Ассемблер: Домашняя работа 3

ИТМО, КТ

April 2, 2015

1 Задание

Третья домашняя работа посвящена программированию на языке ассемблера в 64-битном режиме и использованию векторных расширений набора инструкций.

Вам нужно написать библиотеку для работы с длинными целыми числами. Должны поддерживаться операции сложения, вычитания, умножения, сравнения и взятия знака числа. Также должна быть возможность создавать длинные числа из обычных целых и из строк с десятичной записью числа.

Операция деления необязательна, но принесет вам дополнительные пять баллов, если вы ее реализуете.

Еще одна необязательная функция – вывод числа в строку. За ее реализацию тоже дается пять баллов.

Итого вы сможете набрать за это задание максимум 25 баллов (без учета бонусов).

Вы должны сами выбрать, как хранить большие числа. Не забудьте описать это в комментариях к коду.

1.1 Функции

1.1.1 Создание длинных чисел

```
BigInt biFromInt(int64_t x);
```

Создание из 64-битного целого.

```
BigInt biFromString(char const *s);
```

Создание из строки с десятичным числом. Строка должна удовлетворять регулярному выражению `^-?\d+$`. Если строка не удовлетворяет формату, то функция должна вернуть NULL.

1.1.2 Удаление длинных чисел

`void biDelete(BigInt bi);`
Удаляет ранее созданное число.

1.1.3 Базовая арифметика

`void biAdd(BigInt dst, BigInt src);`
Сложение. К *dst* прибавляется *src*, результат помещается в *dst*.

`void biSub(BigInt dst, BigInt src);`
Вычитание. Из *dst* вычитается *src*, результат помещается в *dst*.

`void biMul(BigInt dst, BigInt src);`
Умножение. *dst* умножается на *src*, результат помещается в *dst*.

`int biCmp(BigInt a, BigInt b);`
Сравнение. Возвращает ноль, если $a = b$, отрицательное число – если $a < b$, положительное число – если $a > b$.

`int biSign(BigInt bi);`
Сравнение. Возвращает ноль, если $bi = 0$, отрицательное число – если $bi < 0$, положительное число – если $bi > 0$.

1.1.4 Деление

`void biDivRem(BigInt *quotient, BigInt *remainder, BigInt numerator, BigInt denominator);`
Деление с остатком. Если *denominator* не ноль, то выполняет деление *numerator* на *denominator*, создает два новых числа (*quotient* и *remainder*) и помещает в них частное и остаток от деления соответственно.

Остаток от деления всегда находится в интервале $[0, denominator)$, если $denominator > 0$, и $(denominator, 0]$, если $denominator < 0$.

Частное и остаток должны удовлетворять формуле:
$quotient * denominator + remainder = numerator$

Если *denominator* ноль, то функция должна поместить NULL в quotient и remainder.

Эта функция не является обязательной. Если вы не собираетесь ее реализовывать, то сделайте заглушку, которая не делает ничего (не трогает значения *quotient* и *remainder*).

1.1.5 Вывод числа в строку

```
void biToString(BigInt bi, char *buffer, size_t limit);
```

В результате работы функции должна получиться строка, содержащая текстовое представление числа в том же формате, какой требуется для *biFromString* (сначала минус, если число отрицательное, затем цифры). Если число не ноль, то ведущих нулей быть не должно. Если число ноль, то минуса быть не должно. Функция должна помещать не более *limit* символов (включая завершающий ноль) в строку. Если строка не помещается в лимит, то в последний символ строки нужно записать ноль (число ноль, а не символ `0`), чтобы функции работы со строкой не вышли за ее пределы.

Эта функция не является обязательной. Если вы не собираетесь ее реализовывать, то сделайте заглушку, которая запишет в строку `"NA"`. Не забудьте завершающий ноль. Лимит в этом случае можно не проверять.

2 Оформление задания

2.1 Репозиторий

Репозиторий находится по адресу `https://github.com/itmoasm2015/Homework3`.

В папке `include` находится заголовочный файл с прототипами функций, которые вам нужно реализовать.

Свое решение кладите в папку с вашей фамилией внутри папки с номером вашей группы.

2.2 Технические детали

Для операций с целыми числами используйте как стандартный набор инструкций, так и векторные расширения системы команд (MMX, SSE2,

SSE3, SSSE3, SSE4, AVX). По возможности используйте векторную арифметику.

Не забывайте сохранять регистры, которые требуется сохранять согласно конвенции вызова (это RBP, RBX, R12-R15). Также перед вызовом внешних функций стек должен быть выровнен на 16 байт.

Не забывайте, что при программировании в 64-битном режиме нужно использовать RIP-относительную адресацию, а не адресацию по абсолютным адресам, так как машинный код позволяет закодировать только 32-битный абсолютный адрес. Чтобы использовать RIP-относительную адресацию, укажите в начале ассемблерного файла директиву `default rel`, и yasm автоматически сгенерирует относительные адреса.

2.3 Результат

Результатом вашей работы должна стать статическая библиотека с именем **libhw.a**, в которой находятся требуемые функции. Библиотека может ссылаться на внешние символы (вам понадобится, например, malloc и free для выделения памяти). Библиотека должна успешно линковаться в 64-битном режиме.

В репозитории (в папке с вашей фамилией) должен лежать Makefile, при сборке с помощью которого (командой `make`) должна получаться требуемая библиотека (в той же папке).

Статическую библиотеку можно создать командой

```
ar rcs libhw.a object1.o object2.o ...
```

2.4 Сдача задания

После того как вы напишете свой код и закоммитите его в репозиторий, создавайте *issue*, в котором указываете свою фамилию, номер группы и название папки с вашим решением. Указываете в качестве *assignee* меня (мой ник на гитхабе imihajlow).

Как только я проверю ваше задание, я или закрою *issue*, если задание принято, или напишу к нему комментарий по поводу исправлений. Датой сдачи задания считается день создания или последнего изменения вами *issue*, после которого я его закрыл.

Срок сдачи задания – 10 мая. При сдаче до 10 мая включительно вы получите баллы полностью, а после 10 мая с коэффициентом 0,6.

2.5 Оценивание

За сданное задание вы получите максимум 25 баллов, из них 7 ставится за комментарии к коду, поэтому обратите особое внимание на читаемость и понятность вашей программы.

Удачи!