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: Терентьев Максим Сергеевич

Группа: НКАбд-05-25

МОСКВА

2025 г.

Содержание

1. Цель работы 3
2. Порядок выполнения лабораторной работы 4
   2.1. Символьные и численные данные в NASM 4
   2.2. Выполнение арифметических операций в NASM 6
   2.3. Ответы на вопросы 8
3. Задание для самостоятельной работы 9
Вывод 10

1. Цель работы

Освоение арифметических инструкций языка ассемблера NASM.

2. Порядок выполнения лабораторной работы

2.1. Символьные и численные данные в NASM

Создадим каталог lab06, и создадим в нем файл lab6-1.asm:

```
msterentjev@fedora:~$ mkdir ~/work/arch-pc/lab06
msterentjev@fedora:~$ ls ~/work/arch-pc
lab04  lab05  lab06
msterentjev@fedora:~$ cd ~/work/arch-pc/lab06
msterentjev@fedora:~/work/arch-pc/lab06$ touch tab6-1.asm
msterentjev@fedora:~/work/arch-pc/lab06$ ls
tab6-1.asm
msterentjev@fedora:~/work/arch-pc/lab06$ mv tab6-1.asm lab6-1.asm
msterentjev@fedora:~/work/arch-pc/lab06$ ls
lab6-1.asm
msterentjev@fedora:~/work/arch-pc/lab06$
```

Рис. 1. Создание lab6-1.asm

Теперь введем в этот файл код из листинга 6.1, создадим исполняемый файл и запустим, предварительно скопировав в каталог файл in_out.asm:

Открыть lab6-1.asm ~/work/arch-pc/lab06

```
%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
```

Рис. 2. Текст программы lab6-1.asm

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

Рис. 3. Запуск lab6-1

Вывод программы отличается от 10, поскольку код посчитал не сумму чисел 6 и 4 а сумму кодов символов ‘6’ и ‘4’ . По таблице ASCII коды этих символов в сумме дают код символа j.

Далее изменим текст программы и вместо символов, запишем в регистры числа:

Открыть *lab6-1.asm ~/work/arch-pc/lab06

```
%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
```

Рис. 4. Измененный текст программы lab6-1.asm

Теперь снова сделаем исполняемый файл и запустим его:

```
msterentjev@fedora:~/work/arch-pc/lab06$ gedit lab6-1.asm
msterentjev@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
msterentjev@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
msterentjev@fedora:~/work/arch-pc/lab06$ ./lab6-1


msterentjev@fedora:~/work/arch-pc/lab06$
```

Рис. 5. Работа измененного файла lab6-1

Как видим, мы снова не получили 10, а получили символ с кодом 10 то есть переход на новую строку.

для работы с числами в файле in_out.asm реализованы подпрограммы для преобразования ASCII символов в числа и обратно. Вставим код из Листинга 6.2 в lab6-2.asm:

Открыть *lab6-2.asm ~/work/arch-pc/lab06

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

Рис. 6. Код lab6-2.asm

Соберем исполняемый файл и проверим работу данной программы:

```
msterentjev@fedora:~/work/arch-pc/lab06$ touch lab6-2.asm
msterentjev@fedora:~/work/arch-pc/lab06$ gedit lab6-2.asm
msterentjev@fedora:~/work/arch-pc/lab06$ gedit lab6-2.asm
msterentjev@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
msterentjev@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
msterentjev@fedora:~/work/arch-pc/lab06$ ./lab6-2
106
msterentjev@fedora:~/work/arch-pc/lab06$
```

Рис. 7. Создание и проверка lab6-2.asm

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.

Аналогично предыдущему примеру изменим символы на числа:

Открыть lab6-2.asm ~/work/arch-pc/lab06

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

Рис. 8. Измененный текст lab6-2.asm

Создадим исполняемый файл и запустим:

```
msterentjev@fedora:~/work/arch-pc/lab06$ gedit lab6-2.asm
msterentjev@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
msterentjev@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
msterentjev@fedora:~/work/arch-pc/lab06$ ./lab6-2
10
msterentjev@fedora:~/work/arch-pc/lab06$
```

Рис. 9. Запуск измененного lab6-2

В этот раз мы получили сумму 6 и 4 – 10.

Заменим функцию iprintLF на iprint, обновим исполняемый файл и проверим работу программы:

Открыть *lab6-2.asm ~/work/arch-pc/lab06

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

Рис. 10. Замена iprintLF на iprint

```
msterentjev@fedora:~/work/arch-pc/lab06$ gedit lab6-2.asm
msterentjev@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
msterentjev@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
msterentjev@fedora:~/work/arch-pc/lab06$ ./lab6-2
10msterentjev@fedora:~/work/arch-pc/lab06$
```

Рис. 11. Вывод с помощью iprint

Мы получили тот же результат, что и у прошлого кода, только без переноса строки.

2.2. Выполнение арифметических операций в NASM

В качестве примера выполнения арифметических операций в NASM приведем программу вычисления арифметического выражения $f(x) = (5*2+3)/3$. Создадим файл lab6-3.asm, вставим в него код из листинга 6.3:

Открыть *lab6-3.asm ~/work/arch-pc/lab06

```
;---;
; Программа вычисления выражения
;---
%include 'in_out.asm' ; подключение внешнего файла

SECTION .data
msg1: DB 'Результат: ',0
msg2: 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,msg1 ; вызов подпрограммы печати
call sprint  ; сообщения 'Результат: '
mov eax,edi  ; вызов подпрограммы печати значения
call iprintLF ; из 'edi' в виде символов

mov eax,msg2 ; 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. 12. Код программы для lab6-3.asm

Создадим исп. файл и запустим:

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

Рис. 13. Работа lab6-3

Вывод программы совпадает с выводом из лабораторной работы.

Изменим текст программы для вычисления выражения $f(x) = (4 * 6 + 2)/5$:

Открыть *lab6-3.asm ~/work/arch-pc/lab06

```
;---;
; Программа вычисления выражения
;---
%include 'in_out.asm' ; подключение внешнего файла

SECTION .data
msg1: DB 'Результат: ',0
msg2: 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,msg1 ; вызов подпрограммы печати
call sprint  ; сообщения 'Результат: '
mov eax,edi  ; вызов подпрограммы печати значения
call iprintLF ; из 'edi' в виде символов

mov eax,msg2 ; 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. 14. Измененный текст программы lab6-3.asm

Снова сделаем исп. файл и запустим:

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

Рис. 15. Работа lab6-3.asm для другого выражения

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

- вывести запрос на введение № студенческого билета
- вычислить номер варианта по формуле: $(Sn \bmod 20)+1$, где Sn – номер студенческого билета (В данном случае $a \bmod b$ – это остаток отделения 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. Создадим файл variant.asm и вставим в него код из листинга 6.4:

Открыть *variant.asm ~/work/arch-pc/lab06

```
;---;
; 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 sprint

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. 16. Код файла variant.asm из листинга 6.4

Теперь создадим исп. файл и проверим работу:

```
msterentjev@fedora:~/work/arch-pc/lab06$ touch variant.asm
msterentjev@fedora:~/work/arch-pc/lab06$ gedit variant.asm
msterentjev@fedora:~/work/arch-pc/lab06$ nasm -f elf variant.asm
msterentjev@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o variant variant.o
msterentjev@fedora:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета: 1032253550
Ваш вариант: 11
msterentjev@fedora:~/work/arch-pc/lab06$
```

Рис. 17. Работа программы variant.asm

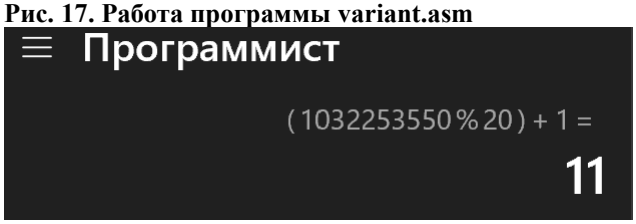

Рис. 18. Расчёт по формуле на калькуляторе

Результат в калькуляторе и в терминале совпадает, значит программа работает успешно.

2.3. 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 инструкции используются для чтения ввода от пользователя:
   - mov ecx, x - помещает адрес буфера x в регистр ECX
   - mov edx, 80 - устанавливает максимальную длину ввода (80 байт)
   - call sread - вызывает функцию чтения строки из файла in_out.asm
3) Для чего используется инструкция "call atoi"?
   Ответ:
   Инструкция call atoi вызывает функцию преобразования строки в число. Она преобразует введенную пользователем строку (номер студенческого билета) из ASCII-формата в целое число, которое сохраняется в регистре EAX.
4) Какие строки листинга 6.4 отвечают за вычисления варианта?
   Ответ:
   xor edx,edx ; обнуление EDX
   mov ebx,20 ; загрузка делителя 20
   div ebx ; деление номера билета на 20
   inc edx ; увеличение остатка на 1
5) В какой регистр записывается остаток от деления при выполнении инструкции "div ebx"?
   Ответ:
   При выполнении инструкции div ebx остаток от деления записывается в регистр EDX.
6) Для чего используется инструкция "inc edx"?
   Ответ:
   Инструкция inc edx увеличивает значение в регистре EDX на 1. Это нужно потому, что остаток от деления может быть от 0 до 19, а варианты заданий обычно нумеруются с 1 до 20.
7) Какие строки листинга 6.4 отвечают за вывод на экран результата вычислений?
   Ответ:

```
mov eax,rem      ; загрузка адреса строки "Ваш вариант: "
call sprint      ; вывод этой строки
mov eax,edx      ; загрузка вычисленного варианта
call iprintLF    ; вывод варианта с переводом строки
```

3. Задание для самостоятельной работы

Создадим программу, вычисляющую значение выражения y по введенному x. По моему варианту (11, см. рис. 17), в таблице дано выражение 10(x+1)−10. x1=1, x2=7:

Открыть *labobus6.asm ~/work/arch-pc/lab06

```
%include 'in_out.asm'

SECTION .data
expr: DB 'Вычисление выражения y = 10(x+1)-10',0
prompt: DB 'Введите x: ',0
result: DB 'Результат: y = ',0

SECTION .bss
x: RESB 10

SECTION .text
GLOBAL _start
_start:

mov eax, expr
call sprintLF

mov eax, prompt
call sprint

mov ecx, x
mov edx, 10
call sread

mov eax, x
call atoi

mov ebx, eax
add ebx, 1
mov eax, ebx
mov ecx, 10
mul ecx
sub eax, 10

mov edi, eax

mov eax, result
call sprint
mov eax, edi
call iprintLF

call quit
```

Рис. 19. Код программы для ср

Создадим исп. файл и проверим работу для x=1 и x=7:

```
msterentjev@fedora:~/work/arch-pc/lab06$ gedit labobus6.asm
msterentjev@fedora:~/work/arch-pc/lab06$ nasm -f elf labobus6.asm
msterentjev@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o labobus6 labobus6.o
msterentjev@fedora:~/work/arch-pc/lab06$ ./labobus6
Вычисление выражения y = 10(x+1)-10
Введите x: 1
Результат: y = 10
msterentjev@fedora:~/work/arch-pc/lab06$ ./labobus6
Вычисление выражения y = 10(x+1)-10
Введите x: 7
Результат: y = 70
msterentjev@fedora:~/work/arch-pc/lab06$
```

Рис. 20. Работа программы для x=1, x=7

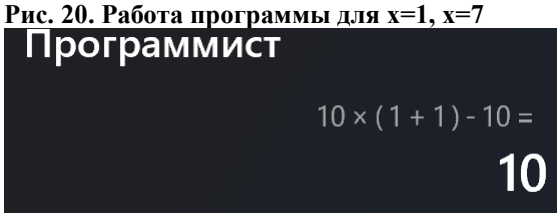

Рис. 21. Рассчет для x=1 в калькуляторе

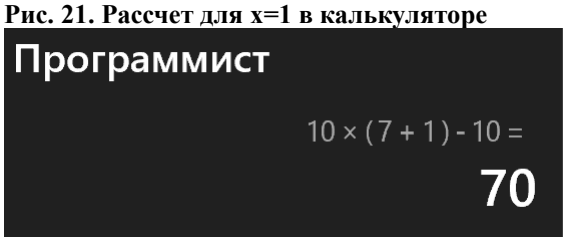

Рис. 22. Рассчет для x=7 в калькуляторе

Вывод программы совпадает с выводом калькулятора, значит программа работает успешно.

Вывод

В ходе работы были освоены арифметические инструкции языка ассемблера NASM, включая операции сложения, вычитания, умножения и деления, а также организация ввода-вывода данных и работа с регистрами процессора