|  |  |
| --- | --- |
| Gerb-BMSTU_01 | ***«*Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ \_\_\_\_\_\_\_ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ\_\_\_\_\_\_\_\_

КАФЕДРА \_\_\_\_\_\_\_\_\_КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ОТЧЕТ**

**по лабораторной работе № 1**

**Дисциплина: Машинно-зависимые языки и основы компиляции.**

**Название лабораторной работы: Изучение среды и отладчика ассемблера**

Студент гр. ИУ6-41Б **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ М. А. Тарасова**

(Подпись, дата) (И.О. Фамилия)

Преподаватель  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ С. С. Данилюк**

(Подпись, дата) (И.О. Фамилия)

Москва, 2023

**Цель работы:** изучение процессов создания, запуска и отладки программ

на ассемблере Nasm под управлением операционной системы Linux, а также

особенностей описания и внутреннего представления данных.

**Задание 1:** создать и запустить программу на языке Assembler.

**Код программы:**

section .data

ExitMsg db "Press Enter to Exit",10

lenExit equ $-ExitMsg

section .bss

InBuf resb 10

lenIn equ $-InBuf

section .text

global \_start

\_start:

mov rax, 1

mov rdi, 1

mov rsi, ExitMsg

mov rdx, lenExit

syscall

mov rax, 0

mov rdi, 0

mov rsi, InBuf

mov rdx, lenIn

syscall

mov rax, 60

xor rdi, rdi

syscall

**Результат программы (представлен на рисунке 1):**

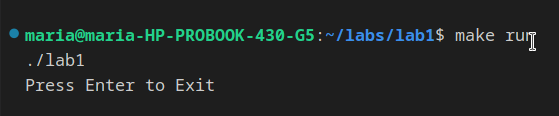


Рисунок 1 - результат программы

**Код Makefile:**

TARGET = lab1

help:

@echo Available goals:

@echo ' run - create and run without debugging '

@echo ' debug - create and debug '

@echo ' help - show this message '

$(TARGET): $(TARGET).asm

nasm -f elf64 -l $(TARGET).lst $(TARGET).asm

ld -o $(TARGET) $(TARGET).o

run: $(TARGET)

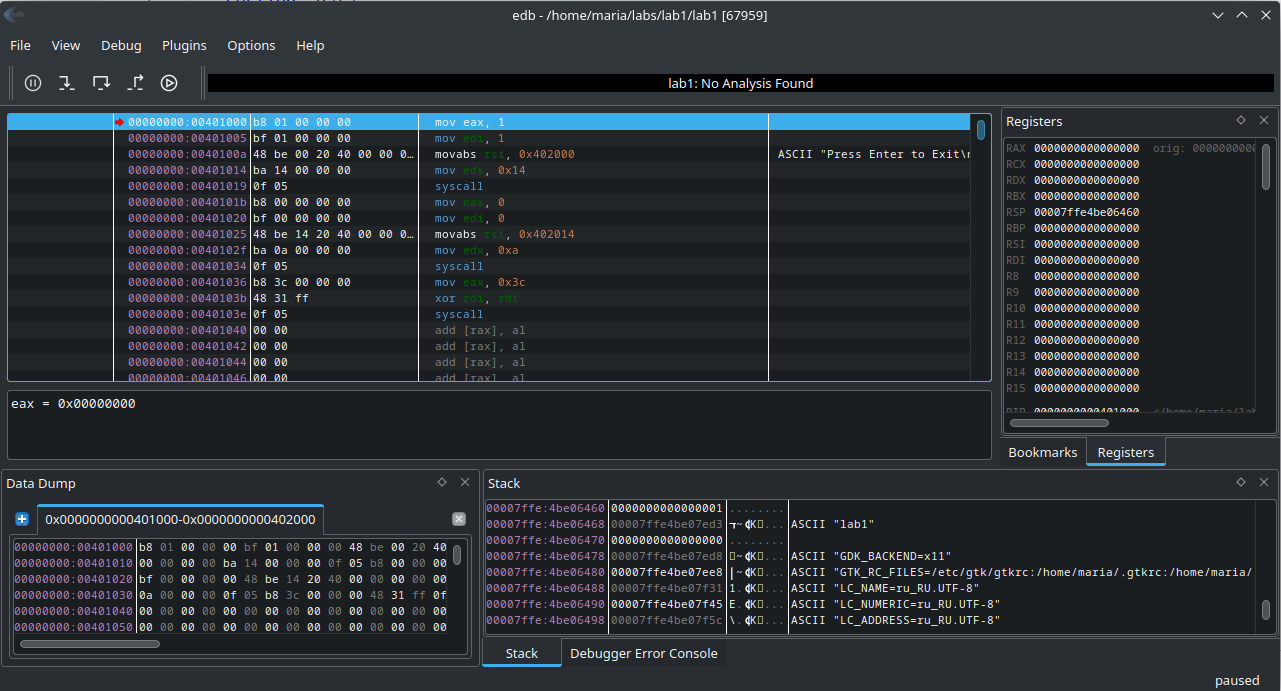
./$(TARGET)

debug: $(TARGET)

edb --run $(TARGET)

**Задание 2:** работа с отладчиком edb.

**Программа в отладчике (представлена на рисунке 2):**

Рисунок 2 - окно отладчика edb.

Добавляем данные в .data, .bssи .textдля изучения работы отладчика. Вычисляем результат выражения X=A+5-B.

**Код обновлённой программы:**

section .data

;val1 db 255

chart dw 256

lue3 dw -128

v5 db 10h

A dw -30

B dw 21

ExitMsg db "Press Enter to Exit",10

lenExit equ $-ExitMsg

section .bss

InBuf resb 10

lenIn equ $-InBuf

X resd 1

section .text

global \_start

\_start:

mov rax,[A] ; загрузить число A в регистр EAX

add rax,5; сложить EAX и 5, результат в EAX

sub rax,[B] ; вычесть число B, результат в EAX

mov [X],rax

mov rax, 1

mov rdi, 1

mov rsi, ExitMsg

mov rdx, lenExit

syscall

mov rax, 0

mov rdi, 0

mov rsi, InBuf

mov rdx, lenIn

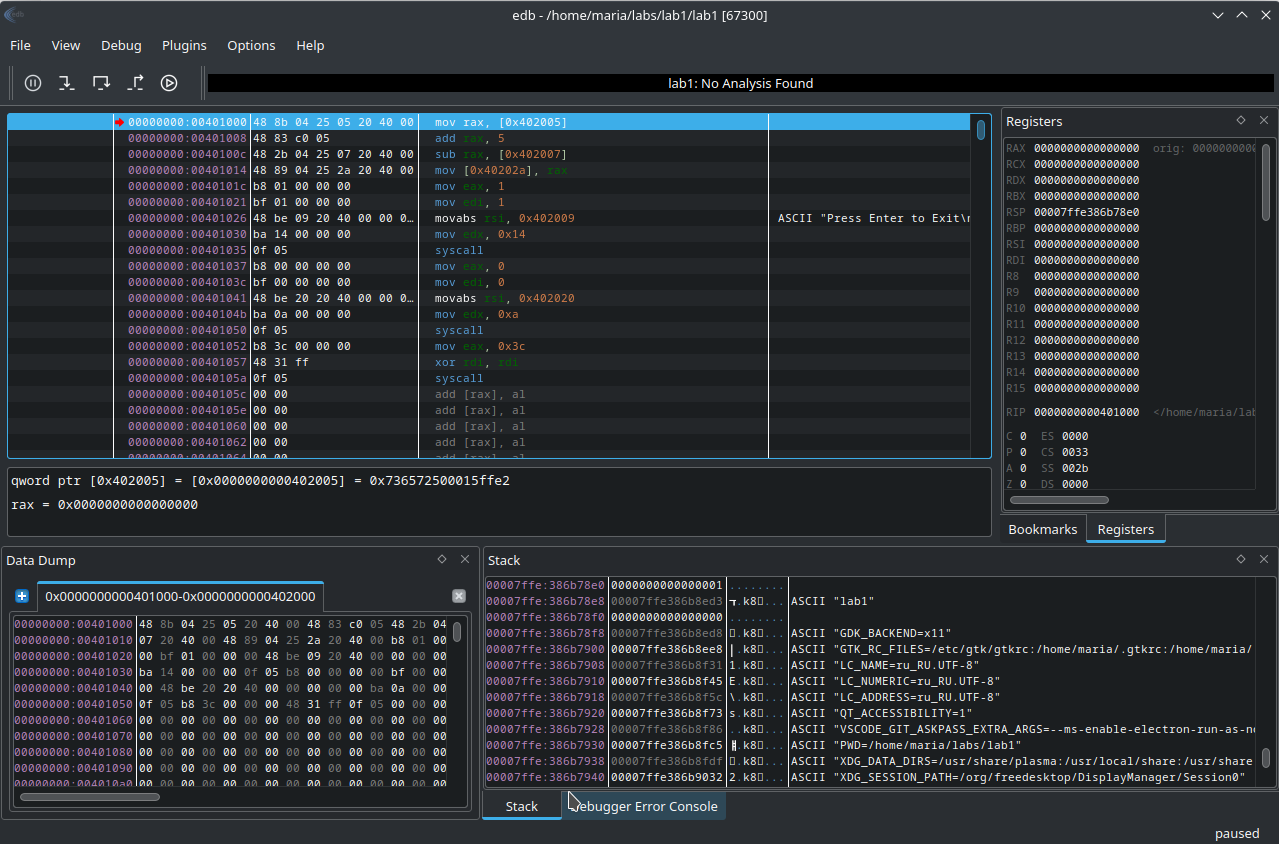
syscall

mov rax, 60

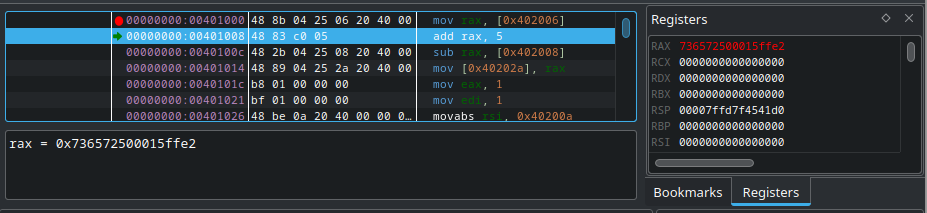
xor rdi, rdi

syscall

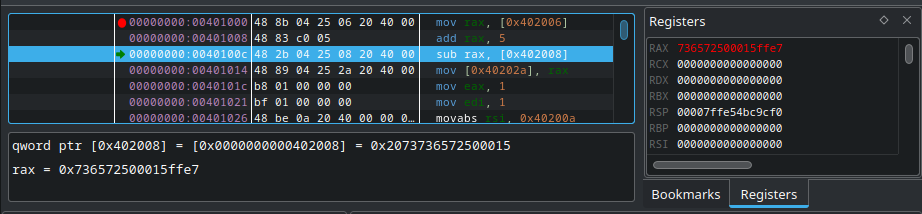
**Новый вид окна отладчика (представлен на рисунке 3):**

Рисунок 3 - обновлённое окно отладчика edb.

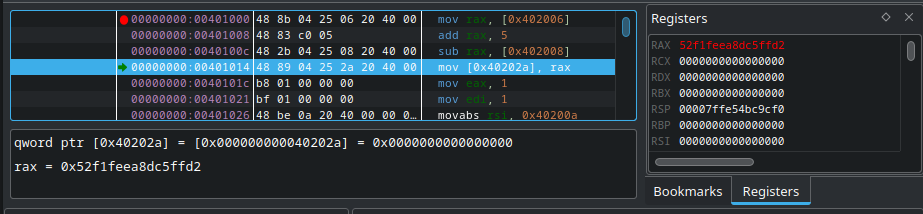
mov rax,[A] ; загрузить число A в регистр EAX

Рисунок 4 - загрузка числа А в регистр EAX.

add rax,5; сложить EAX и 5, результат в EAX

Рисунок 5 - сложение EAX и 5.

sub rax,[B] ; вычесть число B, результат в EAX

Рисунок 6 - вычитание числа В из ЕАХ.

Введём в раздел .data данные инициализированные переменные:

val1 db 255

chart dw 256

lue3 dw -128

v5 db 10h

db 100101B

beta db 23,23h,0ch

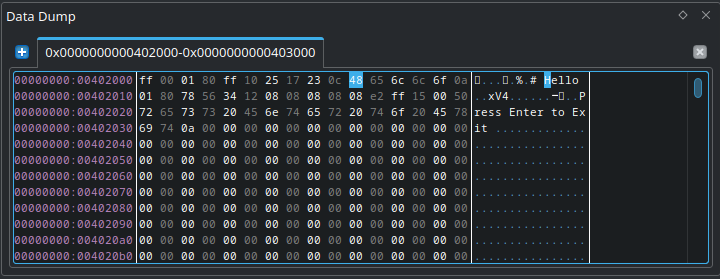
sdk db "Hello",10

min dw -32767

ar dd 12345678h

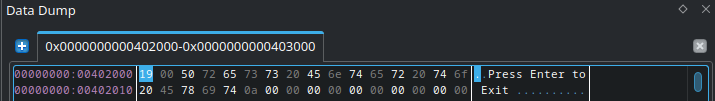
valar times 5 db 8

Их представление в отладчике (представлено на рисунке 7):

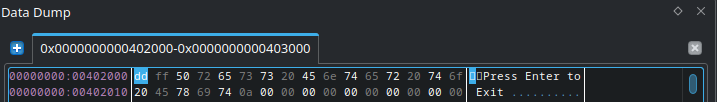
Рисунок 7 - представление инициализированных переменных в отладчике.

Определите в памяти следующие данные:

а) целое число 25 размером 2 байта со знаком (представлено на рисунке 8);

Рисунок 8 - представление 25 в памяти

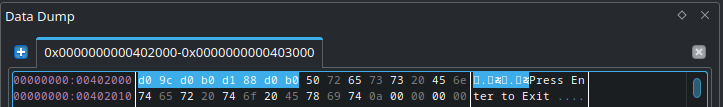
б) двойное слово, содержащее число -35 (представлено на рисунке 9);

Рисунок 9 - представление -35 в памяти

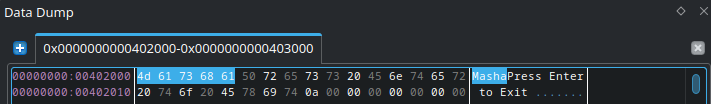
в) символьную строку, содержащую ваше имя (русскими буквами и

латинскими буквами) (представлено на рисунках 10, 11).

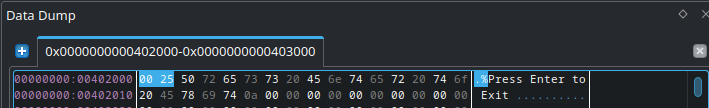
C db "Маша"

Рисунок 10 - представление строки «Маша» в памяти.

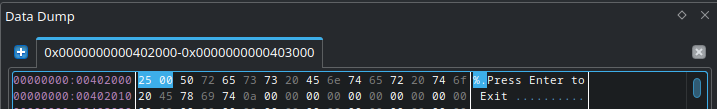
C db "Masha"

Рисунок 11 - представление «Masha» в памяти

Число, в отладчике выглядещее как 00 25: C dw 9472 (представлено на рисунке 12)

Рисунок 12 - представление в отладчике числа 9472

Число, в отладчике выглядещее как 25 00: C dw 37 (представлено на рисунке 13)

Рисунок 13 - представление в отладчике числа 37

Добавим в программу переменную F1=65535 размером слово и

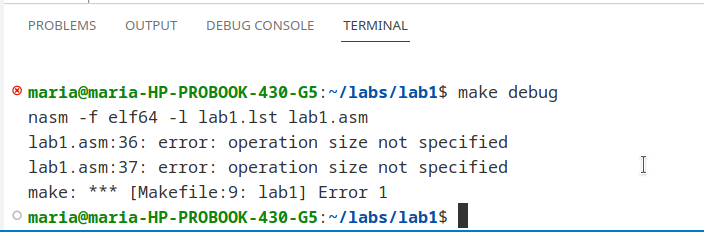
переменную F2= 65535 размером двойное слово. Вставьте в программу

команды сложения этих чисел с 1:

add [F1],1

add [F2],1

В итоге получим ошибку (представлена на рисунке 14):

Рисунок 14 - ошибка при попытке сложить литерал с переменной.

Ошибка произошла из-за того, что операции сложения и вычитания могут осуществляться с присутствием хотя бы одного регистра.

**Вывод:** в данной лабораторной работе мы изучали основы машинного языка Assembler, учились запускать программу в консоли и отладчике, а также смотрели представление данных в памяти.

**Контрольные вопросы:**

1. Дайте определение ассемблеру. К какой группе языков он относится?

Assembler - язык низкого уровня с командами, обычно соответствующими командам процессора (так называемым машинным командам). Он относится к группе машинно-зависимых, к которой относятся также машинные языки.

2. Из каких частей состоит заготовка программы на ассемблере?

Из секций для инициализированных переменных .data, секции неинициализированных переменных .bss и секции кода программы .text.

3. Как запустить программу на ассемблере на выполнение? Что происходит с программой на каждом этапе обработки?

Предложенные в предыдущем разделе заготовки программ можно выполнить. Для этого их необходимо ввести в компьютер, преобразовать в машинный код и скомпоновать со всеми требуемыми подпрограммами. Первую операцию выполняет текстовый редактор, вторую – транслятор, третью – компоновщик. И только после этого полученную выполняемую программу можно запустить на выполнение.

4. Назовите основные режимы работы отладчика. Как осуществить пошаговое выполнение программы и просмотреть результаты выполнения машинных команд.

Отладчик позволяет выполнять программу по шагам, заходя или не заходя в подпрограммы, и при этом анализировать изменения содержимого регистров, памяти и стека. Для этого используют соответствующие пункты меню или следующие клавиши:

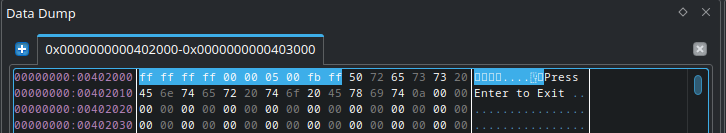
- F7 — выполнить шаг с заходом в подпрограмму;

- F8 — выполнить шаг без захода в подпрограмму;

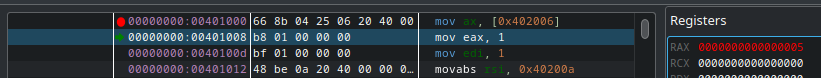
- Ctrl + F2 — начать отладку сначала;

- Ctrl + F9 — выполнить подпрограмму до возврата из нее.

5. В каком виде отладчик показывает положительные и отрицательные целые числа? Как будут представлены в памяти числа: A dw 5,-5 ?

Рисунок 15 - представление в памяти чисел 5, -5

Как те же числа будут выглядеть после загрузки в регистр AX?

Рисунок 16 - числа 5, -5 в регистре АХ

6. Каким образом в ассемблере программируются выражения? Составьте

фрагмент программы для вычисления С=A+B, где A, В и С – целые числа

формата BYTE.

section .data

A db ?

B db ?

section .bss

C resb 1

section .text

global \_start

\_start:

mov rax, [A]

add rax, [B]

mov [C], rax