Отчет по лабораторной работе №9

Архитектура компьютера

Дмитрий Константинович Кобзев

Содержание

# 1 Цель работы

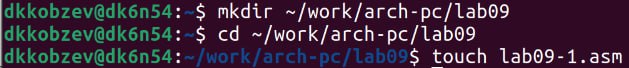
Приобретение навыков написания программ с использованием подпрограмм. Знакомство с методами отладки при помощи GDB и его основными возможностями.

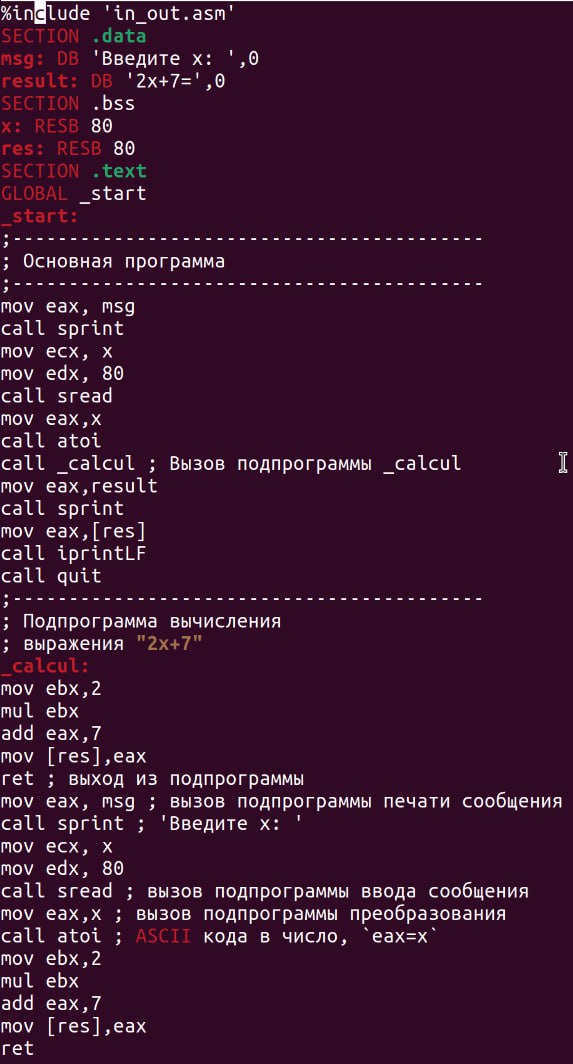
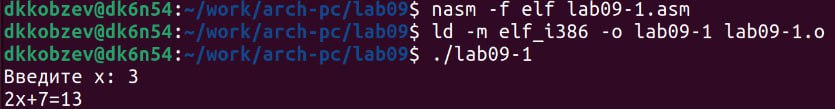
# 2 Задание

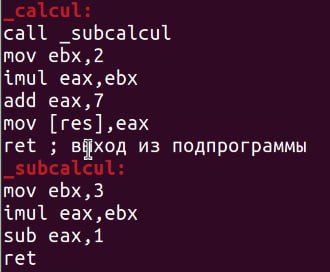
1. Преобразуйте программу из лабораторной работы №8 (Задание №1 для самостоятельной работы), реализовав вычисление значения функции f(x) как подпрограмму.
2. В листинге 9.3 приведена программа вычисления выражения (3 + 2) \* 4 + 5. При запуске данная программа дает неверный результат. Проверьте это. С помощью отладчика GDB, анализируя изменения значений регистров, определите ошибку и исправьте ее.

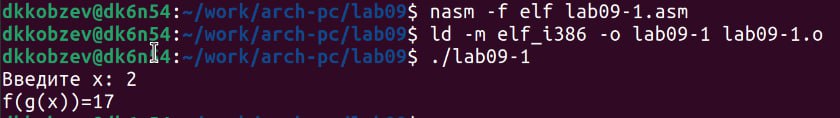
# 3 Выполнение лабораторной работы |

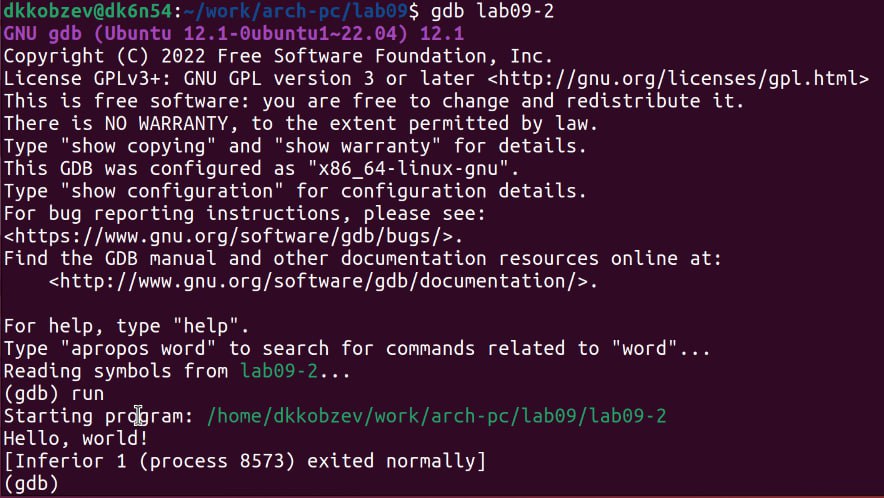
[1–6]

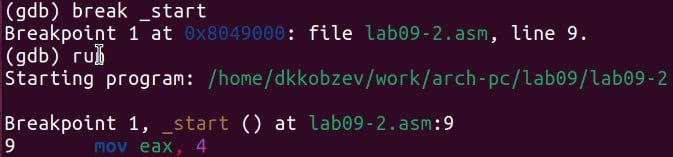
Создаем каталог для программ лабораторной работы № 9, переходим в него и создаем файл lab09-1.asm (рис. 1.1). 

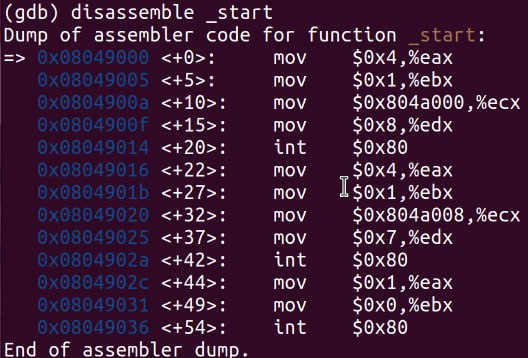
Вводим в файл программу листинга 9.1, создаем исполняемый файл и проверяем его работу (рис. 1.2), (рис. 1.3).  

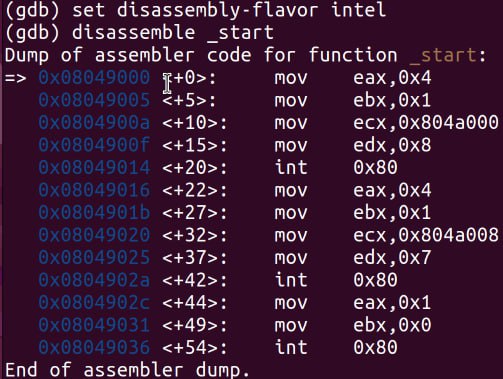
Изменяем текст программы добавив подпрограмму \_subcalcul в подпрограмму \_calcul, для вычисления выражения f(g(x)), где x вводится с клавиатуры, f(x) = 2x + 7, g(x) = 3x − 1 (рис. 1.4). 

Создаем исполняемый файл и запускаем его (рис. 1.5). 

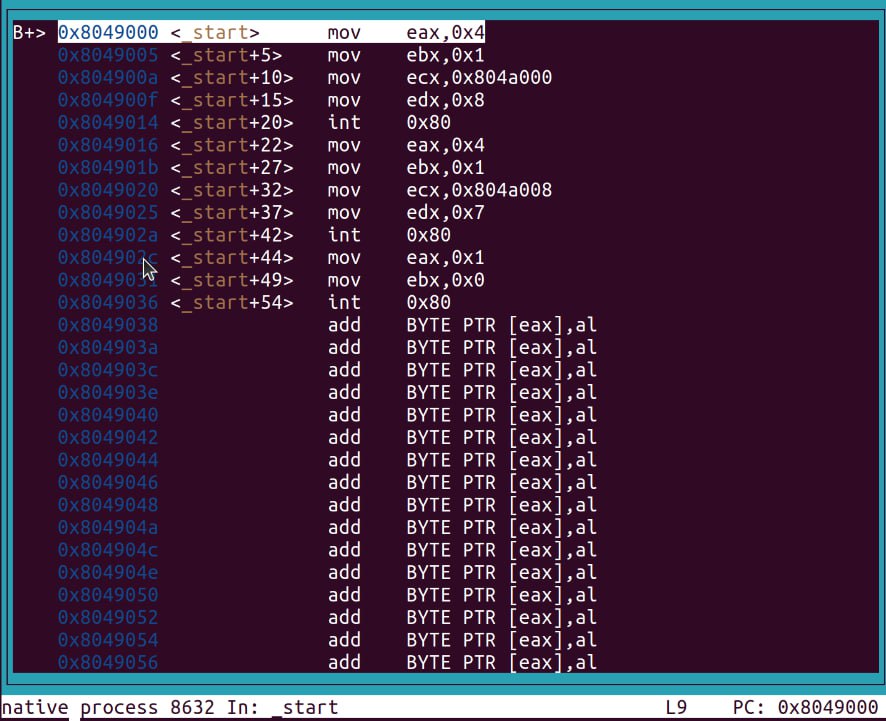
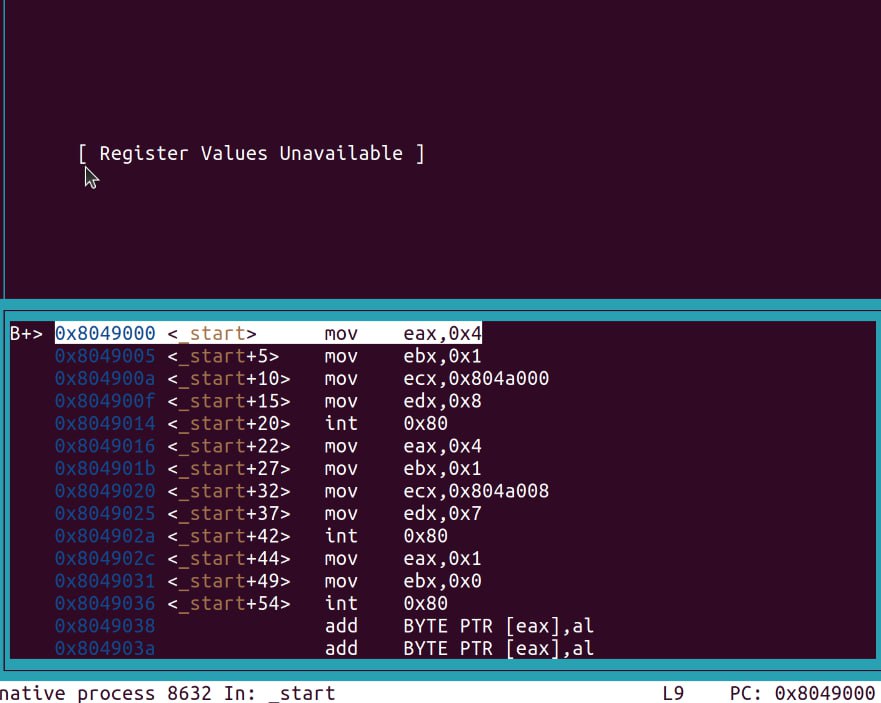
Создаем файл lab09-2.asm с текстом программы из Листинга 9.2. Получаем исполняемый файл. Загружаем исполняемый файл в отладчик gdb. Проверяем работу программы, запустив ее в оболочке GDB с помощью команды run (рис. 1.6), (рис. 1.7). Рис. 1.6: Создание файла lab09-2.asm. 

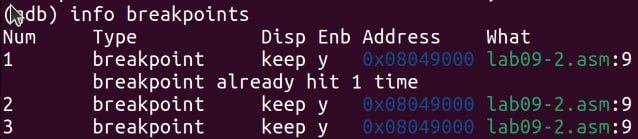
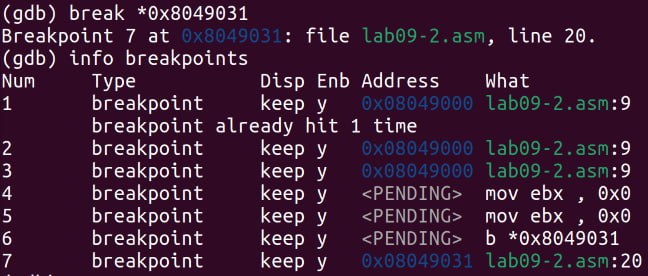
Установливаем брейкпоинт на метку \_start, и запускаем её (рис. 1.8). 

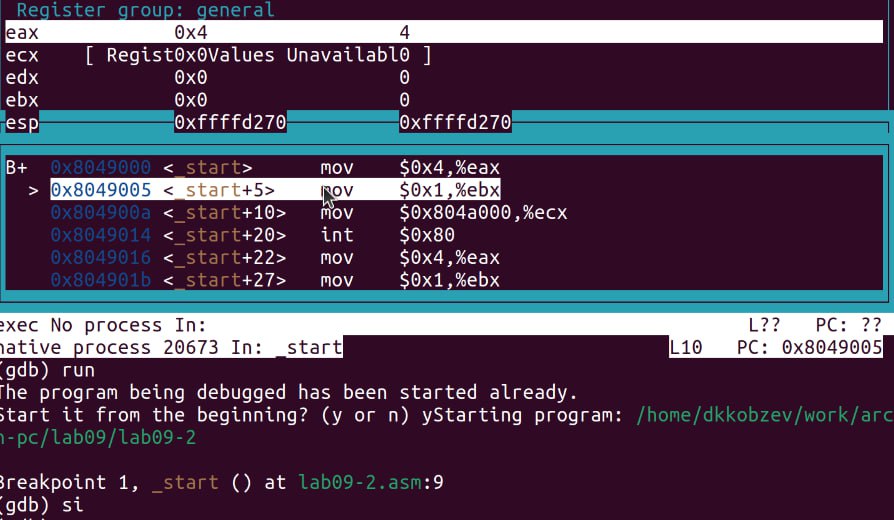
Посмотрим дисассимилированный код программы с помощью команды disassemble начиная с метки \_start (рис. 1.9). 

Переключитесь на отображение команд с Intel’овским синтаксисом, введя команду set disassembly-flavor intel (рис. 1.10). 

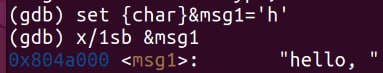
Pазличия отображения синтаксиса машинных команд в режимах ATT и Intel: противоположное расположение операнда-источника и операнда-приемника; в ATT регистры пишутся после ‘%’, а непосредственные операнды после ‘$’, в синтаксисе Intel операнды никак не помечаются.

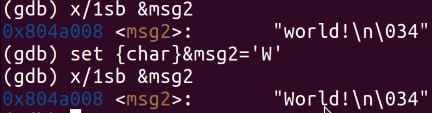
Включаем режим псевдографики для более удобного анализа программы (рис. 1.11).  

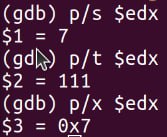
На предыдущих шагах была установлена точка останова по имени метки (\_start). Проверяем это с помощью команды info breakpoints. Установливаем еще одну точку останова по адресу предпоследней инструкции. Смотрим информацию о всех установленных точках останова (рис. 1.13), (рис. 1.14).  

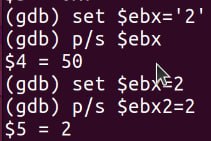
Выполняем 5 инструкций с помощью команды stepi и проследим за изменением значений регистров. В результате изменяются значения регистров eax, ebx, ecx, edx (рис. 1.15). 

Смторим значение переменной msg1 по имени (рис. 1.16). Рис. 1.16: Значение переменной msg1.

Изменяем первый символ переменной msg1 (рис. 1.17). 

Заменяем любой символ во второй переменной msg2 (рис. 1.18). 

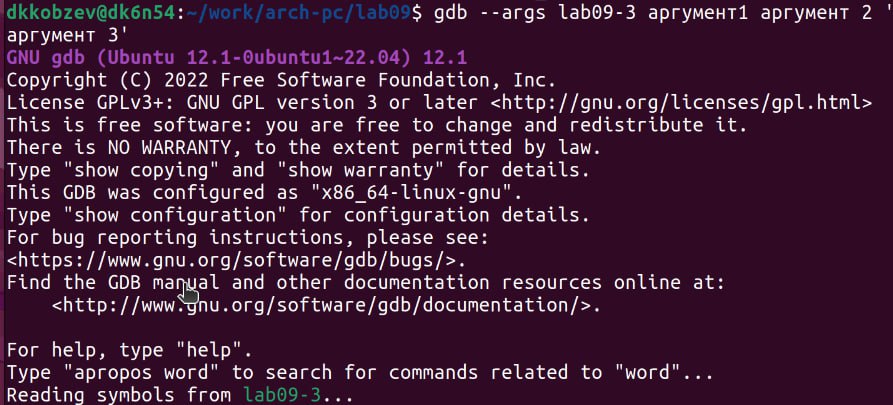
Выводим в различных форматах (в шестнадцатеричном формате, в двоичном формате и в символьном виде) значение регистра edx (рис. 1.19). 

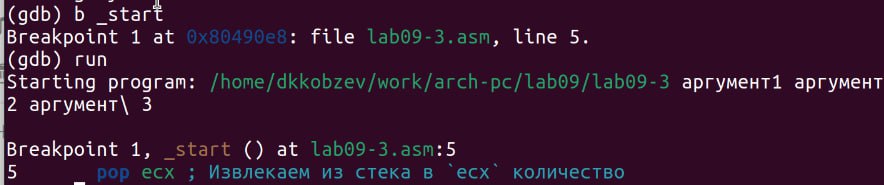
С помощью команды set изменяем значение регистра ebx (рис. 1.20). 

Использовав команду set изменили значение регистра ebx сначала на символ ‘2’, а затем на число 2, и сравнили вывод значения регистра в десятичном формате. В результате присвоения регистра значение символа ‘2’, выводится число 50, что соответствует символу в ‘2’ в таблице ASCII

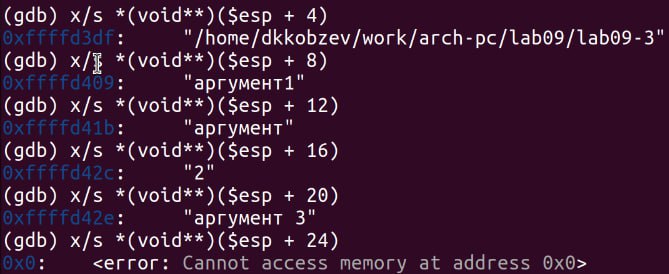
Копируем файл lab8-2.asm, созданный при выполнении лабораторной работы №8, с программой выводящей на экран аргументы командной строки (Листинг 8.2) в файл с именем lab09-3.asm (рис. 1.21). Рис. 1.21: Копия файла lab8-2.asm.

Создаем исполняемый файл (рис. 1.22). Рис. 1.22: msg2.

Загружаем исполняемый файл в отладчик, указав аргументы (рис. 1.23). 

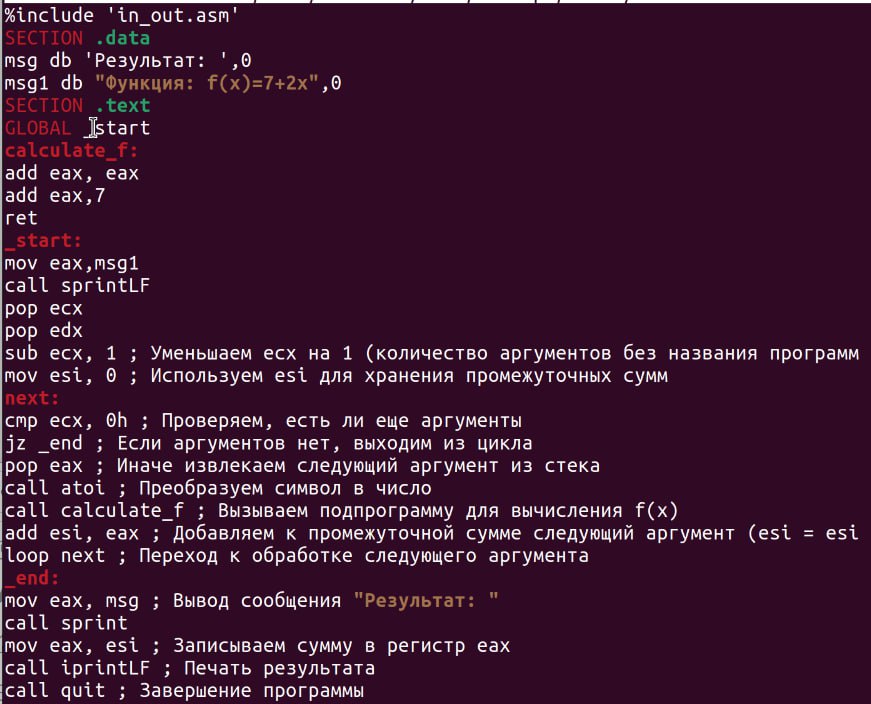
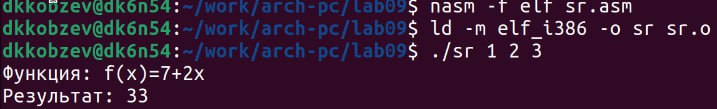
Установливаем точку останова перед первой инструкцией в программе и запускаем ее (рис. 1.24). 

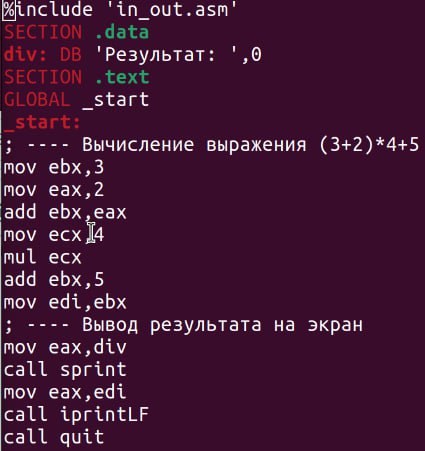
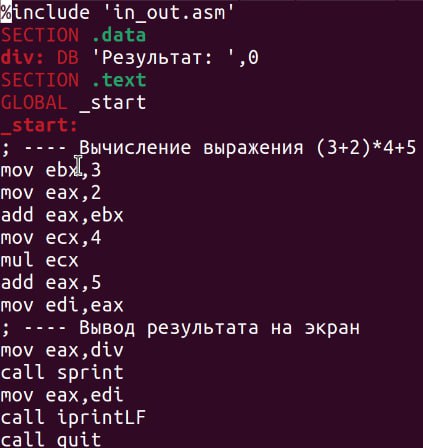
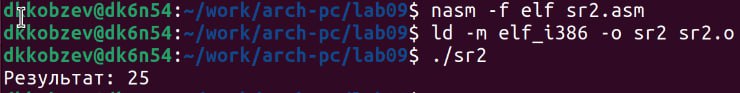
Адрес вершины стека храниться в регистре esp и по этому адресу располагается число равное количеству аргументов командной строки (включая имя программы (рис. 1.25). Рис. 1.25: Количество аргументов командной строки.

Смотрим остальные позиции стека (рис. 1.26). 

В первом хранится адрес, в остальных хранятся элементы. Элементы располо- жены с интервалом в 4 единицы, так как стек может хранить до 4 байт:каждый элемент стека занимает 4 байта, поэтому для получения следующего элемента стека мы добавляем 4 к адресу вершины.

# 4 Самостоятельная работа

Задание 1. Преобразовываем программу из лабораторной работы №8 (Задание №1 для самостоятельной работы), реализовав вычисление значения функции f(x) как подпрограмму (рис. 2.1), (рис. 2.2), (рис. 2.3). Рис. 2.1: Файл sr.asm.  

Задание 2. В листинге 9.3 приведена программа вычисления выражения (3 + 2) \* 4 + 5. При запуске данная программа дает неверный результат. Проверяем это. С помощью отладчика GDB, анализируя изменения значений регистров, определяем ошибку и исправляем ее (рис. 2.4), (рис. 2.5), (рис. 2.6).   

# 5 Выводы

В ходе выполнения лабораторной работы мною были приобретены навыки написания программ с использованием подпрограмм. Также я познакомился с методами отладки при помощи GDB и его основными возможностями.

# Список литературы

1. GNU Bash Manual [Электронный ресурс]. Free Software Foundation, 2016. URL: <https://www.gnu.org/software/bash/manual/>.

2. 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.

3. Zarrelli G. Mastering Bash. Packt Publishing, 2017. 502 с.

4. Robbins A. [Bash Pocket Reference](https://www.ncbi.nlm.nih.gov/pubmed/25246403). O’Reilly Media, 2016. 156 с.

5. Таненбаум Э. Архитектура компьютера. 6-е изд. СПб.: Питер, 2013. 874 с.

6. Таненбаум Э., Бос Х. Современные операционные системы. 4-е изд. СПб.: Питер, 2015. 1120 с.