Учреждение Образования

«Белорусский Государственный Университет Информатики и Радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра Информатики

Дисциплина: АВС

Отчет

По лабораторным работам № 7-8

# ПРОГРАММИРОВАНИЕ АРИФМЕТИЧЕСКОГО СОПРОЦЕССОРА

Выполнил студент

Группы 253505

Волчецкий А.М.

Минск 2014

### *Лабораторная работа 7.*

Написать программу, находящую решение квадратного уравнения

*ax2 + bx + c = 0*

с помощью сопроцессора.

.286

.model small

.data

a dd ?

b dd ?

c dd ?

D dd ?

x1 dd ?

x2 dd ?

printAMessage db 10,13,'Print A: $'

printBMessage db 10,13,'Print B: $'

printCMessage db 10,13,'Print C: $'

pressAnyKeyMessage db 10,13,'Press any key...$'

answerMessage db 10, 13,'Answer : $'

commaMessage db ', $'

noRootsMessage db 'No roots! $'

errorMessage db 'Coefficients A and B cannot be zero at the same time $'

four dd 4.0

two dd 2.0

.stack 256h

.code

; макрос выводит на экран строку

OutputString macro string

push ax

mov dx, offset string

mov ah, 9

int 21h

pop ax

endm

start:

mov ax, @data

mov ds, ax

;вводим коэфициенты

OutputString printAMessage

call InputNumber

fstp a

OutputString printBMessage

call InputNumber

fstp b

OutputString printCMessage

call InputNumber

fstp c

OutputString answerMessage

call SolveExpression

jmp exit

SolveExpression proc

finit ; инициализируем сопроцессор

;проверим, равно ли a нулю

fld a ;добавляем А в стек

ftst ; сравниваем с нулем

fstsw ax ; копируем значение регистра состояния в ax

sahf ;все, что в регистре ah перейдет во флаги cf, pf, zf, sg, af

jnz findDiscr

;раз дошли сюда--значит a=0

fld c

fchs ; меняем знак у c

;проверим, равен ли b нулю

fld b

ftst

fstsw ax

sahf

jz error ;если a=0 и b=0, то получается: 0\*x^2 + 0\*x + c=0 => c=0 => фигня это

;если a=0 получаем такое уравнение: 0\*x^2 + b\*x + c = 0 => b\*x + c = 0 => x = -c/b

fdivp st(1),st(0) ; делим -c на b

fst x1 ; результат помещаем в x1

call outfloat ; ну и все

jmp exit

error:

OutputString errorMessage

jmp exit

; Дискриминант

; D = sqr(b)-4\*a\*c

findDiscr:

fld b ;Загрузка в стек

fmul b ; возводим b в квадрат

fld a

fmul four ; умножаем a на 4

fmul c; умножаем 4a на c

fsubp st(1), st(0) ; b^2 - 4\*a\*c

fst D ; нашли дискриминант. в переменную D его

ftst

fstsw ax

sahf

jc DiscrLessZero

jz DiscrIsZero

; D > 0

; X1 = (-b+sqrt(D))/(2\*a)

; X2 = (-b-sqrt(D))/(2\*a)

finit

fld two

fmul a ; 2\*a

fld D

fsqrt ; sqrt(D)

fld b

fchs ; -b

fsave two[4] ; сохраняем состояние сопроцессора

frstor two[4] ; восстанавливаем состояние сопроцессора из области памяти two[4]

;находим первый корень уравнения

fadd

fdivr

fst x1

;находим второй корень уравнения

frstor two[4]

fsubr

fdivr

fst x2

; выводим найденные корни

fld x1

call outfloat

OutputString commaMessage

fld x2

call outfloat

jmp endSolveExpression

; D < 0 Нет корней

DiscrLessZero:

OutputString noRootsMessage

jmp endSolveExpression

; D = 0

; X = -b/(2\*a)

DiscrIsZero:

finit

fld two

fmul a ;2\*a

fld b

fdivr ; b/(2\*a)

fchs ; -b/(2\*a)

call outfloat

endSolveExpression:

ret

SolveExpression endp

exit: ; выход из программы

OutputString pressAnyKeyMessage

mov ah, 08h ; жмем любую кнопку

int 21h

mov ax,4c00h ; завершаем работу программы

int 21h

OutputSymbol macro simbol

push ax

push dx

mov dl, simbol

mov ah, 02h

int 21h ; выводит на экран содержимое регистра dl. т.е то, что мы ввели

pop dx

pop ax

endm

InputDigit proc

startInputSymbol:

mov ah, 08h

int 21h

cmp al, 13

je endInputDigit

cmp al, '.'

je printSymbol

cmp al, '-'

je printSymbol

cmp al, 39h

ja startInputSymbol

cmp al, 30h

jb startInputSymbol

symbolToDigit:

OutputSymbol al

sub al, 30h

jmp endInputDigit

printSymbol:

OutputSymbol al

endInputDigit:

ret

InputDigit endp

InputNumber proc

push ax

push dx

push si

mov si, 0 ; если в si хранится 0--значит число положительное

push bp

mov bp, sp

push 10

push 0

fldz ; добавляем 0 в стек сопроцессора

call InputDigit

cmp al, '-'

jne numberIsPositive

mov si, 1 ; значит вводимое число отрицательное

inputIntegerPart:

call InputDigit

numberIsPositive:

cmp al, 13

je checkSign

cmp al, '.'

je pointInputed

mov [bp - 4], al ;сохраняем введенную цифру во временной ячейке

fimul word ptr [bp - 2] ; множим существующее число на 10

fiadd word ptr [bp - 4] ; прибавляем к нему сохраненную ранее цифру

jmp inputIntegerPart

pointInputed:

fld1 ;добавляем в стек сопроцессора 1

inputFractionalPart:

call InputDigit

cmp al, 13

je endInputFractionalPart

mov [bp - 4], al ;сохраняем введенную цифру во временной ячейке

fidiv word ptr [bp - 2] ; получаем отрицательную степень десятки

fld st(0) ; продублируем ее

fimul word ptr [bp - 4] ; умножаем степень на введенную цифру

faddp st(2), st ; добавляем к текущему результату

jmp inputFractionalPart

endInputFractionalPart:

fstp st(0) ; выталкиваем степень десятки из стека

checkSign: ;здесь учитываем знак вводимого числа

test si, si

jz endInputNumber

fchs

endInputNumber:

leave ; восстанавливаем значение bp и sp, что было в начале

pop si

pop dx

pop ax

ret

InputNumber endp

outfloat proc near

push ax

push cx

push dx

; Формируем кадр стэка, чтобы хранить в нём десятку

push bp

mov bp, sp

push 10

push 0

; Проверяем число на знак, и если оно отрицательное,

ftst

fstsw ax

sahf

jnc @of1

; то выводим минус

mov ah, 02h

mov dl, '-'

int 21h

; и оставляем модуль числа.

fchs

; Пояснение далее пойдёт на примере. ; ST(0) ST(1) ST(2) ST(3) ...

; Отделим целую часть от дробной. ; 73.25 ... что-то не наше

@of1: fld1 ; 1 73.25 ...

fld st(1) ; 73.25 1 73.25 ...

; Остаток от деления на единицу даст дробную часть.

fprem ; 0.25 1 73.25 ...

; Если вычесть её из исходного числа, получится целая часть.

fsub st(2), st ; 0.25 1 73 ...

fxch st(2) ; 73 1 0.25 ...

; Сначала поработаем с целой частью. Считать количество цифр будем в CX.

xor cx, cx

; Поделим целую часть на десять,

@of2: fidiv word ptr [bp - 2] ; 7.3 1 0.25 ...

fxch st(1) ; 1 7.3 0.25 ...

fld st(1) ; 7.3 1 7.3 0.25 ...

; отделим дробную часть - очередную справа цифру целой части исходного числа,-

fprem ; 0.3 1 7.3 0.25 ...

; от чатсного оставим только целую часть

fsub st(2), st ; 0.3 1 7 0.25 ...

; и сохраним цифру

fimul word ptr [bp - 2] ; 3 1 7 0.25 ...

fistp word ptr [bp - 4] ; 1 7 0.25 ...

inc cx

; в стэке.

push word ptr [bp - 4]

fxch st(1) ; 7 1 0.25 ...

; Так будем повторять, пока от целой части не останется ноль.

ftst

fstsw ax

sahf

jnz short @of2

; Теперь выведем её.

mov ah, 02h

@of3: pop dx

; Вытаскиваем очередную цифру, переводим её в символ и выводим.

add dl, 30h

int 21h

; И так, пока не выведем все цифры.

loop @of3 ; 0 1 0.25 ...

; Итак, теперь возьмёмся за дробную часть, для начала проверив её существование.

fstp st(0) ; 1 0.25 ...

fxch st(1) ; 0.25 1 ...

ftst

fstsw ax

sahf

jz short @of5

; Если она всё-таки ненулевая, выведем точку

mov ah, 02h

mov dl, '.'

int 21h

; и не более шести цифр дробной части.

mov cx, 36

; Помножим дрообную часть на десять,

@of4: fimul word ptr [bp - 2] ; 2.5 1 ...

fxch st(1) ; 1 2.5 ...

fld st(1) ; 2.5 1 2.5 ...

; отделим целую часть - очередную слева цифру дробной части исходного числа,-

fprem ; 0.5 1 2.5 ...

; оставим от произведения лишь дробную часть,

fsub st(2), st ; 0.5 1 2 ...

fxch st(2) ; 2 1 0.5 ...

; сохраним полученную цифру во временной ячейке

fistp word ptr [bp - 4] ; 1 0.5 ...

; и сразу выведем.

mov ah, 02h

mov dl, [bp - 4]

add dl, 30h

int 21h

; Теперь, если остаток дробной части ненулевой

fxch st(1) ; 0.5 1 ...

ftst

fstsw ax

sahf

; и мы вывели менее шести цифр, продолжим.

loopnz @of4 ; 0 1 ...

; Итак, число выведено. Осталось убрать мусор из стэка.

@of5: fstp st(0) ; 1 ...

fstp st(0) ; ...

; Точнее, стэков.

leave

pop dx

pop cx

pop ax

ret

outfloat endp

END start

### *Лабораторная работа 8.*

Значение аргумента **x** изменяется от ***a*** до ***b*** с шагом ***h***. Для каждого ***x*** найти значения функции Y(x), суммы S(x) и число итераций n, при котором достигается требуемая точность ε = |Y(x)-S(x)|. Результат вывести в виде таблицы. Значения ***a****,* ***b****,* ***h*** и ***ε*** вводятся с клавиатуры.

8. .

start:

mov ax, @data

mov ds, ax

;вводим коэфициенты

OutputString printAMessage

call InputNumber

fstp a

OutputString printBMessage

call InputNumber

fstp b

OutputString printHMessage

call InputNumber

fstp h

OutputString printEMessage

call InputNumber

fstp e

call LetsStart

jmp exit

LetsStart proc

sumLoop:

;обнуляем переменные

fld k

fld zero

fmulp

fstp k

fld n

fld zero

fmulp

fstp n

fld Sx

fld zero

fmulp

fstp Sx

fld Yx

fld zero

fmulp

fstp Yx

;когда a будет больше b мы заканчиваем вычисления

finit

fld b

fcomp a

fstsw ax

sahf

jc endLetsStart

; Посчитаем Y(x)

; Y(X) = (1+2\*x^2)\* exp(x^2)

fld a

fmul a ;x^2

;здесь считается exp(x^2)

fldl2e

fmul

fld st

frndint

fsub st(1), st

fxch st(1)

f2xm1

fld1

fadd

fscale

fstp st(1)

fld a

fmul a ; x^2

fmul two ;2\*x^2

fld1

fadd ; 1+2\*x^2

fmulp ;(1+2\*x^2)\* exp(x^2)

fstp Yx

;------------------------------------------

;Посчитаем S(x)

;S(x) = ((2\*k + 1)\*x^(2\*k))/k!

calcuLateSx:

finit

fld k

fmul two ; 2\*k

fld a

;считаем x^(2\*k)

fyl2x ;Стек FPU теперь содержит: ST(0)=y\*log2(x)

fld st(0) ;Создаем еще одну копию z

frndint ;Округляем ST(0)=trunc(z) | ST(1)=z

fxch st(1);ST(0)=z | ST(1)=trunc(z)

fsub st(0),st(1) ;ST(0)=z-trunc(z) | ST(1)=trunc(z)

f2xm1 ;ST(0)=2\*\*(z-trunc(z))-1 | ST(1)=trunc(z)

fld1 ;ST(0)=1 ST(1)=2\*\*(z-trunc(z))-1 | ST(2)=trunc(z)

faddp st(1),st ;ST(0)=2\*\*(z-trunc(z)) | ST(1)=trunc(z)

fscale ;ST(0)=(2\*\*(z-trunc(z)))\*(2\*\*trunc(z))=2\*\*(z)

fld k

fmul two ; 2\*k

fld1

faddp;2\*k + 1

fmul ;((2\*k + 1)\*x^(2\*k))

fstp temp

;считаем k!

fld1

fld k

fact:

ftst

fstsw ax

sahf

jz endFact

fmul st(1), st(0)

fld1

fsubp st(1), st(0)

jmp fact

endFact:

fstp st(0) ; выталкиваем нолик(наверно)

;заканчиваем считать: (2\*k + 1)\*x^(2\*k))/k!

fld temp

fxch

fdivp

sumSx:

fld Sx

faddp ; суммируем текущее значение S(x) и сумму значений, полученную на предыдущих итерациях

fst Sx

checkE: ; тут мы проверяем, достигли мы введенной точности

fld Yx

fsubp st(1), st(0) ;Y(x) - S(x).

fabs

fcomp e

fstsw ax

sahf

jc incrementH

;инкрементируем k

fld k

fld1

faddp

fstp k

jmp calcuLateSx ; требуемой точности не достигли--считаем S(x) дальше

incrementH:

;увеличиваем a на шаг h

call PrintResult

fld a

fadd h

fstp a

jmp sumLoop

endLetsStart:

ret

LetsStart endp

PrintResult proc

OutputString xResultMessage

fld a

call outfloat

OutputString yxResultMessage

fld Yx

call outfloat

OutputString sxResultMessage

fld Sx

call outfloat

OutputString kResultMessage

fld k

call outfloat

ret

PrintResult endp