ВОЛЖСКИЙ УНИВЕРСИТЕТ ИМЕНИ В.Н. ТАТИЩЕВА

Кафедра «Информатика и системы управления»

Курсовая работа

по дисциплине: «Организация ЭВМ»

Вариант - 107

Содержание

Введение

1 Постановка задачи

2 Описание и схема алгоритма

3 Листинг программы

Заключение

Список литературы

# Введение

Если коротко, то язык ассемблера — это символическое представление машинного языка.

Все процессы в машине на самом низком, аппаратном уровне приводятся в действие только командами (инструкциями) машинного языка. Отсюда понятно, что, несмотря на общее название, язык ассемблера для каждого типа компьютера свой. Это касается и внешнего вида программ, написанных на ассемблере, и идей, отражением которых этот язык является.

По-настоящему решить проблемы, связанные с аппаратурой (или даже, более того, зависящие от аппаратуры как, к примеру, повышение быстродействия программы), невозможно без знания ассемблера.

Программист или любой другой пользователь может использовать любые высокоуровневые средства, вплоть до программ построения виртуальных миров и, возможно, даже не подозревать, что на самом деле компьютер выполняет не команды языка, на котором написана его программа, а их трансформированное представление в форме скучной и унылой последовательности команд совсем другого языка — машинного. А теперь представим, что у такого пользователя возникла нестандартная проблема или просто что-то не заладилось. К примеру, его программа должна работать с некоторым необычным устройством или выполнять другие действия, требующие знания принципов работы аппаратуры компьютера. И вот здесь-то и начинается совсем другая история.... Каким бы умным ни был программист, каким бы хорошим ни был язык, на котором он написал свою чудную программу, без знания ассемблера ему не обойтись. И не случайно практически все компиляторы языков высокого уровня содержат средства связи своих модулей с модулями на ассемблере либо поддерживают выход на ассемблерный уровень программирования.

Из всего вышесказанного можно сделать вывод, что, так как язык ассемблера для компьютера “родной”, то и самая эффективная программа может быть написана только на нем (при условии, что ее пишет квалифицированный программист). Здесь есть одно маленькое “но”: это очень трудоемкий, требующий большого внимания и практического опыта процесс. Поэтому реально на ассемблере пишут в основном программы, которые должны обеспечить эффективную работу с аппаратной частью. Иногда на ассемблере пишутся критичные по времени выполнения или расходованию памяти участки программы. Впоследствии они оформляются в виде подпрограмм и совмещаются с кодом на языке высокого уровня.

# 1. Постановка задачи

В данной курсовой необходимо изучить теоретический материал, разработать алгоритм программы и реализовать его при помощи языка низкого уровня ассемблер. Программа должна выполнять следующие функции:

1 Ввод матрицы 4х4

2 Вывод введенной матрицы

3 Создание одномерного массива (первый элемент – минимальный элемент из строки номер 0, второй – максимальный из столбца номер 1 матрицы, третий – минимальный из строки номер 2 матрицы и т.д.)

4 Тестирование созданного массива (если положительные элементы чередуются то вывод на монитор этого массива, иначе вывод соответствующего сообщения).

# 2. Описание и схема алгоритма

Рисунок 1 – Алгоритм программы

Рисунок 1 продолжение – Алгоритм программы

Рисунок 1 продолжение – Алгоритм программы

Рисунок 1 продолжение – Алгоритм программы

Рисунок 1 продолжение – Алгоритм программы

Рисунок 1 продолжение – Алгоритм программы

# 3 Листинг программы

model small

include io.asm

.data

add\_elem db 10, 13, 'Введите элементы матрицы : $'

ishod\_matrix db 10, 13, 'Исходная матрица : $'

preobr\_matrix db 10, 13, 'Преобразованная матрица : $'

tab db 10, 13, ' $'

pustie\_str db 10, 13, 10, 13, 10, 13,'$'

probel db ' $'

m dw 4 ; размерность квадратной матрицы

matrix dw 4 dup (4 dup (?)) ; матрица

mas1 dw 4 dup(?)

zn\_chereduetsa db 10,13,'Знаки чередуются $'

zn\_nechereduetsa db 10,13,'Знаки не чередуются $'

.stack 100h

.code

start:

mov ax, @data

mov ds, ax

mov es, ax

;--------------------------------------------------- очистка экрана

mov ah, 06 ; 6-я функция ...

mov al, 0 ; весь экран прокрутить

mov bh, 7 ; с цветами черный-белый

mov dx, 184Fh ; границы экрана: нижний правый...

mov cx, 0 ; ... и верхний левый углы

int 10h ; ... 10-го прерывания

;--------------------------------- установка курсора в верхний левый угол

MOV AH,2 ; 2 функция...

MOV BH,0 ; координаты

MOV DX,0 ; координаты

INT 10H ; ... 10-го прерывания

;--------------------------------- Ввод матрицы --------------------

lea dx, add\_elem

mov ah, 09h

int 21h

lea dx, pustie\_str

mov ah, 09h

int 21h

lea dx, tab ; выводим смещение на сл. строку для красоты

mov ah, 09h

int 21h

mov cx, 4 ; количество строк

mov bx, 0 ; обнуляем bx - смещение по строкам

m1: ;

push cx ; сохраняем cx - количество строк

mov cx, m ; сl <- количество столбцов(элементов в строке)

mov si, 0 ; обнуляем si - смещение внутри строки(по столбцам)

m2: ; с этой метки начинается ввод элемента в строку

inint matrix[bx][si]

lea dx, probel ; выводим на экран пробел для красоты

mov ah, 09h

int 21h

inc si ; увеличиваем смещение на 1

inc si

loop m2 ; если не равны, значит строка не закончилась т.е. заново вводим

pop cx ; востанавливаем количество строк

mov ax, m ; в al <- размер строки

add bx, ax ; bx=bx+ax (переводим смещение на сл. строку)

add bx, ax

lea dx, tab ; выводим смещение на сл. строку для красоты

mov ah, 09h

int 21

newline

loop m1 ; если не равны значит строки не окончены т.е. заново

;------------------------------------- очистка экрана

mov ah, 06 ; 6-я функция ...

mov al, 0 ; весь экран прокрутить

mov bh, 7 ; с цветами черный-белый

mov dx, 184Fh ; границы экрана: нижний правый...

mov cx, 0 ; ... и верхний левый углы

int 10h ; ... 10-го прерывания

;-------------------------------- установка курсора в верхний левый угол

MOV AH,2 ; 2 функция...

MOV BH,0 ; координаты

MOV DX,0 ; координаты

INT 10H ; ... 10-го прерывания

;-------------------- вывод матрицы на экран

lea dx, ishod\_matrix ; исходная матрица -

mov ah, 09h

int 21h

lea dx, tab ; выводим на экран смещение

mov ah, 09h

int 21h

mov cx, m ; количество строк

mov bx, 0 ; номер строки

\_m1: ;

push cx ; сохраняем

mov cx, m ; для того чтобы ввести количество столбцов

mov si, 0 ; номер столбца

\_m2:

outint matrix[bx][si] ; выводим элемент матрицы

lea dx, probel ; далее выводим пробел

mov ah, 09h

int 21h

inc si; переходим на сл. столбец (элемент строки)

inc si

loop \_m2 ; если не конец строки, то заново

pop cx ; востанавливаем количество строк

mov ax, m ;

add bx, ax ; переход на сл. строку матрицы

add bx, ax

lea dx, tab ; выводим на экран смещение

mov ah, 09h

int 21h

loop \_m1 ; если не последняя строка, заново

;1- преобразование одномерной матрици

mov cx, m ; количество сок

mov bx, 0; номер строки

xor di,di

\_mm1: ;

push cx ; сохраняем

mov cx, m ; для того чтобы ввести количество столбцов

mov si, 0 ; номер столбца

mov dx, matrix[bx][0] ; max=matrix[i][0]

\_mm2:

cmp dx, matrix[bx][si] ; если matrix[i][j]<max то ререход на след элемент

jl bolshe; иначе max=matrix[i][j]

mov dx, matrix[bx][si]

mov mas1[di],dx; mas1=max

bolshe: inc si; переходим на сл. столбец (элемент строки)

inc si

loop \_mm2; если не конец строки, то заново

pop cx ; востанавливаем количество строк

mov ax, 8 ;

add bx, ax ; переход на через строку матрицы

add bx, ax

mov ax, 4; переход через элемент матрици mas1

add di, ax

loop \_mm1 ; если не последняя строка, заново

;2- преобразование одномерной матрици

mov cx, 2 ; количество стОЛБЦОВ

mov di,2

mov si,2

\_m\_m1: ;

push cx ; сохраняем

mov cx, m ; для того чтобы ввести количество столбцов

mov bx, 0 ; елемент i=0 j=2 делаю минимальным

mov dx, matrix[bx][si] ;

\_m\_m2:

cmp dx, matrix[bx][si] ;если [i][j]элемент матрицы > max

jg menshe;

mov dx, matrix[bx][si] ;max= matrix[i][j]

menshe: mov ax, 8 ;

add bx, ax ; переход на сл. стобец матрицы

loop \_m\_m2; если не конец строки, то заново

mov mas1[di],dx; mas1[i]=max i не четное

pop cx ; востанавливаем количество строк

mov ax, 4 ;

add si, ax ; переход на сл. столбец матрицы

mov ax, 4

add di, ax; переход на след елемет+1

loop \_m\_m1 ; если не последняя строка, заново

;вывод приеобразованной матрици

lea dx, preobr\_matrix ; преобразованная матрица -

mov ah, 09h

int 21h

xor si,si

mov cx,4

L: outint mas1[si] ; ФУНКЦИЯ ИЗ МОДУЛЯ IO ДЛЯ ЗНАКОВЫХ ЧИСЕЛ

lea dx, probel ; далее выводим пробел

mov ah, 09h

int 21h

add si, 2

loop L

; =================-=-=-=-=-=-=-=-= sravnenie elementov matrici

xor si,si

mov cx,3

L\_1: mov ax,mas1[si]

imul mas1[si+1] ; умножение i го и i+1 элемента

je znak\_necher ; если болше 0 и = то два соседних элемента с одним знаком

add si, 2 ; АДРЕСС= АДРЕСС +2

loop L\_1

;=--=-=-=-=-=-=-=-=-=-=vivod soobhsenii-=-=--=-=-=--=-=

znak\_cher:

lea dx, pustie\_str

mov ah, 09h

int 21h

lea dx, zn\_chereduetsa ;вывод сообщения о чередовании знаков

mov ah, 09h

int 21h

jmp exit

znak\_necher:

lea dx, pustie\_str

mov ah, 09h

int 21h

lea dx, zn\_nechereduetsa ;вывод сообщения о нечеродовании знаков

mov ah, 09h

int 21h

exit:

mov ah, 7h ; задержка

int 21h

mov ah, 4ch ; выход в DOS

int 21h

end start

# Заключение

В данной курсовой работе был изучен теоретический по работе материал с прерываниями DOS, вводом и выводом чисел при помощи подключаемого модуля IO, работы с одномерными и двухмерными массивами, тестирование состояние флагов. Основные принципы и навыки работы с компилятором и отладчиком. Разработана схема алгоритма программы и реализована на языке низкого уровня ассемблер.

# Список литературы

1. Каган Б. М. Электронные вычислительные машины и системы. Учебное пособие для вузов. – М.: Энергоатомиздат, 1991.
2. Ефремов В.Д., Мелехин В.Ф., Дурандин К.П. и др. Вычислительные машины и системы. Учебник для вузов – 2М.: Высшая школа, 1994.
3. Ларионов А.М., Майоров С.А., Новиков Г.И. Вычислительные комплексы, системы, сети.// Учебник для вузов. – Л.: Энергоатомиздат, 1987.
4. Микропроцессоры: в 3-х кн.: Учебник для вузов. /Под ред. Л.И.Преснухина - М.: Высшая школа, 1986.
5. МикроЭВМ: в 8 кн. /Под ред. Л**..**Преснухина. – М.: Высшая школа, 1988.
6. Соловьев Г.Н. Арифметические устройства ЭВМ. – М.: Энергия, 1978.
7. Смирнов А.Д Архитектура ВС./ Учебное пособие для вузов. – М.: Наука.: Гл. ред. Физ.-мат.- лит., 1990.
8. Головкин Б.А. Параллельные вычислительные системы. – М.: Наука: Гл. ред. Физ.-мат. Лит., 1980.
9. Компьютеры на СБИС в 2-х кн.: пер. с японского/ Т.Мотоока, Х.Тонака и др./ - М.: Мир, 1980.