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 и систем»

ТЕМА: ПРЕДСТАВЛЕНИЕ И ОБРАБОТКА ЦЕЛЫХ ЧИСЕЛ. ОРГАНИЗАЦИЯ ВЕТВЯЩИХСЯ ПРОЦЕССОВ.

Студент гр.0382 _____ Диденко Д.В.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2021

Цель работы.

Изучить условные переходы и арифметические операции на ассемблере.

Задание.

Вариант 4.

Функции:

f1:

$$f1 = \begin{cases} 15-2*i, & \text{при } a>b \\ 3*i+4, & \text{при } a<=b \end{cases}$$

f2:

$$f5 = \begin{cases} 20 - 4*i, & \text{при } a>b \\ -(6*I - 6), & \text{при } a<=b \end{cases}$$

f3:

$$f4 = \begin{cases} \min(|i1 - i2|, 2), & \text{при } k<0 \\ \max(-6, -i2), & \text{при } k>=0 \end{cases}$$

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.

Основные теоретические положения.

Есть 2 типа выполнения условий в ассемблере:

Прыжок без условия (или «безусловный прыжок») — выполняется инструкцией JMP. Выполнение данной инструкции часто включает в себя передачу управления в адрес инструкции, которая не следует за выполняемой в настоящее время инструкцией. Результатом передачи управления может быть выполнение нового набора инструкций или повторное выполнение текущих инструкций.

Прыжок с условием (или «условный прыжок») — выполняется с помощью инструкций типа J<условие> и зависит от самого условия. Условные инструкции, изменяя значение смещения в регистре IP, передают управление, прерывая последовательный поток выполнения кода.

Инструкция CMP

Инструкция CMP (от англ. «COMPARE») сравнивает два операнда. Фактически, она выполняет операцию вычитания между двумя операндами для проверки того, равны ли эти операнды или нет. Используется вместе с инструкцией условного прыжка.

Синтаксис инструкции CMP:

CMP <назначение>, <источник>.

Инструкция CMP сравнивает два числовых поля. Операнд назначения может находиться либо в регистре, либо в памяти. Исходным операндом (источник) могут быть константы, регистры или память.

Прыжок без условия

Безусловный прыжок выполняется инструкцией JMP, которая включает в себя имя метки, куда следует перебросить точку выполнения программы:

JMP *<label>*

Прыжок с условием

Если при выполнении операции условного прыжка выполняется обозначенное условие, то точка выполнения программы переносится в указанную инструкцию. Существует множество инструкций условного прыжка, в зависимости от условия и данных.

Команды условного перехода, использующиеся после команд сравнения операндов со знаком представлены на рис.1.

Рис.1.

Мнемоника	*Описание*	*Состояние флагов*
JG	Переход, если больше (*левоп* > *правоп*)	SF = OF и ZF = 0
JNLE	Переход, если не меньше или равно (синоним команды JG)	SF = OF и ZF= 0
JGE	Переход, если больше или равно (*левоп* >= *правоп*)	SF = 0 или ZF = 1
JNL	Переход, если не меньше (синоним команды JGE)[4]	SF = 0 или ZF = 1
JL	Переход, если меньше (*левоп* < *правоп*)	SF ≠ OF и ZF = 0
JNGE	Переход, если не больше или равно (синоним команды JL)	SF ≠ OF и ZF = 0
JLE	Переход, если меньше или равно (*левоп* <= *правоп*)	SF ≠ OF или ZF = 1
JNG	Переход, если не больше (синоним команды JBE)	SF ≠ OF или ZF = 1

Команды условного перехода, использующиеся после команд сравнения беззнаковых операндов представлены на рис.2.

Рис.2.

Мнемоника	*Описание*	*Состояние флагов*
JA	Переход, если выше[1] (*левоп* > *правоп*)	CF = 0 и ZF = 0
JNBE	Переход, если не ниже или равно (синоним команды JA)	CF = 0 и ZF = 0
JAE	Переход, если выше или равно (*левоп* >= *правоп*)	CF = 0 или ZF = 1
JNB	Переход, если не ниже (синоним команды JAE)[2]	CF = 0 или ZF = 1
JB	Переход, если ниже (*левоп* < *правоп*)	CF = 1 и ZF = 0
JNAE	Переход, если не выше или равно (синоним команды JB)	CF = 1 и ZF = 0
JBE	Переход, если ниже или равно (*левоп* <= *правоп*)	CF = 1 или ZF = 1
JNA	Переход, если не выше (синоним команды JBE)[3]	CF = 1 или ZF = 1

Выполнение работы.

Результаты тестирования программы lab3.exe представлены в табл. 1.

Таблица 1 – Тестирование программы lab3.exe.

№ п/п	Входные данные	Вывод	Результат
1.	a = 6 b = 5 i = 2 k = -1	res = 1	Программа работает верно
2.	a = 6 b = 5 i = 2 k = 2	res = -6	Программа работает верно

3.	a = 5 b = 5 i = 2 k = -1	res = 2	Программа работает верно
4.	a = 5 b = 5 i = 2 k = 2	res = 6	Программа работает верно

Выводы.

Изучены условные переходы и арифметические операции на ассемблере.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lab3.asm

```
; Стек программы

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
res DW 0
tst DW 6
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,0
 mov b,0
 mov i,0
 mov k,0

 mov AX, a
 cmp AX, b
```

```
 JG f1_over
 JLE f1_under
f1_over:
 mov AX, i
 shl AX, 1; = 2i
 mov i1, 15
 sub i1, AX; = 15 - 2i
 JMP f2_over
f1_under:
 mov AX, i
 shl AX, 1; = 2i
 add AX, i; = 3i
 add AX, 4 ; = 3i + 4
 mov i1, AX
 JMP f2_under
f2_over:
 mov AX, i
 shl AX, 1 ; = 2i
 shl AX, 1 ; = 4i
 mov i2, 20
 sub i2, AX ; = 20 - 4i
 JMP f3_choice
f2_under:
 mov AX, i
 shl AX, 1; = 2i
 shl AX, 1; = 4i
 add AX, i; = 5i
 add AX, i; = 6i
 mov i2, 6
 sub i2, AX
 JMP f3_choice
f3_choice:
 mov AX, k
 cmp AX, 0
 JL f3_under
 JMP f3_over

f3_under:
 mov AX,i1
```

```
 sub AX,i2; = i1 - i2
 cmp AX, 0
 JGE positive
 NEG AX; abs AX
 positive:
 cmp AX, 2
 JB bigger
 mov res, 2
 ret

f3_over:
 mov AX, i2
 NEG AX; -i2
 cmp AX, -6
 JGE bigger
 mov res, -6
 ret

bigger:
 mov res, AX
 ret

Main ENDP
CODE ENDS
 END Main
```

Название файла: LAB3.LST

```
'Microsoft (R) Macro Assembler Version 5.10                 10/26/21
25:42:4
                                                            Page
1-1


                        ; Стек программы

 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                    res DW 0
000E  0006                    tst DW 6
0010                    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 0000    mov a,0
000F  C7 06 0002 R 0000    mov b,0
0015  C7 06 0004 R 0000    mov i,0
001B  C7 06 0006 R 0000    mov k,0

0021  A1 0000 R          mov AX, a
0024  3B 06 0002 R             cmp AX, b

0028  7F 02                    JG f1_over
002A  7E 12                    JLE f1_under
002C                    f1_over:
002C  A1 0004 R          mov AX, i
002F  D1 E0                    shl AX, 1; = 2i
0031  C7 06 0008 R 000F    mov i1, 15
0037  29 06 0008 R             sub i1, AX; = 15 - 2i
003B  EB 13 90                 JMP f2_over
```

```
003E                    f1_under:
003E  A1 0004 R          mov AX, i
0041  D1 E0                    shl AX, 1; = 2i
0043  03 06 0004 R             add AX, i; = 3i
0047  05 0004                  add AX, 4 ; = 3i + 4
004A  A3 0008 R          mov i1, AX
004D  EB 15 90                 JMP f2_under
'Microsoft (R) Macro Assembler Version 5.10                    10/26/21
25:42:4
                                                               Page
1-2


0050                    f2_over:
0050  A1 0004 R          mov AX, i
0053  D1 E0                    shl AX, 1 ; = 2i
0055  D1 E0                    shl AX, 1 ; = 4i
0057  C7 06 000A R 0014        mov i2, 20
005D  29 06 000A R             sub i2, AX ; = 20 - 4i
0061  EB 1D 90                 JMP f3_choice
0064                    f2_under:
0064  A1 0004 R          mov AX, i
0067  D1 E0                    shl AX, 1; = 2i
0069  D1 E0                    shl AX, 1; = 4i
006B  03 06 0004 R             add AX, i; = 5i
006F  03 06 0004 R             add AX, i; = 6i
0073  C7 06 000A R 0006        mov i2, 6
0079  29 06 000A R             sub i2, AX
007D  EB 01 90                 JMP f3_choice
0080                    f3_choice:
0080  A1 0006 R          mov AX, k
0083  3D 0000                  cmp AX, 0
0086  7C 03                    JL f3_under
0088  EB 1B 90                 JMP f3_over

008B                    f3_under:
008B  A1 0008 R          mov AX,i1
008E  2B 06 000A R             sub AX,i2; = i1 - i2
0092  3D 0000                  cmp AX, 0
0095  7D 02                    JGE positive
```

```
0097  F7 D8                NEG AX; abs AX
0099                  positive:
0099  3D 0002              cmp AX, 2
009C  72 18                JB bigger
009E  C7 06 000C R 0002    mov res, 2
00A4  CB              ret

00A5                  f3_over:
00A5  A1 000A R        mov AX, i2
00A8  F7 D8                NEG AX; -i2
00AA  3D FFFA              cmp AX, -6
00AD  7D 07                JGE bigger
00AF  C7 06 000C R FFFA    mov res, -6
00B5  CB              ret

00B6                  bigger:
00B6  A3 000C R        mov res, AX
00B9  CB              ret

00BA                  Main ENDP
00BA                  CODE ENDS
                       END Main
```


Segments and Groups:

N a m e	Length	Align	Combine	Class
ASTACK	0018	PARA	STACK	
CODE	00BA	PARA	NONE	
DATA	0010	PARA	NONE	

Symbols:

N a m e	Type	Value	Attr
A	L WORD	0000	DATA

```
B  . . . . . .  L WORD   0002  DATA
BIGGER . . . . . .  L NEAR   00B6  CODE

F1_OVER  . . . . . .  L NEAR   002C  CODE
F1_UNDER . . . . . .  L NEAR   003E  CODE
F2_OVER  . . . . . .  L NEAR   0050  CODE
F2_UNDER . . . . . .  L NEAR   0064  CODE
F3_CHOICE  . . . . . .  L NEAR   0080  CODE
F3_OVER  . . . . . .  L NEAR   00A5  CODE
F3_UNDER . . . . . .  L NEAR   008B  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   =
00BA

POSITIVE . . . . . .  L NEAR   0099  CODE

RES  . . . . . .  L WORD   000C  DATA

TST  . . . . . .  L WORD   000E  DATA

@CPU . . . . . .  TEXT  0101h
@FILENAME  . . . . . .  TEXT  lab3
@VERSION . . . . . .  TEXT  510

'Microsoft (R) Macro Assembler Version 5.10                  10/26/21
25:42:4


Symbols-2
          100 Source  Lines
          100 Total   Lines
           26 Symbols
```

```
  47978 + 461329 Bytes symbol space free

      0 Warning Errors
0 Severe  Errors
```