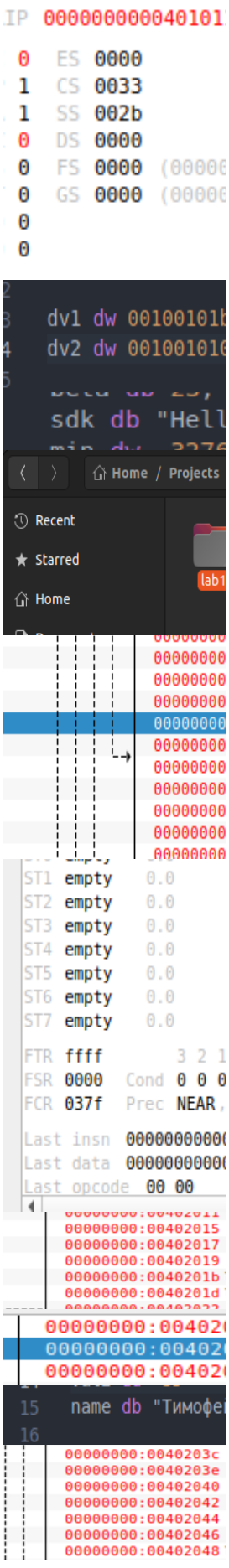


«Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ **ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ**

КАФЕДРА **КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)**

О Т Ч Е Т

по лабораторной работе № 1

Дисциплина: Машинно-зависимые языки и основы компиляции

Название: Изучение среды и отладчика ассемблера

Студент гр. ИУ6-41Б	15.03.2023	Т. Е. Старжевский
	(Подпись, дата)	(И.О. Фамилия)
Преподаватель	15.03.2023	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.

Ход работы

Задание 1.2.1.

Создать каталог для хранения файлов лабораторных работ, в котором создать подкаталог для первой работы (Рисунок 1).

Рисунок 1 — Подкаталог Lab1

Задание 1.2.2.

Объявить в терминале данный подкаталог текущим (Рисунок 2).

Рисунок 2 — Объявление подкаталога текущим

Задание 1.2.3.

Открыть текстовый редактор, ввести заготовку 64-х разрядной программы на ассемблере. Сохранить программу с именем lab1.asm в подкаталоге Labs/Lab1 (Рисунок 3).

Рисунок 3 — заготовка 64-х разрядной программы на ассемблере

Задание 1.2.4.

Выполнить трансляцию программы с листингом (Рисунок 4)

Рисунок 4 — трансляция программы с листингом

Задание 1.2.5.

Выполнить компоновку программы (Рисунок 5)

Рисунок 5 — компоновка программы

Задание 1.2.6.

Запустить программу. В консоли вывелось «Press Enter to Exit» (Рисунок 6).

Рисунок 6 — результат работы программы

Программа завершилась после нажатия клавиши «Enter».

Задание 1.2.7.

Запустить отладчик edb. В отладчике открыть программу lab1. Результат открытия показан на рисунке 7.

Рисунок 7 — Окно отладчика для программы lab1.

На рисунке можно увидеть дисассемблированный код и машинное представление команд (левый верхний сектор), содержимое регистров (правый верхний сектор), шестнадцатеричный дамб сегментов данных (левый нижний сектор), а также шестнадцатеричный дамб стека (правый нижний угол).

Задание 1.2.8.

Добавить в код логику выражения X = A + 5 - B и проследить изменение программы в отладчике. Результаты представлены на рисунке 8.

Рисунок 8 — Окно отладчика и программа для выражения X = A+5-B

Задание 1.2.9.

Найти в отладчике внутреннее представление данных и зафиксировать их изменение при пошаговой отладке. Результат каждого шага представлен на отдельных рисунках ниже.

Рисунок 9 — Полное окно отладчика перед началом пошагового обхода

Изначально значения регистров, которые мы используем для вычисления на нуле. Эти значения указаны в правой верхней стороне скриншота.

Далее происходит команда mov EAX, [0x402000] (Рисунок 10).

Рисунок 10 — Значение регистра RAX при первом шаге

Как и ожидалось значение регистра RAX заполняется содержимым адреса 0x402000, адрес A.

Далее: add EAX, 5. Код добавляет 5 к содержимому регистра RAX, что можно увидеть в отладчике (Рисунок 11).

Рисунок 11 — значение регистра RAX на втором шаге

Затем команда sub eax, [0x402002]. Из регистра RAX вычитается значение, хранящееся по адресу 0x402002 (адрес B) (Рисунок 12).

Рисунок 12 — Значение регистра RAX на третьем шаге

Последним шагом является команда mov [0x402022], eax. Она переместит содержимое регистра RAX по адресу 0x402022 (адрес X) (Рисунок 13).

Рисунок 13 — Значение регистра RAX на четвертом шаге

Задание 1.2.10.

Ввести строки, представленные на рисунке 14 в свою программу и определить их внутреннее представление с помощью отладчика.

Рисунок 14 — данные для определения отладчиком

Введя адрес первого сегмента инициализированных переменных в отладчик, можно получить внутреннее представление всех остальных переменных (Рисунок 15).

Рисунок 15 — внутренне представление констант

Неинициализированные данные, соответственно, заполнены ничем. Память под них зарезервирована, но пуста (Рис. 16)

Рисунок 16 — внутреннее представление переменных

Задание 1.2.11.

Определить в памяти следующие данные:

а) целое число 25 размером 2 байта со знаком;

б) двойное слово, содержащее число -35;

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

Рисунок 17 — определение данных для задания 1.2.11

Зафиксировать внутреннее представление этих данных.

Определение всех данных представлено на рисунке 17.

Внутреннее представление пунктов а и б представлено на рисунке 18

Рисунок 18 — внутреннее представление данных пунктов а и б

Целые числа преобразуются к отрицательным, путём инвертирования и добавления единицы.

Внутреннее представление пункта в представлено на рисунке 19

Рисунок 19 - внутреннее представление данных пункта в

Строка хранится, как последовательность символов. В памяти зарезервирован 1 байт, остальные символы идут непрерывно. Оканчивается строка байтом 00.

Задание 1.2.12.

Определите несколькими способами в программе числа, которые во внутреннем представлении (в отладчике) будут выглядеть как 25 00 и 00 25. Проверьте правильность ваших предположений, введя соответствующие строки в программу.

Определим их в десятичном, шестнадцатеричном, двоичном и представлениях (Рисунок 20)

Рисунок 20 — определение данных для задания 1.2.12

Правильность потдверждает отладчик (Рисунок 21)

Рисунок 21 — данные отладчика для задания 1.2.12.

Задание 1.2.13.

Добавьте в программу переменную F1=65535 размером слово и переменную F2= 65535 размером двойное слово. Вставьте в программу команды сложения этих чисел с 1

Определение данных в коде представлено на рисунке 22.

Рисунок 22 — определение данных для задания 1.2.13.

Во внутреннем представлении после прибавления 1 данные будут выглядеть как на рисунке 23.

Рисунок 23 — внутреннее представление данных для задания 1.2.13

Такой результат получился из-за того, что для F1 был выделен всего 1 байт, не вмещающий число 65536. При прибавлении единицы для нее нет места и возникает переполнение, поэтому остаются только нули. Для F2 такой ситуации не произошло, потому что для нее выделено 2 байта памяти.

Результат виден на флагах. C — признак переноса (1 - есть), P — четность (1 — четное), A — наличие переноса из тетрады (1 — имеется), Z

— признак нуля (1 — значит нуль), O - устанавливается, если целочисленный результат слишком длинный для размещения в целевом операнде.

Рисунок 24 — Значение флагов для F1

Рисунок 25 — Значение флагов для F2

Контрольные вопросы

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

Язык ассемблера — низкоуровневый язык программирования. Его команды транслируются напрямую в машинный код для дальнейшей компоновки. Относится к группе машинно-зависимых языков.

2. Из каких частей состоит заготовка программы на ассемблере?

Заготовка программы на языке ассемблера состоит из 3 частей:

1) .data (сегмент инициализированных данных)

2) .bss (сегмент неинициализированных данных)

3) .text (сегмент кода)

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

Вызываем транслятор nasm (Переводит в машинный код):

```
nasm -f elf64 lab1.asm -l lab1.lst
```

Создается объектный файл .o, далее компонуем (Собираем нужные библиотеки):

```
ld -o lab1 lab1.o
```

Создается исполняемая программа lab1.

Запускаем командой

```
./lab1
```

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

Содержимое в памяти можно посмотреть с помощью Goto Exrpession комбинацией клавиш **Ctrl+G**

– **F7** – выполнить шаг с заходом в тело процедуры;

– **F8** – выполнить шаг, не заходя в тело процедуры.

– **F9** - выполнить выход из тела процедуры.

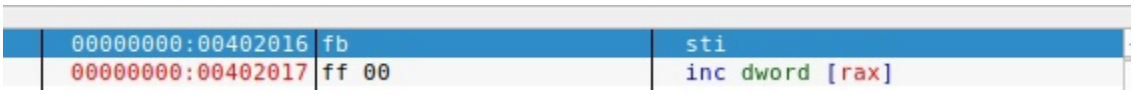
Пошаговое выполнение программы можно осуществить с помощью F7.

5. В каком виде отладчик показывает положительные и отрицательные целые числа? Как будут представлены в памяти числа: A dw 5,-5 ? Как те же числа будут выглядеть после загрузки в регистр AX?

5 в шестнадцатеричной СС будет равно 5, -5 равняется FFFB. Проверим в отладчике:

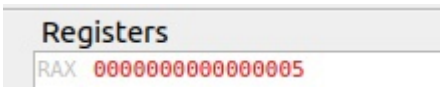
5:

-5:



Загрузим эти числа в регистр AX и проверим содержимое

5:



-5:

Вывод: Изучил процессы создания, запуска и отладки программ на ассемблере Nasm под управлением операционной системы Linux, а также особенности описания и внутреннего представления данных.