«Национальный исследовательский университет

«Московский институт электронной техники»

Лабораторная работа №2 по дисциплине «Архитектура вычислительных систем»

Отладка кода

Выполнили студены группы МП-25

Саядян Артём Грачикович

Калинкин Никита Анатольевич

Констандогло Александр Витальевич

**Москва 2017**

Вариант №2

**Задание 1.** Разработайте программу на языке C++, вычисляющую три целых выражения от целого аргумента (в соответствии с вариантом).

а) *y(x) = −x − 1*

б) *y(x) = x \* 13*

в) *y(x) =*

**Задание 2.** Запустите программу и, используя инструменты отладчика (в частности, дизассемблер), изучите ассемблерный код, соответствующий вычислениям.

Занесите ассемблерный код, соответствующий вычислению *y(x)*, в отчёт (код, не связанный с вычислением *y(x)*, копировать в отчёт не нужно!). Определите и прокомментируйте:

* обращение к переменным *x* и *y*;
* арифметические и логические операции — сложение, вычитание, умножение, деление с остатком, деление на *2n* и т. д. (по возможности);
* сравнения и передачу управления в ветвлениях.

**Результат:**

*Platform: Linux Ubuntu 16.04.3 LTS x86\_64*

*Compiler: GNU GCC 5.3.1*

*IDE: Qt Creator 4.3.1*

а) y = -x-1;

mov -0x14(%rbp),%eax // eax (32 бита) = \*(rbp – 0x14) (bp адресует переменные, хранимые в стеке, 64 бита, так как это длина слова, в данном случае из стека извлекается x)

not %eax // побитовое отрицание x, это и есть -x-1

mov %eax,-0x4(%rbp) // \*(rbp – 0x4) = eax – запись результата обратно в стек

б) y = x\*13;

mov -0x14(%rbp),%edx // edx = \*(rbp – 0x14) (извлекается значение x)

mov %edx,%eax // eax = edx (x)

add %eax,%eax // eax = eax + eax (2x)

add %edx,%eax // eax = eax + edx (3x)

shl $0x2,%eax // eax = eax << 0x2 логический сдвиг влево на 2 (12x)

add %edx,%eax // eax = eax + edx (13x – получили то, что нужно)

mov %eax,-0x4(%rbp) // \*(rbp – 0x4) = eax – запись результата обратно в стек

в) y = (x<7)?0:x;

cmpl $0x6,-0x14(%rbp) // сраниваются значения 0x6 и \*(rbp-0x14)(x)

jle 0x400709 <ex1c(int)+18> // если текущий элемент массива (\*(rbp-0x14), то есть значение x) меньше или равен 6, переходим на нужный участок памяти

mov -0x14(%rbp),%eax // eax = \*(rbp – 0x14)извлекаем x в регистр А (если x<=6, сюда не доходим)

jmp 0x40070e <ex1c(int)+23> // безусловно переходим на нужный участок (если x<=6, сюда не доходим)

mov $0x0,%eax // 0x400709 eax = 0 (если не(x<=6), сюда не доходим)

mov %eax,-0x4(%rbp) // 0x40070e \*(rbp – 0x4) = eax – запись результата обратно в стек

**Задание 3.** Внесите в программу из задания 1, а) изменения (либо, что предпочтительнее, добавьте новые фрагменты кода, выполняющие аналогичные вычисления для других переменных, используя макросы препроцессора или шаблоны C++).

* сделайте переменные глобальными;
* измените тип с int на char, short, long и long long;
* измените тип с int на long double.

Опишите в отчёте различия в ассемблерном коде.

**Результат:**

* сделайте переменные глобальными;

*Y = -X-1;* // X и Y — глобальные переменные

lea 0x2008ae(%rip),%rax # 0x601040 <X> // rax = \*(rip + 0x2008ae) - глобальные переменные расположены в конкретном участке памяти (адрес-смещение следующей команды (64 бита))

mov (%rax),%eax // eax = rax (сужаем область до размера int (младшие байты))

not %eax // вычисление

mov %eax,%edx // edx = eax

lea 0x2008a9(%rip),%rax # 0x601048 <Y> // rax = \*(rip + 0x2008a9)

mov %edx,(%rax) // rax = edx – запись результата

* измените тип с int на char, short, long и long long;

**char:** y = -x-1;

movzbl -0x14(%rbp),%eax // чтение байта по адресу \*(rbp – 0x14) и его расширение до 32 бит путём добавления нулей в 3 старших байта

not %eax

mov %al,-0x1(%rbp) // \*(rbp – 0x1) = al – запись младшего байта регистра eax обратно в стек

**short:** y = -x-1;

movzwl -0x14(%rbp),%eax // чтение 2 байт по адресу \*(rbp – 0x14) и их расширение до 32 бит путём добавления нулей в 2 старших байта

not %eax

mov %ax,-0x2(%rbp) // \*(rbp – 0x2) = ax – запись младших 2 байт регистра eax обратно в стек

**long:** y = -x-1; // Здесь используются все 64 бита регистра A

mov -0x18(%rbp),%rax

not %rax

mov %rax,-0x8(%rbp)

**long long:** y = -x-1; // как long

* измените тип с int на long double.

**long double:** y = -x-1;

fldt 0x10(%rbp) // загрузить временное вещественное (или, как его ещё называют, long double) в вершину стека из \*(rbp + 0x10)

fchs // изменить знак вершины стека

fld1 // загрузка 1 в вершину стека (1 является константой)

fsubrp %st,%st(1) // st(1) = st(0) - st(1)

fstpt -0x10(%rbp) // Переместить значение из st(0) в \*(rdp – 0x10) ??? (должен получиться 0) ???

**Задание 4.** Оформите вычисления из задания 1, а) как целую функцию от целого аргумента. Опишите в отчёте код вызова функции. Как передаётся аргумент? Как возвращается значение?

**Результат:**

*ex1a(1);* // Вызов

mov $0x1,%edi // Передача аргумента: edi = 0x1

callq 0x4006c6 <ex1a(int)> // Передача управления функции с запоминанием в стеке адреса точки возврата

*return y;* // Возврат значения

mov -0x4(%rbp),%eax // В регистр общего назначения A записываются данные из стека

**Задание 5.** Измените тип аргумента и результата на вещественный. Опишите в отчёте код вызова функции. Как передаётся аргумент? Как возвращается значение?

**Результат:**

*ex5(999E-8);* // Вызов

movabs $0x3ee4f357252adccd,%rax // Запись 64-битового значения в регистр A

mov %rax,-0x18(%rbp) // \*(rbp – 0x18) = rax

movsd -0x18(%rbp),%xmm0 // Пересылка данных из \*(rbp – 0x18) в 128-битный SSE регистр xmm0

callq 0x400716 <ex5(double)> // Передача управления функции с запоминанием в стеке адреса точки возврата

*return y;* // Возврат значения

movsd -0x8(%rbp),%xmm0 // В регистр xmm0 записываются данные из стека

**Задание 6.** Используйте в функции статическую переменную. Как выглядит обращение к ней?

**Результат:**

*y = -x-1;* // y – статическая переменная

mov -0x4(%rbp),%eax

not %eax

mov %eax,0x2008f6(%rip) # 0x601050 <\_ZZ3ex6iE1y>

// К статическим переменным обращение происходит как к глобальным, так как это и есть глобальные переменные для данной функции

**Задание 7.** Запустите тестовую программу (программы), используя платформу и/или компилятор, отличные от GNU/Linux и GCC. Результаты с пояснениями внести в конспект.