Лукьянчук Богдан БПИ-213

Отчёт

Идз №3

Вариант № 29

4 балла:

1. Решение задачи на C представлено в файле main.c. Ввод данных осуществляется с помощью файла, путь к которому задан через командную строку. Вывод данных осуществляется тоже в файл, путь у которому задан в качестве аргумента командной строки.
2. С помощью флага gcc -S -O0 -masm=intel -fno-asynchronous-unwind-tables -fcf-protection=none компилирую Assembly код из C кода. В папке main_without_optimize файле main_without_Optimize.s находится сам Assembly код с поясняющими комментариями, так же в папке располагаются перемещаемый объектный файл main_without_Optimize.o и исполняемый объектный файл main_without_Optimize.
3. При использовании флага из 2 пункта были убраны лишние макросы.
4. Модифицированная ассемблерная программа находится в папке main_without_optimize, описание которой представлено во 2 пункте.
5. Для тестирования программы была создана папка Tests с input.txt (готовые входные данные) и output.txt (правильные выходные данные, соответствующие входным) файлами. С помощью скрипта на bash, который представлен в файле test.sh, производилась проверка на корректность работы программ, программа получала на вход данные из input.txt и записывала выходные данные в файл out.txt, который позже сравнивался с output.txt. Наглядно можно увидеть корректность работы программ, используя команды Makefile: make testC (тестирует программу main.c), make testAsm (тестирует програму main_without_Optimize.s).
6. Исходный текст на ассемблере, сформированный компилятором C представлен в файле main.s.

5 баллов

1. Изначально программа была разбита на функции randomNum, function, integral, main. Из функции main вызывалась функция integral, из которой вызывалась функция function, а также в зависимости от ключей в командной строке вызывалась

функция randomNum (полное описание принимаемых аргументов функции, вызов функций и передачу аргументов им можно увидеть в файле main.c).

2. В функции randomNum были инициализированы локальные переменные double temp, double max, double min, double min_for_x. В функции function инициализирована локальная перемення double result.
3. В файле main_wthout_optimize.s, который располагается в папке main_wthout_optimize, описаны все передачи фактических параметров и перенос возвращаемого результата, а также добавлены комментарии, описывающие связь между параметрами языка С и регистрами(стеком).
4. Исходный текст на ассемблере, сформированный компилятором С представлен в файле main.s.

6 баллов:

1. С помощью сохраняемых регистров в функциях: randomNum, function, используемые переменные перемещались в эти регистры, а не на стек. Полное описание изменения ассемблерного кода за счет использования регистров можно увидеть в файле main_with_Registers.s, который располагается в папке main_with_registers. В файле также добавлены комментарии, поясняющие эквивалентное использование регистров вместо переменных. Также в папке представлены исполняемый файл и перемещаемый файл данной программы.
2. Для тестирования программы была создана папка Tests с input.txt (готовые входные данные) и output.txt (правильные выходные данные, соответствующие входным) файлами. С помощью скрипта на bash, который представлен в файле test.sh, производилась проверка на корректность работы программ, программа получала на вход данные из input.txt и записывала выходные данные в файл out.txt, который позже сравнивался с output.txt. Наглядно можно увидеть корректность работы программы, используя команду Makefile: make testAsmReg.
3. Объектный файл модифицированной программы, использующей регистры, имеет размер 5367 байт, а исполняемый файл этой же программы имеет размер 17088 байт. Объектный файл исходной программы имеет размер 5960 байт, а исполняемый 17288 байт.
4. Исходный текст на ассемблере, сформированный компилятором С представлен в файле main.s.

7 баллов:

1. Свою программу модифицирую для того, чтобы она могла считывать команды и расположение файлов из командной строки. Программа считывает следующие аргументы из командной строки: **1)** input.txt(расположение файла с входными данными) output.txt(расположение файла, куда необходимо записать результат работы); **2)** (time) (loop) input.txt(расположение файла с входными данными) output.txt(расположение файла, куда необходимо записать результат работы). Ключи time loop опциональны, time используется для включения замера времени, loop для зацикливания программы. Для примера работы программы в Makefile создана инструкция ReadWriteFile, которая запускает программу и передает ей аргументы **1)** InOutputFiles/input.txt InOutputFiles/output.txt. Здесь представлена передача расположения файлов, откуда мы берем данные, и куда мы записываем результат. Результат выполнения инструкции make ReadWriteFile можно увидеть в файлах output.txt и output_sec.txt, которые расположены в папке InOutputFiles.
2. В файле main.c прописаны условия, отслеживающие корректное число аргументов в командной строке, а также обрабатываются случаи некорректного открытия файлов.
3. Данная программа была разбита на две единицы компиляции: main_part, functions_part, в первой располагается функция main с вызовами определенных функций, во второй единице компиляции располагаются все вызываемые функцией main функции. Код обеих единиц компиляции, а также слинкованный исполняемый файл complete представлены в папке main_with_parts.s.
4. В папке InOutputData созданы файлы для проверки работоспособности программы, а также прописана цель в Makefile ReadWriteFile, которая запускает программу с определенными аргументами (расположение входного и выходного файла).
5. Для тестирования программы была создана папка Tests с input.txt (готовые входные данные) и output.txt (правильные выходные данные, соответствующие входным) файлами. С помощью скрипта на bash, который представлен в файле test.sh, производилась проверка на корректность работы программ, программа получала на вход данные из input.txt и записывала выходные данные в файл out.txt, который позже сравнивался с output.txt. Наглядно можно увидеть корректность работы программы, используя команду Makefile: make testCfile.
6. Исходный текст на ассемблере, сформированный компилятором С представлен в файле main.s.

8 баллов:

1. В исходную программу была добавлена функция randomNum, принимающая адрес переменных double a, double b, double left, double right, которые необходимо заполнить. Генератор подключается при появлении команды random в командной строке, результат записывается в output.txt файл, переданный через командную строку. Генератор случайных чисел обрабатывает случаи некорректных данных (отрицательные интервалы интеграла, левая граница больше правой).
2. Командная строка имеет несколько ключей, с которыми работает: **1)** input.txt(расположение файла с входными данными) output.txt(расположение файла, куда необходимо записать результат работы); **2)** random output.txt(расположение файла, куда необходимо записать результат работы) (программа генерирует переменные случайным образом, результат выводится в output.txt); **3)**(time) (loop) input.txt(расположение файла с входными данными) output.txt(расположение файла, куда необходимо записать результат работы). Ключ time включает таймер времени работы программы и результат выводит в output.txt. Ключ loop включает режим зацикливания программы 10000 раз.
3. В исходные программы была добавлена модификация, которая производит замер времени работы функции integral. При запуске программы с первым ключом time программа будет работать в режиме замера времени, все остальные ключи также работают вместе с time, если вводить их следующими аргументами при запуске. Для замера производительности разных вариантов программы, время работы основной (не модифицированной) программы main.c было увеличено до ~ 1.2 секунды с помощью зацикливания выполнения функции integral и специально подобранных данных. Всего сравнивалось две модификации программы: main (не модифицированная версия), complete (версия, разбитая на единицы компиляции). По результатам замеров времени все программы запускались с ключом random для генерации случайных данных и со специально подобранными данными. Main выполняется ~ 1.15 – 1.2 секунды, complete ~ за 1.1. Для предоставления результатов замеров работы программ в Makefile прописаны инструкции specialDatatimeforMain и specialDatatimeforComplete, которые запускают эти программы с опцией time и специально подобранными данными.
4. Исходный текст на ассемблере, сформированный компилятором C представлен в файле main.s.

9 баллов:

1. Используя опцию компилятора gcc O3 -S main.c -o main_speed.s, сформировал из программы на C исходный ассемблерный код, оптимизированный по скорости. Далее создал бинарный файл main_speed.o, который слинковал в исполняемый main_speed. Все перечисленные файлы находятся в папке main_speed.
2. Используя опцию компилятора gcc Os -S main.c -o main_size.s, сформировал из программы на C исходный ассемблерный код, оптимизированный по размеру. Далее создал бинарный файл main_size.o, который слинковал в исполняемый main_size. Все перечисленные файлы находятся в папке main_size.
3. Далее провожу сравнительный анализ с ассемблерной программой без оптимизации main.s, с собственной программой main_with_registers.s, в которой вместо переменных максимально возможно используются регистры, а также двумя скомпилированными программами speed.s, size.s.
4.

Программа	Кол-во ассемблерных строк	Размер бинарного файла в байтах	Размер исполняемого файла в байтах	Время работы в секундах
speed	682	12006	17320	~ 1
main_without_optimize	611	10861	17288	~ 1.15 - 1.17
main_with_registers	704	21102	17088	~ 1.1
size	488	8357	17320	~ 1.12 - 1.15