

Các bạn đang xem tài liệu lưu ý: Các bạn làm gì tài liệu cũng được, đọc, học, xem, photo cũng được nhưng vui lòng không photo ở quán Huyền Trang - Ao Sen. Bọn mình xin cảm ơn rất nhiều.

ĐỀ CƯƠNG ÔN TẬP MÔN KIẾN TRÚC MÁY TÍNH

I. Lý thuyết

Chương 1:

Các khái niệm liên quan đến kiến trúc máy tính:

Các thành phần của kiến trúc máy tính

Sơ đồ khối và đặc điểm của kiến trúc Von Neuman và Harvard (So sánh 2 kiến trúc này với nhau)

So sánh kiến trúc Von Neumann hiện đại và cổ điển

Chương 2.1

Sơ đồ khối tổng quát của CPU, chu kỳ xử lý lệnh của CPU

Các thanh ghi A, PC, SP, IR, MAR, MBR

Các thành phần, chức năng của thanh ghi cờ và ý nghĩa

Cho vd minh họa về các bit trong thanh ghi cờ: bao gồm các bit zero, cờ dấu, cờ tràn, cờ nhớ phụ, cờ chặn, cờ ngắt.

Sơ đồ khối về chức năng của các khối CU và ALU và bus bên trong CPU

Chương 2.2

Khái niệm về lệnh máy tính và các giai đoạn (pha) xử lý lệnh trong 1 chu kỳ của lệnh và khuôn dạng của lệnh

Cơ chế xử lý xen kẽ dòng lệnh, các sơ đồ minh họa

Các vấn đề liên quan đến xen kẽ dòng lệnh Data Hazard, Branch Hazard. Cách khắc phục

Câu hỏi: Các quy ước dạng lệnh, xử lý lệnh song song

Cho đoạn lệnh chương trình nào đó và yêu cầu xác định các thanh ghi sau khi thực hiện có giá trị bao nhiêu.

Hỏi thêm: đoạn lệnh thực hiện câu lệnh hay yêu cầu gì?, có xung đột dữ liệu không? nếu có thì xử lý ntn

Chương 3:

Khái niệm và vai trò của các cấu trúc phân cấp bộ nhớ

Bộ nhớ cache là gì, vai trò của bộ nhớ cache

Phương thức trao đổi dữ liệu của bộ nhớ cache với các thành phần trong máy tính

Lưu ý: look aside và look through

Các cơ chế, phương thức đọc ghi bộ nhớ.

Các yếu tố tăng hệ số hit và hiệu năng.

Chương 4: Kiến trúc tập lệnh vi xử lý 8086

Chức năng của các thanh ghi

RAID là gì? Tại sao RAID có thể nâng cao tính tin cậy, tốc độ truy cập và dung lượng hệ thống lưu trữ? Cấu hình RAID nào phù hợp hơn với máy chủ cơ sở dữ liệu trong ba loại RAID 0, RAID 1 và RAID 10? Giải thích?

Chương 7:

Vai trò của phương pháp vào ra dữ liệu: Thăm dò, Vào bằng map, trực tiếp DMA

So sánh các phương pháp trên, lấy ví dụ minh họa cho các phương pháp ấy.

II. Bài tập

Chương 3: Bộ nhớ cache

Giải mã về các phương pháp ánh xạ trực tiếp, đầy đủ, trực tiếp kết hợp

- Xác định số bit trong thành phần địa chỉ trong khi sử dụng

- Xác định vị trí 1 ô nhớ được đưa vào bộ cache
- Xác định kích thước cần có cho bộ nhớ cache dựa trên các yêu cầu

Chương 4: Kiến trúc tập lệnh vi xử lý 8086

Cú pháp của chương trình hợp ngữ: rẽ nhánh và lặp

Yêu cầu: Code các chương trình, Vẽ lưu đồ thuật toán, Các chế độ của lệnh ở 8086 (20 bài code)

Điều khiển đèn led

Điều khiển nhiệt độ trong 1 giá trị nào đó

Chương 6: Nối ghép bộ nhớ

Giải mã địa chỉ phối ghép bộ nhớ với CPU

LÝ THUYẾT (Câu 1)

CHƯƠNG 1:

Kiến trúc máy tính: Là khoa học về lựa chọn và kết nối các thành phần phần cứng của máy tính nhằm đạt yêu cầu:

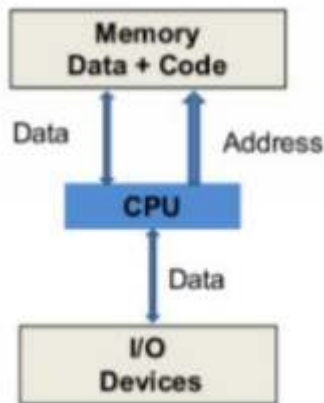
- Hiệu năng: càng nhanh càng tốt
- Chức năng: đáp ứng nhiều chức năng
- Giá thành: càng rẻ càng tốt

Các thành phần cơ bản của kiến trúc máy tính:

- **Kiến trúc tập lệnh:** Mô hình trừu tượng của máy tính ở mức ngôn ngữ máy (hợp ngữ) xác định những gì bộ xử lý thực hiện và cách thực hiện:
 - + Tập lệnh
 - + Các chế độ địa chỉ bộ nhớ
 - + Các thanh ghi
 - + Khuôn dạng địa chỉ và dữ liệu
- **Vi kiến trúc:** Là tổ chức máy tính, mô tả về hệ thống ở mức thấp, liên quan tới:
 - + Các thành phần cứng kết nối với nhau như thế nào
 - + Các thành phần phần cứng phối hợp, tương tác với nhau như thế nào để thực hiện tập lệnh
- **Thiết kế hệ thống:** Bao gồm tất cả các thành phần phần cứng khác trong hệ thống máy tính:
 - + Các hệ thống kết nối như bus và chuyển mạch
 - + Mạch điều khiển bộ nhớ, cấu trúc phân cấp bộ nhớ
 - + Các kỹ thuật giảm tải cho CPU như truy cập trực tiếp bộ nhớ
 - + Các vấn đề như đa xử lý

Sơ đồ khối và đặc điểm của kiến trúc Von Neumann:

- Sơ đồ khối



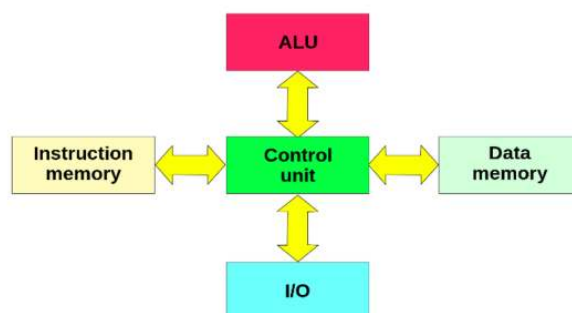
Von Neumann Machine

- Đặc điểm

- + Các máy tính kiến trúc Von-Neumann dựa trên 3 khái niệm cơ bản:
 - Dữ liệu và lệnh được lưu trong bộ nhớ đọc/ viết chia sẻ
 - Bộ nhớ được đánh địa chỉ dựa trên đoạn và không phụ thuộc vào việc nó lưu trữ gì
 - Các lệnh của chương trình được chạy lần lượt, lệnh nọ tiếp sau lệnh kia:
Stored-program digital computer
- + Quá trình thực hiện được chia thành 3 giai đoạn chính:
 - CPU lấy lệnh (fetch) từ bộ nhớ
 - CPU giải mã lệnh và chạy lệnh; nếu lệnh cần dữ liệu thì đọc dữ liệu từ bộ nhớ
 - CPU viết kết quả vào bộ nhớ nếu có

Sơ đồ khối và đặc điểm của kiến trúc Harvard

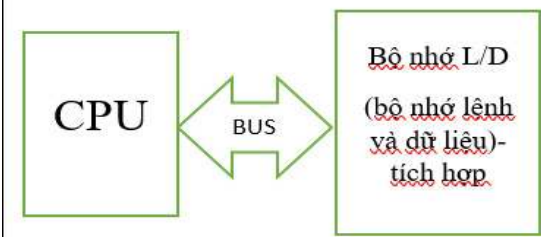
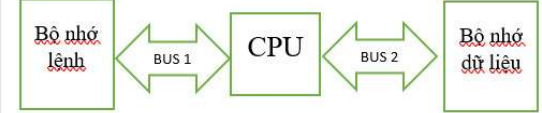
- Sơ đồ khối:



- Đặc điểm:

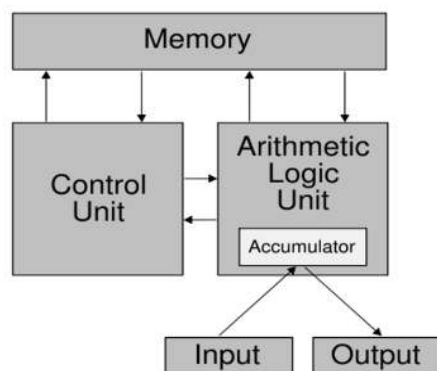
- + Chia bộ nhớ trong thành 2 phần riêng rẽ:
 - Bộ nhớ lưu chương trình (Program Memory)
 - Bộ nhớ lưu dữ liệu (Data Memory)
- + Hai hệ thống bus riêng được sử dụng để kết nối CPU với bộ nhớ lưu chương trình và bộ nhớ lưu dữ liệu
- + Mỗi bộ nhớ đều có 3 thành phần để truyền dẫn các tín hiệu địa chỉ, dữ liệu và điều khiển

So sánh kiến trúc Von Neumann và Harvard

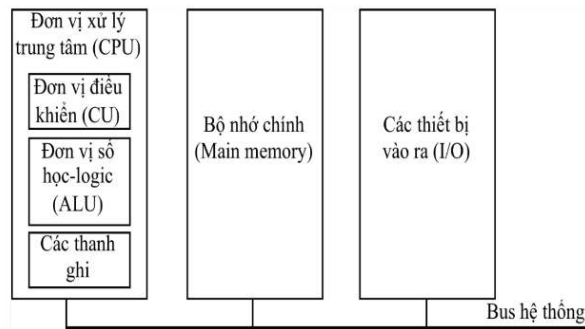
Kiến trúc Von Neumann	Kiến trúc Harvard
	
<p>Định nghĩa: Bộ nhớ lệnh và bộ nhớ dữ liệu nằm đan xen với nhau trong bộ nhớ vật lý. Được kết nối với bộ xử lý trung tâm bằng 1 hệ thống BUS</p>	<p>Định nghĩa: Bộ nhớ lệnh và bộ nhớ dữ liệu nằm trên hai khối vật lý khác nhau. Mỗi khối bộ nhớ được kết nối với bộ xử lý trung tâm (CPU) bằng hệ thống Bus riêng.</p>
<p>việc đọc lệnh và đọc/ghi dữ liệu phải làm tuần tự.</p>	<p>việc đọc lệnh và đọc/ghi dữ liệu có thể diễn ra cùng lúc.</p>
<p>Ứng dụng: trong các hệ thống máy tính phổ thông.</p>	<p>ứng dụng: trong các kiến trúc máy tính phức tạp, hệ thống máy chủ chuyên dụng, bộ xử lý kỹ thuật số.</p>
<p>Đơn giản hơn kiến trúc Harvard và chi phí thấp</p>	<p>phức tạp hơn kiến trúc Von Neumann (thiết kế 2 bus khác nhau cho dữ liệu và chương trình)</p>

So sánh kiến trúc Von Neumann cũ và hiện đại

- Kiến trúc Von Neumann cũ:

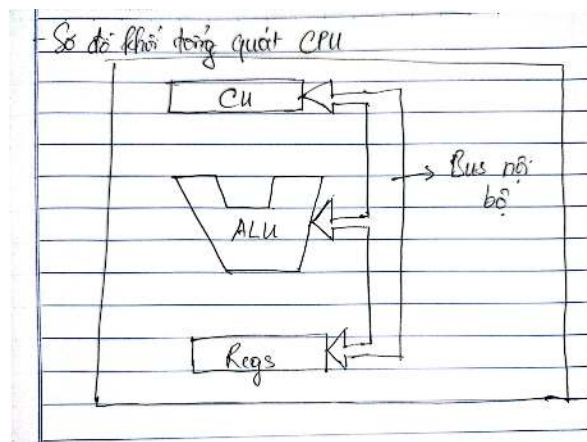
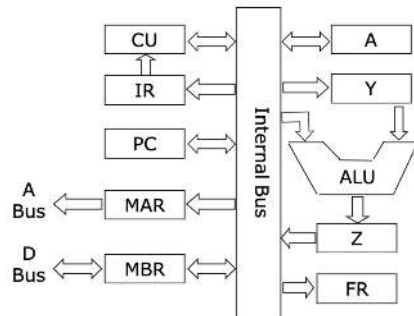


- Kiến trúc Von Neumann hiện đại:



CHƯƠNG 2:

Sơ đồ khối tổng quát của CPU



Thầy Trường

Thầy Sỹ

Cấu tạo của CPU

- gồm 3 thành phần chính:
 - + Khối xử lý (CU)
 - + Khối số học và logic (ALU)
 - + Các thanh ghi (Regs)
- Chức năng của các thành phần:
 - + Chức năng CU: nhận lệnh, giải mã lệnh thành mi lệnh và thi hành các mi lệnh
 - + Chức năng ALU:
 - Thực hiện phép tính số học +, -, *, /
 - Thực hiện phép tính logic and, not, or
 - > Theo sự điều khiển của CU
 - + Chức năng Regs: Chứa lệnh và dữ liệu cho hoạt động của CPU theo chức năng của một số thanh ghi điển hình (**phần các thanh ghi**)

Câu hỏi trong đề thi: nêu chức năng của các thành phần trong quá trình CPU xử lý lệnh ADD R1, R2

-> trả lời: Trong quá trình CPU xử lý lệnh ADD R1, R2, các thành phần chính bao gồm

1. Khối xử lý (Control Unit - CU): Đây là thành phần quản lý và điều khiển hoạt động của CPU. Chức năng chính của CU là lấy lệnh từ bộ nhớ, giải mã lệnh và điều khiển các thành phần khác của CPU để thực hiện lệnh đó. CU sẽ xác định các tín hiệu điều khiển và lựa chọn đường dẫn dữ liệu trong CPU để thực hiện lệnh ADD R1, R2.

2. Khối số học và logic (Arithmetic Logic Unit - ALU): ALU là thành phần chịu trách nhiệm thực hiện các phép toán số học và logic trong CPU. Trong trường hợp lệnh ADD R1, R2, ALU sẽ nhận giá

trị từ thanh ghi R1 và R2, thực hiện phép cộng và đưa kết quả vào một thanh ghi khác hoặc trả về cho CU.

3. Các thanh ghi (Registers): Các thanh ghi là các bộ nhớ nhanh và nhỏ được sử dụng trong CPU để lưu trữ dữ liệu tạm thời và các giá trị trung gian. Trong trường hợp lệnh ADD R1, R2, thanh ghi R1 và R2 chứa các giá trị cần được cộng lại với nhau. Kết quả của phép cộng sẽ được lưu trữ trong một thanh ghi khác hoặc được trả về cho CU.

Tóm lại, trong quá trình xử lý lệnh ADD R1, R2, Khối xử lý (CU) sẽ điều khiển hoạt động của ALU và truy cập vào các thanh ghi (Regs) để thực hiện phép cộng và lưu trữ kết quả.

Chu kỳ xử lý lệnh của CPU:

1. Khi một chương trình được chạy, hệ điều hành tải mã chương trình vào bộ nhớ trong RAM
2. Địa chỉ lệnh đầu tiên của chương trình được đưa vào thanh ghi PC
3. Địa chỉ của ô nhớ chứa lệnh được chuyển tới bus A qua thanh ghi MAR
4. Bus A truyền địa chỉ tới khối quản lý bộ nhớ MMU
5. MMU chọn ô nhớ và sinh ra tín hiệu READ
6. Lệnh chứa trong ô nhớ được chuyển tới thanh ghi MBR qua bus D
7. MBR chuyển lệnh tới thanh ghi IR. Sau đó IR lại chuyển lệnh tới CU
8. CU giải mã lệnh và sinh ra các tín hiệu xử lý cho các đơn vị khác, ví dụ như ALU để thực hiện lệnh cộng
9. Địa chỉ trong PC được tăng lên để trở tới lệnh tiếp theo của chương trình sẽ được thực hiện
10. Thực hiện lại các bước 3 -> 9 để chạy hết các lệnh của chương trình

Các thanh ghi:

1. Thanh ghi tích lũy A

- Thanh ghi tích lũy hay thanh ghi A là một trong những thanh ghi quan trọng nhất của CPU
 - + Lưu trữ các toán hạng đầu vào
 - + Lưu kết quả đầu ra
- Kích thước của thanh ghi A tương ứng với độ dài từ xử lý của CPU: 8, 16, 32, 64 bit
- Cũng được sử dụng để trao đổi dữ liệu với các thiết bị vào ra

2. Bộ đếm chương trình PC

- PC lưu địa chỉ của lệnh tiếp theo
- PC chứa địa chỉ ô nhớ chứa lệnh đầu tiên của chương trình khi nó được kích hoạt và được tải vào bộ nhớ
- Khi CPU chạy xong một lệnh, địa chỉ lệnh tiếp theo được tải vào PC
- Kích thước của PC phụ thuộc vào thiết kế CPU: 8, 16, 32, 64 bit

3. Con trỏ ngăn xếp SP

- Ngăn xếp là 1 đoạn bộ nhớ đặc biệt hoạt động theo nguyên tắc vào sau ra trước
- Con trỏ ngăn xếp là thanh ghi luôn trỏ tới đỉnh của ngăn xếp
- Hai thao tác với ngăn xếp: Push và Pop

4. Thanh ghi lệnh IR

- Lưu trữ lệnh đang được xử lý
- IR lấy lệnh từ MBR và chuyển nó tới CU để giải mã lệnh

5. Thanh ghi MAR

- Thanh ghi địa chỉ bộ nhớ

6. Thanh ghi MBR

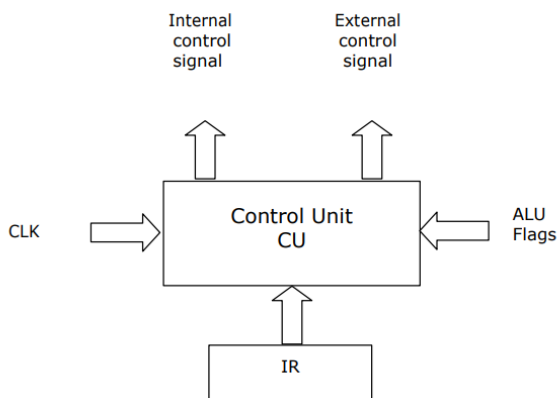
- Thanh ghi nhớ đệm

Thanh ghi trạng thái (Thanh ghi cờ)

- Thanh ghi trạng thái hoặc thanh ghi cờ là thanh ghi đặc biệt của CPU, mỗi bit của thanh ghi cờ lưu trạng thái của kết quả của phép tính ALU
- **Thành phần:**
 - + Cờ trạng thái: CF (cờ nhớ), OF (cờ tràn), AF, ZF (cờ zero), PF (cờ chẵn), SF (cờ dấu)
 - + Cờ điều khiển: IF (cờ ngắt), TF, DF
- **Chức năng:**
 - + Các bit cờ được sử dụng để điều khiển các lệnh rẽ nhánh lệnh tạo logic chương trình.
- **Ý nghĩa:**
 - + ZF: Cờ Zero, ZF=1 nếu kết quả=0 và ZF=0 nếu kết quả \neq 0.
 - + SF: Cờ dấu, SF=1 nếu kết quả âm và SF=0 nếu kết quả dương.
 - + CF: Cờ nhớ, CF=1 nếu có nhớ/mượn, CF=0 trong trường hợp khác.
 - + AF: Cờ nhớ phụ, AF=1 nếu có nhớ/mượn ở nửa thấp của toán hạng.
 - + OF: Cờ tràn, OF=1 nếu xảy ra tràn, OF=0 trong trường hợp khác
 - + PF: Cờ chẵn lẻ, PF=1 nếu tổng số bit 1 trong kết quả là lẻ và PF=0 nếu tổng số bit 1 trong kết quả là chẵn.
 - + IF: Cờ ngắt, IF=1: cho phép ngắt, IF=0: cấm ngắt.

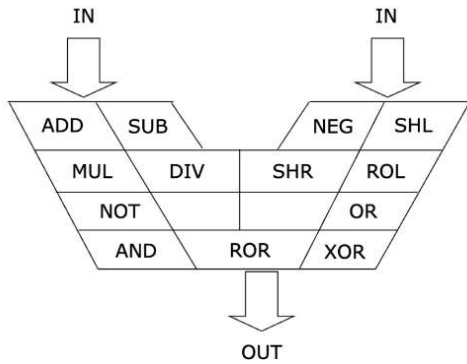
Ví dụ minh họa:

Sơ đồ khối về chức năng của khối CU:



- Điều khiển tất cả các hoạt động của CPU theo xung nhịp đồng hồ
- Nhận 3 tín hiệu đầu vào:
 - + Lệnh từ IR
 - + Giá trị các cờ trạng thái
 - + Xung đồng hồ
- CU sinh 2 nhóm tín hiệu đầu ra:
 - + Nhóm tín hiệu điều khiển các bộ phận bên trong CPU
 - + Nhóm tín hiệu điều khiển các bộ phận bên ngoài CPU
- Sử dụng nhịp đồng hồ để đồng bộ hóa các đơn vị bên trong CPU và giữa CPU với các thành phần bên ngoài

Sơ đồ khối về chức năng của khối ALU:



- ALU có:
 - + 2 cổng IN để nhận đầu vào từ các thanh ghi
 - + 1 cổng OUT được nối với bus trong để gửi kết quả tới các thanh ghi
- Bao gồm các đơn vị chức năng con để thực hiện các phép toán số học và logic:
 - + Bộ cộng (ADD), bộ trừ (SUB), bộ nhân (MUL), bộ chia (DIV), . . .
 - + Các bộ dịch (SHIFT) và quay (ROTATE)
 - + Bộ logic như phủ định (NOT), bộ và (AND), bộ hoặc (OR), và bộ hoặc loại trừ (XOR)

Các bus bên trong CPU:

- Bus trong là kênh liên lạc của tất cả các thành phần trong CPU
- Hỗ trợ liên lạc 2 chiều
- Bus trong có giao diện để trao đổi thông tin với bus ngoài
- Bus trong có băng thông lớn và tốc độ nhanh hơn so với bus ngoài

Khái niệm về lệnh máy tính

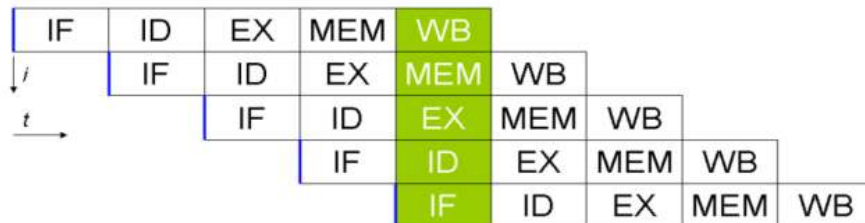
- **Lệnh máy tính:** là một từ nhị phân được gán một nhiệm vụ cụ thể, hướng dẫn cho máy tính biết phải làm gì.
- **Các giai đoạn xử lý lệnh:**
 - + Đọc lệnh IF: lệnh được đọc từ bộ nhớ vào CPU
 - + Giải mã lệnh ID: CPU giải mã lệnh
 - + Chạy lệnh IE: CPU thực hiện lệnh
 - + Ghi WB: kết quả (nếu có) ghi vào thanh ghi/ bộ nhớ
- **Khuôn dạng của lệnh**
 - + Dạng tổng quát của lệnh máy tính gồm 2 phần chính:
 - 1) Mã lệnh: Mã hóa cho thao tác mà bộ xử lý phải thực hiện
 - 2) Địa chỉ của các toán hạng: Chỉ ra nơi chứa các toán hạng mà thao tác sẽ tác động:
 - Toán hạng nguồn: dữ liệu vào của thao tác
 - Toán hạng đích: dữ liệu ra của thao tác

Mã lệnh	Địa chỉ của các toán hạng	
Opcode	Addresses of Operands	
Opcode	Des addr.	Source addr.

Cơ chế xử lý xen kẽ dòng lệnh (Cơ chế ống lệnh pipeline)

- Cơ chế xử lý xen kẽ dòng lệnh là một phương pháp cho phép đồng thời thực hiện nhiều lệnh, giảm thời gian trung bình thực hiện mỗi lệnh, tăng hiệu năng xử lý của CPU.

- Mỗi lệnh thực hiện theo 5 giai đoạn của hệ thống:
 - + IF Đọc lệnh: lấy lệnh từ bộ nhớ (hoặc cache)
 - + ID Giải mã lệnh: thực hiện giải mã lệnh và lấy các toán hạng
 - + IE Thực thi lệnh: Nếu là lệnh truy cập bộ nhớ thì tính toán địa chỉ bộ nhớ
 - + MEM Đọc-ghi bộ nhớ: Đọc và ghi bộ nhớ nếu không truy cập bộ nhớ thì không có giai đoạn này
 - + WB Ghi: Kết quả (nếu có) CPU xử lý được ghi vào thanh ghi/bộ nhớ lưu
- Sơ đồ thực hiện lệnh theo cơ chế xen kẽ dòng lệnh



Các vấn đề liên quan đến xen kẽ dòng lệnh

- **Xung đột tài nguyên:**
 - + Xung đột truy cập bộ nhớ
 - + Xung đột truy cập thanh ghi
- **Xung đột/ tranh chấp dữ liệu (Data Hazard)**
 - + Vấn đề read after write hazard
- **Các câu lệnh rẽ nhánh**
 - + Không điều kiện
 - + Có điều kiện
 - + Gọi thực hiện và trở về từ chương trình con

Cách khắc phục những vấn đề trên:

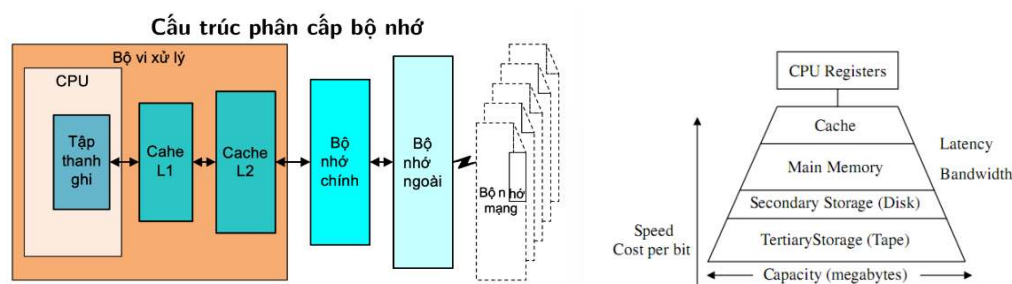
- **Xung đột tài nguyên:** Tài nguyên không đủ
 - + Nâng cao khả năng tài nguyên
 - + Memory/ cache: hỗ trợ nhiều thao tác đọc/ ghi cùng lúc
 - + Chia cache thành cache lệnh và cache dữ liệu để cải thiện truy nhập
- **Xung đột/ tranh chấp dữ liệu (Data Hazard):** Tranh chấp đọc sau khi ghi (RAW) là kiểu hay gặp nhất
 - + Nhận dạng tranh chấp RAW khi nó diễn ra (các lệnh phụ thuộc)
 - + Khi tranh chấp RAW xảy ra, tạm dừng (stall) pipeline cho đến khi lệnh phía trước hoàn tất giai đoạn WB
 - + Có thể dùng compiler để nhận dạng RAW và thực hiện:
 - Chèn các lệnh NO-OP và giữa các lệnh có thể gây ra tranh chấp (NO-OP là lệnh rỗng, và chỉ dùng để tiêu tốn thời của CPU)
 - Thay đổi thứ tự thực hiện của các lệnh trong chương trình và chèn các lệnh độc lập vào giữa các lệnh có thể gây ra tranh chấp RAW.
 - + Sử dụng phần cứng để xác định RAW và dự đoán trước giá trị dữ liệu phụ thuộc.
- **Xử lý rẽ nhánh:** Lệnh rẽ nhánh làm thay đổi bộ đếm của chương trình nên có thể làm thay đổi thứ tự thực hiện các lệnh của chương trình pipeline.
 - + Địch rẽ nhánh
 - + Dự đoán rẽ nhánh
 - + Làm chậm rẽ nhánh:
 - Sử dụng compiler để chèn NO-OP vào vị trí ngay sau lệnh rẽ nhánh
 - Chuyển một lệnh độc lập từ trước tới ngay sau lệnh rẽ nhánh

Phân biệt hai loại máy tính CISC và RISC (chú ý căn cứ vào đầu để phân loại các máy tính này).

Tiêu chí	RISC	CISC
Cấu tạo CU	FSM	Vi chương trình
Số lượng lệnh trong tập lệnh	80-100 lệnh	200-300 lệnh
Tốc độ xử lý lệnh	nhanh	chậm hơn
Mức độ phức tạp khi lập trình	phức tạp hơn	đơn giản hơn
Ứng dụng	Bộ điều khiển đơn giản Bộ vi điều khiển (ICpc - chip đủ chức năng của máy tính PC)	máy tính (PC)

CHƯƠNG 3:

Cấu trúc phân cấp bộ nhớ



Vai trò của cấu trúc phân cấp bộ nhớ

- Nâng cao hiệu năng hệ thống:
 - + Dung hòa được CPU có tốc độ cao với bộ nhớ chính và bộ nhớ phụ có tốc độ thấp
 - + Thời gian truy cập dữ liệu trung bình của CPU từ hệ thống bộ nhớ gần bằng thời gian truy cập cache
- Giảm giá thành sản xuất:
 - + Các thành phần đắt tiền sẽ được sử dụng với dung lượng nhỏ hơn
 - + Các thành phần rẻ hơn được sử dụng với dung lượng lớn hơn
 - + Tổng giá thành của hệ thống nhớ theo mô hình phân cấp sẽ rẻ hơn so với hệ thống nhớ không phân cấp cùng tốc độ

Bộ nhớ cache

- Cache là thành phần nhớ trong sơ đồ phân cấp bộ nhớ máy tính.
 - + Nó hoạt động như thành phần trung gian, trung chuyển dữ liệu từ bộ nhớ chính về CPU và ngược lại
- Vị trí của cache:
 - + Với các hệ thống cũ, cache thường nằm ngoài CPU
 - + Với các CPU mới, cache thường được tích hợp vào trong CPU

- Dung lượng cache thường nhỏ:
- Tốc độ truy nhập của cache nhanh hơn so với tốc độ bộ nhớ chính
- Giá thành cache (tính trên bit) thường đắt hơn so với bộ nhớ chính

Vai trò của bộ nhớ cache

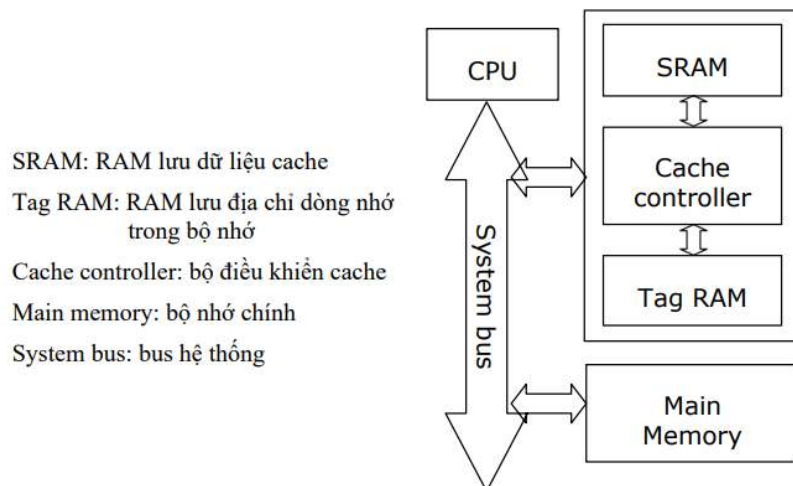
- Nâng cao hiệu năng hệ thống:
 - + Dung hòa giữa CPU có tốc độ cao và bộ nhớ chính tốc độ thấp (giảm số lượng truy cập trực tiếp của CPU vào bộ nhớ chính)
 - + Thời gian trung bình CPU truy cập hệ thống bộ nhớ gần bằng thời gian truy cập cache
- Giảm giá thành sản xuất:
 - + Nếu 2 hệ thống có cùng hiệu năng thì hệ thống có cache sẽ rẻ hơn
 - + Nếu 2 hệ thống cùng giá thành, hệ thống có cache sẽ nhanh hơn

Phương thức trao đổi dữ liệu của bộ nhớ cache với các thành phần trong máy tính

- CPU trao đổi dữ liệu với cache theo các đơn vị cơ sở như byte, từ và từ kép.
- Cache trao đổi dữ liệu với bộ nhớ chính theo các khối, với kích thước 16, 32, 64 bytes.

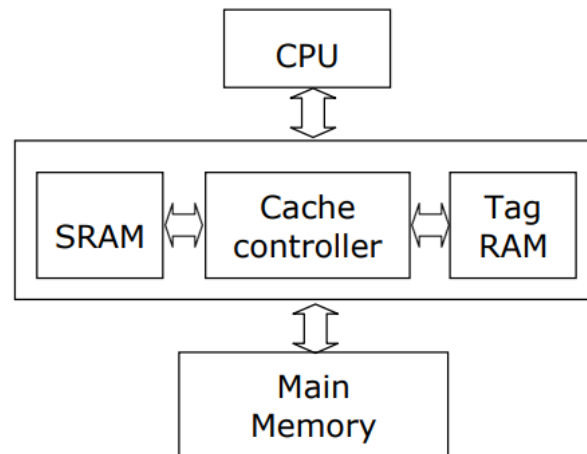
* Lưu ý:

- **Kiến trúc Look aside:**
 - + Cache và bộ nhớ cùng được kết nối tới bus hệ thống
 - + Cache và bộ nhớ chính “thấy” chu kỳ bus CPU tại cùng một thời điểm
 - + Ưu điểm:
 - Thiết kế đơn giản
 - Miss nhanh: khi CPU không tìm thấy mục dữ liệu trong cache nó đồng thời tìm trong bộ nhớ chính cùng 1 chu kỳ xung nhịp
 - + Nhược điểm:
 - Hit chậm: do cache kết nối với CPU sử dụng bus hệ thống có tần số làm việc không cao và băng thông hẹp



- **Kiến trúc Look Through:**
 - + Cache nằm giữa CPU và bộ nhớ chính
 - + Cache “thấy” chu kỳ bus CPU trước sau đó nó “truyền” lại cho bộ nhớ chính
 - + Ưu điểm:
 - Hit nhanh : Cache kết nối với CPU bằng bus riêng tốc độ cao và có băng thông lớn
 - + Nhược điểm:
 - Đắt
 - Thiết kế phức tạp

- Miss chậm : khi CPU không tìm thấy dữ liệu trong cache nó cần tìm trong Bộ nhớ tại 1 xung nhịp tiếp theo



Câu hỏi thêm trong đề: Look Aside và Look Through cái nào được sử dụng nhiều hơn trong thực tế hiện nay. Tại sao?

=> Trả lời:

- Look Aside được sử dụng nhiều hơn trong thực tế vì có những lợi ích sau:
 - + Do Cache và bộ nhớ chính đều kết nối với BUS hệ thống nên nó cho phép việc truy xuất Cache và bộ nhớ chính xảy ra độc lập, giúp giảm thiểu sự chồng chéo và tăng hiệu suất
 - + Nó cho phép kiểm soát rõ ràng hơn về việc cập nhật dữ liệu giữa cache và bộ nhớ chính. Ví dụ: nếu Cache không có dữ liệu, CPU sẽ truy xuất thẳng vào bộ nhớ chính đọc dữ liệu, sau đó Cache sẽ cập nhật dữ liệu CPU đã đọc
 - + Cung cấp khả năng mở rộng tốt hơn vì Cache có thể được mở rộng mà không ảnh hưởng đến việc truy xuất bộ nhớ chính

Dấu /*/: không chắc chắn

/*/Các cơ chế, phương thức đọc ghi bộ nhớ (nguyên lý hoạt động của cache)

- Cục bộ (lân cận) về không gian:
 - + Nếu một vị trí bộ nhớ được truy cập, thì xác suất các vị trí gần đó được truy cập trong thời gian gần tới là cao
 - + Áp dụng với dữ liệu và các lệnh có thứ tự tuần tự theo chương trình
 - + Lệnh trong chương trình thường có thứ tự tuần tự, do đó cache đọc một khối lệnh trong bộ nhớ, mà bao gồm cả các phần tử xung quanh vị trí phần tử hiện tại được truy cập.

-> Ví dụ minh họa: Giả sử chúng ta có một chương trình tính tổng của một mảng lớn chứa các phần tử số nguyên. Chương trình lặp qua mảng và tính tổng bằng cách cộng các phần tử lại với nhau.

Khi chúng ta truy cập vào một phần tử đầu tiên của mảng, nó sẽ được nạp vào cache cùng với một số khối dữ liệu xung quanh. Sau đó, chúng ta tiếp tục truy cập vào các phần tử tiếp theo trong mảng. Vì các phần tử này nằm gần nhau và đã được nạp vào cache, chúng ta có thể truy cập dữ liệu từ cache một cách nhanh chóng, không cần truy cập lại bộ nhớ chính. Điều này giúp tận dụng nguyên lý cận kề về không gian
- Cục bộ (lân cận) về thời gian:
 - + Nếu một vị trí bộ nhớ được truy cập, thì xác suất nó được truy cập lại trong thời gian gần là cao
 - + Áp dụng với các mục dữ liệu và các lệnh trong vòng lặp

- + Cache đọc khối dữ liệu trong bộ nhớ gồm tất cả các thành phần trong vòng lặp
 -> Ví dụ minh họa vẫn ví dụ trên: trong vòng lặp trên, ta truy cập lại các phần tử đã được truy cập trước đó. Khi chúng ta truy cập lại các phần tử này, chúng đã được lưu trong cache và có thể truy xuất nhanh chóng mà không cần truy cập lại bộ nhớ chính. Điều này tận dụng nguyên lý cận kề về thời gian.

Các yếu tố tăng hệ số hit và hiệu năng

1. Kích thước cache:

- Với kích thước lớn, có thể giảm tần suất trao đổi các dòng cache của các chương trình khác nhau với bộ nhớ chính. Đồng thời
- Hỗ trợ đa nhiệm, xử lý song song và các hệ thống CPU nhiều nhân tốt hơn.
- Nhược điểm của cache lớn là chậm

2. Chia tách cache

- Dữ liệu và lệnh có tính lân cận khác nhau;
- Dữ liệu thường có tính lân cận về thời gian cao hơn lân cận về không gian; lệnh có tính lân cận về không gian cao hơn lân cận về thời gian;
- Cache lệnh chỉ cần hỗ trợ thao tác đọc; cache dữ liệu cần hỗ trợ cả 2 thao tác đọc và ghi và tách cache giúp tối ưu hoá dễ dàng hơn;
- Hỗ trợ nhiều lệnh truy nhập đồng thời hệ thống nhớ, giảm xung đột tài nguyên cho CPU pipeline.

3. Tạo cache nhiều mức

- Hệ thống nhớ với nhiều mức cache có khả năng dung hòa tốc độ giữa các thành phần tốt hơn và có thời gian truy nhập trung bình hệ thống nhớ thấp hơn
- Nhiều mức cache có thể giúp giảm giá thành hệ thống nhớ.
-

Hiệu năng cache

Thời gian truy cập trung bình của hệ thống nhớ có cache:

$$\begin{aligned} T_{\text{access}} &= (\text{hit cost}) + (\text{miss cost}) \\ &= (\text{Hit cost}) + (\text{miss rate}) * (\text{miss penalty}) \\ &= t_{\text{cache}} + (1 - H) * (t_{\text{memory}}) \end{aligned}$$

Trong đó H là hệ số hit

Nếu $t_{\text{cache}} = 5\text{ns}$, $t_{\text{memory}} = 60\text{ns}$ và $H=80\%$, ta có:

$$t_{\text{access}} = 5 + (1 - 0.8) * (60) = 5 + 12 = 17\text{ns}$$

Nếu $t_{\text{cache}} = 5\text{ns}$, $t_{\text{memory}} = 60\text{ns}$ và $H=95\%$, ta có:

$$t_{\text{access}} = 5 + (1 - 0.95) * (60) = 5 + 3 = 8\text{ns}$$

Chính sách thay thế dòng Cache:

1. Ngẫu nhiên

- Các dòng cache được chọn ngẫu nhiên để thay thế
- Ưu điểm: Cài đặt đơn giản, nhanh
- Nhược điểm:
 - + Khi đưa dòng cache mới thay thế ngẫu nhiên không xem xét đến các dòng cache đang được sử dụng

- + Nếu một dòng cache đang được sử dụng và bị thay thế -> xảy ra xung đột dữ liệu, bị miss dữ liệu và nó lại cần được đọc từ bộ nhớ chính vào cache.
-> hệ số miss cao
- 2. Thay thế kiểu vào trước ra trước (FIFO - First in First out)
 - Các dòng cache được đọc vào cache trước sẽ bị chọn để thay thế trước (dòng cache có tuổi thọ cao nhất sẽ được đưa vào trước - đọc để hiểu dùng ghi vào bài thi phần mở ngoặc này =)))
 - Ưu điểm: Do biết được thời gian của các dòng cache sẽ bị thay thế nên giảm thiểu việc dữ liệu bị mất => có hệ số miss thấp hơn so với thay thế ngẫu nhiên
 - Nhược điểm:
 - + Thay thế vẫn chưa thực sự xem xét đến các dòng cache đang được sử dụng. Một dòng cache “già” vẫn có thể đang được sử dụng => hệ số miss vẫn còn cao
 - + Cài đặt phức tạp do cần có mạch điện tử (đồng hồ đo thời gian sống của các dòng cache) để theo dõi trật tự nạp các dòng bộ nhớ vào cache
- 3. Thay thế các dòng ít được sử dụng gần đây nhất (LRU)
 - Các dòng cache ít được sử dụng gần đây nhất được lựa chọn để thay thế
 - Ưu điểm:
 - + Có hệ số miss thấp nhất so thay thế ngẫu nhiên và thay thế vào trước ra trước
 - + Do thay thế LRU có xem xét đến các dòng cache đang được sử dụng
 - Nhược điểm: Cài đặt phức tạp do cần có mạch điện tử để theo dõi tần suất sử dụng của dòng cache

Câu hỏi trong đề thi: Chính sách nào cho hiệu quả cao nhất? Vì sao

=> Trả lời: *Chính sách thay thế dòng ít được sử dụng gần đây nhất (LRU) cho hiệu quả cao nhất. Vì nó xem xét đến các dòng cache đang sử dụng. Nếu dòng cache nào ít hoạt động thì sẽ bị thay thế và ngược lại dòng cache hoạt động nhiều, lưu nhiều dữ liệu bộ nhớ sẽ không bị thay thế. Dẫn đến hệ số miss thấp và hệ số Hit cao. Nên chính sách này cho hiệu suất tốt nhất.*

CHƯƠNG 4:

Chức năng của các thanh ghi

*Thanh ghi đa năng

- **AX:** Thanh ghi tổng, thường dùng để lưu kết quả
- **BX:** Thanh ghi cơ sở, thường dùng chứa địa chỉ ô nhớ
- **CX:** Thanh ghi đếm, thường dùng làm con đếm cho các lệnh lặp, chứa số lần dịch hoặc quay trong các lệnh dịch và quay thanh ghi
- **DX:** Thanh ghi dữ liệu, cùng AX chứa dữ liệu trong các phép tính nhân/ chia số 16 bit hoặc chứa địa chỉ cổng trong các lệnh vào ra dữ liệu trực tiếp (IN/OUT)

*Thanh ghi con trỏ và chỉ số

- **SP:** con trỏ ngăn xếp, SP luôn chứa địa chỉ đỉnh ngăn xếp SS:SP
- **BP:** Con trỏ cơ sở, chứa địa chỉ của dữ liệu trong đoạn ngăn xếp SS hoặc các đoạn khác SS:SP
- **SI:** Thanh ghi chỉ số nguồn, SI thường dùng chứa địa chỉ ô nhớ nguồn ở đoạn dữ liệu DS như trong các lệnh chuỗi trong các thao tác truyền dữ liệu DS:SI
- **DI:** Thanh ghi chỉ số đích, DI thường dùng chứa địa chỉ ô nhớ đích ở đoạn dữ liệu DS trong các thao tác chuyển dữ liệu

Các chế độ định địa chỉ của vi xử lý intel 8086 Cho ví dụ minh họa cho các chế độ định địa chỉ?

- Định nghĩa chế độ địa chỉ là cách CPU tổ chức các toán hạng của lệnh. Gồm 9 chế độ định địa chỉ:
 1. Chế độ định địa chỉ tức thì (Im):
 - Định nghĩa: Toán hạng nguồn là 1 hằng số
 - ví dụ:
MOV AH, 08H ;AH=08H
 2. Chế độ định địa chỉ thanh ghi (Reg)
 - Định nghĩa: Dữ liệu toán hạng nằm trong thanh ghi
 - ví dụ
MOV AH, 08H ;AH=08H
MOV AH, AL ;AH=AL
 3. Chế độ định địa chỉ trực tiếp (Mem)
 - Định nghĩa: Dữ liệu toán hạng nằm trong bộ nhớ, có địa chỉ độ lệnh là 1 hằng số
 - Ví dụ:
MOV AH, [08H] ;AH=[08H] =100 toán hạng nguồn
MOV [08H], DL ;toán hạng đích
 4. Chế độ định địa chỉ gián tiếp (Mem)
 - Định nghĩa: Dữ liệu toán hạng nằm trong bộ nhớ, có địa chỉ độ lệnh là giá trị của thanh ghi (thanh ghi chứa địa chỉ này phải là thanh ghi 16 bit)
 - Ví dụ:
MOV AH, [BX] ;AH=[BX]
MOV [BX], DL
 5. Chế độ định địa chỉ gián tiếp thanh ghi
 - Định nghĩa: Dữ liệu toán hạng nằm trong bộ nhớ, có địa chỉ độ lệnh là giá trị của thanh ghi BX/BP/SI/DI
 - Ví dụ:
MOV AH, [BX]
MOV AH, [BP]
 6. Chế độ định địa chỉ cơ sở và chế độ định địa chỉ chỉ số
 - Bản chất là chế độ định địa chỉ gián tiếp thanh ghi. Khác nhau là dùng thanh ghi nào
 - Với chế độ định địa chỉ cơ sở: Dữ liệu toán hạng nằm trong bộ nhớ, có địa chỉ độ lệnh là giá trị của thanh ghi cơ sở BX/BP
 - Với chế độ định địa chỉ chỉ số: Dữ liệu toán hạng nằm trong bộ nhớ, có địa chỉ độ lệnh là giá trị của thanh ghi chỉ số SI/DI
 7. Chế độ định địa chỉ tương đối cơ sở
 - Định nghĩa: Dữ liệu toán hạng nằm trong bộ nhớ, có địa chỉ độ lệnh là sự kết hợp của giá trị thanh ghi cơ sở BX/BP với 1 hằng số
 - Ví dụ: MOV AH, [BX+10]
 8. Chế độ định địa chỉ tương đối chỉ số
 - Định nghĩa: Dữ liệu toán hạng nằm trong bộ nhớ, có địa chỉ độ lệnh là sự kết hợp của giá trị thanh ghi chỉ số SI/DI với 1 hằng số
 - Ví dụ: MOV AH, [SI+10]
 9. Chế độ định địa chỉ cơ sở chỉ số
 - Định nghĩa: Dữ liệu toán hạng nằm trong bộ nhớ, có địa chỉ độ lệnh là sự kết hợp của giá trị thanh ghi cơ sở BX/BP và thanh ghi chỉ số SI/DI
 - Ví dụ:

MOV AH, [BX + SI]

MOV [BP + DI], AL

10. Chế độ định địa chỉ tương đối cơ sở chỉ số

- Định nghĩa: Dữ liệu toán hạng nằm trong bộ nhớ, có địa chỉ độ lệnh là sự kết hợp của giá trị thanh ghi cơ sở BX/BP, thanh ghi chỉ số SI/DI và 1 hằng số

- Ví dụ:

MOV AH, [BX + SI + 10]

MOV [BP + DI + 10], AL

Note: Với câu hỏi đề bài yêu cầu so sánh các chế độ sẽ dùng 2 tiêu chí là định nghĩa và ví dụ để so sánh

RAID là gì? Tại sao RAID có thể nâng cao tính tin cậy, tốc độ truy cập và dung lượng hệ thống lưu trữ? Cấu hình RAID nào phù hợp hơn với máy chủ cơ sở dữ liệu trong ba loại RAID 0, RAID 1 và RAID 10? Giải thích?

- Định nghĩa RAID: là một công nghệ tạo các thiết bị lưu trữ tiên tiến trên cơ sở các ổ đĩa độc lập, nhằm các mục đích:
 - + Tốc độ cao (High performance / speed)
 - + Tính tin cậy cao (High reliability)
 - + Dung lượng lớn (Large volume)
- Cấu hình RAID nào phù hợp hơn với máy chủ cơ sở dữ liệu trong ba loại RAID 0, RAID 1 và RAID 10? Giải thích?

RAID 8 ổ	RAID10 4D + 4D	RAID5 7D + 1P	RAID6 6D + 2P
Tốc độ (IOPS)	X 4	X 7	X 6
Dung lượng hiệu dụng	50%	87,5%	75%
Độ tin cậy	Hỏng 1 ổ	Hỏng 1 ổ	Hỏng 2 ổ
Response Time	1	2	3

- + RAID10 phù hợp máy chủ cơ sở dữ liệu: Thời gian đáp ứng, thời gian phản hồi nhanh
- + RAID5 phù hợp đầu tư kinh doanh: do tốc độ nhanh, dung lượng hiệu dụng lớn 87,5%
- + RAID6 phù hợp cho kĩ thuật: Khả năng đảm bảo lớn do cho phép khả năng hỏng 2 ổ

Các loại hệ thống lưu trữ

So sánh các loại hệ thống lưu trữ

Tiêu chí	SAN Storages	NAS	Object Storages
Phương thức trao đổi	Block	File	Object
Giao thức	FC	NFS, CIFS,..	S3, NFS
Giao diện ghép nối (với máy chủ)	FC	NIC	NIC
Thiết bị mạng	SAN SW	LAN SW	LAN SW
Kết nối	Point to Point	Dùng chung	Dùng chung
Tốc độ	Cao	Bị ảnh hưởng trong mạng	Bị ảnh hưởng trong mạng

CHƯƠNG 7:

Vai trò của vào ra dữ liệu

- Là phương tiện giúp CPU giao tiếp với thế giới bên ngoài
- Cung cấp dữ liệu đầu vào cho CPU xử lý
- Cung cấp phương tiện để CPU kết xuất dữ liệu đầu ra

Vào ra bằng thăm dò

- **Cơ chế vào ra bằng thăm dò**
 - + CPU quản lý danh sách các thiết bị vào ra kèm theo địa chỉ các cổng giao tiếp;
 - + Các thiết bị vào ra định kỳ cập nhật trạng thái sẵn sàng làm việc của mình lên các bit cờ trạng thái vào ra của mình;
 - + CPU định kỳ lần lượt “quét” các thiết bị vào ra để “đọc” các bit cờ trạng thái vào ra;
 - Nếu gặp một thiết bị sẵn sàng làm việc, 2 bên tiến hành trao đổi dữ liệu;
 - Trao đổi dữ liệu xong, CPU tiếp tục quét thiết bị khác.
- **Ưu điểm**
 - + Đơn giản, dễ cài đặt
 - + Có thể được cài đặt bằng phần mềm
- **Nhược điểm**
 - + Hiệu quả thấp do CPU tốn nhiều thời gian để thăm dò các thiết bị
 - + Không thực sự khả thi khi có nhiều thiết bị trong danh sách thăm dò
- **Bên chủ động trong vào ra bằng thăm dò**
 - + CPU là bên chủ động trong quá trình trao đổi dữ liệu
- **Ứng dụng**
 - + Thăm dò thường được sử dụng khi hệ thống khởi động: CPU thăm dò hầu hết các thiết bị để xác lập cấu hình
 - + Thăm dò được sử dụng trong quá trình hoạt động với các thiết bị rời (removable) như ổ đĩa CD/DVD, ổ mềm, ...
- **Ví dụ minh họa:** Trong một chương trình đọc dữ liệu từ bàn phím, hệ thống sử dụng phương pháp vào ra bằng thăm dò. Chương trình sẽ liên tục kiểm tra xem có ký tự mới được nhấn từ bàn phím không. Nếu không có ký tự mới, chương trình sẽ tiếp tục thực hiện các tác vụ khác

hoặc chờ đợi. Khi có ký tự mới được nhận, chương trình sẽ đọc ký tự đó và xử lý theo yêu cầu.

Vào ra bằng ngắt

- **Cơ chế vào ra bằng ngắt**
 - + Thiết bị khởi xướng việc trao đổi vào ra
 - + Thiết bị được nối với chân tín hiệu ngắt (INT), khi thiết bị cần trao đổi dữ liệu thiết bị sẽ sinh ra tín hiệu ngắt
 - + Máy tính hoàn thành câu lệnh hiện thời và lưu nội dung của bộ đếm chương trình và các thanh ghi trạng thái
 - + Sau đó máy tính tự động nạp địa chỉ chương trình phục vụ ngắt vào thanh ghi đếm chương trình
 - + Khôi phục thanh ghi đếm chương trình bị dừng và thanh ghi trạng thái của vi xử lý
- **Ưu điểm**
 - + Hiệu quả hơn vào ra bằng thăm dò, do CPU không phải thăm dò từng thiết bị
- **Nhược điểm**
 - + Phức tạp hơn vào ra bằng thăm dò
 - + Cần mạch phần cứng để điều khiển ngắt
- **Bên chủ động trong vào ra bằng ngắt**
 - + Thiết bị vào ra
- **Ví dụ minh họa:** *Trong một chương trình đọc dữ liệu từ bàn phím, hệ thống sử dụng phương pháp vào ra bằng ngắt. Khi có một phím được nhấn trên bàn phím, thiết bị gửi một tín hiệu ngắt đến CPU. CPU chuyển quyền điều khiển sang chương trình ngắt, được thiết lập trước đó. Chương trình ngắt sẽ đọc ký tự từ bàn phím và xử lý theo yêu cầu. Giao tài cho CPU, CPU chỉ thực hiện khi có lệnh ngắt từ thiết bị vào ra (bàn phím).*

Vào ra bằng DMA

- Phương pháp vào ra bằng DMA cho phép thiết bị vào ra trao đổi dữ liệu trực tiếp với bộ nhớ theo khối thông qua CPU
- DMA thích hợp khi cần trao đổi dữ liệu với khối lượng lớn trong khoảng thời gian ngắn
- **Ưu điểm**
 - + Hiệu suất cao do dữ liệu trao đổi trực tiếp theo khối giữa thiết bị vào ra và bộ nhớ không thông qua CPU
- **Nhược điểm**
 - + Phức tạp hơn vào ra bằng thăm dò và ngắt
 - + Cần mạch phần cứng để điều khiển quá trình DMA
- **Bên chủ động trong vào ra bằng DMA**
 - + Thiết bị vào ra

BÀI TẬP

Dạng bài tập 1: CPU PIPELINE (Câu 4)

Kiến thức cần có để làm bài :)))

Chế độ địa chỉ	Ý nghĩa	Ví dụ	Thực hiện
Tức thì	Giá trị của toán hạng được chứa trong lệnh	LOAD Ri, #1000	$R_i \leftarrow 1000$
Trực tiếp	Địa chỉ của toán hạng được chứa trong lệnh	LOAD Ri, 1000	$R_i \leftarrow M[1000]$
Gián tiếp thanh ghi	Giá trị của thanh ghi trong lệnh là địa chỉ bộ nhớ chứa toán hạng	LOAD Ri, (Rj)	$R_i \leftarrow M[R_j]$
Gián tiếp bộ nhớ	Địa chỉ bộ nhớ trong lệnh chứa địa chỉ bộ nhớ của toán hạng	LOAD Ri, (1000)	$R_i \leftarrow M[M[1000]]$
Chỉ số	Địa chỉ của toán hạng là tổng của hằng số (trong lệnh) và giá trị của một thanh ghi chỉ số	LOAD Ri, X(Rind)	$R_i \leftarrow M[X + Rind]$
Tương đối	Địa chỉ của toán hạng là tổng của hằng số và giá trị của thanh ghi con đếm chương trình	LOAD Ri, X(PC)	$R_i \leftarrow M[X + PC]$

Bài tập:

Bài 1: Cơ chế ống lệnh (pipeline) của CPU thường gặp phải những vấn đề gì? Nêu một hướng giải quyết xung đột dữ liệu trong pipeline khi thực hiện đoạn chương trình sau:

- (1) ADD R1, R2, R3 ; $R1 \leftarrow R2 + R3$
- (2) ADD R4, R4, #300 ; $R4 \leftarrow R4 + 300$
- (3) CMP R1, #100 ; so sánh R1 với 100
- (4) SUB R5, #2000 ; $R5 \leftarrow R5 + 2000$

Biết rằng mỗi lệnh được chia thành 5 giai đoạn trong pipeline: Đọc lệnh (IF), giải mã & đọc toán hạng (ID), truy nhập bộ nhớ (MEM), thực hiện (EX) và lưu kết quả (WB).

Bài làm:

Cơ chế ống lệnh (pipeline) của CPU thường gặp phải những vấn đề sau:

- **Xung đột tài nguyên:**
 - + Xung đột truy cập bộ nhớ
 - + Xung đột truy cập thanh ghi
- **Xung đột/ tranh chấp dữ liệu (Data Hazard)**
 - + Vấn đề read after write hazard
- **Các câu lệnh rẽ nhánh**
 - + Không điều kiện
 - + Có điều kiện
 - + Gọi thực hiện và trở về từ chương trình con

Hướng giải quyết xung đột dữ liệu trong pipeline khi thực hiện đoạn chương trình sau: làm chậm chu kỳ (T)

- Khi thực hiện ống lệnh ta gặp xung đột/ tranh chấp về dữ liệu khi thực hiện chương trình
- Ta tìm ra những câu lệnh phụ thuộc (những câu lệnh có khả năng gây ra xung đột)
- Chèn những câu lệnh rỗng NO-OP vào giữa những câu lệnh phụ thuộc
- Thay đổi trật tự của các câu lệnh không phụ thuộc

Áp dụng

Các câu lệnh phụ thuộc:

(3) -> (1)

Viết lại

(1)	IF	ID	MEM	EX	WB				
(2)		IF	ID	MEM	EX	WB			
(4)			IF	ID	MEM	EX	WB		
NO-OP				NO					

(3)					IF	ID	MEM	EX	WB
-----	--	--	--	--	----	----	-----	----	----

Bài 2: Cơ chế ống lệnh (pipeline) của CPU thường gặp phải những vấn đề gì? Nêu một hướng giải quyết xung đột dữ liệu trong pipeline khi thực hiện đoạn chương trình sau:

- (1) ADD R4, R4, #300 ; $R4 \leq R4 + 300$
- (2) ADD R1, R1, R3 ; $R1 \leq R1 + R3$
- (3) SUB R5, #2000 ; $R5 \leq R5 + 2000$
- (4) SUB R1, R1, #100 ; $R1 \leq R1 - 100$

Bài làm:

Cơ chế ống lệnh (pipeline) của CPU thường gặp phải những vấn đề sau:

- **Xung đột tài nguyên:**
 - + Xung đột truy cập bộ nhớ
 - + Xung đột truy cập thanh ghi
- **Xung đột/ tranh chấp dữ liệu (Data Hazard)**
 - + Vấn đề read after write hazard
- **Các câu lệnh rẽ nhánh**
 - + Không điều kiện
 - + Có điều kiện
 - + Gọi thực hiện và trở về từ chương trình con

Hướng giải quyết xung đột dữ liệu trong pipeline khi thực hiện đoạn chương trình sau: làm chậm chu kì (T)

- Khi thực hiện ống lệnh ta gặp xung đột/ tranh chấp về dữ liệu khi thực hiện chương trình
- Ta tìm ra những câu lệnh phụ thuộc (những câu lệnh có khả năng gây ra xung đột)
- Chèn những câu lệnh rỗng NO-OP vào giữa những câu lệnh phụ thuộc
- Thay đổi trật tự của các câu lệnh không phụ thuộc

Các câu lệnh phụ thuộc:

(4) -> (2)

Viết lại

(2)	IF	ID	MEM	EX	WB				
(1)		IF	ID	MEM	EX	WB			
(3)			IF	ID	MEM	EX	WB		
NO-OP				NO					
(4)					IF	ID	MEM	EX	WB

Bài 3 (D20 – Đề 1): Cho đoạn chương trình sau (R1, R2 là các thanh ghi và lệnh quy ước theo dạng LỆNH <ĐÍCH> <GỐC>):

- (1) MOVE R0, #400;
- (2) LOAD R1, #2000
- (3) STORE (R1), R0
- (4) SUBTRACT R0, #20
- (5) ADD 2000, #10
- (6) ADD R0, (R1)

1. Nêu ý nghĩa của từng lệnh và xác định giá trị R0 sau khi thực hiện xong lệnh số (6)
2. Nêu một hướng giải quyết xung đột dữ liệu trong pipeline khi thực hiện đoạn chương trình trên

Bài làm:

1. Ý nghĩa từng lệnh:
 - (1) MOVE R0, #400 ; R0 <= 400
 - (2) LOAD R1, #2000 ; R1 <= 2000
 - (3) STORE (R1), R0 ; M[R1] <= R0 .Note: địa chỉ ô nhớ
 - (4) SUBSTRACT R0, #20 ; R0 <= R0 - 20
 - (5) ADD 2000, #10 ; M[2000] <= M[2000] + 10
 - (6) ADD R0, (R1) ; R0 <= R0 + M[R1]
2. Hướng giải quyết xung đột dữ liệu trong pipeline khi thực hiện đoạn chương trình: Làm chậm chu kỳ T
 - Khi thực hiện ống lệnh ta gặp xung đột/ tranh chấp về dữ liệu khi thực hiện chương trình
 - Ta tìm ra những câu lệnh phụ thuộc (những câu lệnh có khả năng gây ra xung đột)
 - Chèn những câu lệnh rỗng NO-OP vào giữa những câu lệnh phụ thuộc
 - Thay đổi trật tự của các câu lệnh không phụ thuộc

Các lệnh phụ thuộc:

(6) -> (5), (4), (3)

(5) -> (3) -> (1), (2)

(4) -> (1)

Viết lại

(1)	IF	ID	MEM	EX	WB												
(2)		IF	ID	MEM	EX	WB											
NO-OP			NO														
NO-OP				NO													
(4)					IF	ID	MEM	EX	WB								
(3)						IF	ID	MEM	EX	WB							
NO-OP							NO										
NO-OP								NO									
NO-OP									NO								
(5)										IF	ID	MEM	EX	WB			
NO-OP											NO						
NO-OP												NO					
NO-OP													NO				
(6)														IF	ID	MEM	EX WB

Bài 4 (D20 – Đề 2): Cho đoạn chương trình sau (R1, R2 là các thanh ghi và lệnh quy ước theo dạng LỆNH <ĐÍCH> <GÓC>):

- (1) STORE -100(R2), R1
- (2) LOAD R1, (00FF)

- (3) COMPARE R3, R4
- (4) JUMP-IF-EQUAL Label
- (5) ADD R3, R4
- (6) ADD R2, 2
- (7) Label:

1. Xác định chế độ địa chỉ và ý nghĩa của từng lệnh;
2. Nêu hướng giải quyết xung đột dữ liệu trong pipeline khi thực hiện đoạn chương trình trên biết mỗi lệnh được chia thành 5 giai đoạn.
3. Giả thiết R3 != R4 và mỗi giai đoạn thực hiện lệnh đều thực hiện trong thời gian là 0.1ns, so sánh thời gian CPU chạy hết 6 lệnh đầu tiên trong trường hợp không sử dụng cơ chế pipeline và có sử dụng cơ chế pipeline trong ý 2.

Bài làm:

1. Ý nghĩa từng lệnh:

- (1) STORE -100(R2), R1 ; M[R2-100] <= R1 dịch chuyển gián tiếp bộ nhớ
- (2) LOAD R1, (00FF) ; R1 <= M[00FF]
- (3) COMPARE R3, R4 ; so sánh R3 và R4
- (4) JUMP-IF-EQUA ; Nếu R3=R4 thì nhảy đến Label
- (5) ADD R3, R4 ; R3 <= R3 + R4 trực tiếp
- (6) ADD R2, 2 ; R2 <= R2 + M[2] trực tiếp
- (7) Label: ; Nhãn

2. Hướng giải quyết xung đột dữ liệu trong pipeline khi thực hiện đoạn chương trình: Làm chậm quá trình rẽ nhánh
 - Khi thực hiện ống lệnh ta gặp xung đột/ tranh chấp về dữ liệu khi thực hiện chương trình
 - Chèn 1 câu lệnh rỗng NO-OP vào sau câu lệnh rẽ nhánh hoặc đưa 1 câu lệnh độc lập ở phía trên vào sau câu lệnh rẽ nhánh

Viết lại

(1)	IF	ID	MEM	EX	WB						
(3)		IF	ID	MEM	EX	WB					
(4)			IF	ID	MEM	EX	WB				
(2)				IF	ID	MEM	EX	WB			
(5)					IF	ID	MEM	EX	WB		
(6)						IF	ID	MEM	EX	WB	

3. Tính (Nhưng mà theo mình hiểu thôi nhé :v)

Thời gian để chạy 6 câu lệnh khi không sử dụng cơ chế ống lệnh

$$T1 = 6 * 0.1 = 0.6 \text{ ns}$$

Thời gian để chạy 6 câu lệnh khi sử dụng cơ chế ống lệnh

$$T2 = 0.1 + (6-1) * \frac{0.1}{5} = 0.2 \text{ ns} \quad (5: \text{số giai đoạn thực hiện 1 câu lệnh})$$

Bài 5: D20 – Đề 5: Cho đoạn chương trình sau (R1, R2 là các thanh ghi và lệnh quy ước theo dạng LỆNH <ĐÍCH> <GỐC>):

- (1) LOAD R2, #400

- (2) LOAD R1, #1200
- (3) STORE (R1), R2
- (4) SUBTRACT R2, #20
- (5) ADD 1200, #10
- (6) ADD R2, (R1)

1. Nêu ý nghĩa của từng lệnh;
2. Xác định giá trị của thanh ghi R2 sau khi thực hiện xong lệnh số (6)
3. Nêu một hướng giải quyết xung đột dữ liệu trong pipeline khi thực hiện đoạn chương trình trên

Bài làm:

1. Ý nghĩa của từng lệnh

- (1) LOAD R2, #400 ; $R2 \leq R2 + 400$
- (2) LOAD R1, #1200 ; $R1 \leq R1 + 1200$
- (3) STORE (R1), R2 ; $M[R1] \leq R2$
- (4) SUBTRACT R2, #20 ; $R2 \leq R2 - 20$
- (5) ADD 1200, #10 ; $M[1200] \leq M[1200] + 10$
- (6) ADD R2, (R1) ; $R2 \leq R2 + M[R1]$

2. (1) $R2 = 400$
- (2) $R1 = 1200$
- (3) $[1200] = 400$
- (4) $R2 = 400 - 20 = 380$
- (5) $[1200] + 10 = 400 + 10 = 410$
- (6) $R2 = [1200] + R2 = 410 + 380 = 790$

3. Hướng giải quyết xung đột dữ liệu trong pipeline khi thực hiện đoạn chương trình: Làm chậm chu kỳ T

- Khi thực hiện ống lệnh ta gặp xung đột/ tranh chấp về dữ liệu khi thực hiện chương trình
- Ta tìm ra những câu lệnh phụ thuộc (những câu lệnh có khả năng gây ra xung đột)
- Chèn những câu lệnh rỗng NO-OP vào giữa những câu lệnh phụ thuộc
- Thay đổi trật tự của các câu lệnh không phụ thuộc

Các lệnh phụ thuộc:

- (3) -> (1), (2)
- (4) -> (1)
- (5) -> (3)
- (6) -> (4), (5)

Viết lại:

(1)	IF	ID	ME M	EX	W B													
(2)		IF	ID	ME M	EX	W B												
NO-OP			NO															
NO-OP				NO														
(4)					IF	ID	ME M	EX	W B									
(3)						IF	ID	ME M	EX	W B								

NO-OP								NO										
NO-OP									NO									
NO-OP										NO								
(5)										IF	ID	ME M	EX	W B				
NO-OP											NO							
NO-OP												NO						
NO-OP													NO					
(6)														IF	ID	ME M	EX	W B

Dạng bài tập 2: BỘ NHỚ CACHE (Câu 2)

Ánh xạ trực tiếp

- Trường địa chỉ của ánh xạ trực tiếp

Tag	Line	Word
-----	------	------

- ▶ $Tag + Line + Word = Address_Bus_Size \text{ (bit)} = \log_2 \frac{M_{Memory}}{[Cell]}$
- ▶ Số trang nhớ của bộ nhớ: $P = \frac{M_{Memory}}{M_{Cache}} \Rightarrow tag = \log_2 P$
- ▶ Số dòng (line) trong một trang: $L = \frac{M_{Cache}}{[Word]} \Rightarrow line = \log_2 L$
- ▶ Số ô nhớ trong một dòng (line): $W = \frac{[Word]}{[Cell]} \Rightarrow Word = \log_2 W$

Ánh xạ kết hợp đầy đủ

- Trường địa chỉ của ánh xạ đầy đủ

Tag	Word
-----	------

- ▶ $Tag + Word = Address_Bus_Size \text{ (bit)} = \log_2 \frac{M_{Memory}}{[Cell]}$
- ▶ Số dòng (line) của bộ nhớ: $P = \frac{M_{Memory}}{[Word]} \Rightarrow tag = \log_2 P$
- ▶ Số ô nhớ trong một dòng (line): $W = \frac{[Word]}{[Cell]} \Rightarrow Word = \log_2 W$

Ánh xạ tập kết hợp đầy đủ

- Trường địa chỉ của ánh xạ tập kết hợp

Tag	Set	Word
-----	-----	------

- ▶ $Tag + Set + Word = Address_Bus_Size \text{ (bit)} = \log_2 \frac{M_{Memory}}{[Cell]}$
- ▶ Số trang nhớ của bộ nhớ: $P = \frac{M_{Memory} \times [Way]}{M_{Cache}} \Rightarrow tag = \log_2 P$
- ▶ Số dòng (line) trong một trang: $L = \frac{M_{Cache}}{[Word] \times [Way]} \Rightarrow Set = \log_2 L$
- ▶ Số ô nhớ trong một dòng (line): $W = \frac{[Word]}{[Cell]} \Rightarrow Word = \log_2 W$

Bài tập:

Bài 1: Đề 3 – D20: Câu 2 (2 điểm): Cho máy có dung lượng bộ nhớ chính: 64KB, dòng cache 8 Bytes, bộ nhớ cache được gồm 32 dòng (lines) được tổ chức ánh xạ trực tiếp.

1. Xác định số bit các thành phần địa chỉ của ô nhớ.
2. Ô nhớ có địa chỉ 0D20H trong bộ nhớ chính được nạp vào dòng (lines) nào của cache

Bài làm:

1. Input: Dung lượng bộ nhớ = 64KB
kích thước dòng cache = 8Bytes
dung lượng Word = 8bit = 1B (đề bài k cho mặc định là bằng 1B)
có 32 dòng Lines

Giải:

Mmemory = 64KB = $2^6 \times 10^3 \text{ B} = 2^{16} \text{ B}$

$\Rightarrow Tag + Line + Word = 16 \text{ bit}$

Số ô nhớ trong một dòng: $W = 8 \text{ B} = 2^3 \text{ B}$

$\Rightarrow Word = \log_2(W) = \log_2(8) = 3 \text{ bit}$

Bộ nhớ cache gồm 32 dòng $\Rightarrow L = 32 = 2^5$

$\Rightarrow line = \log_2(32) = 5 \text{ bit}$

Số trang của bộ nhớ là:

$\Rightarrow tag = 16 - line - word = 16 - 5 - 3 = 8 \text{ bit}$

2. Ô nhớ có địa chỉ 0D20H đổi sang hệ nhị phân:

0D20H = **0000 1101 0010 0000**B

Địa chỉ của ô nhớ 0D20H trong bộ nhớ chính:

tag: 5

line: 4

word: 0

Vậy ô nhớ 0D20H trong bộ nhớ chính được nạp vào dòng 4

Bài 2: Câu 3 -Đề 2 - 2018: Giả sử kiến trúc máy tính 32 bit. Tính kích thước dung lượng của bộ nhớ cache có thể chứa 64Kbyte data, và mỗi khối trong bộ nhớ chứa 8 word data với phương pháp ánh xạ sau:

1. Ánh xạ trực tiếp
2. Ánh xạ tập kết hợp 4 đường (4 way set-associative)
3. So sánh hai phương pháp trên?

Bài làm:

Input: kiến trúc máy tính 32bit \Rightarrow số bit cần để địa chỉ một ô nhớ trong kiến trúc này là 32 bits

Dung lượng Cache = 64Kbyte = 2^{16} B

Mỗi khối chứa 8 Word data, Dung lượng Word = 32 bit = 4 byte

Giải:

1. Ánh xạ trực tiếp

Mỗi khối chứa 8 word data và mỗi word = 32 bit = 4 byte

$\Rightarrow W = 8 \times 4 = 32$ byte khi đó word = $\log_2(32) = 5$ bit

Số dòng trong một trang: $L = 2^{16}/(8 \times 4) = 2^{11}$

\Rightarrow line = $\log_2(2^{11}) = 11$ bit

Ta có tag = $32 - \text{line} - \text{word} = 32 - 11 - 5 = 16$ bit

How many total bits are needed for a direct-mapped cache with 64 KBytes of data and 8 word blocks, assuming a 32-bit address?

- 64 Kbytes = 2^{14} words = $(2^{14})/8 = 2^{11}$ blocks
- block size = 32 bytes \Rightarrow offset size = 5 bits,
- #sets = #blocks = $2^{11} \Rightarrow$ index size = 11 bits
- tag size = address size - index size - offset size = $32 - 11 - 5 = 16$ bits
- bits/block = data bits + tag bits + valid bit = $8 \times 32 + 16 + 1 = 273$ bits
- bits in cache = #blocks x bits/block = $2^{11} \times 273 = 68.25$ Kbytes

Computer Architecture 113

Increase block size \Rightarrow decrease bits in cache

Được quét bằng CamScanner

Bài 3: Giả thiết rằng máy tính có 128KiB cache tổ chức theo kiểu ánh xạ liên kết tập hợp 4 way. Cache có tất cả là 1024 Set từ S0 đến S1023. Địa chỉ bộ nhớ chính là 32 bit và đánh địa chỉ cho từng byte.

1. Tính số bit cho các trường địa chỉ khi truy nhập cache ?
2. Xác định byte nhớ có địa chỉ D202023A(H) được ánh xạ vào Set nào của cache ?

Giải:

- Số dòng cache là: 1024

$\Rightarrow \text{set} = \log_2(1024) = 10$ bit

mặt khác:

$L = \text{Mcache} / ([\text{word}] * [\text{way}]) \Rightarrow [\text{word}] = \text{Mcache} / ([L * [\text{way}]) = 128\text{KB} / (1024 * 4) = 2^5$ byte

vậy word = 5 bit

- tag = $32 - \text{set} - \text{word} = 32 - 10 - 5 = 17$ bit

2. Đổi ô nhớ có địa chỉ D202023A(H) sang hệ nhị phân:

D202023A(H) = 1101 0010 0000 0010 0000 0010 0011 1010B

Khi đó ta có ô nhớ địa chỉ D202023A(H) được ánh xạ vào set thứ 17 hay S16 của cache

Bài 4: Máy tính dùng 32 bit địa chỉ để đánh địa chỉ cho bộ nhớ theo byte; bus dữ liệu để kết nối với bộ nhớ chính là 32 bit. Hãy cho biết:

1. Số byte nhớ tối đa được đánh địa chỉ? Địa chỉ đầu và địa chỉ cuối dưới dạng Hexa?

- Hãy cho biết các byte nhớ có địa chỉ sau đây 0FE12C3D(H), 10ABCD06(H) được bố trí ở dòng nhớ (line) nào?

Giải: (Giả sử ý 2 sử dụng ánh xạ kết hợp đầy đủ)

- Số byte nhớ tối đa được đánh địa chỉ là 2^{32} byte
Địa chỉ đầu 00000000(H) và địa chỉ cuối là FFFFFFFF(H)
- Ta có kích thước của ô nhớ trong 1 dòng là 32bit = 8byte
 $\Rightarrow \text{word} = \log_2(8) = 3 \text{ bit}$
Số dòng của bộ nhớ là: $\text{tag} = 32 - 3 = 29 \text{ bit}$
Khi đó
 - Địa chỉ: 0FE12C3D(H) = **0000 1111 1110 0001 0010 1100 0011 1101**B
 \Rightarrow được bố trí ở dòng nhớ thứ $2^0 + 2^1 + 2^2 + 2^7 + 2^8 + 2^{10} + 2^{15} + 2^{16} + 2^{17} + 2^{18} + 2^{19} + 2^{20} + 2^{21}$
 - Địa chỉ: 10ABCD06(H) = **0001 0000 1010 1011 1100 1101 000 0110**

Bài 5: Cho máy tính với 64Kbytes bộ nhớ chính được đánh địa chỉ theo byte, bộ nhớ cache gồm 32 lines được tổ chức ánh xạ trực tiếp, kích thước mỗi line là 8 bytes.

- Xác định số bit của các trường địa chỉ: Tag, Line, Word
- Chỉ ra mỗi byte nhớ của bộ nhớ chính có địa chỉ cho dưới đây được nạp vào line nào của cache:

0001 0001 0001 1011
1100 0011 0011 0100
1101 0000 1101 1101
1010 1010 1010 1010

- Giả thiết byte nhớ có địa chỉ 0001 1010 0001 1010 được nạp vào cache, hãy chỉ ra địa chỉ theo dạng nhị phân của những byte nhớ khác cùng được nạp với byte nhớ đó trong cùng line.

Giải:

- Xác định số bit
 $\text{Mmemory} = 64\text{KB} = 2^{16}\text{B}$
 $\text{Mcache} = 32 \times 8 = 2^5 \times 2^3 = 2^8 \text{ B}$
 - Kích thước mỗi line là 8 byte: $W = 8\text{byte} = 2^3 \text{ byte}$
 $\Rightarrow \text{word} = \log_2(8) = 3 \text{ bit}$
 - Bộ nhớ cache gồm 32 line: $L = 32$
 $\Rightarrow \text{line} = \log_2(32) = 5 \text{ bit}$
 - Số trang trong bộ nhớ: $P = 2^{16}/2^8 = 2^8$
 $\Rightarrow \text{tag} = \log_2(2^8) = 8 \text{ bit}$
- Ta có tag (8 bit) + line (5 bit) + word (3 bit)

0001 0001 **0001** 1011 \Rightarrow nạp vào line thứ 3
1100 0011 **0011** 0100 \Rightarrow nạp vào line thứ 6
1101 0000 **1101** 1101 \Rightarrow nạp vào line thứ 27
1010 1010 **1010** 1010 \Rightarrow nạp vào line thứ 21

- Chỉ ra các địa chỉ cùng line với địa chỉ đã cho

Ta có: địa chỉ 0001 1010 **0001** 1010 được nạp vào line thứ 3

Những địa chỉ cùng được nạp vào line với byte nhớ trên là: những byte nằm ở word 0, 1, 3, 4, 5, 6, 7

0001 1010 **0001** 1000
0001 1010 **0001** 1001
0001 1010 **0001** 1011
0001 1010 **0001** 1100
0001 1010 **0001** 1101
0001 1010 **0001** 1110

0001 1010 0001 1111

Dạng bài tập 3: ĐIỀU KHIỂN ĐÈN LED (Câu 4)

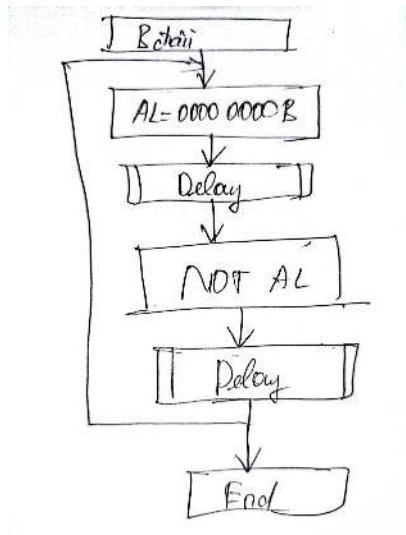
Câu 1:

.Câu 4 (3 điểm): Cho mạch điều khiển 8 đèn LED (D₀-D₇) được nối với hệ vi xử lý 8086 tại cổng ra 2C0H. Biết rằng đèn được bật sáng nếu bit điều khiển tương ứng nhận giá trị 0. Ngược lại khi bit điều khiển bằng 1 thì đèn sẽ tắt. Vẽ lưu đồ và viết chương trình hợp ngữ tạo các hiệu ứng sau:

1. Tắt cả các đèn đều cùng bật tắt liên tục với khoảng trễ giữa hai lần bật tắt là 250 chu kỳ lệnh NOP.
2. Tạo hiệu ứng một đèn sáng chạy từng bước từ trái sang phải, bắt đầu từ đèn D₅, mỗi bước dịch chuyển tương ứng với 1 đèn, với khoảng nghỉ của 1 bước dịch chuyển là 550 chu kỳ lệnh NOP.

Bài làm:

1.Code



DK_LED EQU 2C0H

LAP:

```
MOV AL, 0H
OUT DK_LED, AL
MOV CX, 250
```

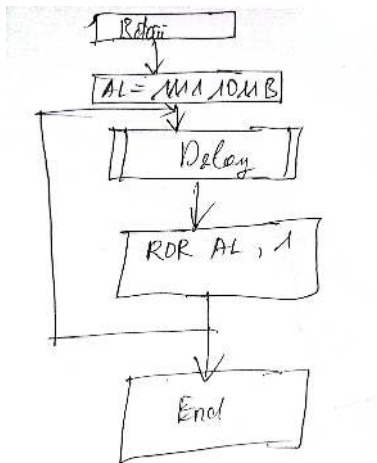
TRE1: NOP

```
LOOP TRE1
NOT AL
OUT DK_LED, AL
MOV CX, 250
```

TRE2: NOP

```
LOOP TRE2
JMP LAP
```

2.Code



Ta có:

D0 D1 D2 D3 D4 D5 D6 D7

1 1 1 1 1 0 1 1

.....

DK_LED EQU 2C0H

MOV AL, 0FBH; AL = 1111 1011B

LAP:

OUT DK_LED, AL

MOV CX, 550

TRE: NOP

LOOP TRE

ROR AL, 1

JMP LAP

Câu 2:

Câu 4 (3 điểm): Cho mạch điều khiển 8 đèn LED (D₀-D₇) được nối với hệ vi xử lý 8086 tại cổng ra 1F0H. Biết rằng đèn được bật sáng nếu bit điều khiển tương ứng nhận giá trị 0. Ngược lại khi bit điều khiển bằng 1 thì đèn sẽ tắt. Vẽ lưu đồ và viết chương trình hợp ngữ tạo các hiệu ứng sau:

1. Tắt cả các đèn đều cùng bật tắt liên tục với khoảng trễ giữa hai lần bật tắt là 150 chu kỳ lệnh NOP.
2. Tạo hiệu ứng một đèn sáng chạy từng bước từ trái sang phải, bắt đầu từ đèn D₁, mỗi bước dịch chuyển tương ứng với 1 đèn, với khoảng nghỉ của 1 bước dịch chuyển là 350 chu kỳ lệnh NOP.

1.Code

DK_LED EQU 1F0H

LAP:

MOV AL, 00H

OUT DK_LED, AL

MOV CX, 150

TRE1: NOP

LOOP TRE1

MOV AL, 0FFH

OUT DK_LED, AL

MOV CX, 150

TRE2: NOP

```

    LOOP TRE2
    JMP LAP

```

2.Code

Ta có:

D0 D1 D2 D3 D4 D5 D6 D7

1 1 0 1 1 1 1 1

1 1 1 0 1 1 1 1

....

DK_LED EQU 1F0H

MOV AL, 0DFH; AL = 1101 1111B

LAP:

 OUT DK_LED, AL

 MOV CX, 350

TRE: NOP

 LOOP TRE

 ROR AL,1

 JMP LAP

Câu 3:

Câu hỏi 3.13: Cho mạch điều khiển 8 đèn (như hình vẽ) được nối với hệ vi xử lý 8086 tại cổng ra 120H. Biết rằng đèn được bật sáng nếu bit điều khiển tương ứng nhận giá trị 1. Ngược lại khi bit điều khiển bằng 0 thì đèn sẽ tắt. Vẽ lưu đồ và viết chương trình (bằng ngôn ngữ assembly) tạo các hiệu ứng sau:

1. Tắt cả các đèn đều cùng bật tắt liên tục với khoảng trễ giữa hai lần bật tắt là 200 chu kỳ lệnh NOP.
2. Tạo hiệu ứng hai đèn kế nhau chạy liên tục từ trái sang phải, mỗi bước dịch chuyển tương ứng với 1 đèn, với khoảng nghỉ của 1 bước dịch chuyển là 150 chu kỳ lệnh NOP.

Bài làm:

1. Code

DK_LED EQU 120H

LAP: MOV AL, 0FFH

 OUT DK_LED, AL

 MOV CX, 200

TRE1: NOP

 LOOP TRE1

 MOV AL, 0H

 OUT DK_LED, AL

 MOV CX, 200

TRE2: NOP

 LOOP TRE2

 JMP LAP

2. Code

D7 D6 D5 D4 D3 D2 D1 D0

1 1 0 0 0 0 0 0

gợi ý: khởi tạo 2 đèn đầu tiên (bật 2 đèn đầu tiên)

AL = 0C0H

.....

DK_LED EQU 120H


```

MOV AL,0C0H          ;AL=0C0H
LAP:  OUT DK_LED, AL
      MOV CX, 150
TRE1: NOP
      LOOP TRE1
      ROR AL, 1      ;dịch trái, mỗi lần 1 đên
      JMP LAP

```

Dạng bài tập 4: GIẢI MÃ

Kiến thức làm bài:

- Lý thuyết cách làm của thầy Sỹ

- Dung lượng của IC nhớ (Cic) = $2^m * 8 \text{ bit} = ?\text{KB/MB?}$...
 - + IC nhớ có dung lượng 16KB -> ? đường địa chỉ (bài toán thuận)
 - > Cic = 16KB = $2^4 * 2^{10} \text{ B} = 2^{14} \text{ B}$
 - > IC nhớ có 14 đường địa chỉ
 - + IC có 13 đường địa chỉ, 8 đường dữ liệu (bài toán ngược)
 - > Cic = $2^{13} * (8 \text{ bit}) = 2^3 * [2^{10}] (\text{B}) = 8 [\text{K}](\text{B}) = 8\text{KB}$
- Thiết kế bộ giải mã địa chỉ gồm 5 bước:

- A_0-A_{m-1} : Các đường địa chỉ trực tiếp
- A_m-A_{19} : Các đường địa chỉ cao
- Input:
 - Dung lượng bộ nhớ=?
 - Dung lượng chip (IC) nhớ=?
 - Địa chỉ cơ sở của bộ nhớ=?
 - Thiết kế bằng mạch gì?
 - Cổng logic cơ bản (NAND)
 - Mạch logic tổ hợp (74138)

- + Viết tóm tắt phần Input (yêu cầu với mỗi bài)

- B1: Xác định số đường địa chỉ trực tiếp
 - Từ $C_{IC} = 2^m \text{ B} \Rightarrow$ có m đường địa chỉ trực tiếp từ A_0-A_{m-1}
- B2: Xác định số đường tín hiệu chọn chip:
 - $N_{IC} = C_{\text{Bộ nhớ}} / C_{IC \text{ nhớ}}$
 - Cần có N tín hiệu chọn chip: CS_1-CS_N
- B3: Phân giải địa chỉ cơ sở
 - Địa chỉ cơ sở (Hexa) \Rightarrow Bit nhị phân $A_{19}-A_0$
 - * Ghi chú:
- B4: Xác định dải địa chỉ của các tín hiệu chọn chip

-> ghi chú ở Bước 3: Khi phân giải địa chỉ cơ sở nếu có bit 1 trong các bit địa chỉ trực tiếp thì ta phải tăng thêm 1 IC nhớ và 1 tín hiệu chọn chip vào bộ nhớ

- B4: Xác định giải địa chỉ của các tín hiệu chọn chip
 - $CS_{1min} = (\text{Địa chỉ trực tiếp} = 0) + (\text{Địa chỉ cao} = \text{Địa chỉ cao của Địa chỉ cơ sở})$
 - $CS_{1max} = (\text{Địa chỉ trực tiếp} = 1) + (\text{Địa chỉ cao} = \text{Địa chỉ cao của } CS_{1min})$
 - $CS_{imin} = CS_{(i-1)max} + 1$
 - $CS_{imax} = (\text{Địa chỉ trực tiếp} = 1) + (\text{Địa chỉ cao} = \text{Địa chỉ cao của } CS_{imin})$

- B5: Vẽ mạch
 - Sử dụng cổng logic cơ bản (NAND)
 - Sử dụng mạch logic tổ hợp (74138)

- Tùy thuộc vào đề bài yêu cầu sử dụng mạch gì

Câu 1: Hãy thiết kế bộ giải mã địa chỉ cho bộ nhớ có dung lượng 4KB từ các IC nhớ có dung lượng 2KB, biết địa chỉ cơ sở của bộ nhớ là 58000H và được thiết kế bằng cổng logic cơ bản (NAND)

Bài làm (Thầy Sỹ):

Input:

- $C_{\text{bộ nhớ}} = 4KB$
- $C_{IC \text{ nhớ}} = 2KB$
- Địa chỉ cơ sở của IC nhớ (ĐCCS = 58000H)
- NAND

B1: Xác định số đường địa chỉ trực tiếp

- Ta có $C_{IC \text{ nhớ}} = 2KB = 2 * 2^{10}B = 2^{11}B$
-> có 11 đường địa chỉ trực tiếp: $A_0 - A_{10}$

B2: Xác định số tín hiệu chọn chip

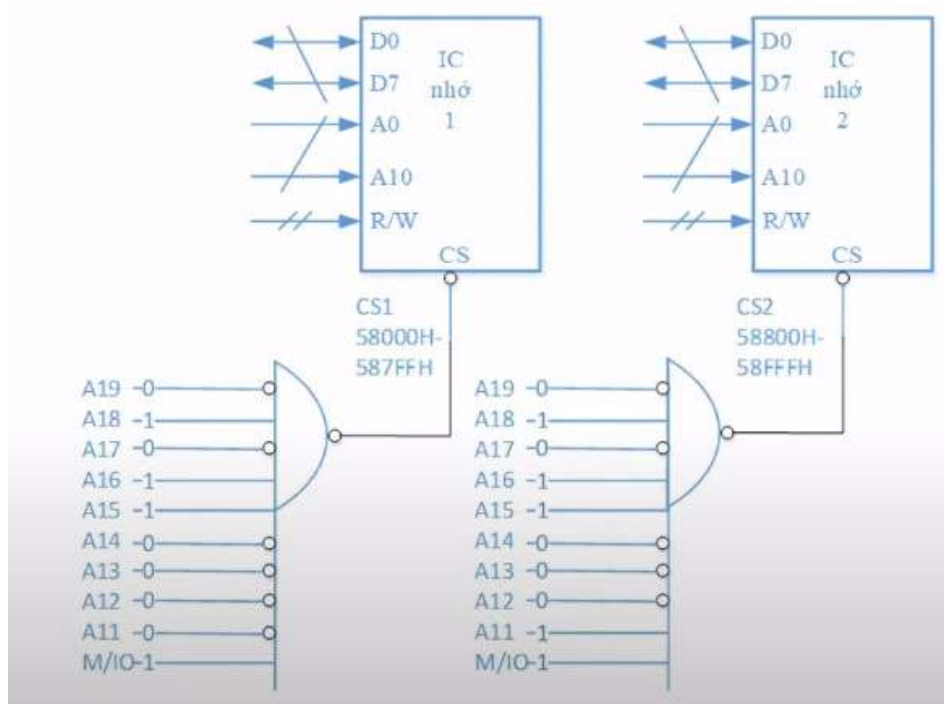
- Ta có $N_{IC} = C_{BN} / C_{IC} = 4KB / 2KB = 2$ (IC)
-> cần có 2 tín hiệu chọn chip: CS1 và CS2

B3: Phân giải địa chỉ cơ sở (căn giấy thi - viết gọn)

ĐCCS	A ₁₉	A ₁₈	A ₁₇	A ₁₆	A ₁₅	A ₁₄	A ₁₃	A ₁₂	A ₁₁	A ₁₀	A ₉	A ₈	A ₇	A ₆	A ₅	A ₄	A ₃	A ₂	A ₁	A ₀	ĐCRG (rút gọn)
58000H	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bước 4																					
CS _{1min}	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	58000H
CS _{1max}	0	1	0	1	1	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	587FFH
NHÁP																				+	
																				1	
CS _{2min}	0	1	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	58800H
CS _{2max}	0	1	0	1	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	58FFFH

Note (hiểu bài k cần ghi vào trong bài làm): các bit địa chỉ từ A₀ - A₁₀ không có bit 1 nên bài này giữ nguyên 2 tín hiệu chọn chip

B5: Vẽ



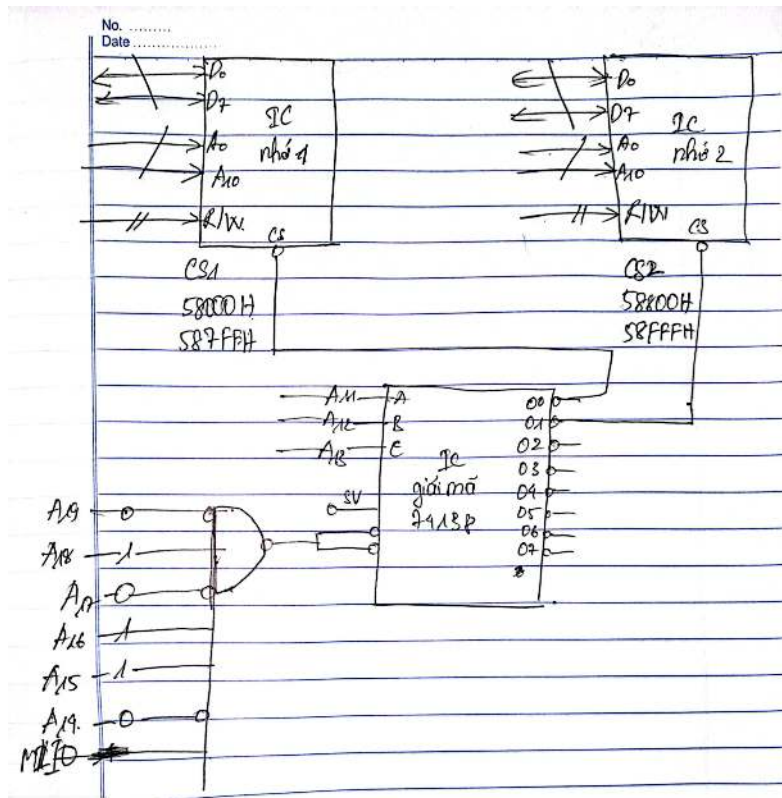
Lưu ý khi vẽ:

- không vẽ chéo, vẽ ngang mà phải kéo thẳng mặt mạch NAND xuống để vẽ
- gạch chéo ở các đường IC nhớ là thể hiện đường nối với nhau
- nguyên tắc 0 đảo, 1 nối thẳng

Câu 2: giống câu 1, nhưng sử dụng Cổng logic 74138

Bài làm:

giống câu 1 nhưng khác bước 5: Vẽ mạch



Câu 3: Hãy thiết kế bộ giải mã địa chỉ cho bộ nhớ có dung lượng 4KB từ các IC nhớ có dung lượng 2KB, biết địa chỉ cơ sở của bộ nhớ là 58080H và bộ giải mã địa chỉ thiết kế bằng các mạch logic tổ hợp và mạch logic cơ bản

Bài làm:

Input:

- C_{bộ nhớ} = 4KB
- C_{IC nhớ} = 2KB
- Địa chỉ cơ sở của IC nhớ (ĐCCS = 58080H)
- NAND

B1: Xác định số đường địa chỉ trực tiếp

- Ta có C_{IC nhớ} = 2KB = 2 * 2¹⁰B = 2¹¹B
-> có 11 đường địa chỉ trực tiếp: A₀ - A₁₀

B2: Xác định số tín hiệu chọn chip

- Ta có N_{IC} = C_{BN} / C_{IC} = 4KB / 2KB = 2 (IC)
-> cần có 2 tín hiệu chọn chip: CS₁ và CS₂

B3:

ĐCCS	A ₁₉	A ₁₈	A ₁₇	A ₁₆	A ₁₅	A ₁₄	A ₁₃	A ₁₂	A ₁₁	A ₁₀	A ₉	A ₈	A ₇	A ₆	A ₅	A ₄	A ₃	A ₂	A ₁	A ₀	ĐCRG (rút gọn)
58080H	0	1	0	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	
Bước 4																					
CS _{1min}	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	58000H
CS _{1max}	0	1	0	1	1	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	587FFH

- 74LS138

B1: Xác định số đường địa chỉ trực tiếp

- Ta có $C_{IC\text{ nhớ}} = 2KB = 2 * 2^{10}B = 2^{11}B$
-> có 11 đường địa chỉ trực tiếp: $A_0 - A_{10}$

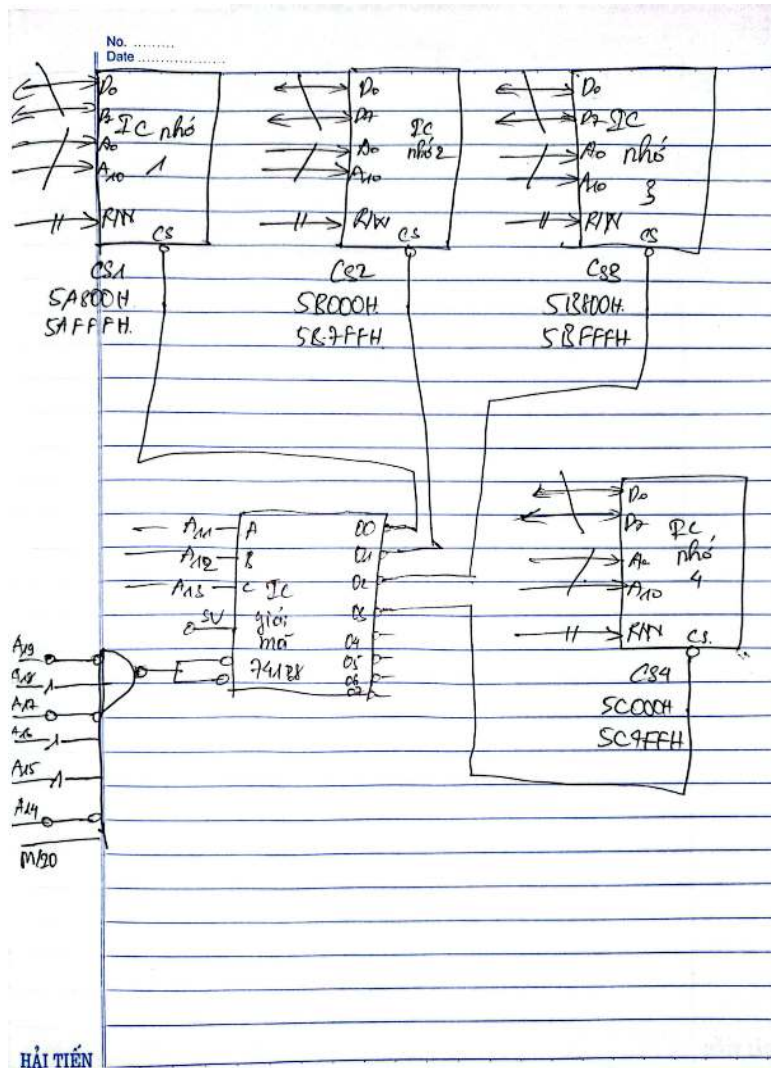
B2: Xác định số tín hiệu chọn chip

- Ta có $N_{IC} = C_{BN} / C_{IC} = 8KB / 2KB = 4 (IC)$
-> cần có 4 tín hiệu chọn chip: CS1, CS2, CS3, CS4

B3:

ĐCCS	A ₁₉	A ₁₈	A ₁₇	A ₁₆	A ₁₅	A ₁₄	A ₁₃	A ₁₂	A ₁₁	A ₁₀	A ₉	A ₈	A ₇	A ₆	A ₅	A ₄	A ₃	A ₂	A ₁	A ₀	ĐCRG (rút gọn)
5A800H	0	1	0	1	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	
Bước 4																					
CS _{1min}	0	1	0	1	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	5A800H
CS _{1max}	0	1	0	1	1	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	5AFFFH
NHẬP																				+	
																				1	
CS _{2min}	0	1	0	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	5B000H
CS _{2max}	0	1	0	1	1	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	5B7FFH
CS _{3min}	0	1	0	1	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	5B800H
CS _{3max}	0	1	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	5BFFFH
CS _{4min}	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5C000H
CS _{4max}	0	1	0	1	1	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	5C7FFH

Bước 5:



- Lý thuyết cách làm của thầy Trường

- Bước 1: Xác định số bit cho địa chỉ nội bộ chip và mạch giải mã
Số bit cho nội bộ chip là $m = \log_2 IC \Rightarrow$ số bit cần cho mạch giải mã là $20 - m$
- Bước 2: Phân giải địa chỉ cơ sở của các chip
Số lượng chip nhớ là $n =$
CI
IC \Rightarrow Dung lượng của 1 chip nhớ là IC
Một chip nhớ có Địa chỉ cuối = Địa chỉ đầu + Dung lượng chip - 1
Địa chỉ của IC 1: Từ ĐCCS đến (ĐCCS + IC - 1H)
Địa chỉ của IC 2: Từ (ĐCCS + IC) đến (ĐCCS + 2xIC - 1H)
...
- Bước 3: Lập bảng bit
- Bước 4: Vẽ hình mạch giải mã

Câu 5:(Đề D20)

Hãy xây dựng mạch giải mã địa chỉ cho 1 bộ nhớ ROM có dung lượng 64KB bằng phương pháp sử dụng mạch logic cơ bản, Biết rằng kích thước 1 vi mạch nhớ là 32KB và địa chỉ cơ sở là 3F000H

Bài làm:

CROM = 64KB

IC = 32KB

ĐCCS = 3F000H

Sử dụng mạch logic cơ bản

Ta có:

Chip nhớ IC 32KB chiếm không gian $32KB = 2^5 \cdot 2^{10} = 2^{15}$ B

=> 15 bit cho địa chỉ nội bộ chip (A0-A14)

Vi xử lý 8086 có 20 bit địa chỉ nên số bit cho mạch giải mã là:

=> $20 - 15 = 5$ bit cho mạch giả mã (A15-A19)

Phân giải địa chỉ cơ sở của các chip

Bộ nhớ ROM có dung lượng CROM = 64KB và mỗi chip nhớ có dung lượng 32KB

=> Số lượng chip nhớ 32KB là $64KB/32KB = 2$ tương ứng IC1, IC2

Đổi kích thước của một chip nhớ sang hệ 16:

$2KB = 2^{15} = 0000\ 1000\ 0000\ 0000\ 0000B = 08000H$

=> Dung lượng một chip nhớ là 08000H

Địa chỉ của một chip nhớ:

Dải địa chỉ của IC1: Từ 3F000H đến $(3F000H + 08000H - 1) = 46FFFH$

Dải địa chỉ của IC2: Từ 04000H đến $(46FFFH + 08000H - 1) = 4EFFFH$

Bảng bit từ địa chỉ cơ sở của các chip

		5 bit cho mạch giải mã (A19-A15)					15 bit địa chỉ nội bộ chip (A14-A0)														
IC1	3F000H	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
IC2	4EFFFH	0	1	0	0	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	0

Câu 6: Xây dựng mạch giải mã địa chỉ cho 1 bộ nhớ ROM có dung lượng 4KB bằng phương pháp sử dụng mạch logic cơ bản;

Biết rằng kích thước 1 vi mạch nhớ là 2Kx8 và địa chỉ cơ sở là 03800H;

Bài làm:

CROM = 4KB

IC = 2Kx8

ĐCCS = 03800H

Sử dụng mạch logic cơ bản

Ta có:

Chip nhớ IC 2Kx8 chiếm không gian $2KB = 2^1 \cdot 2^{10} = 2^{11}$ B

=> 11 bit cho địa chỉ nội bộ chip (A0-A10)

Vi xử lý 8086 có 20 bit địa chỉ nên số bit cho mạch giải mã là:

=> $20 - 11 = 9$ bit cho mạch giả mã (A11-A19)

Phân giải địa chỉ cơ sở của các chip

Bộ nhớ ROM có dung lượng CROM = 4KB và mỗi chip nhớ có dung lượng 2KB

=> Số lượng chip nhớ 2Kx8 là $4KB/2KB = 2$ tương ứng IC1, IC2

Đổi kích thước của một chip nhớ sang hệ 16:

$$2KB = 2^{11} = 0000\ 0000\ 1000\ 0000\ 0000B = 00800H$$

=> Dung lượng một chip nhớ là 00800H

Địa chỉ của một chip nhớ:

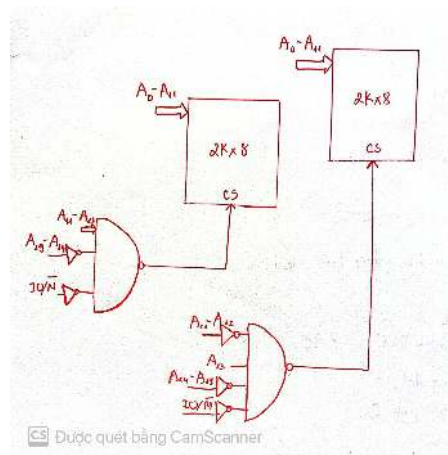
Dải địa chỉ của IC1: Từ 03800H đến $(03800H + 00800H - 1) = 03FFFH$

Dải địa chỉ của IC2: Từ 04000H đến $(04000H + 00800H - 1) = 047FFH$

Bảng bit từ địa chỉ cơ sở của các chip

		9 bit cho mạch giải mã (A19-A11)									11 bit địa chỉ nội bộ chip (A10-A0)										
IC1	03800H	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0
IC2	04000H	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

Vẽ mạch



Câu 7: Xây dựng bộ giải mã địa chỉ bộ nhớ có dung lượng 8KB có địa chỉ bắt đầu là 0F800H với các chip nhớ có dung lượng 2Kx8. Chỉ sử dụng các chip giải mã địa chỉ 74LS139 (các chip giải mã có 2 đầu vào và 4 đầu ra).

Bài làm:

Bước 1: Xác định số bit cho địa chỉ nội bộ chip và mạch giải mã

• Chip nhớ IC 2Kx8 chiếm không gian $2KB = 2^1 * 2^{10} = 2^{11}B$

=> 11 bit cho địa chỉ nội bộ chip ($A0 - A10$)

• Vì xử lý 8086 có 20 bit địa chỉ nên số bit cho mạch giải mã là:

=> $20 - 11 = 9$ bit cho mạch giải mã ($A11 - A19$)

► Bước 2: Phân giải địa chỉ cơ sở của các chip

• Bộ nhớ ROM có dung lượng CROM = 4KB và mỗi chip nhớ có dung lượng 2KB

=> Số lượng chip nhớ 2Kx8 là 8KB

2KB = 4 tương ứng IC1, IC2, IC3 và IC4

► Bước 3: Lập bảng bit từ địa chỉ cơ sở của các chip

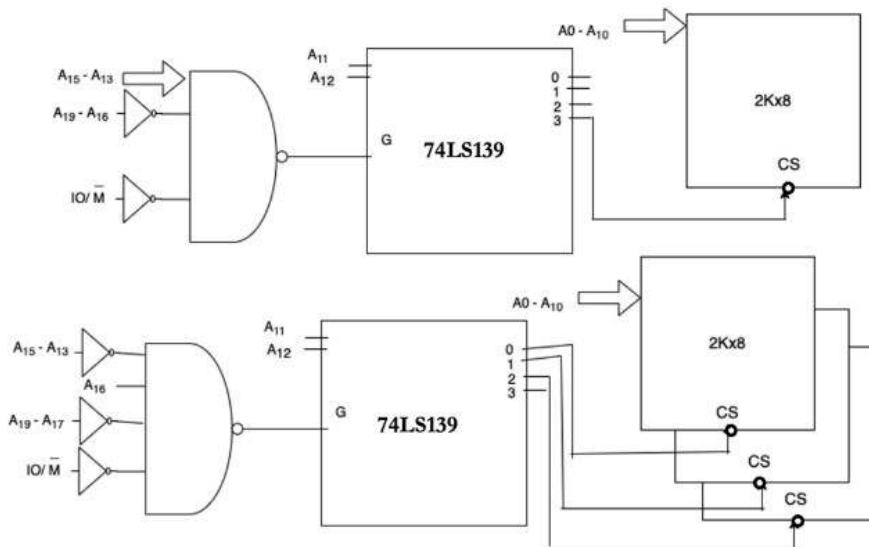
74LS139 là chip giải mã có 2 đầu vào và 4 đầu ra:

		7 bit cao							2 bit vào		11 bit địa chỉ nội bộ chip (A ₀ – A ₁₀)										
IC1	0F800H	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
IC2	10000H	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
IC3	10800H	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
IC4	11000H	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

Bước 4: Vẽ mạch giải mã địa chỉ:

Xét 7 bit cao (A₁₃ – A₁₉) cần 2 chip giải mã 74LS139:

- 1 cho IC1: 0F800H
- 1 cho IC2, IC3 và IC4: 10000H, 10800H, 11000H (7 bit giống nhau)



Câu 8: Xây dựng bộ giải mã địa chỉ bộ nhớ có dung lượng 8KB có địa chỉ bắt đầu là 0F800H với các chip nhớ có dung lượng 2Kx8. Chỉ sử dụng các chip giải mã địa chỉ 74LS138 (các chip giải mã có 3 đầu vào và 8 đầu ra).

Bài làm:

CROM = 8KB

IC = 2Kx8

ĐCCS = 0F800H

Sử dụng chip giải mã địa chỉ 74LS138

Ta có:

Chip nhớ IC 2Kx8 chiếm không gian $2KB = 2^1 * 2^{10} = 2^{11}B$

=> 11 bit cho địa chỉ nội bộ chip (A₀ - A₁₀)

Vì xử lý 8086 có 20 bit địa chỉ nên số bit cho mạch giải mã là

=> $20 - 11 = 9$ bit cho mạch giải mã (A₁₁ - A₁₉)

Bộ nhớ ROM có dung lượng CROM = 8KB và mỗi chip nhớ dung lượng 2Kx8 = 2KB

=> Số lượng chip nhớ 2Kx8 là $8KB/2KB = 4$ tương ứng IC1, IC2, IC3 và IC4

Đổi kích thước của một chip nhớ sang hệ 16:

$2KB = 2^{11}B = 0000\ 0000\ 1000\ 0000\ 0000B = 00800H$

Khi đó ta có:

Dải địa chỉ của IC1: Từ 0F800H đến $(0F800H + 00800H - 1) = 0FFFFH$

Dải địa chỉ của IC2: Từ 10000H đến $(10000H + 00800H - 1) = 107FFH$

Dải địa chỉ của IC3: Từ 10800H đến $(10800H + 00800H - 1) = 10FFFH$

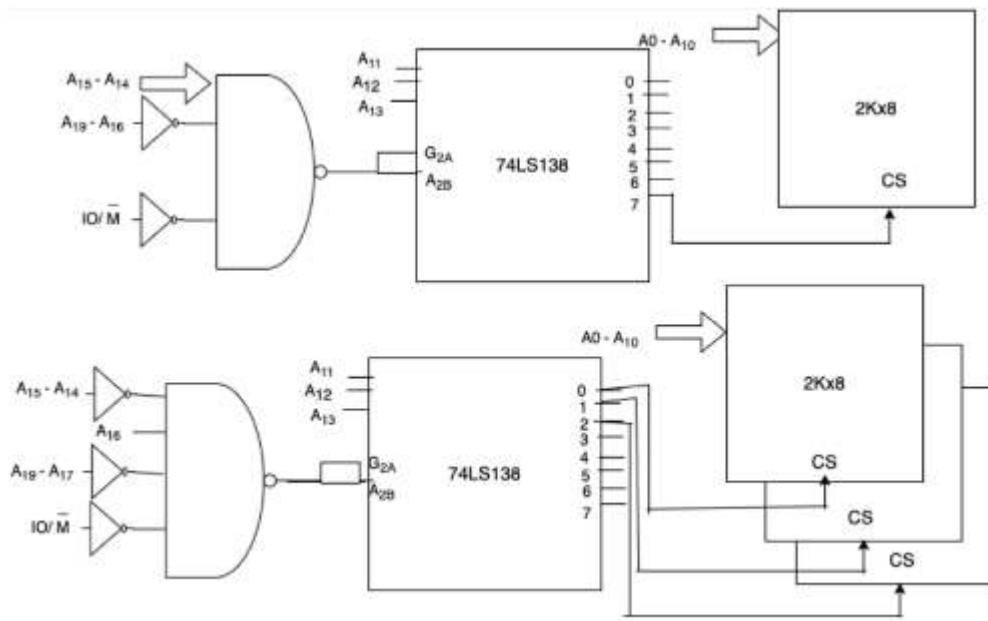
Dải địa chỉ của IC4: Từ 11000H đến $(11000H + 00800H - 1) = 117FFH$

Bảng bit từ địa chỉ cơ sở của các chip

		6 bit mạch giải mã						3 bit vào			11 bit địa chỉ nội bộ chip (A0-A10)										
IC1	0F800H	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
IC2	10000H	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
IC3	10800H	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
IC4	11000H	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

Vẽ mạch

- Xét 6 bit cao (A19-A14) ta cần 2 chip giải mã
 - 1 cho IC1: 0F800H
 - 1 cho IC2, IC3 và IC4: 10000H, 10800H, 11000H (7 bit giống nhau)



Câu 9: Hãy xây dựng mạch giải mã địa chỉ cho 1 bộ nhớ ROM có dung lượng 128KB bằng phương pháp sử dụng mạch tích hợp 78LS138; biết rằng kích thước 1 vi mạch nhớ là 32KB và địa chỉ cơ sở là 58000H

Bài làm:

CROM = 128KB

IC = 32KB

ĐCCS = 58000H

Sử dụng mạch tích hợp 78LS138

Ta có:

Chip nhớ IC chiếm không gian $32KB = 2^5 \cdot 2^{10} = 2^{15}B$

\Rightarrow 15 bit cho địa chỉ nội bộ chip (A0 - A14)

Vi xử lý 8086 có 20 bit địa chỉ nên số bit cho mạch giải mã là:

$\Rightarrow 20 - 15 = 5$ bit cho mạch giải mã (A15 - A19)

Bộ nhớ ROM có dung lượng CROM = 128KB và mỗi chip nhớ có dung lượng 32KB

=> Số lượng chip nhớ là $128KB/32KB = 4$ tương ứng IC1, IC2, IC3 và IC4

Đổi kích thước của một chip nhớ sang hệ 16:

$32KB = 2^{15} B = 0000\ 1000\ 0000\ 0000\ 0000B = 08000H$

=> Dung lượng của một chip nhớ là 08000H

Khi đó ta có:

Dải địa chỉ của IC1: Từ 58000H đến $(58000H + 08000H - 1) = 5FFFFH$

Dải địa chỉ của IC2: Từ 60000H đến $(60000H + 08000H - 1) = 67FFFH$

Dải địa chỉ của IC3: Từ 68000H đến $(68000H + 08000H - 1) = 6FFFFH$

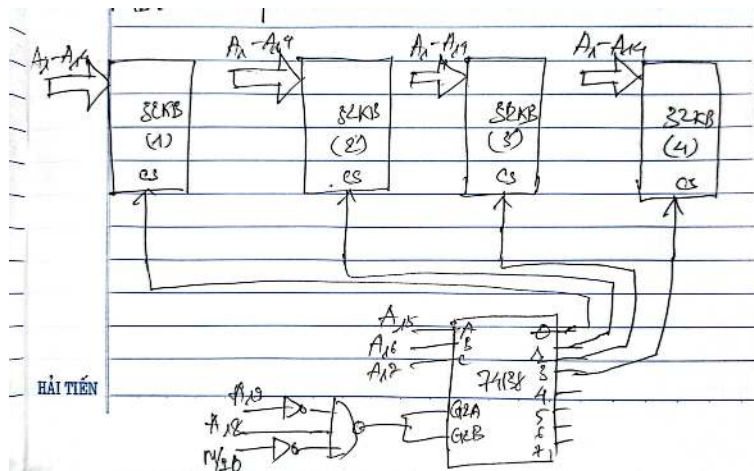
Dải địa chỉ của IC4: Từ 70000H đến $(70000H + 08000H - 1) = 77FFFH$

Bảng bit từ địa chỉ cơ sở của các chip

		D19-D18		3 bit vào			15 bit địa chỉ nội bộ chip (A0-A14)														
IC1	58000H	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
IC2	60000H	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
IC3	68000H	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
IC4	70000H	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Vẽ mạch

- Xét 2 bit cao (A18 - A19) ta cần 1 chip giải mã 74LS138 (2 bit giống nhau)



Câu 10: Hãy xây dựng mạch giải mã địa chỉ cho 1 bộ nhớ ROM có dung lượng 64KB bằng phương pháp sử dụng mạch logic cơ bản; biết rằng kích thước 1 vi mạch nhớ là 16KB và địa chỉ cơ sở là 5C000H

Bài làm:

CROM = 64KB

IC = 16KB

ĐCCS = 5C000H

Sử dụng mạch logic cơ bản

Ta có:

Chip nhớ IC chiếm không gian là $16KB = 2^4 * 2^{10} = 2^{14} B$

=> 14 bit địa chỉ nội bộ chip (A0-A13)

Vi xử lý 8086 có 20 bit địa chỉ nên số bit cho mạch giải mã là

=> $20 - 14 = 6$ bit cho mạch giải mã (A14-A19)

Bộ nhớ ROM có dung lượng CROM = 64KB và mỗi chip nhớ có dung lượng 16KB

=> Số lượng chip nhớ cần sử dụng là $64KB/16KB = 4$ tương ứng IC1, IC2, IC3 và IC4

Đổi kích thước của một chip nhớ sang hệ 16:

$16KB = 2^{14}B = 0000\ 0100\ 0000\ 0000B = 04000H$

Địa chỉ của chip nhớ

Dải địa chỉ của IC1: Từ 5C000H đến $(5C000H + 04000H - 1) = 5FFFFH$

Dải địa chỉ của IC2: Từ 60000H đến $(60000H + 04000H - 1) = 63FFFH$

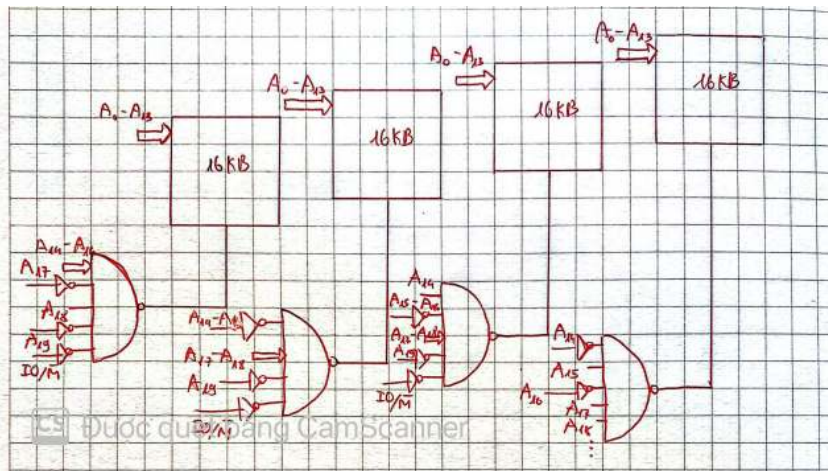
Dải địa chỉ của IC3: Từ 64000H đến $(64000H + 04000H - 1) = 67FFFH$

Dải địa chỉ của IC4: Từ 68000H đến $(68000H + 04000H - 1) = 6BFFFH$

Bảng bit địa chỉ cơ sở của các chip

		6 bit cho mạch giải mã (A19-A14)						14 bit địa chỉ nội bộ chip (A13-A0)													
IC1	5C000H	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
IC2	60000H	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
IC3	64000H	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
IC4	68000H	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Vẽ mạch



Câu 11: Hãy xây dựng mạch giải mã địa chỉ cho 1 bộ nhớ ROM có dung lượng 32KB bằng phương pháp sử dụng mạch tích hợp 74LS138; biết rằng kích thước 1 vi mạch nhớ là 8KB và địa chỉ cơ sở là 58000H

Bài làm:

CROM = 32KB

IC = 8KB

ĐCCS = 58000H

Sử dụng mạch tích hợp 74LS138

Ta có:

Chip nhớ IC chiếm không gian là $8KB = 2^3 * 2^{10} = 2^{13} B$

=> 13 bit địa chỉ nội bộ chip (A0-A12)

Vi xử lý 8086 có 20 bit địa chỉ nên số bit cho mạch giải mã là

=> $20 - 13 = 7$ bit cho mạch giải mã (A13-A19)

Bộ nhớ ROM có dung lượng CROM = 64KB và mỗi chip nhớ có dung lượng 8KB
 \Rightarrow Số lượng chip nhớ cần sử dụng là $32KB/8KB = 4$ tương ứng IC1, IC 2, IC3 và IC4

Đổi kích thước của một chip nhớ sang hệ 16:

$$16KB = 2^{13}B = 0000\ 0010\ 0000\ 0000\ 0000B = 02000H$$

Địa chỉ của chip nhớ

Dải địa chỉ của IC1: Từ 58000H đến $(58000H + 02000H - 1) = 59FFFFH$

Dải địa chỉ của IC2: Từ 5A000H đến $(5A000H + 02000H - 1) = 5BFFFFH$

Dải địa chỉ của IC3: Từ 5C000H đến $(5C000H + 02000H - 1) = 5DFFFFH$

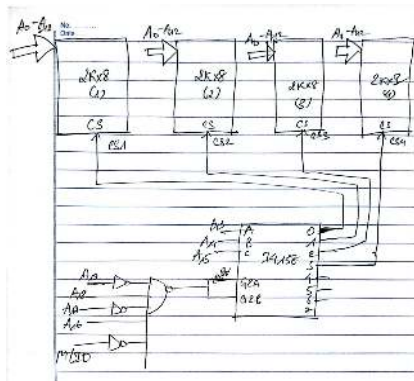
Dải địa chỉ của IC4: Từ 5E000H đến $(5E000H + 02000H - 1) = 5FFFFFH$

Bảng bit địa chỉ cơ sở của các chip

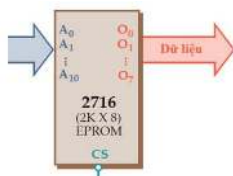
		4 bit cho mạch giải mã (A19-A16)				3 bit vào			13 bit địa chỉ nội bộ chip (A12-A0)												
IC1	58000H	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
IC2	5A000H	0	1	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
IC3	5C000H	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
IC4	5E000H	0	1	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0

Vẽ mạch

Xét 4 bit cao (A16 – A19) cần 1 chip giải mã 74LS139:



Câu 12: Xây dựng mạch giải mã địa chỉ dùng các mạch lô-gíc cơ bản cho bộ nhớ ROM dung lượng 4KB có địa chỉ cơ sở 5800H dùng vi mạch nhớ 2Kx8 (như hình vẽ)



Bài làm:

+ **Cách làm thầy Sỹ**

Input: Cbộ nhớ = 4KB

CIC = 2KB

ĐCCS = 5800H. Note: địa chỉ cơ sở không đủ 5 thêm 0 vào đầu \rightarrow ĐCCS = 05800H

Sử dụng mạch Logic cơ bản

B1: Số đường địa chỉ trực tiếp

$$C_{IC} = 2KB = 2^1 * 2^{10} (B) = 2^{11} (B)$$

-> có 11 đường địa chỉ trực tiếp (A0-A10)

Vi xử lý 8086 có 20 bit địa chỉ nên số bit cho mạch giải mã là:

$$20 - 11 = 9 (\text{bit cho mạch giải mã từ A11-A19})$$

Bước 2: Xác định số tín hiệu chọn chip

$$N_{ic} = C_{BN} / C_{IC} = 4KB / 2KB = 2 (\text{tín hiệu})$$

-> có 2 tín hiệu chọn chip IC1 và IC2

B3: Phân giải địa chỉ cơ sở (cần giấy thi - viết gọn)

ĐCCS	A 19	A 18	A 17	A 16	A 15	A 14	A 13	A 12	A 11	A 10	A 9	A 8	A 7	A 6	A 5	A 4	A 3	A 2	A 1	A 0	ĐCRG (rút gọn)
05800H	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	
Bước 4																					
CS _{1min}	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	05800H
CS _{1max}	0	0	0	0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	05FFFH
NHÁP																				+	
CS _{2min}	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	06000H
CS _{2max}	0	0	0	0	0	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	067FFH

B5: Vẽ mạch (trưng tự cách làm của thầy Trường)

+ **Cách làm thầy Trường**

$$CROM = 4KB$$

$$IC = 2K \times 8$$

$$\text{ĐCCS} = 05800H$$

Sử dụng mạch logic cơ bản

Ta có:

$$\text{Chip nhớ IC } 2K \times 8 \text{ chiếm không gian } 2KB = 2^1 * 2^{10} = 2^{11} B$$

=> 11 bit cho địa chỉ nội bộ chip (A0-A10)

Vi xử lý 8086 có 20 bit địa chỉ nên số bit cho mạch giải mã là:

$$\Rightarrow 20 - 11 = 9 \text{ bit cho mạch giả mã (A11-A19)}$$

Phân giải địa chỉ cơ sở của các chip

Bộ nhớ ROM có dung lượng CROM = 4KB và mỗi chip nhớ có dung lượng 2KB

$$\Rightarrow \text{Số lượng chip nhớ } 2K \times 8 \text{ là } 4KB / 2KB = 2 \text{ tương ứng IC1, IC2}$$

Đổi kích thước của một chip nhớ sang hệ 16:

$$2KB = 2^{11} = 0000 \ 0000 \ 1000 \ 0000 \ 0000B = 00800H$$

=> Dung lượng một chip nhớ là 00800H

Địa chỉ của một chip nhớ:

$$\text{Dải địa chỉ của IC1: Từ } 05800H \text{ đến } (05800H + 00800H - 1) = 05FFFH$$

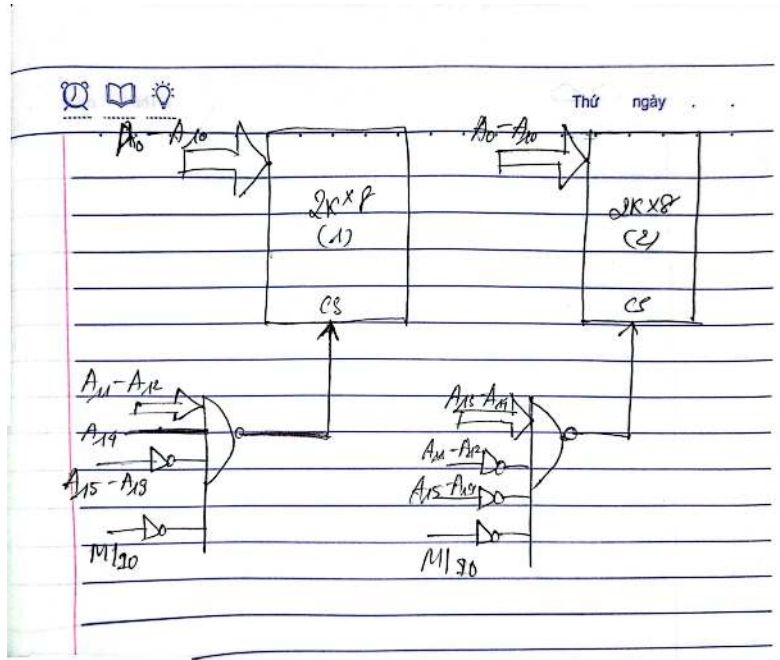
$$\text{Dải địa chỉ của IC2: Từ } 06000H \text{ đến } (06000H + 00800H - 1) = 067FFH$$

Bảng bit từ địa chỉ cơ sở của các chip

		9 bit cho mạch giải mã (A19-A11)	11 bit địa chỉ nội bộ chip (A10-A0)
--	--	----------------------------------	-------------------------------------

IC1	05800H	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0
IC2	06000H	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0

Vẽ mạch



Dạng bài tập 5: Cho đoạn code làm các yêu cầu

Câu 1: Vẽ lưu đồ và giải thích ý nghĩa của đoạn mã sau:

```
MOV CX, 80
```

```
XOR AX, AX ;      AX = 0000 0000B
```

```
XULY: ADD AX, CX
```

```
LOOP XULY
```

RA:

Đoạn mã sau dùng để tính $AX = 80 + 79 + 78 \dots + 1$

$\Rightarrow AX = (80+1)*80/2 = 81*40 = 3240$

Câu 2: Vẽ lưu đồ và giải thích ý nghĩa của đoạn mã sau:

```
CMP AX, 0
```

```
JL TIEP
```

```
MOV CL, 1
```

```
JMP THOAT
```

```
TIEP: XOR CL, CL
```

THOAT:

Ý nghĩa của đoạn mã:

Nếu $AX = 0$ thì cho $CL = 0$

ngược lại cho $CL = 1$

Câu 3: Cho đoạn chương trình hợp ngữ sau:

```
BEGIN:      ADD AX, 0;
```

```
            ADD BX, 1
```

```
LOOP:      CMP AX, CX
```

```

JG FINISH
ADD AX, BX
ADD BX, 2
JMP LOOP

```

```

FINISH: ADD DX, AX

```

Giả thiết thanh ghi CX chứa biến số nguyên n và thanh ghi DX để lưu kết quả, thanh ghi AX và BX lưu các biến a, b của chương trình.

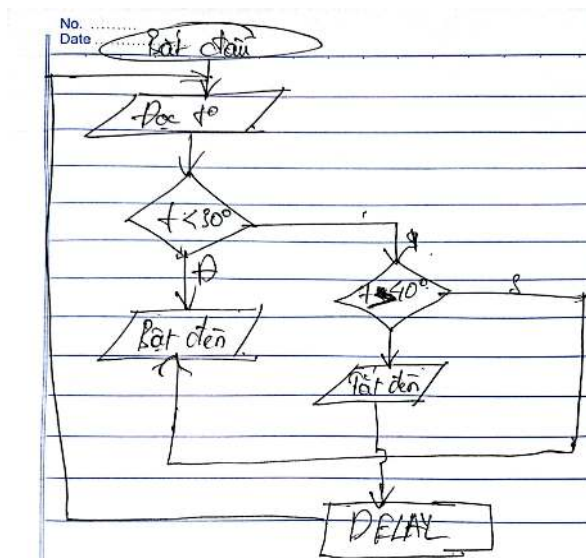
1. Vẽ lưu đồ và giải thích ý nghĩa đoạn mã hợp ngữ theo các biến a, b, n
2. Tính tổng số lệnh và giá trị thanh ghi DX dưới dạng mã Hexa khi chạy đoạn hợp ngữ trên biết CX = 0005H

Dạng bài tập 6: Điều chỉnh nhiệt độ

Ví dụ: Viết chương trình điều khiển lò ấp trứng. Yêu cầu lò phải duy trì nhiệt độ trong khoảng từ 30-40 độ, nhiệt độ mỗi trường luôn nhỏ hơn 30 độ. Biết cổng sensor nhiệt độ (cổng cảm biến đọc nhiệt độ) là 200 và cổng điều khiển đèn nhiệt là 100 với dữ liệu 1 là tắt và 0 là bật. Mỗi lần kiểm tra nhiệt độ cách nhau khoảng thời tương ứng 1000 lệnh NOP

Bài làm:

- Lưu đồ:



- Code:


```

.Model Tiny
.Stack 100h
.Code
    org 100h
    JMP Bdau
Bdau: in AL, 200 ;cổng cảm biến đọc nhiệt độ
      CMP AL,30
      JB NH30
      CMP AL, 40
      JA LH40
      JMP Xong1lan
NH30: XOR AL, AL ;AL = 0
      OUT 100, AL

```



```

        JMP Xong1lan
LH40:   MOV AL, 1    ;AL=1
        OUT 100, AL
        JMP Xong1lan

```

Xong1lan:

```

        MOV CX, 1000

```

Delay: NOP

```

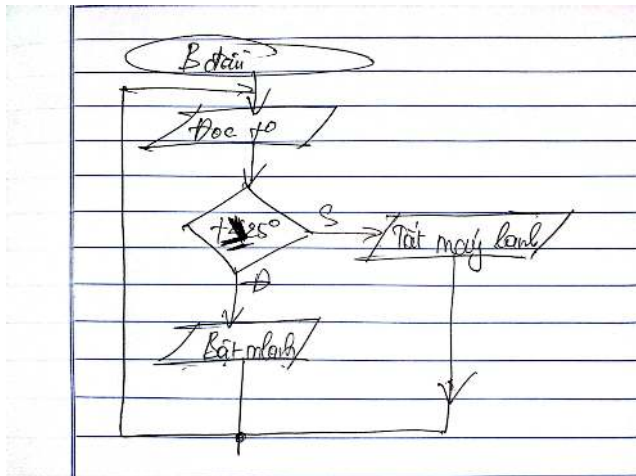
        LOOP Delay

```

Câu 1: Vẽ lưu đồ thuật toán và lập trình hợp ngữ điều khiển nhiệt độ trong phòng bằng máy lạnh để nhiệt độ luôn nhỏ hơn 25°C. Biết cổng cảm biến đọc nhiệt độ là 115H và cổng điều khiển máy lạnh là 116, với 0 là tắt và 1 là bật.

Bài làm:

- Vẽ lưu đồ



- Code:

```

.MODEL Tiny

```

```

.STACK 100h

```

```

.CODE

```

```

    org 100h

```

```

    JMP Bđau

```

Bđau:

```

    IN AL, 115H ; Đọc giá trị nhiệt độ từ cổng 115H

```

```

    CMP AL, 25

```

```

    JA LH25

```

```

    JB NH25

```

LH25:

```

    MOV AL, 1

```

```

    OUT 116, AL ; Bật máy lạnh bằng cách ghi 1 vào cổng 116

```

```

    JMP Bđau

```

NH25:

```

    XOR AL, AL

```

```

    OUT 116, AL ; Tắt máy lạnh bằng cách ghi 0 vào cổng 116

```

```

    JMP Bđau

```


END:

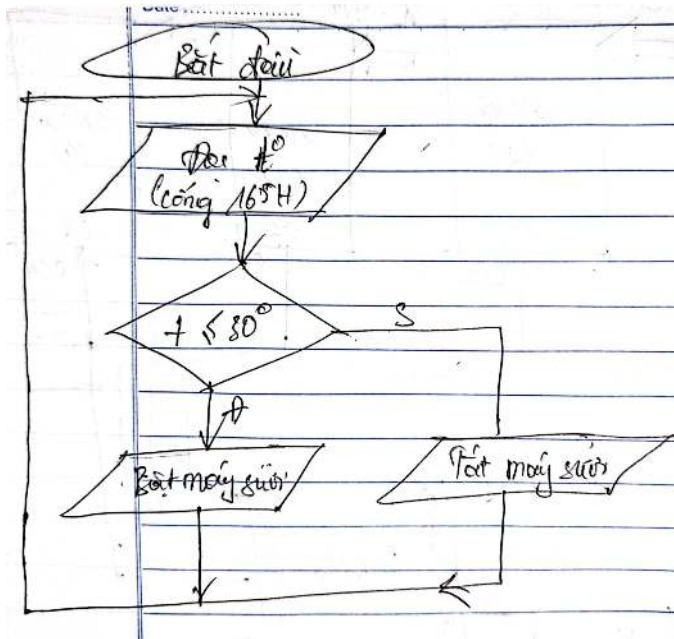
; Kết thúc chương trình

Câu 2: Vẽ lưu đồ thuật toán và lập trình hợp ngữ điều khiển nhiệt độ trong phòng bằng lò sưởi để nhiệt độ luôn lớn hơn 30°C. Biết cổng cảm biến đọc nhiệt độ là 165H và cổng điều khiển máy sưởi là 166, với 0 là tắt và 1 là bật

Bài làm:

gợi ý làm bài: nhiệt độ dưới 30 thì bật máy sưởi, nhiệt độ trên 30 thì tắt máy sưởi

- Lưu đồ thuật toán



- Code:

.Model tiny

.Stack 100h

.Code

org 100h

JMP Bđau

Bđau: IN AL, 165H ;cổng cảm biến đọc nhiệt độ

CMP AL,30

JB NH30

JA LH30

NH30: MOVE AL, 1

OUT 166, AL

JMP Bđau

LH30: XOR AL, AL

OUT 166, AL

JMP Bđau

END:

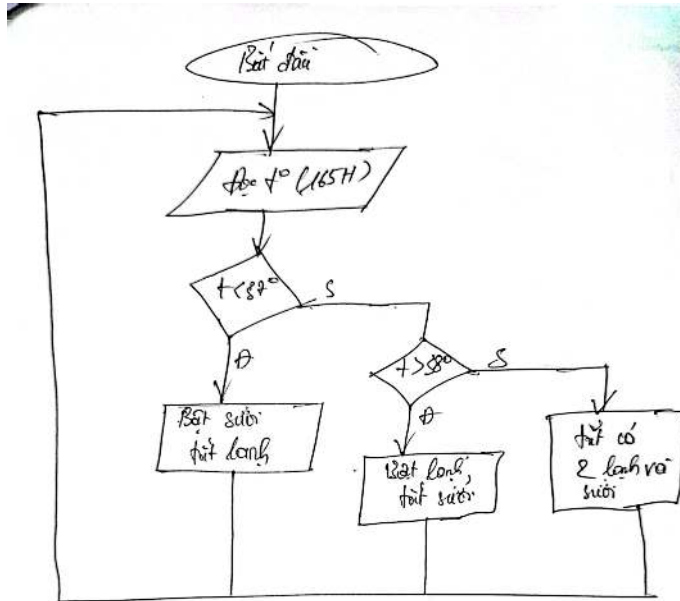
;kết thúc chương trình

Câu 3: Vẽ lưu đồ thuật toán và lập trình hợp ngữ điều khiển nhiệt độ trong lò ấp trứng luôn luôn giữ ở nhiệt độ 37°C đến 38°C. Biết cổng cảm biến đọc nhiệt độ là 165H và cổng điều khiển máy lạnh là 166, cổng điều khiển máy sưởi là 167, với 0 là tắt và 1 là bật.

Bài làm:

gợi ý làm bài: Nhiệt độ trên 38 thì bật máy lạnh (tắt máy sưởi), nhiệt độ dưới 37 thì bật máy sưởi (tắt máy lạnh), ở trong khoảng thì tắt cả lạnh và sưởi

- Lưu đồ:



- Code:

.Model Tiny

.Stack 100h

.Code

org 100h

Bdau:

; Đọc nhiệt độ từ cổng 165H

IN AL, 165H

; So sánh nhiệt độ với 37oC và 38oC

CMP AL, 37

JB LowTemp

CMP AL, 38

JA HighTemp

; Nhiệt độ nằm trong khoảng 37oC đến 38oC, tắt cả máy lạnh và máy sưởi

XOR AL, AL

OUT 166, AL

OUT 167, AL

JMP Bdau

LowTemp:

; Nhiệt độ dưới 37oC, bật máy sưởi và tắt máy lạnh

MOV AL, 1

OUT 166, AL

XOR AL, AL

OUT 167, AL

JMP Bdau

HighTemp:

; Nhiệt độ trên 38oC, bật máy lạnh và tắt máy sưởi

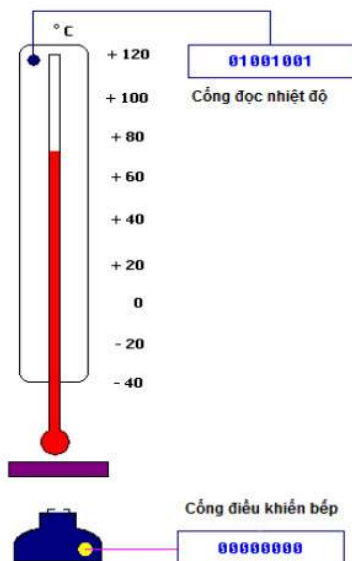
MOV AL, 1

OUT 167, AL
 XOR AL, AL
 OUT 166, AL
 JMP Bđau

END:

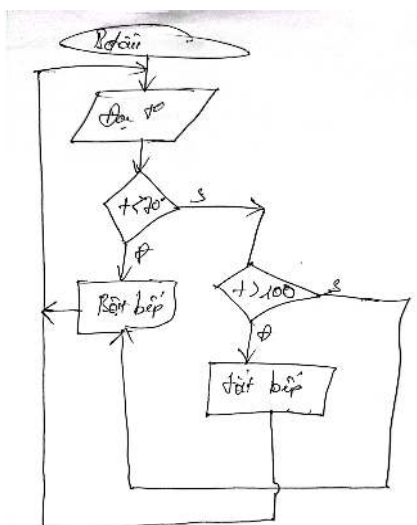
; Kết thúc chương trình

Câu 4: Vẽ lưu đồ và viết chương trình điều khiển bếp (như trong hình vẽ) làm sao cho nhiệt độ bếp luôn ổn định trong dải 70oC đến 100oC. Biết rằng hệ thống trên được nối với hệ vi xử lý 8086 trong đó Cổng đọc nhiệt độ là cổng 100H, giá trị nhiệt độ là số 8 bit có dấu tương ứng với giá trị nhiệt độ thực tế. Cổng điều khiển bếp là 105H, khi đưa giá trị 0 ra cổng thì bếp tắt còn đưa giá trị 1 thì bếp sẽ được đốt.



gợi ý làm bài: Nhiệt độ trong khoảng 70 đến 100 thì bật bếp, lớn hơn 100 thì tắt bếp, nhỏ hơn 70 thì bật bếp

- Lưu đồ thuật toán



.Model Tiny
 .Stack 100h
 .Code

org 100h

Bdau:

; Đọc nhiệt độ từ cổng 100H
IN AL, 100H
; So sánh nhiệt độ với 70oC và 100oC
CMP AL, 70
JB LowTemp
CMP AL, 100
JA HighTemp
; Nhiệt độ nằm trong khoảng 70oC đến 100oC, bật bếp
MOV AL, 1
OUT 105H, AL
JMP Bdau

LowTemp:

; Nhiệt độ dưới 70oC, tắt bếp
MOV AL, 0
OUT 105H, AL
JMP Bdau

HighTemp:

; Nhiệt độ trên hoặc bằng 100oC, bật bếp
MOV AL, 1
OUT 105H, AL
JMP Bdau

END:

; Kết thúc chương trình