МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА»
(БГТУ им. В.Г. Шухова)

ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И УПРАВЛЯЮЩИХ СИСТЕМ

Лабораторная работа №4

по дисциплине: Архитектура вычислительных систем
тема: «Команды передачи управления»

Выполнил: ст. группы ПВ-223
Пахомов Владислав Андреевич

Проверили:
ст. пр. Осипов Олег Васильевич

Белгород 2024 г.

Лабораторная работа №4
Команды передачи управления
Вариант 8

Цель работы: изучение команд перехода для организации циклов и ветвлений, получение навыков создания процедур с аргументами.

Задания для выполнения к работе:

1. Написать программу для вычисления значения арифметического выражения, используя команды условного и безусловного перехода согласно варианту задания. Подобрать набор тестовых данных (не менее 3). При выполнении операций с числами, преобразовывать их к 4-байтовым числам со знаком.
2. Написать программу для вычисления значения арифметического выражения, содержащего функцию. Вычисление функции организовать в виде отдельной подпрограммы по всем правилам, описанным выше. Для обработки массивов использовать команды для работы с циклами и команды условного перехода. Подобрать набор тестовых данных (не менее 3). Результат вывести на экран.

Задание:

8	$a=\begin{cases} x^2+32-xz, & x+y+z>0, \\ z+y, & x+y+z\le 0 \text{ и } x>0, \\ x+y\frac{z}{2}, & x+y+z\le 0 \text{ и } x\le 0 \end{cases}$	x – знаковое однобайтовое y – знаковое 4-байтовое z – беззнаковое 2-байтовое
8	$a=\sum_{i=0}^{m}\sum_{j=0}^{n}\frac{x_i}{k(i,j)}-\frac{y_i^3 x_i i+6}{x_i^2}$, $k(i,j)=\begin{cases} \frac{i}{3}, & i \text{ кратно } 3, \\ 5\frac{i+j}{\lvert i-j\rvert}, & i \text{ не кратно } 3 \end{cases}$	h – беззнаковая переменная размером 2 байта x – массив знаковых 2-байтовых чисел y – массив знаковых 4-байтовых чисел m, n – беззнаковые переменные размером 2 байта

Первая программа:
Условие задания было составлено некорректно, оно приводило к делению на 0. Функция k была изменена:
k = i / 3 + 1, i кратно 3; 5 * (i + j) / (i + 1) + 1, i не кратно 3.

```
.686
.model flat, stdcall
option casemap: none

include windows.inc
include kernel32.inc
include msvcrt.inc
includelib  kernel32.lib
includelib  msvcrt.lib

; Тестовые данные:
; x = -5, y = 5, z = 1, a = 62
; x = -5, y = -5, z = 2, a = -10
; x = 1, y = -5, z = 1, a = -4

.data
    x db 0
    y dd 0
    z dw 0
    input_str db "%hhd %d %hu", 0
    output_str db "%d", 0

.code
start:
    ; Вводим x, y, z
    push offset z
    push offset y
    push offset x
    push offset input_str
    call crt_scanf
    add esp, 4*4

    xor eax, eax   ; Очищаем eax
    movsx edx, x   ; edx = x
    add eax, edx   ; eax = eax + edx = x
    mov edx, y     ; edx = y
    add eax, edx   ; eax = x + y
    movsx edx, z   ; edx = z
    add eax, edx   ; eax = x + y + z

    cmp eax, 0     ; Сравниваем x + y + z с нулём
    ; Если x + y + z > 0, идём к sum_gzero
    jg sum_gzero
    ; Иначе топаем к sum_lezero
    jmp sum_lezero

    sum_gzero:
        movsx eax, x   ; eax = x
        imul eax, eax  ; eax = eax * eax = x ^ 2
        add eax, 32    ; eax = eax + 32 = x ^ 2 + 32
        movsx edx, x   ; edx = x
        movsx ebx, z   ; ebx = z
        imul edx, ebx  ; edx = edx * ebx = x * z
        sub eax, edx   ; eax = eax - edx = x ^ 2 + 32 - x * z
        jmp sum_end

    sum_lezero:
        movsx eax, x  ; eax = x
        cmp eax, 0    ; Сравниваем eax с нулём
        jg x_gzero    ; Если x > 0, топаем к x_gzero
```

```
        ; Иначе - x_lezero
        jmp x_lezero
        x_gzero:
            movsx eax, z ; eax = z
            mov ebx, y   ; ebx = y
            add eax, ebx ; eax = eax + ebx = z + y
            jmp x_end    ; Выходим из условия

        x_lezero:
            mov ebx, 2    ; ebx = 2
            movsx eax, z  ; eax = z
            cdq           ; eax = eax:edx
            idiv ebx      ; eax = eax / ebx = z / 2
            mov ebx, y    ; ebx = y
            imul eax, ebx ; eax = eax * ebx = (z / 2) * y
            movsx ebx, x  ; ebx = x
            add eax, ebx  ; eax = eax + ebx = (z / 2) * y + x
            jmp x_end     ; Выходим из условия

        x_end:
        jmp sum_end ; Выходим из условия

    sum_end:

    ; Выводим результат
    push eax
    push offset output_str
    call crt_printf
    add esp, 4 * 2

    call crt__getch     ; Задержка ввода, getch()
    ; Вызов функции ExitProcess(0)
    push 0      ; Поместить аргумент функции в стек
    call ExitProcess    ; Выход из программы
end start
```

Тестовые данные:

X	y	z	Результат
-5	5	1	62
-5	-5	2	-10
1	-5	1	-4

Вторая программа:

```
.686
.model flat, stdcall
option casemap: none

include windows.inc
include kernel32.inc
include msvcrt.inc
includelib  kernel32.lib
includelib  msvcrt.lib

; Тестовые данные:
; 1 5 5
; 1 2 3 4 5
; 5 4 3 2 1
; -295

; 1 2 6
; 10 99
; 32 1
```

```
; 215

; 1 5 4
; 1 24 31 4 89
; 29 -13 -12 42 123
; -555785

.data
    i dd 0
    j dd 0
    result dd 0
    k_tmp dd 0
    h dw 0
    x dw 512 dup(0)
    y dw 512 dup(0)
    m dw 0
    n dw 0
    str_fmt db "%u", 0
    str_output_fmt db "%u", 13, 10, 0
    input_str db "%hu %hu %hu", 0
    output_str db "%d", 13, 10, 0

.code

; void input (short* a, int n)
input proc
    pushad
    mov esi, [esp + 4 + 8 * 4]
    mov ecx, [esp + 8 + 8 * 4]
    xor ebx, ebx

input_j_loop:
    cmp ebx, ecx
    je input_j_exit
    lea edi, [esi + ebx * 2]

    pushad
    push edi
    push offset str_fmt
    call crt_scanf
    add esp, 8
    popad

    inc ebx
    jmp input_j_loop
input_j_exit:
    popad
    ret 8
input endp

; void output (int* a, int n)
output proc
    pushad
    mov esi, [esp + 4 + 8 * 4]
    mov ecx, [esp + 8 + 8 * 4]
    xor ebx, ebx

output_j_loop:
    cmp ebx, ecx
    je output_j_exit
    lea edi, [esi + ebx * 2]

    pushad
    mov ax, [edi + 0]
    movsx eax, ax
    push eax
    push offset str_output_fmt
```

```
    call crt_printf
    add esp, 8
    popad

    inc ebx
    jmp output_j_loop
output_j_exit:
    popad
    ret 8
output endp

; Внимание! Функцию поменял, бикос в оригинальной может случиться деление на 0, что довольно
грустно.
; i/3 + 1 и 5 * (i + j) / (i + 1) + 1
k proc
    pushad
    mov ebp, [esp + 4 + 8 * 4] ; ebp = i
    mov esi, [esp + 8 + 8 * 4] ; esi = j
    mov eax, ebp ; eax = i
    cdq          ; Расширяем eax
    mov ebx, 3
    idiv ebx     ; eax = i / 3, edx = i % 3
    cmp edx, 0
    je k_i_divides_three
    jmp k_i_not_divides_three
k_i_divides_three:
    add eax, 1
    mov k_tmp, eax ; k_tmp = eax = i / 3
    jmp k_i_end
k_i_not_divides_three:
    mov ebx, ebp ; ebx = i
    add ebx, 1   ; ebx = i + 1
    mov eax, ebp ; eax = i
    add eax, esi ; eax = i + j
    cdq          ; Расширение eax
    idiv ebx     ; eax = eax / ebx = (i + j) / (i + 1)
    mov ebx, 5   ; ebx = 5
    imul ebx     ; eax = eax * ebx = 5 * (i + j) / (i + 1)
    add eax, 1   ; eax = eax + 1 = 5 * (i + j) / (i + 1) + 1
    mov k_tmp, eax ; k_tmp = eax = 5 * (i + j) / (i + 1) + 1

    jmp k_i_end
k_i_end:
    popad
    mov eax, k_tmp
    ret 8
k endp

start:
    ; Вводим x, y, z
    push offset n
    push offset m
    push offset h
    push offset input_str
    call crt_scanf
    add esp, 4*4

    movsx eax, m
    push eax
    push offset x
    call input

    movsx eax, m
    push eax
    push offset y
    call input
```

```
cycle_i:
    mov eax, i
    movsx ebx, m
    cmp eax, ebx
    je cycle_i_end

    mov j, 0
cycle_j:
    mov eax, j
    movsx ebx, n
    cmp eax, ebx
    je cycle_j_end

    mov esi, i
    mov ax, x[esi * 2]
    cwde
    mov ebp, eax ; Записываем в ebp = xi

    mov esi, i
    mov ax, y[esi * 2]
    cwde
    mov esi, eax ; Записали в esi = yi

    push j
    push i
    call k
    mov ebx, eax ; ebx = k(i, j)
    mov eax, ebp ; eax = xi
    cdq
    idiv ebx     ; eax = eax / ebx = xi / k(i, j)
    mov ecx, eax  ; ecx = eax = xi / k(i, j)

    mov ebx, ebp  ; ebx = xi
    imul ebx, ebx ; ebx = ebx * ebx = xi ^ 2
    mov eax, esi  ; eax = esi = yi
    imul eax, esi ; eax = eax * yi = yi^2
    imul eax, esi ; eax = eax * yi = yi^3
    imul eax, ebp ; eax = eax * xi = yi^3 * xi
    imul eax, i   ; eax = eax * i  = yi^3 * xi * i
    add eax, 6    ; eax = eax + 6  = yi^3 * xi * i + 6
    cdq
    idiv ebx  ; eax = eax / ebx = (yi^3 * xi * i + 6) / (xi ^ 2)
    sub ecx, eax ; ecx = xi / k(i, j) - (yi^3 * xi * i + 6) / (xi ^ 2)
    add result, ecx ; Складываем наши вычисления в результат

    inc j ; j++
    jmp cycle_j ; Топаем к началу цикла
cycle_j_end:
    inc i
    jmp cycle_i
cycle_i_end:

    push result
    push offset output_str
    call crt_printf
    add esp, 8

    call crt__getch     ; Задержка ввода, getch()
    ; Вызов функции ExitProcess(0)
    push 0      ; Поместить аргумент функции в стек
    call ExitProcess    ; Выход из программы
end start
```

Тестовые данные:

h	x	y	m	n	Результат
1	1 2 3 4 5	5 4 3 2 1	5	5	-295
1	10 99	32 1	2	6	215
1	1 24 31 4 89	29 -13 -12 42 123	5	4	-555785

Вывод: в ходе лабораторной изучили команды перехода для организации циклов и ветвлений, получили навыки создания процедур с аргументами.