Шаблон отчёта по лабораторной работе 6

Простейший вариант

Абдуллахи Бахара

Содержание

Цель работы

6.1. Цель работы

Освоение арифметических инструкций языка ассемблера NASM.

Задание

6.3. Порядок выполнения лабораторной работы

6.3.1. Символьные и численные данные в NASM

1- Создайте каталог для программам лабораторной работы № 6, перейдите в него и создайте файл lab6-1.asm:

```
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~$ mkdir ~/work/arch-pc/lab
06
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~$ cd ~/work/arch-pc/lab06
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-pc/lab06$ touch
 lab6-1.asm
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-pc/lab06$ ls
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-pc/lab06$
```

2- Рассмотрим примеры программ вывода символьных и численных значений. Програм- мы будут выводить значения записанные в регистр eax.

Введите в файл lab6-1.asm текст программы из листинга 6.1. В данной программе в ре- гистр eax записывается символ 6 (mov eax,‘6’), в регистр ebx символ 4 (mov ebx,‘4’). Далее к значению в регистре eax прибавляем значение регистра ebx (add eax,ebx, ре- зультат сложения запишется в регистр eax). Далее выводим результат. Так как для работы функции sprintLF в регистр eax должен быть записан адрес, необходимо использовать до- полнительную переменную. Для этого запишем значение регистра eax в переменную buf1 (mov [buf1],eax), а затем запишем адрес переменной buf1 в регистр eax (mov eax,buf1) и вызовем функцию sprintLF.

```
        /home/bahara123-virtualbox/work/arch-pc/lab06/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
```

Создайте исполняемый файл и запустите его.

```
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-pc/lab06$ ./lab6-1
j
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-pc/lab06$
```

В данном случае при выводе значения регистра eax мы ожидаем увидеть число 10. Однако результатом будет символ j. Это происходит потому, что код символа 6 равен 00110110 в двоичном представлении (или 54 в десятичном представлении), а код символа 4 – 00110100 (52). Команда add eax,ebx запишет в регистр eax сумму кодов – 01101010 (106), что в свою очередь является кодом символа j (см. таблицу ASCII в приложении).

3- Далее изменим текст программы и вместо символов, запишем в регистры числа. Ис- правьте текст программы (Листинг 6.1) следующим образом: замените строки

```
        /home/bahara123-virtualbox/work/arch-pc/lab06/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
```

Создайте исполняемый файл и запустите его. Как и в предыдущем случае при исполнении программы мы не получим число 10. В данном случае выводится символ с кодом 10. Пользуясь таблицей ASCII определите какому символу соответствует код 10. Отображается ли этот символ при выводе на экран?

```
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-pc/lab06$ nasm
-f elf lab6-1.asm
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-pc/lab06$ ld -m
 elf_i386 -o lab6-1 lab6-1.o
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-pc/lab06$ ./lab
6-1


bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-pc/lab06$
```

4- Как отмечалось выше, для работы с числами в файле in_out.asm реализованы подпро- граммы для преобразования ASCII символов в числа и обратно. Преобразуем текст программы из Листинга 6.1 с использованием этих функций.

Создайте файл lab6-2.asm в каталоге ~/work/arch-pc/lab06 и введите в него текст про- граммы из листинга 6.2.

```
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-pc/lab06$ touch
 lab6-2.asm
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-pc/lab06$ ls
in_out.asm  lab6-1  lab6-1.asm  lab6-1.o  lab6-2.asm
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-pc/lab06$
```

```
        /home/bahara123-virtualbox/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

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^/ Go To Line
```

Создайте исполняемый файл и запустите его

```
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-pc/lab06$ nasm
-f elf lab6-2.asm
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-pc/lab06$ ld -m
 elf_i386 -o lab6-2 lab6-2.o
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-pc/lab06$ ./lab
6-2
106
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-pc/lab06$
```

В результате работы программы мы получим число 106. В данном случае, как и в первом, команда add складывает коды символов ‘6’ и ‘4’ (54+52=106). Однако, в отличии от программы из листинга 6.1, функция iprintLF позволяет вывести число, а не символ, кодом которого является это число.

5- Аналогично предыдущему примеру изменим символы на числа. Замените строки

```
        /home/bahara123-virtualbox/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

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^/ Go To Line
```

Создайте исполняемый файл и запустите его. Какой результат будет получен при исполне- нии программы?

```
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-pc/lab06$ nasm
-f elf lab6-2.asm
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-pc/lab06$ ld -m
 elf_i386 -o lab6-2 lab6-2.o
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-pc/lab06$ ./lab
6-2
10
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-pc/lab06$
```

Замените функцию iprintLF на iprint. Создайте исполняемый файл и запустите его. Чем отличается вывод функций iprintLF и iprint?

```
        /home/bahara123-virtualbox/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 iprint

call quit


^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^/ Go To Line
```

```
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-pc/lab06$ ./lab6-2
10bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-pc/lab06$
```

6.3.2. Выполнение арифметических операций в NASM

6- В качестве примера выполнения арифметических операций в NASM приведем про- грамму вычисления арифметического выражения $f(x) = (5 * 2 + 3)/3$.

Создайте файл lab6-3.asm в каталоге ~/work/arch-pc/lab06:

```
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-pc/lab06$ touch
 ~/work/arch-pc/lab06/lab6-3.asm
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-pc/lab06$ ls
in_out.asm  lab6-1.asm  lab6-2      lab6-2.o
lab6-1      lab6-1.o    lab6-2.asm  lab6-3.asm
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-pc/lab06$
```

```
        /home/bahara123-virtualbox/work/arch-pc/lab06/lab6-3.asm *
------------------------------
; 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

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^/ Go To Line
```

Создайте исполняемый файл и запустите его. Результат работы программы должен быть следующим:

```
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-pc/lab06$ nasm
-f elf lab6-3.asm
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-pc/lab06$ ld -m
 elf_i386 -o lab6-3 lab6-3.o
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-pc/lab06$ ./lab
6-3
Результат: 4
Остаток от деления: 1
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-pc/lab06$
```

{#fig: 001 width100%}

змените текст программы для вычисления выражения $f(x) = (4 * 6 + 2)/5$. Создайте исполняемый файл и проверьте его работу.

```
        /home/bahara123-virtualbox/work/arch-pc/lab06/lab6-3.asm *
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+2
xor edx,edx          ; обнуляем EDX для корректной работы div
mov ebx,5            ; EBX=5
div ebx              ; EAX=EAX/5, EDX=остаток от деления

mov edi,eax          ; запись результата вычисления в 'edi'

; ---- Вывод результата на экран

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^/ Go To Line
```

```
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-pc/lab06$ nasm
-f elf lab6-3.asm
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-pc/lab06$ ld -m
 elf_i386 -o lab6-3 lab6-3.o
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-pc/lab06$ ./lab
6-3
Результат: 5
Остаток от деления: 1
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-pc/lab06$
```

7- В качестве другого примера рассмотрим программу вычисления варианта задания по номеру студенческого билета, работающую по следующему алгоритму:

Создайте файл variant.asm в каталоге ~/work/arch-pc/lab06:

```
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-pc/lab06$ touch
 variant.asm
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-pc/lab06$ ls
in_out.asm  lab6-1.asm  lab6-2      lab6-2.o  lab6-3.asm  variant.asm
lab6-1      lab6-1.o    lab6-2.asm  lab6-3    lab6-3.o
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-pc/lab06$
```

Внимательно изучите текст программы из листинга 6.4 и введите в файл variant.asm.

```
;-----------------------------
; 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

                    [ Wrote 38 lines ]
^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^/ Go To Line
```

Создайте исполняемый файл и запустите его. Проверьте результат работы программы вычислив номер варианта аналитически.

```
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-pc/lab06$ na
sm -f elf variant.asm
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-pc/lab06$ ld
 -m elf_i386 -o variant variant.o
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-pc/lab06$ ./
variant
Введите No студенческого билета:
1032225714
Ваш вариант: 15
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-pc/lab06$
```

Включите в отчет по выполнению лабораторной работы ответы на следующие вопросы:

1- Какие строки листинга 6.4 отвечают за вывод на экран сообщения ‘Ваш вариант:’?

1- Строка mov eax, rem и строка call sprint отвечают за вывод на экран сообщения “Ваш вариант:”.

2- Для чего используется следующие инструкции?

mov ecx, x mov edx, 80 call sread

2- Инструкции mov ecx, x и mov edx, 80 используются для подготовки аргументов перед вызовом подпрограммы sread. mov ecx, x загружает адрес переменной x в регистр ecx, который будет использован в качестве

аргумента для функции sread. mov edx, 80 загружает значение 80 в регистр edx, указывая функции sread, сколько байт нужно прочитать.

3- Для чего используется инструкция “call atoi”?

3- Инструкция call atoi используется для вызова подпрограммы atoi, которая преобразует ASCII-код, хранящийся в регистре eax, в число. Результат преобразования сохраняется в регистре eax.

4- Какие строки листинга 6.4 отвечают за вычисления варианта?

4- Строка xor edx, edx и строка mov ebx, 20 отвечают за подготовку значений перед выполнением вычислений варианта. xor edx, edx устанавливает регистр edx в ноль, а mov ebx, 20 загружает значение 20 в регистр ebx, которое будет использовано для деления.

5- В какой регистр записывается остаток от деления при выполнении инструкции “div ebx”?

5- Остаток от деления при выполнении инструкции div ebx записывается в регистр edx.

6- Для чего используется инструкция “inc edx”?

6- Инструкция inc edx используется для увеличения значения в регистре edx на единицу. В данном случае, она увеличивает значение остатка от деления на 1.

7- Какие строки листинга 6.4 отвечают за вывод на экран результата вычислений?

7- Строка mov eax, edx и строка call iprintLF отвечают за вывод на экран результата вычислений. mov eax, edx загружает значение в регистр eax, чтобы передать его в функцию iprintLF, которая выводит значение на экран с новой строкой.

6.4. Задание для самостоятельной работы

1- Написать программу вычисления выражения П= р(о). грамма должна выводить выражение для вычисления, выводить запрос на ввод значения *x*, вычислять задан- ное выражение в зависимости от введенного *x*, выводить результат вычислений. Вид функции *f*(*x*) выбрать из таблицы 6.3 вариантов заданий в соответствии с номером полученным при выполнении лабораторной работы. Создайте исполняемый файл и проверьте его работу для значений *x*1 и *x*2 из 6.3.

no 9- 10 + (31*x* – 5) x1-3 x2-1

При выполнении задания преобразовывать (упрощать) выражения для *f*(*x*) нельзя. При выполнении деления в качестве результата можно

использовать только целую часть от деления и не учитывать остаток (т.е. 5 : 2 = 2).

```
 GNU nano 7.2                          /home/bahara123-virtualbox/work/arch-
msg: DB 'Введите число: ',0
rem: DB 'результат: ',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`

mov ebx,eax
mov ebx,5
add eax,ebx
mul eax
sub eax,3


mov edi,eax

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

call quit
```

```
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-pc/lab06$ ./variant
Введите число:
5
результат: 97
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-pc/lab06$ ./variant
Введите число:
1
результат: 33
bahara123-virtualbox@bahara123-virtualbox-VirtualBox:~/work/arch-pc/lab06$
```

Теоретическое введение

Здесь описываются теоретические аспекты, связанные с выполнением работы.

Например, в табл. @tbl:std-dir приведено краткое описание стандартных каталогов Unix.

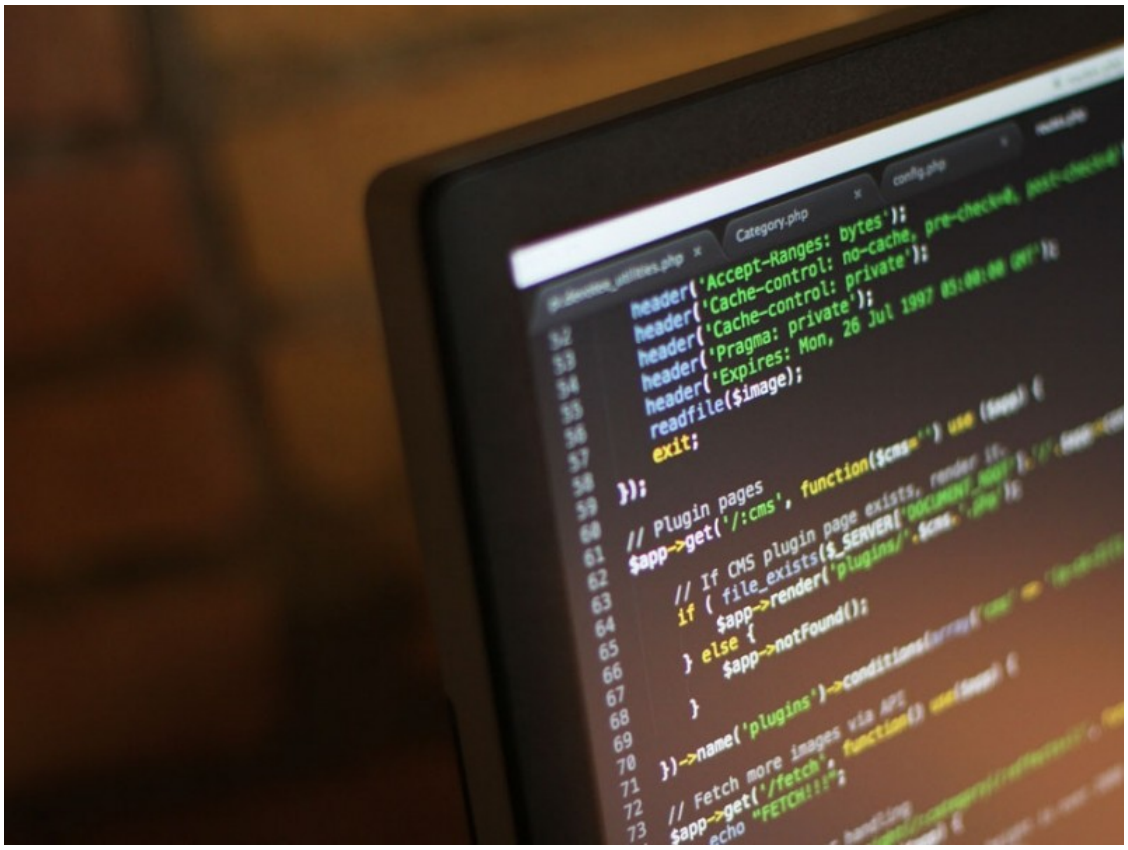
Описание некоторых каталогов файловой системы GNU Linux {#tbl:std-dir}

Имя каталога	Описание каталога
`/`	Корневая директория, содержащая всю файловую
`/bin`	Основные системные утилиты, необходимые как в однопользовательском режиме, так и при обычной работе всем пользователям
`/etc`	Общесистемные конфигурационные файлы и файлы конфигурации установленных программ
`/home`	Содержит домашние директории пользователей, которые, в свою очередь, содержат персональные настройки и данные пользователя
`/media`	Точки монтирования для сменных носителей
`/root`	Домашняя директория пользователя `root`
`/tmp`	Временные файлы
`/usr`	Вторичная иерархия для данных пользователя

Более подробно об Unix см. в [@gnu-doc:bash;@newham:2005:bash;@zarrelli:2017:bash;@robbins:2013:bash;@tannenbaum:arch-pc:ru;@tannenbaum:modern-os:ru].

Выполнение лабораторной работы

Описываются проведённые действия, в качестве иллюстрации даётся ссылка на иллюстрацию (рис. @fig:001).



Название рисунка

Выводы

Здесь кратко описываются итоги проделанной работы.

Список литературы