|  |  |
| --- | --- |
| **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 (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.03 Прикладная информатика.**

**ОТЧЕТ**

|  |  |
| --- | --- |
| **по лабораторной работе №** | 2 |

Название: Программирование целочисленных вычислений.

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент |  |  |  |  |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | Я.С. Петрова |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2023

**Цель работы:** изучение форматов машинных команд, команд целочисленной арифметики ассемблера и программирование целочисленных вычислений.

**Задание:** Разработать программу на языке ассемблера, вычисляющую выражение (см. рисунок 1)

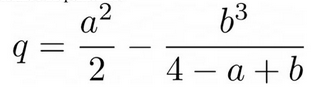


Рисунок 1 — Задание

**Задания 2.2.1 - 2.2.2**

Прочитайте и проанализируйте свой вариант задания. Определите

последовательность вычислений и разработайте программу в соответствии с заданием своего варианта. Введите программу в компьютер с использованием текстового редактора и выполните ее трансляцию и компоновку. Если при этом в программе были обнаружены ошибки, то исправьте их.

**Код программы**

Код программы представлен на Рисунках 2 - 5.

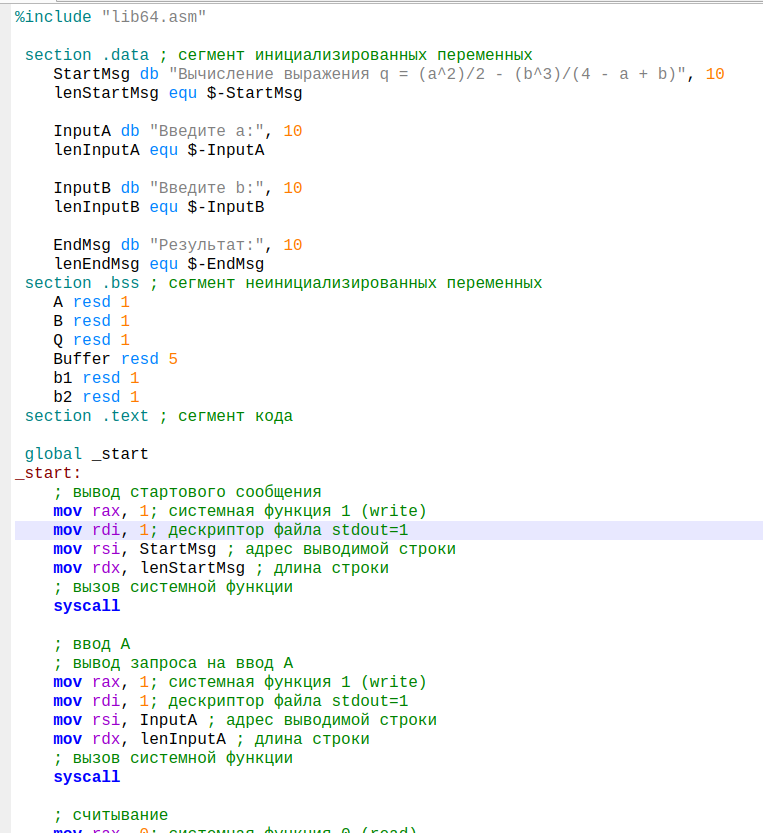


Рисунок 2 - Код программы (Часть 1)

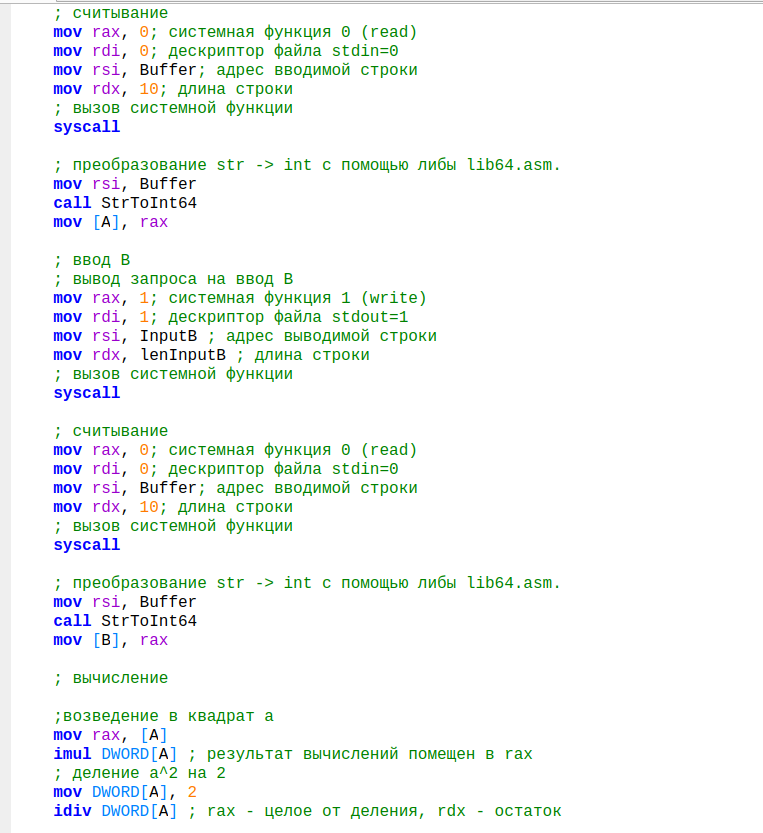


Рисунок 3 - Код программы (Часть 2)

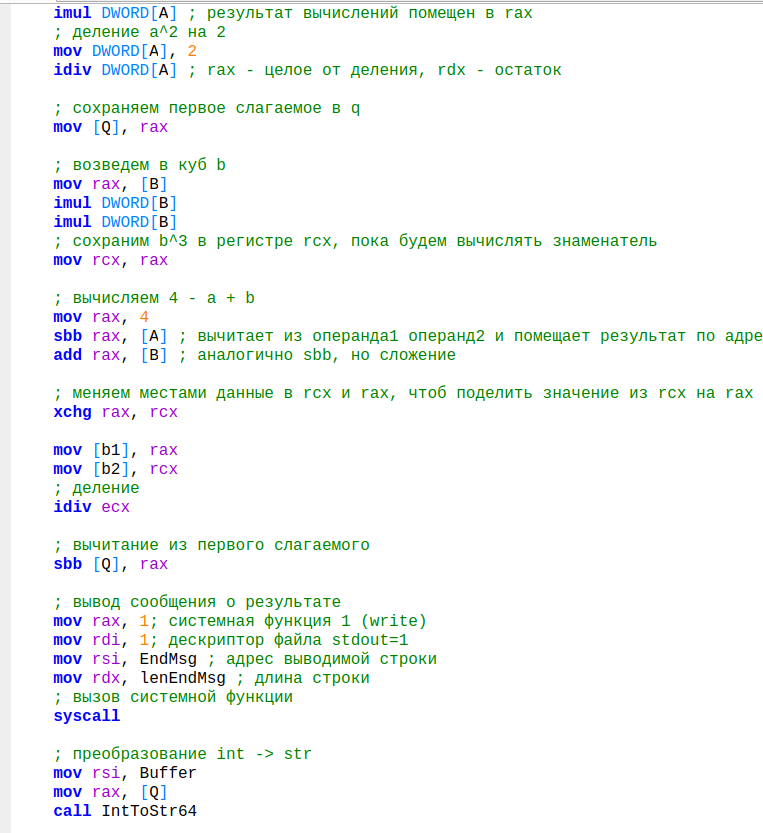


Рисунок 4 - Код программы (Часть 3)

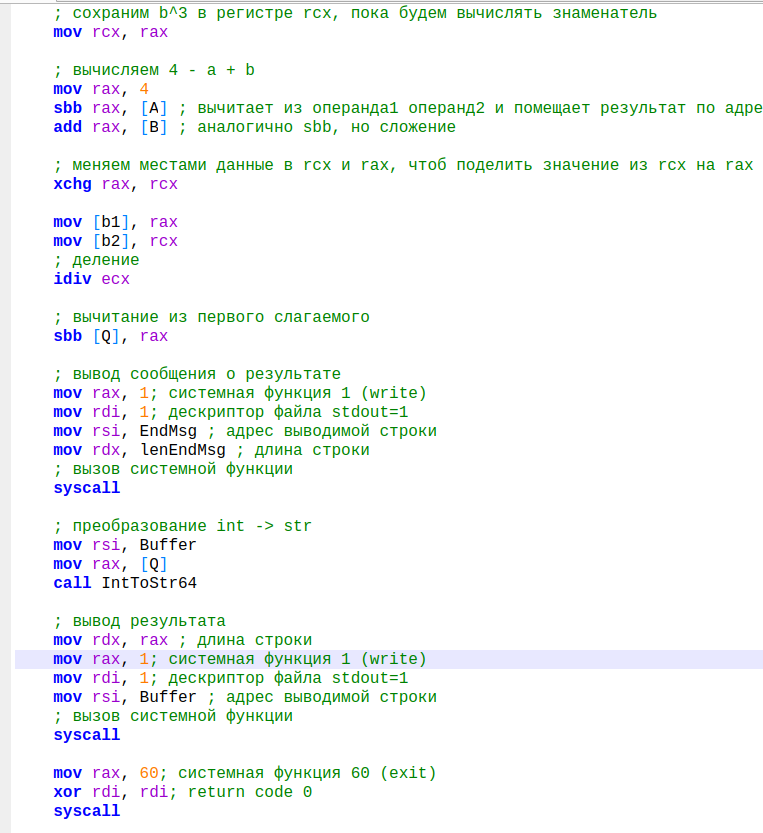


Рисунок 5 - Код программы (Часть 4)

**Задание 2.2.3**

Покомандно посмотрите в отладчике и зафиксируйте в отчете ход

выполнения вычислений. Убедитесь в правильности выполнения программы

на заданных данных.

Разделим программу на части ввода данных, подсчета D и вывода результата вычислений.

**1 Ввод данных**

На рисунке 6 представлен ввод переменных a, b.

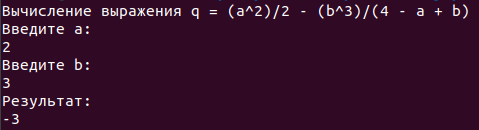


Рисунок 6 - Ввод переменных

При вводе значения записываются в буферные переменные, после вызывается функция StrToInt64, которая в свою очередь переводит преобразовывает строку в число. Проверим корректность перевода, зафиксировав содержимое переменных в отладчике:



Рисунок 7 - Содержимое переменной a



Рисунок 8 - Содержимое переменной b

**2 Вычисление Х**

**2.1 mov rax, [A]**

В регистр rax помещается содержимое переменной A. На рисунке 10 представлена команда в отладчике, на рисунке 11 - содержимое регистра rax после выполнения команды.



Рисунок 9 - Внутреннее представление команды 2.1



Рисунок 11 - Содержимое регистра RAX после выполнения команды 2.1

**2.2 imul DWORD[A]**

Содержимое регистра RAX умножается на операнд. Целое число помещается в RAX.



Рисунок 11 - Внутреннее представление команды 2.2



Рисунок 12 - содержимое регистра RAX после выполнения команды 2.2

**2.3 mov DWORD[A], 2**

Для последующих вычислений заносим в переменную A 2.



Рисунок 13 - Содержимое переменной A

**2.4 idiv DWORD[A]**

Содержимое регистра RAX делится на содержимое переменной A.



Рисунок 14 - Внутреннее представление команды 2.4



Рисунок 15 - Содержимое регистра RAX

**2.5 mov [Q], rax**

Сохраняем первое слагаемое в переменной Q.



Рисунок 16 - Внутреннее представление команды 2.5



Рисунок 17 - Содержимое переменной Q

**3 Вывод данных**

Для вывода данные сперва переводятся в строку, что показано на рисунке 4 при помощи функции IntToStr.

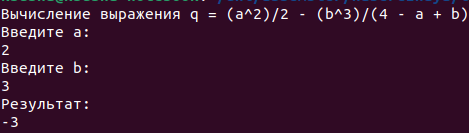


Рисунок 18 - Вывод результата

**Задание 2.2.4.**

Посмотрите в отладчике форматы 3-4-х команд mov и расшифруйте

двоичные коды этих команд, используя материалы лекций и раздела 2.1.1.

**1. mov rax, [A]**

Обратимся к внутреннему представлению команды (рисунок 10).

1. Байт 1: 48.

Этот байт определяет префикс размера операнда и указывает на то, что операнды команды имеют размер 64 бита.

1. Байт 2: 8B.

Этот байт содержит опкод команды mov.

1. Байт 3: 04.

Этот байт указывает на регистр-индекс, который используется для адресации в памяти. В данном случае, это регистр rax.

1. Байт 4: 25.

Этот байт указывает на модификацию команды mov для работы с абсолютным адресом. Здесь используется смещение (offset), указывающее на адрес [A].

1. Байты 5-8: 80 20 40 00.

Эти байты представляют собой смещение (offset) адреса [A] в памяти. Значение смещения зависит от конкретного контекста и используемых адресных режимов.

**2. mov DWORD[A], 2**

1. Байт 1: C7

Биты 7-0: 1100 0111

Значение: Опкод команды mov с префиксом размера операнда DWORD.

1. Байт 2: 05

Биты 7-0: 0000 0101

Значение: Режим адресации памяти с смещением.

1. Байты 3-6: 78 56 34 12

Значение: Это 32-битное смещение (offset) адреса [A]. В данном случае, адрес [A] равен 0x12345678.

1. Байт 7: 02

Биты 7-0: 0000 0010

Значение: 2, которое будет помещено в память по адресу [A].

1. Байты 8-10: 00 00 00

Значение: 2, которое будет помещено в память по адресу [A].

**3. mov [Q], rax**

1. Байт 1: 48

Биты 7-0: 0100 1000

Значение: Префикс размера операнда (64-битное значение) и указывает на регистр RAX.

1. Байт 2: 89

Биты 7-0: 1000 1001

Значение: Опкод команды mov и указывает на адресную операцию с прямой адресацией.

1. Байт 3: 03

Биты 7-0: 0000 0011

Значение: Режим адресации памяти с использованием регистра [Q].

**Контрольные вопросы**

**1. Что такое машинная команда? Какие форматы имеют машинные команды процессора IA32? Чем различаются эти форматы?**

Машинная команда - инструкция машине, выполняемая ею автоматически без каких-либо дополнительных указаний и пояснений.

Размер машинной команды процессора IA-32 может меняться от 1 до 15 байт. Команда состоит из префикса (опционально), кода операции, байта mod, байта sib (опционально), смещения и непосредственно операнда.

Различают два типа форматов: содержащие и не содержащие байта sib: если содержимое поля m (r/m) не равно 100, то байт sib в команде отсутствует.

**2. Назовите мнемоники основных команд целочисленной арифметики. Какие форматы для них можно использовать?**

**• Add - сложение**

**• sub - вычитание**

**• mul - умножение**

**• imul - умножение со знаком**

**• div - деление**

**• idiv - деление со знаком**

Допустимые варианты для сложения и вычитания (см. Рисунок 21)

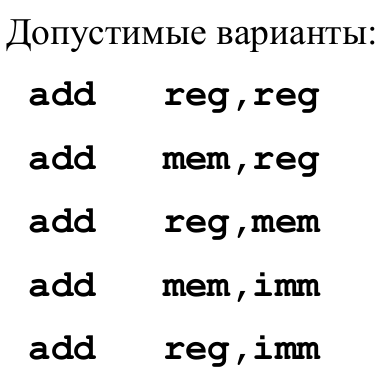


Рисунок 19 - Допустимые варианты для сложения и вычитания

Допустимые варианты для умножения (см. Рисунок 20)

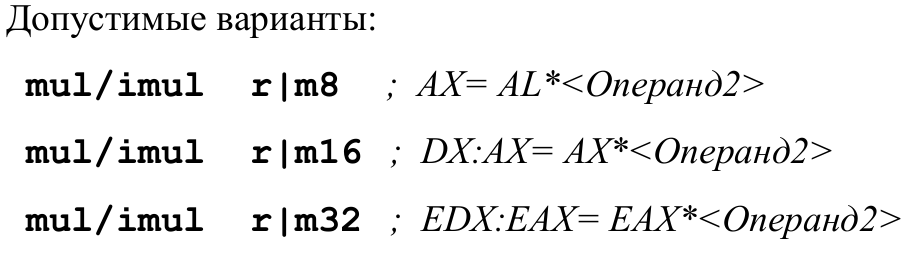


Рисунок 20 – форматы умножения

Допустимые варианты для деления (см. Рисунок 21)

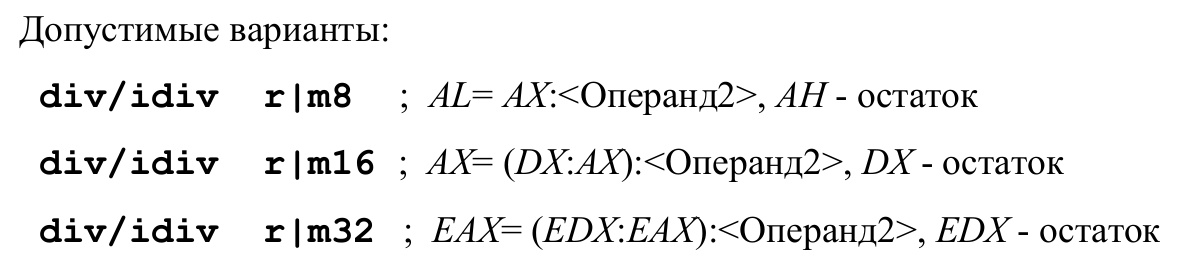


Рисунок 21 – форматы деления

**3.Сформулируйте основные правила построения линейной программы вычисления заданного выражения.**

* Заносим переменные выражения в секцию инициализированных данных .data
* Выделяем переменную для хранения результата выражения в секции неинициализированных данных .bss
* Код счета выражения в секцию кода .text
* При выделении памяти под переменные учитываем диапазон входных данных, чтобы не происходило ошибок переполнения.

**4.Почему ввод-вывод на языке ассемблера не программируют с использованием соответствующих машинных команд? Какая библиотека используется для организации ввода вывода в данной лабораторной?**

Ввод-вывод пишется без использования машинных команд, чтобы не было зависимостей от особенностей языка. В данной лабораторной работе используется библиотека std.

**5. Расскажите, какие операции используют при организации ввода-вывода.**

При организации ввода-вывод используют системные функции read/write, при этом необходимо указать дескриптор файла stdin(0)/stdout(1). Также используют syscall, который вызывает системную функцию.

**Вывод:** был освоен принцип калькулирования выражений на языке Ассемблер, освоена библиотека перевода строк в числа и наоборот, освоен ввод и вывод выражений.