**АННОТАЦИЯ**

Сердце персонального компьютера - микропроцессор, выполняющий арифметические, логические и вспомогательные операции, необходимые для решения задач. Первые микропроцессоры появились в конце 1960-х годов, когда исследователи и разработчики создали интегральную схему (ИС), содержащую различные электронные компоненты на едином кристалле полупроводника. В начале 70-х годов фирма Intel предложила процессор 8008 - представитель первого поколения микропроцессоров.

Каждое семейство процессоров имеет собственный уникальный набор инструкций, используемый для выполнения операций, например, для ввода с клавиатуры, вывода данных на экран и выполнения арифметических вычислений. Этот набор инструкций (машинный язык) слишком сложен для понимания и непосредственного использования для создания программ. Поэтому для процессоров существуют языки ассемблеры, представляющие инструкции машинных языков в более доступной для человеческого понимания символьной форме.

В курсе "машинно-ориентированное" программирование мы изучили язык ассемблера для процессора Intel 8086. Завершающим этапом данного курса является написание расчетно-графической работы.

**Содержание**

Введение

1. Задания к расчетно-графической работе

1. Теоретическая часть

2.1 Уровни языков программирования

2.2 Преимущество языка ассемблера

2.3 Структура программы на языке ассемблера

2.4 Синтаксис языка ассемблера

1. Арифметика

3.1 Алгоритм

3.2 Текст программы

3.3 Листинг кода библиотеки, в которой находятся макросы, вызываемые в данной программе

4.Строки

4.1 Алгоритм

4.2 Текст программы

4.3 Листинг кода библиотеки, в которой находятся макросы, вызываемые в данной программе

Вывод

Список литературы

**ВВЕДЕНИЕ**

Курсовая работа по курсу «Машинно-ориентированное программирование» является завершающим этапом в изучении данной дисциплины.

Одной из основных частей персонального компьютера является микропроцессор, выполняющий арифметические, логические и вспомогательные операции, необходимые для решения задач. Со временем микропроцессоры совершенствовались, так что каждое семейство процессоров имеет собственный уникальный набор инструкций, используемый для выполнения операций, например, для ввода с клавиатуры, вывода данных на экран и выполнения арифметических вычислений. Этот набор инструкций (машинный язык) слишком сложен для понимания и непосредственного использования для создания программ. Поэтому для процессоров существуют языки ассемблеры, представляющие инструкции машинных языков в более доступной для человеческого понимания символьной форме.

В работе рассмотрены целочисленные арифметические операции и операции над строками.

**1. Задания к работе**

1. Арифметика

1.1. Написать модуль на языке Ассемблера для вычисления значения выражения.

1.2. Написать на языке Ассемблера программу корректного ввода исходных данных (с контролем допустимого диапазона) в таблицу и вывода полученного результата в виде таблицы.

2. Строки

2.1. Написать модуль на языке Ассемблера для обработки строк.

2.2. Написать на языке Ассемблера программу корректного ввода исходных данных.

2.4. Произвести тестовые проверки, сделать анализ результатов.

Ввести 2 строки символов. Вставить вторую строку в первую, начиная с заданного символа.

**2.Теоретическая часть**

**2.1 Уровни языков программирования**

Любой язык программирования относится к одному из следующих уровней.

* *Машинный язык –* представляет собой совокупность машинных инструкций, непосредственно исполняемые процессором. Последовательности таких инструкций составляют операционную систему и встроенную в аппаратную часть компьютера программное обеспечение.
* *Низкоуровневые языки программирования (ассемблеры) –* предназначаются для конкретных семейств процессоров. Символьные инструкции этих языков непосредственно соответствуют инструкциям машинных языков, и программы на ассемблере легко транслируются в машинный код.
* *Высокоуровневые языки (С++ или Basic) –* созданы для того, чтобы сосредоточить усилия программистов на решение прикладных задач, не отвлекая их на аппаратные особенности конкретных машин. Команды языков высокого уровня обычно преобразуются в сложные последовательности машинных инструкций.

**2.2 Преимущество языка ассемблера**

Знание и использование языка программирования дает следующие преимущества:

* ясен способ взаимодействия программы с операционной системой, процессором и BIOS;
* доступен выбор способа представления и хранение данных в операционной памяти и во внешних накопителях;
* можно видеть, как процессор получает инструкции, как инструкции выполняются и как обрабатываются данные;
* доступно непосредственное управление внешними устройствами;

Кроме того, язык ассемблера полезен благодаря следующим свойствам:

* программа, написанная на ассемблере, значительно компактнее и быстрее выполняется, чем написанная на языке программирования высокого уровня;
* ассемблер дает возможность решать узкоспециальные задачи, которые трудно, если вообще возможно, решить на языке высокого уровня;
* хотя большинство программистов разрабатывают новые приложения с использованием языков высокого уровня, программы, которые легче сопровождать и модифицировать, или части программ, критичные в скорости выполнения, весьма часто пишутся на ассемблере;
* резидентные программы (находящиеся в памяти во время выполнения других программ) и обработчики прерываний (выполняющие, например, операции ввода/вывода) почти всегда создаются на языке ассемблера.

**2.3 Структура программы на языке ассемблера**

Программа на языке ассемблера представляет собой совокупность блоков памяти, называемых *сегментами памяти. П*рограмма может состоять из одного или нескольких сегментов. Каждый сегмент содержит совокупность предложений языка, каждый из которых занимает отдельную строку кода программы.

Предложения на языке ассемблера бывают 4 типов:

1. Команды или инструкции, представляющие собой символьные аналоги машинных команд. В процессе трансляции эти инструкции преобразуются в соответствующие команды и системы команд микропроцессора.
2. Макрокоманды – оформляемые определенным образом предложения текста программы, замещаемые во время трансляции другими предложениями.
3. Директивы – указания транслятору на выполнение отдельных действий. В директивах нет аналогов в машинном представлении.
4. Строки комментариев – любые последовательности символов, транслятором полностью игнорируются.

**2.4 Синтаксис языка ассемблера**

Для того, что бы транслятор мог предложения языка, они должны формироваться по определенным синтаксическим правилам. Для этого лучше всего использовать формальное описание языка, наподобие грамматики.

Общий формат команд на языке ассемблера.

команда

макрокоманда

директивы

Комментарий

рис.1

Команда

Имя метки

КОП

Операнд1

Операнд1

Комментарий

рис.2

Имя метки – идентификатор, значение которого является адрес 1 байта, предложенного текста программы, кот. он обозначает.

КОП – код операции, мнемоническое обозначение машинной команды или макрокоманды.

Операнды – части команды (макрокоманды) обозначающие объекты над которыми производится действие.

Допустимые символы для написания текста программ:

* все латинские буквы (заглавные и строчные являются эквивалентными);
* цифры 0-9;
* символы ?, \_ , @, $, &;
* разделители и знаки операций.

Предложения языка ассемблера формируется из лексем, представляющих собой синтаксически не разделимые последовательности допустимых символов языка, имеющие смысл для транслятора.

Лексемами являются:

1. Идентификаторы – последовательности допустимых символов, использующиеся для обозначения кодов операций, переменных, констант, меток. Идентификатор может состоять из 1 или нескольких символов. В качестве символов могут использоваться латинский алфавит, цифры, символы ?, \_ , @, $, &. Идентификатор не может начинаться с цифры. Допустимая длина до 255 символов, при этом транслятор воспринимает только 32 первых. Существует возможность указывать транслятору различие заглавных и строчных букв.
2. Цепочки символов – это последовательность символов, заключенные в одинарные или двоичные кавычки.
3. Целые числа – в одной из следующих систем исчисления: 2-ой, 10-ой, 16-ой.

*Отожествление чисел производится по следующим правилам:*

а) 10-е числа не требуют указания каких-либо дополнительных символов (25, 16);

б) 2-е числа требуют, чтобы после них стояла буква b (10011010b)

в) 16-е числа имеют больше условий при записи: во-первых – они состоят как из цифр, так и из символов латинского алфавита; во-вторых – у транслятора могут возникнуть сложности с распознаванием 16-ого числа, так как число может начинаться как с цифры, так и с символа. Для этого 16-е число обязательно заканчивается латинской буквой h (019ch).

Каждое предложение на языке ассемблера содержит описание объекта, над которым или с помощью которого выполняются некоторые действия, эти объекты называются *операндами.*

*Операнды-* это объекты (значение, регистры, ячейки памяти), на которые действуют инструкции или директивы или это объекты, которые уточняют действие инструкций и ли директив.

Операнды могут комбинироваться с арифметически – логическими, побитовыми операторами для расчета некоторого значения или определения ячейки памяти, на которую будет воздействовать команда.

**3. Арифметика**

**3.1 Алгоритм программы**

Блок-схема алгоритма вычисления выражения

НАЧАЛО

КОНЕЦ

макрос позволяющий ввести число с клавиатуры

push ds

push cx

xor cx,cx

cmp cx,datasize

mov pretemp, 5

Сохраняем значения регистров

Проверка типа считываемого числа

Указываем количество знаков для ввода

Mov cx,cs

Mov ds,cx

inSig pretemp ;Считать строку в буфер temp

Инициируем регистр ds

Вызываем макрос InSin

mov,datasiz

pop cx

pop ds

Преобразуем и

восстанавливаем значения

регистров

# ДА

jne

# НЕТ

НАЧАЛО

КОНЕЦ

push dx

Lea dx, Str

mov ah,0ah

int 21h

Сохраняем значения регистров

Вызов прерывания, адрес строки в ds:dx

pop dx

восстанавливаем значения

регистров

**3.2 Текст программы, реализующий алгоритм**

.model small

.386

.stack 100h

.data

str1 db 09h,0d5h ,"========",0B8h,0ah,0dh

db "Enter a ", 0b3h,09h, " ",0b3h,0ah,0dh

db 09h, 0c3h,"========",0b4h,0ah,0dh

db "Enter b ",0b3h,09h, " ",0b3h,0ah,0dh

db 09h, 0c3h,"========",0b4h,0ah,0dh

db "Result :",0b3h,09h, " ",0b3h,0ah,0dh

db 09h, 0d4h,"========",0beh,0ah,0dh, '$'

InBufa DB 7

kola DB ?

a DB ?, ?, ?, ?, ?, ?, ?

InBufb DB 7

kolb DB ?

b DB ?, ?, ?, ?, ?, ?, ?

Error DB "Input data ERROR !!!", 0dh, 0ah, '$'

DivErr DB "Dividing by Zero !!!", 0dh, 0ah, '$'

chA DW 0

chB DW 0

tabl DW 10000, 1000, 100, 10, 1

ResBin DW 0

ResBCD DB '+', '0', ?, ?, ?, ?, 0ah, 0dh, '$'

kol db 3

flag db 0

.code

include math.lib

.startup

cls

locate 0,0

mov ax,dgroup

mov ds,ax

cikl:

mov flag, 0

mov chA, 0

mov chB, 0

mov ResBin, 0

mov bp, offset ResBCD

mov ds:[bp], '+'

mov ds:[bp+1], '0'

mov ds:[bp+2], ?

mov ds:[bp+3], ?

mov ds:[bp+4], ?

mov ds:[bp+5], ?

mov al, 3

mov ah, kol

sub al, ah

imul ax, 7

locate 0, al

push ax

; Вывод таблицы

mov dx,offset str1

mov ah,09h

int 21h

; Ввод чисел

pop ax

push ax

inc al

locate 10, al

mov dx,offset InBufa

mov ah,0ah

int 21h

pop ax

push ax

add al, 3

locate 10, al

mov dx,offset InBufb

mov ah,0ah

int 21h

; Проверка на числа

prov InBufA

prov InBufB

; Проверка на знак

znak a

znak b

; ASCII > BIN

ASCII\_2\_BIN kola, chA

ASCII\_2\_BIN kolb, chB

funtion

; Bin 2 Ascii

mov cx, 5

mov bp, offset tabl

mov si, offset ResBCD

inc si

v:

mov bx, ds:[bp]

mov dx, 0

idiv bx

cmp al, 0

je nul

mov flag, 1

add al, 30h

mov ds:[si], al

inc si

jmp nol

nul: cmp flag, 0

je nol

mov ds:[si], 30h

inc si

nol: mov ax, dx

add bp, 2

loop v

; Выход\_1

pop ax

add al, 5

locate 10, al

mov dx, offset ResBCD

mov ah, 09h

int 21h

locate 0, 35

dec kol

jnz cikl

; Выход

jmp ex

Er: cls

locate 25, 8

mov dx, offset Error

mov ah, 09h

int 21h

jmp ex

DEr: locate 25, 9

mov dx, offset DivErr

mov ah, 09h

int 21h

ex:

.exit(0)

END

**3.3. Листинг кода библиотеки string.lib, в которой находятся макросы, вызываемые в данной программе**

IFDEF @Model

IFIDN @Model, <1>

TINYMODEL TYPEDEF

ENDIF

ENDIF

cls macro

push ax

push bx

push cx

push dx

mov cx,0

mov dh,24

mov dl,79

mov ah,6

mov al,0

mov bh,7

int 10h

pop dx

pop cx

pop bx

pop ax

endm

locate macro col,row

push dx

mov dh,row

mov dl,col

move\_cursor

pop dx

endm

move\_cursor macro

LOCAL ok,vs

push ax

push bx

cmp dh,24

jna ok

sub dh,dh

ok: cmp dl,79

jna vs

sub dl,dl

vs: mov ah,15

int 10h

mov ah,2

int 10h

pop bx

pop ax

endm

prov macro InBuf

local c1

mov bp, offset InBuf

inc bp

mov ch, 0

mov cl, ds:[bp]

dec cl

add bp, 2

c1: cmp byte ptr [ds:[bp]], 30h

jl Er

cmp byte ptr [ds:[bp]], 39h

ja Er

inc bp

loop c1

endm

znak macro per

local m1

cmp per, '-'

je m1

cmp per,'+'

jne Er

m1:

endm

ASCII\_2\_BIN macro kol, res

local p, plus

mov bp, offset kol

mov bx, offset kol

mov ax, 0

mov al, ds:[bx]

add bp, ax

mov ch, 0

mov cl, ds:[bx]

dec cl

mov bx, offset tabl

add bx, 8

mov ax, 0

mov dx, 0

p: mov ax, ds:[bx]

mov dl, ds:[bp]

sub dl, 30h

imul ax, dx

add res, ax

sub bx, 2

sub bp, 1

dec cx

jnz p

; Проверка на знак

mov bp, offset kol

add bp, 1

cmp ds:byte ptr[bp], '-'

jne plus

neg res

plus:

endm

funtion macro

;Задание:

; b / a - 1, если a>b;

;Y = -295, если a=b;

; (a - 235) / b, если a<b;

mov ax, chA

mov bx, chB

cmp ax, bx

jne net

cmp ax, 0

je equal

net:

mov bp, offset a

mov si, offset b

mov dl, ds:[bp]

mov dh, ds:[si]

cmp dl, dh

jl more

ja low

cmp dl, '-'

jne pl

cmp ax, bx

jg more

je equal

ja low

pl: cmp ax, bx

ja more

je equal

jl low

more: cmp al, 0

je DEr

mov ax, chB

mov bx, ax

and bx, 7FFFh

cmp ax, bx

je pluss

mov dx,0FFFFh

jmp did

pluss: mov dx, 0

did: idiv chA

sub ax,1

jmp www

equal: mov ax, 00h

sub ax,295

jmp www

low: cmp bl, 0

je DEr

mov ax,chA

sub ax,235

mov bx, ax

and bx, 7FFFh

cmp ax, bx

je plu2

mov dx,0FFFFh

jmp mnu2

plu2: mov dx, 0

mnu2:

idiv chB

www: mov bx, ax

and bx, 7FFFh

cmp ax, bx

je OK

neg ax

mov ResBCD, '-'

OK: mov ResBin, ax

endm

**4. Строки**

**4.1 Блок-схема алгоритма обработки строки**

НАЧАЛО

КОНЕЦ

ReadStr macro Str – макрос реализующий ввод с клавиатуры строки

push ax

push dx

Lea dx, Str

mov ah,0ah

int 21h

Сохраняем значения регистров

Вызов прерывания, адрес строки в ds:dx

pop cx

pop dx

pop ax

восстанавливаем значения

регистров

НАЧАЛО

КОНЕЦ

ShowStr macro string\_name – макрос выводящий на экран строку

push ax

push dx

Lea dx,ds:string

Mov ah,09

int 21h

Сохраняем значения регистров

Загружает в регистры адрес строки

pop dx

pop ax

восстанавливаем значения

регистров

**4.2 Текст программы, реализующий данный алгоритм:**

.model small

.586

.stack 100h

.data

Mes1 db "Input the first string",0ah,0dh,'$'

Mes2 db "Input the second string",0ah,0dh,'$'

Mes3 db "Input the simbol, from which you want to paste the second string",0ah,0dh,'$'

Str1 db 255

db ?

db 255 dup (0)

Str2 db 255

db ?

db 255 dup (0)

Char1 db 2

db ?

db 0,'$'

Res db "Result is:",0Ah,0Dh,'$'

Str db 255 dup (0)

.code

include string.lib

.startup

mov ax,dgroup

mov ds,ax

cls

locate 0,0

lea dx, Mes1

mov ah,09h

int 21h

lea dx, Str1

mov ah,0ah

int 21h

locate 0, 3

lea dx, Mes2

mov ah,09h

int 21h

lea dx, Str2

mov ah,0ah

int 21h

locate 0, 6

lea dx, Mes3

mov ah,09h

int 21h

lea dx, Char1

mov ah,0ah

int 21h

obrabotka

locate 0, 9

lea dx, Res

mov ah,09h

int 21h

lea dx, Str

mov ah,09h

int 21h

.exit(0)

END

**4.3 Листинг кода библиотеки math.lib, в которой находятся макросы, вызываемые в данной программе**

IFDEF @Model

IFIDN @Model, <1>

TINYMODEL TYPEDEF

ENDIF

ENDIF

cls macro

push ax

push bx

push cx

push dx

mov cx,0

mov dh,24

mov dl,79

mov ah,6

mov al,0

mov bh,7

int 10h

pop dx

pop cx

pop bx

pop ax

endm

locate macro col,row

push dx

mov dh,row

mov dl,col

move\_cursor

pop dx

endm

move\_cursor macro

LOCAL ok,vs

push ax

push bx

cmp dh,24

jna ok

sub dh,dh

ok: cmp dl,79

jna vs

sub dl,dl

vs: mov ah,15

int 10h

mov ah,2

int 10h

pop bx

pop ax

endm

obrabotka macro

mov bp,offset Char1

add bp, 2

mov dl, ds:[bp]

mov bp,offset Str1

add bp, 1

mov cl, ds:[bp]

inc bp

mov si,offset Str2

add si, 1

mov ch, ds:[si]

inc si

mov bx, offset Str

inc bx

lp:

mov al,ds:[bp]

mov ds:[bx],al

cmp al,dl

je m1

inc bx

inc bp

dec cl

jnz lp

m1: inc bx

inc bp

lpo:

mov al,ds:[si]

mov ds:[bx],al

inc bx

inc si

dec ch

jnz lpo

lopa:

mov al,ds:[bp]

mov ds:[bx],al

inc bx

inc bp

dec cl

jnz lopa

inc bx

mov ds:[bx],'$'

endm

**ВЫВОД**

В данной работе были разработаны программы, направленные на выполнение арифметических действий и строковых операций.

Программы были написаны на языке низкого уровня Borland Turbo Assembler(TASM), для процессоров семейства Intel.

В ходе выполнения задания возникали проблемы во время отладки программы, поскольку язык ассемблера TASM имеет сложную структуру.

Были практически закреплены полученные навыки работы с TASM, знание о структуре языка и его синтаксиса.

Программы, разработанные в ходе выполнения данной расчетно-графической работы, характеризуются малым размером выполняемого файла, работай напрямую с аппаратным обеспечением ПК.