Отчёт по лабораторной работе №8

Программирование цикла. Обработка аргументов командной строки.

Корчагин Алексей Павлоаич

Содержание

1	Цель работы	5
2	Теоретическое введение	6
3	Выполнение лабораторной работы	7
4	Выполненние самостоятельной работы	15
5	Выводы	19

Список иллюстраций

3.1	Созданние каталога	7
3.2	Редактирование файла	8
3.3	Запсук исполняймого файла	8
3.4	Редактирование файла	9
3.5	Редактирование файла	10
3.6	Запуск исполняймого файла	10
3.7	Созданние файла	11
3.8	Редактированние файла	11
3.9	, t ,	11
3.10	Созданние файла	12
3.11	Редактирование файла	12
	· · · · · · · · · · · · · · · · · · ·	13
3.13		13
3.14		14
4.1	Созданние файла	15
4.2		16

Список таблиц

1 Цель работы

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

2 Теоретическое введение

##Стек — абстрактный тип данных, представляющий собой список элементов, организованных по принципу LIFO(Первым вошёл последним вышел). Стек является частью архитектуры процессора и реализован на аппаратном уровне. Для работы со стеком в процессоре есть специальные регистры (ss, bp, sp) и команды. Основая функция стека - сохранение адресов возврата и передачи аргументов при вызове процедур. Кроме того, в нём выделяется память для локальных переменных и могут временно храниться значения регистров. Стек имеет вершину (адрес последнего добавленного элемента, который хранится в регистре еsp (указатель стека)) и дно (противоположный конец стека). Значение, помещённое в стек последним, извлекается первым. При помещении значения в стек указатель стека уменьшается, а при извлечении — увеличивается. Для стека существует две основные операции:

```
добавление элемента в вершину стека (push); извлечение элемента из вершины стека (pop).
```

3 Выполнение лабораторной работы

Создал каталог для программ требуемых для выполненния №8 и перешёл в него и создал файл lab8-1.asm(рис. 3.1).

```
apkorchagin@dk5n59 ~ $ mkdir ~/work/arch-pc/lab08
apkorchagin@dk5n59 ~ $ cd ~/work/arch-pc/lab08
apkorchagin@dk5n59 ~/work/arch-pc/lab08 $ touch lab8-1.asm
apkorchagin@dk5n59 ~/work/arch-pc/lab08 $
```

Рис. 3.1: Созданние каталога

Ввёл в файл lab8-1.asm текст программы из листинга (рис. 3.2).

```
1 %include 'in_out.asm'
 2 SECTION .data
 3 msg1 db 'Введите N: ',0h
 4 SECTION .bss
 5 N: resb 10
 6 SECTION .text
 7 global _start
 8 _start:
9; ---- Вывод сообщения 'Введите N: '
10 mov eax, msg1
11 call sprint
12 : ---- Ввод 'N'
13 mov ecx, N
14 mov edx, 10
15 call sread
16 ; ---- Преобразование 'N' из символа в число
17 mov eax, N
18 call atoi
19 mov [N], eax
20; ----- Организация цикла
21 mov ecx,[N]; Счетчик цикла, 'ecx=N'
22 label:
23 mov [N],ecx
24 mov eax,[N]
25 call iprintLF ; Вывод значения 'N'
26 loop label ; 'ecx=ecx-1' и если 'ecx' не '0'
27; переход на 'label'
28 call quit
```

Рис. 3.2: Редактирование файла

Проверил работу файла(рис. 3.3).

```
apkorchagin@dk5n59 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm apkorchagin@dk5n59 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o apkorchagin@dk5n59 ~/work/arch-pc/lab08 $ ./lab8-1 Введите N: 2
2
1 apkorchagin@dk5n59 ~/work/arch-pc/lab08 $ []
```

Рис. 3.3: Запсук исполняймого файла

Изменил текст программы, добавил значения регистра есх в цикле(рис. 3.4).

```
1 %include 'in_out.asm'
 2 SECTION .data
 3 msg1 db 'Введите N: ',0h
 4 SECTION .bss
 5 N: resb 10
 6 SECTION .text
 7 global _start
8 _start:
9; ---- Вывод сообщения 'Введите N: '
10 mov eax, msg1
11 call sprint
12; ---- Ввод 'N'
13 mov ecx, N
14 mov edx, 10
15 call sread
16 ; ---- Преобразование 'N' из символа в число
17 mov eax, N
18 call atoi
19 mov [N], eax
20 ; ---- Организация цикла
21 mov ecx,[N] ; Счетчик цикла, 'ecx=N'
22 label:
23 sub ecx,1; 'ecx=ecx-1'
24 mov [N],ecx
25 mov eax,[N]
26 call iprintLF
27 loop label
```

Рис. 3.4: Редактирование файла

Проверил работу файла, программа выводит значенния через еденицу, получается не n значенний а n/2(рис. ??).

[Запуск исполняймого файла](image/Снимок экрана от 2023-12-08 15-12-21.png{#fig:005 width=70%}

Изменил текскт программы(рис. 3.5).

```
1 %include 'in_out.asm'
 2 SECTION .data
 3 msg1 db 'Введите N: ',0h
 4 SECTION .bss
 5 N: resb 10
 6 SECTION .text
 7 global _start
 8 _start:
9; ---- Вывод сообщения 'Введите N: '
10 mov eax, msg1
11 call sprint
12 ; ---- Ввод 'N'
13 mov ecx, N
14 mov edx, 10
15 call sread
16; ---- Преобразование 'N' из символа в число
17 mov eax, N
18 call atoi
19 mov [N], eax
20; ---- Организация цикла
21 mov ecx,[N] ; Счетчик цикла, 'ecx=N'
22 label:
23 push есх ; добавление значения есх в стек
24 sub ecx,1
25 mov [N],ecx
26 mov eax,[N]
27 call iprintLF
28 рор есх ; извлечение значения есх из стека
29 loop label
```

Рис. 3.5: Редактирование файла

Запустил программу, теперь она работает коректно(рис. 3.6).

```
apkorchagin@dk5n59 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm apkorchagin@dk5n59 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o apkorchagin@dk5n59 ~/work/arch-pc/lab08 $ ./lab8-1 Введите N: 6
5
4
3
2
1
0
```

Рис. 3.6: Запуск исполняймого файла

Создал файл lab8-2.asm(рис. 3.7).

```
apkorchagin@dk5n59 ~/work/arch-pc/lab08 $ touch lab8-2.asm apkorchagin@dk5n59 ~/work/arch-pc/lab08 $ _
```

Рис. 3.7: Созданние файла

Ввёл текст программы в файл lab8-2.asm(рис. 3.8).

```
1 %include 'in_out.asm'
 2 SECTION .text
 3 global _start
 4 _start:
 5 рор есх ; Извлекаем из стека в 'есх' количество
 6; аргументов (первое значение в стеке)
 7 pop edx ; Извлекаем из стека в 'edx' имя программы
8; (второе значение в стеке)
9 sub ecx, 1 ; Уменьшаем 'ecx' на 1 (количество
10; аргументов без названия программы)
11 next:
12 стр есх, ∅ ; проверяем, есть ли еще аргументы
13 jz _end ; если аргументов нет выходим из цикла
14; (переход на метку '_end')
15 рор еах ; иначе извлекаем аргумент из стека
16 call sprintLF ; вызываем функцию печати
17 loop next ; переход к обработке следующего
18; аргумента (переход на метку 'next')
19 _end:
20 call quit
```

Рис. 3.8: Редактированние файла

Запустил файл, указав аргументы(рис. 3.9).

```
apkorchagin@dk5n59 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-2 lab8-2.o apkorchagin@dk5n59 ~/work/arch-pc/lab08 $ ./lab8-2 apryмент 2 'apryмент 3' apryмент 3 apryмент 3 apryмент 3 apkorchagin@dk5n59 ~/work/arch-pc/lab08 $ []
```

Рис. 3.9: Созданние и запуск исполняймого файал

Создал файл lab8-3.asm(рис. 3.10).

```
apkorchagin@dk5n59 \sim/work/arch-pc/lab08 $ touch lab8-3.asm apkorchagin@dk5n59 \sim/work/arch-pc/lab08 $
```

Рис. 3.10: Созданние файла

Ввёл текст программф в файл lab8-3.asm(рис. 3.11).

```
1 %include 'in_out.asm'
 2 SECTION .data
 3 msg db "Результат: ",∅
 4 SECTION .text
 5 global _start
 6 _start:
 7 рор есх ; Извлекаем из стека в 'есх' количество
 8; аргументов (первое значение в стеке)
9 рор edx ; Извлекаем из стека в 'edx' имя программы
10; (второе значение в стеке)
11 sub ecx,1; Уменьшаем 'ecx' на 1 (количество
12: аргументов без названия программы)
13 mov esi, ∅ ; Используем 'esi' для хранения
14; промежуточных сумм
15 next:
16 стр есх,0h; проверяем, есть ли еще аргументы
17 jz _end ; если аргументов нет выходим из цикла
18; (переход на метку '_end')
19 рор еах ; иначе извлекаем следующий аргумент из стека
20 call atoi ; преобразуем символ в число
21 add esi,eax ; добавляем к промежуточной сумме
22; след. apгумент 'esi=esi+eax'
23 loop next ; переход к обработке следующего аргумента
24 _end:
25 mov eax, msg ; вывод сообщения "Результат: "
26 call sprint
27 mov eax, esi ; записываем сумму в регистр 'eax'
28 call iprintLF; печать результата
29 call quit ; завершение программы
```

Рис. 3.11: Редактирование файла

Создал и запустил исполняймый файл ввёл числа от 1 до 5, получил 15(рис. 3.12).

```
apkorchagin@dk5n59 ~/work/arch-pc/lab08 $ nasm -f elf lab8-3.asm apkorchagin@dk5n59 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-3 lab8-3.o apkorchagin@dk5n59 ~/work/arch-pc/lab08 $ ./lab8-3 1 2 3 4 5 Результат: 15 apkorchagin@dk5n59 ~/work/arch-pc/lab08 $ [
```

Рис. 3.12: Созданние и работа исполняймого файла

Изменил код в файле так чтбоы программа выводила не сумму, а произведенние (рис. 3.13).

```
1 %include 'in_out.asm'
 2 SECTION .data
3 msg db "Результат: "
 4 SECTION .text
 5 global _start
 6 _start:
 7 pop ecx
 8 pop edx
9 sub ecx,1
10 mov esi, 1
11 next:
12 cmp ecx,0
13 jz _end
14 pop eax
15 call atoi ; преобразуем символ в число
16 mul esi
17 mov esi,eax
18 loop next ; переход к обработке следующего аргумента
19 _end:
20 mov eax, msg ; вывод сообщения "Результат: "
21 call sprint
22 mov eax, esi
23 call iprintLF ; печать результата
24 call quit ; завершение программы
```

Рис. 3.13: Редактирование файла

Создал исполняймый файл и запустил программу, ввё числа от 1 до 5, Программа посчитала произвиденние коректно(рис. 3.14).

```
apkorchagin@dk5n59 ~/work/arch-pc/lab08 $ nasm -f elf lab8-3.asm apkorchagin@dk5n59 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-3 lab8-3.o apkorchagin@dk5n59 ~/work/arch-pc/lab08 $ ./lab8-3 1 2 3 4 5 Результат: 120 apkorchagin@dk5n59 ~/work/arch-pc/lab08 $
```

Рис. 3.14: Создание Исполняймого файла

4 Выполненние самостоятельной работы

Создал файл lab8-4.asm(рис. 4.1).

```
apkorchagin@dk5n59 ~/work/arch-pc/lab08 $ touch lab8-4.asm
apkorchagin@dk5n59 ~/work/arch-pc/lab08 $ []
```

Рис. 4.1: Созданние файла

Ввёл в созданный файл текст программы, программа находит сумму значений функции f(x)=3x-1 (2 Вариант) для всех введённых пользовтелем аргументов x. (рис. 4.2).

```
1 %include 'in_out.asm'
 2 SECTION .data
 3 msg db "Результат: ", ∅
 4 msg1 db "Функция: f(x)=3x-1"
5 SECTION .text
 6 global _start
7 _start:
8 рор есх ; Извлеченние из стека в 'есх' кол-во
9; аргументов
10 pop edx ; Извлеченние из стека в 'edx' имя программы
11 sub ecx,1; Уменьшенние 'ecx' на 1 (кол-во
12; аргументов без названия программы)
13 mov esi, ∅ ; Используем 'esi' для хранения
14; промежуточных сумм
15 next:
16 cmp ecx,0h ; проверка, есть ли еще аргументы
17 jz _end ; если аргументов нет выходим из цикла
18; (переход на метку '_end')
19 рор еах ; иначе берём следующий аргумент из стека
20 call atoi ; преобразовывание символа в число
21 mov ebx, 3 ; ebx=3
22 mul ebx; eax=eax*ebx
23 sub eax,1 ; eax-1
24 add esi,eax ; добавленние к промежуточной сумме
25 ; след. apryмeнт 'esi=esi+eax'
26 loop next ; переход к обработке следующего аргумента
27 _end:
28 mov eax, msg1;
29 call sprintLF;
30 mov eax, msg ; вывод сообщения "Результат: "
31 call sprint
32 mov eax, esi ; записываем сумму в регистр 'eax'
33 call iprintLF ; вывод результата
34 call quit ; завершение программы
35
```

Рис. 4.2: Редактированние файла

Создал исполняймый файл и проверил коректность работы программы(рис.

```
??).
```

```
apkorchagin@dk5n59 ~/work/arch-pc/lab08 $ nasm -f elf lab8-4.asm apkorchagin@dk5n59 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-4 lab8-4.o apkorchagin@dk5n59 ~/work/arch-pc/lab08 $ ./lab8-4 1 2 3 4 Функция: f(x)=3x-1 Результат: 26 apkorchagin@dk5n59 ~/work/arch-pc/lab08 $ \square
```

Код программый

%include 'in_out.asm'

```
SECTION .data
msq db "Результат: ",0
msg1 db "Функция: f(x)=3x-1"
SECTION .text
global _start
_start:
рор есх ; Извлеченние из стека в `есх` кол-во
; аргументов
pop edx ; Извлеченние из стека в `edx` имя программы
sub ecx,1 ; Уменьшенние `ecx` на 1 (кол-во
; аргументов без названия программы)
mov esi, 0 ; Используем `esi` для хранения
; промежуточных сумм
next:
стр есх,0h; проверка, есть ли еще аргументы
jz _end ; если аргументов нет выходим из цикла
; (переход на метку `_end`)
рор еах ; иначе берём следующий аргумент из стека
call atoi ; преобразовывание символа в число
mov ebx,3; ebx=3
mul ebx; eax=eax*ebx
sub eax,1 ; eax-1
add esi,eax ; добавленние к промежуточной сумме
; след. apryмент `esi=esi+eax`
loop next ; переход к обработке следующего аргумента
end:
mov eax,msq1 ;
call sprintLF ;
mov eax, msg ; вывод сообщения "Результат: "
```

```
call sprint
mov eax, esi ; записываем сумму в регистр `eax`
call iprintLF ; вывод результата
call quit ; завершение программы
```

5 Выводы

В ходе выполениния работы,я получил навыки по организации циклов, и опыт работы со стеком на языке NASM.