**Chương 1. Tổng quan về kiến trúc máy tính**

**Câu 1.1.**

Phân biệt kiến trúc máy tính Von Neumann và Non Von Neumann.

Kiến trúc von Neumann gồm có: bộ nhớ chính, CPU, thiết bị vào ra.Trong CPU thì bao gồm đơn vị số học logic và đơn vị điều khiển chương trình.

Kiến trúc phi Von Neumann gồm có : thiết bị vào ra, đơn vị điều khiển và đơn vị số học logic

**Câu 1.2.**

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

Các thiết bị xuất

Bộ xử lý trung tâm

Các thiết bị nhập

Bộ nhớ chính

Chức năng:

-Các thiết bị vào: Nhận thông tin vào.

-Bộ nhớ chính: lưu trữ thông tin.

- Bộ xử lý trung tâm: Xử lý thông tin theo các dãy lệnh được nhớ sẵn bên trong.

- Các thiết bị xuất: Đưa thông tin ra.

**Câu 1.3.**

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.

Đặc điểm:  Cho biết số lượng ngăn nhớ tối đa được đánh địa chỉ

N bít thì có thể đánh địa chỉ tối đa cho 2N ngăn nhớ ( không gian địa chỉ bộ nhớ).

**Câu 1.4.**

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

Bus địa chỉ có độ rộng 24 bit có thể đánh địa chỉ cho 224bytes nhớ.

**Câu 1.5.**

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.

Đặc điểm: Độ rộng của bus dữ liệu nói chung sẽ xác định được lượng dữ liệu có thể truyền và trao đổi trên bus. Tốc độ truyền dữ liệu được tính bằng byte/s

**Câu 1.6.**

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

Bus dữ liệu có độ rộng là 32 bit sẽ có bang thông là 4Mbyte/s.

**Câu 1.7.**

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

Chức năng: vận chuyển các tín hiệu điều khiển

Đặc điểm: gồm 3 loại tín hiệu điều khiển

-Đọc ghi.

-ngắt.

- bus.

**Câu 1.8.**

Trình bày hiểu biết của em về tín hiệu điều khiển đọc/ghi bộ nhớ.

+Memory read: điều khiển đọc dữ liệu từ một ngăn nhớ có địa chỉ xác định lên bus dữ liệu.

+Memory write: Điều khiển ghi dữ liệu có sẵn trên bus dữ liệu đến một ngăn nhớ có địa chỉ xác định.

**Câu 1.9.**

Trình bày hiểu biết của em về tín hiệu điều khiển đọc/ghi thiết bị vào/ra.

+I/O Read: điều khiển đọc dữ liệu từ một cổng vào ra có địa chỉ xác định lên bus dữ liệu.

+I/O Write: Điều khiển ghi dữ liệu có sẵn trên bus dữ liệu ra một cổng có địa chỉ xác định

**Câu 1.10.**

Trình bày 3 tín hiệu điều khiển ngắt cứng điển hình (NMI, INTR, INTA).

-INTR (interrupt Request): 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.

-INTA (interrupt Acknowledge): Tín hiệu phát ra từ CPU báo cho bộ điều khiển vào ra biết CPU chấp nhận ngắt để trao đổi vào ra.

-NMI (Non Maskable interrupt): Tín hiệu ngắt không che được gửi đến ngắt CPU.

**Chương 2: Kiến trúc tập lệnh**

**Bài 2.1**: Đổi số 1097.75 về hệ 2.

Phần nguyên 1097=210+26+23+20=10001001001b

Phần thực: 0.75 về hệ 2.

0.75×2=0.5 dư 1.

0.5×2=0 dư 1.

🡺 0.75= 0.11 b.

Kết hợp lại ta có **1097.75= 10001001001.11b.**

**Câu 2.2.**

Đổi số 12035.125 về hệ 2.

Phần nguyên : 12035=213+211+210+29+28+21+20=10111100000011b.

Phần thực : 0.125 về hệ 2.

0.125×2=0.25 dư 0

0.25×2=0.5 dư 0

0.5×2=0 dư 1

🡺 0.125=0.001 b.

Kết hợp lại ta có **12035.125=10111100000011.001b**

**Câu 2.3.**

Đổi số 7899.8 về hệ 2

Phần nguyên 7899=212+211+210+29+27+26+24+23+21+20=1111011011011b.

Phần thực 0.8 về hệ 2

0.8×2=0.6 dư 1

0.6×2=0.2 dư 1

0.2×2=0.4 dư 0

0.4×2=0.8 dư 0

0.8×2=0.6 dư 1

….

Kết hợp lại ta có **7899.8=1111011011011.1101(…) b.**

**Câu 2.4.**

Đổi số 10110010.11B về hệ 10.

Đổi 10110010.11b= 1×27+0×26+1×25+1×24+0×23+0×22+1×21+0×20+1×2-1+1×2-2

=178.75.

Vậy với 10110010.11b= 178.75.

**Câu 2.5.**

Đổi số 1111011110110010.110001B về hệ 10.

**Cách 1:**

Đổi 1111011110110010.110001B=1×215+1×214+1×213+1×212+1×210+29+28+27+25+24+2+2-1+2-2+2-6=63410.76563.

**Cách 2:**

Đổi hệ 2 thành hệ 16

1111011110110010.110001B= F7B2.C4h

Đổi hệ 16 về hệ 10

F7B2.C4h=15×163+7×162+11×161+2×160+12×16-1+4×16-2=63410.76563.

**Câu 2.6.**

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

Đổi 234 hệ 10 về hệ 2 rồi đổi về hệ hex

234=27+26+25+23+21=11101010b= EAh.

**Câu 2.7.**

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

7899=212+211+210+29+27+26+24+23+21+20=1111011011011b= 1EDBh

**Câu 2.8.**

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

31634=214+213+212+211+29+28+27+24+21=111101110010010b= 7B92h

**Câu 2.9.**

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

16=24=000000000010000b

Số bù 1 của 16=1111111111101111b

Số bù 2 của 16=1111111111101111b

+ \_ 1

1111111111110000b

Vậy -16= 1111111111110000b= FFF0h.

**Câu 2.10.**

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

116=26+25+24+22=0000000001110100b

Số bù 1 của 116=1111111110001011b

Số bù 2 của 116=1111111110001011b

+ 1

= 1111111110001100b

Vậy -116=1111111110001100b= FF8Ch

**Câu 2.11.**

Biểu diễn số -67 ở dạng nhị phân 16 bit

67=26+2+1=0000000001000011b

Số bù 1 của 67=1111111110111100b

Số bù 2 của 67=1111111110111100b

+ 1

= 1111111110111101b

Vậy -67=1111111110111101b= FFBDh.

**Câu 2.12**.

Đổi -128 về dạng nhị phân.

128=27=0000000010000000b

Số bù 1 của 128=1111111101111111b

Số bù 2 của 128=1111111101111111b

+ 1

= 1111111110000000b

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

**Câu 2.13.**

Đổi -132 về dạng nhị phân.

132=27+22=0000000010000100b

Số bù 1 của 132=1111111101111011b

Số bù 2 của 132=1111111101111011b

+ 1

= 1111111101111100b.

Vậy -132b=1111111101111100b.

**Câu 2.14.**

Đổi -92 về dạng nhị phân.

92=26+25=0000000001100000b.

Số bù 1 của 92=1111111110011111b

Số bù 2 của 92=1111111110011111b

+ 1

= 1111111110100000b

Vậy -92=1111111110100000b=10100000b (Biểu diễn 8 bit, MSB=1)

**Câu 2.15.**

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.

-Với AX=1234h=0001 0010 0011 0100b (là số dương do SEM=0)

=212+29+25+24+22=4660.

* Với BX =900Ah=1001 0000 0000 1010b (là số âm do S=1)

Gọi B là giá trị chứa trong BX

Số bù 1 của B = 0110 1111 1111 0101b

Số bù 2 của B = 0110 1111 1111 0101b

+ 1

= 0110 1111 1111 0110b

= 6FF6h

=6\*16^3+15\*16^2+15\*16+6=28662=.

Vậy giá trị trong BX=B=28662.

* Với CX=FFFAh=1111 1111 1111 1010b (là số âm do S=1)

Gọi C là giá trị chứa trong CX

Số bù 1 của C = 0000 0000 0000 0101b

Số bù 2 của C= 0000 0000 0000 0110b

= 2^2+2=6=.

Vậy giá trị trong CX=C=6.

* Với DX=7FFFh=0111 1111 1111 1111b(là số dương do S=0)

= 32767.

**Câu 2.16.**

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

Ta có R= (-1)‑S×1.M×2E-bias

Bias= (28/2 )-1=127

Phần nguyên 1022=1111111110b

Phần thực 0.9=1110011100(...)b

Kết hợp lại ta có 1022.9=1111111110.1110011100(...)b

Chuẩn hóa 1111111110.1110011100(...)b=(-1)0×1.1111111101110011100(...)×29

S=0.

E-bias=9🡺E=bias+9=127+9=136=27+23=10001000b.

M=111 1111 1011 1001 1100 1110

Lắp khuôn X= 0100010001111111 1011 1001 1100 1110

=447FB9CEh

**Câu 2.17.**

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

Ta có công thức R= (-1)S ×1.M×2E-Bias

Phần nguyên -1012

1012=0000001111110100b

Số bù 1 cua 1022=1111110000001011b

Số bù 2 của 1022=1111110000001100b

🡺-1012=1111110000001100b

Phần thập phân 0,8=1100011000(...)b

Kết hợp -1022.8=1111110000001100.1100011000(...)b

Chuẩn hóa 1111110000001100.1100011000(...)b=(-1)1×1.1111100000011001100011000(...)×215

S=1b

E-bias=15🡺E=127+15=142=10001110b

M=11111000000110011000110b

🡺X=11000111011111000000110011000110b

=C77C0CC6H

**câu 2.18.**

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

Ts có công thức R= (-1)S×1.M×2E-bias

Phần nguyên 912=1110010000b

Phần thực .0625=0001b

Kết hợp lại ta có 916.0625=1110010000.0001b

Chuẩn hóa 1110010000.0001b=(-1)0×1.1100100000001×29

S=0b

E-bias=9🡺E=136=10001011b

M=11001000000010000000000b

🡺X=01000101111001000000010000000000b

=45E40400h.

**Câu 2.19.**

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

Ta có công thức R=(-1)S ×1.M×2E-bias

Đổi 447FB999h=01000100011111111011100110011001b

S=0b

E=10001000b=27+23=136🡺E-bias=136-127=9.

M=11111111011100110011001b

Từ đó ta có R=(-1)S×1. 11111111011100110011001×29

Vậy ta có

Phần nguyên: 1111111110b=1022b

Phần thực=0.11100110011001b=0.E664h=0.8999633789

Kết hợp lại ta có 447FB999h=1022.8999633789.

**Câu 2.20.**

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

409CCCCCh=0100 0000 1001 1100 1100 1100 1100 1100b

S=0b

E=1000 0001b=129🡺E-bias=2.

M=001 1100 1100 1100 1100 1100b

R=(-1)0×1. 001 1100 1100 1100 1100 1100×22

Phần nguyên 100b=4

Phần thập phân 0. 1 1100 1100 1100 1100 1100b=0.E66660h=0.8999996185.

Vậy 409CCCCCh=4.8999996185.

**Câu 2.21.**

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

BF400000h=10111111010000000000000000000000b

S=1b

E=01111110b =126🡺E-bias=-1.

M=10000000000000000000000b

R=(-1)-1 ×1. 10000000000000000000000 ×2-1

Phần nguyên =0

Phần thập phân=-0.110000000000000000000000=-0.75

Vậy BF400000h=-0.75.

**Câu 21.2**

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.

S12.75= 53 31 32 2E 37 35

**Bài 2.23:**

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

41 74 74 61 63 6B 20 61 77 6E=Attack awn

**Bai 2.24 :**

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

+32.

**Bai 2.25:**

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.

-127.

**Câu 2.26.**

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

Số bù 1 của 01111111b =10000000b

Số bù 2 của 01111111b =10000001b

Vậy ta có thể viết lại như sau

00001110b+10000001b=10001111b

**Bài 2.27**.

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

01111011B

+ 11001111B

10111111B

00111101B

01111111B

01111110B

= 1101000011B

**Câu 2.28.**

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

11101101

11011110

10111111

10101101

01111101

= 1110110100B.

**Câu 2.29.**

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

11101111b

11111101b

11110111

00000000

11110111

11110111

11110111

11110111

11110111

11110111

1110110001110011B

Vậy = 1110110001110011B

**Bài 2.30.**

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 (xem thanh ghi số thực dưới đây). Trong một số bộ CPU hiện nay và trước đây, có một thanh ghi đặc biệt là thanh ghi tích lũy tích lũy, được dùng cho nhiều tính toán.

- Thanh ghi địa chỉ chứa địa chỉ bộ nhớ và được dùng để truy cập bộ nhớ. Trong một số CPU, có một thanh ghi địa chỉ đặc biệt thanh ghi chỉ mục, 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 đa năng (GPRs) có thể chứa cả dữ liệu lẫn địa chỉ., là kết hợp giữa thanh ghi dữ liệu và thanh ghi địa chỉ

- Thanh ghi dấu phẩy động (FPRs) dùng để chứa các kiểu số

- Thanh ghi hằng số chứa các giá trị đọc được (Ví dụ: 0, 1, pi,..

- Thanh ghi định hướng chứa dữ liệu cho việc xử lý định hướng đối tượng, thực hiện bởi lệnh SIMD (lệnh đơn, bội dữ kiện)

- Thanh ghi chuyên biệt chứa trạng thái chương trình; thường bao gồm con trỏ chỉ lệnh, con trỏ ngăn xếp, và thanh ghi trạng thái.

- Thanh ghi chỉ lệnh chứa lệnh tạm huỷ

- Thanh ghi chỉ mục dùng cho sửa đổi địa chỉ tác tử trong quá trình thực hiện 1 chương trình

-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

+ Thanh ghi ổ cứng cũng tương tự nhưng ở ngoài CPU

**Bài 2.31.**

Nêu tên và tóm tắt chức năng các cờ trong thanh ghi cờ của CPU 8086:

- Thanh ghi Cờ (F) dài 16 bit, mỗi bit là một cờ. Mỗi cờ có giá trị 1 (gọi là SET –Đặt) hoặc 0 (gọi là CLEAR – Xóa).

- Nhóm cờ điều khiển bao gồm các cờ dùng để điều khiển sự hoạt động của CPU và giá trị của cờ được thiết lập bằng các lệnh phần mềm.

- Nhóm cờ trạng thái bao gồm các cờ phản ánh kết quả thực hiện lệnh cũng như trạng thái của CPU.

**Bài 2.33.**

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

11110001B

01000010b

100110011b

Ta thấy CF =0

C6,7 = 1

Vậy OF = CF C6,7 = 0 1 = 1

**Bài 2.34.**

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

10110001B

11000010b

101110011b

Ta thấy CF=1

C6,7=1

Vậy OF = CF C6,7 = 1 1 =2.

**Chương 3 CPU, đường truyền hệ thống vào ra.**

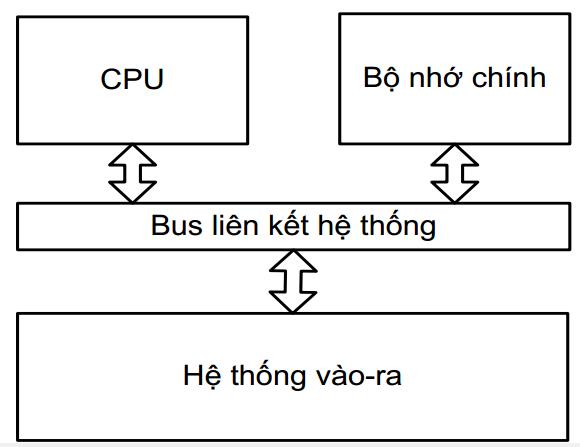
**Câu 3.1:** Cho biết ý nghĩa khi nói Bus địa chỉ có độ rộng 32 bit.

 Bus gồm 32 đường dây dẫn, CPU có khả năng quản lý không gian nhớ là 2 mũ 32 = 42MB

**Câu 3.2**: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.

**Câu 3.3:**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.

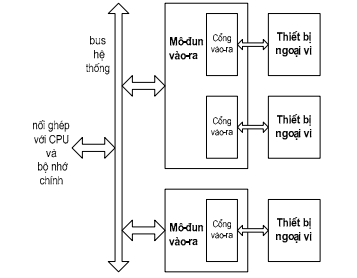
CPU:Đơn vị điều khiển trung tâm

Bộ nhớ chinh:Lưu trữ thông tin

Bus:truyền đường truyền dữ liệu và kết nối hệ thống vào ra với khối trung tâm

Hệ thống vào ra

**Câu 3.4:** 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 từng thành phần :
  + CPU và bộ nhớ chính : điều khiển hệ thống
  + Bus hệ thống :

Vận chuyển lệnh từ bộ nhớ đến CPU

Vận chuyển dữ liêu giữa CPU , module nhớ , module vào/ra vớ nhau .

* + Modul vào/ra :

Điều khiển và đinh thời

Trao đổi thông tin với CPU

Trao đổi thông tin với thiết bị ngoại vi

Đệm giữa bên trong máy với thiết bị ngoại vi

Phát tín hiệu lỗi của thiết bị ngoại vi

**Câu 3.5:** 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.

**Câu 3.6:**Trình bày 2 phương pháp vào ra dữ liệu do CPU chủ động.

 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.

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.

**Câu 3.7:**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.

**Câu 3.8:**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.

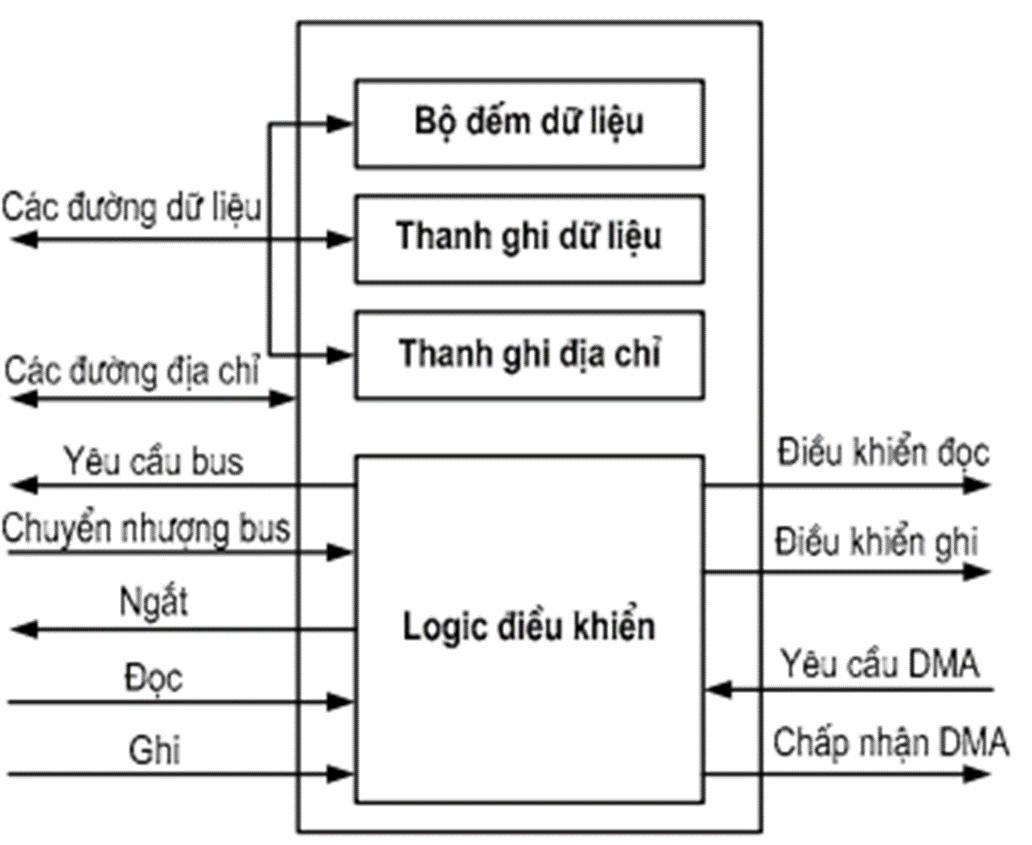
-   Kết thúc, CPU lấy lại trạng thái cũ của CPU và tiến trình A vừa bị ngắt để tiếp tục thi hành.

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

Câu 3.9: Trình bày cấu trúc của hệ thống vào/ra dữ liệu theo kiểu DMA.

-Sơ đồ cấu trúc



Thanh ghi dữ liệu: chứa dữ liệu trao đổi

Thanh ghi địa chỉ: chứa địa chỉ ngăn nhớ dữ liệu

Bộ đếm dữ liệu: chứa số từ dữ liệu cần trao đổi

Logic điều khiển: điều khiển hoạt động của DMAC

**Câu 3.10:** Trình bày quá trình vào/ra dữ liệu kiểu DMA (quá trình DMA).

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.

**Câu 3.11**: 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

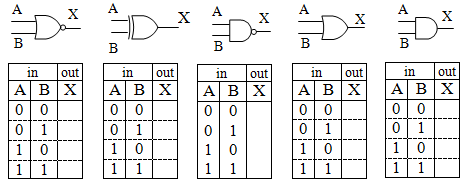
-         Ưu tiên vòng: Kênh nào vừa phục vụ sẽ có mức ưu tiên thấp nhất, lúc mới thiết lập kênh 0 sẽ có mức ưu tiên cao nhất.

**Câu 3.12**: Nhận biết các mạch cổng sau (ghi rõ tên mạch cổng ở bên dưới hình)



XOR, NOT,NOR,AND,OR,NAND.

Câu 3.13:Cho các mạch cổng 2 đầu vào, ghi tên cổng, hoàn thiện các bảng chân lý tương ứng



AND

|  |  |  |
| --- | --- | --- |
| A | B | X |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |
|  |  |  |

NAND

|  |  |  |
| --- | --- | --- |
| A | B | X |
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

OR

|  |  |  |
| --- | --- | --- |
| A | B | X |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

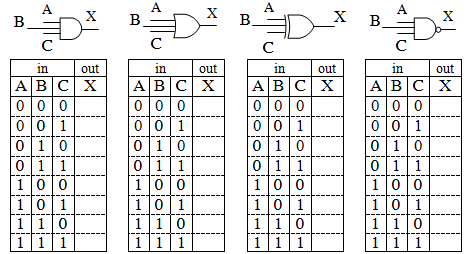
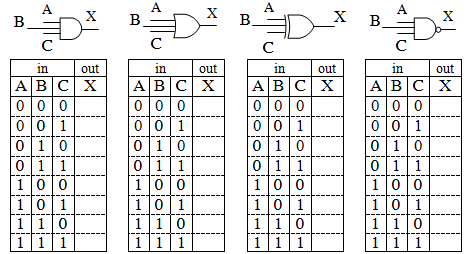
XOR

|  |  |  |
| --- | --- | --- |
| A | B | X |
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

NOR

|  |  |  |
| --- | --- | --- |
| A | B | X |
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |
|  |  |  |

Câu 3.14:Cho các mạch cổng đầu vào , ghi tên cổng , hoàn thiện các bảng chân lý tương ứng .

* ****Mạch 1 :

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

* Mạch 2 :

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

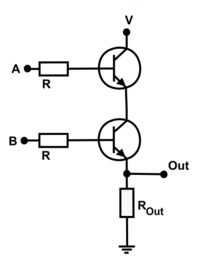
* Mạch 3 :

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

* Mạch 4

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

Câu 3.15: Vẽ sơ đồ mạch điện của cổng AND 3 đầu vào từ các transistor



Câu 3.16.

Cho các mạch cổng 4 đầu váo sau , ghi tên cổng , hoàn thiện các bảng chân lý tương ứng

* Mạch 1 :

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

* Mạch 2 :

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

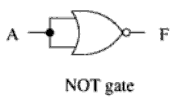
* Mạch 3 :

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

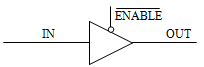
Câu 3.18.

Vẽ mạch điện có chức năng ứng với mạch NOT.

* Sơ đồ mạch điện của mạch NOT :



Câu 3.19: Cho biết tên và nguyên lý làm việc của mạch logic sau:



* Hoạt động:
  + Khi chân ENABLE có mức logic cao (0) thì đầu ra (OUT) bằng đầu vào (IN).
  + Khi chân ENABLE có mức logic thấp (1) thì đầu ra và đầu vào cách ly (đầu ra có mức trở kháng cao).

ENABLE

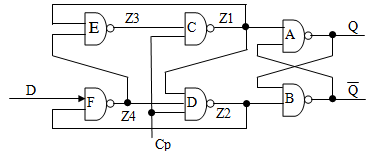
IN

Câu 3.20: Cho biết tên và nguyên lý làm việc của mạch logic sau:

* Hoạt động:
  + Khi chân ENABLE có mức logic cao (1) thì đầu ra (OUT) bằng đầu vào (IN).
  + Khi chân ENABLE có mức logic thấp (0) thì đầu ra và đầu vào cách ly (đầu ra có mức trở kháng cao).

Câu 3.21:

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



Q không đổi có nghĩa là: Q(t) = Q(t-1). Vì Q mang 2 giá trị là 0 hoặc là 1. Nếu Q(t-1) = 0 àQ(t) = 0 (1)

và Nếu Q(t-1)=1 àQ(t) = 1 (2)

Chứng minh:

Giả sử Q(t-1) = 0 à/Q(t-1) = 1. (/ chính là phủ định)

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) = 1à/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.

Câu 3.22.

Thiết kế mạch logic để phát hiện lỗi trong mã BCD . Lỗi vào mà mã BCD , lối ra oẳ trạng thái lỗi .

* Mạch logic phát hiện lỗi trong mã BCD :

B2

B1

C

B3

T

Câu 3.23.

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

Câu 3.24

Chứng minh rằng mạch sau là cổng NOT



* Bảng chân lý cổng NOT :

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

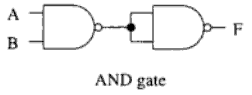
* Chứng minh :

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

Câu 3.25.

Hãy xây dựng cổng AND 2 đầu vào từ các cổng NAND 2 đầu vào

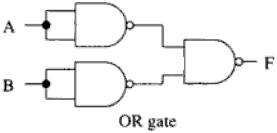
* Cổng AND 2 đầu vào :



C

Câu 3.26.

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



* Bảng chân lý cổng OR

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

* Chứng minh

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| IN | | | | OUT |
| A | B | X | Y | F |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 |

Câu 3.27.

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

- Ý 1: Đáp án: b) 0 1.

- Ý 2: Đáp án: a)

- Ý 3:

Câu 3.28.

Làm sao để xây dựng 1 cổng AND có 2 lối vào từ những cổng NOR 2 lối vào .

* Mạch cổng AND :

X

A

B

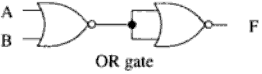
Y

Z

K

Câu 3.29.

Chứng minh mạch sau là cổng OR 2 đầu vào



* Bảng chân lý mạch OR:

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

* Chứng minh :

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

Câu 3.30.

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 .

* Bảng chân lý mạch lẻ 3 đầu vào

|  |  |  |  |
| --- | --- | --- | --- |
| A | B | C | F |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 |

* Chứng minh

F = XOR( C ; K )

K = XOR( A ; B )

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A | B | C | K | F |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 0 | 1 |

Câu 3.31.

Chứng minh rằng mạch sau là cổng NOT .



* Bảng chân lý NOT :

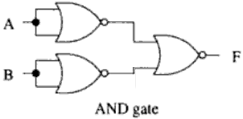
|  |  |
| --- | --- |
| IN | OUT |
| A | B |
| 0 | 1 |
| 1 | 0 |

* Chứng minh :

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

Câu 3.32.

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



* Bảng chân lý AND :

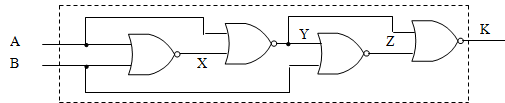
|  |  |  |
| --- | --- | --- |
| IN | | OUT |
| A | B | F |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

* Chứng minh :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| IN | | | | OUT |
| A | B | X | Y | F |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 |

Câu 3.33.

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



* Bảng chân lý mạch AND :

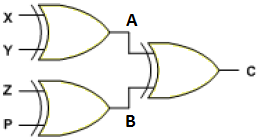
|  |  |  |
| --- | --- | --- |
| IN | | OUT |
| A | B | F |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

* Chứng minh :

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

Câu 3.34.

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



* Bảng chân lý 4 đầu vào :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **IN** | | | | **OUT** |
| **X** | **Y** | **Z** | **P** | **C** |
| **0** | **0** | **0** | **0** | **0** |
| **1** | **0** | **0** | **0** | **1** |
| **0** | **1** | **0** | **0** | **1** |
| **0** | **0** | **1** | **0** | **1** |
| **0** | **0** | **0** | **1** | **1** |
| **1** | **1** | **0** | **0** | **0** |
| **1** | **0** | **1** | **0** | **0** |
| **1** | **0** | **0** | **1** | **0** |
| **0** | **1** | **1** | **0** | **0** |
| **0** | **1** | **0** | **1** | **0** |
| **0** | **0** | **1** | **1** | **0** |
| **1** | **1** | **1** | **0** | **1** |
| **1** | **1** | **0** | **1** | **1** |
| **1** | **0** | **1** | **1** | **1** |
| **0** | **1** | **1** | **1** | **1** |
| **1** | **1** | **1** | **1** | **0** |

* Chứng minh :

A = XOR ( X ; Y )

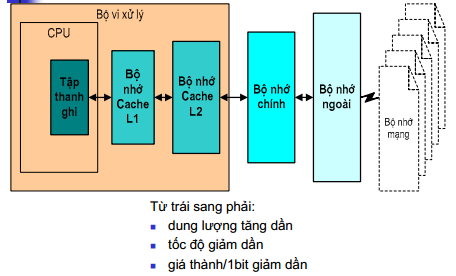
B = XOR ( Z ; P )

C = XOR ( A ; B )

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **IN** | | | | | | **OUT** |
| **X** | **Y** | **Z** | **P** | **A** | **B** | **C** |
| **0** | **0** | **0** | **0** | **0** | **0** | **0** |
| **1** | **0** | **0** | **0** | **1** | **0** | **1** |
| **0** | **1** | **0** | **0** | **1** | **0** | **1** |
| **0** | **0** | **1** | **0** | **0** | **1** | **1** |
| **0** | **0** | **0** | **1** | **0** | **1** | **1** |
| **1** | **1** | **0** | **0** | **0** | **0** | **0** |
| **1** | **0** | **1** | **0** | **1** | **1** | **0** |
| **1** | **0** | **0** | **1** | **1** | **1** | **0** |
| **0** | **1** | **1** | **0** | **1** | **1** | **0** |
| **0** | **1** | **0** | **1** | **1** | **1** | **0** |
| **0** | **0** | **1** | **1** | **0** | **0** | **0** |
| **1** | **1** | **1** | **0** | **0** | **1** | **1** |
| **1** | **1** | **0** | **1** | **0** | **1** | **1** |
| **1** | **0** | **1** | **1** | **1** | **0** | **1** |
| **0** | **1** | **1** | **1** | **1** | **0** | **1** |
| **1** | **1** | **1** | **1** | **0** | **0** | **0** |

**Chương 4. Kiến trúc hệ thống nhớ.**

Câu 4.1:



Câu 4.2: 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.

Câu 4.3: Trình bày nguyên tắc hoạt động của bộ nhớ cache (trao đổi dữ liệu với bộ nhớ và với CPU như thế nào?).

* 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) – Hit cache
* Nếu không có, đọc Block nhớ chứa dữ  
  liệu từ bộ nhớ chính vào cache – Miss cache

-Tiếp đó chuyển dữ liệu từ cache vào  
CPU

Câu 4.4: Trình bày phương pháp chung để tổ chức bộ nhớ cache.

Phương pháp chung để tổ chức bộ nhớ cache :

* + - * Bộ nhớ chính có 2N byte nhớ
      * Bộ nhớ chính và cache được chia thành các khối có kích thước bằng nhau .

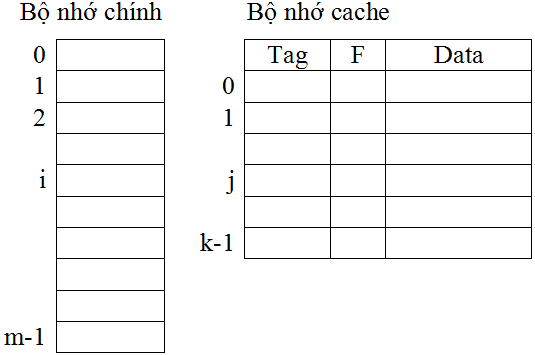
Bộ nhớ : B0 , B1 , B2 ... Bp-1 ( p Blocks )

Bộ nhớ cache : L0 , L1, L2 ... Lm-1 ( m Lines )

* + - * Kích thước của Block = 8,16,32,64,128 byte
      * Một số Block của bộ nhớ chính được nạp vào các Line của Cache
      * Nội dung Tag cho biết Blocl nào của bộ nhớ chính chưa ở Line nào đó
      * Bit F báo hiệu nội dung của Line có thay đổi hay không . F=0 không thay đổi , F=1 thay đổi

Câu 4.5.

Trình bày và vẽ sơ đồ thao tác đọc bộ nhớ cache với mỗi khối chứa một byte dữ liệu .

****

* Khi CPU truy nhập ( đọc/ghi ) một từ nhớ có 2 khả năng xảy ra :
  + - * Từ nhớ có trong cache ( hit cache )
      * Từ nhớ không có trong cache ( miss cache )

Câu 4.7.

Trình bày thao tác ghi bộ nhớ với mỗi khối chứa một byte dữ liệu .

* + - * Bộ nhớ chính có 2N byte nhớ
      * Bộ nhớ chính và cache được chia thành các khối có kích thước bằng nhau .

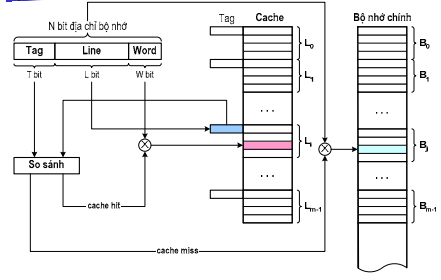
Bộ nhớ : B0 , B1 , B2 ... Bp-1 ( p Blocks )

Bộ nhớ cache : L0 , L1, L2 ... Lm-1 ( m Lines )

* + - * Kích thước của Block = 8,16,32,64,128 byte
      * Một số Block của bộ nhớ chính được nạp vào các Line của Cache
      * Nội dung Tag cho biết Blocl nào của bộ nhớ chính chưa ở Line nào đó
      * Bit F báo hiệu nội dung của Line có thay đổi hay không . F=0 không thay đổi , F=1 thay đổi
* Khi CPU truy nhập ( đọc/ghi ) một từ nhớ có 2 khả năng xảy ra :
  + - * Từ nhớ có trong cache ( hit cache )
      * Từ nhớ không có trong cache ( miss cache )

Câu 4.7.

Trình bày phương pháp ánh xạ trực tiếp trong bộ nhớ cache .

****

* Phương pháp ánh xạ trực tiếp :
  + - * Mỗi block của bộ nhớ chính chỉ có thể được nạp vào một Line của cache .
      * Mỗi một địa chỉ N bit của bộ nhớ chính gồm ba trường :

Trường Word gồm W bit xác định một từ nhớ trong Block hay Line :

2W = kích thước của Block hay Line

Trường Line gồm L bit xác định một trong số các Line trong Cache :

2L = số Line trong cache = m🡪 dung lượng cache : 2L+M

Trường Tag gồm T bit xác định block nhớ cần truy cập

T = N – ( W + L )

* Bộ so sánh đơn giản
* Xác suất cache hit thấp

Câu 4.8.

Trình bày thao tác đọc cache theo phương pháp ánh xạ trực tiếp trong tổ chức bộ nhớ cache .

**Câu 4.9.**

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.

-Xác định dung lượng của một block(line) nhớ

Cblock=Cmemmory/nblock=256KB/512block=28×210/29=29(byte)

🡺word (n1)=9 bits

-Xác định số line của cache

nline =Ccache/Cblock=8KB/29=23×210/29=24(lines)

🡺L(n2)=4 bits

-Xác định số bits để đánh số hiệu của block trong bộ nhớ

Nblock=Số block=512=29

T(n3)=9(bits)

-Xác định số bits địa chỉ hợp lệ

N=W+L+T=9+4+9=22 bits.

Đọc bộ nhớ địa chỉ B7281Ah=101101110010100000011010b

Vì 24 bits >22 bits suy ra địa chỉ không hợp lệ.

**câu 4.10.**

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.

Xác định dung lượng của một block nhớ

Cblock=Cmemmory/nblock=29(byte)

🡺W(n1)=9bits.

Xác định số line của cache

nline=Ccache/Cblock=24(line)

🡺L(n2)=4 bits

Xác định số bits để đánh số hiệu block trong bộ nhớ

Nblock=Số block=512=29

🡺T(n3)=9 bits

N=W+L+T=22 bits

Đọc bộ nhớ địa chỉ 2F1025h=1011110001000000100101

* Địa chỉ hợp lệ vì số bits là 22 bits.

L=1000b=8

🡺truy cập line 8 trong cache

Xác định số hiệu bộ nhớ của block cần truy cập

BI=101111000=178h=376

* Block nhớ cần truy cập là 376.

Block 376 phải nạp vào line 376 mode 16=8

🡺block nạp đúng line🡺địa chỉ hợp lệ.

Khả năng 1 Miss cache Tag **≠376**

Khi đó CPU truy cập vào bộ nhớ sẽ không tìm thấy trường tag 376. Khi đó CPU sẽ yêu cầu một địa chỉ mới có trường tag là 376.

Khả năng 2 Hit cache Tag=376 CPU đọc byte trong line 8 có địa chỉ tương đối là 9 bits thấp của địa chỉ W=000100101b=25+22+20=37.

Địa chỉ vật lý= TW(n3n1)= 10/1111/0000/0010/0101

= 2F025h.

**Câu 4.11.**

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.

-Xác định dung lượng của block

Cblock=Cmemmory/nblock=29

🡺w(n1)=9 bits

Xác định số line trong cache

nline=Ccache/Cblock=24.

🡺L(n2)=4 bits.

Xác định số bít để đánh số hiệu block

Nblock= Số block=29

🡺T(n3)=9 bits

Xác định số địa chỉ hợp lệ

N=W+L+T=22 bits

Đọc bộ nhớ địa chỉ 2F0825h=1011110000100000100101b

Địa chỉ hợp lệ vì <=22bits

Xác định số line L=0100b=4.

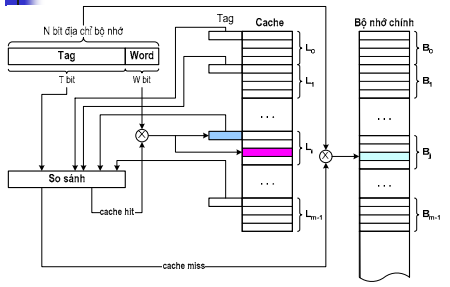
BI=101111000b=376

Block phải nạp vào line 376 mode 16=8 ≠4

🡺Block không nạp đúng line🡺 địa chỉ không hợp lệ

Câu 4.12

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

****

-Mỗi block có thể được nạp vào line bất kỳ của cache

-Địa chỉ của bộ nhớ chính bao gồm 2 trường :

-Trường word

-Trường Tag dùng để xác địng block của bộ nhớ chính .

-Tag xác định Block của bộ nhớ chính

Câu 4.13.

Trình bày thao tác đọc cache theo phương pháp ánh xạ liên kết hoàn toàn trong bộ nhớ cache .

* Bước 1: Xác định dung lượng của 1 block (line) nhớ
* Bước 2: Xác định số bít cần thiết để đánh số hiệu block trong bộ nhớ.
* Bước 3: Xác định số bít địa chỉ hợp lệ cho bộ nhớ
* Bước 4: Đọc cache với địa chỉ mà CPU phát

**Câu 4.14.**

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 3280Ah. Hãy trình bày chi tiết 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.

Xác định dung lượng cua block nhớ

Cblock=Cmemmory/nblock=29byte

🡺W(n1)=9bits

Xác định số bits cần thiết để đánh số hiệu block trong bộ nhớ

Nblock=29

* T(n3)=9 bits

Số bits hợp lệ cho địa chỉ bộ nhớ 256KB là= 9+9=18 bits

Đọc bộ nhớ địa chỉ 3280Ah=110010100000001010b

Địa chỉ hợp lệ vì 18 bits <=18 bits

Số hiệu block bộ nhớ cần truy cập 110010100b=404

Trường hợp 1 Miss cache Tag≠404

CPU phải nạp vào block có số hiệu 404 từ bộ nhớ chính vào cache. CPU tìm line nào trong cache sử dụng kém hiệu quả nhất để thay bằng block mới. Nếu bits F của line bị thay thế =1 thì nội dung của line đó được ghi trả lại bộ nhớ có giá trị đúng bằng trường Tag của line đó. Sau đó xóa F=0 rồi nạp block nhớ có số hiệu 404 vào line này và cập nhật trường Tag của nó bằng 404. Sau đó CPU sẽ đọc byte cần trong line này.

Không có tag nào trong line có địa chỉ là 404🡺 CPU phải nạp 1 block có số hiệu là 404 vào bộ nhớ chính và cache🡺 CPU đọc cache và bộ nhớ chính 1 lần nữa 🡺 đọc dữ liệu ở trong cache.

Trường hợp 2 Hit cache Tag=404

CPU sẽ đọc byte có số hiệu là 9 bits thấp của địa chỉ 000001010b=10🡺 đọc byte có số hiệu 10 của line đó.

Địa chỉ vật lý của ô nhớ có số hiệu cần truy cập là =TW(n3n1)= 110010100000001010b=3280Ah.

**Câu 4.15.**

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 7280Ah. Hãy trình bày chi tiết 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.

Xác định dung lượng bộ nhớ block

Cblock=Cmemmory/nblock=29 byte

🡺W(n1)=9 bits

Số line trong cache là

Nline=Ccache/Cblock=24(lines)

Xác định số bít để đánh số hiệu block trong bộ nhớ.

Nblock=số block=29

🡺T(n3)=9 bits

Số bits hợp lệ cho địa chỉ bộ nhớ 256KB là 9+9=18 bits

Đọc bộ nhớ địa chỉ 7280Ah=1110010100000001010b

Địa chỉ bộ nhớ không hợp lệ vì 19>18 bits

Tiến trình phải dừng.

\*\* Ánh xạ liên kết tập hợp

Địa chỉ CPU 73426h

Dung lượng của 1 block nhớ là Cblock=2^9🡺W=9bits

Số line trong cache 2^4 L=4bits

Số bits để đánh số hiệu block trong bộ nhớ 2^9🡺 T=9 bits

Số line trong 1 set Nline-set=nline/nset=16/4=4 lines

Xác định số bits để đánh số hiệu của set

Nset=4=22🡺S=2.

SI=10b=2

Bi=230

…

Th1 không có trường tag nào có số hiệu 230 🡺 CPU phải nạp vào block nhớ có số hiệu 230 vào 1 line nào đó của set 2

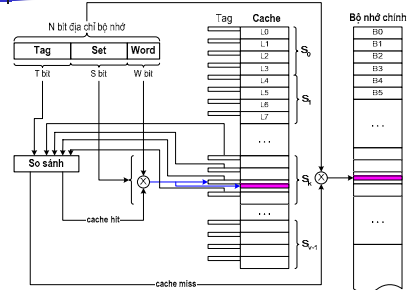
TH2

1 tag có số hiệu 230 🡺 CPU đọc ô nhớ đó có số hiệu =

Địa chỉ vật lý T+W

Câu 4.16.

Trình bày phương pháp ánh xạ liên kết tập hợp trong tổ chức bộ nhớ cache .

****

* Cache được chia thành các tập (set)
* Mỗi một set chứa một số line
* Ánh xạ theo nguyên tắc :

B0🡪S0

B1🡪S1

B2🡪S2

Bm🡪Sm mod q

Câu 4.17.

**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.

- (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 q), thì địa chỉ này không hợp lệ, tiến trình phải dừng (Số lượng Set trong cache là q).

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

**Bước 2:** Đọc cache

**Câu 4.18**.

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 theo trường hợp phát ra địa chỉ trên.

Dung lượng của 1 block nhớ là

Cblock=Cmemory/nblock=29 bytes

🡺W= 9 bits

Số line của cạche

Nline=Ccache/Cblock=2^4 lines

🡺L=4 bits

Số line trong 1 set

Nline-set=nline/nset=16/4=4 line

Số bít để đánh số hiệu của set

Nset=4=2^2🡺S=2 bits

Xác định số bits để đánh số hiệu block

Nblock=2^9 line

🡺T= 9 bits

Số bit tối đa để đánh địa chỉ hợp lệ là

N=T+W+S=20 bits

Địa chỉ CPU phát ra được lệnh truy cập bộ nhớ là

22A07h= 10 0010 1010 0000 0111b

Ta thấy địa chỉ CPU phát ra được lệnh truy cập bộ nhớ là 18 bits<= 20 bits tối đa của địa chỉ hợp lệ🡺 thỏa mãn điều kiện 1.

SI=01b=1

BI=10001010b=7Ah=138.

Kiểm tra block có số hiệu 138 nap vào set 1 của cache có hợp lệ hay không

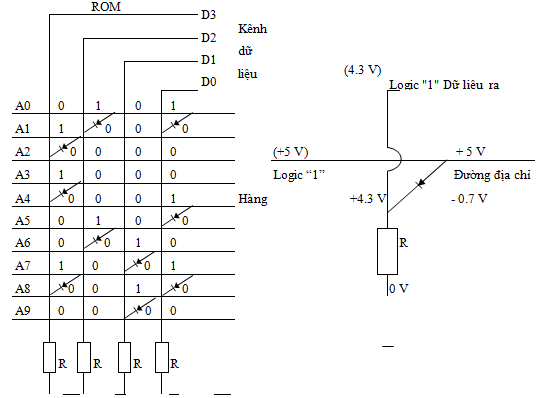
Ta có 138 mode 4=2 khác set =1 vậy block nạp không đúng set 🡺địa chỉ không hợp lệ

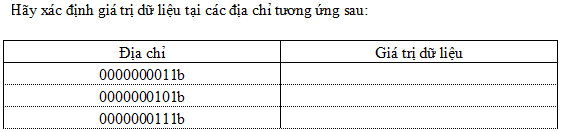
🡺Tiến trình buộc phải dừng.

**Câu 4.19.**

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 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 theo trường hợp phát ra địa chỉ trên.

Câu 4.21:Cho sơ đồ mạch ROM mặt nạ sau:





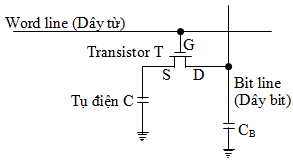
***Bài làm:***

**-**

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

Câu 4.22:

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

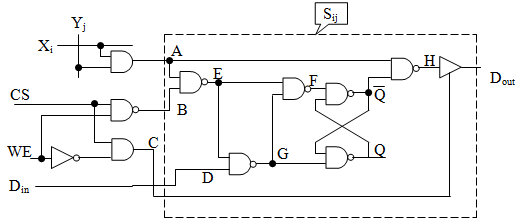


Bài làm

- Phần tử trên là DRAM, nguyên lý hoạt động là:

* Một tế bào DRAM lưu trữ bao gồm một tụ điện và một thiết bị chuyển (MOSFET) hoạt động như một chuyển đổi (hình minh họa).
* Nếu trong tụ điện có điện tích thì tế bào mang giá trị bit là “1” còn ngược lại giá trị đó sẽ là “0”.
* Các tế bào được sắp xếp theo các hàng và cột trực giao với nhau tạo nên các mảng.
* Thông thường người ta hay dùng các mảng phụ ghép thành 1 mảng chính thay vì chỉ dùng 1 mảng để giảm thời gian truy cập tế bào.

Câu 4.23: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.

**Câu 4.25.**

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

<2,43h>

S=2<5🡺 Hợp lệ

Dựa vào mục 2 của bảng SCB ta có

P=1🡺 Đoạn mã đã được nạp vào RAM

A=300h🡺 Đoạn 2 nạp vào RAM tại vị trí bắt đầu 300h

L=700h🡺 Kích thước của đoạn là 700h byte

D=43h<700h 🡺 hợp lệ

🡺Địa chỉ vật lý cần truy cập là PA=A+d=300h+43h=343h.

<3,4FFh>

S=3<5🡺 hợp lệ

Dựa vào mục 3 của bảng SCB có

P=0🡺 Đoạn 3 chưa được nạp vào RAM

🡺hệ thống cần thực hiện nạp đoạn hoặc đổi đoạn để tiến trình được thực thi.

<4,600h>

S=4<5🡺 hợp lệ

Dựa vào mục 4 của bảng SCB ta có

P=1🡺 hợp lệ

A=A00h🡺 đoạn 4 nạp vào RAM tại vị trí bắt đầu A00h

L=600h🡺 Kích thước của đoạn 4 là 600h

D=600h<=600h🡺 hợp lệ

🡺địa chỉ vật lý cần truy cập là PA=A+d=A00h+600h=1000h

<5,100h>

Ta thấy S=5 không nhỏ hơn 5🡺 không hợp lệ

Tiến trình buộc dừng.

**Câu 4.26.**

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:

|  |  |
| --- | --- |
| P | A |
| 1 | 7 |
| 0 | ----- |
| 1 | 3 |
| 1 | 5 |
| 0 | ---- |

Địa chỉ vật lý truy nhập tương ứng với các địa chỉ logic sau<0,2> <2,3> <1,3> <4,1>,<2,4><3,5><5,1>

Kích thước 1 trang vật lý L=32/8=4 byte🡺n1=2

Np=0<5 hợp lệ.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Stt | Bộ nhớ chương trình | | | |
| 0 | 0 | 1 | 2 | 3 |
| 1 | 4 | 5 | 6 | 7 |
| 2 | 8 | 9 | A | B |
| 3 | C | D | E | F |
| 4 | G | H |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Stt | Bộ nhớ vật lý | | | |
| 0 |  |  |  |  |
| 1 |  |  |  |  |
| 2 |  |  |  |  |
| 3 |  |  |  |  |
| 4 |  |  |  |  |
| 5 |  |  |  |  |
| 6 |  |  |  |  |
| 7 |  |  |  |  |

Từ mục 0 của bảng PCB có

P=1🡺 trang logic 0 đã được nạp vào RAM

A=7<8( số trang vật lý) 🡺 trang logic 0 nạp vào frame

D=2<4 ( kích thước 1 trang vật lý)🡺 Thỏa mãn 🡺 địa chỉ logic hợp lệ

Địa chỉ vật lý cần truy nhập là

PA=A\*L+d=7\*4+2=30.

0123/4567/89AB/CDEF/GH.

Nội dung của ô nhớ 30 là khí tự “2”.

ASCII kí tự 2 = 32h=110010b

<2,3>

Kích thước của 1 trang=32/8=4 byte=22

🡺n1=2.

Np=2<5 thỏa mãn.

Từ mục 2 của bảng PCB ta có

P=1🡺 trang logic 2 đã nạp vào RAM

A=3<8 số trang vật lý🡺 trang logic 2 nạp vào frame.

D=3<4 kích thước của 1 trang vật lý🡺 địa chỉ logic hợp lệ.

Địa chỉ vật lý cần truy cập là

PA=A\*L+d=3\*4+3=15.

Nội dung ô nhớ 15 là kí tự B

ASCII vị trí của kí tự B là 42h=1000010b

<1,3>

Kích thước 1 trang vật lý L=32/8=4 byte=22

🡺n1=2.

Np=1<5 hợp lệ

Từ mục 1 của bảng PCB ta có

P=0 🡺 trang logic 1 chưa nạp vào RAM.

🡺Để tiếp tục tiến trình cần thay đổi số trang logic sao cho P=1.Nếu không thay đổi được 🡺 tiến trình dừng.

<4,1>

Kích thước 1 trang vật lý là 32/8=4 byte=22🡺n1=2.

Np=4<5🡺 hợp lệ

Từ mục 4 của bảng PCB ta có

P=0🡺 trang logic 4 chưa nạp vào RAM.

* Để tiếp tục tiến trình thì phải thay đổi trang logic sao cho P=1. Nếu không thay đổi dduocj thì buộc dừng tiến trình.

<2,4>

Kích thước 1 trang logic là L=32/8=4 byte=2^2🡺 n1=2

Np=2<5🡺 hợp lệ

Từ mục 2 của bảng PCB ta có

P=1🡺 trang logic 2 đã nạp RAM

A=3<8 số trang vật lý 🡺 trang logic 2 đã nạp vào frame.

D=4<4 không thỏa mãn🡺 địa chỉ logic bị lỗi.

* Ngừng tiến trình.

<3,5>

Kích thước 1 trang logic là L=32/8=4 byte=22

Np=3<5.

Từ mục 3 của bảng PCB ta có

P=1🡺 trang logic 3 đã nạp vào RAM

A=5<8 số trang vật lý🡺 trang logic 3 nạp vào frame.

D=5<4 kích thước 1 trang logic không thỏa mãn🡺 địa chỉ logic lỗi

🡺tiến trình dừng.

<5,1>

Kích thước 1 trang logic là L=32/8=4 byte 🡺n1=2

Np=5<5 không thỏa mãn 🡺 địa chỉ logic bị lỗi.

**Câu 4.27.**

Giả sử bộ nhớ vật lý (BNVL) có dung lượng 64MB, chương trình gồm 4 modul, xác định địa chỉ truy cập trong chiến lược phân trang – phân đoạn, với:

L=4kb

Kích thước 1 trang vật lý là

L=4kb=2^2\*2^10=2^12byte🡺n1=12 L=1/0000/0000/0000b

Số trang vật lý=dung lượng của BNVL/ kích thước 1 trang vật lý=64MB/2^12=2^14 trang

= 100 0000 0000 0000b=4000h

Số hiệu trang vật lý 0🡪4000h-1=3FFFh

S=0<4 hợp lệ

Từ mục 0 của bảng SCB ta có.

P=1🡺 đoạn 0 đã nạp vào RAM

Ap=400400h🡺 bảng PCB0 bắt đầu tại địa chỉ 400400h

Ls=4🡺 đoạn 0 chia làm 4 trang logic

Np=2<4🡺hợp lệ

Từ mục 2 của bảng PCB0 ta có

P=1🡺 trang logic 2 đã nạp vào RAM

A=406h🡺 trang logic 2 nạp vào frame 406h <=3FFFh

D=20Bh<1000h thỏa mãn 🡺 địa chỉ logic hợp lệ

Địa chỉ vật lý là PA=406h 12 bit or 20Bh

=406000+20Bh=40620Bh.

<1,0,46h>

L=4Kb🡺 Kích thước 1 trang vật lý là L=2^12 byte

🡺n1=12 L=10000 0000 0000b

Số trang vật lý= dung lượng bộ nhớ vật lý / kích thước 1 trang=2^14=4000h

S=1<4 hợp lệ

Từ mục 1 của bảng SCB ta có

P=0🡺 đoạn 1 không được nạp vào RAM.

* Hệ thống cần nạp hoặc đổi đoạn.

<2,1,1000h>

L=4KB

Kích thước 1 trang vật lý là L=2^12 byte🡺n1=12.

L=1000h

Số trang vật lý là = dung lượng bộ nhớ vaath lý / kích thước 1 trang=2^14 trang= 4000h

S=2<4 hợp lệ

Từ mục 2 của bảng SCB ta có

P=1 🡺 đoạn 2 đã nạp vào RAM.

Ap=400C00h🡺 Bảng PCB0 bắt đầu tại địa chỉ 400C00h.

L=3🡺đoạn 0 chia làm 3 trang logic

**Chương 5:Kỹ thuật đường ống RISC**

Câu 5.1:

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

Nhận lệnh (IF: Instruction Fetch).

Giải mã lệnh (ID: Instruction decode).

Tính địa chỉ toán hạng (Operand calculation - OC).

Nhận toán hạng (Operands Fetch – OF).

Thực hiện lệnh (Instruction execution – IE).

Ghi kết quả (WO – Write operation).

Ts=nxmxTc

Tp=mxTc+(n-1)xTc;

🡪m là số công đoạn

N là số câu lệnh

Tc là thời gian thực hiện 1 câu lệnh

Câu 5.2:

Trình bày các khó khăn và hướng khắc phục với kỹ thuật đường ống đơn vị lệnh

Bài làm:

**Khó khăn do điều khiển:**

Các lệnh làm thay đổi tính thi hành các lệnh một cách tuần tự (nghĩa là PC tăng đều đặn sau mỗi lệnh), gây khó khăn về điều khiển. Các lệnh này là lệnh nhảy đến một địa chỉ tuyệt đối chứa trong một thanh ghi, hay lệnh nhảy đến một địa chỉ xác định một cách tương đối so với địa chỉ hiện tại của bộ đếm chương trình PC. Các lệnh nhảy trên có thể có hoặc không điều kiện.

Trong trường hợp đơn giản nhất, tác vụ nhảy không thể biết trước giai đoạn giải mã (xem hình III.4). Như vậy, nếu lệnh nhảy bắt đầu ở chu kỳ C thì lệnh mà chương trình nhảy tới chỉ được bắt đầu ở chu kỳ C+2. Ngoài ra, phải biết địa chỉ cần nhảy đến mà ta có ở cuối giai đoạn giải mã ID. Trong lệnh nhảy tương đối, ta phải cộng độ dời chứa trong thanh ghi lệnh IR vào thanh ghi PC. Việc tính địa chỉ này chỉ được thực hiện vào giai đoạn ID với điều kiện phải có một mạch công việc riêng biệt.

Vậy trong trường hợp lệnh nhảy không điều kiện, lệnh mà chương trình nhảy đến bắt đầu thực hiện ở chu kỳ C+2 nếu lệnh nhảy bắt đầu ở chu kỳ C.

Khắc phục:

Vậy trường hợp đơn giản, người ta có thể được địa chỉ cần nhảy đến và điều kiện nhảy cuối giai đoạn ID. Vậy có chậm đi một chu kỳ mà người ta có thể giải quyết bằng nhiều cách.

Cách thứ nhất là đóng băng kỹ thuật ống dẫn trong một chu kỳ, nghĩa là ngưng thi hành lệnh thứ i+1 đang làm nếu lệnh thư i là lệnh nhảy. Ta mất trắng một chu kỳ cho mỗi lệnh nhảy.

Cách thứ hai là thi hành lệnh sau lệnh nhảy nhưng lưu ý rằng hiệu quả của một lệnh nhảy bị chậm mất một lệnh. Vậy lệnh theo sau lệnh nhảy được thực hiện trước khi lệnh mà chương trình phải nhảy tới được thực hiện. Chương trình dịch hay người lập trình có nhiệm vụ xen vào một lệnh hữu ích sau lệnh nhảy.

Câu 5.3:

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.

Câu 5.4:

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)

= 54 \* (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) = 110/27 = 4 lần.

Câu 5.5: Trình bày tóm tắt các đặc điểm trong kỹ thuật cài đặt RISC

Kích thước cố định trên 4 bytes

Sử dụng lệnh load/store

Nhiều thanh ghi:32 thanh ghi 32 bit cho số nguyên,32 thanh ghi 64 bit cho số thực

Đánh địa chỉ đơn giản

Tập lệnh nhỏ

Hỗ trợ ít kiểu dữ liệu

Các lệnh thực hiện nhanh chỉ cần 1 xung clock

Có bus riêng cho dữ liệu và mã lệnh.

Câu 5.6: -Cho biết ý nghĩa của cụm từ RISC (Reduced Instruction Set Computer)

 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).

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

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.