Отчет по лабораторной работе № 6

Дисциплина: Архитектура компьютера

Дроздова Дарья Игоревна

Содержание

1 Цель работы

Целью данной лабораторной работы является освоение арифметических инструкций языка ассемблера NASM.

2 Выполнение лабораторной работы

1. **Символьные и численные данные в NASM**

   - Создаем каталог для программ лабораторной работы №6, переходим в него и создаем файл lab7-1.asm:

```
[didrozdova@fedora arch-pc]$ mkdir ~/work/arch-pc/lab07
[didrozdova@fedora arch-pc]$ cd ~/work/arch-pc/lab07
[didrozdova@fedora lab07]$ touch lab7-1.asm
[didrozdova@fedora lab07]$
```

   - Рассмотрим примеры программ вывода символьных и численных значений.

Пример №1: Изучаем текст программы из листинга 7.1 и помещаем его в файл lab7-1.asm:

```
lab7-1.asm         [-M--]  9 L:[  1+12  13/ 13] *(172 / 172b) <EOF>       [*][X]
%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
```

Создаем исполняемый файл и запускаем его:

```
[didrozdova@fedora lab07]$ nasm -f elf lab7-1.asm
[didrozdova@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[didrozdova@fedora lab07]$ ./lab7-1
```

```
[didrozdova@fedora lab07]$ ./lab7-1
j
```

Результатом выполнения программы является символ ‘j’. Так вышло потому, что код символа 6 равен 00110110(54) в двоичном(десятичном) представлении, а код символа 4 – 00110100(52). Команда add eax,ebx запишет в регистр eax сумму кодов – 01101010(106), которая является кодом символа ‘j’.

Пример №2: Изменим текст программы(из листинга 7.1) и вместо символов ‘4’ и ‘6’, запишем в регистры числа:

```
lab7-1.asm          [BM--]  9 L:[  1+ 7   8/ 13] *(104 / 168b) 0010 0x0
%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
```

Создаем исполняемый файл и запускаем его:

```
[didrozdova@fedora lab07]$ nasm -f elf lab7-1.asm
[didrozdova@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[didrozdova@fedora lab07]$ ./lab7-1


[didrozdova@fedora lab07]$
```

Результатом выполнения данной программы является пустая строка. Действительно, если мы обратимся к ASCII таблице, то заметим, что сумма 6 и 4, равная 10, является кодом, отвечающим за перенос курсора на другую строку.

- Воспользуемся подпрограммами для преобразования ASCII символов в числа и обратно(подпрограммы подключаем из файла in_out.asm). Создаем файл lab7-2.asm в каталоге ~/work/arch-pc/lab07 и вводим в него текст программы из листинга 7.2:

```
[didrozdova@fedora lab07]$ touch ~/work/arch-pc/lab07/lab7-2.asm
[didrozdova@fedora lab07]$ mcedit lab7-2.asm
```

```
lab7-2.asm          [-M--]  9 L:[  1+ 8   9/  9] *(117 / 117b) <EOF>
%include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,'6'
mov ebx,'4'
add eax,ebx
call iprintLF
call quit
```

Создаем исполняемый файл и запускаем его:

```
[didrozdova@fedora lab07]$ nasm -f elf lab7-2.asm
[didrozdova@fedora lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[didrozdova@fedora lab07]$ ./lab7-2
```

```
[didrozdova@fedora lab07]$ ./lab7-2
106
[didrozdova@fedora lab07]$
```

Как и в первом случае, команда add складывает коды символов ‘6’ и ‘4’ (54+52=106). Однако, в отличии от программы из листинга 7.1, функция iprintLF позволяет вывести число, а не символ, кодом которого является это число.

- Аналогично предыдущему примеру изменим символы на числа:

```
lab7-2.asm          [-M--]  9 L:[  1+ 5   6/  9] *(77  / 113b) 0010 0
%include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,6
mov ebx,4
add eax,ebx
call iprintLF
call quit
```

Создаем исполняемый файл и запускаем его:

```
[didrozdova@fedora lab07]$ nasm -f elf lab7-2.asm
[didrozdova@fedora lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[didrozdova@fedora lab07]$ ./lab7-2
10
```

- Теперь заменим функцию iprintLF на iprint. Создаем исполняемый файл и запускаем его:

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

```
[didrozdova@fedora lab07]$ nasm -f elf lab7-2.asm
[didrozdova@fedora lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[didrozdova@fedora lab07]$ ./lab7-2
10[didrozdova@fedora lab07]$
```

Заменив функцию, мы замечаем, что после вывода результата каретка не перемещается на строку ниже.

2. **Выполнение арифметических операций в NASM**

- В качестве примера выполнения арифметических операций в NASM при- ведем программу вычисления арифметического выражения f(x) = (5 * 2 + 3)/3. Создаем файл lab7-3.asm в каталоге ~/work/arch-pc/lab07:

```
[didrozdova@fedora lab07]$ touch ~/work/arch-pc/lab07/lab7-3.asm
[didrozdova@fedora lab07]$ mcedit lab7-3.asm
```

- Внимательно изучаем текст программы из листинга 7.3 и введим в lab7-3.asm:

```
lab7-3.asm          [-M--]  9 L:[  5+21  26/ 26] *(1177/1177b) <EOF>       [*]
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
```

Создаем исполняемый файл и запускаем его:

```
[didrozdova@fedora lab07]$ nasm -f elf lab7-3.asm
[didrozdova@fedora lab07]$ ld -m elf_i386 -o lab7-3 lab7-3.o
[didrozdova@fedora lab07]$ ./lab7-3
Результат: 4
Остаток от деления: 1
```

- Изменяем текст программы для вычисления выражения f(x) = (4 * 6 + 2)/5.

```
lab7-3.asm          [-M--] 10 L:[  5+10  15/ 26] *(405 /1091b) 0010 0x00A  [
SECTION .text
GLOBAL _start
_start:
; ---- Вычисление выражения
mov eax,4 ;
mov ebx,6 ;.
mul ebx ;.
add eax,2 ;
xor edx,edx ; обнуляем EDX для корректной работы div
mov ebx,5 ;.
div ebx ;.
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
```

Создаем исполняемый файл и проверяем его работу:

```
[didrozdova@fedora lab07]$ nasm -f elf lab7-3.asm
[didrozdova@fedora lab07]$ ld -m elf_i386 -o lab7-3 lab7-3.o
[didrozdova@fedora lab07]$ ./lab7-3
Результат: 5
Остаток от деления: 1
```

- 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: (S mod 20) + 1, где S – номер студенческого билета (В данном случае a mod b– это остаток от деления a на b); вывести на экран номер варианта.

Создаем файл variant.asm в каталоге ~/work/arch-pc/lab07, внимательно изучаем листинг 7.4 и вводим текст программы в файл variant.asm:

```
variant.asm        [-M--] 21 L:[  1+ 0   1/ 25] *(21  / 490b) 0010 0x00A  [
%include 'in_out.asm'
SECTION .data
msg: DB 'Введите No 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
```

Создаем исполняемый файл и запускаем его:

```
[didrozdova@fedora lab07]$ touch ~/work/arch-pc/lab07/variant.asm
[didrozdova@fedora lab07]$ mcedit variant.asm
```

```
[didrozdova@fedora lab07]$ ./variant
Введите No студенческого билета:
1132222826
Ваш вариант: 7
```

Проверяем результат работы программы, вычислив номер варианта аналитически(результат верный).

3. **Вопросы**

1. Какие строки листинга 7.4 отвечают за вывод на экран сообщения ‘Ваш вариант:’?

mov eax,rem
call sprint

2. Для чего используется следующие инструкции?

mov ecx, x #запись адреса переменной x в “EAX”
mov edx, 80 #запись длины вводимого сообщения в “EBX”
call sread #вызов подпрограммы ввода сообщения

3. Для чего используется инструкция “call atoi”?

Для преобразования ASCII кода символа в число

4. Какие строки листинга 7.4 отвечают за вычисления варианта?

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”?

В регистр edx.

6. Для чего используется инструкция “inc edx”?

Увеличивает значение регистра ebx на 1

7. Какие строки листинга 7.4 отвечают за вывод на экран результата вычислений?

mov eax,edx call iprintLF

4. **Задание для самостоятельной работы**

В соответствии с вариантом(в моем случае - вариант №7) пишем программу функции: y = 5(x − 1)2

- Создаем файл function.asm в каталоге ~/work/arch-pc/lab07:

```
[didrozdova@fedora lab07]$ touch function.asm
[didrozdova@fedora lab07]$ mcedit function.asm
```

- В файле function.asm пишем программу, вычисляющую значение функции при x1=3 и x2=5:

```
function.asm       [----] 21 L:[  1+ 0   1/ 39] *(21  / 424b) 0010 0x00A  [*][X]
%include 'in_out.asm'

SECTION .data
request: DB 'Введите значение x: ', 0
result: DB 'Результат: ', 0

SECTION .bss
x: RESB 80

SECTION .text
GLOBAL _start
_start:

mov eax, request
call sprintLF

mov ecx, x
mov edx, 80
call sread

mov eax,x
call atoi
```

```
xor edx,edx
sub eax, 1
mov edx, eax
mul edx
mov ebx, 5
mul ebx
mov edx, eax


mov eax, result
call sprint
mov eax,edx
call iprintLF

call quit
```

- Создаем исполняемый файл, запускаем его и вводим с клавиатуры значения переменной x:

```
[didrozdova@fedora lab07]$ nasm -f elf function.asm
[didrozdova@fedora lab07]$ ld -m elf_i386 -o function function.o
[didrozdova@fedora lab07]$ ./function
Введите значение x:
3
Результат: 20
[didrozdova@fedora lab07]$ ./function
Введите значение x:
5
Результат: 80
```

Результат работы программы корректен.

3 Выводы

В ходе выполнения данной лабораторной работы я освоила арифметические инструкции языка ассемблера NASM: сложение, вычитание, умножение, целое и дробное деление, а так же смена знача числа.