Đề 1

* Câu 1: Thiết bị ảo là gì? Lợi ích sử dụng? VD:
  + LG:
    - Thiết bị ảo: Là 1 hệ thống có tổ chức sao cho các chương trình ứng dụng có thể dễ dàng gọi chương trình hệ thống
    - Lợi ích sử dụng:
    - VD: Hệ điều hành
* Câu 2: Giải thích từng dòng
  + LG:

#include "dos.h"

#include "conio.h"

main(){

union REGS Reg1, Reg2; // Khai báo biến Reg1, Reg2 kiểu REGS

struct SREGS Reg3; // Khai báo biến Reg3 kiểu SREGS

char \*s = "Operating System"; // Khai báo biến con trỏ s giá trị “Operating System”

Reg1.h.ah = 0x40; // Chức năng 40h ngắt 21h --> Ghi dữ liệu từ bộ nhớ vào file

Reg1.x.bx = 4; // Thẻ file 4

Reg1.x.cx = 14; // Số byte cần ghi 14

Reg3.ds = FP\_SEG(s); // Gán địa chỉ ds = segment của s

Reg1.x.dx = FP\_OFF(s); // Gán địa chỉ dx = offset của s

int86x(0x21, &Reg1, &Reg2, &Reg3);// Gọi ngắt 21h

getch(); // dừng màn hình

return 0;

}

* Kết quả : In ra màn hình 14 ký tự “Operating Syst”
* Câu 3: Lập trình xác định vị trí bắt đầu và kết thúc của phân khu chủ động trên đĩa cứng (bao gồm số head, cylinder, sector) (Gợi ý: cần lập trình đọc các thông tin trên bảng partion nằm trên MasterBoot)
  + Tìm PK chủ động: trạng thái PK == 80h
  + Lấy thông tin theo bảng thông tin 1 PK và địa chỉ PK tại bảng partition
  + LG:

#include "dos.h"

#include "iostream.h"

main(){

union REGS bien1, bien2;

bien1.h.ah = 2;

bien1.h.al = 1;

bien1.h.cl = 1;

bien1.h.ch = 0;

bien1.h.dh = 0;

bien1.h.dl = 0x80;

char A[512];

struct SREGS bien3;

bien3.es = FP\_SEG(A);

bien1.x.bx = FP\_OFF(A);

int86x(0x13, &bien1, &bien2, &bien3);

// Tim PK chu dong va in thong so

for(int i = 0; i < 4; i++){

int index = 446 + i \* 16;

if(A[index] == 0x80){

cout<<"Vi tri bat dau cua phan khu "<<i<<" :"<<endl;

cout<<"Head: "<<A[index +1]<<endl;

cout<<"Sector: "<<A[index + 2]<<endl;

cout<<"Cylinder: "<<A[index + 3]<<endl;

cout<<"Vi tri ket thuc cua phan khu "<<i<<" :"<<endl;

cout<<"Head: "<<A[index +5]<<endl;

cout<<"Sector: "<<A[index + 6]<<endl;

cout<<"Cylinder: "<<A[index + 7]<<endl;

}

}

}

* Câu 4: Xét ô nhớ tuyến tính có dạng nhị phân như sau:
  + - 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 1 1 1 | 0 0 0 0 0 0 0 0 1 0 1 1
    - hãy cho biết trang, bảng trang (QL bộ nhớ của 80386)
  + LG: (32bit)

0 0 0 0 0 0 1 1 1 0 | 0 0 0 0 0 0 0 1 1 1 | 0 0 0 0 0 0 0 0 1 0 1 1

số hiệu bảng B = 14 |sô hiệu trang T = 7 |Vị trí ô nhớ trong trang d = 11

Đề 2

* Câu 1: Chế độ bảo vệ? Bộ vi xử lý hỗ trợ chế độ bảo vệ
  + LG:
    - Chế độ bảo vệ (Protected mode):
      * Áp dụng phương pháp quản lý bộ nhớ kiểu phân đoạn
      * Có thể quản lý tối đa 16MB bộ nhớ vật lý (nhờ sử dụng địa chỉ vật lý 24 bít)
      * Cho phép chạy các chương trình có kích thước lớn hơn bộ nhớ vật lý nhờ sử dụng các địa chỉ ảo (Virtual Address)
      * Có cơ chế “bảo vệ các đoạn nhớ”, tránh xung đột khi sử dụng bộ nhớ
      * Địa chỉ logic (ảo) dạng segment:offset (segment dài 16 bít địa chỉ đoạn, offset dài 16 bit vị trí ô nhớ trong đoạn ). Địa chỉ này không xác định một ô nhớ vật lý cụ thể, nó ứng với một ô nhớ logic (có thể nằm trên bộ nhớ vật lý hoặc không)
      * Đối với 80386: có thể áp dụng phương pháp quản lý bộ nhớ kiểu phân đoạn hoặc kết hợp phân trang - đoạn và 1 số thay đổi sử dụng địa chỉ vật lý dài 32 và địa chỉ segment:offset dài 16 bít: 32 bít
    - VXL của 80286 và 80386
* Câu 2:
  + LG:
* Câu 3: Lập trình hiện 4 chuỗi ký tự 4 trang màn hình khác nhau
  + - Chuỗi 1: “Day la trang so 0” (chữ đỏ nền trắng) – giữa trang 0
    - Chuỗi 2: “Day la trang so 1” (chữ trắng nền đen) – giữa trang 1
    - Chuỗi 3: “Day la trang so 2” (chữ đỏ nền trắng) – giữa trang 2
    - Chuỗi 4: “Day la trang so 3” (chữ trắng nền blue) – giữa trang 3
    - Ấn 0,1,2,3 để chuyển đổi giữa các trang
    - Ấn ESC để kết thúc chương trình
  + LG:

#include "dos.h"

#include "conio.h"

#include "iostream.h"

void display(char\*s, char \*c, int size, int tr, int color){

int h = 13;

int x = 31;

int vt = 4096 \* tr + (h-1) \* 80 \* 2 + (x-1) \* 2;

for(int i = 0; i < size; i++){

s[vt++] = c[i];

s[vt++] = color;

}

}

main(){

char \*s;

s = (char\*) MK\_FP(0xB800,0);

char \*c1 = "Day la trang so 0";

char \*c2 = "Day la trang so 1";

char \*c3 = "Day la trang so 2";

char \*c4 = "Day la trang so 3";

display(s, c1, 17, 0, 116);

display(s, c2, 17, 1, 7);

display(s, c3, 17, 2, 116);

display(s, c4, 17, 3, 23);

int tr = 0;

union REGS bien1, bien2;

bien1.h.ah = 5;

bien1.h.al = tr;

int86(0x10, &bien1, &bien2);

char kt;

while(kt != 27){

kt = getch();

if(kt == '0') tr = 0;

if(kt == '1') tr = 1;

if(kt == '2') tr = 2;

if(kt == '3') tr = 3;

bien1.h.al = tr;

int86(0x10, &bien1, &bien2);

}

}

* Câu 4:Trong chế độ QL bộ nhớ theo trang của 80386, ô nhớ cuối cùng của trang 256 trong bảng QL trang số 127 sẽ tương ứng với địa chỉ tuyến tính 32 bit bằng bao nhiêu?
  + LG:

ô nhớ cuối --> d : 111111111111 (12 bit)

trang 256 = 2^8 --> T: 0100000000 (10 bit)

bảng 127 = 2^7 – 1 --> B: 0001111111 (10 bit)

địa chỉ tuyến tính B T d : 0001111111 0100000000111111111111

Đề 3

* Câu 1: Thế nào là chiến lược phân trang? Chiến lược phân trang đặc điểm giống và khác với chiến lược phân đoạn
  + LG:
    - Chương trình ở bộ nhớ ngoài được chia thành nhiều phần bằng nhau, mỗi phần gọi là một trang (logic)
    - Bộ nhớ vật lý cũng được chia thành các trang có cùng kích thước với trang ở bộ nhớ ngoài.
    - Các trang của chương trình sẽ được nạp vào các trang trong bộ nhớ vật lý
    - Để quản lý các trang của chương trình, người ta sử dụng bảng quản lý trang PCB (Pages control block), mỗi phần tử của bảng sẽ ứng với một trang logic
    - Chiến lược phân trang thường xuyên phải thực hiện việc nạp lại trang, đưa trang ra bộ nhớ ngoài => Làm tăng hao phí thời gian và chậm tốc độ hệ thống => Cần có các biện pháp nạp trang và đổi trang sao cho thích hợp nhất, tiết kiệm thời gian nhất
    - Các chiến lược nạp trang: Nạp đơn giản / Nạp trước / Nạp theo yêu cầu
    - Các chiến lược đổi trang: Đổi ngẫu nhiên / FIFO (Nạp trước đổi trước) / LRU (Thay trang có lần sử dụng cuối lâu nhất
    - Giống và khác
      * Giống:
        + Chia nhỏ chương trình và nạp từng phần vào bộ nhớ theo bảng QL
        + Chạy được chương trình lớn hơn RAM
      * Khác:

|  |  |
| --- | --- |
| Phân trang | Phân đoạn |
| Nạp chương trình theo từng trang | Nạp chương trình theo từng module |
| Chia bộ nhớ trước thành các trang = nhau | Khi có yêu cầu thì cấp phát bộ nhớ = module chương trình |
| Quản lý = bảng PCB | Quản lý = bảng SCB |
| Địa chỉ ô nhớ(p: số hiệu trang, d: k/c tới đầu trang) | Địa chỉ ô nhớ (s: số hiệu module, d: k/c tới đầu module) |
| Không phù hợp với chương trình truyền thống | Phù hợp với chương trình truyền thống |

* Câu 2:
  + LG:
* Câu 3:Viết chương trình chiếm ngắt 15h để chuyển phím
  + - A -->B ; B --> C; C--> D; .... ; G-->H; H --> I
  + LG:

#include "dos.h"

#include "iostream.h"

#include "conio.h"

char kt;

void interrupt (\*OldInt)(...);

void interrupt NewInt(...){

if('A' <= kt && kt <= 'H'){

kt++;

// cout<<"True\n";

}

OldInt();

}

main(){

kt = getch();

OldInt = getvect(0x15);

disable();

setvect(0x15, NewInt);

enable();

disable();

//if('A' <= kt && kt <= 'H'){

// kt++;

//}

cout<<kt;

enable();

disable();

setvect(0x15, OldInt);

enable();

}

* Câu 4: Tạo 1 file có kích thước 6020 bytes trên thư mục gốc của ổ đĩa cứng.Cho biết vùng data còn trống trên đĩa cứng giảm đi bao nhiêu bytes. tại sao (Giả thiết HĐH: DOS / Windows mỗi cluster = 8 sector)
  + LG:
    - 1 cluster = 8 sector
    - 1 sector = 512 bytes
    - => 1 cluster = 8 \* 512 = 4096
    - => Lưu file 6020 bytes cần 2 cluster = 4096 \* 2 = 8192 (số hiệu cluster đầu tiên được lưu trong bảng thư mục, số hiệu cluster tiếp theo lưu trong FAT và dựa theo các bước đọc file từ đĩa của window 9x / DOS vào bộ nhớ)