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

# ОТЧЕТ

**по лабораторной работе №7**
**по дисциплине «Организация ЭВМ и систем»**
**Тема: Использование арифметических операций над целыми числами и процедур в Ассемблере.**

Студент гр. 9382 _____ Кодуков А.В.

Преподаватель _____ Ефремов М.А .

Санкт-Петербург
2020

**Задание:**

Разработать на языке Ассемблер процессора IntelX86 две процедуры:

• одна – выполняет прямое преобразование целого числа, заданного в регистре AX ( или в паре регистров DX:AX) в строку, представляющую его символьное изображение в заданной системе счисления (с учетом или без учета знака в зависимости от варианта задания);

• другая - обратное преобразование строки, представляющей символьное изображение числа в заданной системе счисления в целое число, помещаемое в регистр AX ( или в пару регистров DX:AX)

Строка должна храниться в памяти, а также выводиться на экран для индикации.

Отрицательные числа при представлении с учетом знака должны в памяти храниться в дополнительном коде, а на экране изображаться в прямом коде с явным указанием знака или в символьном виде со знаком.

Вариант 2.1.1 (32 бита, с учетом знака, двоичная)

**Выполнение работы:**

Перевод из регистров в строку в двоичном виде происходит с помощью выделения отдельных битов с помощью маски и вычленения знака числа из первого бита, для получения модуля из отрицательного числа, значение инвертируется и добавляется 1.

Обратная процедура так же работает с использованием маски, но уже для выставления необходимых битов, записанных в строке. 0 и максимально отрицательное число данного размера обрабатываются отдельно

Тесты:

DX = 0, AX = 0

```
Transformation from registers: 0
Transformation from string to registers and back: 0
```

DX = 8000h, AX = 0

```
Transformation from registers: -10000000000000000000000000000000
Transformation from string to registers and back: -10000000000000000000000000000000
000
```

DX = 0, AX = 1h

```
Transformation from registers: +1
Transformation from string to registers and back: +1
```

DX = 0FFFFh, AX = 0FFFFh

```
Transformation from registers: -1
Transformation from string to registers and back: -1
```

**Вывод:**

В ходе выполнения работы были использованы арифметические операции над целыми числами и процедуры в ассемблере, разработаны 2 процедуры по преобразованию числа в строку в довичном виде и обратно.

# ПРИЛОЖЕНИЕ А
# ИСХОДНЫЙ КОД ПРОГРАММЫ

func.asm

```
STACKSG SEGMENT  PARA STACK 'Stack'
  DW 1024 DUP(?)
STACKSG     ENDS

DATASG  SEGMENT  PARA 'Data'; SEG DATA
  KEEP_CS DW 0; для хранения сегмента
  KEEP_IP DW 0; и смещения вектора прерывания
  GREETING  DB 'Kodukov Aleksandr 9382 $'
  crlf db 0ah, 0dh, '$'
DATASG      ENDS; ENDS DATA

CODE SEGMENT; SEG CODE
ASSUME DS:DataSG, CS:Code, SS:STACKSG

INTER_TIMER PROC FAR

  PUSH AX; сохранение изменяемых регистров
  PUSH DX

  ; действия по обработке прерывания
  MOV AH, 9; вызов того,
  INT 21H; что хранится в dx

  MOV DX, OFFSET crlf
  MOV AH, 9
  INT 21H

  POP DX; восстановление регистров
  POP AX

  MOV AL, 20H
  OUT 20H, AL

IRET

INTER_TIMER  ENDP

Main PROC FAR
  MOV AX, DATASG; ds setup
  MOV DS, AX

  MOV AH, 35H; функция получения вектора
  MOV AL, 08H; номер вектора
  INT 21H
  MOV KEEP_IP, BX; запоминание смещения
  MOV KEEP_CS, ES; и сегмента вектора прерывания

  CLI
  PUSH DS
  MOV DX, OFFSET INTER_TIMER
  MOV AX, SEG INTER_TIMER; сегмент процедуры
  MOV DS, AX; помещаем в DS
  MOV AH, 25H; функция установки вектора
  MOV AL, 08H; номер вектора
  INT 21H; меняем прерывание
  POP DS
  STI
```

```
  MOV DX, OFFSET GREETING; помещаем строку в DS
  INT 08h

  CLI
  PUSH DS
  MOV DX, KEEP_IP
  MOV AX, KEEP_CS
  MOV DS, AX
  MOV AH, 25H
  MOV AL, 08H
  INT 21H; восстанавливаем старый вектор прерывания
  POP DS
  STI

  MOV AH, 4CH
  INT 21H

Main ENDP
CODE ENDS
END Main
```