Chương 2: Khối xử lí trung tâm CPU

Chương 3.1: Xử lí xen kẽ dòng mã lệnh

Yêu cầu:

- Xác định chế độ địa chỉ các dòng lệnh (?)

Chế độ địa chỉ	Ý nghĩa	Ví dụ	Thực hiện
Tức thì	Giá trị của toán hạng được chứa trong lệnh	LOAD Ri, #1000	Ri ← 1000
Trực tiếp	Địa chỉ của toán hạng được chứa trong lệnh	LOAD Ri, 1000	Ri ← M[1000]
Gián tiếp thanh ghi	Giá trị của thanh ghi trong lệnh là địa chỉ bộ nhớ chứa toán hạng	LOAD Ri, (Rj)	Ri ← M[Rj]
Gián tiếp bộ nhớ	Địa chỉ bộ nhớ trong lệnh chứa địa chỉ bộ nhớ của toán hạng	LOAD Ri, (1000)	Ri ← 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)	Ri ← 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)	Ri ← M[X+ PC]

- Ý nghĩa đoạn mã.
- Xác định giá trị thanh ghi, biến sau khi thực hiện chương trình
- Hiểu 5 giai đoạn thực hiện lệnh: IF, ID, EX, MEM và WB. Phát hiện xung đột dữ liệu và cách giải quyết (Sử dụng NO-OP hoặc đổi vị trí các lệnh)
- So sánh sử dụng và không sử dụng cơ chế ống lệnh có gì khác nhau?

Bài tập:

1. Cho đoạn lệnh sau:

```
MOVE R0, #100;          R0 <- 100
CLEAR R1;               R1 <- 0
CLEAR R2;               R2 <- 0
LAP:
    ADD R1, 2000(R2);   R1 <- R1 + M[2000 + R2]
    DECREMENT R2;       R2 <- R2 – 1
    DECREMENT R0;       R0 <- R0 – 1
    BRANCH_IF>0 LAP;    Quay lại nhãn LAP nếu R0 > 0
STORE 3000, R1;         M[3000] <- R1
```

a) Hãy giải thích ý nghĩa của từng lệnh
b) Chỉ ra chế độ địa chỉ của từng lệnh (đối với các lệnh có 2 toán hạng)
c) Đoạn lệnh trên thực hiện công việc gì?

LG

a) Đã được comment luôn trên đề.
b) Lệnh MOVE R0, #100: Chế độ địa chỉ tức thì
Lệnh ADD R1, 2000(R2): Chế độ địa chỉ chỉ số
Lệnh STORE 3000, R1: Chế độ địa chỉ trực tiếp
c) Đoạn lệnh trên sẽ cộng nội dung 100 ô nhớ từ ô 2000 đến ô 1901, sau đó lưu kết quả vào ô nhớ 3000.

2. Biết R0 = 1500, R1 = 4500, R2 = 1000, M[1500] = 3000, M[4500] = 500.

ADD R2, (R0);	R2 <- R2 + M[1500]	R2 = 4000
SUBSTRACT R2, (R1);	R2 <- R2 – M[4500]	R2 = 3500
MOVE 500(R0), R2;	M[2000] <- R2	M[2000] = 3500
LOAD R2, #5000;	R2 <- 5000	R2 = 5000
STORE 100(R2), R0;	M[5100] <- R0	M[5100] = 1500

a) Chỉ rõ chế độ địa chỉ của từng lệnh.

b) Chỉ ra giá trị của thanh ghi và tại vị trí trong bộ nhớ qua mỗi lệnh thực hiện.

LG

a) Lệnh ADD R2, (R0): Chế độ địa chỉ gián tiếp thanh ghi
Lệnh SUBSTRACT R2, (R1): Chế độ địa chỉ gián tiếp thanh ghi
Lệnh MOV 500(R0), R2: Chế độ địa chỉ chỉ số
Lệnh LOAD R2, #5000: Chế độ địa chỉ tức thì
Lệnh STORE 100(R2), R0: Chế độ địa chỉ chỉ số

b)

ADD R2, (R0);	R2 <- R2 + M[R0]	R2 = 4000
SUBSTRACT R2, (R1);	R2 <- R2 – M[R1]	R2 = 4000 – 500 = 3500
MOVE 500(R0), R2;	M[500 + R0] <- R2	M[2000] = 3500
LOAD R2, #5000;	R2 <- 5000	R2 = 5000
STORE 100(R2), R0;	M[100 + R2] <- R0	M[5100] = 1500

3. Biết rằng mỗi lệnh được chia thành 5 giai đoạn trong pipeline: Đọc lệnh (IF), giải mã & đọc toán hạng (ID), thực hiện (EX), truy nhập bộ nhớ (MEM) và lưu kết quả (WB).
Nêu 1 hướng giải quyết xung đột dữ liệu trong pipeline khi thực hiện:

a)
ADD R1, R2, R3
ADD R4, R4, #300
CMP R1, #100
SUB R5, #2000

b)
ADD R4, R4, #300
ADD R1, R1, R3
SUB R5, #2000
SUB R1, R1, #100

LG

a) Xung đột nằm ở CMP R1, #100 có thể gây sai sót do lệnh ADD R1, R2, R3 chưa cập nhật xong R1. Ta có thể chèn thêm lệnh NO-OP kết hợp sắp xếp lại các lệnh:
(1) ADD R1, R2, R3
(2) NO-OP
(3) ADD R4, R4, #300
(4) SUB R5, #2000
(5) CMP R1, #100

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

b) Xung đột nằm ở SUB R1, R1, #100 có thể gây sai sót do lệnh ADD R1, R1, R3 chưa cập nhật xong R1. Ta có thể chèn thêm lệnh NO-OP kết hợp sắp xếp lại các lệnh:
(1) ADD R1, R1, R3
(2) NO-OP
(3) SUB R5, #2000
(4) ADD R4, R4, #300
(5) SUB R1, R1, #100

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

4. Cho đoạn chương trình sau (R1, R2, R3, R4 là các thanh ghi và lệnh quy ước theo dạng LỆNH <ĐÍCH> <GỐC>):
(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:

a) 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 (IF, ID, EX, MEM và WB)

b) 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.

LG

a) Vấn đề nằm ở lệnh nhảy (4). Ta có thể chuyển các lệnh độc lập (1) và (2) từ trước tới ngay sau lệnh rẽ nhánh:
(1) COMPARE R3, R4
(2) JUMP-IF-EQUAL Label
(3) STORE –100(R2), R1
(4) LOAD R1, (00FF)
(5) ADD R3, R4
(6) ADD R2, 2
(7) Label:

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

b) Thời gian CPU chạy hết 6 lệnh đầu tiên không sử dụng pipeline:

t1 = n * 5 * 0.1 = 6 * 5 * 0.1 = 3(ns)

Thời gian CPU chạy hết 6 lệnh đầu tiên khi sử dụng pipeline:

t1 = (n + 4) * 0.1 = (6 + 4) * 0.1 = 1(ns)

5. Cho đoạn chương trình sau (R0, R1 là các thanh ghi và lệnh quy ước theo dạng LỆNH <ĐÍCH> <GỐC>):

```
(1) MOVE R0, #400          ;R0 <- 400
(2) LOAD R1, #1200         ;R1 <- 1200
(3) STORE (R1), R0         ;M[R1] <- R0
(4) SUBSTRACT R0, #20      ;R0 <- R0 - 20
(5) ADD 1200, #10          ;M[R1] <- M[R1] + 10
(6) ADD R0, (R1)           ;R0 <- M[R1]
```

a) Xác định giá trị thanh ghi R0 sau khi thực hiện xong lệnh (6)

b) 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 (IF, ID, EX, MEM và WB)

LG

a) Ta có:

(1) MOVE R0, #400;	R0 <- 400	R0 = 400
(2) LOAD R1, #1200	R1 <- 1200	R1 = 1200
(3) STORE (R1), R0	M[R1] <- R0	M[1200] = 400
(4) SUBSTRACT R0, #20	R0 <- R0 – 20	R0 = 380
(5) ADD 1200, #10	M[1200] <- M[1200]+10	M[1200] = 410
(6) ADD R0, (R1)	R0 <- R0 + M[R1]	R0 = R0 + M[1200]

=> Giá trị thanh ghi R0 sau khi thực hiện lệnh (6) là 380 + 410 = 790.

b) Xung đột tranh chấp dữ liệu ở các dòng (3), (5) và (6) khi các thanh ghi R0, R1 và giá trị trong ô nhớ chưa cập nhật đúng. Ta có thể chèn thêm các lệnh NO-OP:

- Chèn 3 NO-OP giữa (2) và (3).
- Chèn 2 NO-OP giữa (4) và (5)
- Chèn 3 NO-OP giữa (5) và (6).

6. Cho đoạn chương trình hợp ngữ sau

```
BEGIN:    ADD AX, 0      ;AX = a = 0
          ADD BX, 1      ;BX = b = 1
LOOP:     CMP AX, CX     ;so sánh AX với n
          JG FINISH      ;AX > n nhảy FINISH
          ADD AX, BX     ;AX <- AX + BX
          ADD BX, 2      ;BX <- BX + 2
          JMP LOOP
FINISH:   ADD DX, AX     ;
```

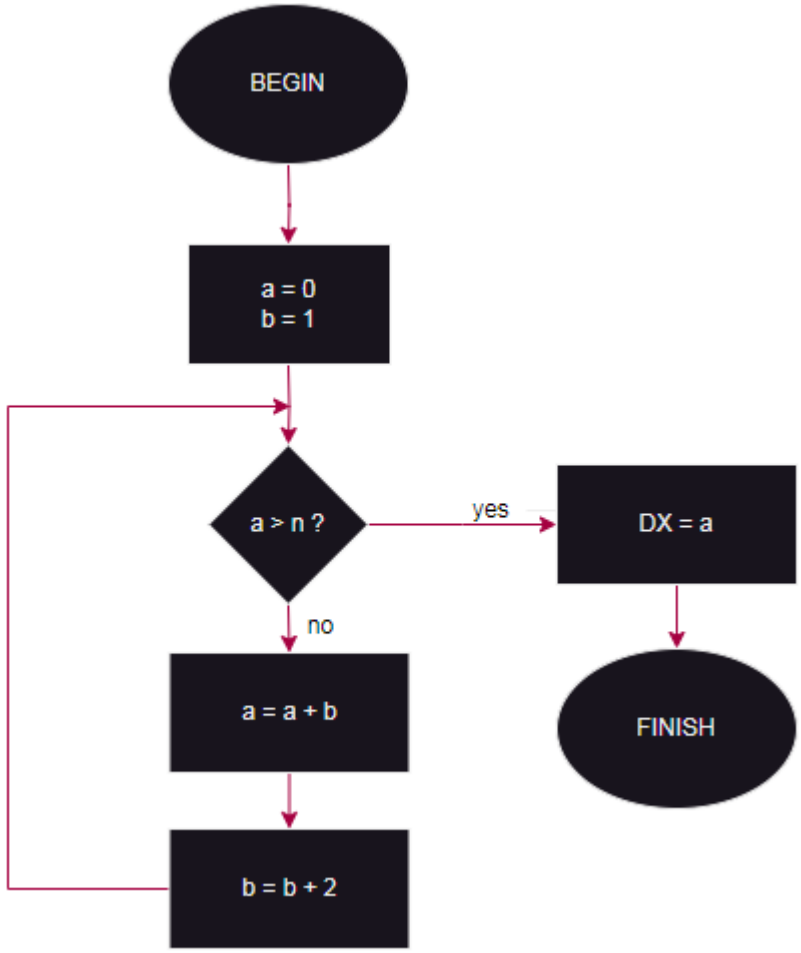
Giả thiết thanh ghi CX chứa biến số nguyên n và thanh ghi DX để lưu kết quả, thanh ghi AX và BX lưu các biến a, b của chương trình.

a) Vẽ lưu đồ và giải thích ý nghĩa đoạn mã hợp ngữ trên theo các biến a, b, n.

b) Tính tổng số lệnh và giá trị thanh ghi DX dưới dạng mã Hexa khi chạy đoạn hợp ngữ trên biết CX = 0005H.

LG

a) Lưu đồ:



Ý nghĩa đoạn mã hợp ngữ: Tính tổng các số nguyên dương lẻ bắt đầu từ 1 sao cho tổng phải lớn hơn n, giá trị tổng thu được là 1 số chính phương

b) Có CX = 0005H = 5 (hệ 10)
Bắt đầu chương trình, ta có 2 lệnh gán đầu tiên.
Với mỗi vòng lặp chương trình thực hiện, sẽ có thêm 5 lệnh nữa và khi điều kiện trong vòng lặp để thoát ra thoả mãn, ta thực hiện thêm 2 lệnh nữa.
Lặp lần 1: a = 0, b = 1 => a < 5 => a = 1, b = 3
Lặp lần 2: a = 1, b = 3 => a < 5 => a = 4, b = 5
Lặp lần 3: a = 4, b = 5 => a < 5 => a = 9, b = 7
Lặp lần 4: a = 9, b = 7 => a > 5. Thoát khỏi vòng lặp
Sau khi thoát khỏi vòng lặp ta thêm 1 lệnh gán ADD DX, AX.
Vậy tổng số lệnh đã thực hiện là: 2 + 3 * 5 + 2 + 1 = 20 (lệnh)
Giá trị thanh ghi DX: DX = 9 (hệ 10) = 0009H

Chương 3.2: Bộ nhớ Cache

Yêu cầu:

- Tính toán trường địa chỉ tương ứng với bộ nhớ trong và cache theo các ánh xạ (trực tiếp, đầy đủ và tập kết hợp).
- Xác định vị trí của 1 ô nhớ trong bộ nhớ cache hoặc trong bộ nhớ: ô nhớ nằm ở dòng nào, ở ô nào, ở trang nào trong bộ nhớ và bộ nhớ cache.
- Tính kích thước của bộ nhớ cache: tính số bit cần có để xây dựng bộ nhớ cache.

Bài tập:

1. 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.

a) Xác định số bit các thành phần địa chỉ của ô nhớ.

b) Ô nhớ có địa chỉ 0D20H trong bộ nhớ chính được nạp vào dòng (lines) nào của cache?

LG

a) Ta có: Mmemory = 64KB = 2^16B.
Kích thước 1 dòng: [Word] = 8B. Bộ nhớ cache có 32 dòng hay L = 32.
Kích thước 1 ô nhớ là 1B
=> Số ô nhớ trong 1 dòng: W = [Word]/1 = 8
=> Số bit cần thiết để xác định ô nhớ trong 1 dòng:

Word = log2(8) = 3bit.

Số bit cần thiết để xác định dòng trong 1 trang: Line = log2(32) = 5bit.
Dung lượng cache: Mcache = [Word] * L = 256B
Số trang nhớ của bộ nhớ: P = Mmemory/Mcache = 2^16/256 = 256B
=> Số bit cần thiết để xác định trang trong bộ nhớ: Tag = log2(256) = 8bit.

b) Chuyển qua nhị phân từng bit địa chỉ ô nhớ 0D20H:

0000 1101 0010 0000

8 bit đầu tiên xác định vị trí trang: 0000 1101 = 13(10)
5 bit tiếp theo xác định vị trí dòng trong 1 trang: 0010 0 = 4(10)
Vậy ô nhớ có địa chỉ 0D20H trong bộ nhớ chính được nạp vào dòng thứ 4 của cache.

2. 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:

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

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

LG

a) Máy tính dùng 32bit địa chỉ để đánh địa chỉ cho bộ nhớ theo byte.

⇨ Kích thước bộ nhớ = số byte nhớ tối đa được đánh địa chỉ = $2^{32} \times 1 = 2^{32}B$

Địa chỉ đầu dưới dạng hexa: 0000 0000(H)

Địa chỉ cuối dưới dạng hexa: FFFF FFFF(H)

b) Bus dữ liệu để kết nối với bộ nhớ chính là 32bit, khi đó:

$$Tag + Word = 32bit$$

Có 1 dòng ~ dữ liệu: 32bit = 4byte = 2^2B => Word = 2bit => Tag = 30bit

Byte nhớ có địa chỉ 0FE12C3D(H) được chuyển qua nhị phân từng bit sẽ là:

0000 1111 1110 0001 0010 1100 0011 1101

Ta có 30 bit đầu tiên để xác định vị trí (địa chỉ) của dòng (tag):

0000 1111 1110 0001 0010 1100 0011 11 = 66603791 (10)

⇨ Byte nhớ có địa chỉ 0FE12C3D(H) được bố trí ở dòng nhớ 66603791.

Ô nhớ 10ABCD06(H) được chuyển qua nhị phân từng bit sẽ là:

0001 0000 1010 1011 1100 1101 0000 0110

Ta có 30 bit đầu tiên để xác định vị trí (địa chỉ) của dòng (tag):

0001 0000 1010 1011 1100 1101 0000 01 = 69923649 (10)

⇨ Byte nhớ có địa chỉ 10ABCD06(H) được bố trí ở dòng nhớ 69923649.

3. Cho 1 máy tính có độ rộng bus dữ liệu 32bit, quản lí được bộ nhớ có dung lượng tối đa 16GB, bộ nhớ cache có dung lượng 4MB có 2 đường cache với dòng cache 16KB. Hãy xác định các thành phần địa chỉ của bộ nhớ khi sử dụng phương pháp ánh xạ tập kết hợp?

LG

Có Mmemory = 16GB = 2^34B.

Với log2(Mmemory/[Cell]) = 34bit > bus_size = 32bit

=> Tag + Set + Word = 32bit. (nếu bus_size lớn hơn thì = log2(Mmemory/[Cell])

Từ đề: Mcache = 4MB = 2^22B; [Way] = 2; [Word] = 16KB = 2^14B

Số ô nhớ trong 1 dòng là: W = [Word]/[Cell] = 2^14

=> Số bit cần thiết để xác định ô nhớ trong 1 dòng: Word = log2(W) = 14bit

Số dòng trong 1 trang: L = Mcache/([Way] * [Word]) = 2^7

=> Số bit cần thiết để xác định dòng trong 1 trang: Set = log2(L) = 7bit

=> Số bit cần thiết để xác định trang trong bộ nhớ: Tag = 32 – 14 – 7 = 11bit

4. Tổng số bit sẽ cần là bao nhiêu cho bộ nhớ cache sử dụng tập kết hợp được thiết lập 4 way kích thước 64KB dữ liệu và 1 dòng cache có kích thước 1 là 1 từ nhớ, giả sử bus địa chỉ 32bit?

LG

Ta có 1 dòng cache 32bit = 4 byte = $2^2 byte$ => $word = 2\ bit$

Số dòng Cache là

$$L = \frac{M_{Cache}}{[Word]*Way} = \frac{64Kbytes}{4byte*4} = \frac{2^{16}B}{2^4B} = 2^{12} \Rightarrow line = 12bit$$

$$tag = address\_size - line - word = 32\text{-}12\text{-}2 = 18bit$$

Số lượng bit cần thiết cho một dòng Cache là:

$$bits/block = data\ bits + tag\ bits + valid\ bit = 32 + 18 + 1 = 51bit$$

Tổng số bit cần thiết cho bộ nhớ cache là = L * [Way] * bits/block = 102KB

5. Tính tổng số bit cần thiết cho 1 bộ nhớ cache tổ chức theo ánh xạ trực tiếp kích thước 64KB và 1 dòng cache có kích thước là 8 từ nhớ, giả sử bus địa chỉ 32bit?

LG

Ta có 1 dòng cache 8x32bit = 32 byte = $2^5 byte$ => $word = 5\ bit$

Số dòng Cache là $L = \frac{M_{Cache}}{[Word]} = \frac{64Kbytes}{32byte} = \frac{2^{16}B}{2^5B} = 2^{11} \Rightarrow line = 11bit$

$tag = address\_size - line - word = 32\text{-}11\text{-}5 = 16bit$

Số lượng bit cần thiết cho một dòng Cache là:
$bits/block = data\ bits + tag\ bits + valid\ bit = 8x32 + 16 + 1 = 273bit$

Tổng số bit cần thiết cho bộ nhớ Cache là $= L \times bits/block = 2^{11} \times 273\ bit = 68.25\ Kbytes$

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

- Giải thích các chế độ địa chỉ của các câu lệnh, ý nghĩa chương trình hợp ngữ, vẽ lưu đồ thuật toán.

Chế độ địa chỉ	Nhận biết	VD	Thực hiện
Thanh ghi	Cả 2 đều là thanh ghi	MOV BX, DX	BX <- DX
Tức thì	1 là thanh ghi/ ô nhớ, 1 là hằng số	MOV AX, 0FF0H MOV [BX], 200	AX <- 0FF0H [DS:BX] <- 200
Trực tiếp	1 là thanh ghi, 1 là hằng số biểu diễn địa chỉ lệch của ô nhớ	MOV AL, [8088H] MOV [1234H], DL	AL <- DS:[8088H] DS:[1234H] <- DL
Gián tiếp thanh ghi	1 có thể là thanh ghi, 1 là 1 thanh ghi chứa địa chỉ lệch của ô nhớ	MOV AL, [BX] MOV AL, [BP]	AL <- [DS:BX] AL <- [SS:BP]
Tương đối cơ sở	1 là địa chỉ ô nhớ tạo bởi (thanh ghi cơ sở BX/BP + hằng số), 1 có thể là thanh ghi	MOV AL, [BX+100] MOV AL, [BP]+200	AL <- [DS:BX+100] AL <- [SS:BP+200]
Tương đối chỉ số	1 là địa chỉ ô nhớ tạo bởi (thanh ghi cơ sở SI/DI + hằng số), 1 có thể là thanh ghi	MOV AL, [SI+100] MOV AX, [SI+10]	AL <- [DS:SI+100] Copy nội dung 2 ô nhớ liên tiếp có địa chỉ DS:SI+10 và DS:SI+11 vào AX
Tương đối chỉ số cơ sở	1 là địa chỉ ô nhớ tạo bởi (thanh ghi BX/BP + SI/DI + hằng số), 1 có thể là thanh ghi	MOV AL, [BX+SI+100] MOV AX,[BX+SI+8]	AL <- [DS:BX+SI+100] Copy nội dung 2 ô nhớ liên tiếp có địa chỉ DS:BX+SI+8 và DS:BX+SI+9 vào AX

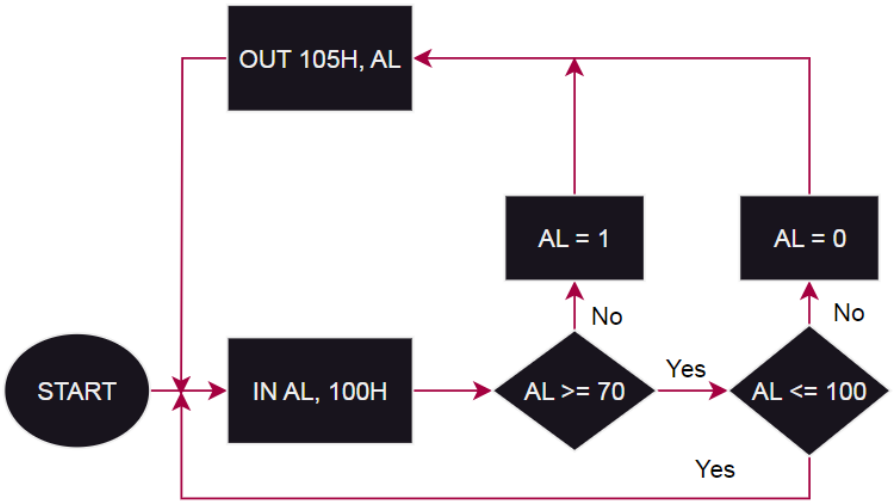
- Code 1 chương trình hoàn chỉnh: hiển thị lời chào tây ta; cộng các số; chuyển đổi các số; đảo ngược các số; viết thường viết hoa (nằm trong 20 câu hỏi BT cá nhân – chủ đạo cấu trúc rẽ nhánh, vòng lặp)
- Vẽ lưu đồ thuật toán và viết chương trình hợp ngữ điều khiển đèn sáng tắt và điều khiển nhiệt độ trong khoảng giá trị nào đấy.

Bài tập:

1. Vẽ lưu đồ và viết chương trình điều khiển bếp sao cho nhiệt độ bếp luôn ổn định trong khoảng nhiệt độ 70-100 độ C. Biết hệ thống được nối với vi xử lý 8086, trong đó: cổng đọc nhiệt độ là 100H, giá trị nhiệt độ là một 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.

LG

Lưu đồ:



Chương trình điều khiển bếp:

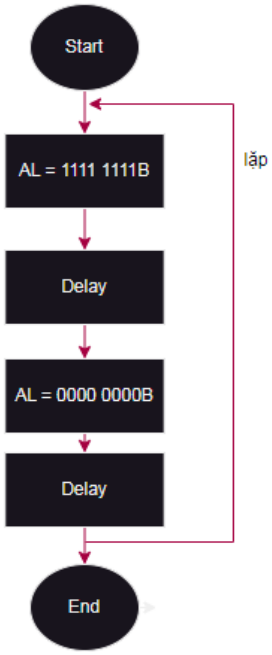
```
.Mode small
.Stack 100H
.Data
.Code
MAIN Proc
  MOV AX, @Data
  MOV DS, AX
START:
  IN AL, 100H          ;doc nhiet do hien tai
  CMP AL, 70
  JL LOW               ;nhay den low neu nhiet do < 70
  CMP AL, 100
  JG HIGH              ;nhay den high neu nhiet do > 100
  JMP START
  LOW:
      MOV AL, 1        ;bat bep
      OUT 105H, AL
      JMP START
```

```
    HIGH:
        MOV AL, 0          ;tat bep
        OUT 105H, AL
        JMP START
    MOV AH, 4CH
    INT 21H
MAIN ENDP
END MAIN
```

2. 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 bit điều khiển bằng 1 thì đèn tắt. Vẽ lưu đồ và viết chương trình hợp ngữ tạo các hiệu ứng sau:

a) Tất cả các đèn đều cùng bật tắt liên tục với khoảng trễ giữa 2 lần bật tắt là 250 chu kì lệnh NOP.

Lưu đồ:



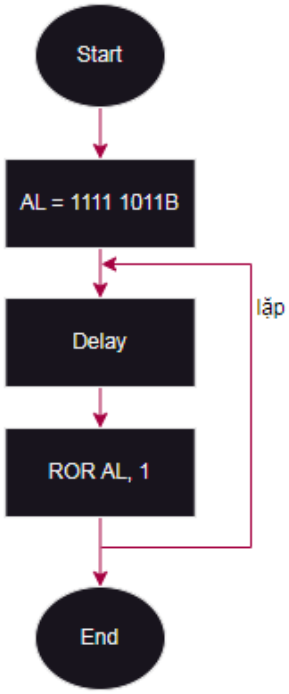
Chương trình hợp ngữ:

```
.MODEL SMALL
.STACK 100H
.DATA
.CODE
MAIN PROC
    MOV AX, @DATA
    MOV DS, AX
    DK_LED EQU 2C0H
    LAP:
        MOV AL, FFH          ;tat tat ca den LED
        OUT DK_LED, AL
        MOV CX, 250          ;tre bang 250 lenh NOP
    TRE_1:
        NOP
        LOOP TRE_1
        XOR AL, AL           ;dat AL = 0, bat tat ca den LED
        OUT DK_LED, AL
```

```
        MOV CX, 250             ;tre bang 250 lenh NOP
    TRE_2:
        NOP
        LOOP TRE_2
        JMP LAP
    MOV AH, 4CH
    INT 21H
MAIN ENDP
END MAIN
```

b) Tạo hiệu ứng 1 đèn sáng chạy từng bước từ trái sang phải, bắt đầu từ đèn D5, mỗi bước dịch chuyển tương ứng với 1 đèn, khoảng nghỉ của 1 bước dịch chuyển là 550 chu kì lệnh NOP.

Lưu đồ:



Chương trình hợp ngữ:

```
.MODEL SMALL
.STACK 100H
.DATA
.CODE
MAIN PROC
    MOV AX, @DATA
    MOV DS, AX
    DK_LED EQU 2C0H
    MOV AL, 0FBH            ;AL = 1111 1011B
    LAP:
        OUT DK_LED, AL
        MOV CX, 550         ;tre bang 550 lenh NOP
    TRE:
        NOP
        LOOP TRE
        ROR AL, 1           ;dich phai 1 bit
        JMP LAP
```

MOV AH, 4CH
INT 21H
MAIN ENDP
END MAIN

c) Tạo hiệu ứng bật tắt các đèn xen kẽ nhau, bắt đầu từ đèn 0, 2, 4, 6 sáng và đèn 1, 3, 5, 7 tắt; sau đó đèn 1, 3, 5, 7 sáng và đèn 0, 2, 4, 6 tắt với khoảng trễ giữa 2 lần bật tắt là 300 chu kì lệnh NOP.

Tương tự câu b), chỉ khác:

MOV AL, 055H ;AL = 0101 0101B

MOV CX, 300

Chương 5: Phối ghép và lập trình điều khiển thiết bị

Yêu cầu: Xây dựng mạch giải mã địa chỉ sử dụng mạch logic cơ bản NAND, mạch tích hợp 74LS138 hoặc 74LS139:

Dữ kiện đề: C_{ROM}, IC, địa chỉ cơ sở.

- **Bước 1:** Xác định số bit cho địa chỉ nội bộ chip IC và mạch giải mã
  Số bit cho nội bộ chip là m=log2(IC) => 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 = C_{ROM}/IC với dung lượng của 1 chip nhớ là IC.
  1 chip nhớ có địa chỉ cuối = địa chỉ đầu + dung lượng chip – 1.
  VD: Địa chỉ của IC 1: Từ ĐCCS đến (ĐCCS + IC – 1H)
  Địa chỉ của IC 2: Từ (ĐCCS + IC) đến (ĐCCS + 2 * IC – 1H)….
- **Bước 3:** Lập bảng bit
  74LS138 là mạch giải mã 3 vào 8 ra
  74LS139 là mạch giải mã 2 vào 4 ra
- **Bước 4:** Vẽ hình mạch giải mã

Bài tập:

1. Hãy 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.

LG

Dữ kiện đề: C_{ROM} = 4KB, IC = 2Kx8 = 2KB, ĐCCS = 03800H

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

Chip nhớ IC 2Kx8 chiếm không gian 2KB = 2^11B

=> Số bit cho địa chỉ nội bộ chip là n = log2(2^11) = 11 (A0 – A10)

VXL 8086 có 20bit địa chỉ=> số bit cho mạch giải mã là 20–11 = 9 (A11–A19)

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

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

=> Số lượng chip nhớ 2Kx8 là C_{ROM}/IC = 2 tương ứng IC1 và IC2.

Đổi dung lượng 1 chip nhớ sang hệ 16:

2^11B = 0000 0000 **1**000 0000 0000 = 00800H

=> Dung lượng 1 chip nhớ là 00800H

Địa chỉ của 1 chip nhớ: (địa chỉ cuối = địa chỉ đầu + dung lượng nhớ - 1)

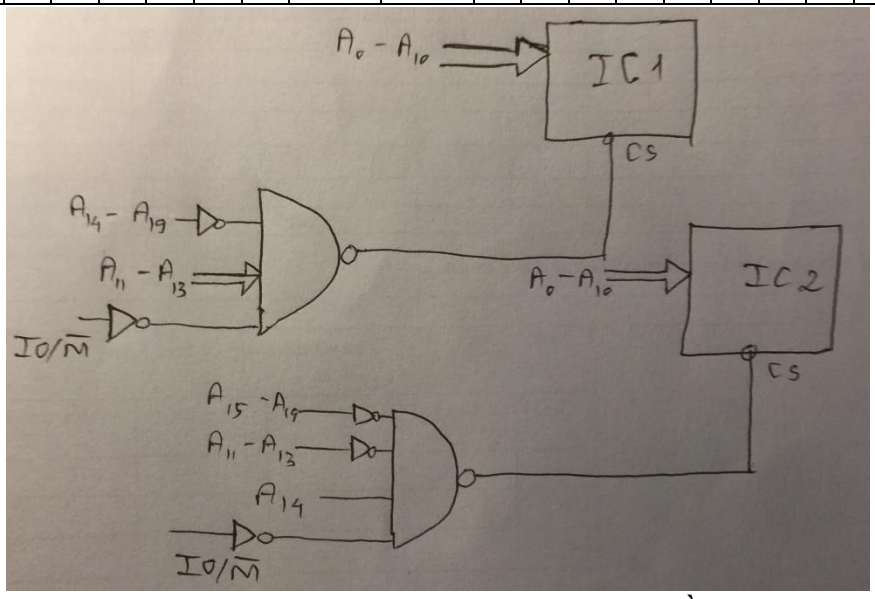
- Dải địa chỉ của IC1: Từ 03800H đến (03800H + 00800H – 00001H) hay từ 03800H đến 03FFFH

- Dải địa chỉ của IC2: Từ 04000H đến (04000H + 00800H – 00001H) hay từ 04000H đến 047FFH

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

	ĐCCS	9bit cho mạch giải mã (A19-A11)									11bit địa chỉ nội bộ chip (A11-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	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		A19								A11	A10										A0

Bước 4: Vẽ mạch giải mã địa chỉ:



2. Xây dựng mạch giải mã địa chỉ cho bộ nhớ có dung lượng 8KB bằng phương pháp sử dụng mạch tích hợp 74LS139; biết rằng bộ nhớ có địa chỉ bắt đầu là 0F800H với các chip nhớ có dung lượng 2Kx8.

LG

Bước 1:

Dung lượng chip nhớ là 2Kx8 = 2KB = 2^11B

=> Số bit cho địa chỉ nội bộ chip là n = log2(IC) = 11bit (A0-A10)

VXL 8086 có 20bit địa chỉ=> số bit cho mạch giải mã là 20-11=9 (A11-A19)

Bước 2:

Bộ nhớ có dung lượng 8KB và dung lượng 1 chip nhớ là 2KB

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

Đổi kích thước chip nhớ sang hệ 16:

(2^11)B = 0000 0000 1000 0000 0000B = 00800H => Dung lượng 1 chip nhớ là 00800H.

Địa chỉ của 1 chip nhớ:

- Dải địa chỉ của IC1: Từ 0F800H đến 0FFFFH
- Dải địa chỉ của IC2: Từ 10000H đến 107FFH
- Dải địa chỉ của IC3: Từ 10800H đến 10FFFH
- Dải địa chỉ của IC4: Từ 11000H đến 117FFH

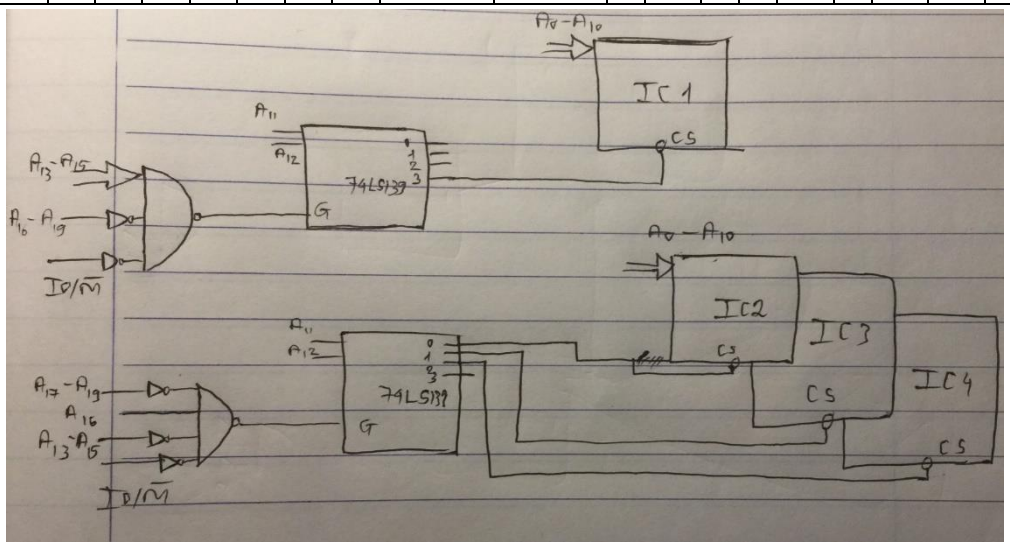
Bước 3:

Bảng bit từ địa chỉ cơ sở các chip: (74LS139 là mạch 2 đầu vào 4 đầu ra)

	ĐCCS	7 bit cao	2 bit vào	11bit địa chỉ nội bộ chip (A10-A0)

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
		A19							A11	A10											A0

Bước 4: Vẽ mạch giải mã địa chỉ:



3. 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 74LS138; biết rằng kích thước 1 vi mạch nhớ là 32KB và địa chỉ cơ sở là 58000H.

LG

Bước 1:

Kích thước 1 vi mạch nhớ là 32KB = 2^15B

=> Số bit địa chỉ nội bộ vi mạch nhớ là n = log2(2^15) = 15bit (A0 – A14)

VXL 8086 có 20bit địa chỉ=> Số bit cho mạch giải mã là 20–15=5bit (A15-A19)

Bước 2:

Bộ nhớ ROM có dung lượng 128KB, kích thước 1 vi mạch nhớ là 32KB

=> Số vi mạch nhớ IC là 128/32 = 4 tương ứng với IC1, IC2, IC3 và IC4.

Đổi dung lượng 1 vi mạch nhớ sang hệ 16:

(2^15)B = 0000 1000 0000 0000 0000B = 08000H => Dung lượng 1 vi mạch IC là 08000H.

Địa chỉ 1 chip nhớ:

Dải địa chỉ của IC1: Từ 58000H đến 5FFFFH

Dải địa chỉ của IC2: Từ 60000H đến 67FFFH

Dải địa chỉ của IC3: Từ 68000H đến 6FFFFH

Dải địa chỉ của IC4: Từ 70000H đến 77FFFH

Bước 3: Lập bảng bit từ ĐCCS các chip: (74LS138 là chip giải mã 3 vào 8 ra)

	ĐCCS	2 bit mạch giải mã		3 bit vào			15 bit địa chỉ nội bộ (A14-A0)														
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

IC 3	68000 H	0	1	**1**	**0**	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0
IC 4	70000 H	0	1	**1**	**1**	**0**	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		A1 9				**A1 5**	A1 4													A 0

Bước 4: Vẽ mạch giải mã: