Lecture 3: Lập trình hợp ngữ cho hệ VXL

1, Tổng quan về ngôn ngữ lập trình

-File nguồn

-Bộ ký tự, từ khóa, tên Assembly

\*Cấu trúc 1 lệnh của assembly

[nhãn:] tên lệnh [toán hạng] [;ghi chú]

-Các chu kỳ trong Assembly

-Chỉ dẫn trong Assembly:

+Nhóm định nghĩa tên tượng trưng: ‘EQU’ hoặc ‘=’

VD: Table EQU [BX][SI]; 1 toán hạng

AD EQU 86; 1 hằng số

move EQU MOV; 1 từ khóa

seg : offset

DS : [BX+SI]

VD: ab = 10;

MOV AL,dong;

ab = 20;

+Nhóm khai báo dữ liệu

++[name] DB data[…]

++[name] DW data[…]

++[name] DD data[…]

VD: table DB 1,2,3,4,5

DB 6,7,8,9

++Count DUP (data[…]);

VD: Mem\_500bytes DB 500 DUP (?);

?: dùng khi khai báo 1 biến hay 1 mảng mà không cần khởi tạo giá trị ban đầu

-Khai báo biến nhãn

Vd:

Warray LABEL word;

Darray LABEL dword;

Barray 100 DUP (?);

-Nhóm khai báo mảng:

Name SEGMENT [align\_type][combine\_type] [‘class’]

Name ENDS

VD:

DATA SEGMENT;

TABLE DB 100DUP(?);

A DB 01;

DATA END

-Khai báo chương trình con:

Name PROC[type]

…

Name ENDP

-Nhóm tham chiếu bên ngoài: PUBLIC và EXTRN

VD:

File 1

PUBLIC CT\_2

CODE SEGMENT

CT\_2 PROC FAR

File 2

EXTR CT\_2

-Nhóm chỉ dẫn điều khiển:

-Nhóm chỉ dẫn chú thích:

COMENT \* [text]

Text

-Toán tử trong assem

+Toán tử số học:

+Toán tử logic:

EQ: =

NE: <>

+Toán tử cung cấp thông tin tìm kiếm SEG

SEG expression ; cho địa chỉ mảng của biếu thức

-Nhóm cung cấp thông tin về biến và nhãn

+Toán tử (:) segment : expression; quy định cách tính địa chỉ của 1 biến hay 1 nhãn đối với 1 mảng được chỉ ra

ES : [BX+4] ⬄ES : [BX]+4

+Toán tử $: cho địa chỉ offset của phát biểu chứa toán tử $ đó, dùng tính độ dài 1 chuỗi

+Toán tử type: TYPE expression; cho biết độ rộng của expression

+Toán tử length: LENGTH var; cho số các đơn vị mà biến var xin cấp phát

+Toán tử PTR: type PTR expression; cho phép thay đổi dạng của biểu thức expression

VD: stuff DD 1

MOV AL, BYTE PRT stuff; nạp byte đầu của mảng table vào AL

-Nhóm thuộc tính:

+Toán tử high và low

HIGH expression

LOW expression

2, Tập lệnh của bộ VXL 80286

A, Nhóm lệnh chuyển dữ liệu

MOV dest,source;

-Chuyển dữ liệu giữa 2 thanh ghi

VD: reg 8 🡨 reg8

MOV AL, AH;

MOV BX, SI;

-Chuyển dữ liệu giữa thanh ghi và bộ nhớ

mem8 🡨 reg8

MOV [BX], Al; chuyển AL vào ngăn nhớ có địa chỉ là nội dung của thanh ghi BX

reg8 🡨 mem8

MOV AL, mem[BX]; DS : mem(BX)

mem16 🡨 reg16

-Gán giá trị hằng vào thanh ghi hay bộ nhớ

reg8 🡨 immed8

MOV AL, 0B5h

Mem8 🡨 immed8

MOV mem[BX],-1

reg16 🡨 immed16

MOV AX, 0B800H

mem16 🡨 immed16

MOV count, 2000

-Chuyển dữ liệu giữa thanh ghi mảng và thanh ghi hay bộ nhớ

segreg 🡨 reg16

MOV DS,AX

segreg 🡨 mem16

MOV ES, screen

reg16 🡨 segreg

MOV AX, CS

mem16 🡨 segreg

MOV [BX + DI], ES

Chú ý:

-CS: không thể dùng làm toán hạng đích trong lệnh MOV

-2 ô nhớ ko thể chuyển trực tiếp với nhau

-2 thanh ghi mảng không thể chuyền trực tiếp với nhau

-Không thể chuyển giá trị trực tiếp (giá trị hằng) vào thanh ghi mảng

PUSH source(reg16 hay mem16)

PUSH immed16 (chỉ dùng với CPU 80286/386/486)

Chức năng: Dùng để cất một hằng hay nội dung một thanh ghi 16bit hay nội dung 1 toán hạng bộ nhớ 16bit vào STACK

VD: PUSH SI

POP dest(reg16 hay mem16)

PUSHA (PUSH ALL) (chỉ dùng với CPU 80286/386/486)

Chức năng: push tất cả các thanh ghi theo thứ tự AX,CX,DX,BX,SP,BP,SI,DI

XLAT (source-cable): chuyển nội dung của ô nhớ nằm trong một bảng các ô nhớ 8bit vào thanh ghi AL. Trong đó bảng có địa chỉ bắt đầu xác định bời cặp thanh ghi DS:BX, vị trí offset của byte nhớ trong bảng được xác định bởi thanh ghi AL

XLAT tương đương với các lệnh sau:

MOV AH,0

MOV SI,AX

MOV AL,[BX+SI]

B, Nhóm lệnh chuyển địa chỉ:

Lệnh LEA: (Load effective address) chuyển địa chỉ offset của 1 toán hạng bộ nhớ mem16 vào thanh ghi đích reg16

LEA reg16,mem16

VD: LEA BX, DS:[500H];

LEA AL, table ⬄ MOV AL,OFFSET table

Lệnh LDS: (Load Point er using DS) chuyển nội dung toán hạng bộ nhớ mem32 vào cặp thanh ghi 16bit, phần cao 16bit của mem32 được nạp vào thanh ghi DS, còn phần tháp được nạp vào thanh ghi reg16 trong lệnh

LDS reg16,mem32

VD: MOV BX, MEM[SI]

Lệnh LES: (Load pointer using ES)

LES reg16, mem32

C, Nhóm lệnh chuyển dữ liệu qua cổng I/O

Lệnh IN: IN AL,port 8

Hay IN AL,DX

VD:

IN AL,16H; đọc 1 byte từ cổng 16H

MOV DX,03f8H

IN AL,DX; đọc một byte từ cổng 03f8H

DX: 100H – FFFFH

Lệnh OUT:

OUT port8, AL

Hay OUT DX,AL

VD:

OUT 61H,AL

MOV DX,03f8H

OUT DX,AL

D, Nhóm lệnh chuyển điều khiển

+Nhảy có điều kiện (J),không điều kiện (JMP)

+Nhảy ngắn, dài

-Lệnh nhảy không điều khiển JMP (Jump):

JMP Label hoặc JMP Target (reg hoặc mem)

-Lệnh nhảy có điều khiện J<điều kiện>:

J<Điều kiện> short\_lable

VD:

JZ/JNZ CT\_1

ZF=1/ZF=0

-Lệnh gọi chương trình con

CALL Label hoặc CALL Target (reg/mem)

+Lệnh quay lại từ chương trình con có cú pháp:

RET (constant) hay

RETN (constant) hay

RETF (constant)

-Lệnh so sánh cú pháp:

CMP left, right

E, Nhóm lệnh lặp

-Lệnh LOOP: LOOP short\_label

Chức năng: giảm nội dung CX đi 1 đơn vị và nhảy đến nhãn short\_label nếu thỏa mãn điều kiện CX <>0. Nếu CX=0 thì lệnh này bị bỏ qua

-Lệnh LOOPE: LOOPE short\_label

Chức năng: giảm nội dung CX đi 1 đơn vị và nhảy đến nhãn short\_label nếu thỏa mãn điều kiện cờ ZF=1 và CX <>0. Nếu không thỏa mãn thì điều kiện này bị bỏ qua

-Lệnh LOOPZ: LOOPZ short\_label

Chức năng: tương đương lệnh LOOPE

-Lệnh LOOPNE: LOOPNE short\_label

F, Nhóm lệnh tính toán số học

-Lệnh tăng INC: INC dest (reg/mem)

-Lệnh giảm DEC: DEC dest (reg/mem)

-Lệnh nhân số không giấu MUL: MUL source

MUL source\_8bit

(AX) <- (AL)\*(source\_8bit)

MUL source\_16bit

(DX AX) <- (AX)\*(source\_16bit)

-Lệnh chia số không dấu DIV: DIV source

(AL) <- (AX) DIV (source\_8bit)

(AH) <- (AX) MOD (source\_8bit)

(AX) <- (DX AX) DIV (source\_16bit)

(DX) <- (DX AX) MOD (source\_16bit)

E, Nhóm lệnh dịch chuyển và quay vòng

-Lệnh SHL: SHL dest,1 hay SHL dest,CL

Chức năng: dịch sang trái toán hạng dest 1 bit (TH1) hay dịch sang trái toán hạng dest với số bước bằng CL bit (TH2)

-Lệnh SHR:

G, Nhóm lệnh xử lý xâu chuỗi

-Lệnh MOVS có các dạng:

MOVSB (chuyển byte),

MOVSW (chuyển word),

MOVS [ES:] dest, [seg:] source

2, Xây dựng chương trình assembly

-Các bước xây dựng chương trình:

B1: Phân tích bài toán

Nội dung:

-Xây dựng lưu đồ thuật toán tổng quát. Tại đây các yếu tố của nhiệm vụ chương trình được xem xét và phân tích nhằm đưa ra giải pháp và phương thức thực hiện tổi ưu. Nếu nhiệm vụ của chương trình là phức tạp thì nó sẽ được tách ra thành các nhiệm vụ con với mối quan hệ được xác định trong thuật toán tổng quát. Mỗi nhiệm vụ con phải có lưu đồ thuật toán riêng để giải quyết trọn vẹn chức năng của nhiệm vụ con đó

-Xây dựng chương trình nguồn assembly. Nếu chức năng không lớn thì chương trình nguồn assembly được viết ở dạng đơn giản, chỉ cần 1 thủ tục chính. Nếu chức năng lớn hay rất lớn thì chương trình nguồn assembly được viết ở dạng phức tạp hơn về cấu trúc. Có thể là 1 file gồm nhiều thủ tục, nhiều marco hay chương trình gồm nhiều file khác nhau

B2: Lưu đồ thuật toán

Tổng quan -> chi tiết

VD1: Viết chương trình chuyển mảng dữ liệu từ vùng nhớ Buffer1 có địa chỉ bắt đầu là 700h tới vùng nhớ Buffer2 có địa chỉ bắt đầu là 1000h. Biết rằng cả hai vùng nhớ đều nằm trong mảng do thanh ghi ghi DS quản lý và kích thước 2 vùng nhớ bằng nhau và bằng 256 byte

B1: Phân tích bài toán

-Chuyển dữ liệu cùng trong 1 mảng

-Dữ liệu nguồn:

+Địa chỉ mảng: DS

+Địa chỉ offset: 700H

+Độ lớn 256 byte

-Dữ liệu đích:

+Địa chỉ mảng: DS

+Địa chỉ offset: 1000H

+Độ lớn 256 byte

* Dùng thanh ghi AL làm thanh ghi trung gian chuyển từng byte dữ liệu từ nguồn sang đích

B2: Lưu đồ thuật toán

B3: Xây dựng chương trình nguồn

**.286**

CODE SEGMENT

ASSUME CS:CODE

ORG 100H ; đầu vào chương trình

CHUYEN\_DL PROC

MOV SI,700H

MOV DI,1000H

MOV CX,256

CHUYEN\_1B

MOV AL,[SI] // DS:[SI]

MOV [DI],AL

INC SI

INC DI

LOOP CHUYEN\_1B

CHUYEN\_DL ENDP

CODE ENDS

END

VD2: Viết chương trình chuyển mảng dữ liệu từ vùng nhớ DISPLAY\_VIDEO (địa chỉ 700h) có  
địa chỉ mảng là DISPLAY\_BASE tới vùng dữ lệu BUFFER\_SAVE có địa chỉ mảng do DS quản lý. Biết rằng vùng nhớ DISPLAY\_VIDEO chứa 1 Kb ký tự (mã hoá ASCII)

B1:

B2:

->

START

->

ES<- DISPLAY\_BASE

SI<-DISPLAY\_VIDEO

DS<-seg BUFFER\_SAVE

DI<-offset BUFFER\_SAVE

CX<-1024

-> (Tiếp)

Chuyển 1 byte dữ liệu từ nguồn sang đích

->

SI=SI+1

DI=DI+1

CX=CX-1

->

CX=0?

-> NO -> (Tiếp)

-> YES -> END

B3: Viết code

DATA SEGMENT

SRC\_SEG EQU DISPLAY\_BASE

SRC\_OFF EQU DISPLAY\_VIDEO

SRC\_SIZE EQU 1024

BUFFER SAVE DB 1024 DUP(?)

DATA ENDS

CODE SEGMENT

ASSUME CS:CODE,DS:DATA

ORG 100H

CHUYEN\_M PROC

MOV AX,SRC\_SEG

MOV ES,AX

MOV SI,SRC\_OFF

LDS DI,BUFFER\_SAVE

MOV CX,SRC\_SIZE

CHUYEN\_1B

MOV AL,ES:[SI]

MOV DS:[DI],AL

INC SI

INC DI

LOOP CHUYEN\_1B

CHUYEN\_M ENDP

CODE ENDS

END

VD2\_Cách2:

B2:

ES <- SEG BUFFER\_SAVE

DI <- OFF BUFFER SAVE

DS <- DISPLAY\_BASE

SI <- DISPLAY\_VIDEO

CX <-1024 :

B3: Code

LES DI,BUFFER\_SAVE ;ES=DS

MOV AX,SRC\_SEG

MOV DS,AX

MOV SI,SRC\_OFF

MOV CX,SRC\_SIZE

CLD ;DF=0

CHUYEN\_1B

MOVSB

LOOP CHUYEN\_1B

**BÀI TẬP VỀ CHUYỂN MÃ**

VD3: Viết chương trình chuyển dữ liệu dạng mã Hex từ cổng 8 bit có địa chỉ 300H tời vùng BufferRam sau khi chuyển đổi thành mã ASCII. Biết rằng

A, Số lượng byte Hex nhận được là 1024 byte

B, Cổng trạng thái 8 bit có địa chỉ 301H có chức năng xác nhận trạng thái có dữ liệu: bit D0=1 báo là có dữ liệu, ngược lại D0=0 báo không có dữ liệu

MOV DX,300H

IN AL,DX

**BÀI TẬP VỀ KHÓA ĐIỆN TỬ**

VD4: