РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 6

дисциплина: Архитектура компьютера

Студент: Грицко Сергей

Группа: НКАбд-02-25

МОСКВА

2025 г.

Оглавление

Цель работы

Освоение арифметических инструкций языка ассемблера NASM.

Описание результатов выполнения лабораторной работы

Сначала я создал каталог lab06 и скопировал в этот каталог файл in_out.asm с прошлой лабы.

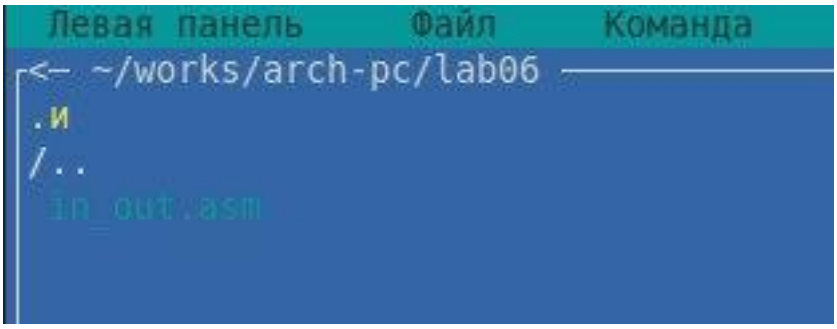

Рисунок 1. Создание каталога

Далее создал файл lab6-1.asm (Листинг 6.1 из методички) и скомпилировал его.

```
sgritsko@sgritsko:~/works/arch-pc/lab06$ nasm -f elf lab6-1.asm
sgritsko@sgritsko:~/works/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
sgritsko@sgritsko:~/works/arch-pc/lab06$ ./lab6-1
j
```

Рисунок 2. Выполнение первой версии lab6-1.asm, где виден вывод 'j'

Когда я запустил первую версию lab6-1.asm (которая складывала eax, '6' и ebx, '4'), программа вывела символ j. Это было ожидаемо, потому что на самом деле сложились не числа, а их ASCII-коды (54 + 52 = 106), а 106 — это как раз код символа j.

Потом я изменил lab6-1.asm, заменив '6' на 6 и '4' на 4. Снова скомпилировал и запустил.

```
sgritsko@sgritsko:~/works/arch-pc/lab06$ nasm -f elf lab6-1.asm
sgritsko@sgritsko:~/works/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
sgritsko@sgritsko:~/works/arch-pc/lab06$ ./lab6-1
```

Рисунок 3. Выполнение второй версии lab6-1.asm, где виден "пустой" вывод

В этот раз программа вывела "пустую строку" (просто перевела курсор). Я посмотрел в таблицу ASCII — код 10, который получился (6 + 4), это символ "Line Feed" или \n (перевод строки). Функция sprintLF выводит строку по адресу, а по адресу 10 в памяти ничего осмысленного нет, но так как в eax было 10, sprintLF (которая ожидает адрес) отработала некорректно, но call quit все равно завершил программу. В общем, sprintLF не предназначена для вывода чисел.

Далее я создал файл lab6-2.asm (Листинг 6.2), который использует iprintLF (функцию для вывода *чисел*).

```
sgritsko@sgritsko:~/works/arch-pc/lab06$ nasm -f elf lab6-2.asm
sgritsko@sgritsko:~/works/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
sgritsko@sgritsko:~/works/arch-pc/lab06$ ./lab6-2
106
```

Рисунок 4. lab6-2.asm с символами, вывод '106'

Сначала со значениями '6' и '4' программа вывела 106. Это логично, iprintLF взяла результат сложения ASCII-кодов (54 + 52) и вывела его уже как число.

Я изменил lab6-2.asm на использование чисел 6 и 4.

```
sgritsko@sgritsko:~/works/arch-pc/lab06$ nasm -f elf lab6-2.asm
sgritsko@sgritsko:~/works/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
sgritsko@sgritsko:~/works/arch-pc/lab06$ ./lab6-2
10
```

Рисунок 5. Выполнение lab6-2.asm с числами, вывод '10'

Наконец-то! Программа корректно сложила 6 и 4 и вывела число 10. Теперь понятно, что iprintLF — для чисел, а sprintLF — для строк (символов).

В конце я заменил iprintLF на iprint.

```
sgritsko@sgritsko:~/works/arch-pc/lab06$ nasm -f elf lab6-2.asm
sgritsko@sgritsko:~/works/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
sgritsko@sgritsko:~/works/arch-pc/lab06$ ./lab6-2
10sgritsko@sgritsko:~/works/arch-pc/lab06$
```

Рисунок 6. Выполнение lab6-2.asm с iprint, где вывод "слипся" со строкой терминала

Разница оказалась в том, что iprintLF (LF = Line Feed) добавляет перевод строки в конце, а iprint — нет. Моя командная строка просто "прилипла" к числу 10.

Я создал файл lab6-3.asm (Листинг 6.3). Он считал (5*2+3)/3.

```
sgritsko@sgritsko:~/works/arch-pc/lab06$ nasm -f elf lab6-3.asm
sgritsko@sgritsko:~/works/arch-pc/lab06$ ld -m elf_i386 -o lab6-3 lab6-3.o
sgritsko@sgritsko:~/works/arch-pc/lab06$ ./lab6-3
Результат: 4
Остаток от деления: 1
```

Рисунок 7. Выполнение lab6-3.asm с первым выражением

Программа успешно вывела Результат: 4 и Остаток от деления: 1. Это верно, т.к. (5*2+3)/3 = 13/3 = 4 и 1 в остатке. Код был понятный, особенно xor edx, edx перед делением для очистки edx.

Я изменил lab6-3.asm для вычисления f(x) = (4*6+2)/5.

```
sgritsko@sgritsko:~/works/arch-pc/lab06$ nasm -f elf lab6-3.asm
sgritsko@sgritsko:~/works/arch-pc/lab06$ ld -m elf_i386 -o lab6-3 lab6-3.o
sgritsko@sgritsko:~/works/arch-pc/lab06$ ./lab6-3
Результат: 5
Остаток от деления: 1
```

Рисунок 8. Выполнение lab6-3.asm со вторым выражением

Я просто заменил константы в коде: mov eax, 4, mov ebx, 6, add eax, 2, mov ebx, 5 и программа вывела Результат: 5 и Остаток от деления: 1. Это тоже верно: (4*6+2)/5 = 26/5 = 5 и 1 в остатке.

Теперь надо написать программу, которая запрашивает номер студенческого билета, считывает его, и вычисляет номер варианта по формуле (Номер mod 20) + 1.

Я создал файл variant.asm и скомпилировал и запустил.

```
sgritsko@sgritsko:~/works/arch-pc/lab06$ nasm -f elf variant.asm
sgritsko@sgritsko:~/works/arch-pc/lab06$ ld -m elf_i386 -o variant variant.o
sgritsko@sgritsko:~/works/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1032251953
Ваш вариант: 14
```

Рисунок 9. Выполнение variant.asm, ввод номера 1032251953 и вывод варианта 14

Я ввел свой (условный) номер 123453. Программа правильно использовала sread для чтения строки, atoi для превращения ее в число, а затем div ebx (где ebx = 20) для получения остатка. Остаток (13) сохранился в edx, потом inc edx (стало 14), и iprintLF вывела мой вариант.

Описание результатов выполнения заданий для самостоятельной работы

Описание задания: Нужно было написать программу для вычисления выражения $y = f(x)$ по номеру варианта. Мой вариант - 14.

- **Функция:** f(x) = (x/2 + 8) * 3
- **Контрольные значения:** x1 = 1, x2 = 4

Программа должна выводить формулу, запрашивать x, считать и выводить результат.

Проверка x1 = 1:

```
sgritsko@sgritsko:~/works/arch-pc/lab06$ nasm -f elf lab6-task.asm
sgritsko@sgritsko:~/works/arch-pc/lab06$ ld -m elf_i386 -o lab6-task lab6-task.o
sgritsko@sgritsko:~/works/arch-pc/lab06$ ./lab6-task
f(x) = (x/2 + 8) * 3

Vvedite x: 1
Resultat: 24
```

Рисунок 10. Выполнение lab6-task.asm с вводом x=1 и выводом 24

Я запустил программу и ввел 1. Программа вывела Resultat: 24. (1 / 2 + 8) * 3 = (0 + 8) * 3 = 8 * 3 = 24. (Примечание: 1/2 = 0 в целочисленном делении). **Вывод:** Совпало. Программа работает верно.

Проверка x2 = 4:

```
sgritsko@sgritsko:~/works/arch-pc/lab06$ ./lab6-task
f(x) = (x/2 + 8) * 3

Vvedite x: 4
Resultat: 30
```

Рисунок 11. Выполнение lab6-task.asm с вводом x=4 и выводом 30

Я запустил программу еще раз и ввел 4. Программа вывела Resultat: 30. (4 / 2 + 8) * 3 = (2 + 8) * 3 = 10 * 3 = 30. **Вывод:** Снова совпало.

Вывод

В ходе выполнения этой лабораторной работы я достиг поставленной цели — **освоил арифметические инструкции языка ассемблера NASM**.

Я на практике увидел разницу между символьными данными (ASCII-кодами) и настоящими числами, и понял, почему для работы с ними нужны разные функции (sprint vs iprint, atoi).

Я научился использовать базовые арифметические команды:

1. add (сложение)
2. inc (инкремент)
3. mul (умножение без знака)
4. div (деление без знака)

Особенно важным было понять, как работает деление: что eax — это делимое, ebx — делитель, результат (частное) кладется в eax, а остаток — в edx, и что edx обязательно нужно обнулять (xor edx, edx) перед делением.

Самостоятельное задание помогло закрепить навыки: я смог с нуля написать программу, которая считывает данные, проводит последовательность вычислений (div, add, mul) и выводит корректный результат.

Список литературы

1. GDB: The GNU Project Debugger. — URL: https://www.gnu.org/software/gdb/.
2. GNU Bash Manual. — 2016. — URL: https://www.gnu.org/software/bash/manual/.
3. Midnight Commander Development Center. — 2021. — URL: https://midnight-commander.org/.
4. NASM Assembly Language Tutorials. — 2021. — URL: https://asmtutor.com/.
5. Newham C. Learning the bash Shell: Unix Shell Programming. — O'Reilly Media, 2005. — 354 c. — (In a Nutshell). — ISBN 0596009658. — URL: http://www.amazon.com/Learning-bash-Shell-Programming-Nutshell/dp/0596009658.
6. Robbins A. Bash Pocket Reference. — O'Reilly Media, 2016. — 156 c. — ISBN 978-1491941591.
7. The NASM documentation. — 2021. — URL: https://www.nasm.us/docs.php.
8. Zarrelli G. Mastering Bash. — Packt Publishing, 2017. — 502 c. — ISBN 9781784396879.
9. Колдаев В. Д., Лупин С. А. Архитектура ЭВМ. — М. : Форум, 2018.
10. Куляс О. Л., Никитин К. А. Курс программирования на ASSEMBLER. — М. : Солон-Пресс, 2017.
11. Новожилов О. П. Архитектура ЭВМ и систем. — М. : Юрайт, 2016.
12. Расширенный ассемблер: NASM. — 2021. — URL: https://www.opennet.ru/docs/RUS/nasm/.
13. Робачевский А., Немнюгин С., Стесик О. Операционная система UNIX. — 2-е изд. — БХВ-Петербург, 2010. — 656 с. — ISBN 978-5-94157-538-1.
14. Столяров А. Программирование на языке ассемблера NASM для ОС Unix. — 2-е изд. — М. : МАКС Пресс, 2011. — URL: http://www.stolyarov.info/books/asm_unix.
15. Таненбаум Э. Архитектура компьютера. — 6-е изд. — СПб. : Питер, 2013. — 874 с. — (Классика Computer Science).
16. Таненбаум Э., Бос Х. Современные операционные системы. — 5-е изд. — СПб. : Питер, 2018. — 1120 с. — (Классика Computer Science).