

Отчёт по лабораторной работе №8

**Дисциплина: Компьютерные технологии и технологии
программирования**

ДЫМОВОЙ Д.Д.

Содержание

1	Цель работы	5
2	Теоретическое введение	6
3	Выполнение лабораторной работы	7
4	Задание для самостоятельной работы	11
5	Выводы	13
	Список литературы	14

Список иллюстраций

3.1	Трансляция и компоновка	7
3.2	Трансляция и компоновка	7
3.3	Вывод значений	8
3.4	Трансляция компоновка и вывод	9
3.5	Трансляция, компоновка, запуск	9
3.6	Трансляция, компоновка, запуск	9
3.7	Программа	10
4.1	Программа	11
4.2	Проверка работы программы	12

Список таблиц

1 Цель работы

Приобретение навыков написания программ с использованием циклов и обработкой аргументов командной строки.

2 Теоретическое введение

Стек — это структура данных, организованная по принципу LIFO («Last In — First Out» или «последним пришёл — первым ушёл»). Стек является частью архитектуры процессора и реализован на аппаратном уровне. Для работы со стеком в процессоре есть специальные регистры (ss, bp, sp) и команды.

push -10 ; Поместить -10 в стек push ebx ; Поместить значение регистра ebx в стек push [buf] ; Поместить значение переменной buf в стек push word [ax] ; Поместить в стек слово по адресу в ax

Существует ещё две команды для добавления значений в стек. Это команда pusha, которая помещает в стек содержимое всех регистров общего назначения в следующем порядке: ax, cx, dx, bx, sp, bp, si, di. А также команда pushf, которая служит для перемещения в стек содержимого регистра флагов. Обе эти команды не имеют операндов.

pop eax ; Поместить значение из стека в регистр eax pop [buf] ; Поместить значение из стека в buf pop word[si] ; Поместить значение из стека в слово по адресу в si.

3 Выполнение лабораторной работы

Я создаю каталог lab08 в директории ~/work/arch-pc/. Ввожу текст программы листинга, создаю исполняемый файл и запускаю его. Этот цикл выводит числа от 5 до 0 (рис. 3.1).

```
dddihmova@dk5n52 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
dddihmova@dk5n52 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
dddihmova@dk5n52 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 5
5
4
3
2
1
```

Рис. 3.1: Трансляция и компоновка

Вношу изменения согласно условиям лабораторной работы, создаю исполняемый файл и запускаю его (рис. 3.2).

```
dddihmova@dk5n52 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
dddihmova@dk5n52 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
dddihmova@dk5n52 ~/work/arch-pc/lab08 $ ./lab8-1
```

Рис. 3.2: Трансляция и компоновка

Фрагмент программы выводимой при вводе с клавиатуры N=5 (рис. 3.3).

```
4294797787
4294797785
4294797783
4294797781
4294797779
4294797777
4294797775
4294797773
4294797771
4294797769
4294797767
4294797765
4294797763
4294797761
4294797759
4294797757
4294797755
4294797753
4294797751
4294797749
4294797747
4294797745
4294797743
4294797741
4294797739
4294797737
4294797735
4294797733
4294797731
4294797729
4294797727
4294797725
4294797723
4294797721
4294797719
4294797717
4294797715
4294797713
4294797711
```

Рис. 3.3: Вывод значений

Значения регистр `esx` принимает `[N]`, введенные с клавиатуры, но число проходов цикла не заканчивается.

Снова вношу изменения согласно условиям, создаю исполняемый файл и запускаю его (рис. 3.4).


```

dddihmova@dk5n52 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
dddihmova@dk5n52 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
dddihmova@dk5n52 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 5
4
3
2
1
0

```

Рис. 3.4: Трансляция компоновка и вывод

Да, в этом случае соответствует.

Создаю файл в каталоге, ввожу в него текст программы листинга 8.2, создаю исполняемый файл и запускаю его, указав аргументы (рис. 3.5).

```

dddihmova@dk5n52 ~/work/arch-pc/lab08 $ nasm -f elf lab8-2.asm
dddihmova@dk5n52 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-2 lab8-2.o
dddihmova@dk5n52 ~/work/arch-pc/lab08 $ ./lab8-2 12 13 7 10 5
Результат: 47

```

Рис. 3.5: Трансляция, компоновка, запуск

Изменяю программу, чтобы вместо суммы выводилось произведение аргументов (рис. 3.6).

```

dddihmova@dk8n68 ~/work/arch-pc/lab08 $ nasm -f elf lab8-2.asm
dddihmova@dk8n68 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-2 lab8-2.o
dddihmova@dk8n68 ~/work/arch-pc/lab08 $ ./lab8-2 2 3
Результат: 6

```

Рис. 3.6: Трансляция, компоновка, запуск

add меняю на mul и сохраняю промежуточные суммы (mov esi, eax). А также изначально задаю esi значение 1, чтобы при умножении аргументов не умножалось на 0.

```

lab8-2.asm      [----] 29 L:[ 1+23 24/ 31] *(1086/1435
#include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
SECTION .text
global _start
_start:
pop ecx ; Извлекаем из стека в 'ecx' количество
; аргументов (первое значение в стеке)
pop edx ; Извлекаем из стека в 'edx' имя программы
; (второе значение в стеке)
sub ecx,1 ; Уменьшаем 'ecx' на 1 (количество
; аргументов без названия программы)
mov esi, 1 ; Используем 'esi' для хранения
; промежуточных сумм
next:
cmp ecx,0h ; проверяем, есть ли еще аргументы
jz _end ; если аргументов нет выходим из цикла
; (переход на метку '_end')
pop eax ; иначе извлекаем следующий аргумент из стека
call atoi ; преобразуем символ в число

mul esi ; добавляем к промежуточной сумме
mov esi,eax
; след. аргумент 'esi=esi*eax'
loop next ; переход к обработке следующего аргумента
_end:
mov eax, msg ; вывод сообщения "Результат: "
call sprint
mov eax, esi ; записываем сумму в регистр 'eax'
call iprintLF ; печать результата
call quit ; завершение программы

```

Рис. 3.7: Программа

4 Задание для самостоятельной работы

Создаю файл `zadanie.asm` в каталоге `~/work/arch-рс/lab08`, пишу программу согласно условиям задачи и функции из 13 варианта (рис. 4.1).

```
zadanie.asm [----] 28 L: [ 1+10 11/ 39] *(182 / 845b) 0010 0x00A
#include "stdafx.h"
SECTION .data
msg1 db "Функция:12х-7 ",0
msg2 db "Результат: ",0
SECTION .text
global _start

_start:

mov eax,msg1 ; вывод функции
call sprint

pop ecx
pop edx
sub ecx,1
mov esi,0

next:
cmp ecx,0h; сравнение, если регистр не ноль то переход к метке _end
jz _end
pop eax
call atoi

mov ebx, 12; выполняется функция, умножение на 12 и сложение с -7
mul ebx
mov ebx, -7
add eax,ebx

add esi, eax; результат записываю в esi

loop next

_end:
mov eax, msg2 ; вывод сообщения "Результат: "
call sprint
mov ecx, esi ; записываем сумму в регистр 'ecx'
call iprintfLF ; печать результата
call quit ; завершение программы
```

Рис. 4.1: Программа

Создаю исполняемый файл и запускаю его (рис. 4.2).

```
ddihmova@dk8n68 ~/work/arch-pc/lab08 $ nasm -f elf zadanie.asm
ddihmova@dk8n68 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o zadanie zadanie.o
ddihmova@dk8n68 ~/work/arch-pc/lab08 $ ./zadanie 1 2 3 4
Функция:12x-7 Результат: 92
```

Рис. 4.2: Проверка работы программы

5 Выводы

Я приобрела навыки написания программ с использованием циклов и обработкой аргументов командной строки.

Список литературы