

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

Kiến trúc máy tính

ThS. Đinh Xuân Trường

truongdx@ptit.edu.vn



Posts and Telecommunications
Institute of Technology

Faculty of Information Technology 1



CNTT1

Học viện Công nghệ Bưu chính Viễn thông

January 15, 2023

Giới thiệu về hợp ngữ

Cú pháp của chương trình hợp ngữ

Dữ liệu cho chương trình hợp ngữ

Khung chương trình hợp ngữ

Giới thiệu phần mềm mô phỏng emu8086

Một số cấu trúc điều khiển và ví dụ

Chương trình con

Marco

Một số ví dụ minh hoạ

Giới thiệu thiết bị ảo - Mô phỏng đèn giao thông

Giới thiệu thiết bị ảo - Nhiệt kế và bếp

8088



8086



- ▶ Hợp ngữ (Assembler) là ngôn ngữ lập trình bậc thấp, chỉ cao hơn ngôn ngữ máy;
- ▶ Hợp ngữ là ngôn ngữ gắn liền với các dòng vi xử lý (processor specific).
 - Các lệnh dùng trong hợp ngữ là lệnh của VXL
 - Chương trình hợp ngữ viết cho một VXL có thể không hoạt động trên VXL khác.
- ▶ Chương trình hợp ngữ khi dịch ra mã máy có kích thước nhỏ gọn, chiếm ít không gian nhớ.
- ▶ Hợp ngữ thường được sử dụng để viết:
 - Các trình điều khiển thiết bị
 - Các môđun chương trình cho vi điều khiển
 - Một số môđun trong nhân HĐH (đòi hỏi kích thước nhỏ gọn và tốc độ cao)

- ▶ Trong chương trình hợp ngữ, mỗi lệnh được đặt trên một dòng – dòng lệnh;
- ▶ Lệnh có 2 dạng:
 - Lệnh thật: là các lệnh gọi nhớ của VXL
 - ▶ VD: MOV, SUB, ADD,...
 - ▶ Khi dịch, lệnh gọi nhớ được dịch ra mã máy
 - Lệnh giả: là các hướng dẫn chương trình dịch
 - ▶ VD: MAIN PROC, .DATA, END MAIN,...
 - ▶ Khi dịch, lệnh giả không được dịch ra mã máy mà chỉ có tác dụng định hướng cho chương trình dịch.
 - Không phân biệt chữ hoa hay chữ thường trong các dòng lệnh hợp ngữ khi được dịch.

- ▶ Cấu trúc dòng lệnh hợp ngữ:
[Tên] [Mã lệnh] [Các toán hạng] [Chú giải]
START: MOV AH, 100 ; Chuyển 100 vào thanh ghi AH
- ▶ Các trường của dòng lệnh:
 - Tên:
 - ▶ Là nhãn, tên biến, hằng hoặc thủ tục. Sau nhãn là dấu hai chấm (:)
 - ▶ Các tên sẽ được chương trình dịch gán địa chỉ ô nhớ.
 - ▶ Tên chỉ có thể gồm các chữ cái, chữ số, dấu gạch dưới và phải bắt đầu bằng 1 chữ cái
- ▶ Mã lệnh: có thể gồm lệnh thật và giả

► Các trường của dòng lệnh:

- Toán hạng:

- Số lượng toán hạng phụ thuộc vào lệnh cụ thể
- Có thể có 0, 1 và 2 toán hạng.

- Chú giải:

- Là chú thích cho dòng lệnh
- Bắt đầu bằng dấu chấm phẩy (;)

START: MOV AH, 100 ; Chuyển 100 vào thanh ghi AH

↑	↑	↑	↑
Tên	Mã lệnh	Toán hạng	Chú giải

► Dữ liệu số:

- Thập phân: 0-9
- Thập lục phân: 0-9, A-F
 - Bắt đầu bằng 1 chữ (A-F) thì thêm 0 vào đầu
 - Thêm ký hiệu H (Hexa) ở cuối
 - Ví dụ: 80H, 0F9H
- Nhị phân: 0-1
 - Thêm ký hiệu B (Binary) ở cuối
 - Ví dụ: 0111B, 1000B

► Dữ liệu ký tự:

- Bao trong cặp nháy đơn hoặc kép
- Có thể dùng ở dạng ký tự hoặc mã ASCII
 - 'A' = 65, 'a' = 97

▶ Hằng (constant):

- Là các đại lượng không thay đổi giá trị
- Hai loại hằng:
 - ▶ Hằng giá trị: ví dụ 100, 'A'
 - ▶ Hằng có tên: ví dụ MAX_VALUE
- Định nghĩa hằng có tên:
 <Tên hằng> EQU <Giá trị>

▶ Ví dụ:

MAX	EQU	100
ENTER	EQU	13
ESC	EQU	27

► Biến (variable):

- Là các đại lượng có thể thay đổi giá trị
- Các loại biến:
 - Biến đơn
 - Biến mảng
 - Biến xâu ký tự
- Khi dịch biến được chuyển thành địa chỉ ô nhớ

► Định nghĩa biến đơn:

- Tên biến DB Giá trị khởi đầu: Định nghĩa biến byte
- Tên biến DW Giá trị khởi đầu: Định nghĩa biến word
- Tên biến DD Giá trị khởi đầu: Định nghĩa biến double word

► Ví dụ:

- X DB 10 ; Khai báo biến X và khởi trị 10

- Y DW ? ; Khai báo biến Y và không khởi trị
- Z DD 1000 ; Khai báo biến X và khởi trị 1000

► Định nghĩa biến mảng :

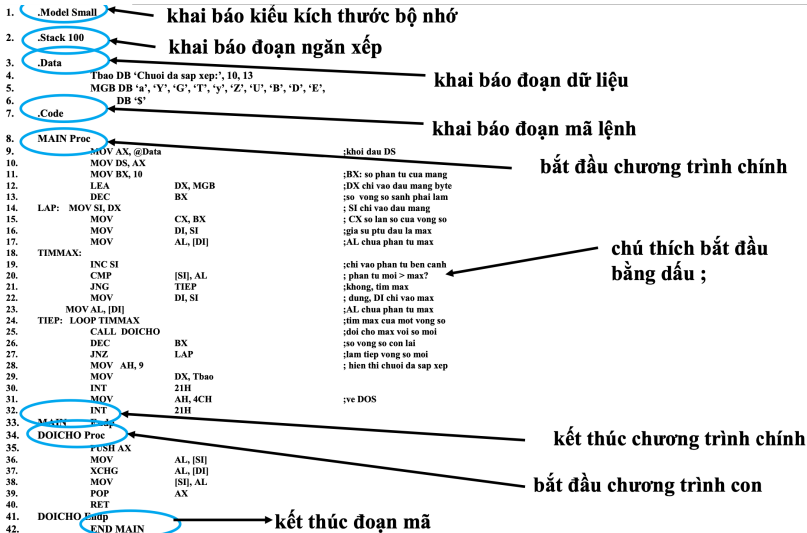
- Tên mảng DB Danh sách giá trị khởi tạo
- Tên mảng DB Số phần tử Dup(Giá trị khởi tạo)
- Tên mảng DB Số phần tử Dup(?)

► Định nghĩa tương tự cho các kiểu DW và DD

► Ví dụ:

- X DB 10, 2, 5, 6, 1; Khai báo mảng X gồm 5 phần tử có khởi trị
- Y DB 5 DUP(0); Khai báo mảng Y gồm 5 phần tử khởi trị 0
- Z DB 5 DUP(?); Khai báo mảng Z gồm 5 phần tử không khởi trị

- ▶ Định nghĩa biến xâu ký tự: có thể được định nghĩa như một xâu ký tự hoặc một mảng các ký tự
- ▶ Ví dụ:
 - str1 DB 'string'
 - str2 DB 73H, 74H, 72H, 69H, 6EH, 67H
 - str3 DB 73H, 74H, 'r', 'i', 69H, 6EH, 67H



- ▶ Khai báo qui mô sử dụng bộ nhớ:
.Model <Kiểu kích thước bộ nhớ>
- ▶ Các kiểu kích thước bộ nhớ:
 - Tiny (hẹp): mã lệnh và dữ liệu gói gọn trong một đoạn
 - Small (nhỏ): mã lệnh gói gọn trong một đoạn, dữ liệu gói gọn trong một đoạn
 - Medium (vừa): mã lệnh không gói gọn trong một đoạn, dữ liệu gói gọn trong một đoạn
 - Compact (gọn): mã lệnh gói gọn trong một đoạn, dữ liệu không gói gọn trong một đoạn
 - Large (lớn): mã lệnh không gói gọn trong một đoạn, dữ liệu không gói gọn trong một đoạn, không có mảng lớn hơn 64K
 - Huge (rất lớn): mã lệnh không gói gọn trong một đoạn, dữ liệu không gói gọn trong một đoạn, có mảng lớn hơn 64K

- ▶ **Khai báo đoạn ngăn xếp:**
Stack <Kích thước ngăn xếp>
- ▶ Ví dụ:
Stack 100H; khai báo kích thước ngăn xếp 100H=256 byte
- ▶ **Khai báo đoạn dữ liệu:**
Data
;Định nghĩa các biến và hằng
;Tất cả các biến và hằng phải được khai báo ở đoạn dữ liệu
- ▶ Ví dụ:
Data
MSG DB 'Hello!\$'
ENTER DB 13
MAX DW 1000

Khai báo đoạn mã:

.Code

Gồm các lệnh của chương trình:

.Code

Jmp Start

; khai báo dữ liệu

Start:

MOV AX,@Data

MOV DS, AX

; các lệnh của chương trình chính

MOV AH, 4CH

INT 21H

End Start ; kết thúc chương trình chính
; các chương trình con – nếu có

Khung chương trình hợp ngữ (cont.)

Tổng hợp khung chương trình



```
1  .Model Small
2  .Stack 100H
3  .Data
4      ; Khai bao cac bien va hang
5  .Code
6      jmp Start
7  Start:
8      ; Khoi tao cho thanh ghi DS
9      MOV AX, @Data; nap dia chi doan du lieu vao AX
10     MOV DS, AX; nap dia chi doan du lieu vao DS
11
12     ; Cac lenh cua chuong trinh chinh
13
14     ; Ket thuc tro ve chuong trinh goi
15     ; Dung ham 4CH cua ngat 21H
16     MOV AH, 4CH
17     INT 21H
18 End Start
19     ; Cac chuong trinh con neu co
```


Khung chương trình hợp ngữ - ví dụ

Viết chương trình in ra thông điệp: Hello World!



```
1  .Model Small
2  .Stack 100H
3  .Data
4      ; Khai bao cac bien va hang
5      CRLF DB 13,10,'$'; Ky tu xuong dong
6      MSG DB 'Hello World!$'
7
8  .Code
9  MAIN Proc
10     ; Khoi tao cho thanh ghi DS
11     MOV AX, @Data; nap dia chi doan du lieu vao AX
12     MOV DS, AX; nap dia chi doan du lieu vao DS
13     ; In gia tri xuong dong
14     MOV AH, 9
15     LEA DX, CRLF; Nap dia chi CRLF vao DX
16     INT 21H
17     ; Hien loi chao dung ham 9 cua ngat 21H
18     MOV AH, 9
19     LEA DX, MSG; Nap dia chi cua thong diep vao DX
20     INT 21H; hien thong diep
21     ; Ket thuc tro ve chuong trinh goi
22     ; dung ham 4CH cua ngat 21H
23     MOV AH, 4CH
24     INT 21H
25
26 MAIN Endp
```

Hàm 1 của ngắt INT 21H: đọc 1 ký tự từ bàn phím

- ▶ Vào: AH = 1
- ▶ Ra: AL = mã ASCII của ký tự cần hiện thị
AL = 0 khi ký tự gõ vào là phím chức năng

Hàm 2 của ngắt INT 21H: hiện 1 ký tự lên màn hình

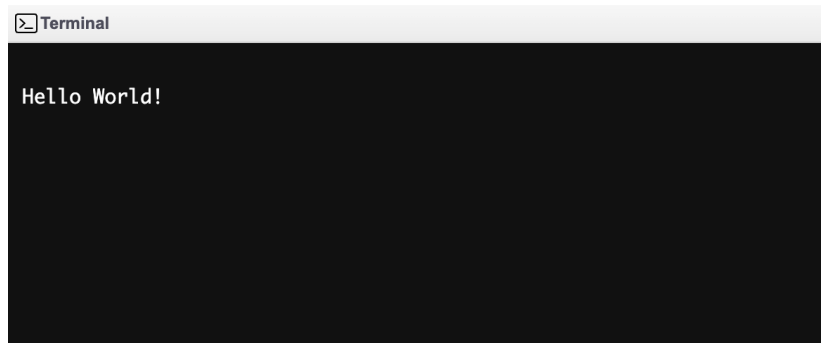
- ▶ Vào: AH = 2
DL = mã ASCII của ký tự cần hiện thị.
- ▶ Ra: Không

Hàm 4CH của ngắt INT 21H: kết thúc chương trình kiểu EXE

- ▶ Vào: AH = 4CH
- ▶ Ra: Không

Hàm 9 của ngắt INT 21H: hiện chuỗi ký tự với \$ ở cuối lên màn hình

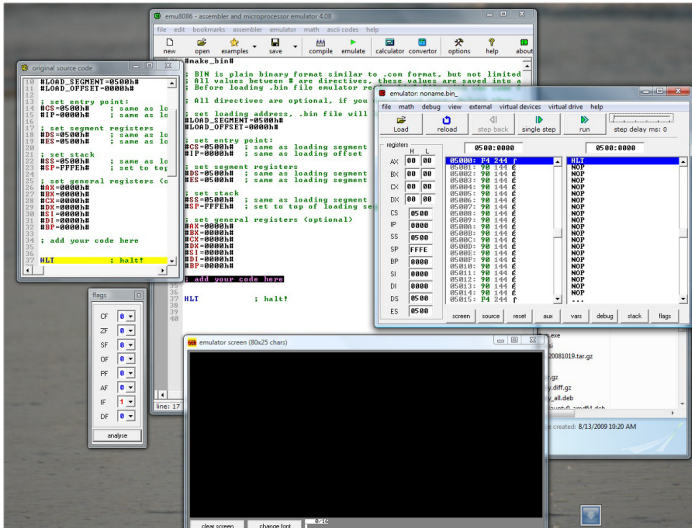
- ▶ Vào: AH = 9
DX = địa chỉ lệch của chuỗi ký tự cần hiện thị.
- ▶ Ra: Không



```
Terminal  
Hello World!
```

Giới thiệu phần mềm mô phỏng emu8086

Emu8086

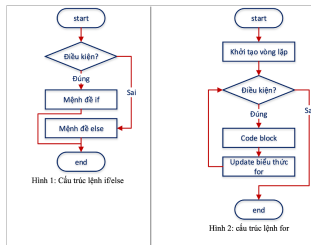


► Cấu trúc lựa chọn

- Rẽ nhánh kiểu IF ... THEN
- Rẽ nhánh kiểu IF ... THEN ... ELSE
- Rẽ nhiều nhánh

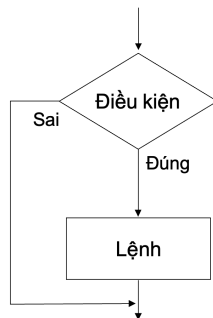
► Cấu trúc lặp

- Lặp kiểu for
- Lặp kiểu repeat ... until



Cấu trúc điều khiển - IF ... THEN:

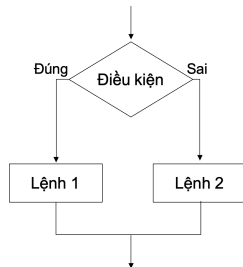
- ▶ IF điều kiện THEN thao tác
- ▶ Gán BX giá trị tuyệt đối AX



```
1  CMP AX,0
2  JNL GAN
3  NEG AX
4  GAN:
5  MOV BX, AX
```

Cấu trúc điều khiển - IF ... THEN ... ELSE:

- ▶ Gán bit dấu của AX cho CL:

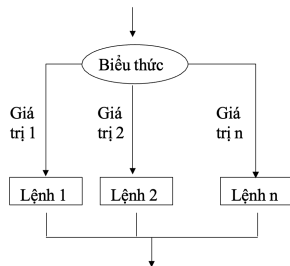


```
1  CMP AX, 0 ; AX > 0 ?
2  JNS DG ; Đúng
3  MOV CL, 1 ; Không, CL <-- 1
4  JMP RA ; Nhảy qua nhanh kia
5  DG: MOV CL, 0 ; CL <-- 0
6  RA:
```

Cấu trúc điều khiển - Rẽ nhiều nhánh:

► Gán giá trị cho CX theo quy tắc:

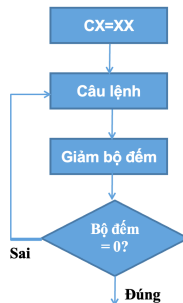
- Nếu $AX < 0$ thì $CX = -1$
- Nếu $AX = 0$ thì $CX = 0$
- Nếu $AX > 0$ thì $CX = 1$



```
1  CMP AX, 0
2  JL  AM
3  JE  KHONG
4  JG  DUONG
5  AM: MOV CX, -1
6      JMP RA
7  DUONG: MOV CX, 1
8          JMP RA
9  KHONG: MOV CX, 0
10 RA:
```


Cấu trúc điều khiển - Lặp kiểu for:

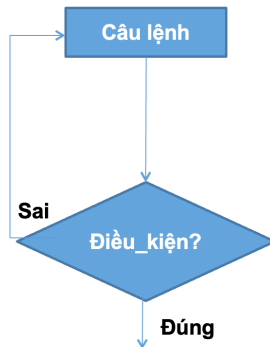
- ▶ Sử dụng lệnh LOOP
- ▶ Số lần lặp CX



```
1  MOV CX,10
2  MOV AH,2
3  MOV DL,'9'
4  Hien: INT 21H
5  LOOP Hien
```

Cấu trúc điều khiển - Lặp kiểu repeat ... until:

- ▶ 1. ...
- ▶ 2. Tiếp:...
- ▶ 3.
- ▶ 4. CMP X,Y; điều kiện
- ▶ Quay lại Tiếp nếu điều_kiện=sai;



- **Bài 1 - Các lệnh số học:** Viết chương trình nhập vào giá trị x và tính giá trị biểu thức f dưới đây. Kết quả được lưu vào thanh ghi AH và in ra màn hình (giả sử x đủ nhỏ sao cho kết quả của biểu thức không vượt quá 8 bit, để đơn giản thử nghiệm với $x = 0$ và $x = 1$).

$$f = ax^3 - bx^2 - cx + d$$

với $a = 4, b = 3, c = 2, d = 1$

*Gợi ý: (theo phương pháp Horner's Method, **sinh viên có thể làm theo cách của riêng mình**)*

- Khởi tạo giá trị cho a, b, c, d, x bằng lệnh số học: add/addi, orx
- Nhân a với x rồi lưu kết quả vào thanh ghi tạm. $t = a * x$
- Thực hiện phép số tính giữa thanh ghi tạm với b . $t = t - b$ // $t = ax - b$
- Nhân thanh ghi tạm với x . $t = t * x$ // $t = (ax - b)x$
- Thực hiện phép số tính giữa thanh ghi tạm với c . $t = t - c$ // $t = ax^2 - bx - c$
- Nhân thanh ghi tạm với x . $t = t * x$ // $t = (ax^2 - bx - c)x$
- Thực hiện phép số tính giữa thanh ghi tạm với d . $t = t + d$ // $t = ax^3 - bx^2 - cx + d$

- **Bài 2 - Các lệnh số học:** Yêu cầu tính toán tương tự Bài 1:

$$f = \frac{ax + b}{cx - d}$$

Với $a = 1, b = 2, c = 1, d = 2$

Hiện tượng gì sẽ xảy ra khi $x = 2$?

- **Bài 3 - Biểu thức If Else:** Nhập giá trị a từ bàn phím, tính và in giá trị c khi biết $b = 20, d = 10$.

```
if (a >= 0) {  
    c = b + d;  
} else {  
    c = b - d;  
}
```

- **Bài 4 - Vòng lặp For:** Dùng vòng lặp for để xuất ra giá trị của số fibonacci thứ n, số n được nhập từ bàn phím.

```
1  f(0) = 0;  
2  f(1) = 1;  
3  for( t0 = n, t0 >= 0, t0 --){  
4      f(t0) = f(t0-1) + f(t0-2)  
5  }
```

- **Bài 5 - Switch - case:** Viết switch-case dưới đây bằng hợp ngữ. Cho biết $b = 10$, $c = 5$. Giá trị input được nhập từ bàn phím.

```
switch (input)  
{  
  
    Case 0: a = b + c; break;  
  
    case 1: a = b - c; break;  
  
    case 2: a = b * c; break;  
  
    case 3: a = b ÷ c; break;  
  
    default: NOP; break;  
  
}
```

- ▶ **Bài 6 - While Loop:** Dùng vòng lặp while để tính tổng của n số đầu tiên. Giá trị n được nhập từ bàn phím và $n > 4$.

```
i    = 0;
sum  = 0;
while ( i != n){
    sum = sum + i;
    i    = i + 1;
}
```

- ▶ **Bài 7 - Sắp xếp mảng:** Cho mảng 10 phần tử, viết chương trình sắp xếp theo giá trị tăng dần.

1 1, 6, 3, 23, 3, 7, 1, 8, 34, 24, 50

- ▶ **Bài 8 - Xử lý chuỗi:** Viết chương trình xử lý chuỗi ký tự sau, những ký tự nào viết hoa chuyển thành viết thường và ngược lại. Lưu ý chỉ xử lý cho ký tự chữ.

1 "Kien_Truc_May_Tinh_CS2023"

- ▶ Chương trình con (còn gọi là thủ tục (procedure) hoặc hàm (function)):
 - Thường gồm một nhóm các lệnh gộp lại;
 - Được sử dụng thông qua tên và các tham số.
- ▶ Ý nghĩa của việc sử dụng chương trình con:
 - Chia chức năng giúp chương trình trong sáng, dễ hiểu, dễ bảo trì;
 - Chương trình con được viết một lần và có thể sử dụng nhiều lần.

Khai báo chương trình con:

```
1  <proc_name> PROC
2
3      ; here goes the code
4      ; of the procedure ...
5
6      RET
7  <proc_name> ENDP
```

Sử dụng chương trình con:

```
1
2  CALL <proc_name>
```

Ví dụ: Hàm **m2** nhân hai

```
1      MOV AL, 1
2      MOV BL, 2
3
4      CALLL m2
5      ; Cac cau lenh khac
6      MOV CX, 30
7
8      ; Dinh nghia mot chuong trinh con
9      ; Input: AL, BL
10     ; Output: AX
11     m2 PROC
12         MUL BL    ; AX = AL * BL
13         RET      ; Tro ve chuong trinh chinh
14     m2 ENDP
```

Truyền tham số Chương trình con

- ▶ Phục vụ trao đổi dữ liệu giữa chương trình gọi và chương trình con;
- ▶ Các phương pháp truyền tham số:
 - Truyền tham số thông qua các thanh ghi
 - ▶ Đưa giá trị vào các thanh ghi lưu tham số cần truyền trước khi gọi hoặc trở về từ chương trình con
 - Truyền tham số thông qua các biến toàn cục
 - ▶ Biến toàn cục (định nghĩa trong đoạn dữ liệu ở chương trình chính) có thể được truy nhập ở cả chương trình chính và chương trình con.
 - Truyền tham số thông qua ngăn xếp
 - ▶ Sử dụng kết hợp các lệnh PUSH / POP để truyền tham số.

- ▶ Bảo vệ các thanh ghi:
 - Cần thiết phải bảo vệ giá trị các thanh ghi sử dụng trong chương trình gọi khi chúng cũng được sử dụng trong chương trình con.
 - Giá trị của các thanh ghi có thể bị thay đổi trong chương trình con
→ sai kết quả ở chương trình gọi.
- ▶ Các phương pháp bảo vệ các thanh ghi:
 - Sử dụng PUSH và POP cho các thanh ghi tổng quát, chỉ số và con trỏ;
 - Sử dụng PUSHF và POPF cho thanh ghi cờ;
 - Sử dụng qui ước thống nhất về sử dụng các thanh ghi.

- ▶ Macro là một đoạn mã được đặt tên và có thể được chèn vào bất cứ vị trí nào trong đoạn mã của chương trình
- ▶ Đặc điểm của macro:
 - Macro hỗ trợ danh sách các tham số
 - Macro chỉ tồn tại khi soạn thảo mã. Khi dịch, các macro sẽ được thay thế bằng đoạn mã thực của macro.
 - Nếu một macro không được sử dụng, mã của nó sẽ bị loại khỏi chương trình sau khi dịch.
 - Macro nhanh hơn thủ tục/hàm do mã của macro được chèn trực tiếp vào chương trình và nó không đòi hỏi cơ chế gọi thực hiện (lưu địa chỉ) và trở về (khôi phục địa chỉ trở về) như chương trình con

Định nghĩa Macro:

```
1  name_macro MACRO [parameters,..]  
2      ;instructions  
3  ENDM
```

Sử dụng Macro:

```
1  name_macro [real parameters,..]
```

Ví dụ về Macro:

```
1  MyMacro MACRO p1, p2, p3
2      MOV AX, p1
3      MOV BX, p2
4      MOV CX, p3
5  ENDM
6  ;...
7  MyMacro 1, 2, 3
8  MyMacro 4, 5, DX
```

Được chuyển thành sau dịch:

```
1  MOV AX, 00001h
2  MOV BX, 00002h
3  MOV CX, 00003h
4  MOV AX, 00004h
5  MOV BX, 00005h
6  MOV CX, DX
```

Một số ví dụ minh họa

Hiện thị lời chào Tây và Ta



```
1  .Model Small
2  .Stack 100H
3  .Data
4      CRLF DB 13, 10, '$'
5      ChaoTay DB 'Hello!$'
6      ChaoTa DB 'Xin_lchao!$'
7  .Code
8  MAIN Proc
9      MOV AX, @Data; Khởi tạo đầu thanh ghi DS
10     MOV DS, AX
11
12     ; Hiện thị lời chào Tây dùng hàm 9 của ngat INT 21H
13
14     MOV AH, 9
15     LEA DX, ChaoTay
16     INT 21H
17
18     ; Xuong 5 dòng dùng hàm 9 ngat INT 21H
19     LEA DX, CRLF
20     INT 21H
21     MOV CX, 6 ; CX chứa số dòng cách + 1
```

LAP:

Một số ví dụ minh hoạ (cont.)

Hiện thị lời chào Tây và Ta



```
23      INT 21H
24      LOOP LAP
25
26      ; Hien thi loi chao Ta dung ham 9 cua ngat INT 21H
27      MOV AH, 9
28      LEA DX, ChaoTa
29      INT 21H
30
31      ; Ket thuc tro ve chuong trinh goi
32      ; Dung ham 4CH cua ngat 21H
33      MOV AH, 4CH
34      INT 21H
35
36      MAIN Endp
37      END MAIN
```

Một số ví dụ minh họa

Đổi các ký tự thường trong 1 chuỗi thành chữ hoa



```
1  .Model Small
2  .Stack 100H
3  .Data
4      ; Source string
5      str1 DB 'a','5', 'B', '?', 'd', 'g', 'P','N','k','*'
6              DB 10,13,'$';
7      ; Destination string
8      str2 DB 10 DUP('')
9              DB '$'
10
11  .Code
12  MAIN Proc
13      MOV AX, @Data; Khởi tạo đầu thanh ghi DS
14      MOV DS, AX
15      MOV ES, AX
16      ; Thiết lập SI trỏ tới str1 và DI trỏ tới str2
17      LEA SI, str1
18      LEA DI, str2
19      CLD
20      MOV CX, 10
21  START:
22      LODSB
23      ; Kiểm tra có phải chữ hoa không
```

Một số ví dụ minh họa (cont.)

Đổi các ký tự thường trong 1 chuỗi thành chữ hoa

```
23      CMP AL, 'a'
24      JL  NotLowerCase
25      CMP AL, 'z'
26      JG  NotLowerCase
27      ; Neu la chu viet thuong chuyen thanh chu viet hoa
28      SUB AL, 20H
29      ;Luu vao string moi
30      NotLowerCase:
31      STOSB
32      LOOP START
33      LEA DX, str1; In ra chuoi goc
34      MOV AH, 9
35      INT 21H
36      LEA DX, str2; In ra chuoi da bien doi
37      MOV AH, 9
38      INT 21H
39      ; Ket thuc tro ve chuong trinh Dung ham 4CH cua ngat 21H
40      MOV AH, 4CH
41      INT 21H
42      MAIN Endp
43      END MAIN
```

Một số ví dụ minh họa

Tìm số lớn nhất trong 1 dãy

```
1  .Model Small
2  .Stack 100H
3  .Data
4      list DB 1,4,0,9,7,2,4,6,2,5
5  .Code
6  MAIN Proc
7      MOV AX, @Data; Khởi tạo đầu thanh ghi DS
8      MOV DS, AX
9
10     CLD
11     MOV CX, 9
12     LEA SI, list; SI trỏ đến list
13     MOV BL, [SI]; Gán Max <-- phần tử đầu tiên
14     INC SI
15 START:
16     LODSB
17
18     CMP AL, BL
19     JLE BYPASS
20     MOV BL, AL; Neu AL > BL --> BL sẽ được cập nhật giá trị
21
22 BYPASS:
```

Một số ví dụ minh họa (cont.)

Tìm số lớn nhất trong 1 dãy



```
23      LOOP START
24
25      ; In ra gia tri lon nhat
26      ADD BL, '0'; bien doi tu so thanh ky tu
27      MOV DL, BL
28      MOV AH, 2
29      INT 21H
30
31      ; Ket thuc tro ve chuong trinh Dung ham 4CH cua ngat 21H
32      MOV AH, 4CH
33      INT 21H
34  MAIN Endp
35      END MAIN
```

Một số ví dụ minh họa

Sắp xếp một dãy số



Sắp xếp và in dãy số theo thứ tự giảm dần:

```
1  .Model Small
2  .Stack 100H
3  .Data
4      LIST_COUNT EQU 10
5      list DB 1,4,0,3,7,2,8
6      CRLF DB 13,10,'$'
7  .Code
8  Main PROC:
9      MOV AX, @Data; nạp địa chỉ đoạn dữ liệu vào AX
10     MOV DS, AX; nạp địa chỉ đoạn dữ liệu vào DS
11
12     ; In danh sách ban đầu
13     MOV CX, LIST_COUNT
14     LEA SI, list
15     CALL printList
16     LEA SI, list
17     MOV BL, 1; Biến đếm lưu vào
18
19     MainLoop:
20         MOV AL, [SI]; AL <-- [SI]
```

Một số ví dụ minh họa (cont.)

Sắp xếp một dãy số

```
21         MOV DI, SI
22         MOV BH, BL; Bien dem phu
23         MOV DX, DI; DX luu vi tri cua gia tri MIN
24     SubLoop:
25         INC DI
26         INC BH
27         CMP , [DI]
28         JLE NotMin
29         MOV AL, [DI]
30         MOV DX, DI
31     NotMin:
32         CMP BH, LIST_COUNT
33         JE ExitSub
34         JMP SubLoop
35     ExitSub:
36         ; Hoan doi vi tri neu MIN khac voi vi tri dau tien
37         MOV DI, DX
38         CMP SI, DI
39         JE NoSwap
40         CALL swapMemLocation
41     NoSwap:
42         INC BL
43         CMP BL, LIST_COUNT
```

Một số ví dụ minh họa (cont.)

Sắp xếp một dãy số

```
44         JE ExitMain
45         INC SI
46         JMP MainLoop
47     ExitMain:
48         LEA DX, CR
49         CALL printString
50
51         MOV CX, LIST_COUNT
52         LEA SI, list
53         CALL printList
54         ; Ket thuc tro ve chuong trinh goi
55         ; Dung ham 4CH cua ngat 21H
56         MOV AH, 4CH
57         INT 21H
58     Main ENDP
59         ; Cac chuong trinh con
60
61         ; Chuong trinh con hoan doi hai vi tri
62         ; Input: SI tro toi vi tri thu nhat
63         ;      DI tro toi vi tri thu hai
64     swapMemLocation PROC
65         PUSH AX,
66         MOV AL, [SI]
```


Một số ví dụ minh họa (cont.)

Sắp xếp một dãy số



```
67         MOV AH, [DI]
68         MOV [SI], AH
69         MOV [DI], AL
70         POP AX
71         RET
72     swapMemLocation ENDP
73
74     ; Chương trình con in ra danh sách - list
75     ; Input: SI lưu địa chỉ bắt đầu của danh sách
76     ;       CX lưu trữ số lượng phần tử của danh sách
77     printList PROC
78         PUSH DX
79         StartPrint:
80             MOV DL, [SI]
81             CALL printSingleDigital
82             INC SI
83             LOOP StartPrint
84             POP DX
85             RET
86     printList ENDP
87
88     ; Chương trình con in ra chuỗi kết thúc bởi $
89     ; Input: DX trỏ tới chuỗi cần in
```

Một số ví dụ minh họa (cont.)

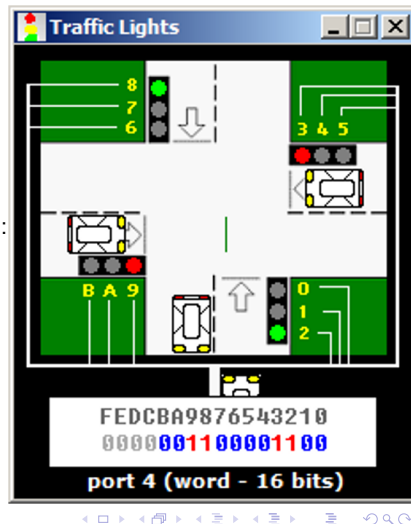
Sắp xếp một dãy số



```
90      printString PROC
91          PUSH AX
92          MOV AH, 9
93          INT 21H
94          POP AX
95          RET
96      printString ENDP
97
98      ; Chương trình con in Mot ky tu so
99      ; Input: DL chua ky tu so can in
100     printSingleDigit PROC
101         PUSH AX,
102         ADD DL, '0'
103         MOV AH, 2
104         INT 21H
105         POP AX
106         RET
107     printSingleDigit ENDP
108
109     END Main
```

Mô phỏng thiết bị - Traffic Lights:

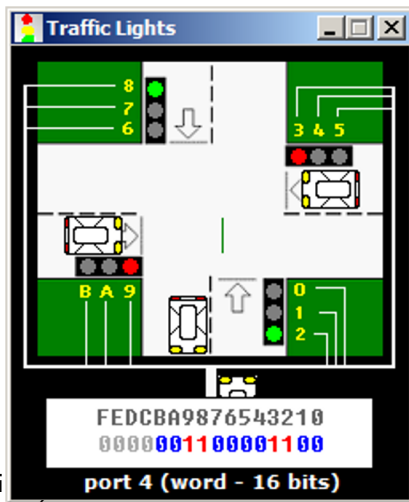
- ▶ Thiết bị ảo hệ thống đèn giao thông sử dụng cổng số 4 – cổng 16 bit để nhận thông tin điều khiển;
- ▶ Sử dụng 12 bit (0-11) cho 4 cụm đèn:
 - Mỗi cụm gồm 3 đèn Green, Yellow và Red;
 - Bit 0 – tắt đèn, bit 1 – bật đèn
- ▶ 4 bit (12-15) không sử dụng – nên đặt là 0.



Giới thiệu thiết bị ảo - Mô phỏng đèn giao thông (cont.)

Điều khiển đèn giao thông:

- ▶ Gửi từ điều khiển (2 bytes) ra cổng số 4;
- ▶ Các bit của từ điều khiển được đặt sao cho phù hợp với ý đồ điều khiển đèn (Bít 0 – tắt đèn, bít 1 – bật đèn)
- ▶ Ví dụ: các bit điều khiển
0000 001 100 001 100
GYR GYR GYR GYR
- ▶ Dùng hàm 86h của ngắt BIOS 15h để tạo thời gian đợi – thời gian giữ trạng thái vừa thiết lập của cụm đèn
- ▶ Số micro giây được đặt vào CX:DX trước khi gọi ngắt.



Giới thiệu thiết bị ảo - Mô phỏng đèn giao thông (cont.)

Code điều khiển mô phỏng:

```
1  .Model Small
2  .Stack 100H
3  .Data
4      ; 0000 GYR GYR GYR GYR
5      R1 DW 0000 0011 0000 1100B; Di doc
6      R2 DW 0000 0010 1000 1010B; Doc --> vang
7      R3 DW 0000 1000 0110 0001B; Di ngang
8      R4 DW 0000 0100 0101 0001B; ngang --> vang
9      ; FEDC BA9 876 543 210
10
11     ALL_RED EQU 0000 0010 0100 1001B
12     PORT EQU 4; Output port
13
14     ; Hang so thoi gian (s)
15     ; 3s = 3,000,000 = 002D C6C0H
16     WAIT_3_SEC_CX EQU 2Dh
17     WAIT_3_SEC_DX EQU 0C6C0h
18
19     ; 10s = 10,000,000 = 0098 9680H
20     WAIT_10_SEC_CX EQU 98h
21     WAIT_10_SEC_DX EQU 9680h
```

Giới thiệu thiết bị ảo - Mô phỏng đèn giao thông (cont.)

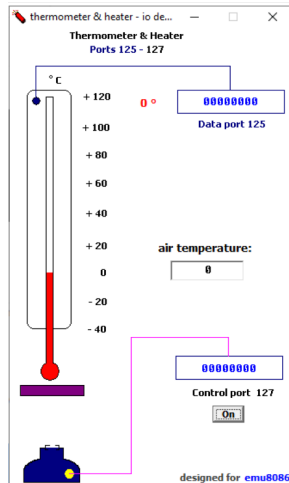


```
22 .Code
23     ; Dinh nghi mot Macro
24     waitMacro MACRO t1, t2
25         MOV CX, t1
26         MOV DX, t2
27         MOV AH, 86H
28         INT 15H
29     ENDM
30 MAIN Proc
31     ; Khoi tao cho thanh ghi DS
32     MOV AX, @Data; nap dia chi doan du lieu vao AX
33     MOV DS, AX; nap dia chi doan du lieu vao DS
34
35     ; Dat den do cho cho tat ca cac huong
36     ; Set lights to Red for all direction
37
38     MOV AX, ALL_RED
39     OUT PORT, AX
40     waitMacro WAIT_3_SEC_CX, WAIT_3_SEC_DX
41 Start:
42     LEA SI, R1
43     MOV AX, [SI]
44     OUT PORT, AX
```

```
45      waitMacro WAIT_10_SEC_CX, WAIT_10_SEC_DX
46      LEA SI, R2
47      MOV AX, [SI]
48      OUT PORT, AX
49      waitMacro WAIT_3_SEC_CX, WAIT_3_SEC_DX
50      LEA SI, R3
51      MOV AX, [SI]
52      OUT PORT, AX
53      waitMacro WAIT_10_SEC_CX, WAIT_10_SEC_DX
54      LEA SI, R4
55      MOV AX, [SI]
56      OUT PORT, AX
57      waitMacro WAIT_3_SEC_CX, WAIT_3_SEC_DX
58      JMP Start
59      ; Ket thuc tro ve chuong trinh goi
60      ; dung ham 4CH cua ngat 21H
61      MOV AH, 4CH
62      INT 21H
63
64  MAIN Endp
65  END MAIN
```

Thiết bị ảo Thermometer & heater:

- ▶ Thiết bị ảo Nhiệt kế và bếp (Thermometer & Heater) sử dụng 2 cổng:
 - Cổng điều khiển số 127 để nhận byte điều khiển bếp:
 - ▶ Gửi 0 để tắt bếp
 - ▶ Gửi 1 để bật bếp.
 - Cổng dữ liệu số 125 để đọc dữ liệu là nhiệt độ được đọc từ nhiệt kế.




```
1  .Model Small
2  .Stack 100H
3  .Data
4      ; Khai bao cac bien va hang
5  .Code
6  MAIN Proc
7      ; Khoi tao cho thanh ghi DS
8      MOV AX, @Data; nap dia chi doan du lieu vao AX
9      MOV DS, AX; nap dia chi doan du lieu vao DS
10
11     ;Doc nhiet do hien tai
12     IN AL, 125
13
14     CMP AL, 60
15     JL low ; Nhay den low neu AL nho hon 60
16
17     CMP AL, 80
18     JLE ok; Nhay den ok neu AL nho hon hoac bang 80
19     JG high; Nhay den high neu AL lon hon 80
20
21     low:
```

```
22         MOV AL, 1
23         OUT 127, AL; Bat bep "ON"
24         JMP ok
25     high:
26         MOV AL, 0
27         OUT 127, AL; Tat bep "OFF"
28
29     ok:
30         JMP Start
31
32         ; Ket thuc tro ve chuong trinh goi
33         ; dung ham 4CH cua ngat 21H
34         MOV AH, 4CH
35         INT 21H
36     MAIN Endp
37     END MAIN
```

Ví dụ: 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 100, 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.

Chương 4

- ▶ Một số ví dụ về Lập trình hợp ngữ
- ▶ Chương trình con
- ▶ Marco
- ▶ Giới thiệu thiết bị ảo – Đền giao thông

Tiếp theo Chương 5 - Phối ghép với bộ nhớ và thiết bị vào ra

- ▶ Các tín hiệu của CPU
- ▶ Các tín hiệu của các mạch phụ trợ
- ▶ Phối ghép CPU với bộ nhớ
- ▶ Phối ghép CPU với thiết bị vào ra
- ▶ Giới thiệu một số mạch hỗ trợ vào ra