Отчёт по лабораторной работе №9

Понятие подпрограммы. Отладчик GDB

Студент:

ФИО: Куашев Бетал Муратович
Группа: НПИБд-02-24
Университет: РУДН

Цель работы

Приобретение навыков написания программ с использованием подпрограмм. Знакомство с методами отладки при помощи GDB и его основными возможностями.

Выполнение задания

Программа 1: Реализация функции f(x) = 2x + 7

Описание: Программа, которая вводит значение (x) с клавиатуры, вычисляет (f(x) = 2x + 7) в подпрограмме и выводит результат на экран.

Код:

```
%include 'in_out.asm'

section .data
    msg db 'Введите x: ', 0
    result db '2x+7=', 0

section .bss
    x resb 80
    res resb 80

section .text
    global _start

_start:
    ; Основная программа
    mov eax, msg
    call sprint
    mov ecx, x
    mov edx, 80
    call sread

    mov eax, x
    call atoi
    call _calcul

    mov eax, result
    call sprint
    mov eax, [res]
    call iprintLF

    call quit

_calcul:
    mov ebx, 2
    mul ebx
    add eax, 7
    mov [res], eax
    ret
```

Программа 2: Реализация функций f(g(x))

Описание: Расширение программы 1. Введено вычисление (g(x) = 3x - 1) в отдельной подпрограмме и вызов этой подпрограммы из функции (f(g(x))).

Код:

```
%include 'in_out.asm'

section .data
    msg db 'Введите x: ', 0
    result db 'f(g(x))=', 0

section .bss
    x resb 80
    res resb 80

section .text
    global _start
```

```
_start:
    ; Основная программа
    mov eax, msg
    call sprint
    mov ecx, x
    mov edx, 80
    call sread

    mov eax, x
    call atoi
    call _calcul

    mov eax, result
    call sprint
    mov eax, [res]
    call iprintLF

    call quit

_calcul:
    push eax
    call _subcalcul
    mov ebx, 2
    mul ebx
    add eax, 7
    mov [res], eax
    ret

_subcalcul:
    mov ebx, 3
    mul ebx
    sub eax, 1
    ret
```

Программа 3: Пример тестовой программы "Hello, world!"

Описание: Программа демонстрирует вывод строки "Hello, world!" на экран.

Код:

```
section .data
    msg1 db 'Hello, ', 0
    msg2 db 'world!', 10, 0

section .text
    global _start

_start:
    mov eax, 4
    mov ebx, 1
    mov ecx, msg1
    mov edx, 7
    int 0x80

    mov eax, 4
    mov ebx, 1
    mov ecx, msg2
    mov edx, 7
    int 0x80

    mov eax, 1
    mov ebx, 0
    int 0x80
```

Результаты

1. **Результаты выполнения программы 1**: Скриншоты демонстрируют корректное вычисление (f(x) = 2x + 7).
2. **Результаты выполнения программы 2**: Программа успешно вычисляет (f(g(x))), результат выводится на экран.
3. **Результаты выполнения программы 3**: Сообщение "Hello, world!" корректно отображается на экране.

Выводы

В ходе выполнения лабораторной работы были достигнуты следующие результаты:

1. Реализованы подпрограммы на языке ассемблера NASM.
2. Освоена работа с инструкциями `call` и `ret` для вызова и возврата из подпрограмм.
3. Изучены методы отладки программ с помощью GDB:
    - Установка точек останова;
    - Пошаговое выполнение;
    - Просмотр содержимого регистров и памяти.
4. Все задания лабораторной работы выполнены в полном объёме. Полученные знания будут полезны для дальнейшего изучения системного программирования.

Лабораторная работа №7

Команды безусловного и условного переходов в NASM.

Программирование ветвлений.

Выполнил: Куашев Бетал Муратович **Группа:** НПИбд-02-24 **Год:** 2024

Цель работы

1. Реализация программы нахождения наименьшего из трёх чисел.
2. Создание программы для вычисления функции (f(x)) с использованием условных переходов.

Описание задач

1. **Задача 1**: Нахождение минимального из трёх чисел (a = 94), (b = 5), (c = 58).
2. **Задача 2**: Вычисление функции (f(x)):
   - (f(x) = 3x), если (x = 3).
   - (f(x) = a + 1), если (x \neq 3).
     Значения:
     - (x_1 = 3), (a_1 = 4).
     - (x_2 = 1), (a_2 = 4).

Листинги программ

1. Нахождение минимального из трёх чисел

```
; find_min.asm - Программа на NASM
; Нахождение минимального из трёх чисел: a, b, c

SECTION .data
    a DB 94
    b DB 5
    c DB 58
    min DB 0

SECTION .text
    GLOBAL _start

_start:
    mov al, [a]
    cmp al, [b]
    jle check_c
    mov al, [b]

check_c:
    cmp al, [c]
    jle set_min
    mov al, [c]

set_min:
    mov [min], al

    mov eax, 1
    xor ebx, ebx
    int 80h
```

2. Вычисление функции f(x)

```
; calculate_fx.asm - Программа на NASM
; Вычисление функции f(x):
;   f(x) = 3x, если x = 3
;   f(x) = a + 1, если x != 3

SECTION .data
    x DB 3              ; Входное значение x
    a DB 4              ; Входное значение a
    result DB 0         ; Результат вычисления f(x)

SECTION .text
    GLOBAL _start

_start:
    ; Сравниваем x с 3
    mov al, [x]         ; Загружаем x в AL
    cmp al, 3           ; Сравниваем x с 3
    je calculate_3x     ; Если x = 3, переходим к 3x

calculate_a_plus_1:
    ; Вычисляем a + 1
    mov al, [a]         ; Загружаем a в AL
    add al, 1           ; Прибавляем 1
    jmp store_result    ; Переходим к сохранению результата

calculate_3x:
    ; Вычисляем 3x
```

```
    mov al, [x]          ; Загружаем x в AL
    add al, al           ; Умножаем x на 2
    add al, [x]          ; Добавляем x ещё раз (3x)

store_result:
    mov [result], al     ; Сохраняем результат в result

    ; Завершаем выполнение программы
    mov eax, 1           ; Код системного вызова: sys_exit
    xor ebx, ebx         ; Код завершения программы: 0
    int 80h              ; Вызов системного прерывания
```

Результаты Программа для нахождения минимального числа корректно определяет минимальное значение из трёх заданных чисел. Программа для вычисления функции f(x) успешно рассчитывает результат для всех заданных случаев.

Вывод Лабораторная работа позволила закрепить навыки работы с командами условного и безусловного переходов в NASM. В ходе выполнения задач была реализована программа нахождения минимального из трёх чисел и программа для вычисления функции f(x)f(x) с учётом различных условий. Эти программы продемонстрировали использование ветвлений для обработки данных и выполнения вычислений. Работы успешно протестированы, и их выполнение соответствует целям задания.

Отчёт по лабораторной работе №8

Программирование цикла. Обработка аргументов командной строки.

Студент:

ФИО: Куашев Бетал Муратович
Группа: НПИБд-02-24
Университет: РУДН

Цель работы

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

Выполнение задания

Программа 1: Вывод значений регистра ECX

Описание: Программа с использованием инструкции `loop`, которая выводит значения регистра `ECX` на каждом шаге цикла.

Код:

```
section .data
    msg db "Введите N: ", 0

section .bss
    N resb 4

section .text
    global _start

_start:
    mov eax, msg
    call sprint
    mov ecx, [N]
label:
    mov eax, ecx
    call iprintLF
    loop label
    call quit
```

Программа 2: Вывод аргументов командной строки

Описание: Программа обрабатывает аргументы командной строки и выводит их на экран.

Код:

```
section .text
global _start

_start:
    pop ecx
    pop edx
    sub ecx, 1
next:
    cmp ecx, 0
    jz _end
    pop eax
    call sprintLF
    loop next
_end:
```

```
    call quit
```

Программа 3: Вычисление суммы значений функции f(x) = 10x - 5

Описание: Программа вычисляет сумму значений функции f(x) = 10x - 5 для аргументов, переданных через командную строку.

Код:

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

section .text
global _start

_start:
    pop ecx
    pop edx
    sub ecx, 1
    mov esi, 0
next:
    cmp ecx, 0
    jz _end
    pop eax
    call atoi
    mov ebx, 10
    mul ebx
    sub eax, 5
    add esi, eax
    loop next
_end:
    mov eax, msg
    call sprint
    mov eax, esi
    call iprintLF
    call quit
```

Выводы

В качестве результатов выполнения заданий были получены следующие выводы:

1. **Программа 1**:
   - Корректно выводит значения регистра ECX на каждом шаге цикла.
   - Была проверена с различными начальными значениями регистра, и все результаты соответствовали ожиданиям.
2. **Программа 2**:
   - Успешно обработала и вывела все переданные аргументы командной строки.
   - Включая случаи с различным количеством аргументов, программа работала стабильно.
3. **Программа 3**:
   - Корректно вычисляет сумму значений функции `f(x) = 10x - 5` для набора переданных аргументов.
   - Были протестированы случаи с положительными, отрицательными и нулевыми аргументами, и программа выдала правильные результаты.

Кроме того, во всех программах реализована работа со стеком и регистрами, что продемонстрировало понимание архитектуры процессора.

В ходе выполнения лабораторной работы были приобретены следующие навыки и достигнуты следующие результаты:

1. Освоено программирование циклов с использованием инструкции `loop` и регистров процессора.
2. Получены навыки работы с аргументами командной строки, их извлечения и обработки.
3. Реализованы программы, которые корректно выполняют поставленные задачи, включая работу со стеком для сохранения и извлечения данных.
4. Программа для вычисления суммы значений функции продемонстрировала уверенное владение операциями арифметики на уровне ассемблера.

Все задачи лабораторной работы были успешно выполнены. Результаты подтверждают корректность работы программ, что демонстрирует понимание теоретического материала и практическое применение навыков программирования на языке ассемблера NASM.

Лабораторная работа №4

Создание и процесс обработки программ на языке ассемблера NASM

Выполнил: Куашев Бетал Муратович
Группа: НПИбд-02-24
Архитектура ЭВМ
Год: 2024

Цель работы

Целью лабораторной работы является освоение процесса компиляции и сборки программ, написанных на языке ассемблера NASM, а также изучение принципов работы с ассемблерными программами, включая системные вызовы, работу с регистрами и операциями ввода-вывода.

Описание задания

В рамках лабораторной работы было выполнено два задания:

1. **Создание программы "Hello World"**: Написание простой программы на языке ассемблера, которая выводит строку "Hello world!" на экран с использованием системных вызовов операционной системы.

2. **Модификация программы для вывода фамилии и имени**: После выполнения первого задания было предложено изменить программу так, чтобы она выводила строку с фамилией и именем студента, а не стандартное "Hello world!".

Результаты выполнения лабораторной работы

Задание 1: Создание программы "Hello World"

Для выполнения задания была написана простая программа на ассемблере, которая выводит на экран строку `"Hello world!"`. В процессе выполнения использовались следующие шаги:

1. Определена секция данных программы, где была размещена строка `Hello world!` и вычислена её длина.
2. Использован системный вызов для вывода строки на стандартный вывод.
3. Программа завершалась с использованием системного вызова для выхода с кодом возврата.

Код программы 'hello.asm'

```
; hello.asm - Программа на языке ассемблера NASM
; Выводит строку 'Hello world!' на стандартный вывод

SECTION .data           ; Секция данных программы
    hello: DB 'Hello world!', 10 ; Строка для вывода с символом новой строки
    helloLen: EQU $-hello        ; Длина строки hello

SECTION .text           ; Секция кода программы
    GLOBAL _start        ; Объявляем точку входа

_start:                 ; Точка входа в программу
    ; Вызов системной функции записи (sys_write)
    mov eax, 4          ; Код функции записи (4)
    mov ebx, 1          ; Описатель файла: 1 - стандартный вывод
    mov ecx, hello      ; Адрес строки для вывода
    mov edx, helloLen   ; Длина строки
    int 80h             ; Вызов системного прерывания

    ; Вызов системной функции выхода (sys_exit)
    mov eax, 1          ; Код функции выхода (1)
    mov ebx, 0          ; Код завершения: 0 (успешное выполнение)
    int 80h             ; Вызов системного прерывания


#### **Задание 2: Модификация программы для вывода фамилии и имени**
; lab4.asm - Программа на языке ассемблера NASM
; Выводит строку с именем и фамилией на стандартный вывод

SECTION .data           ; Секция данных программы
    hello: DB 'Куашев Бетал Муратович', 10 ; Строка для вывода с новой строкой
    helloLen: EQU $-hello                       ; Длина строки hello

SECTION .text           ; Секция кода программы
    GLOBAL _start        ; Объявляем точку входа

_start:                 ; Точка входа в программу
    ; Вызов системной функции записи (sys_write)
    mov eax, 4          ; Код функции записи (4)
    mov ebx, 1          ; Описатель файла: 1 - стандартный вывод
    mov ecx, hello      ; Адрес строки для вывода
    mov edx, helloLen   ; Длина строки
    int 80h             ; Вызов системного прерывания

    ; Вызов системной функции выхода (sys_exit)
    mov eax, 1          ; Код функции выхода (1)
    mov ebx, 0          ; Код завершения: 0 (успешное выполнение)
    int 80h             ; Вызов системного прерывания


<style type="text/css">@media print {
    *, :after, :before {background: 0 0 !important;color: #000 !important;box-shadow: none !important;text-shadow: none !im
    a, a:visited {text-decoration: underline}
    a[href]:after {content: " (" attr(href) ")"}
    abbr[title]:after {content: " (" attr(title) ")"}
    a[href^="#"]:after, a[href^="javascript:"]:after {content: ""}
    blockquote, pre {border: 1px solid #999;page-break-inside: avoid}
    thead {display: table-header-group}
    img, tr {page-break-inside: avoid}
    img {max-width: 100% !important}
    h2, h3, p {orphans: 3;widows: 3}
    h2, h3 {page-break-after: avoid}
}
```

```
html {font-size: 12px}
@media screen and (min-width: 32rem) and (max-width: 48rem) {
    html {font-size: 15px}
}
@media screen and (min-width: 48rem) {
    html {font-size: 16px}
}
body {line-height: 1.85}
.air-p, p {font-size: 1rem;margin-bottom: 1.3rem}
.air-h1, .air-h2, .air-h3, .air-h4, h1, h2, h3, h4 {margin: 1.414rem 0 .5rem;font-weight: inherit;line-height: 1.42}
.air-h1, h1 {margin-top: 0;font-size: 3.998rem}
.air-h2, h2 {font-size: 2.827rem}
.air-h3, h3 {font-size: 1.999rem}
.air-h4, h4 {font-size: 1.414rem}
.air-h5, h5 {font-size: 1.121rem}
.air-h6, h6 {font-size: .88rem}
.air-small, small {font-size: .707em}
canvas, iframe, img, select, svg, textarea, video {max-width: 100%}
body {color: #444;font-family: 'Open Sans', Helvetica, sans-serif;font-weight: 300;margin: 0;text-align: center}
img {border-radius: 50%;height: 200px;margin: 0 auto;width: 200px}
a, a:visited {color: #3498db}
a:active, a:focus, a:hover {color: #2980b9}
pre {background-color: #fafafa;padding: 1rem;text-align: left}
blockquote {margin: 0;border-left: 5px solid #7a7a7a;font-style: italic;padding: 1.33em;text-align: left}
li, ol, ul {text-align: left}
p {color: #777}</style>
```