Ассемблер: Домашняя работа 1

ИТМО, КТ

February 20, 2015

1 Задание

Первая домашняя работа посвящена программированию на языке ассемблера в 32-битном режиме.

Вам нужно написать урезанный аналог функции `sprintf`:

```
void hw_sprintf(char *out, char const *format, ...);
```

1.1 Параметры функции

Параметры аналогичны параметром sprintf:

1. `char *out` – указатель на буфер, в который помещается результат;
2. `char const *format` – строка форматирования;
3. Далее идут остальные параметры в зависимости от строки форматирования.

Строки являются обычными строками C, то есть массивами байт, оканчивающимися нулями.

Строка форматирования содержит обычные символы вперемешку с управляющими последовательностями. Обычные символы и некорректные управляющие последовательности выводятся как есть.

1.2 Управляющие последовательности

Управляющая последовательность имеет следующий вид:

`%[флаги][ширина][размер]тип`

1.2.1 Флаги

Флаг	Назначение
+	всегда указывать знак числа (+ или -)
пробел	помещать перед числом пробел, если первый символ не знак
-	выравнивать значение по левому краю в пределах минимальной ширины (иначе по правому)
0	дополнять до минимальной ширины нулями (иначе пробелами)

Флаг 0 игнорируется, если указан одновременно с -.

1.2.2 Ширина

Если указана, обозначает минимальную ширину поля, в которое выводится число.

1.2.3 Размер

Обозначает размер данных.

Спецификатор	Значение
отсутствует	4 байта (int)
`ll`	8 байт (long long int)

1.2.4 Тип

Тип может быть только знаковым десятичным целым (**i** или **d**) или беззнаковым десятичным целым (**u**). Также можно выводить знак % при помощи типа %.

1.3 Примеры

```
hw_sprintf(out, "Hello world %d", 239)
"Hello world 239"

hw_sprintf(out, "%+5u", 51)
"  +51"

hw_sprintf(out, "%8u=%-8u", 1234, 1234)
"    1234=1234    "
```

```
hw_sprintf(out, "%llu", (long long)-1)
"18446744073709551615"

hw_sprintf(out, "%wtf", 1, 2, 3, 4);
"%wtf"

hw_sprintf(out, "50%%");
"50%"
```

2 Оформление задания

2.1 Репозиторий

Репозиторий находится по адресу `https://github.com/itmoasm2015/Homework1`.

В папке `include` находится заголовочный файл с прототипом функции, на него нужно ориентироваться при написании кода.

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

2.2 Технические детали

Требуется написать на ассемблере функцию, которая удовлетворяла бы конвенции вызова cdecl, принятой в Linux (компилятор gcc).

Результатом вашей работы должна стать статическая библиотека с именем **libhw.a**, в которой находится требуемая функция. Библиотека не должна ссылаться ни на какие внешние символы, то есть использовать стандартные библиотеки не разрешается. Библиотека должна успешно линковаться в 32-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 ...
```

2.3 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.

Напоминаю, что при сдаче до 15 марта включительно вы получите баллы полностью, а после 15 марта с коэффициентом 0,6.

2.4 Оценивание

За сданное задание вы получите максимум 15 баллов, из них 7 ставится за комментарии к коду, поэтому обратите особое внимание на читаемость и понятность вашей программы.

Правила комментирования те же, что и в других языках программирования: не описывать очевидные вещи, а описывать поведение участков кода в целом. Пишите комментарии или полностью на русском языке, или полностью по-английски, не смешивайте. Если делаете какие-то функции для внутреннего использования, обязательно в комментариях пишите, что это именно функция (а не просто метка для перехода), что она делает, как она принимает параметры, как возвращает значение, какие регистры сохраняет и т. п.

Не забывайте также называть метки по-человечески и использовать локальные метки там, где они нужны (начинаются с точки, см. документацию yasm).

Удачи!