РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 6

дисциплина: Архитектура компьютера

Студент: Нурмухаметова Каролина

Группа: НКАбд-05-25

МОСКВА

2025 г.

Оглавление

1 Цель работы

Освоение арифметических инструкций языка ассемблера NASM.

2 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. - 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 Символьные и численные данные в NASM

1. Создала каталог для программ лабораторной работы № 6, перешла в него и создала файл lab6-1.asm (Рис. 3.1.1).

```
kmnurmukhametov@dk6n01 ~ $ mkdir ~/work/arch-pc/lab06
kmnurmukhametov@dk6n01 ~ $ cd ~/work/arch-pc/lab06
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab06 $ touch lab6-1.asm
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab06 $
```

Рис. 3.1.1 — Создание каталога

2. Рассмотрели примеры программ вывода символьных и численных значений. Программы выводили значения записанные в регистр eax. Ввела в файл lab6-1.asm текст программы (Рис, 3.1.2). В данной программе в регистр 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, результат сложения запишется в регистр eax). Далее выводим результат. Так как для работы функции sprintLF в регистр eax должен быть записан адрес, использовали дополнительную переменную. Для этого записали значение регистра eax в переменную buf1 (mov [buf1],eax), а затем записали адрес переменной buf1 в регистр eax (mov eax,buf1) и вызвали функцию sprintLF. Создала исполняемый файл и запустила его (Рис. 3.1.3).

```
1  %include 'in_out.asm'
2
3  SECTION .bss
4  buf1: RESB 80
5
6   SECTION .text
7   GLOBAL _start
8    _start:
9
10   mov eax,'6'
11   mov ebx,'4'
12   add eax,ebx
13   mov [buf1],eax
14   mov eax,buf1
15   call sprintLF
16
17   call quit
```

Рис. 3.1.2 — Текст программы lab6-1.asm

```
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab06 $ ./lab6-1
j
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab06 $
```

Рис. 3.1.3 — Вывод программы lab6-1.asm

В данном случае при выводе значения регистра eax мы ожидали увидеть число 10. Однако результатом был 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 записала в регистр eax сумму кодов – 01101010 (106), что в свою очередь является кодом символа j.

3. Далее изменим текст программы и вместо символов, запишем в регистры числа. Создала файл lab6-1-1.asm следующим образом: заменила строки mov eax,'6' mov ebx,'4' на строки mov eax,6 mov ebx,4 (Рис. 3.1.4).

```
1  %include 'in_out.asm'
2
3  SECTION .bss
4  buf1: RESB 80
5
6   SECTION .text
7   GLOBAL _start
8    _start:
9
10   mov eax,6
11   mov ebx,4
12   add eax,ebx
13   mov [buf1],eax
14   mov eax,buf1
15   call sprintLF
16
17   call quit
```

Рис. 3.1.4 — Текст программы lab6-1-1.asm

Запустила его. Как и в предыдущем случае при исполнении программы мы не получили число 10. В данном случае выводится символ с кодом 10. Символ с кодом 10 в таблице ASCII — это LF (Line Feed), что означает "перевод строки" или "новая строка". Этот символ не отображается на экране (Рис. 3.1.5).

```
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab06 $ touch lab6-1-1.asm
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1-1.asm
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1-1 lab6-1-1.o
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab06 $ ./lab6-1-1


kmnurmukhametov@dk6n01 ~/work/arch-pc/lab06 $
```

Рис. 3.1.5 — Вывод программы lab6-1-1.asm

4. Как отмечалось выше, для работы с числами в файле in_out.asm реализованы подпрограммы для преобразования ASCII символов в числа и обратно. Преобразовали текст программы lab6-1.asm с использованием этих функций. Создала файл lab6-2.asm в каталоге ~/work/arch-pc/lab06 и ввела в него текст программы lab6-2.asm (Рис. 3.1.6). Создала исполняемый файл и запустила его (Рис. 3.1.7).

```
%include 'in_out.asm'

 SECTION .text
 GLOBAL _start
  _start:

  mov eax,'6'
  mov ebx,'4'
  add eax,ebx
  call iprintLF

  call quit
```

Рис. 3.1.6 — Текст программы lab6-2.asm

```
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab06 $ touch lab6-2.asm
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab06 $ ./lab6-2
106
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab06 $
```

Рис. 3.1.7 — Вывод программы lab6-2.asm

В результате работы программы мы получили 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 изменила символы на числа. Заменила строки mov eax, '6' mov ebx, '4' на строки mov eax, 6 mov ebx, 4 (Рис. 3.1.8). Создала файл lab6-2-1.asm и запустила его (Рис. 3.1.9).

```
%include 'in_out.asm'

 SECTION .text
 GLOBAL _start
  _start:

  mov eax,6
  mov ebx,4
  add eax,ebx
  call iprintLF

  call quit
```

Рис. 3.1.8 — Текст программы lab6-2-1.asm

```
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab06 $ touch lab6-2-1.asm
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2-1.asm
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2-1 lab6-2-1.o
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab06 $ ./lab6-2-1
10
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab06 $
```

Рис. 3.1.9 — Вывод программы lab6-2-1.asm

При выполнении программы получили результат 10.

Заменила функцию iprintLF на iprint (Рис. 3.1.10). Создала файл lab6-2-2.asm и запустила его (Рис. 3.1.11). Чем отличается вывод функций iprintLF и iprint?

```
1  %include 'in_out.asm'
2
3   SECTION .text
4   GLOBAL _start
5    _start:
6
7    mov eax,6
8    mov ebx,4
9    add eax,ebx
10   call iprint
11
12   call quit
```

Рис. 3.1.10 — Текст программы lab6-2-2.asm

```
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab06 $ touch lab6-2-2.asm
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2-2.asm
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2-2 lab6-2-2.o
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab06 $ ./lab6-2-2
10kmnurmukhametov@dk6n01 ~/work/arch-pc/lab06 $
```

Рис. 3.1.11 — Вывод программы lab6-2-2.asm

3.2 Выполнение арифметических операций в NASM

6. В качестве примера выполнения арифметических операций в NASM привела программу вычисления арифметического выражения $f(x) = (5 * 2 + 3)/3$. Создала файл lab6-3.asm в каталоге lab06.

Внимательно изучила текст программы из листинга 6.3 и ввела в lab6-3.asm (Рис. 3.2.1).

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

Рис. 3.2.1 — Текст программы lab6-3.asm

Создала исполняемый файл и запустила его (Рис. 3.2.2).

```
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab06 $ touch lab6-3.asm
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab06 $ nasm -f elf lab6-3.asm
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab06 $ ./lab6-3
Результат: 4
Остаток от деления: 1
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab06 $
```

Рис. 3.2.2 — Вывод программы lab6-3.asm

Изменила текст программы для вычисления выражения $f(x) = (4 * 6 + 2)/5$ (Рис. 3.2.3). Создала файл lab6-3-1.asm и проверила его работу (Рис. 3.2.4)

```
;--------------------------------
; 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+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

   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
```

Рис. 3.2.3 — Текст программы lab6-3-1.asm

```
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab06 $ touch lab6-3-1.asm
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab06 $ nasm -f elf lab6-3-1.asm
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-3-1 lab6-3-1.o
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab06 $ ./lab6-3-1
Результат: 5
Остаток от деления: 1
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab06 $
```

Рис. 3.2.4 — Вывод программы lab6-3-1.asm

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

• вывести запрос на введение № студенческого билета

• вычислить номер варианта по формуле: (Sn mod 20) + 1, где Sn – номер студенческого билета (В данном случае a mod b – это остаток от деления a на b).

• вывести на экран номер варианта.

В данном случае число, над которым необходимо проводить арифметические

операции, вводится с клавиатуры. Как отмечалось выше ввод с клавиатуры осуществляется в символьном виде и для корректной работы арифметических операций в NASM символы необходимо преобразовать в числа. Для этого может быть использована функция atoi из файла in_out.asm.

Создала файл variant.asm в каталоге lab06.

Внимательно изучила текст программы и ввела в файл variant.asm (Рис. 3.2.5).

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

Рис. 3.2.5 — Текст программы variant.asm

Создала исполняемый файл и запустила его. Проверила результат работы программы вычислив номер варианта аналитически (Рис. 3.2.6).

```
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab06 $ touch variant.asm
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab06 $ nasm -f elf variant.asm
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o variant variant.o
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab06 $ ./variant
Введите № студенческого билета:
1032253526
Ваш вариант: 7
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab06 $
```

Рис. 3.2.6 — Вывод программы variant.asm

Ответы на вопросы:

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

Ответ: mov eax,rem

call sprint

2. Для чего используется следующие инструкции?

mov ecx, x

mov edx, 80

call sread

Ответ: эти строки кода отвечают за чтение ввода пользователя с клавиатуры.

3. Для чего используется инструкция “call atoi”?

Ответ: эта инструкция используется для преобразования ASCII кода в число.

4. Какие строки листинга 6.4 отвечают за вычисления варианта?

Ответ: mov eax,x

call atoi

xor edx,edx

mov ebx,20

div ebx

inc edx

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

Ответ: в edx.

6. Для чего используется инструкция “inc edx”?

Ответ: увеличивает edx на 1.

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

Ответ: mov eax,rem

call sprint

mov eax,edx

call iprintLF

4 Задание для самостоятельной работы

4.1 Написала программу вычисления выражения y = f(x). Программа должна выводить выражение для вычисления, выводить запрос на ввод значения x, вычислять заданное выражение в зависимости от введённого x, выводить результат вычислений. Вид функции f(x) выбрала из таблицы 6.3 вариантов заданий в соответствии с номером полученным при выполнении лабораторной работы (7): $5(x - 1)^2$. Создала исполняемый файл lab6-sam.asm (Рис. 4.1) и проверила его работу для значений x_1 и x_2 из 6.3: 3, 5 (Рис. 4.2).

```
;---------------------------------
; Программа вычисления выражения y = 5(x-1)^2
;---------------------------------

%include 'in_out.asm'

 SECTION .data
 msg: DB 'Введите x: ',0
 result: DB 'Ответ: ',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`

  sub eax, 1        ; eax = x - 1
  mov ebx, eax      ; ebx = x - 1

  imul eax, ebx     ; eax = (x - 1)^2
  mov ebx, 5        ; ebx = 5
  mul ebx      ; eax = 5 * (x - 1)^2
  mov edx, eax      ; сохранить результат в edx

  mov eax, result   ; подготовить аргумент для функции sprint
  call sprint       ; вызвать функцию для вывода результата
  mov eax, edx      ; вернуть результат в eax
  call iprintLF

  call quit
```

Рис. 4.1 — Текст программы lab6-sam.asm

```
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab06 $ nasm -f elf lab6-sam.asm
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-sam lab6-sam.o
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab06 $ ./lab6-sam
Введите x:
3
Ответ: 20
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab06 $ ./lab6-sam
Введите x:
5
Ответ: 80
kmnurmukhametov@dk6n01 ~/work/arch-pc/lab06 $
```

Рис. 4.2 — Вывод программы lab6-sam.asm

5 Выводы

При выполнении данной лабораторной работы я освоила арифметические инструкции языка ассемблера NASM.

6 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/Learning-bash-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. — БХВ-Петербург, 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).