1. Vẽ mạch điện có chức năng tương ứng mạch NOT



* 1. Cho biết tên và nguyên lý làm việc của mạch logic sau:



Thiết bị 3 trạng thái *bình thường là thông*:

Hoạt động:

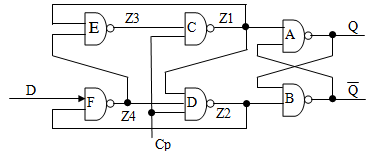
* + Khi chân /ENABLE có mức logic thấp (0) thì đầu ra (OUT) bằng đầu vào (IN).
  + Khi chân /ENABLE có mức logic cao (1) thì đầu ra và đầu vào cách ly (đầu ra có mức trở kháng cao).
  1. Cho biết tên và nguyên lý làm việc của mạch logic sau:



Thiết bị 3 trạng thái *bình thường là ngắt*:

Hoạt động:

* + Khi chân ENABLE có mức logic thấp (0) thì đầu ra và đầu vào cách ly (đầu ra có mức trở kháng cao).
  + Khi chân ENABLE có mức logic cao (1) thì đầu ra (OUT) bằng đầu vào (IN).
  1. Cho sơ đồ mạch Flip-Flop sau, chứng minh rằng khi Cp = 0, trạng thái đầu ra Q không đổi, khi Cp = 1 (chuyển từ 0 → 1) thì Q = D



Bài làm

TH1: Cp=0

Giả sử Q(t-1) = 0 🡺 /Q(t-1) = 1.

Tính Z1(t) = NAND (Cp, Z3(t)) = NAND (0, Z3(t)) = 1.

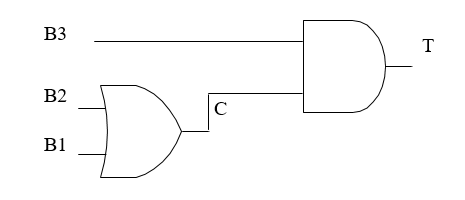
Q(t) = NAND (Z1(t), /Q(t-1)) = NAND (1,1) = 0 (1)

Giả sử Q(t-1) = 1🡺 /Q(t-1) = 0

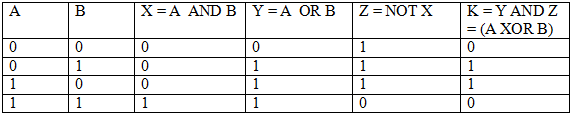
Q(t) = NAND(Z1(t), /Q(t-1)) = NAND (1,0) = 1 (2)

Từ (1) và (2) à Q(t) = Q(t-1) à Q không thay đổi.

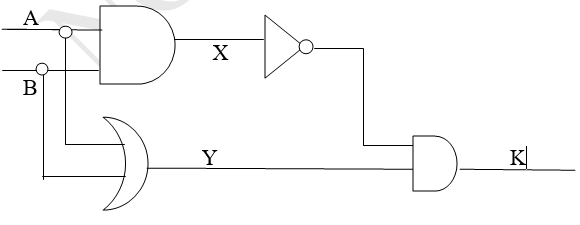
* 1. Thiết kế mạch logic để phát hiện lỗi trong mã BCD. Lối vào là mã BCD, lối ra ở trạng thái 1 khi có lỗi.



* 1. Hãy xây dựng cổng XOR 2 đầu vào từ các cổng NOT, AND, OR.



Từ bảng chân lý này ta vẽ được sơ đồ mạch logic số.



* 1. Chứng minh rằng mạch sau là cổng NOT

|  |  |
| --- | --- |
| IN | OUT |
| A | F |
| 0 | 1 |
| 1 | 0 |

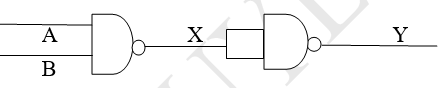
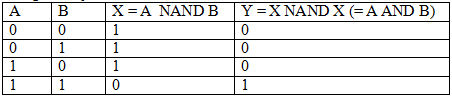


Bài làm F=NOR( A, A)

***🡺*** KL: Dựa vào bảng chân lý của mạch NOT và bảng tính toán

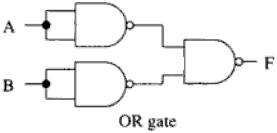
từ mạch🡪Mạch cần kiểm tra là mạch NOT 1 đầu vào A , đầu ra F.

* 1. Hãy xây dựng cổng AND 2 đầu vào từ các cổng NAND 2 đầu vào.



* 1. Chứng minh rằng mạch sau là cổng OR 2 đầu vào

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| IN1 | | IN2 | | OUT |
| A | B | C | D | F |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 |



C=NAND (A, A)

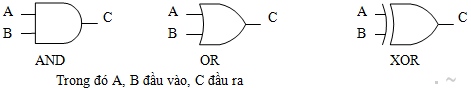
D=NAND (B, B)

F=NAND (C, D)

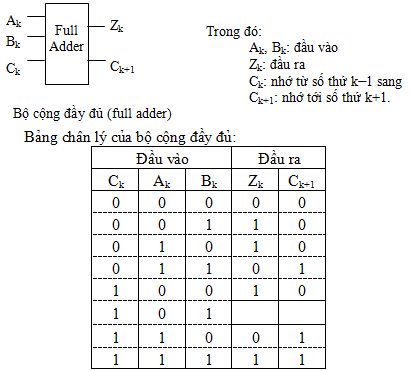
***🡺*** KL: Dựa vào bảng chân lý của mạch OR và bảng tính toán từ mạch🡪Mạch cần kiểm tra là mạch OR 2 đầu vào A,B và đầu ra F.

* 1. Hãy đọc những mô tả dưới đây về các phép toán logic và bộ cộng đầy đủ, sau đó trả lời các ý từ 1 tới 3.

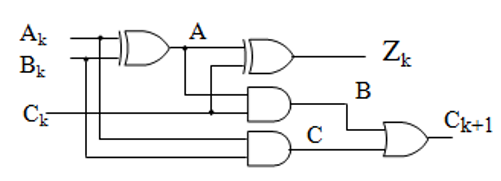
a. Các kí hiệu mạch logic cho các phép toán logic chính như sau.



b. Sau đây là hình vẽ bộ cộng đầy đủ, thực hiện việc cộng các số nhị phân theo từng chữ số có tính tới việc nhớ. Bảng cho dưới đây là bảng chân lí cho bộ cộng đầy đủ đó.



Từ bảng chân lý ta có sơ đồ bộ cộng đầy đủ như sau



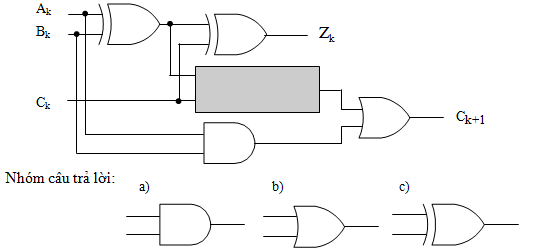
Ý 1 : Từ nhóm câu trả lời dưới đây, hãy chọn câu trả lời đúng để điền vào hộp   
 trống trong bảng chân lí của bộ cộng đầy đủ.



Nhóm câu trả lời:

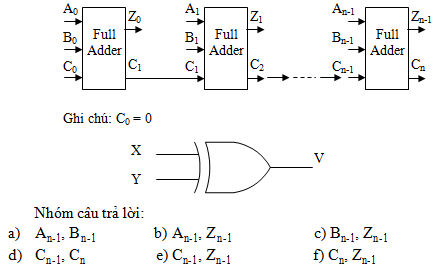
Nhóm câu trả lời đúng là : b

Ý 2: Từ nhóm câu trả lời dưới đây, hãy chọn câu trả lời đúng để điền vào hộp   
 trống trong mạch logic của bộ cộng đầy đủ.



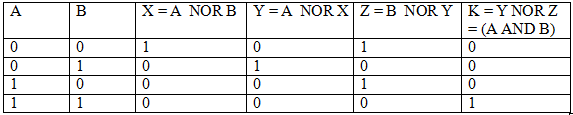
Nhóm câu trả lời đúng là : a

Ý 3: Khi một mạch logic được cấu tạo bằng các bộ cộng đầy đủ để cộng các số nhị phân n-chữ số được biểu diễn như phần bù hai, việc cộng các chữ số có ý nghĩa nhất (An, Bn và Cn) gây ra sự tràn (phần tô đậm của bảng chân lí của bộ cộng đầy đủ). Mạch logic để phát hiện việc này có thể được cấu tạo bằng một mạch XOR. Hãy chọn từ nhóm câu trả lời dưới đây tổ hợp đúng của các đầu vào X và Y cho mạch logic này.

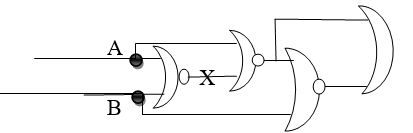


Nhóm câu trả lời đúng là: c

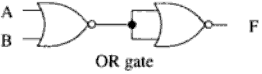
* 1. Làm thế nào để xây dựng 1 cổng AND có 2 lối vào từ những cổng NOR 2 lối vào.



Từ bảng chân lý này ta vẽ được sơ đồ mạch.



* 1. Chứng minh rằng mạch sau là cổng OR 2 đầu vào



|  |  |  |  |
| --- | --- | --- | --- |
| IN1 | | IN2 | OUT |
| A | B | C | F |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 |

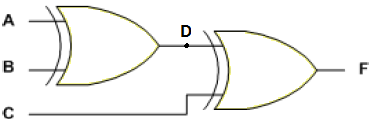
C= NOR(A ,B)

F=NOR( C, C)

***🡺*** KL: Dựa vào bảng chân lý của mạch OR và bảng tính toán từ mạch🡪Mạch cần kiểm tra là mạch OR 2 đầu vào A ,B và đầu ra F.

* 1. Chứng minh rằng mạch sau là mạch lẻ 3 đầu vào (số bit 1 là lẻ thì đầu ra là 1, ngược lại đầu ra là 0), với đầu vào là A, B, C ; đầu ra là F

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| A | B | C | D | F (mạch) | F bảng chân lý(lẻ) |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 1 | 1 |



Bài làm

D = XOR(A,B)

F = XOR(C,D)

Từ bảng tính toán và bảng chân lý

🡪 Mạch cần kiểm tra là mạch lẻ 3

đầu vào.

* 1. Chứng minh rằng mạch sau là cổng NOT

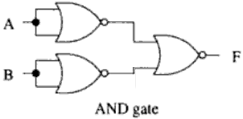
|  |  |
| --- | --- |
| IN | OUT |
| A | F |
| 0 | 1 |
| 1 | 0 |



F=NOR( A, A)

***🡺*** KL: Dựa vào bảng chân lý của mạch NOT và bảng tính toán từ mạch🡪Mạch cần kiểm tra là mạch NOT 1 đầu vào A , đầu ra F.

* 1. Chứng minh rằng mạch sau là cổng AND 2 đầu vào



Bài làm

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| IN1 | | IN2 | | OUT |
| A | B | C | D | F |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 |

C=NOR (A, A)

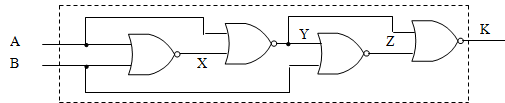
D=NOR (B, B)

F=NOR (C, D)

***🡺*** KL: Dựa vào bảng chân lý của mạch AND và bảng tính toán từ mạch🡪Mạch cần kiểm tra là mạch AND 2 đầu vào A,B và đầu ra F.

* 1. Chứng minh rằng đây là mạch AND 2 đầu vào A, B, kết quả ra K.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| IN | | | | | OUT |
| A | B | X | Y | Z | K |
| 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 |



X=NOR(A, B)

Y=NOR(A, X)

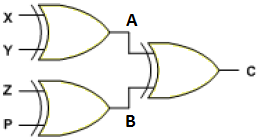
Z=NOR( Y, B)

K=NOR( Y, Z)

***🡺*** KL: Dựa vào bảng chân lý của mạch AND và bảng tính toán từ mạch🡪Mạch cần kiểm tra là mạch AND 2 đầu vào A,B và đầu ra K.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| IN | | | | | | OUT | OUT |
| X | Y | Z | P | A | B | C(mạch) | C(bảng chân lý lẻ) |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |

Chứng minh rằng mạch sau là mạch lẻ 4 đầu vào (số bit 1 là lẻ thì đầu ra là 1, ngược lại đầu ra là 0), với đầu vào là X, Y, Z, P ; đầu ra là C



Bài làm

A=XOR ( X, Y)

B=XOR ( Z, P)

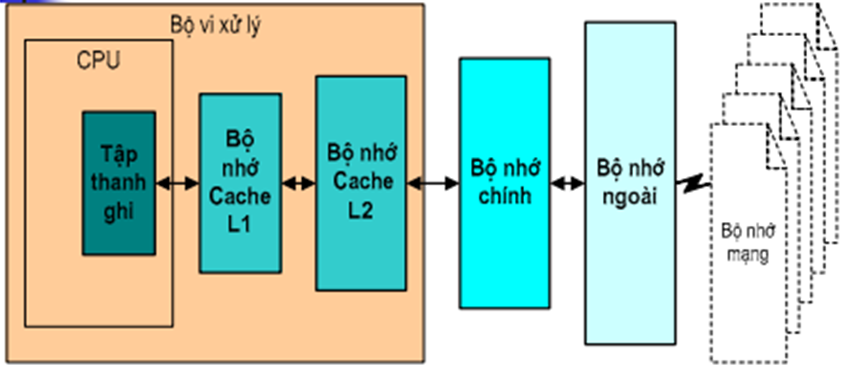
C=XOR ( A, B)

Từ bảng tính toán và bảng chân lý

🡪 Mạch cần kiểm tra là mạch lẻ 4

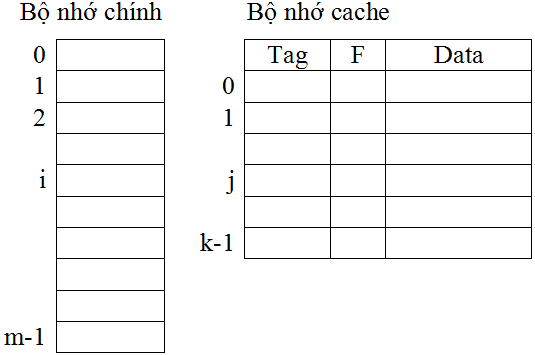
đầu vào.

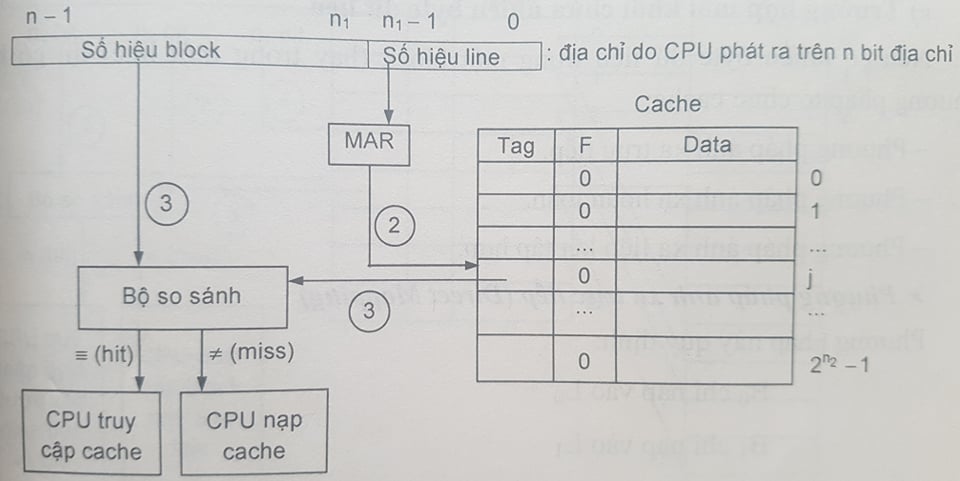
1. Kiến trúc hệ thống nhớ
   1. Vẽ và nêu ý nghĩa của mô hình phân cấp hệ thống nhớ.

Bài làm

Ý nghĩa của mô hình phân cấp:

Tăng tốc độ xử lý của CPU , đồng thời đảm bảo khả năng lưu trữ dữ liệu với khối lượng lớn.

* 1. Việc xây dựng bộ nhớ cache dựa trên 2 nguyên lý nào?
* Nguyên lý cục bộ hoá tham chiếu bộ nhớ:
  + Thời gian: Một lệnh hoặc dữ liệu vừa được truy nhập thì thường sẽ được truy nhập ngay sau đó.
  + Không gian: Một lệnh hoặc một dữ liệu vừa được truy nhập thì thường những lệnh hoặc dữ liệu lân cận sẽ được truy nhập ngay sau đó.
  1. Trình bày nguyên tắc hoạt động của bộ nhớ cache
* CPU yêu cầu nội dung của ngăn nhớ
* CPU kiểm tra trên cache với dữ liệu này
* Nếu có, CPU nhận dữ liệu từ cache (nhanh)
* Nếu không có, đọc Block nhớ chứa dữ liệu từ bộ nhớ chính vào cache
* Tiếp đó chuyển dữ liệu từ cache vào CPU
  1. Trình bày phương pháp chung để tổ chức bộ nhớ cache.
* Bộ nhớ chính chia làm m block bằng nhau,kí hiệu là:Bi[i=0 ÷ (m-1)].
* Bộ nhớ cache được chia làm k line, kí hiệu là:Lj[j=0 ÷ (k-1)]. Mỗi line chứa các thông tin sau: số hiệu thẻ(tag), bít cờ ( F) và phân chứa dữ liệu (data). Dung lượng của phần chứa dữ liệu trong 1 line bằng dung lượng của 1 block (có thể nói, khối dữ liệu trong cache là bản sao của khối dữ liệu trong bộ nhớ chính và ngược lại)
* Tag dùng để ghi số hiệu block nhớ được nạp vào line. Bit cờ F cho biết nội dung của line đã ghi ra bộ nhớ hay chưa. Nếu F =0, tức nội dung của line đã ghi ra bộ nhớ chính, nếu F =1, tức nội dung line chưa được ghi ra bộ nhó chính.
* Khi CPU truy cập dữ liệu, dữ liệu đó dã có sẵn trong cache. Trường hợp này được gọi là trúng (hit). Ngược lại thông tin khong được tìm thấy trong bộ nhớ cache thì gọi là trượt (miss), khi đó CPU phải đọc thông tin từ bộ nhớ chính và thực hiện nạp cache ( nạp từ block vào line).
* Nếu xác suất xuất hiện th hit cao thì hiệu năng hệ thống cao và ngược lại, xác suất xuất hiện th miss cao thì hiệu năng hệ thống thấp.
  1. Trình bày và vẽ sơ đồ thao tác đọc bộ nhớ cache với mỗi khối chứa một byte dữ liệu.

Khi 1 byte dữ liệu cần được đọc thì CPU cung cấp địa chỉ cho bộ điều khiển bộ nhớ. Bộ điều khiển bộ nhớ tách n bit địa chỉ làm 2 phần :

-n1 bit thấp được đặt vào thanh ghi đải chỉ MAR của cache, các bit này xác định địa chỉ của line cần tìm trong cache từ 0 đến (2n1 -1).

n2=n-n1 và có 2n2 block, được đánh số hiệu từ 0 đến (nn2 – 1)

* Bước 1: CPU kiểm tra tính hợp lệ của địa chỉ truy cập, nếu số bit có nghĩa biểu diễn truy cập lớn hơn số bit hợp lệ (lớn hơn n) thì địa chỉ này không hợp lệ, tiến trình phải dừng. ngược lai CPU phát địa chỉ truy nhập.
* Bước 2: bộ điều khiển cache tìm đến line dữ liệu trong cache tại vị trí có số thứ tự trùng vói phân số thứ tự do n1 bit thấp xác định.
* Bước 3: bộ điều khiển cache đọc giá trị tên tag và kiểm tra xem giá trị này có trùng với giá trị n2 bit xác định hay không.
* Bước 4:

Nếu trùng giá trị, đây là trường hợp cache hit và 1 byte dữ liệu được đọc vào CPU.

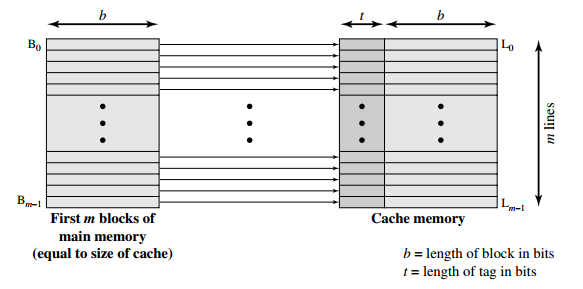
Nếu không trùng giá trị , đây là trường hợp cache miss thì 1 byte dữ kiệu phải được lấy từ bộ nhớ chính. Trong th này, nếu bít cờ F =1 ( CPU mớii ghi dữ liệu ra cache, chưa ghi ra bộ nhớ) thì trước hết cần phai sao lưu dữ liệu đang có trong line tìm được vào bộ nhớ chính theo địa chỉ của nó ( tại vị trí có số hiệu trùng với giá trị tag trong line ), sửa bit cờ F = 0 ( ghi nhận nội dung line trùng với nội dung của một block trên bộ nhớ chính), sau đó mới nạp dữ liệu từ block hó cần truy cập vào line và cung ghi luôn số hiệu block vừa nạp vào trong tag của line. Dữ liệu được lấy tù line này để cung cấp cho CPU.

* 1. Trình bày thao tác ghi bộ nhớ với mỗi khối chứa một byte dữ liệu.
* Ở loại cache ghi xuyên: dữ liệu dược ghi lên cả cache lẫn bộ nhớ cung một lúc, trường hợp này không dung đén cờ bit F và F luôn luon = 0 ( dữu liệu có trong line thì cũng được ghi luôn vào block nhớ). Kỹ thuật này làm cho thời gian ghi bộ nhớ tăng lên, hiệu suất thấp và tốn them đườn dây dữ liệu nối trực tiếp từ CPU đến bộ nhớ.
* Ở loại cache sao lưu: dữ liệu chỉ được ghi ra cache và bit F trê line được lập( F=1), ghi nhận nội dung của line khác với nội dung của mọi block tren bộ nhớ chính. Sau đó khi đọc line, nếu line này cần được thay thế nội dung bằng nộ dung của một block khác từ bộ nhớ chính (trong trường hợp đọc cach mà gặp cacahe miss),bthif bít cờ F được kiểm tra đẻ xác định xem có cần thực hiện thao tác sao lưu dư liệu từ line này sang bộ nhớ chính hay không; nếu F= 1 tì cần thực hiện sao lưu; nếu F = 0 thì không cần sao lưu (F = 0 tức là line đã được ghi ở block nhớ). Kỹ thuật này làm tăng tốc đọ thao tác với bộ nhớ và được gọi là kỹ thuật sao lưu dựng cờ.

Sau khi đã ghi nọi dung của line ra block nhớ, thì số hiệu của block nhớ vừa được nạp sẽ ghi vào trường tag của line.

* 1. Trình bày phương pháp ánh xạ trực tiếp trong tổ chức bộ nhớ cache.

Mỗi Block của bộ nhớ chính chỉ có thể được nạp vào một Line của cache:

* + B0 → L0
  + B1 → L1
  + ....
  + Bm-1 → Lm-1
  + Bm → L0
  + Bm+1 → L1
  + ....

Tổng quát

* + Bj chỉ có thể nạp vào L j mod m m là số Line của cache
  1. Trình bày thao tác đọc cache theo phương pháp ánh xạ trực tiếp trong tổ chức bộ nhớ cache.

Ký hiệu LI là số hiệu line cần truy cập

BI là số hiệu block cần truy cập

Bước 1: CPU kiểm tra tính hợp lệ của địa chỉ truy cập

Trương hợp 1: nếu số bit có nghĩa biểu diễn địa chỉ truy cập lớn hơn số bit hợp lệ ( lớn hơn n), địa chỉ này khong hợp lệ, tiến hành phải dừng lại.

Trường hợp 2: nếu LI ≠ BI mod 2n2, thì địa chỉ này không hợp lệ, tiến hành phải dừng (số lượng line trong cache là 2n2).

Ngược lại, CPU thực hiện phát ra địa chỉ truy cập tren bus A.

Bước 2: căn cứ vào n2 bit dùng để so sánh cho line bộ điều khiển cache xác định được line cần truy cập.

Bước 3: bộ điều khiển cache đọc giá trị trên tag thuộc line vừa tìm được và so sánh với giá trị do n3 bit xác định.

Nếu đây là trường hợp cache hit, thực hiện bước 4 của thao tác.

Nếu là trường hợp cache miss thì CPU phải truy cập vào block nhớ có địa chỉ xác định bởi n3 bit và nạp nội dung của nó vào line vừa tìm được. trước khi nạp vào cache, nếu F =1 thì CPU sao lưu dữ liệu đang có trong line tìm được vào bộ nhớ chính theo địa chỉ của nó ( tại vị trí khối có số hiệu trùng với giá trị tag trong line), sửa F =0 (ghi nhập nội dung line trùng với nội dung 1 block trên bộ nhớ chính). Sau đó mới nạp dữ liệu từ block vừa nạp vào trong tag của line. Và cuối cùng thực hiện bước 4 của thao tác.

Bước 4: byte dữ liệu cần đọc có địa chỉ do n1 bit thấp xác định trong line được đọc vào CPU.

* 1. CPU có 24 bit địa chỉ, bộ nhớ chính 256 KB chia làm 512 block nhớ, bộ nhớ cache có dung lượng 8 KB. Khi CPU được lệnh phát ra địa chỉ truy nhập bộ nhớ là B7281Ah. Hãy trình bày chi tiết phương pháp đọc cache theo kỹ thuật ánh xạ trực tiếp và địa chỉ ô nhớ cần truy cập trong block cho trường hợp phát ra địa chỉ trên.

Bước 1: xác định dung lượng từng block(line) và số bit W:

CBLOCK = CMEMORY / nBLOCK = 256KB/512 =0,5 KB = 512 byte =29 byte.

🡺số bit để đánh địa chỉ lệnh trong block(line) là W =9 bits

Bước 2: xác định số lne trong cache và số Bit L:

nline- cache = Ccache / Cblock = 8KB / 0,5 KB =16 = 24 (lines)

🡺L = 4 bits.

Bước 3: xác định số bit để đánh số hiệu block trong bộ nhớ chính:

Nblock = 512 =29 🡺T =9 bits

Bước 4: Số bit địa chỉ hợp lệ: n= W + L+ T = 9+4+9=22.

Bước 5: đọc cache với giá trị địa chỉ B7218A h:

B7218Ah=101101110010000110001010🡺 địa chỉ không hợp lệ vì số bit là 24> n=22

* 1. CPU có 24 bit địa chỉ, bộ nhớ chính 256 KB chia làm 512 block nhớ, bộ nhớ cache có dung lượng 8 KB. Khi CPU được lệnh phát ra địa chỉ truy nhập bộ nhớ là 2F1025h. Hãy trình bày chi tiết phương pháp đọc cache theo kỹ thuật ánh xạ trực tiếp cho trường hợp phát ra địa chỉ trên và địa chỉ ô nhớ cần truy cập trong block theo từng trường hợp.

Bước 1: xác định dung lượng từng block(line) và số bit W:

CBLOCK = CMEMORY / nBLOCK = 256KB/512 =0,5 KB = 512 byte =29 byte.

🡺số bit để đánh địa chỉ lệnh trong block(line) là W =9 bits

Bước 2: xác định số lne trong cache và số Bit L:

nline- cache = Ccache / Cblock = 8KB / 0,5 KB =16 = 24 (lines)

🡺L = 4 bits.

Bước 3: xác định số bit để đánh số hiệu block trong bộ nhớ chính:

Nblock = 512 =29 🡺T =9 bits

Bước 4: Số bit địa chỉ hợp lệ: n= W + L+ T = 9+4+9=22.

Bước 5: đọc cache với giá trị địa chỉ 2F1025 h:

* + 2F1025h=1011110001000000100101🡺 địa chỉ hợp lệ vì số bit là 22.
  + Số line trong cache cần truy cập: 1000 = 8
  + Bộ nhớ block cần truy cập là: Block =101111000= 28 +26 +25 +24 +23 = 376🡺 block nhớ cần truy cập là 376.
  + Xác định block nạp đúng địa chỉ line cache không:

Số hiệu line = số hiệu block mod số line của cache

→ số hiệu line = 376 mod 16 = 8→ block nạp đúng → địa chỉ hợp lệ

==>CPU phát địa chỉ của ô nhớ cần truy cập ra bus địa chỉ.

Bước 6: bộ điều khiển cache thực hiện truy cập vào line 8, đọc giá trị trên tag của nó và thực hiện so sánh với Block 376. Có 2 th xảy ra:

* Khả năng 1: Miss cache → **Tag** ≠ 376.
  + CPU phải tiến hành nạp cache: trước khi nạp cache, nếu line 8 có F = 1, tức là nội dung có sự thay đổi, CPU phải ghi nội dung của nó ra block nhớ có số hiệu bằng giá trị trên **tag** của line, sửa F = 0 (còn trường hợp F = 0 thì thao tác này không cần thiết).
  + CPU ra bộ nhớ đọc block nhớ 376, nạp vào line 8, sửa tag của line 8 thành 376. Cuối cùng CPU tiến hành đọc byte trong line.
* Khả năng 2: Hit cache → **Tag** = 376 → CPU đọc byte trong line 4 có địa chỉ tương đối là 9 bits thấp của địa chỉ W = 000100101 = 25 + 22 +20= 37.
* Địa chỉ vật lý của ô nhớ trong bộ nhớ chính được tính bằng cách ghép địa chỉ của block và địa chỉ của từ nhớ lại.

→ Địa chỉ vật lý = TW

→ Địa chỉ vật lý = ***101111000000100101***= 2F0**25h**

* 1. CPU có 24 bit địa chỉ, bộ nhớ chính 256 KB chia làm 512 block nhớ, bộ nhớ cache có dung lượng 8 KB. Khi CPU được lệnh phát ra địa chỉ truy nhập bộ nhớ là 2F0825h. Hãy trình bày chi tiết phương pháp đọc cache theo kỹ thuật ánh xạ trực tiếp cho trường hợp phát ra địa chỉ trên và địa chỉ ô nhớ cần truy cập trong block theo từng trường hợp.

Bước 1: xác định dung lượng từng block(line) và số bit W:

CBLOCK = CMEMORY / nBLOCK = 256KB/512 =0,5 KB = 512 byte =29 byte.

🡺số bit để đánh địa chỉ lệnh trong block(line) là W =9 bits

Bước 2: xác định số lne trong cache và số Bit L:

nline- cache = Ccache / Cblock = 8KB / 0,5 KB =16 = 24 (lines)

🡺L = 4 bits.

Bước 3: xác định số bit để đánh số hiệu block trong bộ nhớ chính:

Nblock = 512 =29 🡺T =9 bits

Bước 4: Số bit địa chỉ hợp lệ: n= W + L+ T = 9+4+9=22.

Bước 5: đọc cache với giá trị địa chỉ 2F1025 h:

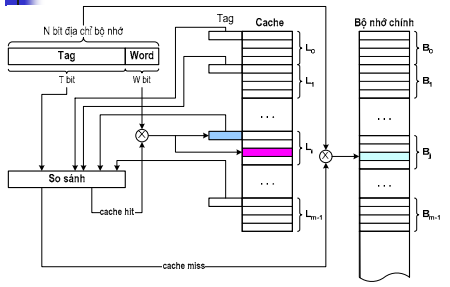
* + 2F0825h=1011110000100000100101🡺 địa chỉ hợp lệ vì số bit là 22.
  + Số line trong cache cần truy cập: 0100 = 4
  + Bộ nhớ block cần truy cập là: Block =101111000= 28 +26 +25 +24 +23 =376🡺 block nhớ cần truy cập là 376.
  + Xác định block nạp đúng địa chỉ line cache không:

Số hiệu line = số hiệu block mod số line của cache

→ số hiệu line = 376 mod 16 = 8→ block nạp không đúng → địa chỉ không hợp lệ

* 1. Trình bày phương pháp ánh xạ liên kết hoàn toàn trong tổ chức bộ nhớ cache.

Bài làm



* Mỗi Block có thể được nạp vào line bất kỳ của cache.
* Địa chỉ của bộ nhớ chính bao gồm hai trường:
  + Trường **Word** giống như trường hợp ở trên.
  + Trường ***Tag***dùng để xác định *Block* của bộ nhớ chính.
* Tag xác định Block nào đang nằm ở Line đó.
  1. Trình bày thao tác đọc cache theo phương pháp ánh xạ liên kết hoàn toàn

Bước 1: CPU kiểm tra tính hợp lệ của địa chỉ truy cập, nếu số bit có nghĩa biểu diễn địa chỉ truy cập lớn hơn số bit hợp lệ ( lớn hơn n), địa chỉ này khong hợp lệ, tiến hành phải dừng lại.

Bước 2: bộ điều khiển đọc lần lượt các giá trị trên tag thuộc từng line trong cache và so sánh với giá trị T bit dùng để xác định.

Nếu đây là trường hợp cache hit, thực hiện bước 3 của thao tác.

Nếu là trường hợp cache miss thì CPU chọn 1 line nào đó đẻ nạp cache, nếu không có line rỗng CPU phải chọn 1 line đang chứa block nhớ để nạp cache, nếu F =1 thì CPU sao lưu dữ liệu đang có trong line ra block nhớ có số hiệu bằng giá trị trên tag của line, sửa F =0 (ghi nhận nội dung line trùng với nội dung 1 block trên bộ nhớ chính). Sau đó truy cập vào block vừa nạp. Và cuối cùng thực hiện bước 3 của thao tác.

Bước 3:byte dữ liệu cần đọc có địa chỉ do n1 bit thấp xác định trong line được đọc vào CPU.

* 1. CPU có 24 bit địa chỉ, bộ nhớ chính 256 KB, 512 block nhớ, bộ nhớ cache 8 KB. Khi CPU được lệnh phát ra địa chỉ truy nhập bộ nhớ là 3280Ah. Hãy trình bày phương pháp đọc cache theo kỹ thuật ánh xạ liên kết hoàn toàn và chỉ ra địa chỉ ô nhớ cần truy cập trong bộ nhớ theo trường hợp phát ra địa chỉ trên.
* Bước 1: Xác định dung lượng của 1 block (line) nhớ.→ W = 9 bits
* Bước 2: Xác định số bít cần thiết để đánh số hiệu block trong bộ nhớ.

Số block = 512 = 29 → Cần T=9 bits để đánh số hiệu block.

* Bước 3: Xác định số bít địa chỉ hợp lệ cho bộ nhớ 256KB là: 9 + 9 = 18 bits.
* Bước 4: Đọc cache với địa chỉ 3280Ah

Chuyển địa chỉ từ hệ hex sang hệ nhị phân: 3280Ah = **110010100**000001010

* + Kiểm tra xem địa chỉ cần truy cập có hợp lệ không: Hợp lệ vì số bít của địa chỉ là 18 bằng số bít địa chỉ hợp lệ → CPU phát địa chỉ này lên bus địa chỉ.
  + Bộ điều khiển cache xác định giá trị của trường Tag là 9 bít cao của địa chỉ trên để xác định số hiệu block bộ nhớ cần truy cập: **110010100**= 404.
  + Bộ điều khiển cache sẽ so sánh trường Tag của tất cả các line với 404. Có 2 khả năng xảy ra.
  + Trường hợp 1: Không có Tag của line nào có giá trị 388 → Block cần truy cập chưa có trong cache (miss cache). CPU phải nạp block có số hiệu 388 từ bộ nhớ chính vào cache. CPU tìm line nào trong cache sử dụng kém hiệu quả nhất để thay bằng block mới. Nếu bít F của line bị thay thế = 1 thì nội dung của line đó sẽ được ghi trả lại bộ nhớ vào đúng block có giá trị bằng trường Tag của line đó. Sau đó xóa F = 0 rồi nạp block nhớ có số hiệu 388 vào line này và cập nhật trường Tag của nó bằng 388. Sau đó CPU sẽ đọc byte cần trong line này.
  + Trường hợp 2: Có một Tag của line nào đó = 404 (hit cache). CPU sẽ đọc byte có số hiệu là 9 bit thấp của địa chỉ: 000001010 = 10 → đọc byte có số hiệu 10 của line đó.
  + Địa chỉ vật lý của ô nhớ cần truy cập chính là: 3280Ah
  1. CPU có 24 bit địa chỉ, bộ nhớ chính 256 KB, 512 block nhớ, bộ nhớ cache 8 KB. Khi CPU được lệnh phát ra địa chỉ truy nhập bộ nhớ là 7280Ah. Hãy trình bày phương pháp đọc cache theo kỹ thuật ánh xạ liên kết hoàn toàn và chỉ ra địa chỉ ô nhớ cần truy cập trong bộ nhớ theo trường hợp phát ra địa chỉ trên.

Bước 1:Xác định dung lượng của 1 block (line):

Cblock = Cmemory/nblock = 256KB/512 = 256KB/0.5K = 512 B = 29 B🡪 W (n1) = 9 (bits).

Bước 2: Xác định số bit để đánh số hiệu cho block.

Nblock = 512 (block) = 29 (block) 🡪 T (n3) = 9 (bits).

Bước 3:Xác định số line của cache.

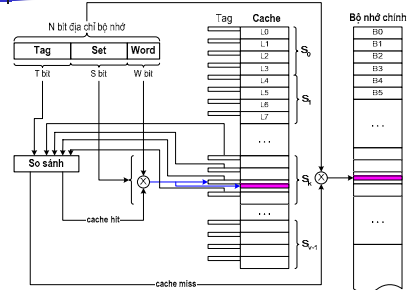
Nline = Ccache/Cblock = 8KB/0.5KB = 16 (lines)

Bước 4:Xác định chiều dài tối đa của địa chỉ hợp lệ:

N = T + W = 9 + 9 = 18 (bits).

Bước 5: Xác định địa chỉ CPU phát ra có hợp lệ không?

Địa chỉ CPU phát là: 7280Ah = 1110010100000001010b 🡪 Địa chỉ này có độ dài 19 bits🡪Địa chỉ này không hợp lệ.

* 1. Trình bày phương pháp ánh xạ liên kết tập hợp trong tổ chức bộ nhớ cache.
* Cache đươc chia thành các Tập (Set)
* Mỗi một Set chứa một số Line
* Ánh xạ theo nguyên tắc sau:
  + B0 → S0
  + B1 → S1
  + B2 → S2
  + …
  + Bm → Sm mod q
* N bits địa chỉ hợp lệ được chia làm 3 phần:
  + W bits thấp nhất dùng để xác định số hiệu của từ nhớ trong line (block) cần truy cập → kích thước của một block = 2W
  + S bits kế tiếp dùng để xác định một trong 2S set cần truy cập.
  + T bits trọng số cao nhất dùng để xác định block cần truy cập.→ N = T + S + W
  1. Trình bày thao tác đọc cache theo phương pháp ánh xạ liên kết tập hợp trong tổ chức bộ nhớ cache.

**SI (Set Index)– là số hiệu Set cần truy nhập.**

**BI (Block Index)- là số hiệu Block nhớ cần truy nhập.**

**Bước 1: CPU kiểm tra tính hợp lệ của địa chỉ truy nhập.**

**Trường hợp 1**: Nếu số bit có nghĩa biểu diễn địa chỉ truy nhập lớn hơn số bit hợp lệ (lớn hơn n), thì địa chỉ này không hợp lệ, tiến trình phải dừng.

**Trường hợp 2**: Nếu SI ≠ (BI mod n2), thì địa chỉ này không hợp lệ, tiến trình phải dừng (Số lượng Set trong cache là n2).

Ngược lại, CPU thực hiện phát ra địa chỉ truy nhập lên Bus A, sang bước 2.

**Bước 2:** Căn cứ vào n2 bit dùng để đánh số hiệu cho set bộ điều khiển cache xác định được set cần truy cập.

**Bước 3:**bộ điều kiển cache đọc lần lượt từng giá trị trên trên tag thuộc từng ine trong set vừa tìm được và so sánh với giá trị BI do n3 xác định

Nếu đây là trường hợp cache hit, thực hiện bước 4 của thao tác.

Nếu là trường hợp cache miss thì CPU phải tiến hành nạp cache: CPU phải chọn 1 line nào đó trong set đẻ nạp cache, nếu không có line rỗng CPU phải phải chọn 1 line đang chứa block nhớ đẻ nạp cache, nếu F =1 thì CPU sao lưu dữ liệu đang có trong line ra block nhớ có số hiệu bằng giá trị trên tag line, sửa F =0 (ghi nhập nội dung line trùng với nội dung 1 block trên bộ nhớ chính). Sau đó truy cập block nhớ có địa chỉ xác định bơi n3 bit cao trong n bit và nạp nội dung của nó vào line được chọn, sửa tag thành số hiệu block vừa nạp. Và cuối cùng thực hiện bước 4 của thao tác.

Bước 4: byte dữ liệu cần đọc có địa chỉ do n1 bit thấp xác định trong line được đọc vào CPU.

* 1. CPU có 24 bit địa chỉ, bộ nhớ chính 256 KB chia làm 512 block nhớ, bộ nhớ cache có dung lượng 8 KB chia làm 4 set. Khi CPU được lệnh phát ra địa chỉ truy nhập bộ nhớ là 22A07h. Hãy trình bày chi tiết phương pháp đọc cache và chỉ ra địa chỉ ô nhớ cần truy cập trong bộ nhớ
* Bước 1: xác định dung lượng của 1 Block( line)

Cblock=Cmemory/Nblock=256KB/512=29🡪W=9( bits)

* Bước 2 : xác định số lượng line của cache

Nline=Ccache/Cblock=8/0.5KB=16=24🡪L=4(bits)

* Bước 3: xác định số line trong 1 set

Nline-set=Nline/Nset=16/4=4(lines)

* Bước 4: xác định số bits để đánh số hiệu của set

Nset=4=22🡪 S=2

* Bước 5: xác định số bits để đánh số hiệu cho Block

Nblock=512=29🡪 T=9 bits

* Bước 6: xác định số bits của địa chỉ hợp lệ

N=T+S+W=9+2+9=20(bits)

* Bước 7: đọc cache với địa chỉ 22A07h

22A07h=100010101000000111b🡪địa chỉ có 18 bits<N=20 bits🡪 địa chỉ hợp lệ

S=01b🡪 số hiệu của tập cần truy cập Sl=1

T=1000101b=1+4+26=69🡪Bl=69

Kiểm tra xem block có số hiệu 69 nạp vào set 1 của cache có hợp lệ không: 69 mod 4 =1🡪 hợp lệ

* Bước 8. Truy xuất cache.

Từ địa chỉ CPU phát ra thì đơn vị điều khiển cache (MMU) sẽ đi kiểm tra 4 line của set 2 xem trường tag có bằng với 69 không? Có 2 khả năng xảy ra.

* Không có trường tag nào bằng 69 🡪 miss cache …
* Có 1 tag của line nào đó trong set 2 bằng 69 🡪 hit cache 🡪 CPU sẽ đọc ô nhớ có số hiệu (địa chỉ lệch) w = 000000111b = 1+2+4 = 7.
* Địa chỉ vật lý: TW = 1000101000000111b = 8A07h.
  1. CPU có 24 bit địa chỉ, bộ nhớ chính 256 KB chia làm 512 block nhớ, bộ nhớ cache 8 KB. Khi CPU được lệnh phát ra địa chỉ truy nhập bộ nhớ là 2280Ah. Hãy trình bày chi tiết phương pháp đọc cache theo kỹ thuật ánh xạ liên kết tập hợp và chỉ ra địa chỉ ô nhớ cần truy cập trong bộ nhớ.

Bước 1: xác định dung lượng của 1 Block( line)

Cblock=Cmemory/Nblock=256KB/512=29🡪W=9( bits)

Bước 2 : xác định số lượng line của cache

Nline=Ccache/Cblock=8/0.5KB=16=24🡪L=4(bits)

Bước 3: xác định số line trong 1 set

Nline-set=Nline/Nset=16/4=4(lines)

Bước 4: xác định số bits để đánh số hiệu của set

Nset=4=22🡪 S=2

Bước 5: xác định số bits để đánh số hiệu cho Block

Nblock=512=29🡪 T=9 bits

Bước 6: xác định số bits của địa chỉ hợp lệ

N=T+S+W=9+2+9=20(bits)

Bước 7: đọc cache với địa chỉ 2280Ah

2280Ah=100010100000001010b🡪địa chỉ có 18 bits<N=20 bits🡪 địa chỉ hợp lệ

S=00b🡪 số hiệu của tập cần truy cập Sl=0

T=1000101b=1+4+26=69🡪Bl=69

Kiểm tra xem block có số hiệu 69 nạp vào set 1 của cache có hợp lệ không: 69 mod 4 =1🡪không hợp lệ

* 1. CPU có 24 bit địa chỉ, bộ nhớ chính 256 KB chia làm 512 block nhớ, bộ nhớ cache có dung lượng 8 KB. Khi CPU được lệnh phát ra địa chỉ truy nhập bộ nhớ là 120812h. Hãy trình bày chi tiết phương pháp đọc cache theo kỹ thuật ánh xạ liên kết tập hợp và chỉ ra địa chỉ ô nhớ cần truy cập trong bộ nhớ.

Bước 1: xác định dung lượng của 1 Block( line)

Cblock=Cmemory/Nblock=256KB/512=29🡪W=9( bits)

Bước 2 : xác định số lượng line của cache

Nline=Ccache/Cblock=8/0.5KB=16=24🡪L=4(bits)

Bước 3: xác định số line trong 1 set

Nline-set=Nline/Nset=16/4=4(lines)

Bước 4: xác định số bits để đánh số hiệu của set

Nset=4=22🡪 S=2

Bước 5 xác định số bits để đánh số hiệu cho Block

Nblock=512=29🡪 T=9 bits

Bước 6: xác định số bits của địa chỉ hợp lệ

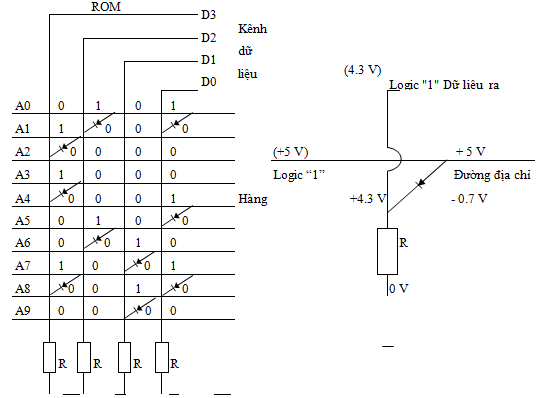
N=T+S+W=9+2+9=20(bits)

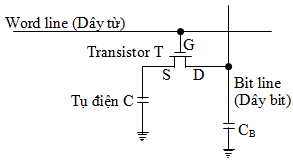
Bước 7: đọc cache với địa chỉ 120812h

120812h=000100100000100000010010b🡪địa chỉ có 24 bits>N=20 bits🡪 địa chỉ không hợp lệ

* 1. Cho sơ đồ mạch ROM mặt nạ. Hãy xác định giá trị dữa liệu tại các địa chỉ

|  |  |
| --- | --- |
| Địa chỉ | Giá trị dữ liệu |
| 000000011b | 1101 |
| 000000101b | 0001 |
| 000000111b | 0101 |



* 1. Trình bày nguyên lý ghi, đọc của phần tử nhớ động 1 bóng sau:
* Việc ghi:

Khi dây từ có mức tích cực (=1), T ở trạng thái mở, nối tụ C với dây bit. Nếu thao tác là ghi thì giá trị cần ghi phải đạt trên dây bit. Nếu giá trị đó là 1 thì tụ C sẽ được nạp tới ddienj áp tương ứng với giá rị 1 trên dây bit, còn nếu giá trị đó là 0 thì tụ C sẽ bị phóng hết điện, tức có giá trị =0.

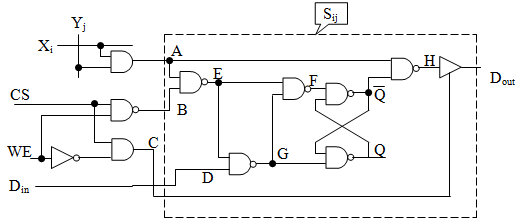
* Việc đọc:

Việc đọc phức tạp hơn việc ghi 1 chút do diện tích trên tụ C ứng với giá trị cần đọc rất nhỏ. Điện áp trên tụ C điều khiển mức điện áp trên tụ Cb, tức đưa ra dữ liệu. giả sử trên C có mức điện áp V, thì sau khi đọc mức điện áp trên dây là :

VB = V x C/(C+B).

Do nhiều phần tử nối vào dây bit nên điện dung CB tương đói lớn (CB >> C). Dữ liệu đọc ra có mức điện bé: giá trị mức logic 0 và mức logic 1 trên dây bit không khác xa nhau. Để xác định mức điện áp thay đổi nhỏ trên dây bit cần phải dùng bộ khuếch đại đọc ra đọ nhạy cao.

* 1. Cho phần tử nhớ sau:

a.Chứng minh với Yj = 1, Xi = 1, WE = 1 và CS = 1 thì Q = Din đồng thời H và Dout ngắt mạch:

theo đề ra ta có:

=>C = 0, lúc này đầu ra DOUT ở trạng thái trở kháng cao, có thể nói đường dây ra DOUT ngắt mạch với phần tử nhớ.

WE=1🡺/WE=0.

C= AND ( CS;/WE) = AND (1;0) = 0.

A= AND ( Xi; Y j) = AND (1;1) =1

B=NAND (CS; WE ) = NAND (1;1) = 0.

E=NAND (A ;B) =NAND (1; 0) = 1.

Khả năng 1: DIN=0.

G= NAND(E;DIN) =NAND(1;0) =1.

F = NAND(E;G )= NAND( 1;1) =0.

/Q = NAND (F;Q) = NAND(0;Q)= 1🡺 Q=0 (1)

Khả năng 2: DIN= 1

G= NAND ( E; DIN) =NAND (1; 1)= 0.

Q= NAND (G; /Q) =NAND( 0;/Q)=1 (2)

(1) và (2) 🡺 Q=DIN

b.Yj = 1, Xi = 1, WE = 0 và CS = 1 thì Q không phụ thuộc Din và Dout = Q:

C= AND ( CS;/WE) = AND (1;1) = 1 🡺 DOUT=H.

A= AND ( XI; YJ ) = AND (1;1)=1 🡺 DOUT = Q.

H= AND (A; /Q ) = AND (1;/Q)= Q

B=NAND (CS; WE ) = NAND (1;0) = 1.

E=NAND (A ;B) =NAND (1; 1) = 0.

F = NAND(E;G )= NAND( 0;G) =1.

Xét Qt-1 = 0:

/Q(t) = NAND ( F; Qt-1) = NAND (1; Qt-1) = NAND( 1; 0) =1 🡺Q(t)= 0 (1)

Xét Qt-1 = 1:

/Q(t) = NAND ( F; Qt-1) = NAND (1; Qt-1) = NAND( 1; 1) =0 🡺Q(t)= 1 (2)

(1),(2) 🡺 Qt =Qt-1 🡺 Q không đổi.

c.Yj = 1, Xi = 1, WE = 0/1 và CS = 0 thì Q không phụ thuộc Din, đồng thời H và Dout ở trạng thái trở kháng cao (ngắt mạch).

theo đề ra ta có:

WE=1🡺/WE=0.

C= AND ( CS;/WE) = AND (0;1) = 0. =>C = 0, lúc này đầu ra DOUT ở trạng thái trở kháng cao, có thể nói đường dây ra DOUT ngắt mạch với phần tử nhớ.

A= AND ( XI; YJ ) = AND (1;1)=1.

B=NAND (CS; WE ) = NAND (0;1) = 1.

E=NAND (A ;B) =NAND (1; 1) = 0.

F = NAND(E;G )= NAND( 0;G) =1.

Xét Qt-1 = 0:

/Q(t) = NAND ( F; Qt-1) = NAND (1; Qt-1) = NAND( 1; 0) =1 🡺Q(t)= 0 (1)

Xét Qt-1 = 1:

/Q(t) = NAND ( F; Qt-1) = NAND (1; Qt-1) = NAND( 1; 1) =0 🡺Q(t)= 1 (2)

(1),(2) 🡺 Qt =Qt-1 🡺 Q không đổi.

WE=0🡺/WE=1.

A= AND ( XI; YJ ) = AND (1;1)=1.

B=NAND (CS; WE ) = NAND (0;0) = 1.

E=NAND (A ;B) =NAND (1; 1) = 0.

F = NAND(E;G )= NAND( 0;G) =1.

Xét Qt-1 = 0:

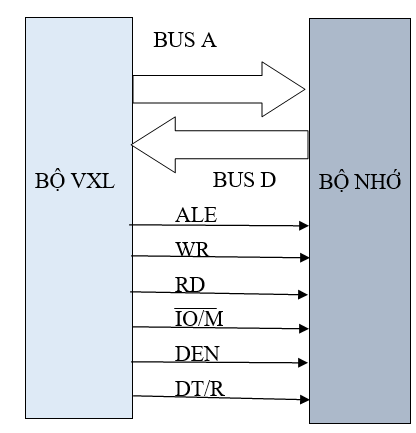
/Q(t) = NAND ( F; Qt-1) = NAND (1; Qt-1) = NAND( 1; 0) =1 🡺Q(t)= 0 (1)

Xét Qt-1 = 1:

/Q(t) = NAND ( F; Qt-1) = NAND (1; Qt-1) = NAND( 1; 1) =0 🡺Q(t)= 1 (2)

(1),(2) 🡺 Qt =Qt-1 🡺 Q không đổi.

* 1. Vẽ sơ đồ khối ghép nối giữa CPU và bộ nhớ chính, trình bày chức năng của các đường dây ghép nối.

1.ALE:để chốt địa chỉ nhớ. Địa chỉ này được chốt nên có thể tiến hành việc đọc ghi cho đến ki ALE trở lại mức cao.

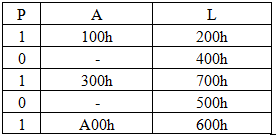
2.WR/RD(Write/ Read): khi địa chỉ được mở, bộ vi xử lý sẽ tác động vào tín hiệu WR hoặc RD đẻ tiến hành quá trình ghi đọc bộ nhớ. Tín hiwwuj 3 trạng thái có mức logic thấp. hoắc mức logic cao và không thể mở cả 2 ô nhớ cùng lúc.

3.IO/M(Input-Output/Memmory): tín hiệu cho phép chọn cổng ra vào hoặc bộ nhớ. Tín hiệu này cần thiết vì ghi cùng 1 địa chỉ có thể được dùng cho bộ nhớ hoặc một cổng I/O.

4. DEN(Data Enable): để phân định thười gian cho BUS,báo hiệu cho BUS liệu đã ổn định và mở thông mạch đệm. hầu hết các hệ thống nối với BUS dữ liệu đều phải qua mạch đệm. tín hiệu DEN là tín hiệu trạng thái hiệu lực dùng đẻ mở mạch này.

5. DT/R( Data transmit/ Receive): dùng đẻ xác định chiều truyền dữ liệu. mức cao lhi bọ xử lý truyền dữ liệu và mức thấp khi nhập dữ liệu.

* 1. Giả sử bộ nhớ vật lý có dung lượng 512 MB, chương trình gồm 5 module, xác định địa chỉ vật lý tương ứng với các địa chỉ logic phát ra là <2, 43h>, <3, 4FFh>, <4, 600h>, <5, 100h>, cho biết bảng quản lý phân đoạn như sau:



1. Với các địa chỉ logic phát ra là <2, 43h>

- Bảng SCB cho thấy chương trình có 5 đoạn, đánh số từ 0 đến 4.

- Số hiệu đoạn cần truy nhập là S = 2 < 5, địa chỉ đoạn cần truy nhập là hợp lệ.

- Phần tử 2 trong bảng SCB có P = 1, cho biết đoạn 2 đã được nạp vào vùng nhớ có địa chỉ đầu A = 300h.

- Phần tử 2 trong bảng SCB, cho biết đoạn 2 có độ dài là L = 700h byte, so sánh : L = 700h > d = 43h, nên địa chỉ lệch cần truy nhập là hợp lệ

- Địa chỉ vật lý cần truy nhập = A + d = 300h + 43h = 343h

1. Với các địa chỉ logic phát ra là <4, 4FFh>

- Bảng SCB cho thấy chương trình có 5 đoạn, đánh số từ 0 đến 4.

- Số hiệu đoạn cần truy nhập là S = 4 < 5, địa chỉ đoạn cần truy nhập là hợp lệ.

- Phần tử 4 trong bảng SCB có P = 1, cho biết đoạn 4 đã được nạp vào vùng nhớ có địa chỉ đầu A = A00h.

- Phần tử 4 trong bảng SCB, cho biết đoạn 4 có độ dài là L = 600h byte, so sánh : L = 600h > d = 4FFh, nên địa chỉ lệch cần truy nhập là hợp lệ

- Địa chỉ vật lý cần truy nhập = A + d = 600h + 4FFh = AFFh

1. Với các địa chỉ logic phát ra là <3, 4FFh>

- Bảng SCB cho thấy chương trình có 5 đoạn, đánh số từ 0 đến 4.

- Số hiệu đoạn cần truy nhập là S = 3 < 5, địa chỉ đoạn cần truy nhập là hợp lệ.

- Phần tử 3 trong bảng SCB có P = 0, cho biết đoạn 3 đã chưa được nạp 🡺 dừng lại.

1. Với các địa chỉ logic phát ra là <1, 306h>

- Bảng SCB cho thấy chương trình có 5 đoạn, đánh số từ 0 đến 4.

- Số hiệu đoạn cần truy nhập là S = 1 < 5, địa chỉ đoạn cần truy nhập là hợp lệ.

- Phần tử 1 trong bảng SCB có P = 0, cho biết đoạn 1 đã chưa được nạp 🡺 dừng lại.

- Địa chỉ vật lý cần truy nhập = A + d = 300h + 43h = 343h

1. Với các địa chỉ logic phát ra là <4, 600h>

- Bảng SCB cho thấy chương trình có 5 đoạn, đánh số từ 0 đến 4.

- Số hiệu đoạn cần truy nhập là S = 4 < 5, địa chỉ đoạn cần truy nhập là hợp lệ.

- Phần tử 4 trong bảng SCB có P = 1, cho biết đoạn 4 đã được nạp vào vùng nhớ có địa chỉ đầu A = A00h.

- Phần tử 4 trong bảng SCB, cho biết đoạn 4 có độ dài là L = 600h byte, so sánh : L = 600h =d = 4FFh, nên địa chỉ lệch cần truy nhập là không hợp lệ

1. Với các địa chỉ logic phát ra là <2, 701h>

- Bảng SCB cho thấy chương trình có 5 đoạn, đánh số từ 0 đến 4.

- Số hiệu đoạn cần truy nhập là S = 2 < 5, địa chỉ đoạn cần truy nhập là hợp lệ.

- Phần tử 2 trong bảng SCB có P = 1, cho biết đoạn 2 đã được nạp vào vùng nhớ có địa chỉ đầu A = 300h.

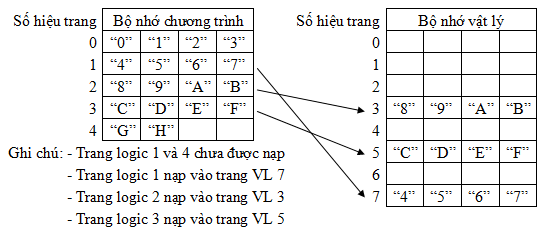
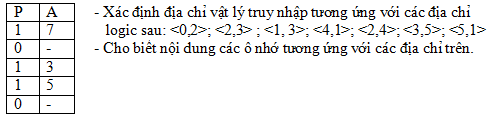
- Phần tử 2 trong bảng SCB, cho biết đoạn 2 có độ dài là L = 700h byte, so sánh : L = 700h < d = 701h, nên địa chỉ lệch cần truy nhập là KHÔNG hợp lệ

1. Với các địa chỉ logic phát ra là <5, 33h>

- Bảng SCB cho thấy chương trình có 5 đoạn, đánh số từ 0 đến 4.

- Số hiệu đoạn cần truy nhập là S = 5 =nSCB, địa chỉ đoạn cần truy nhập là không hợp lệ.

* 1. Giả sử bộ nhớ vật lý, kích thước 32 bytes, chia làm 8 trang vật lý. Chương trình có kích thước 18 bytes có nội dung ‘0123456789ABCDEFGH’. Và bảng quản lý trang (PCB) có nội dung sau:



\* Phần tính toán chung

- Kích thước một trang vật lý là: L = dung lượng bộ nhớ vật lý / số trang vật lý.

= 32/8 = 4 bytes = 22.

- Vậy số bit để đánh độ lệch trong trang là: n1 = 2.

- Số bit để đánh số hiệu trang vật lý là: 8 = 23, vậy n2 = 3.

- Số bit để đánh địa chỉ vật lý là: 32 = 25, vậy n = 5 (n = n2 + n1).

- Số trang logic của chương trình = dung lượng chương trình/ L = 18/4 = 4,5. Vậy số trang logic của chương trình phải là 5. (bảng PCB trên là phù hợp).

a. Xác định địa chỉ vật lý (ĐCVL) truy nhập tương ứng với địa chỉ logic <0,2>

- Số hiệu trang logic cần truy nhập là: Np = 0 < 5 (số trang logic của chương trình) . nên số hiệu trang cần truy nhập là hợp lệ.

- Địa chỉ lệch cần truy nhập trong trang là: d = 2 < 4 (độ dài trang), nên địa chỉ lệch cần truy nhập trong trang là hợp lệ.

- Phần tử 0 trong bảng PCB có P = 1, cho biết trang logic 1 đã được nạp vào trang vật lý A = 7. A = 7 < 8 (số trang vật lý), nên giá trị này là hợp lệ.

- ĐCVL cần truy nhập là:

ĐCVL = A SHL n1 OR d = 7 SHL 2 OR 2 = 0111b SHL 2 OR 10b

= 11100b OR 10b = 11110b = 30.

- Nhìn vào sơ đồ ta thây nội dung ô nhớ có địa chỉ 30 là “6”.

b. Xác định địa chỉ vật lý truy nhập tương ứng với địa chỉ logic <2,3>

- Số hiệu trang logic cần truy nhập là: Np = 2 < 5 (số trang logic của chương trình) . nên số hiệu trang cần truy nhập là hợp lệ.

- Địa chỉ lệch cần truy nhập trong trang là: d = 3 < 4 (độ dài trang), nên địa chỉ lệch cần truy nhập trong trang là hợp lệ.

- Phần tử 2 trong bảng PCB có P = 1, cho biết trang logic 3 đã được nạp vào trang vật lý A = 5. A = 5< 8 (số trang vật lý), nên giá trị này là hợp lệ.

- ĐCVL cần truy nhập là: ĐCVL = A SHL n1 OR d = 3 SHL 2 OR 3 = 0011b SHL 2 OR 11b

= 1100b OR 11b = 1111b = 15.

- Nhìn vào sơ đồ ta thây nội dung ô nhớ có địa chỉ 15 là “B”.

c. Xác định địa chỉ vật lý truy nhập tương ứng với địa chỉ logic <1,3>

- Số hiệu trang logic cần truy nhập là: Np = 1 < 5 (số trang logic của chương trình) . nên số hiệu trang cần truy nhập là hợp lệ.

- Địa chỉ lệch cần truy nhập trong trang là: d = 3 < 4 (độ dài trang), nên địa chỉ lệch cần truy nhập trong trang là hợp lệ.

- Phần tử 1 trong bảng PCB có P = 0, cho biết trang logic 0 chưa được nạp vào bộ nhớ vật lý, nên hệ thống cần nạp trang hoặc đổi trang.

d. Xác định địa chỉ vật lý truy nhập tương ứng với địa chỉ logic <4,1>

- Số hiệu trang logic cần truy nhập là: Np = 4 < 5 (số trang logic của chương trình) . nên số hiệu trang cần truy nhập là hợp lệ.

- Địa chỉ lệch cần truy nhập trong trang là: d = 1 < 4 (độ dài trang), nên địa chỉ lệch cần truy nhập trong trang là hợp lệ.

- Phần tử 4 trong bảng PCB có P = 0, cho biết trang logic 0 chưa được nạp vào bộ nhớ vật lý, nên hệ thống cần nạp trang hoặc đổi trang.

e. Xác định địa chỉ vật lý truy nhập tương ứng với địa chỉ logic <3,5>

- Số hiệu trang logic cần truy nhập là: Np = 3 < 5 (số trang logic của chương trình) . nên số hiệu trang cần truy nhập là hợp lệ.

- Địa chỉ lệch cần truy nhập trong trang là: d = 5 > 4 (độ dài trang), nên địa chỉ lệch cần truy nhập trong trang là không hợp lệ. Do vậy tiến trình phải dừng.

f. Xác định địa chỉ vật lý truy nhập tương ứng với địa chỉ logic <4,4>

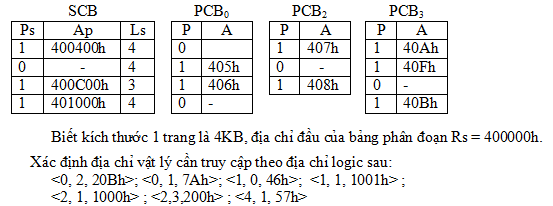
- Số hiệu trang logic cần truy nhập là: Np = < 5 (số trang logic của chương trình) . nên số hiệu trang cần truy nhập là hợp lệ.

- Địa chỉ lệch cần truy nhập trong trang là: d = 4= 4 (độ dài trang), nên địa chỉ lệch cần truy nhập trong trang là không hợp lệ. Do vậy tiến trình phải dừng.

g. Xác định địa chỉ vật lý truy nhập tương ứng với địa chỉ logic <5,3>

- Số hiệu trang logic cần truy nhập là: Np = 5 = 5 (số trang logic của chương trình) . nên số hiệu trang cần truy nhập là Không hợp lệ.

* 1. Giả sử bộ nhớ vật lý (BNVL) có dung lượng 32MB, chương trình gồm 4 modul, xác định địa chỉ truy cập trong chiến lược phân trang – phân đoạn, với:



\* Phần tính toán chung

- Kích thước một trang vật lý là:

L = 4 KB = 22\*210 bytes = 212 bytes

= 1000000000000B bytes = 1000H bytes

- Vậy số bit để đánh độ lệch trong trang là: n1 = 12.

- Số trang vật lý = dung lượng BNVL / kích thước 1 trang

= 64MB/4KB = 64\*210 KB/4 KB = 16 \* 210 = 24 \* 210 = 214

= 100000000000000B = 4000H

- Số bit để đánh số hiệu trang vật lý là: n2 = 14.

- Số bit để đánh địa chỉ vật lý là: 64 MB = 226, vậy n = 26 (n = n2 + n1).

a. Xác định địa chỉ vật lý cần truy cập theo địa chỉ logic <0, 2, 20Bh>

- Số hiệu đoạn cần truy nhập là: S = 0 < 4 (số đoạn của chương trình), địa chỉ đoạn cần truy nhập là hợp lệ.

- Số hiệu trang logic cần truy nhập là: Np = 2 < L0 = 4 (độ dài đoạn 0 – xem phần tử 0 bảng SCB), số hiệu trang logic cần truy nhập là hợp lệ.

- Địa chỉ lệch cần truy nhập trong trang là: d = 20BH < L = 1000H, địa chỉ lệch cần truy nhập trong trang là hợp lệ.

- Phần tử 0 trong SCB có Ps = 1, cho biết đoạn 0 đã được nạp và Ap = 400400h là địa chỉ đầu của bảng PCB0 trong bộ nhớ vật lý.

- Truy cập bảng PCB0, ta thấy phần tử 2 (quản lý trang Np = 2) cho biết trang Np =2 đã được nạp (P = 1) vào trang vật lý có số hiệu A = 406H.

- Vậy địa chỉ vật lý cần truy nhập là:

ĐCVL = A SHL n1 OR d = 406H SHL 12 OR 20BH

= 406000H OR 20BH = 40620BH.

b. Xác định địa chỉ vật lý cần truy cập theo địa chỉ logic <0, 1, 7Ah>

- Số hiệu đoạn cần truy nhập là: S = 0 < 4 (số đoạn của chương trình), địa chỉ đoạn cần truy nhập là hợp lệ.

- Số hiệu trang logic cần truy nhập là: Np = 1< L0 = 4 (độ dài đoạn 0 – xem phần tử 0 bảng SCB), số hiệu trang logic cần truy nhập là hợp lệ.

- Địa chỉ lệch cần truy nhập trong trang là: d = 7Ah < L = 1000h, địa chỉ lệch cần truy nhập trong trang là hợp lệ.

- Phần tử 0 trong SCB có Ps = 1, cho biết đoạn 0 đã được nạp và Ap = 400400h là địa chỉ đầu của bảng PCB0 trong bộ nhớ vật lý.

- Truy cập bảng PCB0, ta thấy phần tử 1 (quản lý trang Np = 1) cho biết trang Np =1 đã được nạp (P = 1) vào trang vật lý có số hiệu A = 405h.

- Vậy địa chỉ vật lý cần truy nhập là:

ĐCVL = A SHL n1 OR d = 405h SHL 12 OR 7Ah

= 405000h OR 7Ah = 40507Ah.

c. Xác định địa chỉ vật lý cần truy cập theo địa chỉ logic <1, 0, 46h>

- Số hiệu đoạn cần truy nhập là: S = 1 < 4 (số đoạn của chương trình), địa chỉ đoạn cần truy nhập là hợp lệ.

- Số hiệu trang logic cần truy nhập là: Np = 0 < L1 = 4 (độ dài đoạn 0 – xem phần tử 0 bảng SCB), số hiệu trang logic cần truy nhập là hợp lệ.

- Địa chỉ lệch cần truy nhập trong trang là: d = 46H < L = 1000H, địa chỉ lệch cần truy nhập trong trang là hợp lệ.

- Phần tử 1 trong SCB có Ps = 0, cho biết đoạn 1 chưa được nạp vào bộ nhớ vật lý. Hệ thống cần nạp đoạn hoặc đổi đoạn.

d. Xác định địa chỉ vật lý cần truy cập theo địa chỉ logic <1, 1, 1001h>

- Số hiệu đoạn cần truy nhập là: S = 1< 4 (số đoạn của chương trình), địa chỉ đoạn cần truy nhập là hợp lệ.

- Số hiệu trang logic cần truy nhập là: Np = 1< L0 = 4 (độ dài đoạn 0 – xem phần tử 0 bảng SCB), số hiệu trang logic cần truy nhập là hợp lệ.

- Địa chỉ lệch cần truy nhập trong trang là: d = 1001h > L = 1000h, địa chỉ lệch cần truy nhập trong trang là không hợp lệ.

e. Xác định địa chỉ vật lý cần truy cập theo địa chỉ logic <2, 1, 1000h>

- Số hiệu đoạn cần truy nhập là: S = 2 < 4 (số đoạn của chương trình), địa chỉ đoạn cần truy nhập là hợp lệ.

- Số hiệu trang logic cần truy nhập là: Np = 1 < L2 = 4 (độ dài đoạn 0 – xem phần tử 0 bảng SCB), số hiệu trang logic cần truy nhập là hợp lệ.

- Địa chỉ lệch cần truy nhập trong trang là: d = 1000H ≥ L = 1000H, địa chỉ lệch cần truy nhập trong trang là không hợp lệ, do vậy tiến trình phải dừng.

f. Xác định địa chỉ vật lý cần truy cập theo địa chỉ logic <2,3, 200h>

- Số hiệu đoạn cần truy nhập là: S =2< 4 (số đoạn của chương trình), địa chỉ đoạn cần truy nhập là hợp lệ.

- Số hiệu trang logic cần truy nhập là: Np = 3< L0 = 4 (độ dài đoạn 0 – xem phần tử 0 bảng SCB), số hiệu trang logic cần truy nhập là hợp lệ.

- Địa chỉ lệch cần truy nhập trong trang là: d = 200h < L = 1000h, địa chỉ lệch cần truy nhập trong trang là hợp lệ.

- Phần tử 2 trong SCB có Ps = 1, cho biết đoạn 2 đã được nạp và Ap = 400C00h là địa chỉ đầu của bảng PCB2 trong bộ nhớ vật lý.

- Truy cập bảng PCB2, ta thấy phần tử 3 (quản lý trang Np = 3) không có nên không hợp lý.

g. Xác định địa chỉ vật lý cần truy cập theo địa chỉ logic <4, 1, 57h>

- Số hiệu đoạn cần truy nhập là: S = 4 =4 (số đoạn của chương trình), địa chỉ đoạn cần truy nhập là không hợp lệ.

1. Kỹ thuật đường ống và RISC
   1. Trình bày kỹ thuật đường ống (pipeline) để đưa ra công thức tính thời gian thực hiện n lệnh, với ví dụ một đường ống gồm 6 công đoạn: Nhận lệnh, giải mã lệnh, tính địa chỉ toán hạng, nhận toán hạng, thi hành lệnh, ghi toán hạng.

Bài làm

Ý tưởng: Chia một lệnh ra làm nhiều công đoạn và cho xử lý các công đoạn gối lênh nhau.

1. Nhận lệnh (IF: Instruction Fetch).
2. Giải mã lệnh (ID: Instruction decode).
3. Tính địa chỉ toán hạng (Operand calculation - OC).
4. Nhận toán hạng (Operands Fetch – OF).
5. Thực hiện lệnh (Instruction execution – IE).
6. Ghi kết quả (WO – Write operation).
   1. Trình bày các khó khăn và hướng khắc phục với kỹ thuật đường ống đơn vị lệnh

Bài làm

* Xung đột về cấu trúc: tại 1 thời điểm có thể xử lý liền n lệnh, sẽ có thể xảy ra trường hợp có một số công đoạn của các lệnh khác nhau cần dùng chung một tài nguyên phần cứng.

Khắc phục: bằng cách tăng số tài nguyên lên.

* Xung đột về mặt dữ liệu:

VD: khi thực hiện 2 lệnh sau:

ADD R1, R2, R3; R3= R1+R2

ADD R3, R4, R5; R5= R3+R4

Ta nhận lệnh thứ 2 muốn thực hện thì lệnh thứ nhất phải hoàn thành xong, song 2 lệnh nằm sát cạnh nhau nên lệnh thứ 2 ở công đoạn nhận toán hạng ( cần R3) thì lệnh thứ nhất mới ở công đoạn xử lý lệnh. Như vậy, lệnh 2 phải chờ, hiệu quả thực hiện peppeline giảm đi.

Khắc phục: chương trình dịch sẽ sắp lại thứ tự lệnh để các lệnh gần nhau không ảnh hưởng lẫn nhau nếu việc sắp lại đó không ảnh hưởng đến sự thực thi của chương trình.

* Xung đột về mặt điều khiển: khi lệnh rẽ nhánh có điều kiện (IF…. THEN….ELSE) thì khi chưa kịp giửi mã lệnh, không biết lệnh rẽ theo nhánh nào. Do vậy phải chờ,…

Khắc phục: cho phép hệ thống được nạp lệnh tiếp theo cho cả 2 nhánh, khi giải mã xong điều kiện nhảy sẽ chỉ lấy kết quả thi hành của lệnh thỏa mãn điều kiện hoặc hệ thống phải có khả năng dự đoán rẽ nhánh chính xác. Tuy nhiên xử lý vấn đề xung đọt này không đơn giản.

* 1. Giả sử một lệnh được chia làm 5 công đoạn: nhận lệnh, giải mã lệnh, nhận toán hạng, xử lý, cất kết quả. Mỗi công đoạn thực hiện trong 5\*10-9 giây. Một lệnh nếu thực hiện tuần tự hết 22\*10-9 giây. So sánh thời gian thực hiện 40 lệnh giữa kỹ thuật pipeline và kỹ thuật tuần tự.

Gợi ý

- Áp dụng công thức Tp = m\*Tc + (n-1)\*Tc, trong đó:

Tp: thời gian thực hiện n lệnh theo kỹ thuật pipeline.

Tc: thời gian thực hiện một công đoạn lớn nhất trong m công đoạn theo kỹ thuật pipeline.

n: số lệnh.

M: số công đoạn trong một lệnh.

- Và công thức Tt = n\*T, trong đó:

T: thời gian thực hiện 1 lệnh khi thực hiện tuần tự.

Bài làm

Ta có: Tt = n \* T = 40 \* 22 \*10-9 (giây)

Tp = m\*Tc + (n-1)\*Tc = 5 \* (5\*10-9) + (40 -1)\* (5\*10-9)

= 44 \* (5\*10-9) (giây)

Vậy thực hiện 40 lệnh với kỹ thuật pipeline nhanh hơn 40 lệnh với kỹ thuật tuần tự là : Tt/Tp = 40\*22\*10-9 / (44\*5\*10-9) = 8\*22/44 = 4 lần.

* 1. Giả sử một lệnh được chia làm 6 công đoạn: nhận lệnh, giải mã lệnh, tính địa chỉ toán hạng, nhận toán hạng, xử lý, cất kết quả. Mỗi công đoạn thực hiện trong thời gian 5\*10-9 giây. Một lệnh nếu thực hiện tuần tự hết 22\*10-9 giây. So sánh thời gian thực hiện 50 lệnh giữa kỹ thuật pipeline và kỹ thuật tuần tự.

Bài làm

Ta có: Tt = n \* T = 50 \* 22 \*10-9 (giây)

Tp = m\*Tc + (n-1)\*Tc = 6 \* (5\*10-9) + (50 -1)\* (5\*10-9)

= 55 \* (5\*10-9) (giây)

Vậy thực hiện 40 lệnh với kỹ thuật pipeline nhanh hơn 40 lệnh với kỹ thuật tuần tự là : Tt/Tp = 50\*22\*10-9 / (55\*5\*10-9) = 4 lần

* 1. Trình bày tóm tắt các đặc điểm trong kỹ thuật cài đặt RISC.

Bài làm

Đặc điểm 1: các bộ vi xử lý RISC có kích thước lệnh cố định hay còn gọi là định dạng chuẩn mã lệnh sẽ làm quá trình giải mã lện đơn giản hơn. Đói với bộ vi xử lý RISC kích thước của các lệnh là cố định 4 byte (32 bit). Với các lệnh không đòi hỏi đến 32 bit thì các byte còn điền lại được điền giá trị 0. Vì vậy CPU có thể giải mã các lệnh một cách nhanh chóng. Việc này cũng tương tự như việc xếp các lớp gạch có các kích thước gạch giống nhau và khác nhau. Tất nhiên nếu gạch có cùng kích thước thì việc sắp xếp sẽ hiệu quả hơn.

Đặc điểm 2: RISC sử dụng kiến trúc nạp/ lưu (load/ store). ở các bộ vi xử lý CISC, dữ liệu có thể được xử lý khi ở trong bộ nhớ.

Đặc điểm 3: có số lượng thanh ghi lớn. ccs thanh ghi đồng chất, do đó chúng có thể được sử dụng thay thế nhau trong mọi tình huống. tất cả các bộ vi xử lý RISC có 32 thanh ghi 32 bit R0 – R31.

Đặc điểm 4: cách đánh giá địa chỉ đơn giản. đẻ có những pương pháp đánh địa chỉ phức tạp cần kết hợp với các phép toán số học.

Đặc điểm 5; các bộ vi xử lý RISC có một tập lệnh nhỏ. Chúng chỉ có các lệnh cơ sở như ADD, SUB, AND, OR, EXOR, SHR, CALL và JMP. Không có các lệnh như INC, DEC, NOT, NEG, DAA, DAS,…Do RISC có rất ít lệnh nên công việc của người lập trình là phải thực hiện những lệnh này thông qua các lệnh khác của RISC.

Đặc điểm 6: hỗ trợ rất ít kiểu dữ liệu.

Đặc điểm 7: trên 95% các lênh đều được thực hiện chỉ với 1 nhịp đồng hồ.

Đặc điểm 8: bộ vi xử lý RISC có các BUS riêng cho dữ liệu và riêng cho mã.

* 1. Cho biết ý nghĩa của cụm từ RISC (Reduced Instruction Set Computer.

*Bài làm*

RISC (Reduced Instruction Set Computer): máy tính với tập lệnh đơn giản hóa. là một phương pháp thiết kế các bộ [vi xử lý](https://vi.wikipedia.org/wiki/Vi_x%E1%BB%AD_l%C3%BD) (VXL) theo hướng đơn giản hóa tập lệnh, trong đó thời gian thực thi tất cả các lệnh đều như nhau. Hiện nay các bộ vi xử lý RISC phổ biến là [ARM](https://vi.wikipedia.org/wiki/C%E1%BA%A5u_tr%C3%BAc_ARM), SuperH, [MIPS](https://vi.wikipedia.org/wiki/MIPS), [SPARC](https://vi.wikipedia.org/w/index.php?title=SPARC&action=edit&redlink=1), [DEC Alpha](https://vi.wikipedia.org/w/index.php?title=DEC_Alpha&action=edit&redlink=1), [PA-RISC](https://vi.wikipedia.org/w/index.php?title=PA-RISC&action=edit&redlink=1), [PIC](https://vi.wikipedia.org/wiki/Vi_%C4%91i%E1%BB%81u_khi%E1%BB%83n_PIC), và [PowerPC](https://vi.wikipedia.org/w/index.php?title=PowerPC&action=edit&redlink=1) của [IBM](https://vi.wikipedia.org/wiki/IBM).

* 1. Trình bày ngắn gọn sự khác biệt giữa kiến trúc CISC và kiến trúc RISC.

Bài làm

* Trong RISC kích thước tập lệnh là nhỏ trong khi ở CISC kích thước tập lệnh là lớn.
* RISC sử dụng định dạng cố định (32 bit) và chủ yếu là các hướng dẫn dựa trên đăng ký trong khi CISC sử dụng định dạng biến trong phạm vi từ 16-64 bit cho mỗi lệnh.
* RISC sử dụng đồng hồ đơn và chế độ địa chỉ giới hạn (ví dụ: 3-5). Mặt khác, CISC sử dụng nhiều chế độ địa chỉ 12 đến 24 đồng hồ.
* Số lượng các thanh ghi mục đích chung mà RISC sử dụng nằm trong khoảng từ 32-192. Ngược lại, kiến ​​trúc CISC sử dụng 8-24 GPR.
* Cơ chế bộ nhớ đăng ký để đăng ký được sử dụng trong RISC với các hướng dẫn LOAD và STORE độc lập. Ngược lại, CISC sử dụng bộ nhớ vào cơ chế bộ nhớ để thực hiện các hoạt động, hơn nữa, kết hợp các hướng dẫn LOAD và STORE.
* RISC đã phân chia dữ liệu và thiết kế bộ đệm hướng dẫn. Đối với, CISC sử dụng bộ đệm hợp nhất cho dữ liệu và hướng dẫn, mặc dù các thiết kế mới nhất cũng sử dụng bộ đệm tách.
* Hầu hết các điều khiển CPU trong RISC đều được gắn cứng mà không có bộ nhớ điều khiển. Ngược lại, CISC được mã hóa và sử dụng bộ nhớ điều khiển (ROM), nhưng CISC hiện đại cũng sử dụng điều khiển cứng.
* **KẾT LUẬN: Các hướng dẫn CISC rất phức tạp và có xu hướng chậm hơn RISC nhưng sử dụng ít chu kỳ hơn với ít hướng dẫn hơn.**

**Mục lục**

[Chương 1. Tổng quan về kiến trúc máy tính 1](#_Toc40858996)

[Chương 2. Kiến trúc tập lệnh 3](#_Toc40858997)

[Chương 3. CPU, Đường truyền và hệ thống vào/ra 14](#_Toc40858998)

[Chương 4. Kiến trúc hệ thống nhớ 25](#_Toc40858999)

[Chương 5. Kỹ thuật đường ống và RISC 44](#_Toc40859000)