Лукьянчук Богдан БПИ-213

Отчёт

Идз №1

Вариант № 13

4 балла:

1. Решение задачи на C представлено в файле main.c.
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).

5 баллов

1. Изначально программа была разбита на функции readArray, findMin, generateArray, writeArray, main. Из функции main по очереди вызывались данные функции, в которые передавались, соответствующие данные (полное описание принимаемых аргументов функции, вызов функций и передачу аргументов им можно увидеть в файле main.c).
2. В каждой функции был реализован цикл for, в котором использовалась локальная переменная i, использовались переменные, переданные в функцию, в функции

findMin создана локальная переменная valueMin, в функции generateArray создана локальная переменная i и для цикла использовалась переменная j.

3. В файле main_wthout_Optimize.s, который располагается в папке main_with_registers, описаны все передачи фактических параметров и перенос возвращаемого результата, а также добавлены комментарии, описывающие связь между параметрами языка С и регистрами(стеком).

6 баллов:

1. С помощью сохраняемых регистров в функциях: readArray, findMin, generateArray, writeArray, все 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. 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 testASMRegisters.

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)** read input.txt(расположение файла с входными данными) write output.txt(расположение файла, куда необходимо записать результат работы); **2)** write output.txt(расположение файла, куда необходимо записать результат работы) read input.txt(расположение файла с входными данными); **3)** без аргументов (программа ожидает входные данные со стандартного потока ввода stdin и выводит результат работы в stdout. Для примера работы программы в Makefile создана инструкция ReadWriteFile, которая запускает программу и передает ей аргументы **1)** read InOutputFiles/input.txt write InOutputFiles/output.txt **2)** write InOutputFiles/output_sec.txt read InOutputFiles/input_sec.txt. В обоих случаях 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. Данная программа была разбита на две единицы компиляции: main_first, functions_second, 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 представлены в папке main_with_parts.