# Отчёт по лабораторной работе №6

### Дисциплина: архитектура компьютеров и операционные системы

Хань Цзянтао

# Содержание

1. [Цель работы](#_bookmark0) 5
2. [Задание](#_bookmark1) 6
3. [Теоретическое введение](#_bookmark2) 7
4. [Выполнение лабораторной работы](#_bookmark4) 9
   1. [Символьные и численные данные в NASM](#_bookmark5) . . . . . . . . . . . 9
   2. [Выполнение арифметических операций в NASM](#_bookmark18) 13
      1. [Ответы на вопросы по листингу 6.4](#_bookmark25) 16
   3. [Задание для самостоятельной работы](#_bookmark26) 17
5. [Выводы](#_bookmark29) 20
6. [Список литературы](#_bookmark30) 21

**Список иллюстраций**

* 1. [Создание каталога и файла](#_bookmark6) . . . . . . . . . . . . . . . . . . . . . 9
  2. [Ввод текста программы](#_bookmark7) . . . . . . . . . . . . . . . . . . . . . . . 9
  3. [Запуск программы](#_bookmark8) 10
  4. [Замена некоторых строк кода](#_bookmark9) 10
  5. [Запуск кода](#_bookmark10) 11
  6. [Создание файла](#_bookmark11) 11
  7. [Ввод текста программы](#_bookmark12) 11
  8. [Запуск исполняемого файла](#_bookmark13) 12
  9. [Изменение кода](#_bookmark14) 12
  10. [Запуск исполняемого файла](#_bookmark15) 12
  11. [Изменение кода](#_bookmark16) 13
  12. [Запуск исполняемого файла](#_bookmark17) 13
  13. [Создание файла](#_bookmark19) 14
  14. [Запуск исполняемого файла](#_bookmark20) 14
  15. [Изменение текста программы](#_bookmark21) 15
  16. [Запуск исполняемого файла](#_bookmark22) 15
  17. [Создание файла](#_bookmark23) 15
  18. [Результат работы кода](#_bookmark24) 16
  19. [Создание программы](#_bookmark27) 17
  20. [Результат работы кода](#_bookmark28) 19

# Список таблиц

* 1. [Описание некоторых каталогов файловой системы GNU Linux](#_bookmark3) . . 8

# Цель работы

Освоение арифметических инструкций языка ассемблера NASM.

# Задание

1. Символьные и численные данные в NASM.
2. Выполнение арифметических операций в NASM.
3. Ответы на вопросы по листингу 6.4
4. Задание для самостоятельной работы.

# Теоретическое введение

Большинство инструкций на языке ассемблера требуют обработки операн- дов. Адрес операнда предоставляет место, где хранятся данные, подлежащие обработке. Это могут быть данные, хранящиеся в регистре или в ячейке памяти.

Существует три основных способа адресации:

* Регистровая адресация – операнды хранятся в регистрах и в команде исполь- зуются 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.

Схема команды целочисленного сложения add (от англ. addition - добавление) выполняет сложение двух операндов и записывает результат по адресу перво- го операнда. Допустимые сочетания операндов для команды add аналогичны сочетаниям операндов для команды mov. Так, например, команда add eax,ebx прибавит значение из регистра eax к значению из регистра ebx и запишет ре- зультат в регистр eax. Довольно часто при написании программ встречается операция прибавления или вычитания единицы. Прибавление единицы называ- ется инкрементом, а вычитание — декрементом. Для этих операций существуют специальные команды: inc (от англ. increment) и dec (от англ. decrement), кото- рые увеличивают и уменьшают на 1 свой операнд. Команда neg рассматривает свой операнд как число со знаком и меняет знак операнда на противоположный.

Операндом может быть регистр или ячейка памяти любого размера. Для деления, как и для умножения, существует 2 команды div (от англ. divide - деление) и idiv. Например, в табл. [3.1](#_bookmark3) приведено краткое описание стандартных каталогов

Unix.

# Выполнение лабораторной работы

## Символьные и численные данные в NASM

Создаю каталог для программ лабораторной работы № 6, перехожу в него и создаю файл lab6-1.asm. (рис. [4.20).](#_bookmark28)

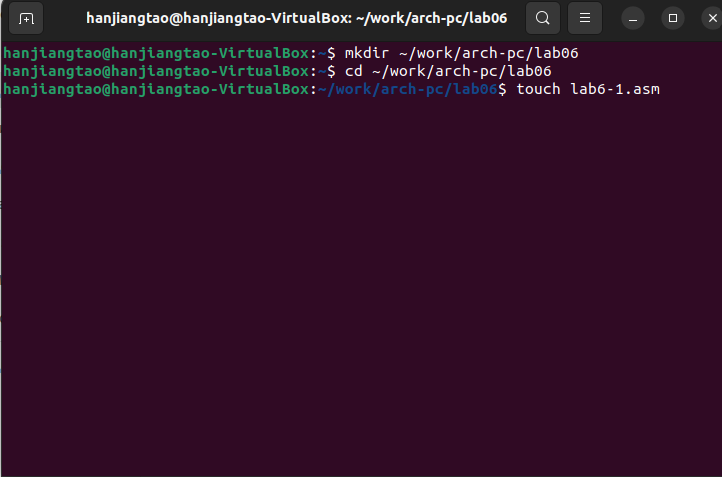


Рис. 4.1: Создание каталога и файла

Ввожу в файл lab6-1.asm текст программы из листинга 6.1. (рис. [4.20).](#_bookmark28)

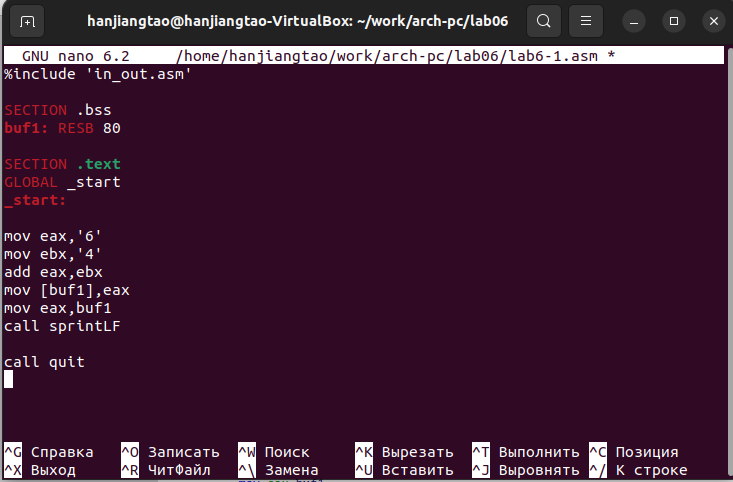


Рис. 4.2: Ввод текста программы

Создаю исполняемый файл и запускаю его. (рис. [4.20).](#_bookmark28)

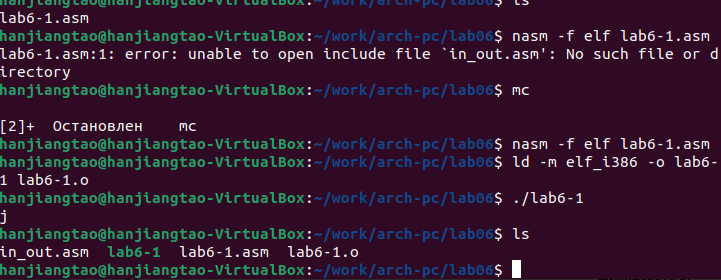


Рис. 4.3: Запуск программы

Далее изменю текст программы и вместо символов запишу в регистры числа.

Исправляю текст программы следующим образом: заменяю строки

mov eax,‘6’ mov ebx,‘4’ на строки mov eax,6 mov ebx,4 (рис. [4.20).](#_bookmark28)

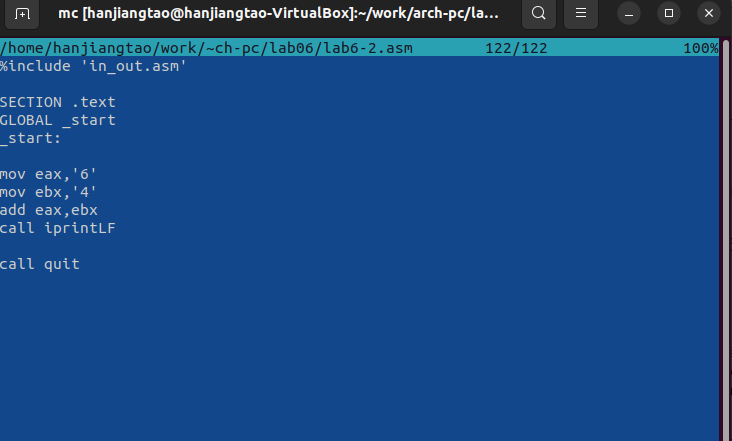


Рис. 4.4: Замена некоторых строк кода

Создаю исполняемый файл и запускаю его. (рис. [4.20).](#_bookmark28)

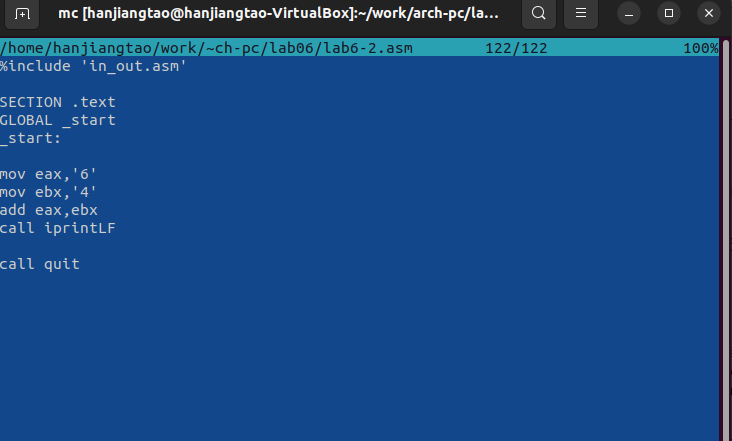


Рис. 4.5: Запуск кода

#### Данному коду (10) соответствует символ “LF, n”, который перемещает курсор на следующую строку. Сам символ при выводе на экран не отобра- жается.

Создаю файл lab6-2.asm в каталоге ~/work/arch-pc/lab06 (рис. [4.20).](#_bookmark28)

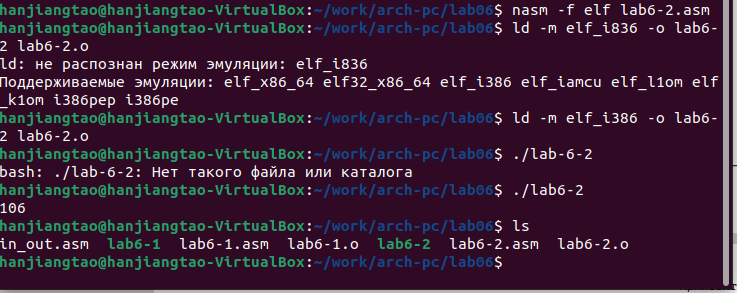


Рис. 4.6: Создание файла

и ввожу в него текст программы из листинга 6.2. (рис. [4.20).](#_bookmark28)

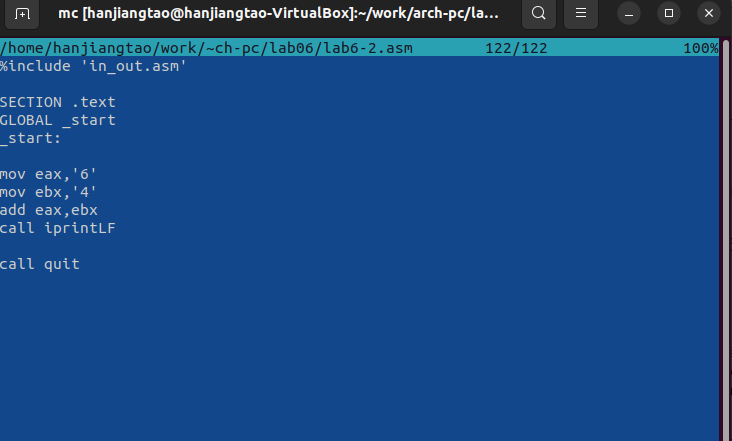


Рис. 4.7: Ввод текста программы

Создаю исполняемый файл и запускаю его. (рис. [4.20).](#_bookmark28)

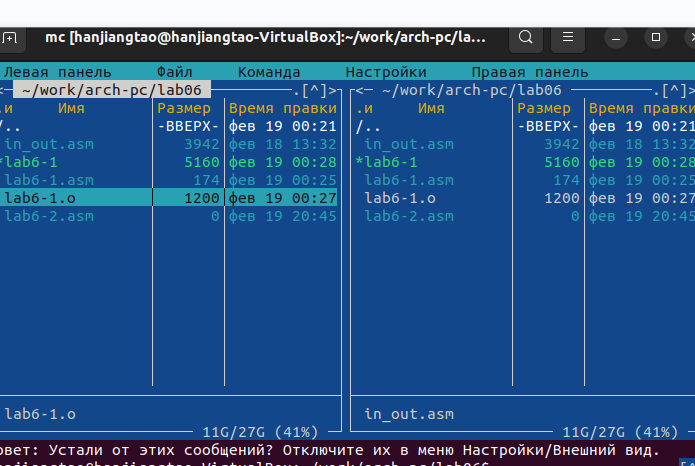


Рис. 4.8: Запуск исполняемого файла

В этой программе заменяю строки mov eax,‘6’

mov ebx,‘4’ на строки mov eax,6 mov ebx,4 (рис. [4.20).](#_bookmark28)

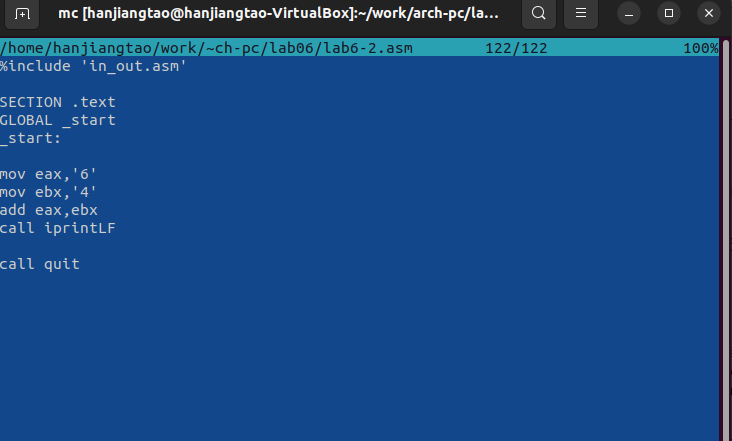


Рис. 4.9: Изменение кода

Создаю исполняемый файл и запускаю его. (рис. [4.20).](#_bookmark28)

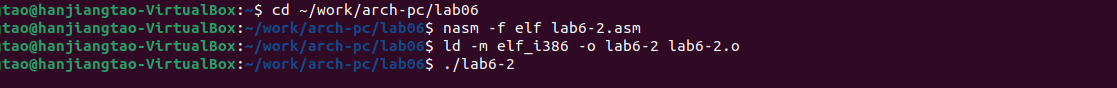


Рис. 4.10: Запуск исполняемого файла

#### В результате получаем число 10.

Заменяю функцию iprintLF на iprint. (рис. [4.20).](#_bookmark28)

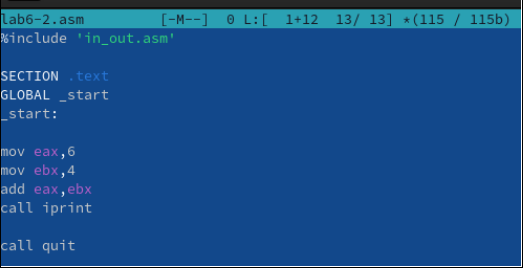


Рис. 4.11: Изменение кода

Создаю исполняемый файл и запускаю его. (рис. [4.20).](#_bookmark28)

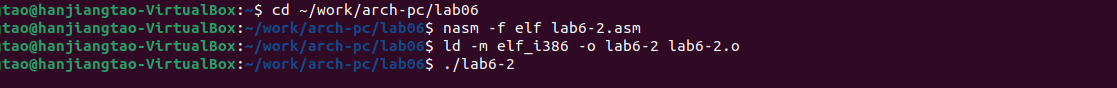


Рис. 4.12: Запуск исполняемого файла

**Вывод функций iprintLF и iprint отличается тем, что при использовании первой выполняется перенос на следующую строку после вывода, а при использовании второй этого не происходит.**

## Выполнение арифметических операций в NASM

Создаю файл lab6-3.asm в каталоге ~/work/arch-pc/lab06 и ввожу в него текст из листинга 6.3. (рис. [4.20).](#_bookmark28)

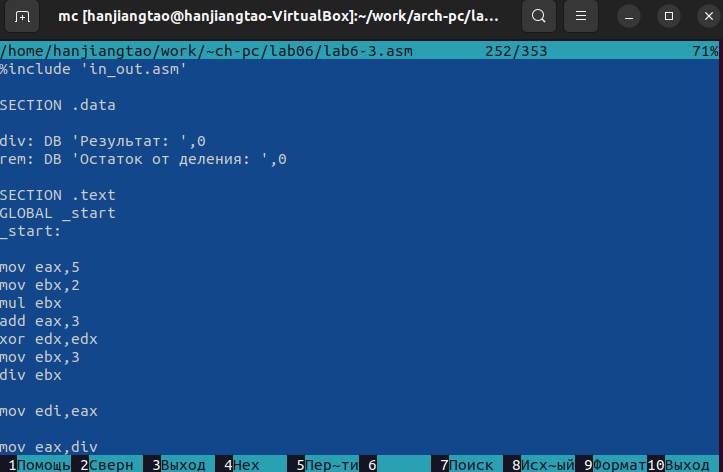


Рис. 4.13: Создание файла

Создаю исполняемый файл и запускаю его. (рис. [4.20).](#_bookmark28)

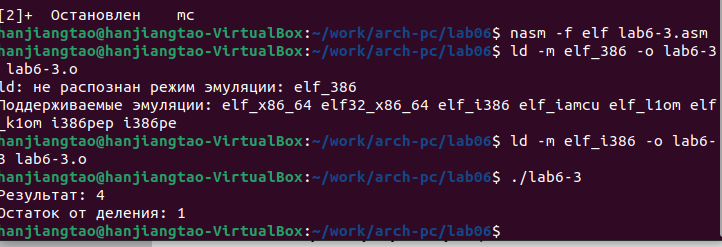
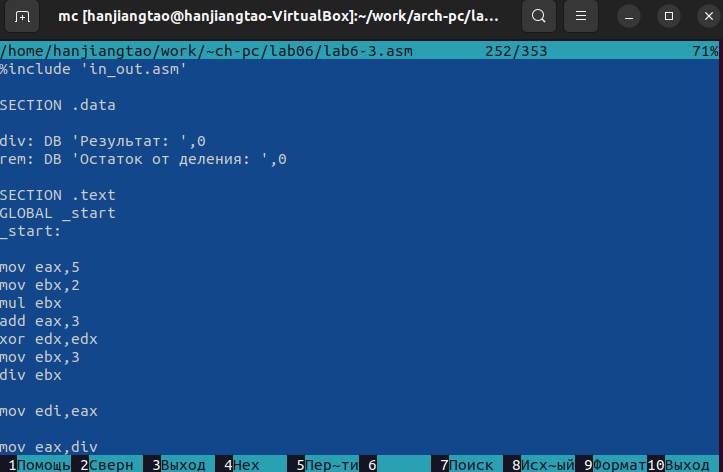


Рис. 4.14: Запуск исполняемого файла

Изменяю текст программы для вычисления выражения f(x) = (4 \* 6 + 2)/5, делая замену чисел в регистрах. (рис. [4.20).](#_bookmark28)

Рис. 4.15: Изменение текста программы Создаю исполняемый файл и проверяю его работу. (рис. [4.20).](#_bookmark28)

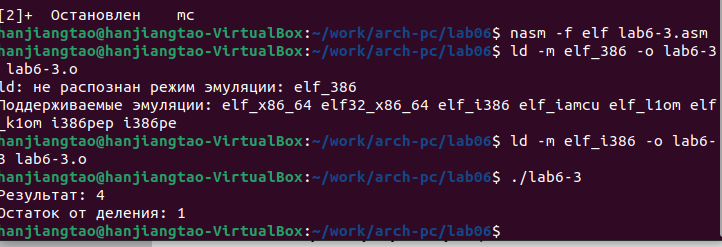


Рис. 4.16: Запуск исполняемого файла

Создаю файл variant.asm в каталоге ~/work/arch-pc/lab06 (рис. [4.20).](#_bookmark28)

17

Рис. 4.17: Создание файла

Текст программы из листинга 6.4 ввожу в файл variant.asm, создаю исполня- емый файл и запускаю его. Проверяю результат работы программы, вычислив номер варианта аналитически (ответ верный). (рис. [4.20).](#_bookmark28)

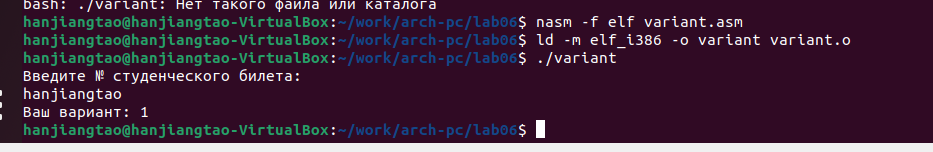


Рис. 4.18: Результат работы кода

### Ответы на вопросы по листингу 6.4

* + - 1. За вывод сообщения “Ваш вариант” отвечают строки кода: mov eax,rem

call sprint

* + - 1. mov ecx, x - Используется, чтобы положить адрес вводимой строки x в ре- гистр.

mov edx, 80 - Используется для записи в регистр edx длины вводимой строки. call sread - Используется для вызова подпрограммы из внешнего файла, обес-

печивающей ввод сообщения с клавиатуры.

* + - 1. “call atoi” используется для вызова подпрограммы из внешнего файла, ко- торая преобразует ascii-код символа в целое число и записывает результат в регистр eax.
      2. За вычисления варианта отвечают строки:

xor edx,edx mov ebx,20 div ebx

inc edx

* + - 1. При выполнении инструкции div ebx остаток от деления записывается в регистр edx.
      2. Инструкция “inc edx” увеличивает значение регистра edx на 1.
      3. За вывод на экран результатов вычислений отвечают строки: mov eax,edx

call iprintLF

## Задание для самостоятельной работы

Вывод программы variant.asm показал, что мой номер варианта - 10, поэтому мне нужно написать программу (rabota.asm) для вычисления выражения 5(x + 18) − 28 и проверить ее работу для значений х1 = 2 и х2 = 3. (рис. [4.20).](#_bookmark28)

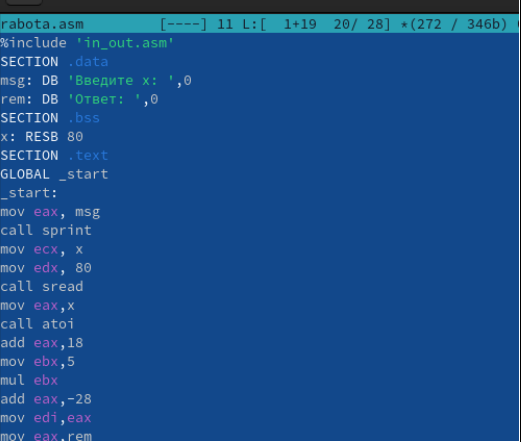


Рис. 4.19: Создание программы

Код программы:

%include ‘in\_out.asm’

SECTION .data

msg: DB ‘Введите х:’,0 rem: DB ‘Ответ:’,0 SECTION .bss

x: RESB 80

SECTION .text GLOBAL \_start

\_start:

mov eax, msg call sprint mov ecx, x mov edx, 80 call sread mov eax,x call atoi

add eax,18 mov ebx,5 mul ebx add eax,-28 mov edi,eax

mov eax,rem call sprint mov eax,edi call iprintLF call quit

Создаю исполняемый файл и проверяю его работу. (рис. [4.20).](#_bookmark28)

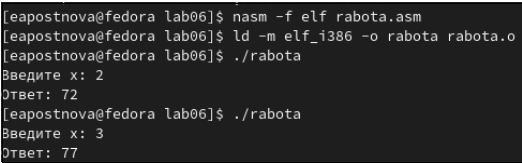


Рис. 4.20: Результат работы кода

# Выводы

С помощью данной лабораторной работы я освоила арифметические инструк- ции языка ассемблер NASM, что пригодится мне при выполнении последующих лабораторных работ.

# Список литературы

1. GDB: The GNU Project Debugger. — URL: ht[tps://w](http://www.gnu.org/software/gdb/)ww[.gnu.org/software/gdb/.](http://www.gnu.org/software/gdb/)
2. GNU Bash Manual. — 2016. — URL: ht[tps://w](http://www.gnu.org/software/bash/manual/)ww[.gnu.org/software/bash/manual/.](http://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 с. — (In a Nutshell). — ISBN 0596009658. — URL:

[http://www.amazon.com/Learningbash-Shell-Programming-Nutshell/dp/0596009658.](http://www.amazon.com/Learningbash-Shell-Programming-Nutshell/dp/0596009658)

1. Robbins A. Bash Pocket Reference. — O’Reilly Media, 2016. — 156 с. — ISBN 978-1491941591.
2. The NASM documentation. — 2021. — URL: ht[tps://w](http://www.nasm.us/docs.php)ww[.nasm.us/docs.php.](http://www.nasm.us/docs.php)
3. Zarrelli G. Mastering Bash. — Packt Publishing, 2017. — 502 с. — ISBN 9781784396879.
4. Колдаев В. Д., Лупин С. А. Архитектура ЭВМ. — М. : Форум, 2018.
5. Куляс О. Л., Никитин К. А. Курс программирования на ASSEMBLER. — М. : Солон-Пресс, 2017.
6. Новожилов О. П. Архитектура ЭВМ и систем. — М. : Юрайт, 2016.
7. Расширенный ассемблер: NASM.— 2021.— URL: [https://w](http://www.opennet.ru/docs/RUS/nasm/)ww[.opennet.ru/docs/RUS/nasm/.](http://www.opennet.ru/docs/RUS/nasm/)
8. Робачевский А., Немнюгин С., Стесик О. Операционная система UNIX. — 2-е изд. — БХВПетербург, 2010. — 656 с. — ISBN 978-5-94157-538-1.
9. 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.](http://www.stolyarov.info/books/asm_unix)

1. Таненбаум Э. Архитектура компьютера. — 6-е изд. — СПб. : Питер, 2013. — 874 с. — (Классика Computer Science).
2. Таненбаум Э., Бос Х. Современные операционные системы. — 4-е изд. — СПб. : Питер,2015. — 1120 с. — (Классика Computer Science).
3. GNU Bash Manual [Электронный ресурс]. Free Software Foundation, 2016. URL: [https://www.gnu.org/software/bash/manual/.](https://www.gnu.org/software/bash/manual/)
4. Newham C. [Learning the bash Shell: Unix Shell Programming.](http://www.amazon.com/Learning-bash-Shell-Programming-Nutshell/dp/0596009658) O’Reilly Media, 2005. 354 с.
5. Zarrelli G. Mastering Bash. Packt Publishing, 2017. 502 с.
6. Robbins A. [Bash Pocket Referenc](https://www.ncbi.nlm.nih.gov/pubmed/25246403)e. O’Reilly Media, 2016. 156 с.
7. Таненбаум Э. Архитектура компьютера. 6-е изд. СПб.: Питер, 2013. 874 с.
8. Таненбаум Э., Бос Х. Современные операционные системы. 4-е изд. СПб.: Питер, 2015. 1120 с.