Отчёт по лабораторной работе №9

Дисциплина:Архитектура компьютера

Ванюшкина Татьяна Валерьевна

Содержание

# 1 Цель работы

Целью работы является приобретение навыков написания программ с использованием подпрограмм и знакомство с методами отладки при помощи GDB и его основными возможностями.

# 2 Теоретическое введение

GDB (GNU Debugger — отладчик проекта GNU) [1] работает на многих UNIX-подобных системах и умеет производить отладку многих языков программирования.GDB предлагает обширные средства для слежения и контроля за выполнением компьютерных программ.От ладчик не содержит собственного графического пользовательского интерфейса и использует стандартный текстовый интерфейс консоли.Однакодля GDB существует несколько сторон них графических надстроек,а крометого,некоторые интегрированные среды разработки используютего в качестве базовой подсистемы отладки. Отладчик GDB(как и любой другой отладчик)позволяет увидеть,что происходит «внутри» программы в момент её выполнения или что делает программа в момент сбоя. GDB может выполнять следующие действия: • начатьвыполнение программы,задав всё,что можетповлиятьна её поведение; • остановит ьпрограмму при указанных условиях; • исследовать,что случилось,когда программа остановилась;

# 3 Выполнение лабораторной работы

1. Реализация подпрограмм в NASM

Создаю каталог для выполнения лабораторной работы №9:

(рис.1 **¿fig:001?**) ![Создание каталога](data:application/octet-stream;base64,) {#fig:001}

Перехожу в него и создаю файл lab09-1.asm:

(рис.2 **¿fig:002?**) ![Переход в каталог и создание файла](data:application/octet-stream;base64,) {#fig:002}

Ввожу в файл lab09-1.asm текст программы из листинга 9.1.

(рис.3 **¿fig:003?**) ![Ввод текста программы](data:application/octet-stream;base64,) {#fig:003}

Создаю исполняемый файл и проверяю его работу:

(рис.4 **¿fig:004?**) ![Создание и проверка файла](data:application/octet-stream;base64,) {#fig:004}

Изменяю текст программы,добавив подпрограмму \_subcalcul в подпрограмму \_calcul, для вычисления выражения 𝑓(𝑔(𝑥)), где 𝑥 вводится с клавиатуры, 𝑓(𝑥) = 2𝑥 + 7, 𝑔(𝑥) = 3𝑥 −1.

(рис.5 **¿fig:005?**) ![Замена текста программы](data:application/octet-stream;base64,) {#fig:005}

Создаю исполняемый файл и проверяю его работу:

(рис.6 **¿fig:006?**) ![Создание и проверка файла](data:application/octet-stream;base64,) {#fig:006}

Создаю файл lab09-2.asm :

(рис.7 **¿fig:007?**) ![Создание файла](data:application/octet-stream;base64,) {#fig:007}

Ввожу в него текст программы из Листинга 9.2:

(рис.8 **¿fig:008?**) ![Ввод текста программы](data:application/octet-stream;base64,) {#fig:008}

Получаю исполняемый файл.Для работы с GDB в исполняемый файл добавляю отладочную информацию,для этого трансляцию программ провожу с ключом ‘-g’.

(рис.9 **¿fig:009?**) ![исполнение файла](data:application/octet-stream;base64,) {#fig:009}

Загружаю исполняемый файл в отладчик gdb:

(рис.10 **¿fig:010?**) ![Загрузка файла в откладчик gdb](data:application/octet-stream;base64,) {#fig:010}

Проверяю работу программы,запустив ее в оболочке GDB с помощью команды run:

(рис.11 **¿fig:011?**) ![команда run](data:application/octet-stream;base64,) {#fig:011}

Для более подробного анализа программы устанавливаю брейкпоинт на метку \_start, с которой начинается выполнениелюбой ассемблерной программы,и запускаю её:

(рис.12 **¿fig:012?**) ![Команда break _start](data:application/octet-stream;base64,) {#fig:012}

Смотрю дисассимилированный код программы с помощью команды disassemble начиная с метки \_start:

(рис.13 **¿fig:013?**) ![Команда disassemble _start](data:application/octet-stream;base64,) {#fig:013}

Переключаюсь на отображение команд с Intel’овским синтаксисом,введя команду set disassembly-flavor intel:

(рис.14 **¿fig:014?**) ![Команда disassemble _start](data:application/octet-stream;base64,) {#fig:014}

Включаю режим псевдографики для более удобного анализа программы:

(рис.15 **¿fig:015?**) ![команды layout asm layout regs](data:application/octet-stream;base64,) {#fig:015}

1. Добавление точек останова

На предыдущих шагах была установлена точка останова по имениметки(\_start).Про верьте это с помощью команды info breakpoints:

(рис.16 **¿fig:016?**) ![команда info breakpoints](data:application/octet-stream;base64,) {#fig:016}

Устанавливаю еще одну точку останова по адресу инструкции:

(рис.17 **¿fig:017?**) ![Команда break](data:application/octet-stream;base64,) {#fig:017}

Смотрю информацию о всех установленных точках останова:

(рис.18 **¿fig:018?**) ![Команда i b](data:application/octet-stream;base64,) {#fig:018}

1. Работа с данными программы в GDB

Смотрю содержание регистров с помощью команды info registers:

(рис.19 **¿fig:019?**) ![info registers](data:application/octet-stream;base64,) {#fig:019}

Смотрю значение переменной msg1 по имени:

(рис.20 **¿fig:020?**) ![x/1sb &msg1](data:application/octet-stream;base64,) {#fig:020}

Меню первый символ переменной msg1:

(рис.21 **¿fig:021?**) ![Замена первого символа](data:application/octet-stream;base64,) {#fig:021}

С помощью команды set меняю значение регистра ebx:

(рис.22 **¿fig:022?**) ![Замена значения регистра ebx](data:application/octet-stream;base64,) {#fig:022}

(рис.23 **¿fig:023?**) ![Замена значения регистра ebx](data:application/octet-stream;base64,) {#fig:023}

1. Обработка аргументов командной строки в GDB

Копирую файл lab8-2.asm, созданный при выполнении лабораторной работы №8, с программой выводящей на экран аргументы командной строки (Листинг 8.2) в файл с именемlab09-3.asm:

(рис.24 **¿fig:024?**) ![Копирование файла](data:application/octet-stream;base64,) {#fig:024}

Создаю исполняемый файл:

(рис.25 **¿fig:025?**) ![Создание исполняемого файла](data:application/octet-stream;base64,) {#fig:025}

Загружаю исполняемый файл в отладчик,указав аргументы:

(рис.26 **¿fig:026?**) ![Загрузка файла в откладчик](data:application/octet-stream;base64,) {#fig:026}

Установливаю точку останова перед первой инструкцией в программе и запускаю ее:

(рис.27 **¿fig:027?**) ![Установка точки останова](data:application/octet-stream;base64,) {#fig:027}

Адрес вершины стека храниться в регистре esp и по этому адресу располагается число равное количеству аргументов командной строки (включая имя программы):

(рис.28 **¿fig:028?**) ![x/x $esp](data:application/octet-stream;base64,) {#fig:028}

Посмотриваю остальные позиции стека–по адесу [esp+4] располагается адрес в памяти где находиться имя программы,по адесу [esp+8] храниться адрес первого аргумента,по аресу [esp+12]–второго ит.д:

(рис.29 **¿fig:029?**) ![gdb](data:application/octet-stream;base64,) {#fig:029}

Задание для самостоятельной работы:

1. Преобразовываю программу из лабораторной работы №8,реализовав вычисление значения функции 𝑓(𝑥) как подпрограмму:

(рис.30 **¿fig:030?**) ![Преобразование программы](data:application/octet-stream;base64,) {#fig:030}

1. Создаю файл lab09-5.asm:

(рис.31 **¿fig:031?**) ![Создание файла](data:application/octet-stream;base64,) {#fig:031}

Ввожу в него текст программы листигна 9.3:

(рис.32 **¿fig:032?**) ![Ввод текста программы](data:application/octet-stream;base64,) {#fig:032}

Получаю исполняемый файл и загружаю его в отладчик gdb:

(рис.33 **¿fig:033?**) ![Загрузка файла в отладчик gdb](data:application/octet-stream;base64,) {#fig:033}

Просматриваю изменение значений регистров.

(рис.34 **¿fig:034?**) ![ошибка программы](data:application/octet-stream;base64,) {#fig:034}

При выполнении инструкции mul результат умножения меняет edx, значение регистра ebx не обновляется напрямую. Из-за этого программа неверно подсчитывает функцию

Исправляю ошибку программы и запускаю иполняемый файл:

(рис.35 **¿fig:035?**) ![запуск исправленной программы](data:application/octet-stream;base64,) {#fig:035}

# 4 Выводы

В ходе выполнения лабораторной работы я приобрела навыки написания программ с использованием подпрограмм и познакомилась с методами откладки при помощи GDB и его основными возможностями.

# Список литературы

Курс: Архитектура компьютеров и операционные системы. Раздел “Архитектура компьютеров” (02.03.00, УГСН) (rudn.ru)