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
Вариант 3

Студентка гр. 0382 _____ Деткова А.С.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2021

Цель работы.

Изучить представление и обработку целых чисел. Научиться писать программы с ветвящимся алгоритмом.

Задание.

Вариант 3. Шифр задания — 1.4.3.

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.

Функци варианта 3:

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

$$f2 = \begin{cases} -(6*i - 4), & \text{при } a>b \\ 3*(i+2), & \text{при } a<=b \end{cases}$$

$$f3 = \begin{cases} / |i1 + i2|, \text{ при } k=0 \\ \backslash \min(i1,i2), \text{ при } k/=0 \end{cases}$$

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.

Основные теоретические сведения.

Используются следующие условные переходы:

- JZ — переход делается, если равно нулю (проверяет флаг ZF)
- JNZ — переход делается, если не равно нулю (проверяет флаг ZF)
- JG — переход делается, если выше (проверяет флаги ZF, SF, OF)
- JLE— переход делается, если меньше или равно (проверяет флаги ZF, SF, OF)
- JL— переход делается, если меньше (проверяет флаги SF, OF)
- JGE — переход делается, если выше или равно (проверяет флаги SF, OF)

Также используется безусловный переход JMP - переход осуществляется в любом случае.

Выполнение работы.

Сегмент стека.

Под сегмент стека отведено 12 слов в памяти (24 байта).

Сегмент данных.

A — исходные данные, слово в памяти, равно 0; *B* — исходные данные, слово, равно 0; *I* — исходные данные, слово, равно 4; *K* — слово, равно 0; *I1* — результат выполнения функции *f1*, слово, равно 0; *I2* — результат выполнения функции *f2*, слово, равно 0; *RES* — результат функции *f3*, слово, равно 0. Все данные инициализированы 0, потому что их значения изменялись для тестирования разных случаев. Значение I одинаково во все случаях.

Сегмент кода.

Выполняется директива *ASSUME*, которая соотносит сегментные регистры и сегменты.

*Процедура F3 (*имеет операнд *NEAR*, значит процедура находится в том же сегменте кода, где и головная процедура) — организуется вычисление функции *f3*. Делается сравнение *к* с 0.

Если *k == 0*, то функция считает модуль разности *I1* и *I2*. Делается условный переход по метке *IfKzero*, где сравниваются *I1* и *I2* через регистр общего назначения *AX*. Если *I1 > I2*, то делается условный переход по метке *absI1moreI2*, где *RES = I1 — I2*, в конце метки процедура возвращает в вызывающую функцию. Если *I2 > I1*, то делается условный переход по метке *absI1lessI2*, где *RES = I2 — I1*, в конце метки процедура возвращает в вызывающую функцию.

Если *k != 0*, то функция считает минимальное значение среди *I1* и *I2*. Делается условный переход по метке *IfKnzero*, где сравниваются I1 и I2 через регистр общего назначения *AX*. Если *I1 < I2*, то делается условный переход по метке *minI1lessI2*, где *RES = I1*, значение в память загружается через

регистр общего назначения *AX*, в конце метки процедура возвращает в вызывающую функцию. Если *I1 >= I2*, то делается условный переход по метке *minI1moreI2*, где *RES = I2*, значение в память загружается через регистр общего назначения *AX*, в конце метки процедура возвращает в вызывающую функцию.

Процедура MAIN (имеет операнд *FAR*, значит процедура является точкой входа в программу) — вычисляет *I1* и *I2*, вызывает процедуру *F3* для вычисления результата.

В начале программы по стандарту на стеке сохраняется начало *PSP* для последующего восстановления по команде *RET*, завершающей процедуру. Также загружается сегментный регистр данных через регистр общего назначения *AX*.

Для вычисления функций *f1* и *f2* происходит сравнение *A* и *B* через регистр общего назначения *AX*.

Если *A>B*, то делается условный переход по метке *IFAmoreB*, где вычисляется значение *I1* и *I2* через регистр общего назначения *AX* с использованием лишь операций сдвига, сложения, вычитания и смены знака у числа. В результате выполнения: *I1 = 15-2*I* и *I2 = -(6*I-4)*. В конце метки делается безусловный переход на метку *Continue*, чтобы продолжить выполнение процедуры *MAIN*.

Если *A<=B*, то делается условный переход по метке *IFAmoreB*, где вычисляется значение *I1* и *I2* через регистр общего назначения *AX* с использованием лишь операций сдвига, сложения, вычитания и смены знака у числа. В результате выполнения: *I1 = I*3+4* и *I2 = 3*(I+2)*. В конце метки управление автоматически переходит на метку *Continue*.

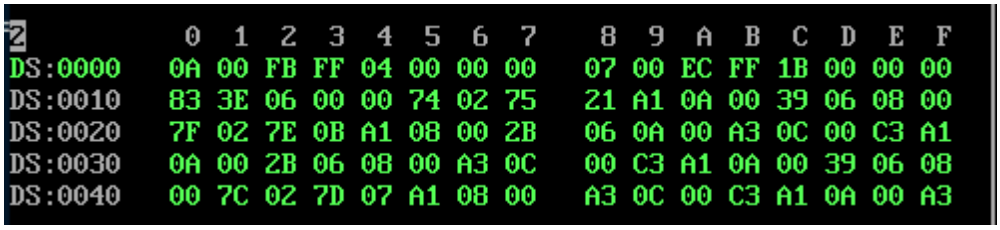
В конце делается вызов процедуры *F3*.

Команда *RET* в конце процедуры возвращает в *DOS*.

Тестирование.

Для тестирования значения *A*, *B* и *K* задавались в сегменте данных при редактировании в программы. Программа, залитая на GitHub имеет нули в области памяти. I = 4 всегда.

- A = 10, B = -5, K = 0



A: 0000-0001: 0A00 → A = 000A = 10

B: 0002-0003: FBFF → B = FFFB = -5

I: 0004-0005: 0400 → I = 0004 = 4

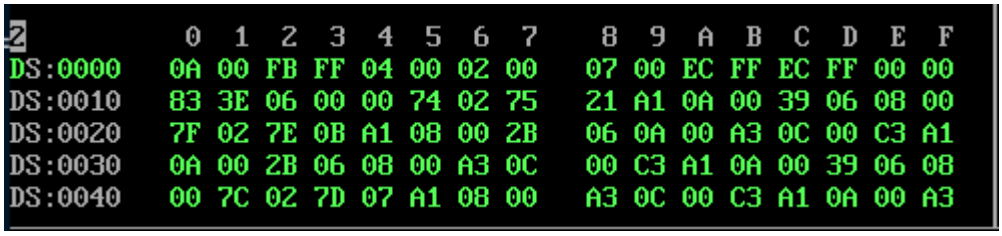
K: 0006-0007: 0000 → K = 0000 = 0

I1: 0008-0009: 0D00 → I1 = 0007 = 7

I2: 000A-000B: ECFF → I2 = FFEC = -20

RES: 000C-000D: 1B00 - > RES = 001B = 27

- A = 10, B = -5, K = 2



A: 0000-0001: 0A00 → A = 000A = 10

B: 0002-0003: FBFF → B = FFFB = -5

I: 0004-0005: 0400 → I = 0004 = 4

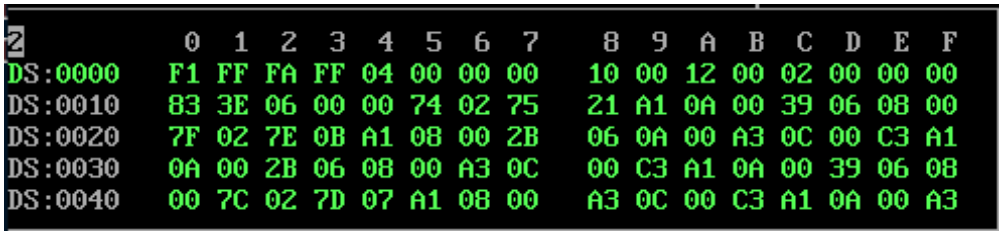
K: 0006-0007: 0200 → K = 0002 = 2

I1: 0008-0009: 0700 → I1 = 0007 = 7

I2: 000A-000B: ECFF → I2 = FFEC = -20

RES: 000C-000D: ECFF → RES = FFEC = -20

- A = -15, B = -6, K = 0



Z	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
DS:0000	F1	FF	FA	FF	04	00	00	00	10	00	12	00	02	00	00	00
DS:0010	83	3E	06	00	00	74	02	75	21	A1	0A	00	39	06	08	00
DS:0020	7F	02	7E	0B	A1	08	00	2B	06	0A	00	A3	0C	00	C3	A1
DS:0030	0A	00	2B	06	08	00	A3	0C	00	C3	A1	0A	00	39	06	08
DS:0040	00	7C	02	7D	07	A1	08	00	A3	0C	00	C3	A1	0A	00	A3

A: 0000-0001: F1FF → A = FFF1 = -15

B: 0002-0003: FAFF → B = FFFA = -6

I: 0004-0005: 0400 → I = 0004 = 4

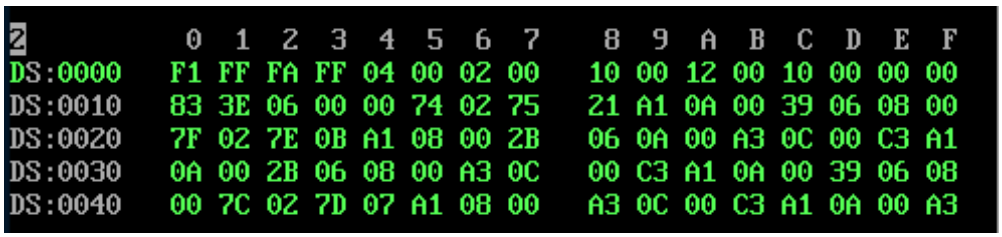
K: 0006-0007: 0000 → K = 0000 = 2

I1: 0008-0009: 1000 → I1 = 0010 = 16

I2: 000A-000B: 1200 → I2 = 0012 = 18

RES: 000C-000D: 0200 → RES = 0002 = 2

- A = -15, B = -6, K = 0



A: 0000-0001: F1FF → A = FFF1 = -15

B: 0002-0003: FAFF → B = FFFA = -6

I: 0004-0005: 0400 → I = 0004 = 4

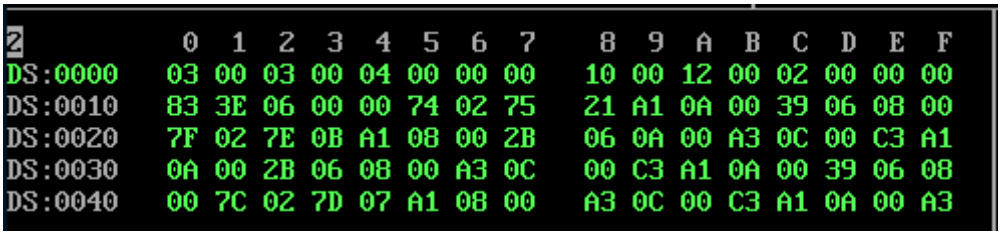
K: 0006-0007: 0000 → K = 0000 = 2

I1: 0008-0009: 1000 → I1 = 0010 = 16

I2: 000A-000B: 1200 → I2 = 0012 = 18

RES: 000C-000D: 1000 → RES = 0010 = 16

- A = 3, B = 3, K = 0



A: 0000-0001: 0300→ A = 0003 = 3

B: 0002-0003: 0300 → B = 0003 = 3

I: 0004-0005: 0400 → I = 0004 = 4

K: 0006-0007: 0000 → K = 0000 = 0

I1: 0008-0009: 1000 → I1 = 0010 = 16

I2: 000A-000B: 1200 → I2 = 0012 = 18

RES: 000C-000D: 0200 → RES = 0002 = 2

Получены верные результаты.

Выводы.

Было изучено представление и обработку целых чисел.

В результате лабораторной работы была разработана программа, которая, используя условные переходы, вычисляет значение функций.

ПРИЛОЖЕНИЕ А

КОД ПРОГРАММЫ

Название файла: Lab3.asm

```
; Стек  программы


AStack    SEGMENT  STACK
          DW 12 DUP('!')    ; Отводится 12 слов памяти
AStack    ENDS

; Данные программы

DATA      SEGMENT

;  Директивы описания данных

A       DW 0
B       DW 0
I       DW 4
K       DW 0
I1      DW 0
I2      DW 0
RES     DW 0

DATA      ENDS

; Код программы

CODE      SEGMENT

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

F3       PROC NEAR
         CMP K,0 ; Compare k and 0
         JZ IfKzero
         JNZ IfKnzero
  IfKzero:
           mov AX,I2
           CMP I1,AX
           JG absI1moreI2
           JLE absI1lessI2
  absI1moreI2:
           mov AX,I1 ; AX = I1
           SUB AX,I2 ; AX = I1 - I2
           mov RES,AX ; RES - res f3, res = I1 - I2, I1>I2
           ret
  absI1lessI2:
           mov AX,I2 ; AX = I2
           SUB AX,I1 ; AX = I2 - I1
           mov RES,AX ; RES - res f3, res = I2 - I1, I2>=I1
```

```
          ret
  IfKnzero:
          mov AX,I2
          CMP I1,AX
          JL minI1lessI2
          JGE minI1moreI2
  minI1lessI2:
          mov AX,I1
          mov RES,AX ; RES - res f3, RES = I1, I1 - min(I1,I2)
          ret
  minI1moreI2:
          mov AX,I2
            mov RES,AX ; RES - res f3, RES = I2, I2 - min(I1,I2) or
I2==I1
          ret

F3      ENDP

; Головная процедура
MAIN    PROC  FAR

        push  DS        ;\  Сохранение адреса начала PSP в стеке
        sub   AX,AX     ; > для последующего восстановления по
        push  AX        ;/  команде ret, завершающей процедуру.
        mov   AX,DATA             ; Загрузка сегментного
        mov   DS,AX               ; регистра данных.

        mov AX,B
        CMP A,AX
        JG IFAmoreB
        JLE IFAlessB

  IFAmoreB:
        mov AX,I ; AX = I
        SAL AX,1 ; AX = I*2
        NEG AX ; AX = -I*2
        ADD AX,15 ; AX = -I*2+15
        mov I1,AX ; I1 - res f1, I1 = 15-2*I

        mov AX,I ; AX = I
        SAL AX,1 ; AX = I*2
        MOV BX,AX ; BX = AX
        ADD AX,BX ; AX = AX*2 = I*4
        ADD AX,BX ; AX = AX*3 = I*6
        SUB AX,4 ; AX = I*6-4
        NEG AX ; AX = -(I*6-4)
        mov I2,AX ; I2 - res f2, I2 = -(6*I-4)

        JMP Continue ; return to code

  IFAlessB:
        mov AX,I ; AX = I
        mov BX, AX ; BX = AX
        ADD AX,BX ; AX = AX*2
        ADD AX,BX ; AX = AX*3 = I*3
```

```
        ADD AX,4 ; AX = I*3+4
        mov I1,AX ; I1 - res f1, I1 = I*3+4

        mov AX,I ; AX = I
        ADD AX,2 ; AX = I+2
        mov BX,AX ; BX = AX
        ADD AX,BX ; AX = 2*(I+2)
        ADD AX,BX ; AX = 3*(I+2)
        mov I2,AX ; I2 - res f2, I2 = 3*(I+2)

  Continue:
        CALL F3


          ret                       ; Выход в DOS по команде,
                                    ; находящейся в 1-ом слове PSP.
Main      ENDP
CODE      ENDS
          END MAIN
```

ПРИЛОЖЕНИЕ Б

ЛИСТИНГ ПРОГРАММЫ

Название файла: Lab3.LST

```
Microsoft (R) Macro Assembler Version 5.10                  10/24/21
02:54:2
                                                            Page     1-1


 0000                   AStack    SEGMENT  STACK
 0000  000C[                            DW 12 DUP('!')
         0021
                   ]

 0018                   AStack    ENDS


 0000                   DATA      SEGMENT


 0000  0000                   A       DW 0
 0002  0000                   B       DW 0
 0004  0000                   I       DW 4
 0006  0000                   K       DW 0
 0008  0000                   I1      DW 0
 000A  0000                   I2      DW 0
 000C  0000                   RES     DW 0

 000E                   DATA      ENDS


 0000                   CODE      SEGMENT

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

```
0000                F3      PROC NEAR
0000  83 3E 0006 R 00                  CMP K,0 ; Compare k and 0
0005  74 02                       JZ IfKzero
0007  75 21                       JNZ IfKnzero
0009                  IfKzero:
0009  A1 000A R                 mov AX,I2
000C  39 06 0008 R                    CMP I1,AX
0010  7F 02                          JG absI1moreI2
0012  7E 0B                          JLE absI1lessI2
0014                  absI1moreI2:
0014  A1 0008 R                 mov AX,I1 ; AX = I1
0017  2B 06 000A R                    SUB AX,I2 ; AX = I1 - I2
001B  A3 000C R                 mov RES,AX ; RES - res f3, res = I1
                    - I2, I1>I2
001E  C3                        ret
001F                  absI1lessI2:
001F  A1 000A R                 mov AX,I2 ; AX = I2
0022  2B 06 0008 R                    SUB AX,I1 ; AX = I2 - I1
0026  A3 000C R                 mov RES,AX ; RES - res f3, res = I2
                    - I1, I2>=I1
```



```
0029  C3                        ret
002A                  IfKnzero:
002A  A1 000A R                 mov AX,I2
002D  39 06 0008 R                    CMP I1,AX
0031  7C 02                          JL minI1lessI2
0033  7D 07                          JGE minI1moreI2
0035                  minI1lessI2:
0035  A1 0008 R                 mov AX,I1
0038  A3 000C R                 mov RES,AX ; RES - res f3, RES = I1,
                   I1 - min(I1,I2)
003B  C3                        ret
```

```
003C                    minI1moreI2:
003C  A1 000A R                 mov AX,I2
003F  A3 000C R                 mov RES,AX ; RES - res f3, RES = I2,
                      I2 - min(I1,I2) or I2==I1
0042  C3                        ret

0043                F3      ENDP


0043                MAIN    PROC  FAR

0043  1E                       push  DS
0044  2B C0                        sub   AX,AX
0046  50                       push  AX
0047  B8 ---- R                mov   AX,DATA
004A  8E D8                        mov   DS,AX

004C  A1 0002 R                mov AX,B
004F  39 06 0000 R                  CMP A,AX
0053  7F 02                        JG IFAmoreB
0055  7E 23                        JLE IFAlessB

0057                    IFAmoreB:
0057  A1 0004 R                mov AX,I ; AX = I
005A  D1 F8                         SAR AX,1 ; AX = I*2
005C  F7 D8                         NEG AX ; AX = -I*2
005E  05 000F                       ADD AX,15 ; AX = -I*2+15
0061  A3 0008 R                mov I1,AX ; I1 - res f1, I1 = 15-2*I

0064  A1 0004 R                mov AX,I ; AX = I
0067  D1 F8                         SAR AX,1 ; AX = I*2
0069  8B D8                         MOV BX,AX ; BX = AX
006B  03 C3                         ADD AX,BX ; AX = AX*2 = I*4
006D  03 C3                         ADD AX,BX ; AX = AX*3 = I*6
006F  2D 0004                       SUB AX,4 ; AX = I*6-4
0072  F7 D8                         NEG AX ; AX = -(I*6-4)
0074  A3 000A R                mov I2,AX ; I2 - res f2, I2 = -(6*I-4)
```

```
0077  EB 1F 90                  JMP Continue ; return to code

007A                    IFAlessB:
007A  A1 0004 R               mov AX,I ; AX = I
007D  8B D8                        mov BX, AX ; BX = AX
007F  03 C3                        ADD AX,BX ; AX = AX*2
0081  03 C3                        ADD AX,BX ; AX = AX*3 = I*3
0083  05 0004                      ADD AX,4 ; AX = I*3+4
0086  A3 0008 R               mov I1,AX ; I1 - res f1, I1 = I*3+4

0089  A1 0004 R               mov AX,I ; AX = I
008C  05 0002                      ADD AX,2 ; AX = I+2
008F  8B D8                        mov BX,AX ; BX = AX
0091  03 C3                        ADD AX,BX ; AX = 2*(I+2)
0093  03 C3                        ADD AX,BX ; AX = 3*(I+2)
0095  A3 000A R               mov I2,AX ; I2 - res f2, I2 = 3*(I+2)

0098                    Continue:
0098  E8 0000 R               CALL F3


009B  CB                        ret

009C                  Main      ENDP
009C                  CODE      ENDS
                                END MAIN
```

```
Segments and Groups:

                N a m e         Length      Align     Combine Class

ASTACK . . . . . .              0018  PARA  STACK
CODE . . . . . .                009C  PARA  NONE
DATA . . . . . .                000E  PARA  NONE

Symbols:

                N a m e         Type   Value      Attr

A  . . . . . .                  L WORD     0000  DATA
ABSI1LESSI2  . . . . . .        L NEAR     001F  CODE
ABSI1MOREI2  . . . . . .        L NEAR     0014  CODE

B  . . . . . .                  L WORD     0002  DATA

CONTINUE . . . . . .            L NEAR     0098  CODE

F3 . . . . . .                  N PROC     0000  CODE  Length    =
0043

I  . . . . . .                  L WORD     0004  DATA
I1 . . . . . .                  L WORD     0008  DATA
I2 . . . . . .                  L WORD     000A  DATA
IFALESSB . . . . . .            L NEAR     007A  CODE
IFAMOREB . . . . . .            L NEAR     0057  CODE
IFKNZERO . . . . . .            L NEAR     002A  CODE
IFKZERO  . . . . . .            L NEAR     0009  CODE

K  . . . . . .                  L WORD     0006  DATA

MAIN . . . . . .                F PROC     0043  CODE  Length    =
0059
```

```
MINI1LESSI2  . . . . . .      L NEAR     0035  CODE
MINI1MOREI2  . . . . . .      L NEAR     003C  CODE

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

@CPU . . . . . .              TEXT  0101h
@FILENAME  . . . . . .        TEXT  LAB3
@VERSION . . . . . .          TEXT  510


    120 Source  Lines
    120 Total   Lines
     26 Symbols

  48056 + 457154 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors
```