МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №3
по дисциплине «ОЭВМиС»
Тема: Представление и обработка целых чисел. Организация ветвящихся процессов

Студент гр. 9383 _____ Крейсманн К.В.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2020

Цель работы.

Изучить представление целых чисел, научиться их обрабатывать, познакомиться с организаций ветвящихся процессов.

Задание:

Разработать на языке Ассемблера программу, которая по заданным целочисленным значениям параметров a, b, i, k вычисляет:

a) значения функций i1 = f1(a,b,i) и i2 = f2(a,b,i);

b) значения результирующей функции res = f3(i1,i2,k),

где вид функций f1 и f2 определяется из табл. 2, а функции f3 - из табл.3 по цифрам шифра индивидуального задания (n1,n2,n3), приведенным в табл.4. Значения a, b, i, k являются исходными данными, которые должны выбираться студентом самостоятельно и задаваться в процессе исполнения программы в режиме отладки. При этом следует рассмотреть всевозможные комбинации параметров a, b и k, позволяющие проверить различные маршруты выполнения программы, а также различные знаки параметров a и b.

Замечания:

1) при разработке программы нельзя использовать фрагменты, представленные на ЯВУ, в частности, для ввода-вывода данных. Исходные данные должны вводиться, а результаты контролироваться в режиме отладки;

2) при вычислении функций f1 и f2 вместо операции умножения следует использовать арифметический сдвиг и, возможно, сложение;

3) при вычислении функций f1 и f2 нельзя использовать процедуры;

4) при разработке программы следует минимизировать длину кода, для чего, если надо, следует преобразовать исходные выражения для вычисления функций.

f1 = {15-2*i , при a>b

{3*i+4, при a<=b

f2 = {-(6*I +8), при a>b

$$f3 = \begin{cases} 9 - 3*(i-1), & \text{при } a<=b \\ |i1|+|i2|, & \text{при } k<0 \\ \max(6,|i1|), & \text{при } k>=0 \end{cases}$$

Ход работы:

Для считывания чисел была разработана процедура input. В этой процедуре считывается код символа, если он является кодом знака минуса, то в регистр dx заносится значение 1, которое затем используется, для преобразования числа в отрицательное. Если является кодом цифры, то цифра записывается в число и так продолжается до нажатия enter. Если является кодом клавиши enter то считывание завершается и в нужную переменную кладется значение.

Процедура Input вызывается в цикле 4 раза для получения значений a ,b ,i ,k.

Первая функция вычисляется следующим образом:

сравниваются значения a и b, если a<=b, то происходит переход на метку, иначе в переменную результата первой функции resf1 кладется значение 15, затем в ax заносится значение i и оно умножается на 2 с помощью сдвига влево. Затем из переменной resf1 вычитается полученное в ax. Если произошел переход на метку, то в resf1 заносится 4, затем в ax заносится i и умножается на 3 сдвигом влево и сложением, после этого к resf1 прибавляется ax.

Для уменьшение количества команд во второй функции были преобразованы выражения –(6*i+8) = -8-6*i , 9-3*(i-1) = 12-3*i. Функция вычисляется следующим образом:

сравниваются значения a и b, если a<=b, то происходит переход на метку, иначе в переменную результата второй функции resf2 кладется значение -8, а в ax кладется значение i, затем оно умножается на 6, путем двойного сдвига влево и прибавления выражения 2*I, которое вычисляется одним сдвигом влево. После этого из resf2 вычитается значение, хранящееся в ax. Если произошел переход на метку, то в resf2 заносится значение 12, затем в ax заносится значение i и оно

умножается на 3 путем сдвига влево и прибавления i. После этого из resf2 вычитается значение, лежащее в ax.

Третья функция вычисляется следующим образом:

Сравниваем resf1 и resf2 с нулем, если они отрицательны, то делаем их положительными с помощью команды neg. Затем сравниваем k с 0, если k отрицательно, то переходим на метку, где в resf3 кладем сумму модулей resf1 и resf2. Если k>=0, то сравниваем модуль resf1 с 6 и устанавливаем максимум из них в resf3.

Тестирование программы:

1. Входные данные: 3 4 5 2
   Значение resf1 = 19
   Значение resf2 = -3
   Значение resf3 = 19
2. Входные данные: 3 3 -4 1
   Значение resf1 = -8
   Значение resf2 = 24
   Значение resf3 = 8
3. Входные данные: 3 3 -4 -100
   Значение resf1 = -8
   Значение resf2 = 24
   Значение resf3 = 32
4. Входные данные: -2 4 20 6
   Значение resf1 = 64
   Значение resf2 = -48
   Значение resf3 = 64
5. Входные данные: -2 4 20 -12
   Значение resf1 = 64
   Значение resf2 = -48
   Значение resf3 = 112

6. Входные данные: 10 5 5 3

   Значение resf1 = 5

   Значение resf2 = -38

   Значение resf3 = 6

Содержимое файла lr3.asm представлено в приложении А.

Содержимое файла l3.lst представлено в приложении Б.

Вывод:

Изучено представление целых чисел, получены навыки работы с целыми числами и ветвящимися процессами.

Приложение А

lr.asm:

```
.186
DOSSEG
  .model small
  .STACK 100h
  .DATA
a dw 0
b dw 0
i dw 0
k dw 0
resf1 dw 0
resf2 dw 0
resf3 dw 0
  .CODE
begin:
     mov ax,@DATA
     mov ds,ax
     mov di,offset a
     mov cx,4                    ;счетчик , 4, т.к. нужно ввести 4 значения
inputNumbers:                    ;ввод значений
     mov dx,0                    ;для знака
     call input                  ;вызов процедуры которая считывает число
     cmp dx,0                    ;сравниваем dx с 0
     je ContinueInputNumbers     ;если равен 0 то переходим на метку
     neg Word Ptr es:[di]        ;делаем отрицательной переменную
ContinueInputNumbers:
     inc di
     inc di
     loop inputNumbers
;Вычисляем первую функцию: 15 - 2*i if a>b  else 3*i+4
     mov ax,a
     cmp ax,b
     mov ax,i                    ;заносим в ax i
     JNG MarkF1                  ;a<=b
     shl ax,1                    ;умножаем ax на 2 сдвигом
     mov resf1,15                ;заносим в результат 15
     sub resf1,ax                ;resf1 = 15-2*i
     jmp F2                      ;переходим в вычислению второй функции
  MarkF1:                        ;a<=b
     mov resf1,4                 ;заносим в результат 4
     shl ax,1                    ;умножаем ax на 2 сдвигом
     add ax,i                    ;прибавляем к ax i
     add resf1,ax                ;resf1  = 4+3*i
F2:                              ;Вычисляем вторую функцию:-(6*i+8) if a>b else 9-3*(i-1)
     mov ax,a                    ;заносим в ax a
     cmp ax,b                    ;сравниваем a и b
     mov ax,i                    ;заносим в ax значение i
     mov dx,ax                   ;копируем ax в dx
```

```
    JNG MarkF2          ;a<=b
    mov resf2,-8        ;заносим в результат -8
    shl dx,1            ;в dx: i*2
    shl ax,2            ;в ax: i*4
    add ax,dx           ;ax = i*4+i*2 =i*6
    sub resf2,ax        ;resf2 = -8 - i*6 = -(6*i+8)
    jmp F3              ;переходим к вычислению третьей функции
MarkF2:                 ;a<=b
    mov resf2,12        ;заносим в ax 12
    shl ax,1            ;умножаем ax на 2 сдвигом
    add ax,i            ;ax = 3*i
    sub resf2,ax        ; resf2=12 - 3*i = 9 - 3*(i-1)
F3:                     ;Вычисляем третью функцию |resf1| + |resf2| if k<0 else max(6,|resf1|)
    mov ax,RESF1        ;заносим в ax resf1
    mov bx,RESF2        ;заносим в bx resf2
    cmp ax,0            ;сравниваем ax с 0
    jnl MarkF3_1        ;если ax>= 0 то переходим на метку
    neg ax              ;иначе делаем положительным
MarkF3_1:
    cmp bx,0            ;сравниваем bx c0
    jnl MarkF3_2        ;если bx>=0 то переходим на метку
    neg bx              ;иначе делаем положительным
MarkF3_2:
    mov cx,k            ;заносим в cx значение k
    cmp cx,0            ;сравниваем с 0
    jl MarkF3_3         ;если k<0 переходим на метку
    cmp ax,6            ;сравниваем ax с 6
    jl Set_6            ; если 6 больше то установить значение 6
    mov resf3,ax        ; иначе установить значение |resf1|
    jmp Endprog         ; переходим в конец программы
Set_6:
    mov resf3,6         ; устанавливаем 6
    jmp Endprog         ; переходим в конец программы
MarkF3_3:               ;если k<0
    add ax,bx           ;складываем ax и bx
    mov resf3,ax        ;заносим в resf3
Endprog:
    mov ah,4ch
    int 21h
INPUT PROC NEAR         ;процедура ввода числа
    mov bx,10           ;для увеличения разряда
    push cx             ;сохраняем значение cx
Mark1:
    mov ah,1h
    int 21h
    cmp al,2dh          ; сравниваем с кодом минуса
    jne Continue        ; если не минус переходим на метку
    mov dx,1            ; если dx=1 то число затем будет преобразовано в отрицательное
    jmp Mark1
Continue:
    sub al,30h          ; вычитаем чтобы получить цифру а не код символа
    mov ah,0            ; расширяем до слова
```

```
    mov cx,ax              ; первая цифра в cx
Mark2:
    mov ah,1h
    int 21h
    cmp al,0dh             ;сравнивем с кодом enter
    je EndInput            ;если enter то заканчиваем ввода числа
    sub al,30h             ;получаем цифру
    mov ah,0h              ;расширяем до слова
    xchg ax,cx             ;в cx следующее число, в ax предыдущее
    push dx                ;сохраняем dx в стек
    mul bx                 ;умножаем предыдущее число на 10
    pop dx                 ;вытаскиваем dx
    add cx,ax              ;cx = ax*10 + cx
    jmp Mark2
EndInput:                  ;конец ввода
    mov ax,seg a           ;кладем в ax начало сегмента с переменными
    mov es, ax             ;переносим его в es
    mov WORD PTR es:[di],cx   ;переносим значение из cx в переменную
    pop cx
    ret
input endp
   end
end begin
```

Приложение Б

l3.lst:

```
Microsoft (R) Macro Assembler Version 5.10              10/22/20 01:28:0
                                                        Page     1-1


                        .186
                        DOSSEG
                         .model small
                         .STACK 100h
                         .DATA
0000  0000              a dw 0
0002  0000              b dw 0
0004  0000              i dw 0
0006  0000              k dw 0
0008  0000              resf1 dw 0
000A  0000              resf2 dw 0
000C  0000              resf3 dw 0
                         .CODE
0000                    begin:
0000  B8 ---- R             mov ax,@DATA
0003  8E D8                 mov ds,ax
0005  BF 0000 R             mov di,offset a
0008  B9 0004                   mov cx,4                ;счет
                        чик , 4, т.к. нужно ввести 4 з
                        начения
000B                    inputNumbers:                   ;ввод
                         значений
000B  BA 0000                   mov dx,0                ;для ?
                        ?нака
000E  E8 00B5 R             call input              ;вызо
                        в процедуры которая считы
                        вает число
0011  83 FA 00                  cmp dx,0                ;срав
                        ниваем dx с 0
0014  74 03                 je ContinueInputNumbers     ;если
                         равен 0 то переходим на ме
                        тку
0016  26: F7 1D             neg Word Ptr es:[di]        ;дела
                        ем отрицательной перемен?
                        ?ую
0019                    ContinueInputNumbers:
0019  47                    inc di
001A  47                    inc di
001B  E2 EE                 loop inputNumbers
                        ;Вычисляем первую функцию
                        : 15 - 2*i if a>b  else 3*i+4
001D  A1 0000 R             mov ax,a
0020  3B 06 0002 R          cmp ax,b
0024  A1 0004 R             mov ax,i                ;заносим
                         в ax i
```

```
0027  7E 0F                    JNG MarkF1          ;a<=b
0029  D1 E0                    shl ax,1            ;умножае
                        м ax на 2 сдвигом
002B  C7 06 0008 R 000F        mov resf1,15        ;заносим
                         в результат 15
0031  29 06 0008 R             sub resf1,ax        ;resf1 = 15-2*i
0035  EB 11 90                      jmp F2              ;переход
                        им в вычислению второй фу
                        кции
Microsoft (R) Macro Assembler Version 5.10          10/22/20 01:28:0
                                                    Page     1-2


0038                            MarkF1:             ;a<=b
0038  C7 06 0008 R 0004        mov resf1,4         ;заносим
                         в результат 4
003E  D1 E0                    shl ax,1            ;умножае
                        м ax на 2 сдвигом
0040  03 06 0004 R             add ax,i            ;прибавл
                        яем к ax i
0044  01 06 0008 R             add resf1,ax        ;resf1  = 4+3*i
0048                            F2:                 ;Вычисля
                        ем вторую функцию:-(6*i+8) if a>
                        b else 9-3*(i-1)
0048  A1 0000 R                mov ax,a            ;заносим
                         в ax a
004B  3B 06 0002 R             cmp ax,b            ;сравнив
                        аем a и b
004F  A1 0004 R                mov ax,i            ;заносим
                         в ax значение i
0052  8B D0                    mov dx,ax           ;копируе
                        м ax в dx
0054  7E 14                    JNG MarkF2          ;a<=b
0056  C7 06 000A R FFF8        mov resf2,-8        ;заносим
                         в результат -8
005C  D1 E2                    shl dx,1            ;в dx: i*2
005E  C1 E0 02                      shl ax,2            ;в ax: i*4
0061  03 C2                    add ax,dx           ;ax = i*4+i*2 =
                        i*6
0063  29 06 000A R             sub resf2,ax        ;resf2 = -8 - i
                        *6 = -(6*i+8)
0067  EB 11 90                      jmp F3              ;переход
                        им к вычислению третьей ф
                        нкции
006A                            MarkF2:             ;a<=b
006A  C7 06 000A R 000C        mov resf2,12        ;заносим
                         в ax 12
0070  D1 E0                    shl ax,1            ;умножае
                        м ax на 2 сдвигом
0072  03 06 0004 R             add ax,i            ;ax = 3*i
0076  29 06 000A R             sub resf2,ax        ; resf2=12 - 3*
                        i = 9 - 3*(i-1)
```

```
007A                    F3:                     ;Вычисля
                        ем третью функцию |resf1| + |res
                        f2| if k<0 else max(6,|resf1|)
007A  A1 0008 R             mov ax,RESF1        ;заносим
                         в ax resf1
007D  8B 1E 000A R          mov bx,RESF2        ;заносим
                         в bx resf2
0081  3D 0000                   cmp ax,0        ;сравнив
                        аем ax с 0
0084  7D 02                 jnl MarkF3_1        ;если ax>=
                        0 то переходим на метку
0086  F7 D8                 neg ax              ;иначе д
                        лаем положительным
0088                    MarkF3_1:
0088  83 FB 00                cmp bx,0         ;сравнив
 Microsoft (R) Macro Assembler Version 5.10          10/22/20 01:28:0
                                Page     1-3


                        аем bx c0
008B  7D 02                 jnl MarkF3_2        ;если bx>=0
                         то переходим на метку
008D  F7 DB                 neg bx              ;иначе д
                        лаем положительным
008F                    MarkF3_2:
008F  8B 0E 0006 R          mov cx,k            ;заносим
                         в cx значение k
0093  83 F9 00                 cmp cx,0        ;сравнив
                        аем с 0
0096  7C 14                 jl MarkF3_3         ;если k<0
                        ереходим на метку
0098  3D 0006                   cmp ax,6        ;сравнив
                        аем ax с 6
009B  7C 06                 jl Set_6            ; если 6 б
                        ольше то установить значе
                        ние 6
009D  A3 000C R             mov resf3,ax        ; иначе у
                        становить значение |resf1|
00A0  EB 0F 90                  jmp Endprog     ; перехо
                        им в конец программы
00A3                    Set_6:
00A3  C7 06 000C R 0006     mov resf3,6         ; устана
                        ливаем 6
00A9  EB 06 90                  jmp Endprog     ; перехо
                        им в конец программы
00AC                    MarkF3_3:               ;если k<0
00AC  03 C3                 add ax,bx           ;складыв
                        аем ax и bx
00AE  A3 000C R             mov resf3,ax        ;заносим
                         в resf3
00B1                    Endprog:
```

```
00B1  B4 4C             mov ah,4ch
00B3  CD 21             int 21h
00B5                    INPUT PROC NEAR         ;процеду
                        ра ввода числа
00B5  BB 000A               mov bx,10           ;для уве
                        личения разряда
00B8  51                push cx             ;сохраня
                        ем значение cx
00B9                    Mark1:
00B9  B4 01             mov ah,1h
00BB  CD 21             int 21h
00BD  3C 2D             cmp al,2dh          ; сравни
                        ваем с кодом минуса
00BF  75 05             jne Continue        ; если не
                        минус переходим на метку
00C1  BA 0001               mov dx,1            ; если dx=1
                        то число затем будет прео
                        бразовано в отрицательно
                        
00C4  EB F3             jmp Mark1
00C6                    Continue:
00C6  2C 30             sub al,30h          ; вычита
Microsoft (R) Macro Assembler Version 5.10      10/22/20 01:28:0
                                                Page    1-4


                        м чтобы получить цифру а
                        е код символа
00C8  B4 00             mov ah,0            ; расшир
                        ем до слова
00CA  8B C8             mov cx,ax           ; первая
                        цифра в cx
00CC                    Mark2:
00CC  B4 01             mov ah,1h
00CE  CD 21             int 21h
00D0  3C 0D             cmp al,0dh          ;сравнив
                        ем с кодом enter
00D2  74 0D             je EndInput         ;если enter
                        то заканчиваем ввода чис
                        а
00D4  2C 30             sub al,30h          ;получае
                        м цифру
00D6  B4 00             mov ah,0h           ;расширя
                        ем до слова
00D8  91                xchg ax,cx          ;в cx след
                        ующее число, в ax предыдуще
                        е
00D9  52                push dx             ;сохраня
                        ем dx в стек
00DA  F7 E3             mul bx              ;умножае
                        м предыдущее число на 10
```

```
00DC  5A                  pop dx                ;вытаски
                          ваем dx
00DD  03 C8               add cx,ax             ;cx = ax*10 + c
                          x
00DF  EB EB               jmp Mark2
00E1                      EndInput:             ;конец в
                          ода
00E1  B8 ---- R           mov ax,seg a          ;кладем
                          ax начало сегмента с пере
                          енными
00E4  8E C0               mov es, ax            ;перенос
                          им его в es
00E6  26: 89 0D           mov WORD PTR es:[di],cx   ;перенос
                          им значение из cx в перемен
                          ную
00E9  59                  pop cx
00EA  C3                  ret
00EB                      input endp
                          end
Microsoft (R) Macro Assembler Version 5.10          10/22/20 01:28:0
                                                    Symbols-1


Segments and Groups:

                N a m e         Length  Align  Combine Class

DGROUP . . . . . .  . . . .     GROUP
 _DATA . . . . . .  . . . .     000E    WORD   PUBLIC      'DATA'
 STACK . . . . . .  . . . .     0100    PARA   STACK       'STACK'
_TEXT . . . . . .  . . . .      00EB    WORD   PUBLIC      'CODE'

Symbols:

                N a m e         Type    Value   Attr

A . . . . . .  . . . . . .      L WORD  0000    _DATA

B . . . . . .  . . . . . .      L WORD  0002    _DATA
BEGIN . . . . . .  . . . .      L NEAR  0000    _TEXT

CONTINUE . . . . . .  . . .     L NEAR  00C6    _TEXT
CONTINUEINPUTNUMBERS . . . . . .        L NEAR  0019    _TEXT

ENDINPUT . . . . . .  . . .     L NEAR  00E1    _TEXT
ENDPROG . . . . . .  . . . .    L NEAR  00B1    _TEXT

F2 . . . . . .  . . . . . .     L NEAR  0048    _TEXT
F3 . . . . . .  . . . . . .     L NEAR  007A    _TEXT

I . . . . . .  . . . . . .  L WORD      0004    _DATA
INPUT . . . . . .  . . . .      N PROC  00B5    _TEXT   Length = 0036
```

```
INPUTNUMBERS . . . . . .      L NEAR      000B   _TEXT

K . . . . . .          L WORD      0006   _DATA

MARK1 . . . . . .      L NEAR      00B9   _TEXT
MARK2 . . . . . .      L NEAR      00CC   _TEXT
MARKF1 . . . . . .     L NEAR      0038   _TEXT
MARKF2 . . . . . .     L NEAR      006A   _TEXT
MARKF3_1 . . . . . .   L NEAR      0088   _TEXT
MARKF3_2 . . . . . .   L NEAR      008F   _TEXT
MARKF3_3 . . . . . .   L NEAR      00AC   _TEXT

RESF1 . . . . . .      L WORD      0008   _DATA
RESF2 . . . . . .      L WORD      000A   _DATA
RESF3 . . . . . .      L WORD      000C   _DATA

SET_6 . . . . . .      L NEAR      00A3   _TEXT

@CODE . . . . . .      TEXT  _TEXT
@CODESIZE . . . . . .          TEXT  0
@CPU . . . . . .       TEXT  259
@DATASIZE . . . . . .  TEXT  0
@FILENAME . . . . . .          TEXT  l3
@VERSION . . . . . .   TEXT  510
```



```
121 Source  Lines
121 Total   Lines
 41 Symbols

47870 + 449148 Bytes symbol space free

 0 Warning Errors
 0 Severe  Errors
```