**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ,**

**СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ**

**УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА»**

**(СПбГУТ)**

**Санкт-Петербургский колледж телекоммуникаций им Э.Т. Кренкеля**

**Отчёт о выполнении лабораторной работы №4**

**по дисциплине "Системное программирование"**

**Тема:** Низкоуровневое программирование.

Технология создания ассемблерных программ.

Принял: преподаватель Кривоносова Н.В

Выполнил: студенты группы ЗФ-053/054

Фомина Е.А.

Ткачук А.В

Шилаковски М.Э

Семененков К.И

Санкт-Петербург

2022 год

**Низкоуровневое программирование.**

**Технология создания ассемблерных программ.**

1. **Цель работы**

В соответствии с рабочей программой по дисциплине «Системное программирование»

в результате выполнения заданий по лабораторным работам студент должен:

уметь:

* освоить технику создания программ в SDK MASM32;
* изучить использование шаблонов ассемблерных программ;
* экспериментально изучить работу макрогенератора;
* изучить технику создания и применения макросов.
* научиться наблюдать за результатами работы макрогенератора, встроенного в ml.exe.

1. **Задания**
   1. **Создание консольной программы**
      1. Скопируем файл textio.asm

.386

.model flat, stdcall

option casemap :none

include \masm32\include\windows.inc

include \masm32\include\user32.inc

include \masm32\include\kernel32.inc

include \masm32\include\masm32.inc

includelib \masm32\lib\user32.lib

includelib \masm32\lib\kernel32.lib

includelib \masm32\lib\masm32.lib

print MACRO Quoted\_Text:VARARG

LOCAL Txt

.data

Txt db Quoted\_Text,0

.code

invoke StdOut,ADDR Txt

ENDM

input MACRO Quoted\_Prompt\_Text:VARARG

LOCAL Txt

LOCAL Buffer

.data

Txt db Quoted\_Prompt\_Text,0

Buffer db 128 dup(?)

.code

invoke StdOut,ADDR Txt

invoke StdIn,ADDR Buffer,LENGTHOF Buffer

mov eax, offset Buffer

ENDM

cls MACRO

invoke ClearScreen

ENDM

Main PROTO

.data

Msg1 db "Type something > ",0

Msg2 db "You typed > ",0

.code

start:

invoke Main

invoke ExitProcess,0

Main proc

LOCAL InputBuffer[128]:BYTE

cls

print "Console function test",13,10,13,10

input "Enter Some Text > "

invoke StdOut,eax

invoke locate,10,10

invoke StdOut,ADDR Msg1

invoke StdIn,ADDR InputBuffer,LENGTHOF InputBuffer

invoke locate,10,11

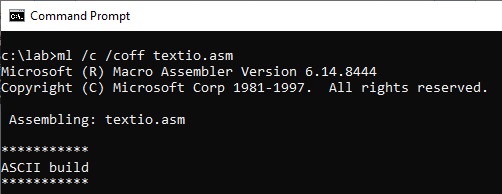
invoke StdOut,ADDR Msg2

invoke StdOut,ADDR InputBuffer

ret

Main endp

end start

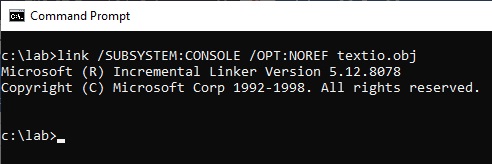
* + 1. ****

Создался файл textio.obj

* + 1. Значение ключей:

/c - компиляция без связывания, предотвращает автоматический вызов LINK, создаются только OBJ-файлы

/coff – включает COFF-формат (Common Object File Format) для создания обычных Win32-программ

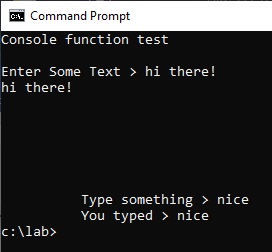
* + 1. ****

Создался файл textio.exe

* + 1. Значение ключей:

/SUBSYSTEM:CONSOLE - выполняет компоновку для 32-битной консольной системы Windows, необходим для линковки консольных приложений

/OPT:NOREF - управляет оптимизацией, выполняемой во время сборки с помощью программы LINK, хранит функции и данные, на которые нет ссылок

* + 1. 
  1. **Создание программы с графическим интерфейсом**
     1. Скопируем файл minimum.asm

.386

.model flat, stdcall

option casemap :none ; case sensitive

include \masm32\include\windows.inc

include \masm32\include\user32.inc

include \masm32\include\kernel32.inc

includelib \masm32\lib\user32.lib

includelib \masm32\lib\kernel32.lib

.code

start:

jmp @F

szDlgTitle db "Minimum MASM",0

szMsg db " --- Assembler Pure and Simple --- ",0

@@:

push MB\_OK

push offset szDlgTitle

push offset szMsg

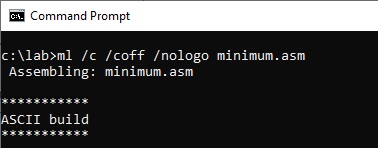
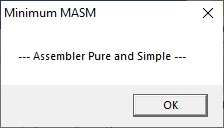
push 0

call MessageBox

push 0

call ExitProcess

end start

* + 1. ****
    2. ключ /nologo - подавляет сообщения о версии компилятора и авторских правах
    3. Создан файл minimum.exe
    4. ****
  1. **Использование шаблона простой программы**
     1. Рассмотрим build.bat

@echo off

if exist %1.obj del %1.obj

if exist %1.exe del %1.exe

\masm32\bin\ml /c /coff %1.asm

if errorlevel 1 goto errasm

if not exist rsrc.obj goto nores

\masm32\bin\Link /SUBSYSTEM:WINDOWS /OPT:NOREF %1.obj rsrc.obj

if errorlevel 1 goto errlink

dir %1.\*

goto TheEnd

:nores

\masm32\bin\Link /SUBSYSTEM:WINDOWS /OPT:NOREF %1.obj

if errorlevel 1 goto errlink

dir %1.\*

goto TheEnd

:errlink

echo \_

echo Link error

goto TheEnd

:errasm

echo \_

echo Assembly Error

goto TheEnd

:TheEnd

pause

* + 1. Рассмотрим buildc.bat

@echo off

if exist %1.obj del %1.obj

if exist %1.exe del %1.exe

\masm32\bin\ml /c /coff %1.asm

if errorlevel 1 goto errasm

if not exist rsrc.obj goto nores

\masm32\bin\Link /SUBSYSTEM:CONSOLE /OPT:NOREF %1.obj rsrc.obj

if errorlevel 1 goto errlink

dir %1.\*

goto TheEnd

:nores

\masm32\bin\Link /SUBSYSTEM:CONSOLE /OPT:NOREF %1.obj

if errorlevel 1 goto errlink

dir %1.\*

goto TheEnd

:errlink

echo \_

echo Link error

goto TheEnd

:errasm

echo \_

echo Assembly Error

goto TheEnd

:TheEnd

pause

* + 1. Создали hello.asm

include \masm32\include\masm32rt.inc

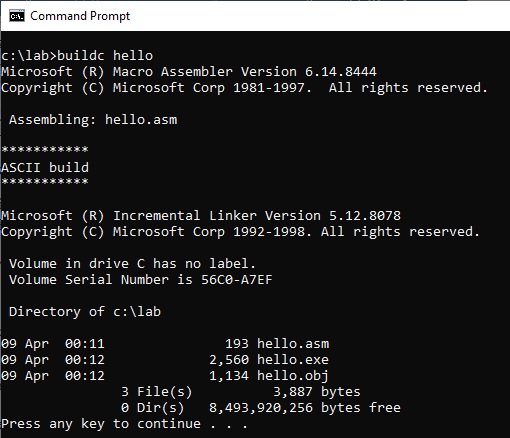
.data

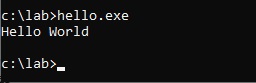
.code

start:

print "Hello World",13,10

end start

* + 1. ****
    2. Результат работы программы

****

* 1. **Создание рабочей папки d:\MACRO**

Шаблон m0.asm:

.486

.MODEL flat, stdcall

.data

.code

end

Пакетный файл mg.bat

@ echo off

chcp 1251 >nul

echo Обрабатывется файл %1.ASM.

@if exist %1.lst del %1.lst

\masm32\bin\ml /EP /nologo %1.asm > %1.lst

@echo.

echo Результат работы макрогенератора (%1.LST)

echo -------------------------------------------------

@type %1.lst

@pause

* 1. **Макроопределение и макрорасширение**
     1. Макроопределение

mdef MACRO

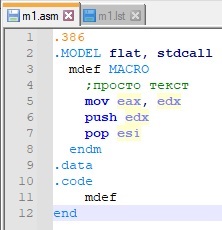
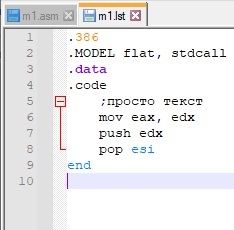
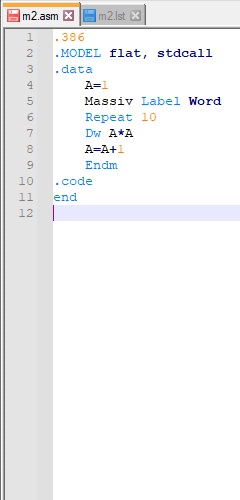
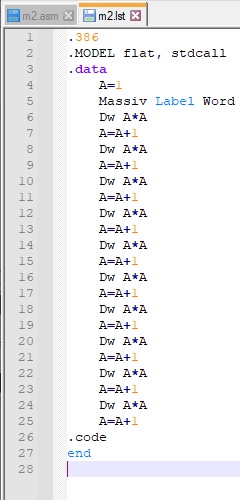
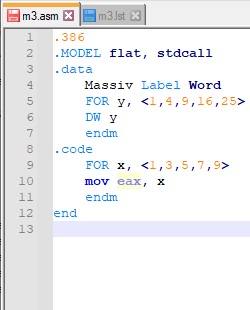
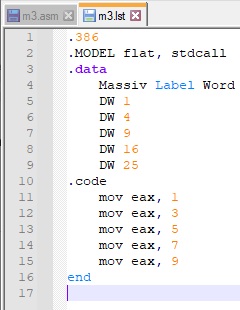
просто текст

mov eax, edx

push edx

pop esi

endm

* + 1. Макрогенератор пытался воспринимать строку «просто текст» как команду, закомментированная строка «просто текст» вывелась текстом. Скрытые комментарии остаются, но не показываются (как будто пустая строка)
    2. **..**
  1. **Повторение строк REPEAT**
     1. **. .**
     2. При макрорасширении цикл был разложен на шаги
  2. **Цикл FOR**
     1. **.gh  
         .hd**
     2. При макрорасширении цикл был разложен на шаги
  3. **Анализ расширенного листинга программы**
     1. **coff.asm**

include \masm32\include\masm32rt.inc

;EXTENDED LISTING

;< 1.>

macrodef MACRO par1, par2, par3

LOCAL M3

;;hiden comment

mov par2, par1

M3:

push par3

pop par2

endm

.data

;<2.>

Num1 dd 11223344h, 55667788h

Str1 db "ABCDEF", 0

.code

start:

;<3.>

Mov EAX, Num1 ; 11223344

Mov dl, [Str2 + 3] ; J

macrodef EAX, EDX, Num1

.data

;<4.>

Num2 dd 99aabbcch, 0ddeeff00h

Str2 db "GHIJKLM",0

.code

;<5.>

Mov EBX,[Num2 + 4] ; DDEEFF00

Mov dl, [Str1 +4] ; E

macrodef Num2, ESI, EAX

;<6.>

print "Hello, world!"

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

invoke ExitProcess,0

end start

* + 1. **coff.asm.ELST**

;EXTENDED LISTING

;< 1.>

macrodef MACRO par1, par2, par3

LOCAL M3

;hiden comment

mov par2, par1

M3:

push par3

pop par2

endm

00000000 .data

;<2.>

00000000 11223344 Num1 dd 11223344h, 55667788h

55667788

00000008 41 42 43 44 45 Str1 db "ABCDEF", 0

46 00

00000000 .code

00000000 start:

;<3.>

00000000 A1 00000000 R Mov EAX, Num1 ; 11223344

00000005 8A 15 0000001A R Mov dl, [Str2 + 3] ; J

macrodef EAX, EDX, Num1

1 LOCAL M3

1

0000000B 8B D0 1 mov EDX, EAX

0000000D 1 ??0019:

0000000D FF 35 00000000 R 1 push Num1

00000013 5A 1 pop EDX

0000000F .data

;<4.>

0000000F 99AABBCC Num2 dd 99aabbcch, 0ddeeff00h

DDEEFF00

00000017 47 48 49 4A 4B Str2 db "GHIJKLM",0

4C 4D 00

00000014 .code

;<5.>

00000014 8B 1D 00000013 R Mov EBX,[Num2 + 4] ; DDEEFF00

0000001A 8A 15 0000000C R Mov dl, [Str1 +4] ; E

macrodef Num2, ESI, EAX

1 LOCAL M3

1

00000020 8B 35 0000000F R 1 mov ESI, Num2

00000026 1 ??001A:

00000026 50 1 push EAX

00000027 5E 1 pop ESI

;<6.>

print "Hello, world!"

1 IFNDEF \_\_UNICODE\_\_

3 LOCAL nustr

3 LOCAL quot

= " 3 ??001C SUBSTR <"Hello, world!">,1,1

3 IFIDN ??001C,<">

3

3 IFNDEF \_\_UNICODE\_\_

0000001F 3 .data

0000001F 48 65 6C 6C 6F 3 ??001B db "Hello, world!",0

2C 20 77 6F

72 6C 64 21

00

00000028 3 .code

3 EXITM <OFFSET ??001B>

2 LOCAL prefix1,wrd,nu,varname

2

= O 2 ??001D SUBSTR <OFFSET ??001B>,1,1

2

2

2

2 IFIDN ??001D,<&>

2 ??001F SUBSTR <OFFSET ??001B>,2

2 ??001E CATSTR <ADDR >,??001F

2 EXITM <??001E>

2 ENDIF

2

2 IFIDN ??001D,<\*>

2 ??001F SUBSTR <OFFSET ??001B>,2

2 .data?

2 ??0020 dd ?

2 .code

2 push ebx

2 mov ebx, ??001F

2 mov ebx, [ebx]

2 mov ??0020, ebx

2 pop ebx

2 EXITM <??0020>

2 ENDIF

2

2 EXITM <OFFSET ??001B>

00000028 68 0000001F R \* push dword ptr OFFSET FLAT:??001B

0000002D E8 00000000 E \* call StdOut

1 invoke StdOut,expand\_prefix(reparg("Hello, world!"))

1 ELSE

1 invoke StdOutW,expand\_prefix(reparg("Hello, world!"))

1 ENDIF

1 IFNB <>

1 IFNDEF \_\_UNICODE\_\_

1 invoke StdOut,chr$()

1 ELSE

1 invoke StdOutW,chr$()

1 ENDIF

1 ENDIF

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

invoke ExitProcess,0

00000032 6A 00 \* push +000000000h

00000034 E8 00000000 E \* call ExitProcess

end start

* + 1. Компилятор строит расширенный листинг. Он просматривает исходный текст сверху вниз. Каждая строка исходного текста переносится в ELST «как есть». Если эта строка является макровызовом, то ниже вставляются строки макрорасширения. Они метятся слева числом – уровнем вложенности макрорасширения. Если строка исходника или вставленная строка макрорасширения является командой, то она компилируется. Что положено - пишется в объектный файл.obj.

1. **Контрольные вопросы.**
   1. **Какова функция программы ML.EXE?**

Это транслятор языка макроассемблера. Главная функция - из исходника перевести все команды ассемблера, набранные текстом в байты машинных команд.

* 1. **Какова функция программы LINK.EXE?**

Это линковщик. Главная функция - компоновать объектные файлы в исполняемые модули определённого формата.

* 1. **В чем разница в подготовке консольных приложений и оконных?**

Принципиальное отличие, только в одном флажке в заголовке и соответствующей различной обработке при запуске приложения. /SUBSYSTEM:CONSOLE или /SUBSYSTEM:WINDOW

* 1. **Что конкретно делает компоновщик во время компоновки ехе-модуля?**

Компоновка - это фактически «сборка» ехе-модуля из отдельных секций, которые извлекаются из объектных файлов и кода подключаемых объектных библиотек. В процессе компоновки все однородные секции объединяются -отдельные секции кода сливаются в одну соединённую секцию кода, аналогично отдельные секции данных соединяются в общую секцию данных. Общая секция кода и общая секция данных соединяются вместе в «загрузочный модуль». После этого он «разрешает» (вычисляет окончательно) неразрешенные (до этого окончательно недоопределенные) внешние ссылки. Наконец, компоновщик создает на диске выходной файл ехе-модуля.

* 1. **Чем отличаются интегрированная среда разработки и набор разработчика программ? К какому классу программных продуктов относится MASM32?**

IDE (Integrated Development Environment) – это единая среда разработки, комплекс инструментов, позволяющих создавать приложения. Иными словами – это программа, в которой создаются другие программы. SDK (Software Developer Kit) – работоспособный набор поддержки разработчиков, содержит компоновочные блоки, средства отладки, а зачастую фреймворк и группу библиотек кода. MASM32 относится к SDK.

* 1. **Какие особенности имеет инсталляция MASM32 на компьютер?**

MASM32 вовсе не компилятор, а сборник для программирования под Win32, в который входит 32-битный компилятор MASM. Достаточно просто разархивировать в удобное место. Для облегчения работы нужно прописать расположение в переменную PATH

* 1. **Какую роль выполняет в операционной системе переменная path?**

Переменная PATH содержит список директорий, в которых операционная система пытается искать исполняемые файлы, если пользователь при запуске не указал явно путь к нужному исполняемому файлу.

* 1. **Какими преимуществами обладает свободный текстовый редактор Notepad++?**

Полная бесплатность

Работа с большим количеством вкладок одновременно.

Легкость установки и минимальный размер установочного файла.

Подсветка синтаксиса.

Работа с большим количеством языков и форматов файлов.

Использование различных шрифтов для вывода фрагментов кода определенного типа.

Функция сворачивания блоков текста.

Функция автоматического завершения набираемого слова.

* 1. **Как задать обработчик для файлов исходных текстов на ассемблере?**

Выбрать файл. Нажать правой кнопкой мыши на документ. В меню «**открыть с помощью…**» выбрать, в нашем случае, Notepad++. Установить флаг «всегда использовать для файлов такого типа»

* 1. **Что такое макроопределение?**

Это описание, которое связывает текст макроса с его именем.

* 1. **Как выполняется расширение макроопределения с параметрами?**

Макрорасширение – это замена встретившегося в исходнике имени макроса на его тело. Макросы могут иметь формальные строковые параметры, которые перечисляются в строке заголовка после слова MACRO через запятую. Тогда при макровызове можно задать фактические параметры, которые будут подставлены на место формальных при макрорасширении. Фактический параметр заменяет в теле макроопределения формальный по принципу «строка заменяет строку».

* 1. **Как обеспечивается уникальность меток в исходном тексте при многократных вставках одного и того же макроса с метками?**

Для этого макрогенератор сам «выдумывает» уникальные имена меток при выполнении макрорасширений. Чтобы он это стал делать, надо объявить встречающиеся в теле макроса метки как «локальные». Это надо написать сразу после заголовка макроопределения. Макрогенератор в одной программе способен создать 0FFFFh = 65536 искусственных меток. Такое возможное их количество перекрывает все мыслимые практические потребности.

* 1. **Как получить расширенный листинг компиляции ассемблерного модуля?**

Добавить при трансляции ключ

/FL [имя\_файла] - создает листинг собранного кода.

/SA – это самый полный формат листинга, задаёт для листинга все опции формата.