

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:

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 .

Функции варианта 3:

$$\begin{aligned} f1 &= \begin{cases} 15-2*i, & \text{при } a>b \\ 3*i+4, & \text{при } a\leq b \end{cases} \\ f2 &= \begin{cases} -(6*i - 4), & \text{при } a>b \\ 3*(i+2), & \text{при } a\leq b \end{cases} \end{aligned}$$

$$f_3 = \begin{cases} / |i_1 + i_2|, & \text{при } k=0 \\ \setminus \min(i_1, i_2), & \text{при } k \neq 0 \end{cases}$$

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 f_1 и f_2 вместо операции умножения следует использовать арифметический сдвиг и, возможно, сложение;
- 3) при вычислении функций f_1 и f_2 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.

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 $k \neq 0$, 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 $I1 \geq I2$, 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 $A \leq B$, 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*.

Далее организуется вычисление функции *f3*. Делается сравнение *k* с 0.

Если $k == 0$, то функция считает модуль разности *I1* и *I2*. Делается условный переход по метке *IfKzero*, где выполняется вычитание *I1-I2* в регистре *AX*, если число получилось отрицательное, то выполняется переход

по метке *ABS*, где сменяется знак у *AX* и результат записывается в *RES*.
Выполняется безусловный переход по метке *Finish* — переход к команде *RET*.

Если $k \neq 0$, то переход по метке *IfKzero* не осуществляется, а сравниваются *I1* и *I2* через *AX*. Если $I1 < I2$, то осуществляется переход по метке *minI1lessI2*, в результат записывается *I1*, после осуществляется автоматический переход к метке *Finish*. Если $I1 \geq I2$, то переход по метке *minI1lessI2* не осуществляется, в результат записывается *I2*, после осуществляется безусловный переход к метке *Finish*.

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.

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

2	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
DS:0000	0A	00	FB	FF	04	00	00	00	07	00	EC	FF	1B	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: 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: 0700 → $I1 = 0007 = 7$

I2: 000A-000B: ECFF → $I2 = FFEC = -20$

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

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

2	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
DS:0000	0A	00	FB	FF	04	00	02	00	07	00	EC	FF	EC	FF	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: 0A00 \rightarrow A = 000A = 10

B: 0002-0003: FBFF \rightarrow B = FFFB = -5

I: 0004-0005: 0400 \rightarrow I = 0004 = 4

K: 0006-0007: 0200 \rightarrow K = 0002 = 2

I1: 0008-0009: 0700 \rightarrow I1 = 0007 = 7

I2: 000A-000B: ECFF \rightarrow I2 = FFEC = -20

RES: 000C-000D: ECFF \rightarrow RES = FFEC = -20

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

2	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 \rightarrow A = FFF1 = -15

B: 0002-0003: FAFF \rightarrow B = FFFA = -6

I: 0004-0005: 0400 \rightarrow I = 0004 = 4

K: 0006-0007: 0000 \rightarrow K = 0000 = 0

I1: 0008-0009: 1000 \rightarrow I1 = 0010 = 16

I2: 000A-000B: 1200 \rightarrow I2 = 0012 = 18

RES: 000C-000D: 0200 \rightarrow RES = 0002 = 2

- A = -15, B = -6, K = 2

2	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
DS:0000	F1	FF	FA	FF	04	00	02	00	10	00	12	00	10	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 \rightarrow A = FFF1 = -15

B: 0002-0003: FAFF \rightarrow B = FFFA = -6

I: 0004-0005: 0400 \rightarrow I = 0004 = 4

K: 0006-0007: 0200 \rightarrow K = 0002 = 2

I1: 0008-0009: 1000 \rightarrow I1 = 0010 = 16

I2: 000A-000B: 1200 \rightarrow I2 = 0012 = 18

RES: 000C-000D: 1000 \rightarrow RES = 0010 = 16

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

2	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
DS:0000	03	00	03	00	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: 0300 \rightarrow A = 0003 = 3

B: 0002-0003: 0300 \rightarrow B = 0003 = 3

I: 0004-0005: 0400 \rightarrow I = 0004 = 4

K: 0006-0007: 0000 \rightarrow K = 0000 = 0

I1: 0008-0009: 1000 \rightarrow I1 = 0010 = 16

I2: 000A-000B: 1200 \rightarrow I2 = 0012 = 18

RES: 000C-000D: 0200 \rightarrow 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

```
AStack    SEGMENT    STACK
           DW 12 DUP('!')
AStack    ENDS

DATA       SEGMENT

A          DW -15
B          DW -6
I          DW 4
K          DW 1
I1         DW 0
I2         DW 0
RES        DW 0

DATA       ENDS

CODE       SEGMENT

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

MAIN       PROC    FAR

           push    DS
           sub     AX,AX
           push    AX
           mov     AX,DATA
           mov     DS,AX

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

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

           SAL     AX,1 ; AX = 30-4*I
           SUB     AX,I ; AX = 30-5*I
           SUB     AX,I ; AX = 30-6*I
```

```

SUB AX,26 ; AX = 4-6*I
mov I2,AX ; I2 = res f2, I2 = -(6*I-4)

JMP Continue ; return to code

IFAlessB:
    mov AX,I ; AX = I
    SAL AX,1 ; AX = 2*I
    ADD AX,I ; AX = 3*I
    ADD AX,4 ; AX = I*3+4
    mov I1,AX ; I1 = res f1, I1 = I*3+4

    ADD AX,2 ; AX = I*3+6
    mov I2,AX ; I2 = res f2, I2 = 3*(I+2)

Continue:

    CMP K,0 ; Compare k and 0
    JZ IfKzero
    mov AX,I2
    CMP I1,AX
    JL minI1lessI2
    mov RES,AX ; RES = res f3, RES = I2, I2 = min(I1,I2) or
I2==I1
    JMP Finish
IfKzero:
    mov AX,I1 ; AX = I1
    SUB AX,I2 ; AX = I1-I2
    CMP AX,0
    JL Abs
    mov RES,AX
    JMP Finish
Abs:
    NEG AX
    mov RES,AX
    JMP Finish
minI1lessI2:
    mov AX,I1
    mov RES,AX ; RES = res f3, RES = I1, I1 = min(I1,I2)
Finish:
    ret

Main      ENDP
CODE      ENDS
          END MAIN

```

ПРИЛОЖЕНИЕ Б

ЛИСТИНГ ПРОГРАММЫ

Название файла: Lab3.LST

#Microsoft (R) Macro Assembler Version 5.10
21:53:0

10/24/21

Page 1-1

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

```
0018          AStack  ENDS
```

```
0000          DATA   SEGMENT
```

```
0000  FFF1          A      DW -15
0002  FFFA          B      DW -6
0004  0004          I      DW 4
0006  0001          K      DW 1
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                                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

0009  A1 0002 R                        mov  AX,B
000C  39 06 0000 R                    CMP  A,AX
0010  7F 02                            JG   IFAMoreB
0012  7E 20                            JLE  IFAllessB

0014                                IFAMoreB:
0014  A1 0004 R                        mov  AX,I ; AX = I
0017  D1 E0                            SAL  AX,1 ; AX = I*2
0019  F7 D8                            NEG  AX ; AX = -I*2
001B  05 000F                        ADD  AX,15 ; AX = -I*2+15
001E  A3 0008 R                        mov  I1,AX ; I1 - res f1, I1 = 15-2*I

0021  D1 E0                            SAL  AX,1 ; AX = 30-4*I
0023  2B 06 0004 R                    SUB  AX,I ; AX = 30-5*I
0027  2B 06 0004 R                    SUB  AX,I ; AX = 30-6*I
002B  2D 001A                        SUB  AX,26 ; AX = 4-6*I
002E  A3 000A R                        mov  I2,AX ; I2 - res f2, I2 = -(6*I-4
)

```

21:53:0

Page 1-2

```
0031 EB 16 90                JMP Continue ; return to code

0034                        IFAllessB:
0034 A1 0004 R                mov AX,I ; AX = I
0037 D1 E0                  SAL AX,1 ; AX = 2*I
0039 03 06 0004 R          ADD AX,I ; AX = 3*I
003D 05 0004              ADD AX,4 ; AX = I*3+4
0040 A3 0008 R            mov I1,AX ; I1 = res f1, I1 = I*3+4

0043 05 0002              ADD AX,2 ; AX = I*3+6
0046 A3 000A R            mov I2,AX ; I2 = res f2, I2 = 3*(I+2)

0049                        Continue:

0049 83 3E 0006 R 00        CMP K,0 ; Compare k and 0
004E 74 0F                JZ IfKzero
0050 A1 000A R            mov AX,I2
0053 39 06 0008 R        CMP I1,AX
0057 7C 20                JL minI1lessI2
0059 A3 000C R            mov RES,AX ; RES = res f3, RES = I2,
                        I2 = min(I1,I2) or I2==I1
005C EB 21 90            JMP Finish
005F                        IfKzero:
005F A1 0008 R            mov AX,I1 ; AX = I1
0062 2B 06 000A R        SUB AX,I2 ; AX = I1-I2
0066 3D 0000              CMP AX,0
0069 7C 06                JL Abs
006B A3 000C R            mov RES,AX
006E EB 0F 90            JMP Finish
0071                        Abs:
0071 F7 D8                NEG AX
0073 A3 000C R            mov RES,AX
```

```

0076 EB 07 90                                JMP Finish
0079                                minI1lessI2:
0079 A1 0008 R                                mov AX,I1
007C A3 000C R                                mov RES,AX ; RES - res f3, RES = I1,
I1 - min(I1,I2)
007F                                Finish:
007F CB                                ret

0080                                Main      ENDP
0080                                CODE      ENDS
                                END MAIN

```

Segments and Groups:

N a m e	Length	Align	Combine Class
ASTACK	0018	PARA	STACK
CODE	0080	PARA	NONE
DATA	000E	PARA	NONE

Symbols:

N a m e	Type	Value	Attr	
A	L WORD	0000	DATA	
ABS	L NEAR	0071	CODE	
B	L WORD	0002	DATA	
CONTINUE	L NEAR	0049	CODE	
FINISH	L NEAR	007F	CODE	
I	L WORD	0004	DATA	
I1	L WORD	0008	DATA	
I2	L WORD	000A	DATA	
IFALESSB	L NEAR	0034	CODE	
IFAMOREB	L NEAR	0014	CODE	
IFKZERO	L NEAR	005F	CODE	
K	L WORD	0006	DATA	
MAIN	F PROC	0000	CODE	Length =
0080				

MINI1LESSI2	L NEAR	0079	CODE
RES	L WORD	000C	DATA
@CPU	TEXT	0101h	
@FILENAME	TEXT	lab3	
@VERSION	TEXT	510	

91 Source Lines

91 Total Lines

23 Symbols

47998 + 459262 Bytes symbol space free

0 Warning Errors

0 Severe Errors