

Отчет по лабораторной работе №6

Освоение арифметических инструкций языка ассемблера NASM

Карапетян Мари Рафаеловна

Содержание

| | | |
|----------|---------------------------------------|-----------|
| 1 | Цель работы | 5 |
| 2 | Задание | 6 |
| 3 | Теоретическое введение | 7 |
| 4 | Выполнение лабораторной работы | 8 |
| 5 | Ответы на вопросы | 13 |
| 6 | Выводы | 14 |
| | Список литературы | 15 |

Список иллюстраций

| | | |
|------|---|----|
| 4.1 | Создание каталога и файла лаб 6-1.asm | 8 |
| 4.2 | Программа 1 | 8 |
| 4.3 | Запуск программы | 9 |
| 4.4 | Создание файла lab6-2.asm | 9 |
| 4.5 | Введение текста | 9 |
| 4.6 | Запуск файла | 9 |
| 4.7 | Замена символов на числа | 10 |
| 4.8 | Запуск файла | 10 |
| 4.9 | Замена функции | 10 |
| 4.10 | Запуск файла | 11 |
| 4.11 | Создание файла lab6-3.asm | 11 |
| 4.12 | Введение текста | 11 |
| 4.13 | Запуск файла | 12 |
| 4.14 | Введение текста | 12 |
| 4.15 | Запуск файла | 12 |

Список таблиц

1 Цель работы

Целью работы является освоение арифметических инструкций языка ассемблера NASM.

2 Задание

1. Написать программу вычисления выражения $y = f(x)$. Вид функции $f(x)$ выбрать из таблицы 6.3 вариантов заданий в соответствии с номером полученным при выполнении лабораторной работы. 2. Загрузите файлы на GitHub.

3 Теоретическое введение

Большинство инструкций на языке ассемблера требуют обработки операндов. Адрес операнда предоставляет место, где хранятся данные, подлежащие обработке. Это могут быть данные хранящиеся в регистре или в ячейке памяти. Далее рассмотрены все существующие способы задания адреса хранения операндов – способы адресации. Существует три основных способа адресации:

- Регистровая адресация – операнды хранятся в регистрах и в команде используются имена этих регистров, например: `mov ax,bx`.
- Непосредственная адресация – значение операнда задается непосредственно в команде, Например: `mov ax,2`.
- Адресация памяти – операнд задает адрес в памяти. В команде указывается символическое обозначение ячейки памяти, над содержимым которой требуется выполнить операцию. Например, определим переменную `intg DD 3` – это означает, что задается область памяти размером 4 байта, адрес которой обозначен меткой `intg`. В таком случае, команда `mov eax,[intg]` копирует из памяти по адресу `intg` данные в регистр `eax`. В свою очередь команда `mov [intg],eax` запишет в память по адресу `intg` данные из регистра `eax`. Также рассмотрим команду `mov eax,intg`. В этом случае в регистр `eax` запишется адрес `intg`. Допустим, для `intg` выделена память начиная с ячейки с адресом `0x600144`, тогда команда `mov eax,intg` аналогична команде `mov eax,0x600144` – т.е. эта команда запишет в регистр `eax` число `0x600144`.

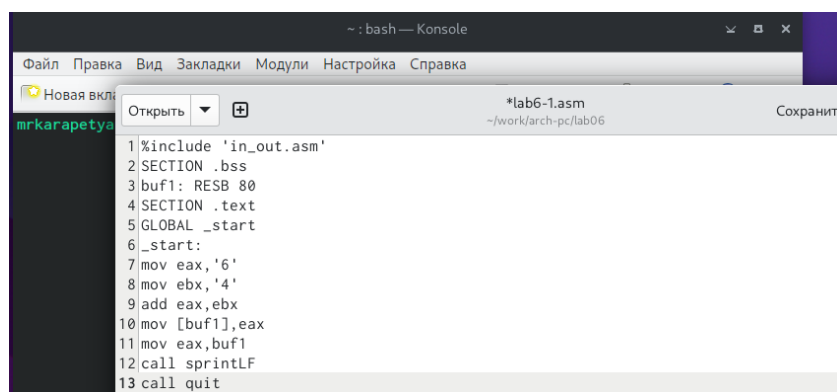
4 Выполнение лабораторной работы

Создайте каталог для программ лабораторной работы No 6, перейдите в него и создайте файл lab6-1.asm (рис. 4.1).

```
mrkarapetyan@dk6n55 ~ $ mkdir ~/work/arch-pc/lab06
mrkarapetyan@dk6n55 ~ $ cd ~/work/arch-pc/lab06
mrkarapetyan@dk6n55 ~/work/arch-pc/lab06 $ touch lab6-1.asm
mrkarapetyan@dk6n55 ~/work/arch-pc/lab06 $
```

Рис. 4.1: Создание каталога и файла лаб 6-1.asm

Рассмотрим примеры программ вывода символьных и численных значений. Программы будут выводить значения записанные в регистры еах. (рис. 4.2).



```
1 %include 'in_out.asm'
2 SECTION .bss
3 buf1: RESB 80
4 SECTION .text
5 GLOBAL _start
6 _start:
7 mov eax, '6'
8 mov ebx, '4'
9 add eax, ebx
10 mov [buf1], eax
11 mov eax, buf1
12 call printf
13 call quit
```

Рис. 4.2: Программа 1

Создайте исполняемый файл и запустите его. (рис. 4.3).


```
mrkarapetyan@dk4n70 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
mrkarapetyan@dk4n70 ~/work/arch-pc/lab06 $ d -m elf_i386 -o lab6-1 lab6-1.o
bash: d: команда не найдена
mrkarapetyan@dk4n70 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
mrkarapetyan@dk4n70 ~/work/arch-pc/lab06 $ ./lab6-1
j
mrkarapetyan@dk4n70 ~/work/arch-pc/lab06 $
```

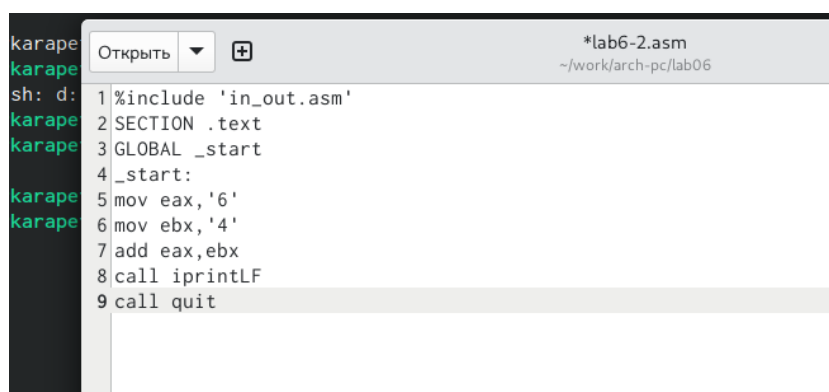
Рис. 4.3: Запуск программы

Создайте файл lab6-2.asm в каталоге ~/work/arch-pc/lab06 (рис. 4.4).

```
mrkarapetyan@dk4n70 ~/work/arch-pc/lab06 $ touch ~/work/arch-pc/lab06/lab6-2.asm
mrkarapetyan@dk4n70 ~/work/arch-pc/lab06 $
```

Рис. 4.4: Создание файла lab6-2.asm

Введите в него текст программы из листинга 6.2. (рис. 4.14).



```
karape sh: d: 1 %include 'in_out.asm'
karape 2 SECTION .text
karape 3 GLOBAL _start
karape 4 _start:
karape 5 mov eax, '6'
karape 6 mov ebx, '4'
karape 7 add eax, ebx
karape 8 call iprintLF
karape 9 call quit
```

Рис. 4.5: Введение текста

Создайте исполняемый файл и запустите его. (рис. 4.6).

```
mrkarapetyan@dk4n70 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
mrkarapetyan@dk4n70 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
mrkarapetyan@dk4n70 ~/work/arch-pc/lab06 $ ./lab6-2
106
mrkarapetyan@dk4n70 ~/work/arch-pc/lab06 $
```

Рис. 4.6: Запуск файла

Аналогично предыдущему примеру изменим символы на числа. Замените строки (рис. 4.7).

```
/afs/.dk.sci.pfu.edu.ru/home/m/r/mrkarapetyan/work/arch-pc/lab06/lab6-2.asm
%include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,6
mov ebx,4
add eax,ebx
call iprintLF
call quit
```

Рис. 4.7: Замена символов на числа

Создайте исполняемый файл и запустите его. (рис. 4.8).

```
mrkarapetyan@dk8n60 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
mrkarapetyan@dk8n60 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
mrkarapetyan@dk8n60 ~/work/arch-pc/lab06 $ ./lab6-2
10
mrkarapetyan@dk8n60 ~/work/arch-pc/lab06 $
```

Рис. 4.8: Запуск файла

В результате получили число 10.

Замените функцию `iprintLF` на `iprint` (рис. 4.9).

```
/afs/.dk.sci.pfu.edu.ru/home/m/r/mrkarapetyan/work/arch-pc/lab06/lab6-2.asm Изменё
%include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,6
mov ebx,4
add eax,ebx
call iprint
call quit
```

Рис. 4.9: Замена функции

Создайте исполняемый файл и запустите его. (рис. 4.10).

```
mrkarapetyan@dk8n60 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
mrkarapetyan@dk8n60 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
mrkarapetyan@dk8n60 ~/work/arch-pc/lab06 $ ./lab6-2
10mrkarapetyan@dk8n60 ~/work/arch-pc/lab06 $
```

Рис. 4.10: Запуск файла

В качестве примера выполнения арифметических операций в NASM приведем программу вычисления арифметического выражения $(5 \times 2 + 3) / 3$

Создайте файл lab6-3.asm в каталоге ~/work/arch-pc/lab06 (рис. 4.11).

```
mrkarapetyan@dk8n60 ~/work/arch-pc/lab06 $ touch lab6-3.asm
mrkarapetyan@dk8n60 ~/work/arch-pc/lab06 $
```

Рис. 4.11: Создание файла lab6-3.asm

Внимательно изучите текст программы из листинга 6.3 и введите в lab6-3.asm. (рис. 4.12).

```
/afs/.dk.sci.pfu.edu.ru/home/m/r/mrkarapetyan/work/arch-pc/lab06/lab6-3.asm
#include 'in_out.asm' ; подключение внешнего файла
SECTION .data
div: DB 'Результат: ',0
rem: DB 'Остаток от деления: ',0
SECTION .text
GLOBAL _start
_start:
; ---- Вычисление выражения
mov eax,4
mov ebx,6
mul ebx
add eax,2
xor edx,edx
mov ebx,5
div ebx
mov edi,eax
mov eax,div
call sprint
mov eax,edi
call iprintLF
mov eax,rem
call sprint
mov eax,edx
call iprintLF
call quit
```

Рис. 4.12: Введение текста

Создайте исполняемый файл и запустите его. (рис. 4.13).

```
mrkarapetyan@dk8n60 ~/work/arch-pc/lab06 $ nasm -f elf lab6-3.asm
mrkarapetyan@dk8n60 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
mrkarapetyan@dk8n60 ~/work/arch-pc/lab06 $ ./lab6-3
Результат: 4
Остаток от деления: 1
mrkarapetyan@dk8n60 ~/work/arch-pc/lab06 $
```

Рис. 4.13: Запуск файла

Введите в него текст программы variant.asm (рис. 4.14).

```
/afs/.dk.sci.pfu.edu.ru/home/m/r/mrkarapetyan/work/arch-pc/lab06/variant.asm
#include 'in_out.asm'
SECTION .data
msg: DB 'Введите No студенческого билета: ',0
rem: DB 'Ваш вариант: ',0
SECTION .bss
x: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax, msg
call sprintf
mov ecx, x
mov edx, 80
call sread
mov eax, x ; вызов подпрограммы преобразования
call atoi ; ASCII кода в число, 'eax=x'
xor edx, edx
mov ebx, 20
div ebx
inc edx
mov eax, rem
call sprintf
mov eax, edx
call iprintLF
call quit
```

Рис. 4.14: Введение текста

Создайте исполняемый файл и проверьте его работу. (рис. 4.15).

```
mrkarapetyan@dk8n60 ~/work/arch-pc/lab06 $ nasm -f elf lab6-3.asm
mrkarapetyan@dk8n60 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
mrkarapetyan@dk8n60 ~/work/arch-pc/lab06 $ ./lab6-3
Результат: 5
Остаток от деления: 1
mrkarapetyan@dk8n60 ~/work/arch-pc/lab06 $
```

Рис. 4.15: Запуск файла

5 Ответы на вопросы

1 `“mov eax, rem”` и `“call sprint”` в секции кода отвечают за вывод сообщения “Ваш вариант:” на экран. 2 `“mov ecx, x”` и `“mov edx, 80”` загружают адрес буфера (x) и длину буфера (80) соответственно в регистры ecx и edx для вызова подпрограммы `sread`, которая считывает строку из консоли. 3 `“call atoi”` вызывает подпрограмму `atoi` для преобразования ASCII кодов символов в число, результат которого сохраняется в регистре eax. 4 Код для вычисления варианта начинается с `“xor edx, edx”` и `“mov ebx, 20”`, после чего происходит деление числа, сохраненного в eax, на 20 с помощью инструкции `“div ebx”`. Результат деления (цифра варианта) записывается в нижнюю часть регистра AX (AL). Затем происходит увеличение цифры варианта на единицу с помощью инструкции `“inc edx”`. 5 Результат деления (цифра варианта) записывается в нижнюю часть регистра AX (AL). 6 Инструкция `“inc edx”` увеличивает цифру варианта на единицу для того, чтобы результат деления не оказывался равным нулю. 7 `“mov eax, edx”` загружает цифру варианта из AL (нижней части регистра AX) в регистр eax для вывода результата на экран с помощью подпрограммы `iprintLF`.

6 Выводы

В ходе выполнения работы, я освоил работу с арифметическими операциями на языке assembly.

Список литературы

GDB: The GNU Project Debugger. — URL: <https://www.gnu.org/software/gdb/>. GNU Bash Manual. — 2016. — URL: <https://www.gnu.org/software/bash/manual/>. Midnight Commander Development Center. — 2021. — URL: <https://midnight-commander.org/>. NASM Assembly Language Tutorials. — 2021. — URL: <https://asmtutor.com/>. Newham C. Learning the bash Shell: Unix Shell Programming. — O'Reilly Media, 2005. — 354 с. — (In a Nutshell). — ISBN 0596009658. — URL: <http://www.amazon.com/Learning-bash-Shell-Programming-Nutshell/dp/0596009658>. Robbins A. Bash Pocket Reference. — O'Reilly Media, 2016. — 156 с. — ISBN 978-1491941591. The NASM documentation. — 2021. — URL: <https://www.nasm.us/docs.php>. Zarrelli G. Mastering Bash. — Packt Publishing, 2017. — 502 с. — ISBN 9781784396879. Колдаев В. Д., Лупин С. А. Архитектура ЭВМ. — М. : Форум, 2018. Куляс О. Л., Никитин К. А. Курс программирования на ASSEMBLER. — М. : Солон-Пресс, 2017. Новожилов О. П. Архитектура ЭВМ и систем. — М. : Юрайт, 2016. Расширенный ассемблер: NASM. — 2021. — URL: <https://www.opennet.ru/docs/RUS/nasm/>. Робачевский А., Немнюгин С., Стесик О. Операционная система UNIX. — 2-е изд. — БХВ- Петербург, 2010. — 656 с. — ISBN 978-5-94157-538-1. 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. Таненбаум Э. Архитектура компьютера. — 6-е изд. — СПб. : Питер, 2013. — 874 с. — (Классика Computer Science). Таненбаум Э., Бос Х. Современные операционные системы. — 4-е изд. — СПб. : Питер, 2015 — 1120 с. — (Классика Computer Science). :: {#refs} ::