Отчёт по лабораторной работе №8

Программирование цикла. Обработка аргументов командной строки.

Корчагин Алексей Павлоаич

Содержание

Список иллюстраций

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 ре- гистре esp (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).

```
%include 'in_out.asm'
SECTION .data
msg1 db 'Введите N: ',0h
SECTION .bss
N: resb 10
SECTION .text
global _start
_start:
; ----- Вывод сообщения 'Введите N: '
mov eax,msg1
call sprint
; ----- Ввод 'N'
mov ecx, N
mov edx, 10
call sread
; ----- Преобразование 'N' из символа в число
mov eax,N
call atoi
mov [N],eax
; ------ Организация цикла
mov ecx,[N] ; Счетчик цикла, `ecx=N`
label:
mov [N],ecx
mov eax,[N]
call iprintLF ; Вывод значения `N`
loop label ; `ecx=ecx-1` и если `ecx` не '0'
; переход на `label`
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 ecx в цикле(рис. 3.4).

```
%include 'in_out.asm'
SECTION .data
msg1 db 'Введите N: ',0h
SECTION .bss
N: resb 10
SECTION .text
global _start
_start:
; ----- Вывод сообщения 'Введите N: '
mov eax,msg1
call sprint
; ----- Ввод 'N'
mov ecx, N
mov edx, 10
call sread
; ----- Преобразование 'N' из символа в число
mov eax,N
call atoi
mov [N],eax
; ------ Организация цикла
mov ecx,[N] ; Счетчик цикла, `ecx=N`
label:
sub ecx,1 ; `ecx=ecx-1`
mov [N],ecx
mov eax,[N]
call iprintLF
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 тексkт программы(рис. 3.5).

```
%include 'in_out.asm'
SECTION .data
msg1 db 'Введите N: ',0h
SECTION .bss
N: resb 10
SECTION .text
global _start
_start:
; ----- Вывод сообщения 'Введите N: '
mov eax,msg1
call sprint
; ----- Ввод 'N'
mov ecx, N
mov edx, 10
call sread
; ----- Преобразование 'N' из символа в число
mov eax,N
call atoi
mov [N],eax
; ------ Организация цикла
mov ecx,[N] ; Счетчик цикла, `ecx=N`
label:
push ecx ; добавление значения ecx в стек
sub ecx,1
mov [N],ecx
mov eax,[N]
call iprintLF
pop ecx ; извлечение значения ecx из стека
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).

```
%include 'in_out.asm'
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)
next:
cmp ecx, 0 ; проверяем, есть ли еще аргументы
jz _end ; если аргументов нет выходим из цикла
; (переход на метку `_end`)
pop eax ; иначе извлекаем аргумент из стека
call sprintLF ; вызываем функцию печати
loop next ; переход к обработке следующего
; аргумента (переход на метку `next`)
_end:
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
apkorchagin@dk5n59 ~/work/arch-pc/lab08 $ ./lab8-2 аргумент1 аргумент 2 'аргумент 3'
аргумент1
аргумент
2
аргумент 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 ~/work/arch-pc/lab08 $ touch lab8-3.asm
apkorchagin@dk5n59 ~/work/arch-pc/lab08 $
```

Рис. 3.10: Созданние файла

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

```
%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, 0 ; 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
add esi,eax ; добавляем к промежуточной сумме
; 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. 3.11: 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).

```
%include 'in_out.asm'
SECTION .data
msg db "Результат: "
SECTION .text
global _start
_start:
pop ecx
pop edx
sub ecx,1
mov esi, 1
next:
cmp ecx,0
jz _end
pop eax
call atoi ; преобразуем символ в число
mul esi
mov esi,eax
loop next ; переход к обработке следующего аргумента
_end:
mov eax, msg ; вывод сообщения "Результат: "
call sprint
mov eax, esi
call iprintLF ; печать результата
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).

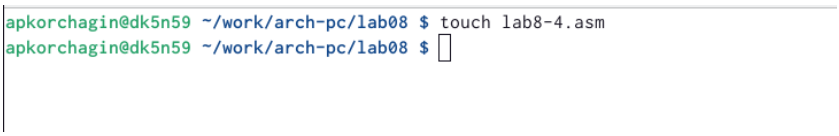

Рис. 4.1: Созданние файла

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).

```
%include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
msg1 db "Функция: f(x)=3x-1"
SECTION .text
global _start
_start:
pop ecx ; Извлеченние из стека в `ecx` 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:
cmp ecx,0h ; проверка, есть ли еще аргументы
jz _end ; если аргументов нет выходим из цикла
; (переход на метку `_end`)
pop eax ; 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. аргумент `esi=esi+eax`
loop next ; переход к обработке следующего аргумента
_end:
mov eax,msg1 ;
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
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 $
```

Код программый

```
%include 'in_out.asm'
```

```
SECTION .data
msg db "Результат: ",0
msg1 db "Функция: f(x)=3x-1"
SECTION .text
global _start
_start:
pop ecx ; Извлеченние из стека в `ecx` 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:
cmp ecx,0h ; проверка, есть ли еще аргументы
jz _end ; если аргументов нет выходим из цикла
; (переход на метку `_end`)
pop eax ; 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. аргумент `esi=esi+eax`
loop next ; переход к обработке следующего аргумента
_end:
mov eax,msg1 ;
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.