Отчёт по лабораторной работе

Дисциплина: архитектура компьютера

Назармамадов Умед Джамшедович

Содержание

# 1 Цель работы

Цель данной лабораторной работы - освоить процедуры компиляции и сборки программ, написанных на ассемблере NASM.

# 2 Задание

1.Создание программы Hello World! 2.Работа с транслятором NASM. 3.Работа с расширенным синтаксисом командной строки NASM. 4.Работа с компоновщиком LD. 5.Запуск исполняемого файла. 6.Выполнение заданий для самостоятельной работы.

# 3 Теоретическое введение

Основными функциональными элементами любой электронно-вычислительной машины

(ЭВМ) являются центральный процессор, память и периферийные устройства. Взаимодействие этих устройств осуществляется через общую шину, к которой они подклю- чены. Физически шина представляет собой большое количество проводников, соединяющих устройства друг с другом. В современных компьютерах проводники выполнены в виде элек- тропроводящих дорожек на материнской (системной) плате. Основной задачей процессора является обработка информации, а также организация координации всех узлов компьютера. В состав центрального процессора (ЦП) входят следующие устройства: • арифметико-логическое устройство (АЛУ) — выполняет логические и арифметиче- ские действия, необходимые для обработки информации, хранящейся в памяти; • устройство управления (УУ) — обеспечивает управление и контроль всех устройств компьютера; • регистры — сверхбыстрая оперативная память небольшого объёма, входящая в со- став процессора, для временного хранения промежуточных результатов выполнения инструкций; регистры процессора делятся на два типа: регистры общего назначения и специальные регистры. Для того, чтобы писать программы на ассемблере, необходимо знать, какие регистры процессора существуют и как их можно использовать. Большинство команд в программах написанных на ассемблере используют регистры в качестве операндов. Практически все команды представляют собой преобразование данных хранящихся в регистрах процессора, это например пересылка данных между регистрами или между регистрами и памятью, пре- образование (арифметические или логические операции) данных хранящихся в регистрах.

Доступ к регистрам осуществляется не по адресам, как к основной памяти, а по именам.

Каждый регистр процессора архитектуры x86 имеет свое название, состоящее из 2 или 3 букв латинского алфавита. В качестве примера приведем названия основных регистров общего назначения (именно эти регистры чаще всего используются при написании программ): • RAX, RCX, RDX, RBX, RSI, RDI — 64-битные • EAX, ECX, EDX, EBX, ESI, EDI — 32-битные • AX, CX, DX, BX, SI, DI — 16-битные • AH, AL, CH, CL, DH, DL, BH, BL — 8-битные (половинки 16-битных регистров). Например, AH (high AX) — старшие 8 бит регистра AX, AL (low AX) — младшие 8 бит регистра AX.

Таким образом можно отметить, что вы можете написать в своей программе, например,

такие команды (mov – команда пересылки данных на языке ассемблера): mov ax, 1 mov eax, 1 Обе команды поместят в регистр AX число 1. Разница будет заключаться только в том, что вторая команда обнулит старшие разряды регистра EAX, то есть после выполнения второй команды в регистре EAX будет число 1. А первая команда оставит в старших разрядах регистра EAX старые данные. И если там были данные, отличные от нуля, то после выполнения первой команды в регистре EAX будет какое-то число, но не 1. А вот в регистре AX будет число 1. Другим важным узлом ЭВМ является оперативное запоминающее устройство (ОЗУ). ОЗУ — это быстродействующее энергозависимое запоминающее устройство, которое на- прямую взаимодействует с узлами процессора, предназначенное для хранения программ и данных, с которыми процессор непосредственно работает в текущий момент. ОЗУ состоит из одинаковых пронумерованных ячеек памяти. Номер ячейки памяти — это адрес хранящихся в ней данных. В состав ЭВМ также входят периферийные устройства, которые можно разделить на: • устройства внешней памяти, которые предназначены для долговременного хране- ния больших объёмов данных (жёсткие диски, твердотельные накопители, магнитные ленты); • устройства ввода-вывода, которые обеспечивают взаимодействие ЦП с внешней средой. В основе вычислительного процесса ЭВМ лежит принцип программного управления. Это означает, что компьютер решает поставленную задачу как последовательность действий, записанных в виде программы. Программа состоит из машинных команд, которые указыва- ют, какие операции и над какими данными (или операндами), в какой последовательности необходимо выполнить. Набор машинных команд определяется устройством конкретного процессора. Коды ко- манд представляют собой многоразрядные двоичные комбинации из 0 и 1. В коде машинной команды можно выделить две части: операционную и адресную. В операционной части хра- нится код команды, которую необходимо выполнить. В адресной части хранятся данные или адреса данных, которые участвуют в выполнении данной операции. При выполнении каждой команды процессор выполняет определённую последователь- ность стандартных действий, которая называется командным циклом процессора. В самом общем виде он заключается в следующем: 1. формирование адреса в памяти очередной команды; 2. считывание кода команды из памяти и её дешифрация; 3. выполнение команды; 4. переход к следующей команде.

Язык ассемблера (assembly language, сокращённо asm) — машинно-ориентированный

язык низкого уровня. Можно считать, что он больше любых других языков приближен к архитектуре ЭВМ и её аппаратным возможностям, что позволяет получить к ним более полный доступ, нежели в языках высокого уровня, таких как C/C++, Perl, Python и пр. Заметим, что получить полный доступ к ресурсам компьютера в современных архитектурах нельзя, самым низким уровнем работы прикладной программы является обращение напрямую к ядру операционной системы. Именно на этом уровне и работают программы, написанные на ассемблере. Но в отличие от языков высокого уровня ассемблерная программа содержит только тот код, который ввёл программист. Таким образом язык ассемблера — это язык, с помощью которого понятным для человека образом пишутся команды для процессора. # Выполнение лабораторной работы ##Создание программы Hello world! Создаю каталог и с помощью утилиты cd перемещаюсь в этот каталог.

Рис. 1: Перемещение между директориями

Рис. 1: Перемещение между директориями

Создаю в этом же каталоге пустой файл с помощью утилити touch с именем hello.asm.

Рис. 2: Создание пустого файла

Рис. 2: Создание пустого файла

Открываю созданный файл.

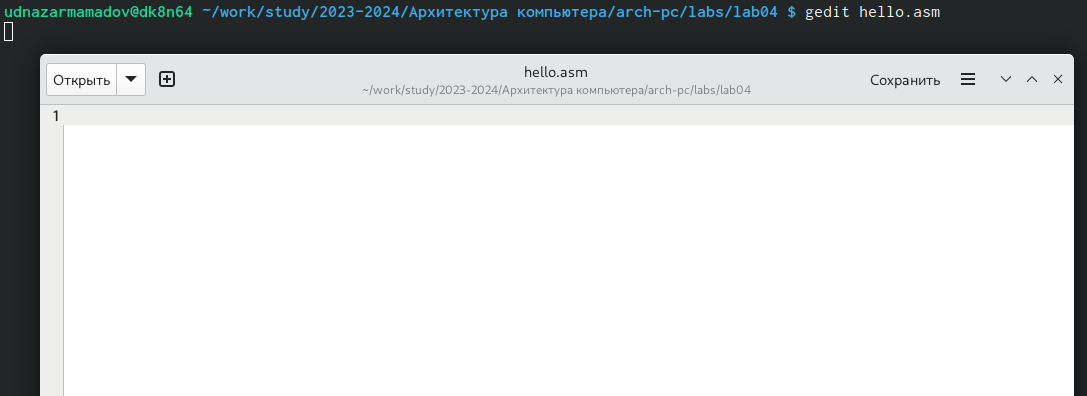


Рис. 3: Открытие файла

Заполняю файл, вставляя в него программу для вывода “Hello world!”.

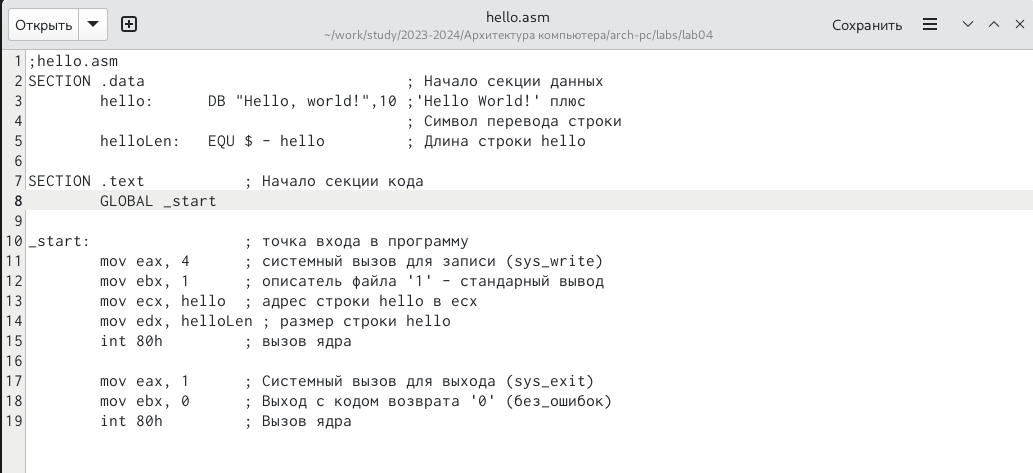


Рис. 4: Заполнение файла

##Работа с транслятором NASM

Текст который я писал в текстовом редакторе для вывода “Hello world!”, превращаю в код с помощью транслятора NASM. И проверяю правильность выполнения команды.

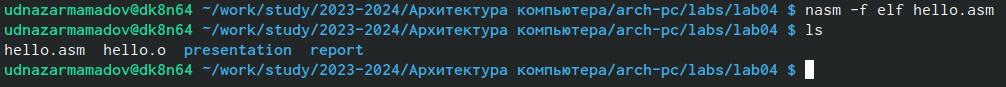


Рис. 5: Компиляция текста

##Работа с расширенным синтаксисом командной строки NASM

Ввожу команду, которая скомпилирует файл hello.asm в файл bj.о, при этом в файл будут включены символы для отладки (ключ -8), также с помощью ключа -1 будет создан файл листинга list Ist (рис. 4.6). Далее проверяю с помощью утилиты Is правильность выполнения команды.

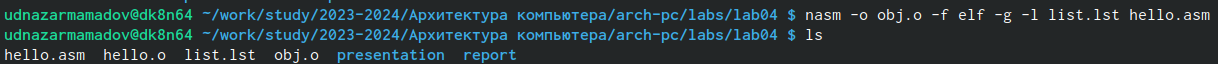


Рис. 6: Компиляция текста

##Работа с компоновщиком LD

Передаю объектный файл hello.о на обработку компоновщику LD, чтобы получить исполняемый файл hello. Ключ -о задает имя создаваемого исполняемого файла. Далее проверяю с помощью утилиты Is правильность выполнения команды.

Рис. 7: Передача файла на обработку

Рис. 7: Передача файла на обработку

Выполняю следующую команду. Исполняемый файл будет иметь имя таіп, т.к. после ключа -о было задано значение main. Объектный файл, из которого собран этот исполняемый файл, имеет имя bj.о.

Рис. 8: Передача файла на обработку

Рис. 8: Передача файла на обработку

##Запуск исполняемого файла

Запускаю созданный файл hello.

Рис. 9: Запуск файла

Рис. 9: Запуск файла

##Выполнение заданий для самостоятельной работы

Создаю в текущем каталоге копию файла hello.asm с именем lab4.asm.

Рис. 10: Создание копии файла

Рис. 10: Создание копии файла

Открываю файл и вношу изменения в программу, чтобы она выводила мои имя и фамилию.

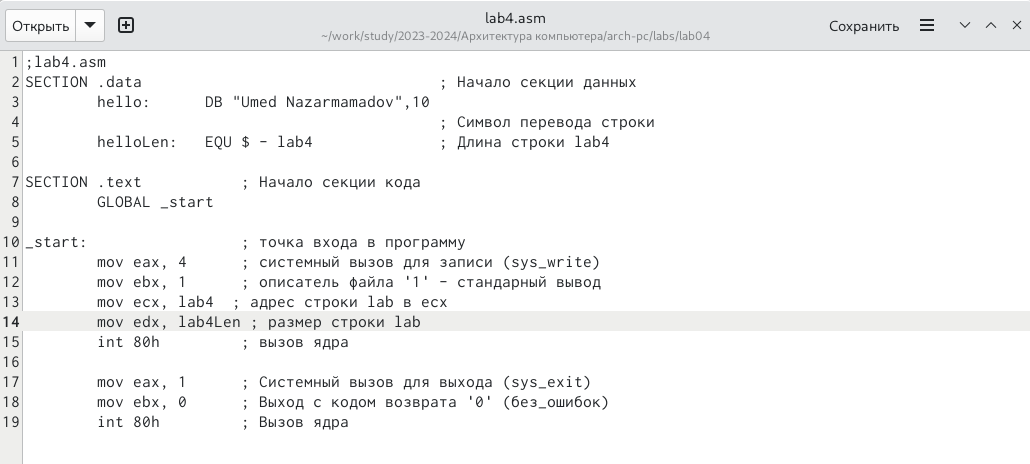


Рис. 11: Изменение программы

Компилирую текст программы в объектный файл.

Рис. 12: Компиляция текста

Рис. 12: Компиляция текста

Передаю объектный файл lab4.0 на обработку компоновщику LD, чтобы получить исполняемый файл lab4.

Рис. 13: Передача объектного файла на обработку компоновщику

Рис. 13: Передача объектного файла на обработку компоновщику

Запускаю исполняемый файл lab4.

Рис. 14: Запуск файла

Рис. 14: Запуск файла

Так как программа успешнj выполнила работу, теперь мне необходимо отправить файл на сервер с помощью команды git push.

# 4 Выводы

При выполнении лабораторной работы я освоил процедуры компиляции и сборки программ, которые были написани на языке ассемблере NASM.

# Список литературы

1. https://esystem.rudn.ru/mod/page/view.php?id=1030505