Составил: Садовский Р.В.

Лабораторная работа №1
Установка и настройка среды разработки на языке низкого уровня

1 Цель работы

1.1 Установить и настроить среду разработки на языке низкого уровня;

1.2 Изучить средства компиляции и компоновки для языка низкого уровня;

1.3 Изучить процесс разработки на языке низкого уровня;

2 Литература

2.1 Ассемблер NASM. Первая программа на Windows – Текст : электронный // METANIT, 2025. – URL: https://metanit.com/assembler/nasm/1.5.php

2.2 SASM. Simple crossplatform IDE for NASM, MASM, GAS, FASM assembly languages – Текст : электронный // SASM, 2013. – URL: https://dman95.github.io/SASM/

3 Подготовка к работе

3.1 Повторить теоретический материал (см. п.2).

3.2 Изучить описание лабораторной работы.

4 Основное оборудование

4.1 Персональный компьютер.

5 Задание

5.1 Установка ассемблера NASM.

5.1.1 На официальном сайте перейти на страницу https://www.nasm.us/pub/nasm/releasebuilds/2.16.01/win64/ и загрузить zip-файл ассемблера NASM.

5.1.2 Распаковать архив, в нем должны быть 3 файла: файл лицензии, программа-ассемблер nasm.exe и программа-дизассемблер ndisasm.exe.

5.1.3 Запустить командную строку в каталоге распакованного архива и выполнить команду:

```
.\nasm -v
```

5.2 Компиляция ассемблерного кода в машинный код

5.2.1 Создать новый файл исходного кода hello.asm со следующим наплнением:

```
global _start        ; делаем метку метку _start видимой извне

section .text        ; объявление секции кода
_start:              ; метка _start - точка входа в программу
    mov rax, 22      ; произвольный код возврата - 22
    ret              ; выход из программы
```

5.2.2 Выполнить компиляцию файла hello.asm при помощи команды

```
nasm -f win64 hello.asm -o hello.o
```

После флага -f указывается формат выходного файла, для 64-разрядного файла указывается win64, а для 32-разрядного – win32 (elf64 и elf32 для Linux)

Флаг -o позволяет указать имя выходного файла компиляции

5.2.3 Открыть и изучить содержимое файла hello.o

5.3 Компоновка объектных файлов в исполняемые

5.3.1 Открыть меню **Пуск** и в разделе **VisualStudio 2022** открыть **x64 Native Tools Command Prompt for VS 2022**

5.3.2 Выполнить команду в открывшемся интерфейсе командной строки, и изучить вывод:

```
link
```

5.3.3 Выполнить компоновку файла hello.o при помощи команды:

```
link hello.o /entry:_start /subsystem:console /out:hello.exe
```

Параметр hello.o – входной файл

Параметр /entry:_start указывает точку входа прогаммы – метку _start

Параметр /subsystem:console указывает, что собирается консольное приложение

Параметр /out:hello.exe указывает имя выходного файла компоновки.

5.3.4 Запустить CMD в папке скомпонованной программы и выполнить программу в консоли:

```
hello.exe
echo %ERRORLEVEL%
```

В итоге выполнения команд должно быть выведено число 22 (код возврата указанный в программе)

5.4 Компоновка файлов с библиотеками

5.4.1 Создать файл hello2.asm с следующим содержимым:

```
global _start       ; делаем метку метку _start видимой извне

extern WriteFile        ; подключем функцию WriteFile
extern GetStdHandle     ; подключем функцию GetStdHandle

section .data   ; секция данных
message: db "Hello World!",10  ; строка для вывода на консоль

section .text       ; объявление секции кода
_start:             ; метка _start - точка входа в программу
    sub    rsp,  40      ;  Для  параметров  функций  WriteFile  и
GetStdHandle резервируем 40 байт (5 параметров по 8 байт)
    mov  rcx, -11  ; Аргумент для GetStdHandle - STD_OUTPUT
    call GetStdHandle ; вызываем функцию GetStdHandle
    mov  rcx, rax     ; Первый параметр WriteFile - в регистр RCX
помещаем дескриптор файла - консоли
    mov  rdx, message     ; Второй параметр WriteFile - загружаем
указатель на строку в регистр RDX
    mov  r8d, 18      ; Третий параметр WriteFile - длина строки
для записи в регистре R8D
    xor  r9, r9       ; Четвертый параметр WriteFile - адрес для
получения записанных байтов
    mov  qword [rsp + 32], 0  ; Пятый параметр WriteFile
```

```
    call WriteFile ; вызываем функцию WriteFile
    add  rsp, 40
    ret             ; выход из программы
```

5.4.2 Выполнить компиляцию файла hello2.asm, аналогично п.5.2.2

5.4.3 Выполнить компоновку файла hello2.o, аналогично п.5.3.3, но дополнительно указав в качестве входного файла библиотеку kernel32.lib для доступа к функциям WriteFile и GetStdHandle

5.4.4 Запустить скомпонованный файл hello2.exe при помощи командной строки

5.5 Загрузка IDE SASM

5.5.1 Загрузить последний релиз SASM в формате zip из репозитория https://github.com/Dman95/SASM

5.5.2 Распаковать архив и запустить файл sasm.exe

5.5.3 На главной странице кликнуть на кнопку «Создать новый проект» и изучить стандартный код, который генерируется IDE

5.5.4 Открыть файл с примером NASMHello.asm из папки SASM/Windows/Projects

5.5.5 Выполнить код, нажав на кнопку 

5.5.6 Наведясь на любую строку кода после метки main поставить точку останова. Запустить отладку нажав на кнопку  , выполнить код, проверить, что он останавливает выполнение на точке останова.

5.5.7 Снова запустить отладку и во вкладке Отладка поставить галочку в пункте Показать регистры. Изучить, как изменяются значения регистров в ходе выполнения кода по шагам (Шаг без захода F10/ с заходом F11)

5.5.8 Открыть настройки SASM, перейти во вкладку «Построение», изучить доступные опции. Попробовать изменить параметр Режим, посмотреть, как это повлияет на другие настройки.

5.5.9 Открыть вкладку «Помощь» изучить ее содержимое.

6 Порядок выполнения работы

6.1 Выполнить задания п.5.1-5.4

6.2 Изучить IDE SASM п.5.5

6.3 Ответить на контрольные вопросы.

7 Содержание отчета

7.1 Титульный лист

7.2 Цель работы

7.3 Скриншоты результатов выполненных заданий п.5

7.4 Ответы на контрольные вопросы

7.5 Вывод

8 Контрольные вопросы

8.1 Что такое ассемблер?

8.2 Что такое компилятор?

8.3 Что такое компоновщик?

8.4 Для чего предназначена IDE SASM?