­Министерство высшего и профессионального образования РФ

Государственное образовательное учреждение

высшего образования

Ульяновский государственный технический университет

Кафедра «Вычислительная техника»

Дисциплина «МОП»

**Лабораторная работа №1**

«Основы технологии программирования на ассемблере»

Вариант 5

Выполнил:

студент 3 курса, гр. ИВТВМбд-31

Захарычев Н.А.

Проверил:

преподаватель кафедры ВТ

Лылова А.В.

**г.Ульяновск 2017**

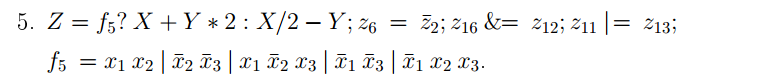
# Цель работы:

Приобретение умений и навыков выполнения основных технологических операций, используемых при программировании на ассемблере.

Общие соглашения:

1. Во всех вариантах заданий требуется добавить в программу функции ввода двух двадцатиразрядных двоичных чисел, вычисления заданного выражения и вывода результата.
2. Все умножения и деления на константы, кратные степени двойки, необходимо выполнять командами сдвига.

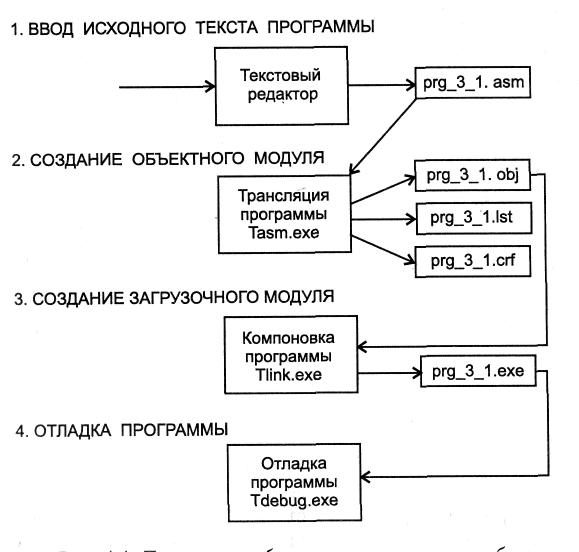
# Задание по варианту:



# Выполнение лабораторной работы

## Описание основных операций циклического процесса «редактирование – ассемблирование – компоновка - выполнение» и основных опций запуска программ ассемблирования и компоновки.

Процесс создания программы на языке Assembler:



## Редактирование

На первом этапе, когда вводится код программы, можно использовать любой текстовый редактор. При выборе редактора нужно учитывать, что он не должен вставлять «посторонних» символов (специальных символов форматирования)

## Трансляция программы

Следующий шаг на пути создания исполняемого модуля — трансляция программы.

На этапе трансляции решается несколько задач:

* перевод команд ассемблера в соответствующие машинные команды;
* построение таблицы символов;
* расширение макросов;
* формирование файла листинга и объектного модуля.

Объектный модуль включает в себя представление исходной программы в машинных кодах и некоторую другую информацию, необходимую для отладки и компоновки его с другими модулями.

Программа, которая реализует эти задачи, называется ассемблером. Итог работы ассемблера — два файла: файл объектного модуля и файл листинга.

*TASM [опция] имя\_исходного\_файла [.имя\_объектного\_файла] [.имя\_файла\_листинга] [.имя\_файла\_перекрестных ссылок]*

Опции транслятора, использованные при выполнении текущей лабораторной работы:

**/l** — указывает на необходимость создания файла листинга, даже если он не “заказывается” в командной строке;

## Компоновка программы

Главная цель этого шага – преобразовать код и данные в объектных файлах в их перемещаемое выполняемое отображение. Результатом работы компоновщика является создание загрузочного файла с расширением .EXE. После этого операционная система может загрузить такой файл в память и выполнить его.

Полный формат командной строки для запуска компоновщика достаточно сложен. Поэтому рассмотрим не полный формат командной строки для запуска компоновщика:

*TLINK [опции] список\_объектных\_файлов [.имя\_загрузочного\_модуля] [.имя\_файла\_карты] [.имя\_файла\_библиотеки] [.имя\_файла\_определений] [.имя\_ресурсного\_файла]*

Опции компоновщика, использованные при выполнении текущей лабораторной работы:

**/v**- включить отладочную информацию в выполняемый файл

**/m**- создать файл карты

## Отладка программы

Отладчик позволяет решить две главные задачи:

* определить место логической ошибки
* определить причину логической ошибки

Перечислим некоторые возможности TD:

* выполнение трассировки программы в прямом направлении, то есть последовательное исполнение программы, при котором за один шаг выполняется одна машинная инструкция
* выполнение трассировки программы в обратном направлении, то есть выполнение программы по одной команде, но в обратном направлении
* просмотр и изменение состояния аппаратных ресурсов процессора во время трассировки.

Для выполнения лабораторной работы был выбран программный пакет ассемблера Turbo Assembler TASM фирмы Borland.

Для удобства выполнения процесса создания программы удобно использовать пакетные файлы (\*.bat).

Пакетный файл ([англ.](http://ru.wikipedia.org/wiki/Английский_язык)*batch file*)— [текстовый файл](http://ru.wikipedia.org/wiki/Текстовый_файл) в [MS-DOS](http://ru.wikipedia.org/wiki/MS-DOS), [OS/2](http://ru.wikipedia.org/wiki/OS/2) или [Windows](http://ru.wikipedia.org/wiki/Microsoft_Windows), содержащий последовательность команд, предназначенных для исполнения командным интерпретатором. После запуска пакетного файла, программа — интерпретатор (как правило [COMMAND.COM](http://ru.wikipedia.org/wiki/COMMAND.COM) или [CMD.EXE](http://ru.wikipedia.org/wiki/CMD.EXE)) читает его строка за строкой и последовательно исполняет команды.

В нашем случае пакетный файл требовался для процесса «редактирование – ассемблировани – компоновка - выполнение».

**Код bat файла (с запуском отладки):**

@echo off

mount c d:/Study/TASM/

:start

tasm mopl1 /l

if errorlevel 1 goto mopl1e

tasm mopl1l

if errorlevel 1 goto mopl1le

tlink mopl1+mopl1l

if errorlevel 1 goto tle

mopl1

goto exit

:mopl1e

td mopl1.asm

pause

goto start

:mopl1le

td mopl1l.asm

pause

goto start

:tle

echo ошибка в tlink

goto exit

:exit

pause

exit

## Анализ исходного кода лабораторной работы

Назначения файлов:

* *mopl1.asm –* основная часть программы
* *mopl1l.asm –* набор библиотека часто используемых процедур
* *mopl1.inc –* набор констант для работы программы
* *mopl1.mac –* набор макросов связывающих библиотеку с основной частью программы

Назначение процедур файла mopl1l*.asm*

PUTSS – вывод на экран строки с задержкой по времени между символами

PUTC – вывод символа на терминал

PUTSSTR - процедура вывода строки по адресу si

GETCH - Подпрограмма ввода символа в AL с терминала

GETS - Подпрограмма ввода строки в буфер, адресуемый DXGETS

DIUBINARY - Подпрограмма преобразования в беззнаковое десятичное

SLEN – определение длины строки

UBINARY – преобразование числа в беззнаковое, десятичное

DILAY – выполнение задержки выполнения программы.

Назначение макросов файла mopl1*.mac*

PUTL – вывод строки на экран

PUTLS – вывод строки с задержкой

PUTLSC – вывод строки с задержкой и выравниванием по центру

FIXTIME – макрос фиксации начала измерения времени

DURAT – макрос окончания измерения времени и получения числа тактов

EXIT – завершение работы программы

## Анализ листинга

Файл листинга содержит код ассемблера исходной программы, а также расширенную информацию об этом коде. Для каждой команды ассемблера указываются ее машинный (объектный) код и смещение в кодовом сегменте. В конце листинга TASM формирует таблицы с информацией о метках и сегментах, используемых в программе.

Строки в файле листинга имеют следующий формат:

*глубина\_вложенности номер\_строки* *смещение машинный\_код исходный\_код*

Особенности структуры файла листинга:

* При обработке транслятором очередной команды ассемблера счетчик адреса увеличивается на длину сформированной машинной команды.
* В листинг включается содержимое файлов, подключенных через INCLUDE, а именно текст макросов и константы.
* Вызовы макросов также заменяются на их код.
* Имена меток внутри макросов заменяются на их порядковые номера, во избежание повтора имен.
* Содержится дополнительная информация.

## Анализ карты загрузки

Карта загрузки mopl1*.map* содержит следующую информацию:

* Адреса сегментов данных, кода и стека
* Точку входа в программу
* Адреса процедур, отсортированные по имени и по значению

## Сравнение объемов объектных файлов.

Размеры файлов без отладочной информации:

*mopl1.obj -* 2620 byte

*mopl1l.obj -* 559 byte

*mopl1.exe -* 2602byte

Размеры файлов с отладочной информацией:

*mopl1.obj -*  4013 байт

*mopl1l.obj -*  2047 байт

*mopl1.exe -*  6628 байт

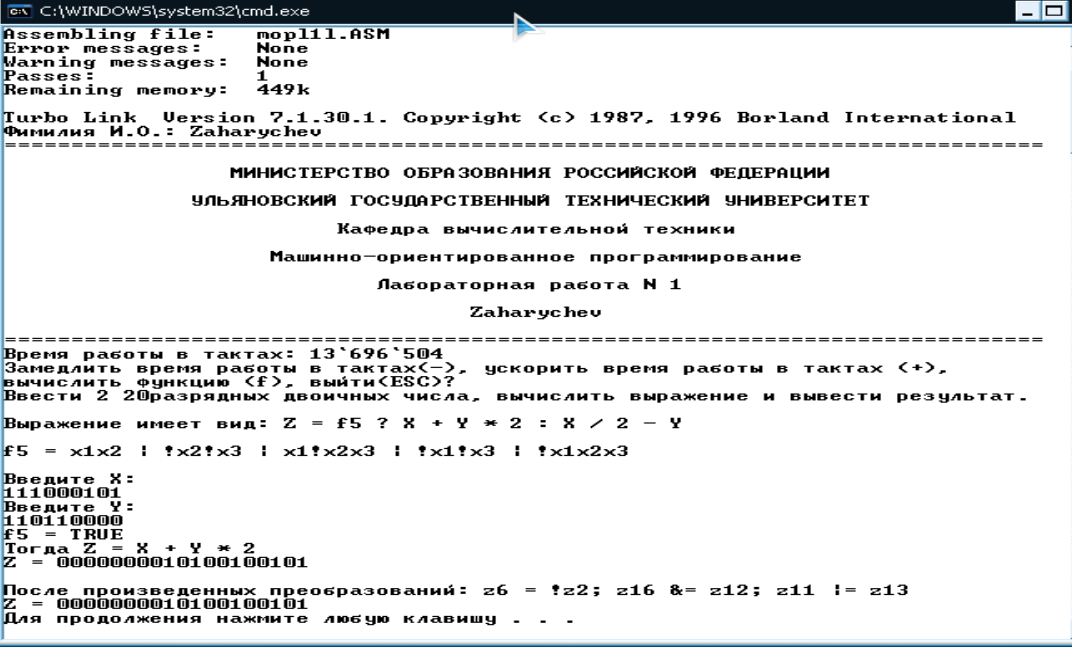
Как видно, размер файлов с отладочной информацией значительно больше, чем размер соответствующих файлов без нее.

# Контрольные примеры:

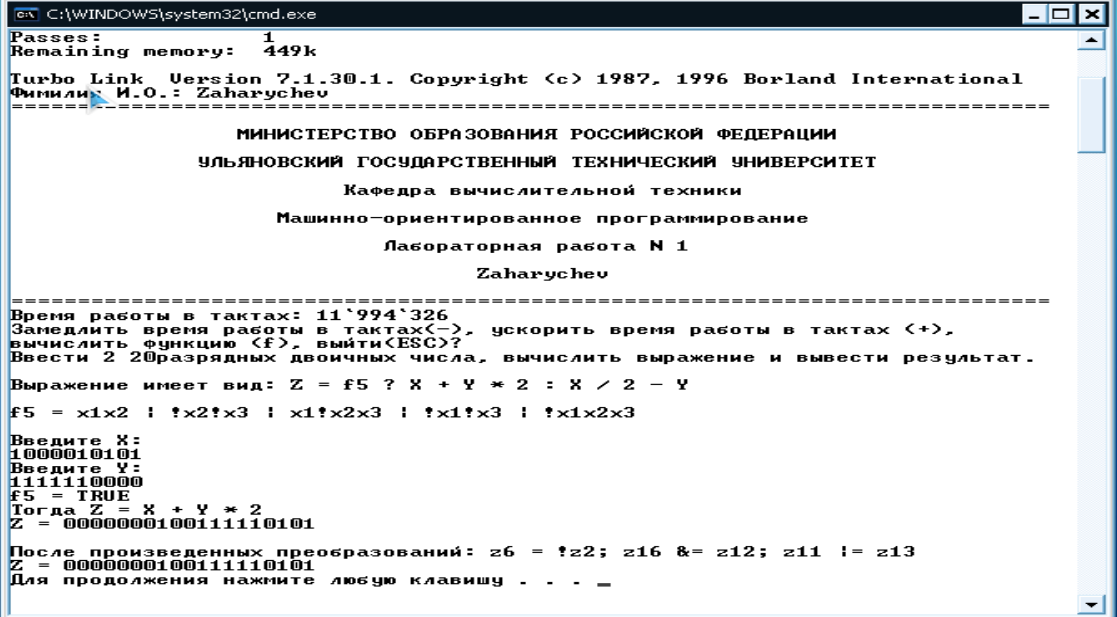
На вход программы подаются два бинарных числа X,Y

На выходе в первой строке результат функции f

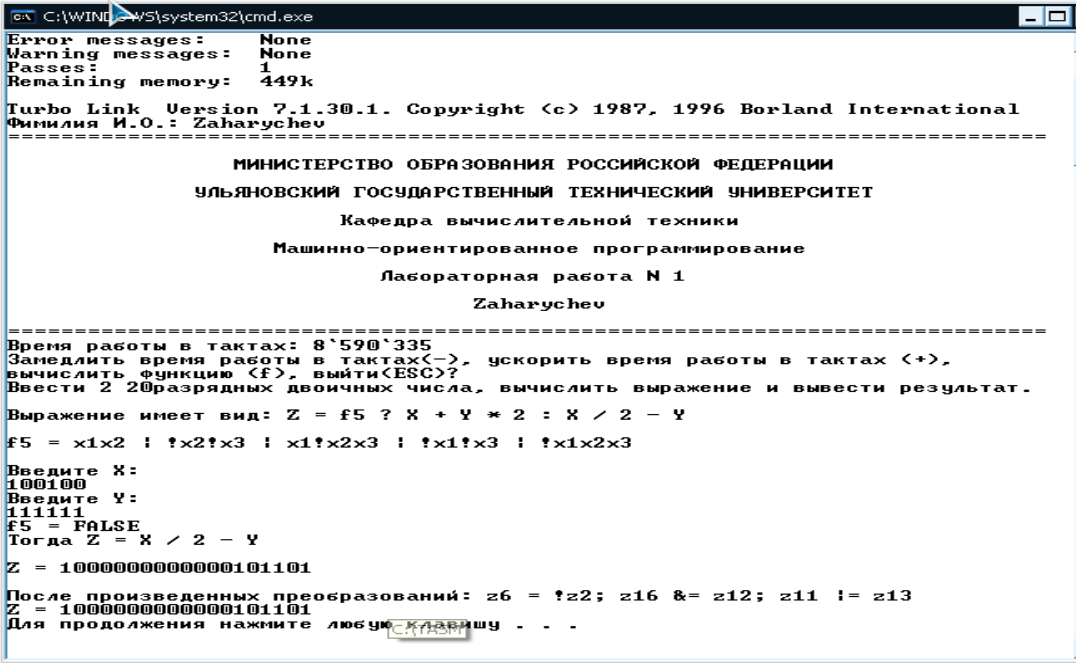
**Пример 1**



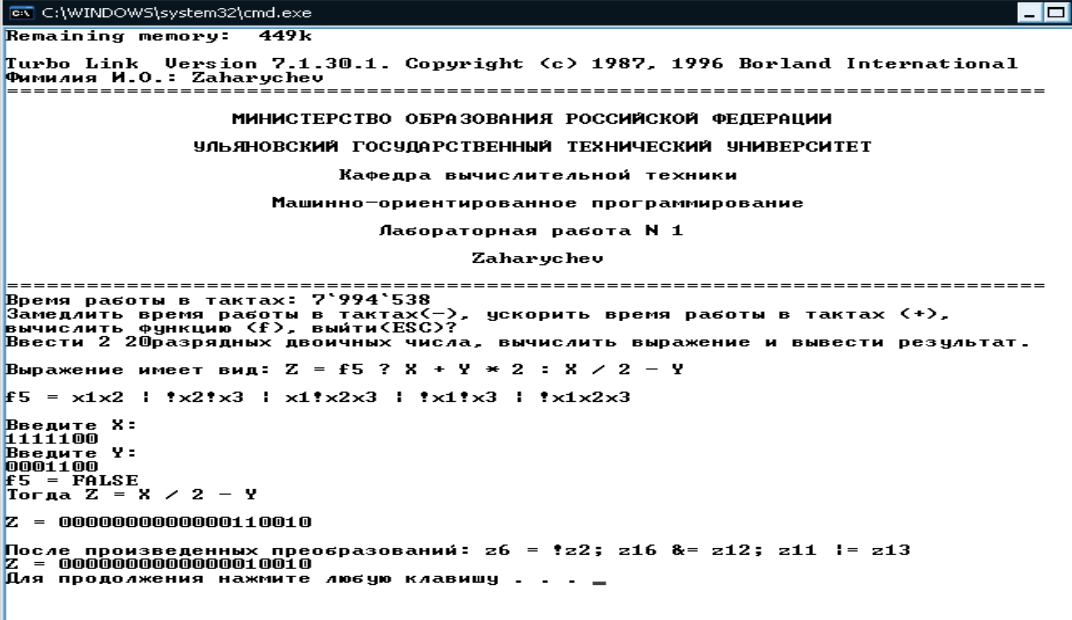
**Пример 2**



**Пример 3**



**Пример 4**



# Исходный код:

**mopl1.asm**

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; MOPL1.ASM - учебный пример для выполнения

; лабораторной работы N1 по машинно-ориентированному программированию

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

.MODEL SMALL

.STACK 200h

.386

; Используются декларации констант и макросов

INCLUDE MOPL1.INC

INCLUDE MOPL1.MAC

; Декларации данных

.DATA

SLINE DB 78 DUP (CHSEP), 0

REQ DB "Фимилия И.О.: ",0FFh

MINIS DB "МИНИСТЕРСТВО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ",0

ULSTU DB "УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ",0

DEPT DB "Кафедра вычислительной техники",0

MOP DB "Машинно-ориентированное программирование",0

LABR DB "Лабораторная работа N 1",0

REQ1 DB "Замедлить время работы в тактах(-), ускорить время работы в тактах (+),", 0

;------------- Новые переменные ------------------------------------------------------------------

REQ2 DB "вычислить функцию (f), выйти(ESC)?", 0FFh

;-------------------------------------------------------------------------------------------------

STR\_TASK DB "Ввести 2 20разрядных двоичных числа, вычислить выражение и вывести результат.", 0

FUNCTION\_TEXT DB "Выражение имеет вид: "

FUNCTION\_EQUAL DB "Z = f5 ? "

FUNC\_PART1 DB "X + Y \* 2", 0FFh

FUNC\_PART\_SEP DB " : ", 0FFh

FUNC\_PART2 DB "X / 2 - Y", 0

BOOL\_FUNC DB "f5 = x1x2 | !x2!x3 | x1!x2x3 | !x1!x3 | !x1x2x3", 0

EQUAL\_F5 DB "f5 = ", 0FFh

NEXT\_EQUAL DB "Тогда "

Z\_EQUAL DB "Z = ", 0FFh

TACTS DB "Время работы в тактах: ",0FFh

WRITE\_X DB "Введите X:", 0FFh

WRITE\_Y DB "Введите Y:", 0FFh

STR\_TRUE DB "TRUE", 0

STR\_FALSE DB "FALSE", 0

AFTER\_CHANGE DB "После произведенных преобразований: z6 = !z2; z16 &= z12; z11 |= z13", 0

EMPTYS DB 0

BUFLEN = 70

BUF DB BUFLEN

LENS DB ?

SNAME DB BUFLEN DUP (0)

BUF\_LEN\_NUMBER = 21

BUF\_NUMBER DB BUF\_LEN\_NUMBER

LEN\_NUMBER DB ?

W\_NUMBER DB BUF\_LEN\_NUMBER DUP (0)

X DD 0

Y DD 0

Z DD 0

PAUSE DW 0, 0 ; младшее и старшее слова задержки при выводе строки

TI DB LENNUM+LENNUM/2 DUP(?), 0 ; строка вывода числа тактов

; запас для разделительных "`"

;========================= Программа =========================

.CODE

; Макрос заполнения строки LINE от позиции POS содержимым CNT объектов,

; адресуемых адресом ADR при ширине поля вывода WFLD

BEGIN LABEL NEAR

; инициализация сегментного регистра

MOV AX, @DATA

MOV DS, AX

; инициализация задержки

MOV PAUSE, PAUSE\_L

MOV PAUSE+2,PAUSE\_H

PUTLS REQ ; запрос имени

; ввод имени

LEA DX, BUF

CALL GETS

@@L: ; циклический процесс повторения вывода заставки

; вывод заставки

; ИЗМЕРЕНИЕ ВРЕМЕНИ НАЧАТЬ ЗДЕСЬ

FIXTIME

PUTL EMPTYS

PUTL SLINE ; разделительная черта

PUTL EMPTYS

PUTLSC MINIS ; первая

PUTL EMPTYS

PUTLSC ULSTU ; и

PUTL EMPTYS

PUTLSC DEPT ; последующие

PUTL EMPTYS

PUTLSC MOP ; строки

PUTL EMPTYS

PUTLSC LABR ; заставки

PUTL EMPTYS

; приветствие

PUTLSC SNAME ; ФИО студента

PUTL EMPTYS

; разделительная черта

PUTL SLINE

; ИЗМЕРЕНИЕ ВРЕМЕНИ ЗАКОНЧИТЬ ЗДЕСЬ

DURAT ; подсчет затраченного времени

; Преобразование числа тиков в строку и вывод

LEA DI, TI

CALL UTOA10

PUTL TACTS

PUTL TI ; вывод числа тактов

; обработка команды

PUTL REQ1

;------Вывод своих строк с действиями -------------------

PUTL REQ2

;--------------------------------------------------------

CALL GETCH

CMP AL, 'f'

JNE PROC\_TIME

INP\_NUM:

PUTL EMPTYS

PUTL STR\_TASK

PUTL EMPTYS

PUTL FUNCTION\_TEXT

PUTL FUNC\_PART\_SEP

PUTL FUNC\_PART2

PUTL EMPTYS

PUTL BOOL\_FUNC

PUTL EMPTYS

PUTL WRITE\_X

PUTL EMPTYS

CALL INP\_NUM\_EBX

MOV X, ebx

PUTL EMPTYS

PUTL WRITE\_Y

PUTL EMPTYS

CALL INP\_NUM\_EBX

MOV Y, ebx

FUN\_BOOL\_CALC:

MOV eax, X

BT eax, 1

JNC B2

BT eax, 2

JC TRUE

B2:

BT eax, 2

JC B3

BT eax, 3

JNC TRUE

B3:

BT eax, 1

JNC B4

BT eax, 2

JC B4

BT eax, 3

JC TRUE

B4:

BT eax, 1

JC B5

BT eax, 3

JNC TRUE

B5:

BT eax, 1

JC FALSE

BT eax, 2

JNC FALSE

BT eax, 3

JC TRUE

FALSE:

PUTL EMPTYS

PUTL EQUAL\_F5

PUTL STR\_FALSE

PUTL NEXT\_EQUAL

PUTL FUNC\_PART2

JMP CALC\_Z\_0

TRUE:

PUTL EMPTYS

PUTL EQUAL\_F5

PUTL STR\_TRUE

PUTL NEXT\_EQUAL

PUTL FUNC\_PART1

JMP CALC\_Z\_1

CALC\_Z\_1:

MOV ebx, X

MOV eax, Y

SHL eax, 1

ADD ebx, eax

MOV Z, ebx

PUTL EMPTYS

PUTL Z\_EQUAL

CALL PRINT\_NUMBER\_EBX

JMP CNG\_Z

CALC\_Z\_0:

MOV ebx, X

MOV eax, Y

SHR ebx, 1

MOV ecx, ebx

SUB ecx, eax

CMP ECX, 0

JNS POSIT

SUB eax, ebx

MOV ebx, eax

BTS ebx, 19

JMP SAP

POSIT:

SUB ebx, eax

SAP:

MOV Z, ebx

PUTL EMPTYS

PUTL Z\_EQUAL

CALL PRINT\_NUMBER\_EBX

JMP CNG\_Z

CNG\_Z:

MOV ebx, Z

BT ebx, 2

JC ZERO

BTS ebx, 6

JMP Z\_2

ZERO:

BTR ebx, 6

Z\_2:

BT ebx,12

JC Z\_3

BTR ebx, 16

Z\_3:

BT ebx, 13

JNC FIN\_Z

BTS ebx, 11

FIN\_Z:

PUTL EMPTYS

PUTL EMPTYS

PUTL AFTER\_CHANGE

PUTL Z\_EQUAL

CALL PRINT\_NUMBER\_EBX

PUTL EMPTYS

JMP @@E

PRINT:

PUTL EMPTYS

MOV ebx, X

CALL PRINT\_NUMBER\_EBX

PUTL EMPTYS

MOV ebx, Y

CALL PRINT\_NUMBER\_EBX

PUTL EMPTYS

JMP @@E

PROC\_TIME: CMP AL, '-' ; удлиннять задержку?

JNE CMINUS

INC PAUSE+2 ; добавить 65536 мкс

JMP @@L

CMINUS: CMP AL, '+' ; укорачивать задержку?

JNE CEXIT

CMP WORD PTR PAUSE+2, 0

JE BACK

DEC PAUSE+2 ; убавить 65536 мкс

BACK: JMP @@L

CEXIT: CMP AL, CHESC

JE @@E

TEST AL, AL

JNE BACK

CALL GETCH

JMP @@L

; Выход из программы

@@E: EXIT

EXTRN PUTSS: NEAR

EXTRN PUTC: NEAR

EXTRN GETCH: NEAR

EXTRN GETECH: NEAR

EXTRN GETS: NEAR

EXTRN SLEN: NEAR

EXTRN PRINT\_NUMBER\_EBX: NEAR

EXTRN INP\_NUM\_EBX: NEAR

EXTRN UTOA10: NEAR

END BEGIN

**mopl1.mac**

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Макросы для учебного примера лабораторной работы N 1 \*

; по МОП. \*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Макрос вывода строки LINE

PUTL MACRO LINE

LEA SI, LINE

XOR CX, CX

XOR DX, DX

XOR AX, AX

CALL PUTSS

ENDM

; Макрос вывода строки LINE с задержкой PAUSE

PUTLS MACRO LINE

XOR AX, AX

LEA SI, LINE

; MOV DX, PAUSE

; MOV CX, PAUSE+2

CALL PUTSS

ENDM

; Как и PUTLS, но с центрированием посередине

PUTLSC MACRO LINE

LOCAL LO, NOMARG

LEA SI, LINE

CALL SLEN

MOV CX, 80

SUB CX, AX

SAR CX, 1

JZ NOMARG

LO: MOV AL, ' '

CALL PUTC

LOOP LO

NOMARG: PUTLS LINE

ENDM

; Макрос фиксации начала измерения времени

; Заносит в стек младшее слово числа тактов системных часов

FIXTIME MACRO

DB 0Fh, 31h; Команда RDTSC чтения счетчика тактов

PUSH EDX

PUSH EAX

ENDM

; Макрос получения в <EDX,EAX> числа тактов, прошедших с последнего

; момента фиксации времени. Извлекает продукт FIXTIME из стека

DURAT MACRO

DB 0Fh, 31h; Команда RDTSC чтения счетчика тактов

POP EBX

SUB EAX, EBX

POP EBX

SBB EDX, EBX

ENDM

; Макрос завершения программы

EXIT MACRO

MOV AH, FUEXIT

MOV AL, 0

INT DOSFU

ENDM

**mopl1l.asm**

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Процедуры для учебного примера лабораторной работы N 1 \*

; по МОП. \*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

.MODEL SMALL

.CODE

.386

INCLUDE MOPL1.INC

LOCALS

;=====================================================

; Подпрограмма вывода на экран строки, адресуемой SI,

; с задержкой времени между символами в <CX,DX> mcs.

; Завершителями строки являеются байты 0 или 0FFh.

; ЕСЛИ строка заканчивается байтом 0,

; ТО добавляется переход в начало новой строки

;

;=====================================================

PUTSS PROC NEAR

@@L: MOV AL, [SI]

CMP AL, 0FFH

JE @@R

CMP AL, 0

JZ @@E

CALL PUTC

INC SI

CALL DILAY

JMP SHORT @@L

; Переход на следующую строку

@@E: MOV AL, CHCR

CALL PUTC

MOV AL, CHLF

CALL PUTC

@@R: RET

PUTSS ENDP

;==============================================

; Подпрограмма вывода AL на терминал

;==============================================

PUTC PROC NEAR

PUSH DX

MOV DL, AL

MOV AH, FUPUTC

INT DOSFU

POP DX

RET

PUTC ENDP

;==============================================

; Подпрограмма ввода символа в AL с терминала

;==============================================

GETCH PROC NEAR

MOV AH, FUGETCH

INT DOSFU

RET

GETCH ENDP

;==============================================

; Подпрограмма посимвольного считывания с терминала

;==============================================

GETECH PROC NEAR

MOV AH, 1h

INT DOSFU

RET

GETECH ENDP

;==============================================

; Подпрограмма вывода ЕВХ на терминал

;==============================================

PRINT\_NUMBER\_EBX PROC NEAR

PUSH ebx

MOV cx, 19

CICLE: BT ebx, 19

JNC ZERO

MOV AH, 2h

MOV DL, '1'

INT 21h

JMP INCREM

ZERO:

MOV AH, 2h

MOV DL, '0'

INT 21h

INCREM:

SHL ebx, 1

LOOP CICLE

R: POP ebx

RET

PRINT\_NUMBER\_EBX ENDP

;==============================================

; Подпрограмма ввода числа ЕВХ с терминала

;==============================================

INP\_NUM\_EBX PROC NEAR

MOV cx, 19

XOR ebx, ebx

READ\_NEXT:

CALL GETECH

CMP AL, '1'

JNE NOT\_E

BTS ebx, 0

SHL ebx, 1

JMP DECREM

NOT\_E:

CMP AL, '0'

JNE FINISH

SHL ebx, 1

DECREM:

CMP cx, 0

JE RETURN

DEC cx

JMP READ\_NEXT

FINISH:

CMP cx, 0

JE RETURN

;;SHR ebx, 1

DEC cx

JMP FINISH

RETURN: RET

INP\_NUM\_EBX ENDP

;=================================================

; Подпрограмма ввода строки в буфер, адресуемый DX

; и имеющий структуру:

; { char size; // размер буфера

; char len; // реально введено

; char str[size]; // символы строки }

;=================================================

GETS PROC NEAR

PUSH SI

MOV SI, DX

MOV AH, FUGETS

INT DOSFU

; прописать байт 0 в конец строки

XOR AH, AH

MOV AL, [SI+1]

ADD SI, AX

MOV BYTE PTR [SI+2], 0

POP SI

RET

GETS ENDP

;==============================================

; Подпрограмма подсчета числа символов в строке,

; адресуемой SI. Завершители строки: 0 и 0FFh

; Результат возвращается в AX

;==============================================

SLEN PROC NEAR

XOR AX, AX

LSLEN: CMP BYTE PTR [SI], 0

JE RSLEN

CMP BYTE PTR [SI], 0FFh

JE RSLEN

INC AX

INC SI

JMP SHORT LSLEN

RSLEN: RET

SLEN ENDP

;====================================================

; Подпрограмма преобразования <EDX,EAX> в беззнаковое

; десятичное, размещаемое по адресу DI

;==============================================

.DATA

UBINARY DQ 0 ; Исходное двоичное 64-разрядное

UPACK DT 0 ; Упакованные 18 десятичных цифр

.CODE

UTOA10 PROC NEAR

PUSH CX

PUSH DI

MOV DWORD PTR [UBINARY], EAX

MOV DWORD PTR [UBINARY+4], EDX

FINIT ; инициализация сопроцессора

FILD UBINARY ; забрасывание в него бинарного

FBSTP UPACK ; извлечение упаковонного десятичного

MOV CX, LENPACK ; получено 9 пар цифр

PUSH DS ; писать

POP ES ; будем

CLD ; через stosw

LEA SI, UPACK ; с конца

ADD SI, LENPACK ; буфера upack

; Цикл преобразования пар полубайтов в ASCII-коды цифр

@@L: XOR AX, AX

DEC SI

MOV AL, [SI]

SHL AX, 4

SHR AL, 4

ADD AX, 3030h

XCHG AL, AH

STOSW

LOOP @@L

; Фиксация конца строки

XOR AL, AL

STOSB

; Улучшим читабельность слишком длинного числа

CLD

MOV AX, LENNUM-4

@@L1: MOV CX, AX

POP DI ; встаем на начало строки

PUSH DI

MOV SI, DI

INC SI

REP MOVSB

MOV BYTE PTR [DI], CHCOMMA ; вставить разделитель троек цифр

SUB AX, 4 ; 3 цифры + разделитель обработаны

JS @@E ; прекратить, если осталось не больше 3-х цифр

JMP SHORT @@L1

@@E: POP SI

PUSH SI

XOR CX, CX

; Съедаем первые нули

; сначала подсчитываем

@@L2: CMP BYTE PTR [SI], '0'

JE @@N

CMP BYTE PTR [SI], CHCOMMA

JNE @@N1

@@N: INC CX

INC SI

JMP SHORT @@L2

@@N1: ; а теперь съедаем

POP DI

SUB CX, LENNUM+1

NEG CX

REP MOVSB

POP CX

RET

UTOA10 ENDP

;==============================================

; Подпрограмма задержки выполнения программы

; на <CX,DX> микросекунд

;==============================================

DILAY PROC NEAR

MOV AH, 86h

INT 15h

RET

DILAY ENDP

PUBLIC PUTSS, PUTC, GETCH, GETS, DILAY, SLEN, UTOA10, GETECH, PRINT\_NUMBER\_EBX, INP\_NUM\_EBX

END

**mopl1.inc**

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Константы для учебного примера лабораторной работы N 1 \*

; по МОП. \*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

CHESC = 1Bh

CHCR = 13

CHSEP = '='

CHCOMMA = '`'

CHLF = 10

DOSFU = 21h

FUPUTC = 2

FUGETCH = 8

FUGETS = 0Ah

FUEXIT = 4Ch

PAUSE\_L = 0000 ; Младшее слово стартового значения паузы

PAUSE\_H = 0 ; Старшее слово

LENNUM = 18 ; Число символов в строке десятичного числа

LENPACK = 9 ; Упаковонное в два раза короче