РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 6.

дисциплина: *Архитектура компьютеров*

Студент: Подхалюзина Виолетта Михайловна

Группа: НКАбд-04-24

МОСКВА

2024 г.

Оглавление

1 Цель работы

Целью данной лабораторной работы является приобретение jсвоение арифметических инструкций языка ассемблера NASM.

2 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.

- 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.

Ввод информации с клавиатуры и вывод её на экран осуществляется в символьном виде. Кодирование этой информации производится согласно кодовой таблице символов ASCII. ASCII – сокращение от American Standard Code for Information Interchange (Американский стандартный код для обмена информацией). Согласно стандарту ASCII каждый символ кодируется одним байтом. Среди инструкций NASM нет такой, которая выводит числа (не в символьном виде). Поэтому, например, чтобы вывести число, надо предварительно преобразовать его цифры в ASCII-коды этих цифр и выводить на экран эти коды, а не само число. Если же выводить число на экран непосредственно, то экран воспримет его не как число, а как последовательность ASCII-символов – каждый байт числа будет воспринят как один ASCII-символ – и выведет на экран эти символы. Аналогичная ситуация происходит и при вводе данных с клавиатуры. Введенные данные

будут представлять собой символы, что сделает невозможным получение корректного результата при выполнении над ними арифметических операций. Для решения этой проблемы необходимо проводить преобразование ASCII символов в числа и обратно.

3 Выполнение лабораторной работы

3.1 Начало работы

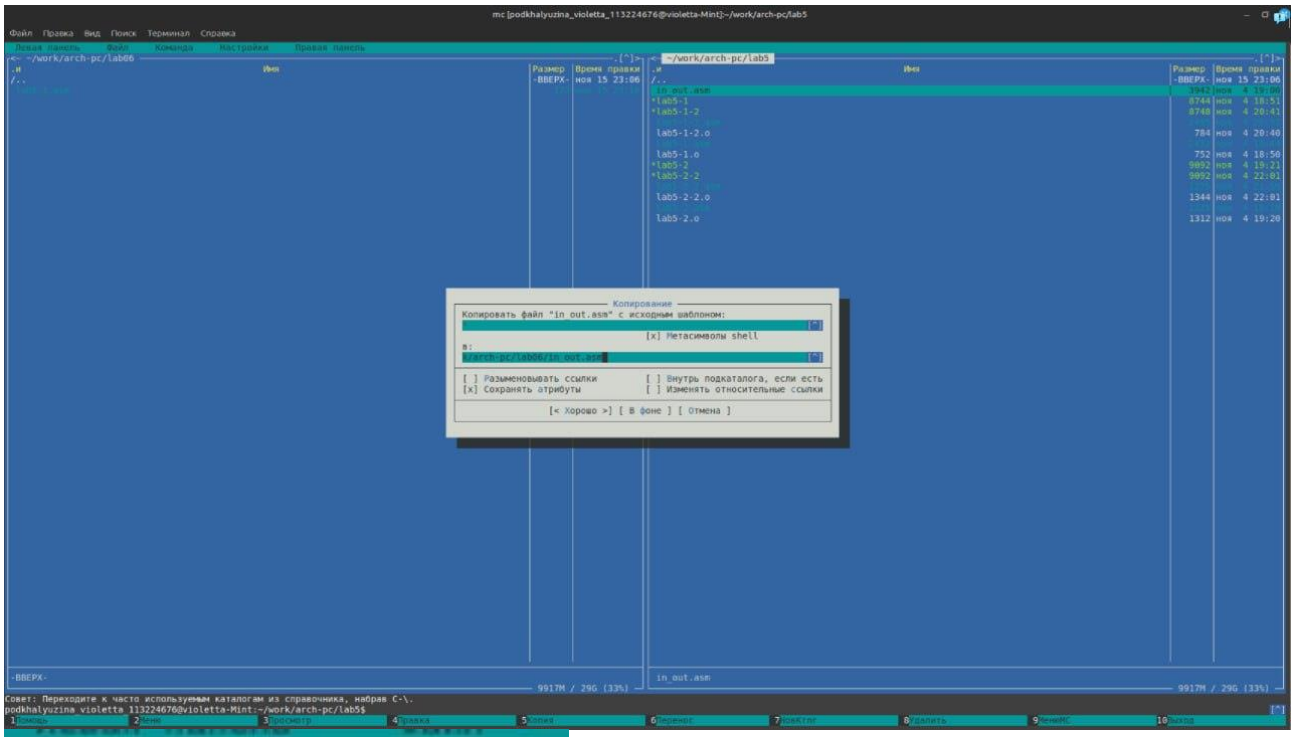
Я сделала каталог для программ лабораторной работы № 6, перешла в него и создала файл lab6-1.asm:

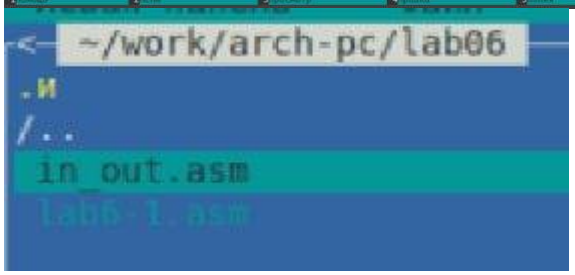
```
podkhalyuzina_violetta_1132246760violetta-Mint:~$ ls
 work     Документы   Изображения   Общедоступные   Шаблоны
 Видео    Загрузки    Музыка        'Рабочий стол'
podkhalyuzina_violetta_1132246760violetta-Mint:~$ mkdir ~/work/arch-pc/lab06
podkhalyuzina_violetta_1132246760violetta-Mint:~$ cd ~/work/arch-pc/lab
lab04/ lab06/ lab5/
podkhalyuzina_violetta_1132246760violetta-Mint:~$ cd ~/work/arch-pc/lab
lab04/ lab06/ lab5/
podkhalyuzina_violetta_1132246760violetta-Mint:~$ cd ~/work/arch-pc/lab06
podkhalyuzina_violetta_1132246760violetta-Mint:~/work/arch-pc/lab06$ touch lab6-
1.asm
podkhalyuzina_violetta_1132246760violetta-Mint:~/work/arch-pc/lab06$
```

Открыла созданный файл lab6-1.asm, вставила в него программу вывода значения регистра eax и редактировала файл с помощью редактора nano

```
 GNU nano 7.2                                  lab6-1.asm *
%include 'in_out.asm'
SECTION .bss
buf1: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax,'6'
mov ebx,'4'
add eax,ebx
mov [buf1],eax
mov eax,buf1
call sprintLF
call quit
```

Скопировала файл in_out.asm из папки lab5 в lab6 и проверила, всё ли прошло успешно.





Создала исполняемый файл и запустила его

```
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab06$ ./lab6-1
j
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab06$
```

Далее изменила текст программы и вместо символов, записала в регистры числа

```
 GNU nano 7.2
%include 'in_out.asm'
SECTION .bss
buf1: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax,6
mov ebx,4
add eax,ebx
mov [buf1],eax
mov eax,buf1
call sprintLF
call quit
```

В таблице ascii коду 10 не принадлежит символа, поэтому вывелась пустая

строка

```
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab06$ nano lab6-1.asm
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab06$ ./lab6-1


podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab06$
```

Я создала файл lab6-2.asm в каталоге ~/work/arch-pc/lab06 и ввела в него текст программы

```
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab06$ touch lab6-2.asm
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab06$ ls
in_out.asm  lab6-1  lab6-1.asm  lab6-1.o  lab6-2.asm
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab06$
```

```
 GNU nano 7.2
%include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,'6'
mov ebx,'4'
add eax,ebx
call iprintLF
call quit
```

После создала исполняемый файл и запустила его

```
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab06$ ./lab6-2
106
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab06$
```

Аналогично предыдущему примеру изменила символы на числа

```
 GNU nano 7.2
%include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,6
mov ebx,4
add eax,ebx
call iprintLF
call quit
```

Далее смотрю на полученный результат

```
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab06$ ./lab6-2
10
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab06$
```

iprintLF делает отступ строки после вывода в отличии от iprint

```
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab06$ ./lab6-2
10podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab06$
```

Далее я создала файл lab6-3.asm в каталоге ~/work/arch-pc/lab06

```
10podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab06$ touch lab6-3.asm
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab06$ ls
in_out.asm  lab6-1  lab6-1.asm  lab6-1.o  lab6-2  lab6-2.asm  lab6-2.o  lab6-3.asm
```

```
  GNU nano 7.2
;--------------------------------
; Программа вычисления выражения
;--------------------------------
%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,5 ; EAX=5
mov ebx,2 ; EBX=2
mul ebx ; EAX=EAX*EBX
add eax,3 ; EAX=EAX+3
xor edx,edx ; обнуляем EDX для корректной работы div
mov ebx,3 ; EBX=3
div ebx ; EAX=EAX/3, EDX=остаток от деления
mov edi,eax ; запись результата вычисления в 'edi'
; ---- Вывод результата на экран
mov eax,div ; вызов подпрограммы печати
call sprint ; сообщения 'Результат: '
mov eax,edi ; вызов подпрограммы печати значения
call iprintLF ; из 'edi' в виде символов
mov eax,rem ; вызов подпрограммы печати
call sprint ; сообщения 'Остаток от деления: '
mov eax,edx ; вызов подпрограммы печати значения
call iprintLF ; из 'edx' (остаток) в виде символов
call quit ; вызов подпрограммы завершения
```

Далее создала исполняемый файл и запустила его

```
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab06$ nasm -f elf lab6-3.asm
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-3 lab6-3.o
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab06$ ./lab6-3
Результат: 4
Остаток от деления: 1
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab06$
```

Изменила текст программы для вычисления выражения $f(x) = (4 * 6 + 2)/5$. И создала исполняемый файл, проверила его работу

```
  GNU nano 7.2
;---------------------------------
; Программа вычисления выражения
;---------------------------------
%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 ; EAX=4
mov ebx,6 ; EBX=6
mul ebx ; EAX=EAX*EBX
add eax,2 ; EAX=EAX+3
xor edx,edx ; обнуляем EDX для корректной работы div
mov ebx,5 ; EBX=3
div ebx ; EAX=EAX/3, EDX=остаток от деления
mov edi,eax ; запись результата вычисления в 'edi'
; ---- Вывод результата на экран
mov eax,div ; вызов подпрограммы печати
call sprint ; сообщения 'Результат: '
mov eax,edi ; вызов подпрограммы печати значения
call iprintLF ; из 'edi' в виде символов
mov eax,rem ; вызов подпрограммы печати
call sprint ; сообщения 'Остаток от деления: '
mov eax,edx ; вызов подпрограммы печати значения
call iprintLF ; из 'edx' (остаток) в виде символов
call quit ; вызов подпрограммы завершения
```

```
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab06$ nasm -f elf lab6-3.asm
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-3 lab6-3.o
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab06$ ./lab6-3
Результат: 5
Остаток от деления: 1
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab06$
```

Далее я создала файл variant.asm в каталоге ~/work/arch-pc/lab06

```
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab06$ touch variant.asm
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab06$ ls
in_out.asm  lab6-1  lab6-1.asm  lab6-1.o  lab6-2  lab6-2.asm  lab6-2.o  lab6-3  lab6-3.asm  lab6-3.o  variant.asm
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab06$
```

Далее создала исполняемый файл и запустила его

```
 GNU nano 7.2
;-----------------------------
; Программа вычисления варианта
;-----------------------------
%include 'in_out.asm'
SECTION .data
msg: DB 'Введите № студенческого билета: ',0
rem: DB 'Ваш вариант: ',0
SECTION .bss
x: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax, msg
call sprintLF
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 sprint
mov eax,edx
call iprintLF
call quit
```

```
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab06$ nasm -f elf variant.asm
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab06$ ld -m elf_i386 -o variant variant.o
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1132246761
Ваш вариант: 2
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab06$
```

1. За вывод сообщения “Ваш вариант” отвечают строки кода:

```
mov  eax,rem
call sprint
```

2.

1) Чтобы внести адрес вводимой строки x в регистр ecx mov edx, 80 - запись в регистр edx длины вводимой строки

2) Чтобы вызвать подпрограмму из внешнего файла, обеспечивающую ввод сообщения с клавиатуры

3) call atoi используется для вызова подпрограммы из внешнего файла, которая преобразует ascii-код символа в целое число и записывает результат в регистр eax

4) За вычисления варианта отвечают строки:

```
xor edx,edx
mov ebx,20
div ebx
inc edx
```

5. При выполнении инструкции div ebx остаток от деления записывается в регистр edx
6. Инструкция inc edx увеличивает значение регистра edx на 1
7. За вывод на экран результатов вычислений отвечают строки:

```
mov  eax,edx
call iprintLF
```

3.2 Самостоятельная работа

Я написала программу вычисления выражения $y = f(x)$. Программа должна выводит выражение для вычисления, выводит запрос на ввод значения x, вычисляет заданное выражение в зависимости от введенного x, выводить результат вычислений.

```
 GNU nano 7.2
%include 'in_out.asm'
SECTION .data
msg: DB 'Введите X: ',0
div: DB 'Результат y = ',0
fun: DB 'y = (12x + 3) * 5',0
SECTION .bss
x: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax,fun
call sprintLF
mov eax, msg
call sprintLF
mov ecx, x
mov edx, 80
call sread
mov eax,x
call atoi
; ---- Вычисление
mov ebx,12
mul ebx
add eax,3
xor ebx,ebx
mov ebx,5
mul ebx
mov edi,eax
; ---- Вывод результата на экран
mov eax,div
call sprint
mov eax,edi
call iprintLF
call quit
```

```
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab06$ nasm -f elf sWork.asm
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab06$ ld -m elf_i386 -o sWork sWork.o
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab06$ ./sWork
y = (12x + 3) * 5
Введите X:
1
Результат y = 75
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab06$ ./sWork
y = (12x + 3) * 5
Введите X:
6
Результат y = 375
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab06$
```

4 Контрольные вопросы для самопроверки

1. Какой синтаксис команды сложения чисел?
add destination, source (например, add ax, bx)
2. Какая команда выполняет умножение без знака?
mul source
3. Какой синтаксис команды деления чисел без знака?
div source
4. Куда помещается результат при умножении двухбайтовых операндов?
В регистры DX:AX
5. Перечислите арифметические команды с целочисленными операндами и их назначение:
- ADD: сложение
- SUB: вычитание
- MUL: умножение без знака
- IMUL: умножение со знаком
- DIV: деление без знака. - IDIV: деление со знаком
- INC: увеличение на 1. - DEC: уменьшение на 1
- NEG: смена знака
6. Где находится делимое при целочисленном делении?
В регистре DX:AX (для 16 бит) или EDX:EAX (для 32 бит)
7. Куда помещаются неполное частное и остаток при делении целочисленных операндов?
Частное — в AX (или EAX), остаток — в DX (или EDX)

5 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 с. — (In a Nutshell). — ISBN 0596009658. — URL: http://www.amazon.com/Learningbash-Shell-Programming-Nutshell/dp/0596009658.

6. Robbins A. Bash Pocket Reference. — O'Reilly Media, 2016. — 156 с. — ISBN 978-1491941591.

7. The NASM documentation. — 2021. — URL: https://www.nasm.us/docs.php.

8. Zarrelli G. Mastering Bash. — Packt Publishing, 2017. — 502 с. — 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. — БХВПетербург, 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. — 4-е изд. — СПб. : Питер, 2015. — 1120 с. — (Классика Computer Science).