

Лабораторная работа №7.Арифметические

Дисциплина Архитектура ЭВМ

Осокин георгий иванович

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
2.1	Складывание строк	6
2.2	Складывание чисел	8
2.3	Программа вывода значение регистра еах	8
2.4	Программа для вычисления выражения $f(x) = (5 * 2 + 3)/3$. .	10
2.5	Изменение программы для вычисления выражения $f(x) = (4 * 6 + 2)/5$	10
2.6	Вычисление варианта по студенческому билеты	11
3	Задания для самостоятельной работы	14
3.1	Написать программу по вычислению $3(x + 10) - 20$	14
4	Выводы	16

Список иллюстраций

2.1	Создание lab-1.asm	6
2.2	Текст в lab-1.asm	7
2.3	Компиляция и запуск кода	7
2.4	Выполнение кода, с замененными строками на числа	8
2.5	Создание файла lab7-2.asm	8
2.6	Текст в lab7-2.asm	9
2.7	Компиляция и запуск	9
2.8	Запуск измененной программы	9
2.9	Выпуск кода с замененным iprintLF на iprint	10
2.10	Создание файла lab7-3.asm	10
2.11	Компиляция и запуск lab7-3.asm	10
2.12	Изменения кода для вычисления другой функции	11
2.13	Вывод измененный программы lab7-3.asm	11
2.14	Объяснение блоков кода	12
2.15	Запуск программы по вычислению варианта	13
3.1	Код, вычисления выражения $3(x+10) - 20$	14
3.2	Вывод вычислений	15

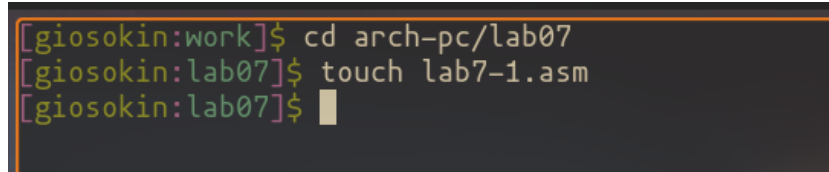
Список таблиц

1 Цель работы

Освоение арифметических инструкций языка ассемблера NASM.

2 Выполнение лабораторной работы

Создадим каталог и перейдем в него. Создадим lab7-1.asm

A terminal window with a dark background and light-colored text. It shows three lines of commands and their execution. The first line is '[giosokin:work]\$ cd arch-pc/lab07', the second is '[giosokin:lab07]\$ touch lab7-1.asm', and the third is '[giosokin:lab07]\$' followed by a cursor. The prompt changes from 'work' to 'lab07' after the first command.

```
[giosokin:work]$ cd arch-pc/lab07
[giosokin:lab07]$ touch lab7-1.asm
[giosokin:lab07]$
```

Рис. 2.1: Создание lab-1.asm

2.1 Складывание строк

Введем текст листинга в файл

```

1 include 'in_out.asm'
2 SECTION .bss
3
4 buf1: RESB 80
5     SECTION .text
6     GLOBAL _start
7
8
9 _start:
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. 2.2: Текст в lab-1.asm

Оттранслируем, слинкуем и запустим

```

[giosokin:lab07]$ f=lab7-1
[giosokin:lab07]$ nasm -f elf ./f.asm && ld -m elf_i386 -o lab7-1 ./f.o && ./f
./lab7-1.asm:1: error: unable to open include file 'in_out.asm': No such file or direc
tory
[giosokin:lab07]$ cp ../lab06/
in_out.asm lab6/ lab61/ lab61copy/ lab62/ lab62copy/
[giosokin:lab07]$ cp ../lab06/in_out.asm .
[giosokin:lab07]$ nasm -f elf ./f.asm && ld -m elf_i386 -o lab7-1 ./f.o && ./f
j

```

Рис. 2.3: Компиляция и запуск кода

Видим, что вывелось “j” потому что мы складываем не числа, а коды строк, и в результате выводим код строки “j”

2.2 Складывание чисел

Заменим строки на числа, скомпилируем и запустим

```
[giosokin:lab07]$ nasm -f elf ./5f.asm && ld -m elf_i386 -o lab7-1 ./5f.o && ./5f
[giosokin:lab07]$
```

Рис. 2.4: Выполнение кода, с замененными строками на числа

Нам вывелся перенос строки, так как мы хоть и сложили числа, но так же вывели не число 10, а символ которому присвоен код 10, то есть перенос строки

2.3 Программа вывода значение регистра eax

Создадим файл lab7-2.asm

```
[giosokin:lab07]$ touch lab7-2.asm
[giosokin:lab07]$ ls
in_out.asm lab7-1 lab7-1.asm lab7-1.asm~ lab7-1.o lab7-2.asm
[giosokin:lab07]$
```

Рис. 2.5: Создание файла lab7-2.asm

Введем текст


```

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

```

Рис. 2.6: Текст в lab7-2.asm

Запустим код

```

[giosokin:lab07]$ f=lab7-2
[giosokin:lab07]$ nasm -f elf ./f.asm && ld -m elf_i386 -o f ./f.o && ./f
106
[giosokin:lab07]$ 

```

Рис. 2.7: Компиляция и запуск

Программа вывела 106, так как мы выводим адрес символа, который мы получаем, суммируя адреса двух строк

Заменяем sprintLF на iprintLF и запустим

```

[giosokin:lab07]$ f=lab7-2
[giosokin:lab07]$ nasm -f elf ./f.asm && ld -m elf_i386 -o f ./f.o && ./f
10
[giosokin:lab07]$ 

```

Рис. 2.8: Запуск измененной программы

Вывелось $6 + 4 = 10$

Заменим `iprintLF` на `iprint`

```
10
[giosokin:lab07]$ nasm -f elf ./f.asm && ld -m elf_i386
10[giosokin:lab07]$
```

Рис. 2.9: Выпуск кода с замененным `iprintLF` на `iprint`

Вывелось 10, без перевода строки

2.4 Программа для вычисления выражения

$$f(x) = (5 * 2 + 3) / 3$$

Создадим файл `lab7-3.asm`

```
[giosokin:lab07]$ touch lab7-3.asm
[giosokin:lab07]$ ls
in_out.asm  lab7-1.asm  lab7-1.o  lab7-2.asm  lab7-2.o
lab7-1      lab7-1.asm~ lab7-2    lab7-2.asm~ lab7-3.asm
[giosokin:lab07]$
```

Рис. 2.10: Создание файла `lab7-3.asm`

Запишем текст в файл и запустим.

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

Рис. 2.11: Компиляция и запуск `lab7-3.asm`

2.5 Изменение программы для вычисления выражения

$$f(x) = (4 * 6 + 2) / 5$$

Заменим числа и исполним

```

6
7 SECTION .text
8 GLOBAL _start
9 _start: ; ---- Вычисление выражения
10 mov eax,4 ; EAX=5
11 mov ebx,6 ; EBX=2
12 mul ebx ; EAX=EAX*EBX
13
14
15 add eax,2 ; EAX=EAX+3
16 xor edx,edx ; обнуляем EDX для корректной работы
17 mov ebx,5 ; EBX=3
18 div ebx ; EAX=EAX/3, EDX=остаток от деления
19
20 mov edi,eax ; запись результата вычисления в '
21
22 ; ---- Вывод результата на экран
23
24 mov eax,div ; вызов подпрограммы печати
25 call sprint ; сообщения 'Результат: '
26 mov eax,edi ; вызов подпрограммы печати знаков

```

Рис. 2.12: Изменения кода для вычисления другой функции

Скомпилируем и запустим

```

[giosokin:lab07]$ nasm -f elf ./5f.asm && ld -m elf_i386 -o 5f ./5f.o && ./5f
Результат: 5
Остаток от деления: 1
[giosokin:lab07]$ 

```

Рис. 2.13: Вывод измененный программы lab7-3.asm

2.6 Вычисление варианта по студенческому билету

Создадим файл variant.asm с текстом листинга 7.4

```

12
13 SECTION .text
14 GLOBAL _start
15 _start:
16     mov eax, msg
17     call sprintf
18
19     mov ecx, x      ; ввод      ; адресс `x` в ecx
20     mov edx, 80     ; текста   ; размер буфера в edx
21     call sread      ; в `x`    ; чтение ввода в `x`
22                   ; из консоли
23
24     mov eax, x      ; вызов подпрограммы преобразования
25     call atoi       ; ASCII кода в число, `eax=x`
26
27     ; преобразование строки в число
28     xor edx, edx    ; очищение edx
29     mov ebx, 20     ; 20 в ebx
30     div ebx         ; делим значение из eax на 20
31     inc edx         ; увеличиваем остаток от деления на 1
32
33     mov eax, rem     ; Запись адреса строки 'Ваш вариант' в eax
34     call sprint      ; Вызов вывода строки без переноса строки
35
36     mov eax, edx     ; перенос вычисленного остатка в eax
37     call iprintfLF   ; Вывод вычисленного остатка с переносом строки
38
39     call quit        ; вывод

```

Рис. 2.14: Объяснение блоков кода

1. Для вывода на экран “Ваш вариант” отвечают строки 31-32 (выделены красным)
2. инструкции на строка 19-21 (выделены голубым) используется для ввода текста в x из консоли
3. `call atoi` используется для преобразования ASCII кода в число
4. За вычисление варианта отвечают строки 26-29
5. Остаток от деления записывается в `edx/dx`
6. `inc edx` увеличивает значение `edx` на 1
7. За вывод результата вычислений отвечают строки 33-34 (выделенно желтым)

Запустим программу что бы узнать номер варианта

```
[giosokin:lab07]$ f=variant
[giosokin:lab07]$ nasm -f elf ./f.asm && ld -m elf_i386 -o f ./f.o && ./f
Введите №студенческого билета:
1132226517
Ваш вариант: 18
[giosokin:lab07]$
```

Рис. 2.15: Запуск программы по вычислению варианта

Наш вариант 18

3 Задания для самостоятельной работы

3.1 Написать программу по вычислению $3(x + 10) - 20$

Напишем, код, который будет вычислять.

Изменим размер буфера с 80 на 10, так как для чисел 80 байт многовато.

Так как введенное значение уже в еах просто добавим к нему 10, домножим на 3 и отнимем 20

```
3 ;-----
4 %include 'in_out.asm'
5
6 SECTION .data
7 msg: DB 'Введите X: ',0
8 rem: DB 'Ответ: ',0
9
10 SECTION .bss
11 x: RESB 10
12
13 SECTION .text
14 GLOBAL _start
15 _start:
16     mov eax, msg
17     call sprintf
18
19     mov ecx, x          ; адресс `x` в есх
20     mov edx, 10         ; размер буфера в edx
21     call sread          ; чтение ввода в `x`
22
23     mov eax, x ; вызов подпрограммы преобразования
24     call atoi ; ASCII кода в число, `eax=x`
25
26     add eax, 10         ; Прибавим 10 к еах
27     mov cx, 3           ; 3 в ECX
28     mul cx              ; Умножим еах на три (ввод + 10)*3
29     sub eax, 20
30     mov edx, eax
31     xor eax, eax
32
33     mov eax, rem        ; Запись адреса строки 'Ваш вариант' в еах
34     call sprintf        ; Вызов вывода строки без переноса строки
35     mov eax, edx         ; перенос вычисленного остатка в еах
36     call iprintLF       ; Вывод вычисленного остатка с переносом строки
37
38     call quit
```

Рис. 3.1: Код, вычисления выражения $3(x+10) - 20$

Запустим и убедимся в верности выполненных вычислений

```
[giosokin:lab07]$ nasm -f elf ./f.asm && ld -m elf_i386 -o f ./f.o && ./f
Введите X:
1
Ответ: 13
[giosokin:lab07]$ ./function
Введите X:
5
Ответ: 25
[giosokin:lab07]$
```

Рис. 3.2: Вывод вычислений

4 Выводы

Мы освоили арифметические операции в NASM и написали программу, которое вычисляет значение функции от x