

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

ОТЧЕТ
по лабораторной работе №3
по дисциплине «Архитектура ЭВМ и систем»
Тема: Представление и обработка целых чисел. Организация
ветвящихся процессов.

Студент гр. 9383

Моисейченко К.А.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2020

Цель работы.

Изучить представление и обработку целых чисел на языке Ассемблер.
Научиться строить программы с условными переходами.

Задание.

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

- а) значения функций $i1 = f1(a,b,i)$ и $i2 = f2(a,b,i)$;
- б) 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.

Исходные данные.

Вариант 11

$$/ - (4 * i + 3), \text{ при } a > b$$

$f1 = <$

$$\backslash 6 * i - 10, \text{ при } a \leq b$$

$$/ 2 * (i + 1) - 4, \text{ при } a > b$$

$f2 = <$

$$\backslash 5 - 3 * (i + 1), \text{ при } a \leq b$$

$$/ \min(|i1|, 6), \text{ при } k = 0$$

$f5 = <$

$$\backslash |i1| + |i2|, \text{ при } k \neq 0$$

Ход работы.

Была разработана программа, которая вычисляет значение функции по заданным целочисленным параметрам.

Исходные и выходные данные записываются в сегмент данных. Правильность записи была проверена с помощью отладчика.

Для подсчета значений функции были использованы следующие операнды:

add – для суммирования

sub – для вычитания

shl – для логического сдвига влево, что равнозначно умножению на два

Результаты записывались по заранее заданным адресам переменных i1, i2 и res.

Для реализации условных переходов были использованы следующие операнды:

сmp – для сравнения двух чисел. При использовании данного операнда его результат записывается с помощью выставления соответствующих флагов.

jb – условный переход, срабатывающий, если левый аргумент выражения в сmp был больше второго.

jl – условный переход, срабатывающий, если левый аргумент выражения в сmp был меньше второго.

jmp – безусловный переход. Используется, если для перехода к следующему адресу не нужно делать дополнительных проверок.

Исходный код и листинг программы представлены в приложении А.

Тестирование.

1. $a = 1, b = 2, i = 3, k = 4 \Rightarrow i1 = 8, i2 = -7, res = 15$
2. $a = 2, b = 1, i = 3, k = 4 \Rightarrow i1 = -15, i2 = 4, res = 19$
3. $a = 1, b = 2, i = 3, k = 0 \Rightarrow i1 = 8, i2 = -7, res = 6$
4. $a = 2, b = 1, i = -1, k = 0 \Rightarrow i1 = 1, i2 = 5, res = 1$

Выводы.

Было изучено представление и обработка целых чисел на языке Ассемблер. Была построена программа с условными переходами, которая считает значения функций с заданными целочисленными параметрами.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Файл lr3.asm:

```
AStack SEGMENT STACK
    DW 32 DUP(?)
AStack ENDS

DATA SEGMENT
a      DW      1
b      DW      2
i      DW      3
k      DW      4
i1     DW      ?
i2     DW      ?
res     DW      ?
DATA ENDS

CODE SEGMENT
    ASSUME CS:CODE, DS:DATA, SS:AStack

Main PROC FAR
    mov ax, DATA
    mov ds, ax

f2:
    mov ax, a
    cmp ax, b
    jg f2_1      ;if a>b
    mov ax, i
    shl ax, 1    ;ax = 2*ax
    mov bx, ax   ;bx = 2*ax
    shl ax, 1    ;ax = 4*ax
    add ax, bx   ;ax = 6*ax
    sub ax, 10   ;ax = ax - 10
    mov i1, ax
    jmp f6
f2_1:
```

```

    mov ax, i
    shl ax, 1
    shl ax, 1
    add ax, 3
    neg ax
    mov i1, ax
    jmp f6_1

```

f6:

```

    mov ax, i
    add ax, 1          ; ax = i + 1
    mov bx, ax         ; bx = i + 1
    shl ax, 1          ; ax = 2 * (i + 1)
    add ax, bx         ; ax = 3 * (i + 1)
    mov bx, 5
    sub bx, ax         ; bx = 5 - (1 + i) * 3
    mov i2, bx
    jmp f5

```

f6_1:

```

    mov ax, i
    add ax, 1          ; ax = i + 1
    shl ax, 1          ; ax = 2 * (i + 1)
    sub ax, 4          ; ax = ax - 4
    mov i2, ax
    jmp f5

```

f5:

```

    mov ax, k
    cmp ax, 0
    je f5_cmp_6        ; k = 0
    jmp f5_sum_1       ; k != 0

```

f5_cmp_6: ; проверяем i1 на знак

```

    mov bx, i1
    cmp bx, 0
    jl f5_neg
    jmp f5_cmp_6_1

```

f5_neg: ; если i1 < 0, меняем знак по модулю

```

neg bx
jmp f5_cmp_6_1

f5_cmp_6_1:                ;основная функция сравнения
mov cx, 6
cmp bx, cx
j1 res_i1
jmp res_6

f5_sum_1:                  ;проверка на положительность i1
mov bx, i1
cmp bx, 0
j1 f5_neg_sum
jmp f5_sum_2

f5_sum_2:                  ;проверка на положительность i2
mov cx, i2
cmp cx, 0
j1 f5_neg_sum_2
jmp f5_res_sum

f5_neg_sum:
neg bx
jmp f5_sum_2

                                ;две функции для смены знака

f5_neg_sum_2:
neg cx
jmp f5_res_sum

f5_res_sum:                ;сложение
mov ax, bx
add ax, cx
mov res, ax
jmp f_end

res_6:
mov res, 6
jmp f_end

```

```

;запись результата функции при k = 0

res_i1:
    mov res, bx
    jmp f_end

f_end:
    mov ah, 4ch
    int 21h

Main     ENDP
CODE     ENDS
        END Main

```

Файл lr3.lst:

Microsoft (R) Macro Assembler Version 5.10
06:01:5

12/10/20

Page

1-1

```

0000          AStack SEGMENT STACK
0000 0020[      DW 32 DUP(?)
      ????
      ]

0040          AStack ENDS

0000          DATA SEGMENT
0000 0001      a      DW      1
0002 0002      b      DW      2
0004 0003      i      DW      3
0006 0004      k      DW      4
0008 0000      i1     DW      ?
000A 0000      i2     DW      ?
000C 0000      res    DW      ?
000E          DATA ENDS

0000          CODE SEGMENT
          ASSUME CS:CODE, DS:DATA, SS:AStack

0000          Main PROC FAR
0000 B8 ---- R      mov ax, DATA
0003 8E D8          mov ds, ax

0005          f2:
0005 A1 0000 R      mov ax, a
0008 3B 06 0002 R   cmp ax, b

```



```

000C 7F 14                jg f2_1                ;if a>b
000E A1 0004 R            mov ax, i
0011 D1 E0                shl ax, 1                ;ax = 2*ax
0013 8B D8                mov bx, ax                ;bx = 2*ax
0015 D1 E0                shl ax, 1                ;ax = 4*ax
0017 03 C3                add ax, bx                ;ax = 6*ax
0019 2D 000A              sub ax, 10                ;ax = ax - 10
001C A3 0008 R            mov i1, ax
001F EB 13 90              jmp f6
0022                      f2_1:
0022 A1 0004 R            mov ax, i
0025 D1 E0                shl ax, 1
0027 D1 E0                shl ax, 1
0029 05 0003              add ax, 3
002C F7 D8                neg ax
002E A3 0008 R            mov i1, ax
0031 EB 19 90              jmp f6_1

0034                      f6:
0034 A1 0004 R            mov ax, i
0037 05 0001              add ax, 1                ; ax = i + 1
003A 8B D8                mov bx, ax                ; bx = i + 1
003C D1 E0                shl ax, 1                ; ax = 2 * (i + 1)
003E 03 C3                add ax, bx                ; ax = 3 * (i + 1)
0040 BB 0005              mov bx, 5
0043 2B D8                sub bx, ax                ; bx = 5 - (1 + i) * 3
0045 89 1E 000A R          mov i2, bx
Microsoft (R) Macro Assembler Version 5.10

```

12/10/20

06:01:5

Page

1-2

```

0049 EB 12 90              jmp f5
004C                      f6_1:
004C A1 0004 R            mov ax, i
004F 05 0001              add ax, 1                ; ax = i + 1
0052 D1 E0                shl ax, 1                ; ax = 2 * (i + 1)
0054 2D 0004              sub ax, 4                ; ax = ax - 4
0057 A3 000A R            mov i2, ax
005A EB 01 90              jmp f5
005D                      f5:
005D A1 0006 R            mov ax, k
0060 3D 0000              cmp ax, 0
0063 74 03                je f5_cmp_6                ;k = 0
0065 EB 1C 90              jmp f5_sum_1                ;k != 0

0068                      f5_cmp_6:                ;проверяем i1 на знак
0068 8B 1E 0008 R          mov bx, i1
006C 83 FB 00              cmp bx, 0
006F 7C 03                jl f5_neg
0071 EB 06 90              jmp f5_cmp_6_1

0074                      f5_neg:                ;если i1 < 0, меняем
                                ак по модулю
0074 F7 DB                neg bx

```

ЗН

```

0076 EB 01 90                jmp f5_cmp_6_1

0079                f5_cmp_6_1:                ;основная функция сравн
ения
0079 B9 0006                mov cx, 6
007C 3B D9                cmp bx, cx
007E 7C 37                jl res_i1
0080 EB 2C 90                jmp res_6

0083                f5_sum_1:                ;проверка на положитель
ность i1
0083 8B 1E 0008 R          mov bx, i1
0087 83 FB 00                cmp bx, 0
008A 7C 0F                jl f5_neg_sum
008C EB 01 90                jmp f5_sum_2
008F                f5_sum_2:
008F 8B 0E 000A R          mov cx, i2                ;проверка на
положительно
ность i2
0093 83 F9 00                cmp cx, 0
0096 7C 07                jl f5_neg_sum_2
0098 EB 0A 90                jmp f5_res_sum

009B                f5_neg_sum:
009B F7 DB                neg bx
009D EB F0                jmp f5_sum_2

                                ;две функции для сме
ны знака
009F                f5_neg_sum_2:
009F F7 D9                neg cx
00A1 EB 01 90                jmp f5_res_sum

```

Microsoft (R) Macro Assembler Version 5.10
06:01:5

12/10/20

Page

1-3

```

00A4                f5_res_sum:                ;сложение
00A4 8B C3                mov ax, bx
00A6 03 C1                add ax, cx
00A8 A3 000C R          mov res, ax
00AB EB 11 90                jmp f_end

00AE                res_6:
00AE C7 06 000C R 0006    mov res, 6
00B4 EB 08 90                jmp f_end

                                ;запись результата
функции при k = 0
00B7                res_i1:
00B7 89 1E 000C R          mov res, bx
00BB EB 01 90                jmp f_end

00BE                f_end:

```

```

00BE B4 4C          mov ah, 4ch
00C0 CD 21          int 21h

```

```

00C2                Main      ENDP
00C2                CODE      ENDS
                        END Main

```

Microsoft (R) Macro Assembler Version 5.10 12/10/20
06:01:5

Symbols-1

Segments and Groups:

Class	N a m e	Length	Align	Combine
ASTACK	0040	PARA	STACK
CODE	00C2	PARA	NONE
DATA	000E	PARA	NONE

Symbols:

	N a m e	Type	Value	Attr
A	L WORD	0000	DATA
B	L WORD	0002	DATA
F2	L NEAR	0005	CODE
F2_1	L NEAR	0022	CODE
F5	L NEAR	005D	CODE
F5_CMP_6	L NEAR	0068	CODE
F5_CMP_6_1	L NEAR	0079	CODE
F5_NEG	L NEAR	0074	CODE
F5_NEG_SUM	L NEAR	009B	CODE
F5_NEG_SUM_2	L NEAR	009F	CODE
F5_RES_SUM	L NEAR	00A4	CODE
F5_SUM_1	L NEAR	0083	CODE
F5_SUM_2	L NEAR	008F	CODE
F6	L NEAR	0034	CODE
F6_1	L NEAR	004C	CODE
F_END	L NEAR	00BE	CODE
I	L WORD	0004	DATA
I1	L WORD	0008	DATA
I2	L WORD	000A	DATA
K	L WORD	0006	DATA
MAIN	F PROC	0000	CODE
Length = 00C2				
RES	L WORD	000C	DATA
RES_6	L NEAR	00AE	CODE
RES_I1	L NEAR	00B7	CODE

@CPU	TEXT	0101h
@FILENAME	TEXT	lr3
@VERSION	TEXT	510

Microsoft (R) Macro Assembler Version 5.10
06:01:5

12/10/20

Symbols-2

123	Source	Lines
123	Total	Lines
32	Symbols	

48068 + 459192 Bytes symbol space free

0	Warning	Errors
0	Severe	Errors