РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

направление: *Компьютерные и информационные науки*

Лабораторная работа №8

дисциплина: *Архитектура компьютеров и операционные системы*

студент: Гробман Александр Евгеньевич

Группа: НКАбд-02-23

Цель работы

Целью работы является приобретение навыков написания программ с использованием циклов и обработкой аргументов командной строки..

Задание

1. Изучение циклов и стека в ассемблере
2. Изучение процесса передачи аргументов командной строки
3. Рассмотрение примеров с циклами и стеком
4. Выполнение заданий для самостоятельной работы

Теория

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. Основной функцией стека является функция сохранения 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 esp (указатель стека). Противоположный конец стека называется дном. 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) Для организации циклов существуют специальные инструкции. Для всех инструкций максимальное количество проходов задаётся в регистре ecx. Наиболее простой является инструкция loop. Она позволяет организовать безусловный цикл. Иструкция loop выполняется в два этапа. Сначала из регистра ecx вычитается единица и его значение сравнивается с нулём. Если регистр не равен нулю, то выполняется переход к указанной метке. Иначе переход не выполняется и управление передаётся команде, которая следует сразу после команды loop.

Выполнение лабораторной работы

1. **Символьные и численные данные в NASM**

Я создаю папку для выполнения лабораторной работы № 8 и файл с именем lab8-1.asm. Стоит отметить, что при использовании команды loop в NASM для реализации циклов, необходимо помнить, что эта команда использует регистр ecx в роли четчика и на каждом шаге уменьшает его значение на единицу. Посмотрим на пример программы, которая выводит значение регистра ecx. В файл lab8-1.asm я внес текст программы из листинга 8.1.

```
%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]
label:
mov [N],ecx
mov eax,[N]
call iprintLF
loop label

call quit
```

```
Введите №: 4
4
3
2
1
```

В этом примере показано, что использование регистра ecx в команде loop может привести к неправильному выполнению программы. В текст программы я вношу изменения, которые включают в себя изменение значения регистра ecx внутри цикла.

```
%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]
label:
sub ecx,1
mov [N],ecx
mov eax,[N]
call iprintLF
loop label

call quit
```

Программа запускает бесконечный цикл при нечетном значении N и выводит только нечетные числа при четном значении N.

```
Введите №: 4
3
1
```

Чтобы использовать регистр ecx в цикле и обеспечить правильную работу программы, используется стек. Я внес изменения в текст программы, добавив команды push и pop для сохранения значения счётчика цикла loop в стеке.

```
%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]
label:
push ecx
sub ecx,1
mov [N],ecx
mov eax,[N]
call iprintLF
pop ecx
loop label
call quit
```

Программа выводит числа от N-1 до 0, где количество проходов цикла соответствует значению N.

```
Введите №: 4
3
2
1
0
```

2. **Изучение структуры файлы листинга**

```
%include 'in_out.asm'
SECTION .text
global _start
_start:
pop ecx

pop edx

sub ecx,

next:
cmp ecx, 0
jz _end

pop eax
call sprintLF
loop next

_end:
call quit
```

Был создан исполняемый файл, который я запустил с указанными аргументами. Программа эффективно обработала пять аргументов, которые представляют собой слова или числа, разделенные пробелами.

```
argument
1
argument
2
argument
3
```

Эта программа выводит общую сумму чисел, которые были переданы в программу в качестве аргументов командной строки.

```
%include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
SECTION .text
global _start
_start:
pop ecx

pop edx

sub ecx,1

mov esi, 0

next:
cmp ecx,0h
jz _end

pop eax
call atoi
add esi,eax

loop next
_end:
mov eax, msg
call sprint
mov eax, esi
call iprintLF
call quit
```

```
Результат: 0
```

```
Результат: 18
```

Я внес изменения в код программы с целью расчета произведения аргументов командной строки.

```
%include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
SECTION .text
global _start
_start:
pop ecx

pop edx

sub ecx,1

mov esi, 1

next:
cmp ecx,0h
jz _end

pop eax
call atoi
mov ebx,eax
mov eax,esi
mul ebx
mov esi,eax

loop next
_end:
mov eax, msg
call sprint
mov eax, esi
call iprintLF
call quit
```

```
Результат: 1
```

```
Результат: 360
```

3. **Задания для самостоятельной работы**

вариант 8: □ (□) = 7 + 2□

```
%include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
fx: db 'f(x)= 7 + 2x',0

SECTION .text
global _start
_start:
mov eax, fx
call sprintLF
pop ecx
pop edx
sub ecx,1
mov esi, 0

next:
cmp ecx,0h
jz _end
pop eax
call atoi
mov ebx,2
mul ebx
add eax,7
add esi,eax

loop next

_end:
mov eax, msg
call sprint
```

```
f(x)= 7 = 2x
Результат: 9
```

```
f(x)= 7 = 2x
Результат: 156
```

5. **Вывод**

Я освоил работу со стеком, циклом и аргументами на ассемблере nasm

Отправляем файлы на гитхаб.

Ссылка на отчёт https://github.com/DaOneme/AEGrobman_study_2023-2024_arhpc/tree/main/Labs/Lab08