**Chương 4. Data Transfers, Addressing, and Arithmetic**

## Bài 1:

.data

Sarray SWORD -1,-2,-3,-4

Viết chương trình copy giá trị của mảng vào thanh ghi EAX, EBX, ECX, EDX.Gọi hàm DumpRegs.Giải thích các giá trị trong các thanh ghi.

|  |  |
| --- | --- |
| INCLUDE Irvine32.inc  .data  Sarray SWORD - 1, -2, -3, -4  .code  main PRO  mov esi, 0  mov eax,0  sub ax, sarray[esi]  neg eax  mov ebx, 0  sub bx, sarray[esi+type Sarray]  neg ebx  mov ecx, 0 | sub cx, sarray[esi + 2\* type Sarray]  neg ecx  mov edx, 0  sub dx, sarray[esi + 3\* type Sarray]  neg edx  call DumpRegs  exit  main ENDP  END main |

## Bài 2

Viết chương trình sử dụng vòng lặp để tính toán bảy giá trị đầu tiên của dãy số Fibonacci,

Các giá trị trong thanh ghi EAX và hiển thị nó bằng câu lệnh gọi DumpRegs bên trong vòng lặp

|  |  |
| --- | --- |
| INCLUDE Irvine32.inc  .data  .code  main PROC  mov ecx, 7  mov eax,0  call DumpRegs  mov ebx, 0  mov eax, 1 | Fibonacci:  call DumpRegs  mov edx,eax  add eax,ebx  mov ebx,edx  loop Fibonacci  exit  main ENDP  END main |

## Bài 3\*:

Sử dụng một vòng lặp với địa chỉ gián tiếp hoặc được lập chỉ mục để đảo ngược các phần tử của một mảng số nguyên tại chỗ.

Không sao chép các phần tử sang bất kỳ mảng nào khác.

|  |  |
| --- | --- |
| INCLUDE Irvine32.inc  .data  Mang DWORD 1, 2, 3, 4, 5, 6, 7, 8, 9  .code  main PROC  mov ecx, (lengthof Mang) / 2  mov esi, offset Mang  mov edi, offset Mang  mov edx, sizeof Mang  sub edx, 4  Dao:  mov eax, [esi]  mov ebx, [edi + edx]  mov[esi], ebx  mov[ed i + edx], eax  add esi, 4  sub edi, 4  Loop Dao | mov ecx, lengthof Mang  mov esi, offset Mang  HienThi :  mov eax, [esi]  add esi, 4  call WriteDec  Loop HienThi  exit  main ENDP  END main |

## Bài 4:

Viết chương trình thực hiện biểu thức số học sau:

EAX = −val2 + 7 - val3 + val1

Sử dụng các định nghĩa dữ liệu sau:

val1 SDWORD 8

val2 SDWORD 15

val3 SDWORD 20

Trong các chú thích bên cạnh mỗi lệnh, hãy viết giá trị thập lục phân của EAX. Chèn lệnh gọi DumpRegs vào cuối chương trình.

|  |  |
| --- | --- |
| INCLUDE Irvine32.inc  .data  val1 SDWORD 8  val2 SDWORD 15  val3 SDWORD 20  .code  main PROC  mov eax,0  sub eax,val2 | add eax,7  sub eax,val3  add eax,val1  call WriteInt  call Dumpregs  exit  main ENDP  END main |

## Bài 5:

Viết chương trình sử dụng lệnh LOOP với địa chỉ gián tiếp sao chép một chuỗi từ nguồn đến đích, đảo ngược thứ tự ký tự trong quá trình. Sử dụng các biến sau:: source BYTE "This is the source string",0

target BYTE SIZEOF source DUP('#')

|  |  |
| --- | --- |
| INCLUDE Irvine32.inc  .data  source BYTE "This is the source string", 0  target BYTE SIZEOF source DUP('#')  .code  main PROC  mov esi,offset source  mov edi, offset target  mov ecx,lengthof source  dec ecx  mov ebx, SIZEOF source | sub ebx,type source  Dao:  mov al,[esi]  mov [edi+ebx],al  inc esi  sub ebx, type source  Loop Dao  mov edx, OFFSET target  call WriteString  exit  main ENDP  END main |

Chương 5. Procedures

## Bài 1:

Xóa màn hình, delay chương trình trong 500 mili giây và kết xuất các thanh ghi và cờ.

|  |  |
| --- | --- |
| INCLUDE Irvine32.inc  .data  .code  main PROC  call Clrscr  mov eax,500 | call delay  call dumpregs  exit  main ENDP  END main |

## Bài 2:

Hiển thị một chuỗi kết thúc bằng null và di chuyển con trỏ đến đầu dòng màn hình tiếp theo.

|  |  |
| --- | --- |
| INCLUDE Irvine32.inc  .data  prompt BYTE "10 điểm nhé thầy: "  .code  main PROC  mov edx, OFFSET prompt  call WriteString | call Crlf  call readchar  exit  main ENDP  END main |

## Bài 3:

Tạo và hiển thị mười số nguyên có dấu giả ngẫu nhiên trong phạm vi 0 - 99. Chuyển từng số nguyên tới WriteInt trong EAX và hiển thị trên một dòng riêng biệt.

|  |  |
| --- | --- |
| INCLUDE Irvine32.inc  .data  RandVal Dword ?  .code  main PROC  call Randomize  mov ecx,9  lap:  mov eax,100  call RandomRange | call writeInt  call crlf  loop lap  exit  main ENDP  END main |

## Bài 4:

Viết chương trình xóa màn hình, nhắc người dùng nhập hai số nguyên, cộng các số nguyên và hiển thị tổng của chúng.

|  |  |
| --- | --- |
| INCLUDE Irvine32.inc  .data  tb1 byte "Nhập vào 2 số nguyên: ",0  tb2 byte "Tổng 2 số là: ",0  .code  main PROC  mov edx,offset tb1  call WriteString  call crlf  call readInt  mov ebx,eax  call readInt  add eax,ebx | mov edx, offset tb2  call WriteString  call WriteInt  call crlf  call waitMsg  exit  main ENDP  END main  ; tổng 2 số |

## Bài 5:

Thủ tục RandomRange từ thư viện Irvine32 tạo ra một số nguyên giả ngẫu nhiên từ 0 đến N-1.

Viết một thủ tục BetterRandomRange cải tiến tạo ra một số nguyên ngẫu nhiên từ M đến N-1 với M, N nhập từ bàn phím.

mov ebx,-300 ; lower bound

mov eax,100 ; upper bound call BetterRandomRange

|  |  |
| --- | --- |
| INCLUDE Irvine32.inc  .data  .code  main PROC  call Randomize  call Clrscr  mov eax, -300  mov ebx, 100  mov ecx, 50  L1:  push eax  push ebx  call BetterRandomRange  pop ebx  pop eax  Loop L1 | call WaitMsg  exit  main ENDP  BetterRandomRange PROC  sub ebx, eax  xchg ebx, eax  call RandomRange  neg ebx  sub eax, ebx  call WriteInt  call Crlf  ret  BetterRandomRange ENDP  END main  ;random số |

## Bài 6:

Viết chương trình tạo và hiển thị 20 chuỗi ngẫu nhiên, mỗi chuỗi gồm 10 chữ cái viết hoa {A..Z}.

|  |  |
| --- | --- |
| INCLUDE Irvine32.inc  strLen = 10  .data  arr BYTE strLen DUP(? )  .code  main PROC  call Clrscr  call Randomize  mov esi, offset arr  mov ecx, 20  L1:  call GenerateRandomString  Loop L1  call WaitMsg  exit  main ENDP | GenerateRandomString PROC USES ecx  mov ecx, lengthOf arr  L2 :  mov eax, 26  call RandomRange  add eax, 65  call WriteChar  loop L2  call Crlf  ret  GenerateRandomString ENDP  END main  ;random chuỗi ngẫu nhiên |

## Bài 7:

Nhập vào một mảng các số nguyên Dword. Tính tổng các phần tử của mảng

|  |  |
| --- | --- |
| INCLUDE Irvine32.inc  .data  tb1 byte "Nhập vào số phần từ của mảng: ",0  tb2 byte "Nhập mảng: ", 0  tb3 byte "Tổng bảng là: ",0  arr dword 100 dup(0)  .code  main PROC  mov edx,offset tb1  call WriteString  call ReadDec  mov ecx,eax  mov ebx,eax  mov edx, offset tb2  call WriteString  call crlf  mov esi,0 | l1:  call ReadDec  mov arr[esi],eax  add esi, type arr  loop l1  mov eax, 0  mov esi,0  mov ecx,ebx  l2:  add eax, arr[esi]  add esi, type arr  loop l2  mov edx, offset tb3  call WriteString  call WriteDec  call crlf  call WaitMsg  exit  main ENDP  END main  ; tính tổng của mảng |

## Bài 8:

Nhập vào một xâu ký tự, đảo ngược xâu không dùng thêm mảng phụ

|  |  |
| --- | --- |
| INCLUDE Irvine32.inc  .data  tb1 byte "Nhập vào 1 chuỗi: ",0  arr byte 40 dup(?)  .code  main PROC  mov edx, offset tb1  call WriteString  mov edx, offset arr  mov ecx,40  call ReadString  mov ecx,eax  mov ebx,eax  mov edx, offset arr  lap:  mov al, [edx]  movzx eax,al  push eax  inc edx  loop lap | mov ecx,ebx  mov edx,offset arr  lap2 :  pop eax  mov [edx],al  inc edx  loop lap2  mov edx, offset arr  call WriteString  exit  main ENDP  END main  ; đảo ngược chuỗi |

Chương 6. Conditional Processing

## Bài 1:

Triển khai mã C ++ sau bằng hợp ngữ, sử dụng .IF có cấu trúc khối và

Chỉ thị .WHILE. Giả sử rằng tất cả các biến là số nguyên có dấu 32 bit:

int array [] = {10,60,20,33,72,89,45,65,72,18};

int sample= 50;

int ArraySize = sizeof array / sizeof sample;

int index = 0;

int sum = 0;

while (index <ArraySize)

{

if (array [index] <= sample)

{

sum + = array [index];

}

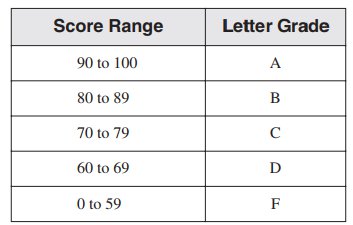
index ++;

}

|  |  |
| --- | --- |
| INCLUDE Irvine32.inc  .data  Array DWORD 10, 60, 20, 33, 72, 89, 45, 65, 72, 18  .code  main PROC  mov esi, 0  mov ecx, LENGTHOF Array  mov edx, -1  LOOP1\_main:  inc edx  cmp edx, ecx  je EXIT1\_main  mov eax, DWORD PTR [Array+edx \* 4] | call DumpRegs  cmp eax, 50  ja LOOP1\_main  add esi, eax  jmp LOOP1\_main  EXIT1\_main:  mov eax, esi  call WriteInt  exit  main ENDP  END |

## Bài 2:

Sử dụng bảng sau làm hướng dẫn, hãy viết một chương trình yêu cầu người dùng nhập điểm kiểm tra số nguyên từ 0 đến 100. Chương trình sẽ hiển thị loại chữ cái thích hợp:



|  |  |
| --- | --- |
| INCLUDE Irvine32.inc  .data  Str1 BYTE "Nhap diem: ", 00h  Str2 BYTE "So diem khong dung. Vui long nhap lai.", 0ah, 00h  Str3 BYTE "Diem cua ban la: ", 00h  .code  main PROC  LOOP1\_main:  mov edx, OFFSET Str1  call WriteString  call ReadInt  cmp eax, 0  jl ERROR1\_main  cmp eax, 100  ja ERROR1\_main  cmp eax, 60  jb F\_LETTER  cmp eax, 70  jb D\_LETTER  cmp eax, 80  jb C\_LETTER  cmp eax, 90  jb B\_LETTER  jmp A\_LETTER  ERROR1\_main:  mov edx, OFFSET Str2  call WriteString  jmp LOOP1\_main | A\_LETTER:  mov al, 'A'  jmp EXIT1\_main  B\_LETTER:  mov al, 'B'  jmp EXIT1\_main  C\_LETTER:  mov al, 'C'  jmp EXIT1\_main  D\_LETTER:  mov al, 'D'  jmp EXIT1\_main  F\_LETTER:  mov al, 'F'  jmp EXIT1\_main  EXIT1\_main:  mov edx, OFFSET Str3  call WriteString  call WriteChar  exit  main ENDP  END |

## Bài 3:

Sử dụng chương trình giải từ bài tập trước làm điểm bắt đầu, thêm các tính năng sau:

• Chạy lặp lại để có thể nhập nhiều điểm thi.

• Tích lũy bộ đếm số điểm kiểm tra.

• Thực hiện kiểm tra phạm vi trên đầu vào của người dùng: Hiển thị thông báo lỗi nếu điểm kiểm tra nhỏ hơn 0 hoặc lớn hơn 10\

|  |  |
| --- | --- |
| INCLUDE Irvine32.inc  .data  Str1 BYTE "Nhap diem: ", 00h  Str2 BYTE "So diem khong dung. Vui long nhap lai.", 0ah, 00h  Str3 BYTE "Diem cua ban la: ", 00h  Str4 BYTE "Diem thu 1", 0ah, 00h  .code  main PROC  mov ecx, 0  mov edx, OFFSET Str4  mov Str4[9], 31h  add Str4[9], cl  call WriteString  LOOP1\_main:  mov edx, OFFSET Str1  call WriteString  call ReadInt  cmp eax, 0  jl ERROR1\_main  cmp eax, 100  ja ERROR1\_main  cmp eax, 60  jb F\_LETTER  cmp eax, 70  jb D\_LETTER  cmp eax, 80  jb C\_LETTER  cmp eax, 90  jb B\_LETTER  jmp A\_LETTER | ERROR1\_main:  mov edx, OFFSET Str2  call WriteString  jmp LOOP1\_main  A\_LETTER:  mov al, 'A'  jmp EXIT1\_main  B\_LETTER:  mov al, 'B'  jmp EXIT1\_main  C\_LETTER:  mov al, 'C'  jmp EXIT1\_main  D\_LETTER:  mov al, 'D'  jmp EXIT1\_main  F\_LETTER:  mov al, 'F'  jmp EXIT1\_main  EXIT1\_main:  mov edx, OFFSET Str3  call WriteString  call WriteChar  mov al, 0ah  call WriteChar  mov al, 0ah  call WriteChar  inc ecx  mov edx, OFFSET Str4  mov Str4[9], 31h  add Str4[9], cl  call WriteString  jmp LOOP1\_main  exit  main ENDP |

## Bài 4:

Sử dụng ví dụ về Đăng ký đại học từ Phần 6.7.3 làm điểm bắt đầu, hãy làm như sau:

• Mã hóa logic bằng cách sử dụng CMP và các lệnh nhảy có điều kiện (thay vì các lệnh .IF và .ELSEIF).

• Thực hiện kiểm tra phạm vi về giá trị tín dụng; nó không được nhỏ hơn 1 hoặc lớn hơn 30. Nếu mục nhập không hợp lệ được phát hiện, hãy hiển thị thông báo lỗi thích hợp.

• Nhắc người dùng về các giá trị điểm trung bình và tín chỉ.

• Hiển thị một thông báo hiển thị kết quả đánh giá, chẳng hạn như “Học sinh có thể đăng ký” hoặc “Học sinh không thể đăng ký”.

## Bài 5:

Tạo một chương trình có chức năng như một máy tính boolean đơn giản cho các số nguyên 32 bit. Nó sẽ hiển thị một menu yêu cầu người dùng thực hiện lựa chọn từ danh sách sau:

1. x AND y

2. x OR y

3. NOT x

4. x XOR y

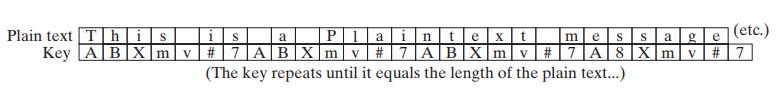
5. Exit program

Khi người dùng thực hiện một lựa chọn, hãy gọi một thủ tục hiển thị tên của thao tác sắp được thực hiện.

|  |  |
| --- | --- |
| INCLUDE Irvine32.inc  .data  Str1 BYTE 0ah, 0ah, "1. x and y", 0ah, "2. x or y", 0ah, "3. not x", 0ah, "4. x xor y", 0ah, "5. Exit Program", 0ah, 0ah, 00h  Str2 BYTE "Nhap lua chon: ", 00h  Str3 BYTE "Lua chon khong dung. Vui long chon lai.", 0ah, 0ah, 00h  Str4 BYTE "Nhap X: ", 00h  Str5 BYTE "Nhap Y: ", 00h  xandy BYTE "x & y = ", 00h  xory BYTE "x | y = ", 00h  notx BYTE "~x = ", 00h  xxory BYTE "x ^ y = ", 00h  .code  main PROC  LOOP1\_main:  mov edx, OFFSET Str1  call WriteString  mov edx, OFFSET Str2  call WriteString  call ReadInt  cmp eax, 1  je X\_AND\_Y  cmp eax, 2  je X\_OR\_Y  cmp eax, 3  je NOT\_X  cmp eax, 4  je X\_XOR\_Y  cmp eax, 5  je EXIT1\_main  mov edx, OFFSET Str3  call WriteString  jmp LOOP1\_main  X\_AND\_Y:  call Nhap  mov edx, OFFSET xandy  call WriteString  and ebx, ecx  mov eax, ebx  call WriteInt  jmp LOOP1\_main  X\_OR\_Y:  call Nhap  mov edx, OFFSET xory  call WriteString | or ebx, ecx  mov eax, ebx  call WriteInt  jmp LOOP1\_main  NOT\_X:  call Nhap  mov edx, OFFSET notx  call WriteString  neg ebx  mov eax, ebx  call WriteInt  jmp LOOP1\_main  X\_XOR\_Y:  call Nhap  mov edx, OFFSET xxory  call WriteString  xor ebx, ecx  mov eax, ebx  call WriteInt  jmp LOOP1\_main  EXIT1\_main:  exit  main ENDP  Nhap PROC  push ebp  mov ebp, esp  mov edx, OFFSET Str4  call WriteString  call ReadInt  mov ebx, eax  mov edx, OFFSET Str5  call WriteString  call ReadInt  mov ecx, eax  pop ebp  ret  Nhap ENDP  END |

## Bài 6:

Sửa đổi chương trình mã hóa trong Phần 6.3.4 theo cách sau: Cho phép người dùng nhập khóa mã hóa gồm nhiều ký tự. Sử dụng khóa này để mã hóa và giải mã bản rõ bằng cách XOR từng ký tự của khóa với một byte tương ứng trong thông báo. Lặp lại phím nhiều lần nếu cần cho đến khi tất cả các byte văn bản thuần túy được dịch. Ví dụ, giả sử khóa bằng “ABXmv # 7”. Đây là cách khóa sẽ mã với các byte văn bản thuần túy:



# Chương 7. Interger Arithmetic

## Bài 1:

Ước chung lớn nhất (GCD) của hai số nguyên là số nguyên lớn nhất sẽ chia đều cả hai số nguyên. Thuật toán GCD liên quan đến phép chia số nguyên trong một vòng lặp, được mô tả bằng mã C ++ sau:

int GCD (int x, int y)

{

x = abs (x); // giá trị tuyệt đối

y = abs (y);

do {

int n = x% y;

x = y;

y = n;

} while (y> 0);

trả về x;

}

|  |  |
| --- | --- |
| INCLUDE Irvine32.inc  .data  Str0 BYTE "Nhap X: ", 00h  Str1 BYTE "Nhap Y: ", 00h  Str2 BYTE "Uoc chung lon nhat cua X va Y la: ", 00h  Str3 BYTE "X va Y phai khac 0.", 0ah, 00h  .code  main PROC  LOOP1\_main:  mov edx, OFFSET Str0  call WriteString  call ReadInt  cmp eax, 0  je ERROR1\_main  push eax  mov edx, OFFSET Str1  call WriteString  call ReadInt  cmp eax, 0  je ERROR1\_main  push eax  call GCD  mov edx, OFFSET Str2  call WriteString  call WriteInt  mov al, 0ah  call WriteChar  jmp LOOP1\_main  ERROR1\_main:  mov edx, OFFSET Str3  call WriteString  jmp LOOP1\_main  main ENDP | GCD PROC  push ebp  mov ebp, esp  mov ebx, [ebp + 12]  mov ecx, [ebp + 8]  push ebx  call abs  add esp, 4  mov ebx, eax  push ecx  call abs  add esp, 4  mov ecx, eax  LOOP1\_GCD:  mov eax, ebx  mov edx, 0  div ecx  mov ebx, ecx  mov ecx, edx  cmp ecx, 0  jg LOOP1\_GCD  mov eax, ebx  pop ebp  ret  GCD ENDP  abs PROC  push ebp  mov ebp, esp  mov eax, [ebp + 8]  cmp eax, 0  jg EXIT1\_abs  neg eax  EXIT1\_abs:  pop ebp  ret  abs ENDP  END |

## Bài 3:

Thời gian của thư mục tệp MS-DOS sử dụng các bit từ 0 đến 4 cho giây, bit 5 đến 10 cho phút và bit 11 đến 15 cho giờ (đồng hồ 24 giờ).

Ví dụ: giá trị nhị phân sau cho biết thời gian là 02:16:14, ở định dạng hh: mm: ss: 00010 010000 00111

|  |  |
| --- | --- |
| INCLUDE Irvine32.inc  .data  .code  main PROC  mov ebx, 0  mov ax, 0001001000001110b  mov cx, ax  mov bx, cx  and bx, 1111100000000000b  shr bx, 11  mov eax, ebx  call WriteInt  mov al, ":"  call WriteChar | mov bx, cx  and bx, 0000011111100000b  shr bx, 5  mov eax, ebx  call WriteInt  mov al, ":"  call WriteChar  mov bx, cx  and bx, 0000000000011111b  mov eax, ebx  call WriteInt  exit  main ENDP  END |

Viết thủ tục có tên ShowFileTime nhận giá trị thời gian tệp nhị phân trong thanh ghi AX và hiển thị thời gian bằng hh : mm: định dạng ss.

## Bài 4:

Viết một chương trình thực hiện mã hóa đơn giản bằng cách xoay mỗi byte bản rõ một số vị trí khác nhau theo các hướng khác nhau. Ví dụ: trong mảng sau đại diện cho khóa mã hóa, giá trị âm cho biết xoay sang trái và giá trị dương cho biết xoay sang phải. Số nguyên ở mỗi vị trí cho biết độ lớn của chuyển động quay:

|  |  |
| --- | --- |
| INCLUDE Irvine32.inc  .data  key BYTE -2, 4, 1, 0, -3, 5, 2, -4, -4, 6  .code  main PROC  mov eax, 0  mov ebx, 0  mov edx, LENGTHOF key  LOOP1\_main:  mov al, key[ebx]  cmp al, 0  jl LABEL1\_main  mov cl, al | rol al, cl  call WriteInt  mov al, ","  call WriteChar  inc ebx  cmp ebx, edx  je EXIT1\_main  jmp LOOP1\_main |

## Bài 5:

Viết một thủ tục tính tích từ 1-n với N nhập vào từ bàn phím

|  |  |
| --- | --- |
| INCLUDE Irvine32.inc  .data  Str0 BYTE "Nhap N: ", 00h  Str1 BYTE "Tich tu 1 den N la: ", 00h  Str2 BYTE "N phai lon hon 0. Vui long nhap lai.", 0ah, 00h  .code  main PROC  LABEL1\_main:  mov esi, 1  mov edx, 0  mov edx, OFFSET Str0  call WriteString  call ReadInt  cmp eax, 0  jle ERROR1\_main  mov ecx, eax  inc ecx | mov eax, esi  LOOP1\_main:  mul esi  inc esi  cmp esi, ecx  jne LOOP1\_main  mov edx, OFFSET Str1  call WriteString  call WriteInt  exit  ERROR1\_main:  mov edx, OFFSET Str2  call WriteString  jmp LABEL1\_main  main ENDP  END |

## Bài 7:

Viết chương trình tìm BCNN của 2 số a, b

|  |  |
| --- | --- |
| INCLUDE Irvine32.inc  .data  Str0 BYTE "Nhap X: ", 00h  Str1 BYTE "Nhap Y: ", 00h  Str2 BYTE "Boi chung nho nhat cua X va Y la: ", 00h  Str3 BYTE "X va Y phai lon hon 0.", 0ah, 00h  .code  main PROC  LOOP1\_main:  mov edx, OFFSET Str0  call WriteString  call ReadInt  cmp eax, 0  jle ERROR1\_main  push eax  mov edx, OFFSET Str1  call WriteString  call ReadInt  cmp eax, 0  jle ERROR1\_main  push eax  call GCD  mov edx, OFFSET Str2  call WriteString  call WriteInt  mov al, 0ah  call WriteChar  jmp LOOP1\_main | ERROR1\_main:  mov edx, OFFSET Str3  call WriteString  jmp LOOP1\_main  main ENDP  GCD PROC  push ebp  mov ebp, esp  mov ebx, [ebp + 12]  mov ecx, [ebp + 8]  LOOP1\_GCD:  mov eax, ebx  mov edx, 0  div ecx  mov ebx, ecx  mov ecx, edx  cmp ecx, 0  jg LOOP1\_GCD  mov eax, [ebp + 12]  mov ecx, [ebp + 8]  mul ecx  div ebx  pop ebp  ret  GCD ENDP  END |

## Bài 8:

Nhập vào một số a. Viết một thủ tục kiểm tra xem số a của phải là số hoàn chỉnh không

|  |  |
| --- | --- |
| INCLUDE Irvine32.inc  .data  Str0 BYTE "Nhap so: ", 00h  Str1 BYTE "Day la so hoan hao.", 0ah, 00h  Str2 BYTE "Day khong phai la mot so hoan hao.", 0ah, 00h  Str3 BYTE "So phai lon hon 0. Vui long nhap lai.", 0ah, 00h  .code  main PROC  LOOP1\_main:  mov edi, 0  mov edx, OFFSET Str0  call WriteString  call ReadInt  cmp eax, 0  mov ebx, eax  jle ERROR1\_main  mov ecx, 0 | LOOP2\_main:  inc ecx  mov edx, 0  mov eax, ecx  mov edx, 2  mul edx  cmp eax, ebx  ja EXIT1\_main  mov edx, 0  mov eax, ebx  div ecx  cmp edx, 0  jne LOOP2\_main  add edi, ecx  jmp LOOP2\_main  EXIT1\_main:  cmp edi, ebx  je SUCCESS1\_main  mov edx, OFFSET Str2  call WriteString  exit  SUCCESS1\_main:  mov edx, OFFSET Str1  call WriteString  exit  ERROR1\_main:  mov edx, OFFSET Str0  call WriteString  jmp LOOP1\_main  main ENDP  END |

## Bài 9:

Viết chương trình nhập dãy các số nguyên dương từ bàn phím, cho tới khi nhập số âm thì kết thúc nhập, in giá trị lớn nhất

|  |  |
| --- | --- |
| INCLUDE Irvine32.inc  .data  Str0 BYTE "Nhap so: ", 00h  Str1 BYTE "So lon nhat: ", 0ah, 00h  Str2 BYTE "So phai khac 0. Vui long nhap lai.", 0ah, 00h  .code  main PROC  mov ebx, 0  LOOP1\_main:  mov edx, OFFSET Str0  call WriteString  call ReadInt  cmp eax, 0  je ERROR1\_main  cmp eax, 0  jl EXIT1\_main  cmp eax, ebx  jbe LOOP1\_main  mov ebx, eax  jmp LOOP1\_main | ERROR1\_main:  mov edx, OFFSET Str2  call WriteString  jmp LOOP1\_main  EXIT1\_main:  mov edx, OFFSET Str1  call WriteString  mov eax, ebx  call WriteInt  exit  main ENDP  END |

## Bài 10:

Lập trình hợp ngữ theo yêu cầu sau:

* Đếm số ký tự X có trong một String

|  |  |
| --- | --- |
| INCLUDE Irvine32.inc  .data  Str0 BYTE "Nhap chuoi: ", 00h  Str1 BYTE "Nhap ky tu: ", 00h  Str2 BYTE "So ky tu trong chuoi la: ", 00h  Source BYTE 100 DUP(0), 00h  .code  main PROC  mov edi, 0  mov edx, OFFSET Str0  call WriteString  mov edx, OFFSET Source  mov ecx, 101  call ReadString  mov ecx, eax  dec ecx  mov edx, OFFSET Str1  call WriteString  call ReadChar | LOOP1\_main:  cmp ecx, 0  jl EXIT1\_main  mov bl, BYTE PTR Source[ecx]  dec ecx  cmp bl, al  jne LOOP1\_main  inc edi  jmp LOOP1\_main  EXIT1\_main:  mov edx, OFFSET Str2  call WriteString  mov eax, edi  call WriteInt  exit  main ENDP  END |

## Bài 11:

Lập trình hợp ngữ Triển khai theo thủ tục theo yêu cầu sau:

* Thay thế ký tự X bằng ký tự Y có trong một String

|  |  |
| --- | --- |
| INCLUDE Irvine32.inc  .data  Str0 BYTE "Nhap chuoi: ", 00h  Str1 BYTE "Nhap ky tu X: ", 00h  Str3 BYTE "Nhap ky tu Y: ", 00h  Source BYTE 100 DUP(0), 00h  .code  main PROC  mov edi, 0  mov edx, OFFSET Str0  call WriteString  mov edx, OFFSET Source  mov ecx, 101  call ReadString  mov ecx, eax  dec ecx  mov edx, OFFSET Str1  call WriteString  call ReadChar  mov bl, al  mov al, 10  call WriteChar | mov edx, OFFSET Str3  call WriteString  call ReadChar  mov cl, al  mov al, 10  call WriteChar  mov al, cl  mov ah, bl  LOOP1\_main:  cmp ecx, 0  jl EXIT1\_main  mov bl, BYTE PTR Source[ecx]  dec ecx  cmp bl, ah  jne LOOP1\_main  mov Source[ecx + 1], al  jmp LOOP1\_main  EXIT1\_main:  mov edx, OFFSET Source  call WriteString  exit  main ENDP  END |

## Bài 12:

Lập trình hợp ngữ theo yêu cầu sau:

- Thể hiện lại bộ cấu trúc if … then … else của ngôn ngữ C

- Thể hiện lại cấu trúc while của ngôn ngữ C

- Thể hiện lại cấu trúc for của ngôn ngữ C

## Bài 13:

Câu hỏi: Lập trình hợp ngữ theo yêu cầu sau:

- Tính tổng của dãy số từ 1 đến n

- Triển khai theo thủ tục

# Chương 8: Advanced Procedures

## Bài 1:

Viết một chương trình thử nghiệm sử dụng INVOKE để gọi WriteColorChar và hiển thị một hình vuông màu (10 hàng x 20 cột) với các cặp màu xanh lam xen kẽ và thanh dọc màu trắng. Gọi một thủ tục riêng khi in từng hàng của hình vuông.

|  |  |
| --- | --- |
| INCLUDE Irvine32.inc  .data  .code  WriteColorChar PROC, color:DWORD, digit:BYTE  mov eax, 0  mov eax, color  call SetTextColor  mov al, digit  call WriteChar  ret  WriteColorChar ENDP  main PROC  mov edx, 0  mov ecx, 0  LOOP1\_main:  mov ebx, 0 | LOOP2\_main:  Invoke WriteColorChar, 48, ' '  Invoke WriteColorChar, 240, ' '  add ebx, 2  cmp ebx, 20  jne LOOP2\_main  mov al, 10  call WriteChar  inc ecx  cmp ecx, 10  jne LOOP1\_main  Invoke WriteColorChar, 0, ' '  exit  main ENDP  END |

## Bài 2:

Tạo một mảng các số nguyên được sắp xếp ngẫu nhiên. Viết Hàm hoán đổỉ Swap, hãy viết một vòng lặp trao đổi từng cặp số nguyên liên tiếp trong mảng.

|  |  |
| --- | --- |
| INCLUDE Irvine32.inc  .data  Array DWORD 43, 12, 34, 67, -33, 12, 53, 63, -50  .code  main PROC  mov ecx, LENGTHOF Array  sub ecx, 1  mov edx, 0  LOOP1\_main:  mov eax, DWORD PTR [Array + edx\*4]  mov ebx, DWORD PTR [Array + edx\*4 + 4]  mov [Array + edx\*4], ebx  mov [Array + edx\*4 + 4], eax | inc edx  cmp edx, ecx  jne LOOP1\_main  mov edx, 0  LOOP2\_main:  mov eax, DWORD PTR [Array + edx\*4]  call WriteInt  mov al, ","  call WriteChar  inc edx  cmp edx, ecx  jbe LOOP2\_main  exit  main ENDP  END |

## Bài 3:

Tính n giai thừa dùng hàm đệ quy

|  |  |
| --- | --- |
| INCLUDE Irvine32.inc  .data  Str1 BYTE "Nhap N: ", 00h  Str2 BYTE "N phai lon hon 0", 0ah, 00h  Str3 BYTE "N! = ", 00h  .code  main PROC  LOOP1\_main:  mov edx, OFFSET Str1  call WriteString  call ReadInt  cmp eax, 1  jl ERROR1\_main  push eax  call GiaiThua  mov edx, OFFSET Str3  call WriteString  call WriteInt  exit  ERROR1\_main:  mov edx, OFFSET Str2 | call WriteString  jmp LOOP1\_main  main ENDP  GiaiThua PROC  push ebp  mov ebp, esp  mov ebx, DWORD PTR [ebp + 8]  cmp ebx, 1  je LABEL1\_GiaiThua  dec ebx  push ebx  call GiaiThua  add esp, 4  inc ebx  mul ebx  jmp RET1\_GiaiThua  LABEL1\_GiaiThua:  mov eax, 1  RET1\_GiaiThua:  pop ebp  ret  GiaiThua ENDP  END |

## Bài 4:

Nhập vào một mảng gồm n phẩn tử. Viết hàm sắp xếp mảng

|  |  |
| --- | --- |
| INCLUDE Irvine32.inc  .data  Array DWORD 1, 10, 5, 4, 3, 7, 8, 2, 9, 6  .code  main PROC  mov ecx, LENGTHOF Array  mov edi, 1  LOOP1\_main:  mov eax, DWORD PTR [Array + edi\*4]  mov esi, edi  LOOP2\_main:  mov ebx, DWORD PTR [Array + esi\*4 - 4]  cmp ebx, eax  jbe LABEL1\_main  cmp esi, 0 | je LABEL1\_main  mov [Array + esi\*4], ebx  dec esi  jmp LOOP2\_main  LABEL1\_main:  mov [Array + esi\*4], eax  inc edi  cmp edi, ecx  jne LOOP1\_main  mov edi, 0  LOOP3\_main:  mov eax, DWORD PTR [Array + edi\*4]  call WriteInt  mov al, ","  call WriteChar  inc edi  cmp edi, ecx  jne LOOP3\_main  exit  main ENDP  END |

## Bài 5:

Viết chương trình nhập N từ bàn phím. Kiểm tra xem N có là số nguyên tố hay không

## Bài 6:

Nhập vào N. Viết hàm in ra các số nguyên tố từ 1 đến N

## Bài 7:

Tìm n số nguyên tố đầu tiên

## Bài 8:

Viết chương trình nhập dãy các số nguyên dương từ bàn phím, cho tới khi nhập số âm thì kết thúc nhập, tìm giá trị lớn nhất

## Bài 9:

Viết chương trình chọn ngẫu nhiên một kí tự c nào đó, cho phép lặp nhập một kí tự từ bàn phím, nếu trùng với c thì thông báo chọn đúng và kết thúc

|  |  |
| --- | --- |
| .data  Str0 BYTE "Nhap ky tu: ", 00h  Success BYTE 0ah, "Thanh Cong!!", 0ah, 00h  .code  main PROC  mov eax, 127  LOOP1\_main:  call RandomRange  cmp eax, 32  jb LOOP1\_main  mov bl, al | LOOP2\_main:  mov al, 0ah  call WriteChar  mov edx, OFFSET Str0  call WriteString  call ReadChar  cmp al, bl  jne LOOP2\_main  mov edx, OFFSET Success  call WriteString  exit  main ENDP  END |

## Bài 10:

Viết chương in bảng cửu chương ra màn hình

|  |  |
| --- | --- |
| INCLUDE Irvine32.inc  .data  Str0 BYTE "a x b = ", 00h  .code  main PROC  mov ecx, 1  LOOP1\_main:  mov ebx, 1  mov Str0[0], 30h  add Str0[0], cl  LOOP2\_main:  mov Str0[4], 30h  add Str0[4], bl  mov edx, OFFSET Str0 | call WriteString  mov eax, ecx  mul ebx  call WriteInt  mov al, 0ah  call WriteChar  inc ebx  cmp ebx, 10  jne LOOP2\_main  inc ecx  cmp ecx, 10  jne LOOP1\_main  exit  main ENDP  END |

## Bài 11:

Viết chương trình nhập vào một xâu ký tự bất kỳ và xoá k ký tự của xâu ký tự bắt đầu từ vị trí thứ n.

|  |  |
| --- | --- |
| INCLUDE Irvine32.inc  .data  Source BYTE 100 DUP(?), 00h  Str0 BYTE "Nhap chuoi: ", 00h  Str1 BYTE "Nhap N: ", 00h  Str2 BYTE "Nhap K: ", 00h  Str3 BYTE "N, K khong dung. Vui long Nhap lai", 0ah, 00h  .code  main PROC  mov edx, OFFSET Str0  call WriteString  mov edx, OFFSET Source  mov ecx, 101  call ReadString  mov edi, eax  push edi  push OFFSET Str1  call Check  add esp, 8  mov ebx, eax  push edi  push OFFSET Str2  call Check  add esp, 8  mov ecx, eax  dec ebx  LOOP1\_main:  mov edx, ebx  add edx, ecx  cmp edx, edi  ja EXIT1\_main  mov al, BYTE PTR Source[edx]  mov Source[ebx], al  inc ebx  jmp LOOP1\_main | EXIT1\_main:  mov edx, OFFSET Source  call WriteString  exit  main ENDP  Check PROC  push ebp  mov ebp, esp  mov edi, DWORD PTR [ebp + 12]  LOOP1\_Check:  mov edx, DWORD PTR [ebp + 8]  call WriteString  call ReadInt  cmp eax, edi  ja ERROR1\_Check  cmp eax, 0  jle ERROR1\_Check  jmp RET1\_Check  ERROR1\_Check:  mov edx, OFFSET Str3  call WriteString  jmp LOOP1\_Check  RET1\_Check:  pop ebp  ret  Check ENDP |

## Bài 12:

Nhập xâu họ tên (không quá 40 kí tự), chuẩn hoá xâu đó (kí tự đầu từ viết hoa, các kí tự khác viết thường, các từ cách nhau 1 dấu cách)

|  |  |
| --- | --- |
| INCLUDE Irvine32.inc  .data  Source BYTE 40 DUP(?), 00h  Target BYTE 40 DUP(?), 00h  Str0 BYTE "Nhap chuoi: ", 00h  Str1 BYTE "Chuoi chi co chu thuong hoac chu hoa. Vui long nhap lai.", 0ah, 00h  .code  main PROC  call NhapChuoi  mov edi, 0  mov al, BYTE PTR Source[0]  mov Target[0], al  mov esi, 1  LOOP1\_main:  inc edi  cmp edi, ecx  je EXIT1\_main  mov al, BYTE PTR [Source + edi]  mov bl, BYTE PTR [Source + edi - 1]  cmp al, 32  je SPACE  cmp bl, 32  je FIRST\_DIGIT  mov Target[esi], al  inc esi  jmp LOOP1\_main  EXIT1\_main:  mov edx, OFFSET Target  call WriteString  exit  FIRST\_DIGIT:  sub al, 32  mov Target[esi], al  inc esi  jmp LOOP1\_main  SPACE:  cmp bl, 32  je LABEL1  mov Target[esi], al  inc esi  LABEL1:  jmp LOOP1\_main  main ENDP  NhapChuoi PROC  push esp  mov ebp, esp | LOOP1\_NhapChuoi:  mov edx, OFFSET Str0  call WriteString  mov edx, OFFSET Source  mov ecx, 41  call ReadString  mov ecx, eax  mov edi, -1  LOOP2\_NhapChuoi:  inc edi  cmp edi, ecx  je RET1\_NhapChuoi  mov al, BYTE PTR [Source + edi]  cmp al, 32  je LOOP2\_NhapChuoi    cmp al, 65  jb ERROR1\_NhapChuoi  cmp al, 90  jbe LOOP2\_NhapChuoi  cmp al, 97  jb ERROR1\_NhapChuoi  cmp al, 122  ja ERROR1\_NhapChuoi  jmp LOOP2\_NhapChuoi  RET1\_NhapChuoi:  mov edi, -1  LOOP3\_NhapChuoi:  inc edi  cmp edi, ecx  je RET2\_NhapChuoi  mov al, BYTE PTR [Source + edi]  cmp al, 32  je LOOP3\_NhapChuoi  cmp al, 97  jae LOOP3\_NhapChuoi  add Source[edi], 32  jmp LOOP3\_NhapChuoi  RET2\_NhapChuoi:  sub Source[0], 32  mov edx, OFFSET Source  pop ebp  ret  ERROR1\_NhapChuoi:  mov edx, OFFSET Str1  call WriteString  jmp LOOP1\_NhapChuoi  NhapChuoi ENDP  END |

## Bài 13:

Viết hàm nhập mảng, in mảng, hàm sắp xếp mảng.

|  |  |
| --- | --- |
| INCLUDE Irvine32.inc  .data  arr dword ?  tb byte "Nhap vao so phan tu mang",0ah,00h,0  .code  main PROC  mov edx, offset tb  call writestring  call readint  mov ecx,eax  call Nhap  Call Xuat  Call SapXep  exit  main ENDP  Nhap proc uses ecx  mov esi, offset arr  lap:  call readint  mov [esi],eax  add esi,4  loop lap  ret  Nhap endp  Xuat proc uses ecx  mov esi, offset arr  lap1:  mov eax,[esi]  call writeint  add esi,4  loop lap1  ret  Xuat endp | SapXep proc uses ecx  mov edi, 1  LOOP1\_main:  mov eax, DWORD PTR [Arr + edi\*4]  mov esi, edi  LOOP2\_main:  mov ebx, DWORD PTR [Arr + esi\*4 - 4]  cmp ebx, eax  jbe LABEL1\_main  cmp esi, 0  je LABEL1\_main  mov [Arr + esi\*4], ebx  dec esi  jmp LOOP2\_main  LABEL1\_main:  mov [Arr + esi\*4], eax  inc edi  cmp edi, ecx  jne LOOP1\_main  mov edi, 0  LOOP3\_main:  mov eax, DWORD PTR [Arr + edi\*4]  call WriteInt  mov al, ","  call WriteChar  inc edi  cmp edi, ecx  jne LOOP3\_main  ret  SapXep endp  END main |