

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

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

Студент гр. 0382

Кривенцова Л.С.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2021

Цель работы.

Изучение основ Ассемблера, адресации и работы с числами: научиться организовывать ветвящиеся процессы на языке Ассемблера, реализовать простой алгоритм на языке программирования Ассемблер.

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 $res = f3(i1, i2, k)$,

Вариант 9:

$$f1 = \begin{cases} -(4*i+3), & \text{при } a > b \end{cases}$$

$$f2 = \begin{cases} 6*i - 10, & \text{при } a \leq b \end{cases}$$

$$f2 = \begin{cases} 20 - 4i, & \text{при } a > b \end{cases}$$

$$f2 = \begin{cases} -(6*i - 4), & \text{при } a > b \end{cases}$$

$$f2 = \begin{cases} 3*(i+2), & \text{при } a \leq b \end{cases}$$

$$f2 = \begin{cases} 3*(i+2), & \text{при } a \leq b \end{cases}$$

$$f3 = \begin{cases} |i1| + |i2|, & \text{при } k < 0 \end{cases}$$

$$f3 = \begin{cases} \max(6, |i1|), & \text{при } k \geq 0 \end{cases}$$

$$f3 = \begin{cases} \max(6, |i1|), & \text{при } k \geq 0 \end{cases}$$

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 .

Выполнение работы.

Таблица 3.3 Команды условного перехода		
Условные переходы без знаков		
Мнемоника	Статус флагов	Описание
JA/JNBE	$(CF \text{ or } ZF) = 0$	выше/не ниже не равно
JAЕ/JNB	$CF = 0$	выше или равно/не ниже
JB/JNAE	$CF = 1$	ниже/не выше не равно
JBE/JNA	$(CF \text{ or } ZF) = 1$	ниже или равно/не выше
JC	$CF = 1$	перенос
JE/JZ	$ZF = 1$	равно/ноль
JNC	$CF = 0$	нет переноса
JNE/JNZ	$ZF = 0$	не равно/не ноль
JNP/JPO	$PF = 0$	нет четности/нечетное
JP/JPE	$PF = 1$	четность/четное
Условные переходы со знаком		
JG/JNLE	$((SF \text{ xor } OF) \text{ or } ZF) = 0$	больше/не меньше не равно
JGE/JNL	$(SF \text{ xor } OF) = 0$	больше или равно/не меньше
JL/JNGE	$(SF \text{ xor } OF) = 1$	меньше/не больше не равно
JLE/JNG	$((SF \text{ xor } OF) \text{ or } ZF) = 1$	меньше или равно/не больше
JNO	$OF = 0$	нет переполнения
JNS	$SF = 1$	нет знака (неотрицательное)
JO	$OF = 1$	переполнение
JS	$SF = 1$	знак (отрицательное)

В начале создаётся три сегмента: данных, кода и стэка (*DATA*, *CODE*, *AStack*). Их метки записываются в соответствующие им регистры (*ASSUME CS:CODE*, *DS:DATA*, *SS:AStack*). Тело программы находится в сегменте кода (*Main*), а переменные объявляются в сегменте данных (*a*, *b*, *i*, *k*, *i1*, *i2*, *result*). В главной процедуре переменные инициализируются значением 0.

С помощью инструкции *CMP* (с условными переходами *kg* и *jle*) сравниваются значения *a* и *b*, вызывается соответствующий вариант функции *f1*. После выполнения тела функции с помощью тех же инструкции и переходов проверяется значение переменной *k*. Т.к. если значение неотрицательное, то программа переходит к выполнению заключительной функции (второго варианта *f3*. В таком случае пропускается этап *f2*, т.к. переменная *i2* не задействована при вычислении результата, и вычислять её нерационально). Все остальные переходы идут закономерно и не требуют пояснений.

Также написаны вспомогательные функции *i1abs* и *i2abs* для получения абсолютного значения переменных *i1* и *i2*. Кроме них используется также

функция *case*, чтобы разделить случаи (во втором варианте *f3* результат зависит от сравнения).

Исходный код программы см. в приложении А.

Листинговый файл программы см. в приложении В.

Вывод.

Изучены основы Ассемблера, адресации и работы с числами: получены навыки организации ветвящихся процессы на языке Ассемблера, реализован простой алгоритм на языке программирования Ассемблер.

ТЕСТИРОВАНИЕ

Таблица 1. Результат тестирования.

№ т.	Входные данные	Результат	Комментарий
1	a = 1 b = 2 i = 3 k = 4	i1 = 8 result = 8	Программа работает верно
2	a = 2 b = 1 i = 0 k = 3	i1 = -3 result = 6	Программа работает верно
3	a = 4 b = 0 i = 1 k = -1	i1 = -7 i2 = -2 result = 9	Программа работает верно
4	a = 2 b = 8 i = 2 k = -100	i1 = 2 i2 = 12 result = 14	Программа работает верно

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММ

Файл lb3.ASM

```
AStack SEGMENT STACK
    DW 12 DUP(?)
AStack ENDS
```

```
DATA SEGMENT
a DW 0
b DW 0
i DW 0
k DW 0
i1 DW 0
i2 DW 0
result DW 0
DATA ENDS
```

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

```
Main PROC FAR
    push DS
    sub AX,AX
    push AX
    mov AX,DATA
    mov DS,AX

    ;Entering data
    mov a,1
    mov b,2
    mov i,3
    mov k,4
```

```
    mov AX, a
    cmp AX, b
    jg f1_first
```

```
f1_second:
    mov AX, i
    shl AX, 1 ; = 2i
    add AX, i ; = 3i
    shl AX, 1 ; = 6i
    mov i1, -10
    add i1, AX ; = -10 + 6i
    mov AX, k
    cmp AX, 0
    jge f3_second
```

```
f2_second:
    mov AX, i
    shl AX, 1 ; = 2i
    add AX, i ; = 3i
    mov i2, 6
    add i2, AX ; = 3i + 6
```

```

i2abs:
    neg i2 ; = |i2|
js i2abs

ilabs:
    neg i1
js ilabs ; = |i1|

f3_first:
    js i2abs
    mov AX,i1
    add AX,i2 ; = |i1| + |i2|
    mov result, AX
    ret

f1_first:
    mov AX, i
    shl AX, 1 ; = 2i
    shl AX, 1 ; = 4i
    mov i1, -3
    sub i1, AX ; = -3 - 4i
    mov AX, k
    cmp AX, 0
    jge f3_second

f2_first:
    mov AX, i
    shl AX, 1 ; = 2i
    shl AX, 1 ; = 4i
    add AX, i ; = 5i
    add AX, i ; = 6i
    mov i2, 4
    sub i2, AX ; = 4 - 6i
    jmp f3_first

f3_second:
    js ilabs ; = |i1|
    mov AX, i1
    cmp AX, 6
    jge case
    mov result, 6
    ret

case:
    mov result, AX
    ret

Main ENDP
CODE ENDS
END Main

```

ПРИЛОЖЕНИЕ В
ДИАГНОСТИЧЕСКОЕ СООБЩЕНИЕ

Файл lb3.lst

19:05:01

Page

1-1

```

0000                                AStack SEGMENT STACK
0000  000C[                          DW 12 DUP(?)
                                ????
                                ]

0018                                AStack ENDS


0000                                DATA SEGMENT
0000  0000                                a DW 0
0002  0000                                b DW 0
0004  0000                                i DW 0
0006  0000                                k DW 0
0008  0000                                i1 DW 0
000A  0000                                i2 DW 0
000C  0000                                result DW 0
000E                                DATA ENDS


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


0000                                Main PROC FAR
0000  1E                                push DS
0001  2B C0                                sub AX,AX
0003  50                                push AX
0004  B8 ---- R                                mov AX,DATA
0007  8E D8                                mov DS,AX


                                ;Entering data
0009  C7 06 0000 R 0001                                mov a,1
000F  C7 06 0002 R 0002                                mov b,2
0015  C7 06 0004 R 0003                                mov i,3
001B  C7 06 0006 R 0004                                mov k,4

```

```

0021  A1 0000 R      mov AX, a
0024  3B 06 0002 R      cmp AX, b
0028  7F 49          jg fl_first

002A          fl_second:
002A  A1 0004 R      mov AX, i
002D  D1 E0          shl AX, 1 ; = 2i
002F  03 06 0004 R      add AX, i ; = 3i
0033  D1 E0          shl AX, 1 ; = 6i
0035  C7 06 0008 R FFF6      mov i1, -10
003B  01 06 0008 R      add i1, AX ; = -10 + 6i
003F  A1 0006 R      mov AX, k
0042  3D 0000          cmp AX, 0
0045  7D 60          jge f3_second

0047          f2_second:
0047  A1 0004 R      mov AX, i
004A  D1 E0          shl AX, 1 ; = 2i
004C  03 06 0004 R      add AX, i ; = 3i
0050  C7 06 000A R 0006      mov i2, 6

```

19:05:01

Page

1-2

```

0056 01 06 000A R          add i2, AX ; = 3i + 6

005A                                i2abs:
005A F7 1E 000A R          neg i2 ; = |i2|
005E 78 FA                                js i2abs

0060                                ilabs:
0060 F7 1E 0008 R          neg i1
0064 78 FA                                js ilabs ; = |i1|

0066                                f3_first:
0066 78 F2                                js i2abs
0068 A1 0008 R          mov AX,i1
006B 03 06 000A R          add AX,i2 ; = |i1| + |i2|
006F A3 000C R          mov result, AX
0072 CB                                ret

0073                                f1_first:
0073 A1 0004 R          mov AX, i
0076 D1 E0                                shl AX, 1 ; = 2i
0078 D1 E0                                shl AX, 1 ; = 4i
007A C7 06 0008 R FFFD          mov i1, -3
0080 29 06 0008 R          sub i1, AX ; = -3 - 4i
0084 A1 0006 R          mov AX, k
0087 3D 0000                                cmp AX, 0
008A 7D 1B                                jge f3_second

008C                                f2_first:
008C A1 0004 R          mov AX, i
008F D1 E0                                shl AX, 1 ; = 2i
0091 D1 E0                                shl AX, 1 ; = 4i
0093 03 06 0004 R          add AX, i ; = 5i
0097 03 06 0004 R          add AX, i ; = 6i

```

```

009B  C7 06 000A R 0004      mov i2, 4
00A1  29 06 000A R          sub i2, AX ; = 4 - 6i
00A5  EB BF                  jmp f3_first

00A7                                f3_second:
00A7  78 B7                  js ilabs ; = |i1|
00A9  A1 0008 R              mov AX, i1
00AC  3D 0006                cmp AX, 6
00AF  7D 07                  jge case
00B1  C7 06 000C R 0006      mov result, 6
00B7  CB                    ret

00B8                                case:
00B8  A3 000C R              mov result, AX
00BB  CB                    ret

00BC                                Main ENDP
00BC                                CODE ENDS
                                END Main

```

19:05:01

Symbols-1

Segments and Groups:

N a m e	Length	Align	Combine Class
ASTACK	0018	PARA	STACK
CODE	00BC	PARA	NONE
DATA	000E	PARA	NONE

Symbols:

N a m e	Type	Value	Attr
A	L WORD	0000	DATA
B	L WORD	0002	DATA
CASE	L NEAR	00B8	CODE
F1_FIRST	L NEAR	0073	CODE
F1_SECOND	L NEAR	002A	CODE
F2_FIRST	L NEAR	008C	CODE
F2_SECOND	L NEAR	0047	CODE
F3_FIRST	L NEAR	0066	CODE
F3_SECOND	L NEAR	00A7	CODE
I	L WORD	0004	DATA
I1	L WORD	0008	DATA
I1ABS	L NEAR	0060	CODE
I2	L WORD	000A	DATA
I2ABS	L NEAR	005A	CODE
K	L WORD	0006	DATA

MAIN F PROC 0000 CODE Length =
00BC

RESULT L WORD 000C DATA

@CPU TEXT 0101h

@FILENAME TEXT 1b3

@VERSION TEXT 510

103 Source Lines

103 Total Lines

25 Symbols

47996 + 461311 Bytes symbol space free

0 Warning Errors

0 Severe Errors