ИДЗ по архитектуре вычислительных систем

Вариант 23

Выполнил Чугунов Андрей Игоревич БПИ 213

Условие задания

Сформировать массив B, элементы которого являются расстояниями пройденными телом при свободном падении на землю за время в секундах, указанное в массиве A.

Описание работы

В репозитории находятся:

1. report.pdf - файл с отчетом
2. main.c - код программы на языке си
3. main.s - асемблерный листинг без отладочных опций
4. without_macros - программа без лишних макросов
5. refactor.s - прокомментированная программа, прошедшая рефакторинг
6. папка screenshots - там находятся скриншоты с результатами тестов
7. Папка big_test - тест с большим набором данных и его результаты в виде txt файлов.

Реализация и описание отдельных файлов

Си-программа main.c написана так, что изначально содержит функцию и локальную переменную для критерия на 5 баллов. Значение переменной g принято за 10. Реализация исходного массива сделана так:

1. Создается массив int a размером 10000 элементов.
2. Вводится n - число элементов.
3. Если n > 10000 или n < 0, то программа завершается с кодом возврата -1, выдавая сообщение об ошибке

Также было решено ограничить значения элементов массива. 0 <= a[i] <= 10000. a[i] - время, а значит оно не может быть отрицательным. Также я решил ограничить a[i] сверху. Расстояние пройденное телом при падении за 10000 секунд = 5 * 10^8 метров, чего явно достаточно. Атмосфера земли заканчивается примерно на 3*10^6 метров. Так как контекст задачи явно указывает именно на нашу планету, я решил ввести такие ограничения. Если найдется a[i], которое не подходит под описание выше, то функция f не будет вызвана, а в значение b[i] будет записано -1. Такое число не может быть результатом функции, т.к. расстояние всегда положительно. Значит -1 может получится только при недопустимом вводе данных.

- Программа without_macros.s была получена с помощью таких опций командной строки:

  gcc -masm=intel -fno-asynchronous-unwind-tables -fno-jump-tables -fno-stack-protector -fno-exceptions

  ./main.c -S -o ./without_macros.s

  Оставшиеся макросы были удалены вручную.

Программа main.s была получена так:

gcc ./main.c -S -o ./main.s

- Программа refactor.s была получена путем ручного редактирования программы without_macros.s. Данная программа содержит подробные комментарии о соответствии переменных и регистов, а также о принципе ее общей работы.

Сравнение программ

- Из программы without_macros.s после компиляции были удалены следующие макросы:

Имя файла, размер функций, все строки, идущие после последней инструкции ret. Таким образом количество строк получилось 116, а в main.s - 162.

- Программа refactor.s была получена ручным редактированием программы without_macros.s, опишем ее изменения:

1. В функции f переменная t, которая является аргументом, была положена в регистр r11d, а не на стек. Из-за этого, получилось уменьшить количество обращений в память на 2. Также отмечу существование локальной переменной g на стеке. Подробную работу функции и описание вызова можно найти в комментариях к коду.
2. В начале функции main, вместо циклического выделения памяти блоками по 4096 байт, я сразу выделил нужный кусок памяти в 80032 байт.
3. После первого вызова функции scanf, я сохраняю переменную n в регистр r14d и после этого всегда пользуюсь этим регистром, если нужно значение переменной n, чтобы избежать ненужных обращений в память.
4. Аналогично переменную i сохраняю в регистр r15d. Адрес массива a в регистр r12. Адрес массива b в регистр r13.
5. После сохранения всех нужных значений, код меняется соответственно. Все подробности в комментариях к коду.

Значимых изменений больше не было. В итоге в полученной программе используются все регистры процессора(или их младшие части), кроме регистров rbx и rcx. Также количество обращений в память сильно уменьшилось. Общее число строк - 94.

Тестовые прогоны

Чтобы проверить корректную работу программ without_macros.s и refactor.s, мы будем сравнивать их результаты на одинаковых тестах с программой main.s полученной в результате компиляции main.c, то есть main.s точно работает корректно.

Всего будет 4 теста, которые проверяют:

1. Корректность расчетов при корректном вводе данных - build_and_first_test.png
2. Результат при вводе недопустимого n – size_test.png
3. Общую работу программы - ultimate_test.png
4. Общую работу программы на больших данных (1000 элементов) - big_test.txt

Из скриншотов видно, что на первых 3 тестах программы выдали такой же результат, как и main.s. Результаты 4-ого теста неудобно показывать в качестве скриншотов, поэтому было создано 3 файла - big_test_main_output.txt, big_test_refactor_output.txt и big_test_without_macros_output.txt в которые я записал вывод всех 3 программ. Все 3 файла оказались идентичны, поэтому 4 тест также успешно пройден.

Таким образом, можно сделать вывод, что программы without_macros.s и refactor.s корректно работают.

Итоги

По критериям для оценки 6 нужно:

1. Решение задачи на языке C
2. Откомпилированная программа
3. Откомпилированная программа без лишних макросов
4. Используемые опции компиляции и описание модификаций
5. Использовать функции и локальные переменные
6. Рефакторинг программы за счет максимального использование регистров
7. Описание изменений в этой программе
8. Последняя версия(refactor.s) должна быть прокомментирована.
9. Представление тестового покрытия и одинаковые результаты тестов

В проделанной работе выполнены все данные пункты:

1. main.c
2. main.s
3. without_macros.s
4. Описано в "Реализация и описание отдельных файлов" и "Сравнение программ".
5. Выполнено и описано в комментариях к refactor.s.
6. refactor.s.
7. "Сравнение программ" и комментарии к коду.
8. Выполнено.
9. Описано в "Тестовые прогоны", а сами тесты вместе с результатами лежат в репозитории.