

ИДЗ-2 Вариант 2

Лобанов Кирилл Сергеевич БПИ-213

Ноябрь 2022

Условие задачи - Разработать программу, вычисляющую с помощью степенного ряда да с точностью не хуже 0.001 значение функции гиперболического синуса $sh(x) = (e^x - e^{-x})/2$ для заданного параметра x . счисления. Файл `avs.c` - программа на Си, `avs_asm.S` - изначальная программа на Ассемблере, `avs_reg.S` - версия с регистрами, `avs_full.S` и `avs_compute.S` - версия с регистрами, разбитая на 2 единицы компиляции.

4 балла) Была написана программа на языке С `avs.c`, затем она была собрана с помощью флагов `gcc -masm=intel -fno-asynchronous-unwind-tables -fno-jump-tables -fno-stack-protector -fno-exceptions ./avs.c -S -o ./avs_asm.S` После чего был получен ассемблерный код `avs_asm.S`, к нему были добавлены необходимые комментарии, затем была проверена корректность работы программ с помощью тестов из папки `tests`.

5 баллов) Программа `avs.c` была изначально разбита на функции `main`, `compute` и `write`, которые считывали исходное число x , вычисляли $sh(x)$ и выводили его, все переменные были локальные и параметры передавались в функции, поэтому код на С и на ассемблере не поменялся.

6 баллов) В функции `compute`, которая самая большая по времени работы, потому что мы сами задаем число её повторений, были использованы регистры `r12`, `edi`, `xmm5` и `xmm6`, что позволило полностью отказаться от использования в этой функции стека. Так как значения этих регистров не нужны после завершения этой функции, нам не важно - затираются они или нет. Замена не сделала код меньше по объему, но сильно ускорила, потому что обращение к памяти дольше, чем к регистрам. Более точную оценку ускорения можно увидеть ниже.

7-8 баллов) Код на ассемблере был разбит на 2 части - в одной функция `compute`, в другой все остальное: получилось 2 файла `avs_compute.S` и `avs_full.S`, затем они оба были преобразованы в машинный код - получили файлы `avs_compute.o` и `avs_full.o`, затем они слинковывались в исполняемый файл `a.out` командой `gcc avs_compute.o avs_full.o` Был сделан файловый ввод/вывод и возможность генерации случайных тестов, выбор пользователю для генерации/ввода предоставляется с помощью командной строки - для генерации надо первым параметром ввести `-random`, для ввода - название файла или его путь, для вывода в файл всегда надо третьим параметром передавать его название или путь. Второй параметр отвечает за число повторений в цикле, чтобы было удобнее замерять время работы программы.

В итоге время работы программы при $x = 10$ без регистров:

1000000 повторений - 0.06 секунды

10000000 повторений - 0.617 секунды

100000000 повторений - 6.095 секунды

После добавления регистров вместо обращения к памяти:

1000000 повторений - 0.04 секунды

10000000 повторений - 0.392 секунды

100000000 повторений - 3.924 секунды

Из этих экспериментов видно, что разница во времени составляет от 33 до 36 процентов, что является очень хорошим ускорением.