Отчет по лабораторной работе №6.

Арифметические операции в NASM

Зайцева П. Е.

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Освоение арифметических инструкций языка ассемблера NASM.

2 Теоретическое введение

Существует три основных способа адресации: • Регистровая адресация – операнды хранятся в регистрах и в команде используются имена этих регистров, например: mov ax,bx. • Непосредственная адресация – значение операнда задается непосредственно в ко- манде, Например: mov ax,2. • Адресация памяти – операнд задает адрес в памяти. В команде указывается символи- ческое обозначение ячейки памяти, над содержимым которой требуется выполнить операцию.

3 Выполнение лабораторной работы

Создала каталог для программ по лабораторных работ №6, перешла в него и создала файл lab6-1.asm

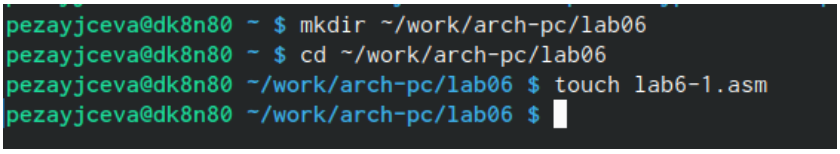

Рис. 3.1: рис.1

Ввела в файл lab6-1.asm текст программы из листинга 6.1. Далее создала исполняемый файл и запустила его.

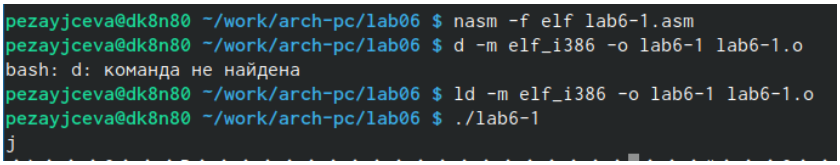

Рис. 3.2: рис.2

В данном случае при выводе значения регистра eax мы ожидаем увидеть число 10. Однако результатом будет символ j.

После изменила текст программы и вместо символов, записала в регистры числа.

Рис. 3.3: рис.3

Создала исполняемый файл и запустила его.

Создала файл lab6-2.asm в каталоге ~/work/arch-pc/lab06 .

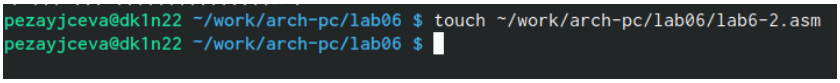

Рис. 3.4: рис.5

Ввела в него текст программы из листинга 6.2.

Рис. 3.5: рис.6

Создала исполняемый файл и запустила его.

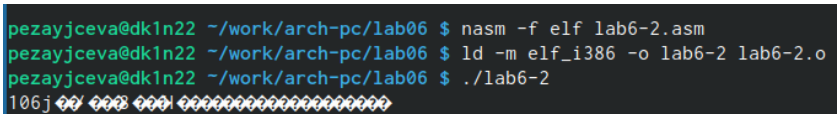

Рис. 3.6: рис.7

В результате работы программы получила число 106.

Аналогично предыдущему примеру изменила символы на числа.

Создала исполняемый файл и запустила его. Результат 10.

```
pezayjceva@dk1n22 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
pezayjceva@dk1n22 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
pezayjceva@dk1n22 ~/work/arch-pc/lab06 $ ./lab6-2
10
```

Рис. 3.7: рис.8

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

```
pezayjceva@dk1n22 ~/work/arch-pc/lab06 $ touch ~/work/arch-pc/lab06/lab6-3.asm
pezayjceva@dk1n22 ~/work/arch-pc/lab06 $
```

Рис. 3.8: рис.9

Из листинга 6.3 ввела текст в lab6-3.asm

Создала исполняемый файл и запустила его.

Далее изменила текст программы под выражение ⊠(⊠) = (4 ⊠ 6 + 2)/5 и заупстила.

```
 GNU nano 7.2  /afs/.dk.sci.pfu.edu.ru/home/p/e/pezayjceva/work/arch-pc/lab06/lab6-3.asm   И
;-------------------------------
%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

^G Справка    ^O Записать   ^W Поиск      ^K Вырезать   ^T Выполнить  M-U Отмена    M-A Устан
^X Выход      ^R ЧитФайл    ^\ Замена     ^U Вставить   ^C Позиция    M-E Повтор    M-6 Копир
```

Рис. 3.9: рис.10

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

Запустила файл и получила свой номер варианта.

Рис. 3.10: рис.15

1. За вывод сообщения “Ваш вариант” отвечают строки кода:

mov eax,rem call sprint

2. Инструкция mov ecx, x используется, чтобы положить адрес вводимой строки x в регистр ecx mov edx, 80 - запись в регистр edx длины вводимой строки call sread - вызов подпрограммы из внешнего файла, обеспечивающей ввод сообщения с клавиатуры

3. call atoi используется для вызова подпрограммы из внешнего файла, которая преобразует ascii-код символа в целое число и записывает результат в регистр eax

4. За вычисления варианта отвечают строки:

xor edx,edx ; обнуление edx для корректной работы div mov ebx,20 ; ebx = 20 div ebx ; eax = eax/20, edx - остаток от деления inc edx ; edx = edx + 1

5. При выполнении инструкции div ebx остаток от деления записывается в регистр edx

6. Инструкция inc edx увеличивает значение регистра edx на 1

7. За вывод на экран результатов вычислений отвечают строки:

mov eax,edx call iprintLF

Написать программу вычисления выражения 18(☒ + 1)/6

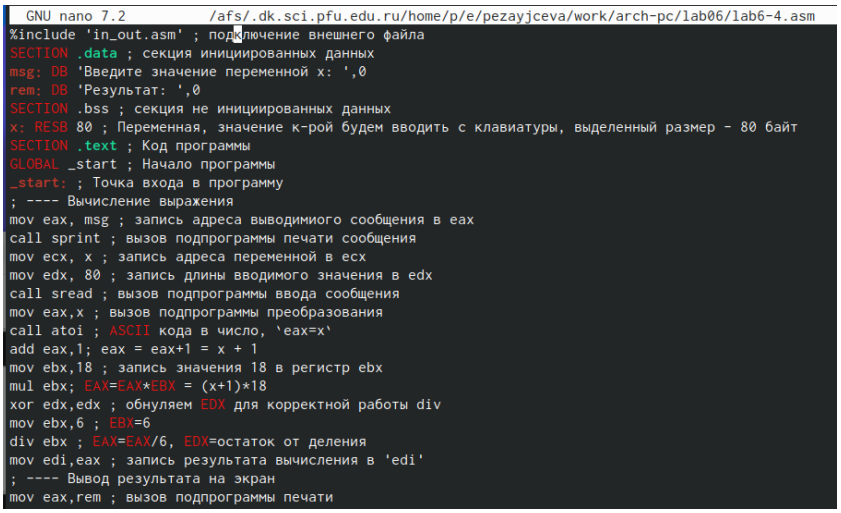

Рис. 3.11: рис.12

Вывод при x=3

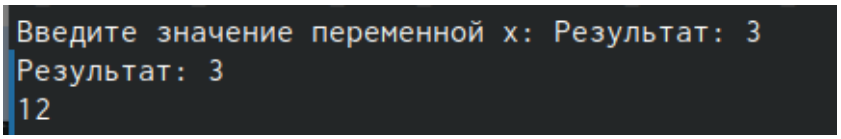

Рис. 3.12: рис.13

Вывод при x=1

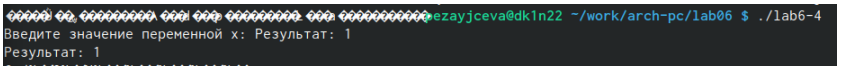

Рис. 3.13: рис.14

Код для уравнеия 17.

https://github.com/yuillee/study_2023_2024_arh-pc/tree/master/labs/lab06/report%include ‘in_out.asm’ ; подключение внешнего> SECTION .data ; секция инициированных данных msg: DB ‘Введите значение переменной x:’,0 rem: DB ‘Результат:’,0

SECTION .bss ; секция не инициированных данных x: RESB 80 ; Переменная, значение к-рой будем вводить с клавиатуры, выделенный размер - 80 байт SECTION .text ; Код программы

GLOBAL _start ; Начало программы _start: ; Точка входа в программу ; —- Вычисление выражения

mov eax, msg ; запись адреса выводимиого сообщения в eax call sprint ; вызов подпрограммы печати сообщения mov ecx, x ; запись адреса переменной в ecx mov edx, 80 ; запись длины вводимого значения в edx call sread ; вызов подпрограммы ввода сообщения mov eax,x ; вызов подпрограммы преобразования call atoi ; ASCII кода в число, `eax=x` add eax,1; eax = eax+1 = x + 1 mov ebx,18 ; запись значения 18 в регистр ebx mul ebx; EAX=EAX*EBX = (x+1)*18 xor edx,edx ; обнуляем EDX для корректной работы div mov ebx,6 ; EBX=6 div ebx ; EAX=EAX/6, EDX=остаток от деления mov edi,eax ; запись результата вычисления в ‘edi’ ; —- Вывод результата на экран mov eax,rem ; вызов подпрограммы печати call sprint ; сообщения ‘Результат:’ mov eax,edi ; вызов подпрограммы печати значения call iprint ; из ‘edi’ в виде символов call quit ; вызов подпрограммы завершения

4 Выводы

Освоила арифметические инструкции языка ассемблера NASM.

Список литературы