Отчет по ИДЗ №2 по Архитектуре вычислительных систем.
Вариант 4.
Харитонов Кирилл БПИ 214

Вариант 4. Разработать программу, находящую в заданной ASCII-строке последнюю при перемещении слева направо последовательность N символов, каждый элемент которой определяется по условию «больше предшествующего» (N вводится как отдельный параметр).

Претендую на оценку 9.

- Тесты, демонстрирующие проверку программ содержатся в Makefile по директивам test_default и test_assembly.
- Результаты их прогонов:
  test_default:
  ./default < input3
  Enter N:
  Enter string:
  Answer is: 0uw
  Time taken to execute this program: 0.000001
  ./default < input4
  Enter N:
  Enter string:
  Answer is: 7cny
  Time taken to execute this program: 0.000000
  ./default -f 3:input1:output1
  Time taken to execute this program: 0.000003
  cat output1
  Answer is: 0uw
  ./default -f 4:input2:output2
  Time taken to execute this program: 0.000002
  cat output2
  Answer is: 7cny
  ./default -r 2:7:1
  Following string was generated: Nf$Nq^v
  Answer is: ^v
  Time taken to execute this program: 0.000000
  ./default -r 3:100:10000
  Following string was generated:
  Nf$Nq^vo3d\m0;S!3;>/E%~{??(n{lnIRRXC1Ns$2/r"JE#]@AleGk`FjH4etb/G4GJ%u} h-\|jW!nuAoafvLF<wO0\CrkJgrUlH*
  Answer is: Jgr
  Time taken to execute this program: 0.000098

  test_assembly:

./assembly < input3
Enter N:
Enter string:
Answer is: 0uw
Time taken to execute this program: 0.000001
./assembly < input4
Enter N:
Enter string:
Answer is: 7cny
Time taken to execute this program: 0.000001
./assembly -f 3:input1:output1
Time taken to execute this program: 0.000003
cat output1
Answer is: 0uw
./assembly -f 4:input2:output2
Time taken to execute this program: 0.000002
cat output2
Answer is: 7cny
./assembly -r 2:7:1
Following string was generated: [@\g;nx
Answer is: nx
Time taken to execute this program: 0.000000
./assembly -r 3:100:10000
*Following string was generated: [@\g;nx□n!F,VDCt-V3w*g tMF*
*0N(u)(1P$□(#MjJY@n]4\3H3}o3Q=Y2-':#p#4 'ti*AST[sC8g_+orh_%:|^L)EFluIauP5^*
Answer is: Iau
Time taken to execute this program: 0.000064

- Исходные тексты программы на С содержатся в файлах main.c и functions.c
- Исходные тексты программы на ассемблере содержатся в файлах main.s и functions.s
- Текст на ассемблере программы, полученный после компиляции программы на С представлен в файлах main_original.s и functions_original.s

Критерии:
4 балла

- Исходные тексты программы на С **содержатся в файлах main.c и functions.c**
- Добавлены комментарии, представляющие эквивалентное представление переменных в программе на ассемблере. **Например,**

  mov DWORD PTR -28[rbp], 1 # amount = 1
  mov DWORD PTR -4[rbp], 1 # iterations = 1
- Из ассемблерной программы убраны лишние макросы за счет использования соответствующих аргументов командной строки (откомпилирована с аргументами -S -masm=intel -fno-asynchronous-unwind-tables -fno-jump-tables -fno-stack-protector -fno-exceptions -O0) и/или за счет ручного редактирования исходного текста ассемблерной программы. **(например удалена метка .name и информация о ОС, на которой была скомпилирована программа)**
- Модифицированная ассемблерная программа отдельно откомпилирована и скомпонована без использования опций отладки. **(директива assembly в Makefile и исполняемый файл assembly)**

- Представлено полное тестовое покрытие, дающее одинаковый результат на обоих программах. Приведены результаты тестовых прогонов для обоих программ, демонстрирующие эквивалентность функционирования. **(есть на первой странице отчета)**

5 баллов

- В реализованной программе использованы функции с передачей данных через параметры. **Например, void findLength(char string[1000000], int* length).**
- Использованы локальные переменные. **Например, int i = length - 2.**
- В ассемблерную программу при вызове функции добавлены комментарии, описывающие передачу фактических параметров и перенос возвращаемого результата. **Например,**

  lea r8, -100[rbp] # кладем указатель на optionIndex в r8
  mov rsi, QWORD PTR -1000160[rbp] # rsi = argc
  mov edi, DWORD PTR -1000148[rbp] # edi = argv
  lea rcx, longOptions.0[rip] # кладем указатель на структуру longOptions в rcx
  lea rdx, .LC0[rip] # кладем указатель на строку "f:r:" в rdx
  call getopt_long@PLT # вызываем getopt_long(argc, argv, "f:r:", longOptions, &optionIndex)
- В функциях для формальных параметров добавлены комментарии, описывающие связь между параметрами языка Си и регистрами (стеком). **Например,** *findAnswer: # void findAnswer(char string[1000000], int amount, int* position, int* flag, int length, int iterations, int n)*
  push rbp # пролог
  mov rbp, rsp
  mov r14, rdx # r14 = position
  mov r15, rcx # r15 = flag

6 баллов

- Рефакторинг программы на ассемблере за счет оптимизации использования регистров процессора. (файлы main.s и functions.s). **Пример проведенной оптимизации:**

  Было:

```
findLength:
 endbr64
 push   rbp
 mov    rbp, rsp
 mov    QWORD PTR -24[rbp], rdi
 mov    QWORD PTR -32[rbp], rsi
 mov    DWORD PTR -4[rbp], 0
 jmp    .L2
.L5:
 mov    eax, DWORD PTR -4[rbp]
 movsx  rdx, eax
 mov    rax, QWORD PTR -24[rbp]
 add    rax, rdx
 movzx  eax, BYTE PTR [rax]
 test   al, al
 jne    .L3
 mov    rax, QWORD PTR -32[rbp]
```

```
  mov    edx, DWORD PTR -4[rbp]
  mov    DWORD PTR [rax], edx
  jmp    .L4
.L3:
  add    DWORD PTR -4[rbp], 1
.L2:
  cmp    DWORD PTR -4[rbp], 999999
  jle    .L5
  nop
.L4:
  nop
  pop    rbp
  ret
```

Стало:

```
findLength: # void findLength(char string[1000000], int* length)
  push   rbp # пролог
  mov    rbp, rsp
  mov    r12, 0 # i = 0
  jmp    .L2
.L5:
  mov    eax, r12d # eax = i
  movsx  rdx, eax
  mov    rax, rdi # string
  add    rax, rdx # string + i
  movzx  eax, BYTE PTR [rax] # eax = string[i]
  test   al, al # string[i] == 0
  jne    .L3 # если !=
  mov    rax, rsi # rax = length
  mov    edx, r12d # edx = i
  mov    DWORD PTR [rax], edx # length = i
  jmp    .L4
.L3:
  add    r12d, 1 # i++
.L2:
  cmp    r12d, 999999 # i ? 999999
  jle    .L5 # i <= 999999
.L4:
  pop    rbp # эпилог
  ret
```

- Добавление комментариев в разработанную программу, поясняющих эквивалентное использование регистров вместо переменных исходной программы на C. **(можно увидеть в примере из предыдущего критерия)**
- Представление результатов тестовых прогонов для разработанной программы. Оценка корректности ее выполнения на основе сравнения тестовых прогонов результатами тестирования программы, разработанной на языке C. **(есть на первой странице отчета)**

- Сопоставление размеров программы на ассемблере, полученной после компиляции с языка С с модифицированной программой, использующей регистры. **Так, размер программы полученной после компиляции программы на С 502 строки, размер модифицированной программы на ассемблере 405 строк.**

7 баллов

- Реализация программы на ассемблере в виде двух или более единиц компиляции (программу на языке С разделять допускается, но не обязательно). **Так, программа на С разделена на файлы main.c и functions.c, аналогично и программа на ассемблере.**
- Использование файлов с исходными данными и файлов для вывода результатов. Имена файлов задаются с использованием аргументов командной строки. Командная строка проверяется на корректность числа аргументов и корректное открытие файлов. **Так, для использования файлов с исходными данными есть ключ -f с аргументами N:input_file_name:output_file_name.**
- Подготовка нескольких файлов, обеспечивающих тестовое покрытие разработанной программы. **Это файлы input1, input2.**

8 баллов

- Использование в разрабатываемых программах генератора случайных наборов данных, расширяющих возможности тестирования. **Так, для использования генератора используется ключ -r с аргументами N:длина:число_повторов. Генератор создает строку заданной длины с символами с кодами от 32 до 127 и выполняет поиск в ней нужной последовательности заданное число раз.**
- Изменение формата командной строки с учетом выбора ввода из файлов или с использованием генератора. **Так, добавлен соответствующий ключ командной строки.**
- Включение в программы функций, обеспечивающих замеры времени для проведения сравнения на производительность. Необходимо добавить замеры во времени, которые не учитывают время ввода и вывода данных. Для увеличения времени работы минимум до 1 секунды, в зависимости от особенностей программы, можно либо выбирать соответствующие размеры исходных данных, либо зацикливать для многократного выполнения ту часть программы, которая выполняет вычисления. В последнем случае можно использовать соответствующую опцию командной строки, задающей количество повторений. **Так, с помощью функций библиотеки time.h, вычисляется длительность выполнения программы, эти данные можно увидеть на первой странице отчете. Для увеличения времени работы времени программы до 1 секунды используется следующий тест: ./assembly -r 5:999999:5000000. При его выполнении время выполнения программы составило 1.867588 секунд.**

9 баллов

- Используя опции оптимизации по скорости, сформированы из программы на С исходный код ассемблере. Провести сравнительный анализ оценки с ассемблерной программой без оптимизации по размеру ассемблерного кода, размеру исполняемого файла и производительности. Сопоставить эти программы с собственной программой, разработанной на ассемблере, в которой вместо переменных максимально возможно используются регистры.

Так, с использованиям опции компиляции -O3 были скомпилированы файлы main_O3.s и functions_O3.s. Суммарный размер этих ассемблерных файлов 506, размер исполняемого файла 17.1 kb, производительность тестируется на тесте ./O3 -f 3:test_speed:test_speed_output. Скорость выполнения программы: 0.000005 секунд. Размер main_original.s и function_original.s (скомпилированы без оптимизаций) 502 строки суммарно, размер исполняемого файла 17.0 kb, производительность на аналогичном тесте 0.000011 секунд. Размер main.s и function.s (вручную отредактированы) 405 строк, размер исполняемого файла 16.9 kb, производительность 0.000009 секунд. Получается, что программа с оптимизациями по скорости имеет больший размер кода и исполняемого файла, но является более производительной относительно остальных.

- Аналогично, используя опции оптимизации по размеру, сформировать код на ассемблере. Провести сравнительный анализ с неоптимизированной программой по размеру ассемблерного кода, размеру исполняемого файла и производительности. Сопоставить эти программы с собственной программой, разработанной на ассемблере, в которой вместо переменных максимально возможно используются регистры. **Так, с помощью опции компиляции -Os были получены файлы с ассемблерным кодом main_Os.s и functions_Os.s и исполняемых файл Os. Размер файлов main_Os.s и functions_Os.s 453 строки, размер исполняемого файла 17.0 kb, производительность на аналогичном тесте 0.000010 секунд. Данные о других программах приведены в предыдущем критерии. Получается, что оптимизация по размеру с помощью опции компилятора, проигрывает ручному редактированию по размеру исполняемого файла, размеру ассемблерного кода и производтельности, но превосходит по всем параметрам программу скомпилированную без оптимизаций.**