РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 8

дисциплина: Архитектура компьютера

Программирование цикла. Обработка аргументов командной строки

Студент: ТУЙИШИМЕ Тьерри

Группа: НКАбд-05-25

Оглавление

1. Цель работы

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

2. Теоретическое введение

2.1. Организация стека

Стек: структура данных по принципу LIFO (Last In — First Out). В архитектуре процессора стек реализован аппаратно с использованием регистров:

- ESP (Stack Pointer) - указатель на вершину стека
- EBP (Base Pointer) - указатель на базу стека

Основные операции:

- ✓ `PUSH` - добавление элемента в стек
- ✓ `POP` - извлечение элемента из стека

2.2. Инструкции организации циклов

Команда LOOP:

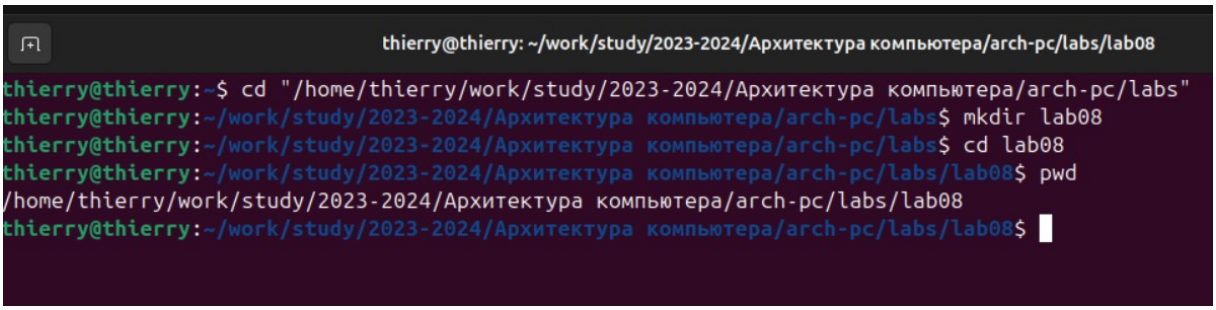
- ✓ Уменьшает ECX на 1
- ✓ Если ECX ≠ 0, переход к метке
- ✓ Если ECX = 0, продолжение выполнения

3. Порядок выполнения лабораторной работы

3.1. Реализация циклов в NASM

Шаг 1: Создание рабочего каталога

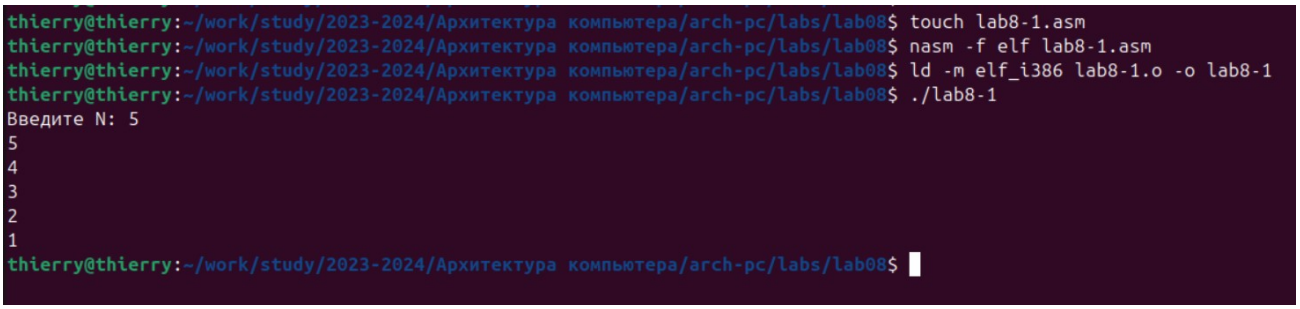
Организация рабочего пространства для лабораторной работы



Результат: Создан каталог для файлов лабораторной работы

Шаг 2: Создание базовой программы с циклом

Изучение работы инструкции LOOP и регистра ECX



Программа запрашивает число N и выводит значения от N до 1, используя LOOP для организации цикла.

Шаг 3: Модификация с изменением ECX в цикле

Демонстрация проблемы изменения ECX в теле цикла

Проблема: LOOP также уменьшает ECX, поэтому общее уменьшение составляет 2 за итерацию.

Шаг 4: Использование стека для сохранения ECX

Цель: Решение проблемы с использованием стека

Open • lab8-1.asm
~/work/study/2023-2024/Архитектура компьют

```
mov eax,N
call atoi
mov [N],eax

mov ecx,[N]
label:
push ecx    ; сохранение ECX в стек
sub ecx,1
mov [N],ecx
mov eax,[N]
call iprintLF
pop ecx     ; восстановление ECX из стека
loop label

call quit
```

Результат:Корректная работа цикла благодаря сохранению/восстановлению ECX

3.2. Обработка аргументов командной строки

Шаг 5: Программа вывода аргументов

Цель: Изучение механизма передачи аргументов через стек

```
thierry@thierry:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08$ touch lab8-2.asm
thierry@thierry:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08$ nasm -f elf lab8-2.asm
thierry@thierry:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08$ ld -m elf_i386 lab8-2.o -o lab8-2
thierry@thierry:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08$ ./lab8-2 аргумент1 аргумент 2 'аргумен
т 3'
аргумент1
аргумент
2
аргумент 3
```

Open

lab8-2.asm

~/work/study/2023-2024/Архитектура компьютер

lab8-1.asm | lab8-2.asm

```
%include 'in_out.asm'

SECTION .text
global _start

_start:
pop ecx ; Извлекаем из стека в `ecx` количество аргументов
pop edx ; Извлекаем из стека в `edx` имя программы
sub ecx, 1 ; Уменьшаем `ecx` на 1 (без имени программы)

next:
cmp ecx, 0 ; проверяем, есть ли еще аргументы
jz _end ; если аргументов нет выходим из цикла
pop eax ; иначе извлекаем аргумент из стека
call sprintLF ; вызываем функцию печати
loop next ; переход к обработке следующего аргумента

_end:
call quit
```

Программа демонстрирует извлечение аргументов из стека в обратном порядке

Шаг 6: Программа вычисления суммы аргументов

Цель: Обработка числовых аргументов и вычисления

```
thierry@thierry:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08$ touch lab8-3.asm
thierry@thierry:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08$ nasm -f elf lab8-3.asm
thierry@thierry:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08$ ld -m elf_i386 lab8-3.o -o lab8-3
thierry@thierry:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08$ ./lab8-3 12 13 7 10 5
Результат: 47
thierry@thierry:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08$
```

Open

lab8-3.asm

~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab0

lab8-3.asm

```
SECTION .data
msg db "Результат: ",0

SECTION .text
global _start

_start:
pop ecx ; Извлекаем количество аргументов
pop edx ; Извлекаем имя программы
sub ecx,1 ; Уменьшаем количество аргументов на 1
mov esi, 0 ; Используем `esi` для хранения суммы

next:
cmp ecx,0h ; проверяем, есть ли еще аргументы
jz _end ; если аргументов нет выходим из цикла
pop eax ; извлекаем следующий аргумент из стека
call atoi ; преобразуем символ в число
add esi,eax ; добавляем к сумме
loop next ; переход к следующему аргументу

_end:
mov eax, msg ; вывод сообщения "Результат: "
call sprint
```

Результат: Сумма аргументов 12 + 13 + 7 + 10 + 5 = 47

4. Задание для самостоятельной работы

Задача: написать программу вычисления суммы значений функции f(x) для аргументов командной строки

Вариант 1: f(x) = 2x + 15

Шаг 7: Реализация программы

```
thierry@thierry:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08$ touch lab8-4.asm
thierry@thierry:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08$ nasm -f elf lab8-4.asm
thierry@thierry:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08$ ld -m elf_i386 lab8-4.o -o lab8-4
thierry@thierry:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08$ ./lab8-4 1 2 3 4
Функция: f(x)=2x+15
Результат: 80
thierry@thierry:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab08$
```

Open

lab8-4.asm

~/work/study/2023-2024/Архитектура компьютера/arc

Recently Used Documents

```
next_arg:
cmp ecx, 0 ; Проверяем, остались ли аргументы
jz print_result ; Если нет, переходим к выводу результата

pop eax ; Извлекаем аргумент
call atoi ; Преобразуем в число

; Вычисляем f(x) = 2x + 15
mov ebx, eax ; Сохраняем x в ebx
add eax, eax ; Умножаем на 2 (eax = 2x)
add eax, 15  ; Добавляем 15 (eax = 2x + 15)

add esi, eax ; Добавляем к общей сумме

loop next_arg

print_result:
mov eax, result_msg
call sprint
mov eax, esi
call iprintLF

call quit
```

Проверка вычислений:

- f(1) = 2×1 + 15 = 17
- f(2) = 2×2 + 15 = 19
- f(3) = 2×3 + 15 = 21
- f(4) = 2×4 + 15 = 23

Сумма:17 + 19 + 21 + 23 = 80

5. Содержание отчета

Отчет должен содержать:

- Титульный лист с номером работы и ФИО
- Формулировка цели работы
- Описание результатов выполнения:
- Описание заданий
- Скриншоты выполнения
- Комментарии и выводы
- Листинги всех программ
- Выводы, согласованные с целью

6. Вопросы для самопроверки

1. Опишите работу команды Loop.

- Уменьшает ECX на 1
- Если ECX ≠ 0, переход к метке
- Если ECX = 0, продолжение выполнения

2. Как организовать цикл без LOOP?

- ✓ Использовать CMP и условные переходы (JNE, JNZ)

3. Дайте определение стека.

- ✓ Структура LIFO для временного хранения данных

4. Порядок выборки из стека?

- Данные извлекаются в обратном порядке относительно добавления

7. Выводы

В ходе лабораторной работы успешно:

- ✓ Изучены принципы организации циклов в NASM
- ✓ Освоена обработка аргументов командной строки
- ✓ Приобретены навыки работы со стеком
- ✓ Реализованы программы вычисления сумм и математических функций
- ✓ Решена задача самостоятельной работы для варианта f(x) = 2x + 15