Федеральное государственное автономное образовательное учреждение высшего образования

САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

ФАКУЛЬТЕТ ПРОГРАММНОЙ ИНЖЕНЕРИИ И КОМПЬЮТЕРНОЙ ТЕХНИКИ

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №2

«Языки системного программирования: Использование ассемблерных вставок в языках высого уровня»

Проверил:
Сентерев Ю.А. _____
«_____» _____ 201_г.

Выполнил:
Студент группы P3255
Кабардинов Д. В. _____

Оценка _____

Санкт-Петербург

2019

Задание:
Составить программу на языке C, использующую ассемблерную вставку

Цель работы:
Рассмотреть на примере языка C, как можно использовать ассемблерные вставки в программах, написанных на языках высокого уровня

Ход выполнения работы
Ассемблерные вставки используются для «насильственного» размещения в Си - программах ассемблерного кода, явно заданного программистом. Содержимое ассемблерной вставки никак компилятором не анализируется, но имеется возможность описать то, как это содержимое взаимодействует с переменными Си - программы и как изменятся регистры после выполнения этого ассемблерного кода.
Синтаксис оператора ассемблерной вставки следующий:
__asm__ (вставка : ***список_выходных_операндов*** : ***список_входных_операндов*** : ***список_разрушаемых_регистров***);
Начинается оператор ассемблерной вставки с ключевого слова *asm* или __asm__, после чего в круглых скобках следует ее описание. вставка представляет собой строковую константу с ассемблерными инструкциями.
В теле вставки могут находиться не только ассемблерные инструкции, но и вообще любые директивы, распознаваемые ассемблером gas. В частности, это позволяет изменить используемый им синтаксис инструкций по-умолчанию.

Пример 1

```
__asm__ (  ".intel_syntax noprefix\n\t"
           "mov eax, %1\n\t"
           "mov %0, eax\n\t"    /* ассемблерная вставка */
         : "=r"(b)              /* выходные операнды */
         : "r"(a)               /* входные операнды */
         : "%eax"               /* разрушаемые регистры */
);
```

Директива .intel_syntax меняет синтаксис AT&T на синтаксис Intel; необходимо дополнительно указывать смену синтаксиса для операндов инструкций, **noprefix**, что позволит писать код в более близком к диалекту nasm виде, не используя при записи имен регистров префикс Допустима сокращенная версия вставки, состоящая только из строки с командами:

```
asm("hlt\n\t");
```

Для связи ассемблерных инструкций с переменными Си-программы используются следующие за вставкой два элемента оператора: списки операндов, в которых операнды перечислены через запятую. Каждый описанный операнд затем может использоваться в ассемблерных инструкциях, обращение к нему осуществляется по номеру с префиксом %. Нумерация начинается с 0, и идет непрерывно, объединяя все элементы списков выходных и входных операндов.
1. Напишем простую программу для сложения двух чисел, с использованием ассемблерной вставки

```
int main(void)
{
  int foo = 10, bar = 15;
  __asm__ __volatile__("addl  %%ebx,%%eax"
                       :"=a"(foo)
                       :"a"(foo), "b"(bar)
                       );
  printf("foo+bar=%d\n", foo);
  return 0;
}
```

Здесь мы настраиваем GCC для записи значения из переменной foo в регистр %eax, a bar в %ebx, а результат будет в регистре %eax. Знак '=' указывает, что это выходной регистр.
2. Теперь рассмотрим более сложную, но полезную функцию.

Копирование строки.

```
static inline char * strcpy(char * dest,const char *src)
{
int d0, d1, d2;
__asm__ __volatile__("1:\tlodsb\n\t"
                     "stosb\n\t"
                     "testb %%al,%%al\n\t"
                     "jne 1b"
                     : "=&S" (d0), "=&D" (d1), "=&a" (d2)
                     : "0" (src),"1" (dest)
                     : "memory");
return dest;
}
```

Адрес источника записывается в esi, приёмника - в edi, а затем начинается копирование. Когда мы считываем 0, копирование завершается. Ограничители "&S "&D "&a"указывают, что регистры esi, edi и eax являются ранними используемыми регистрами, то есть их содержимое будет изменено перед завершением функции. Здесь также понятно, почему память в списке используемых.

Вывод

В результете выполнения работы я узнал, как осуществляются ассемблерные вставки в языки высокого уровня, реализовав две простые программы на языке C, наглядно демонстрирующие работу с ассемблером. Таким образом, цель работы достигнута.

Список используемой литературы:

1. `http://av-assembler.ru/asm/high-level-languages/assembler-gcc.php` - Встроенный ассемблер GCC
2. `http://www.ibiblio.org/gferg/ldp/GCC-Inline-Assembly-HOWTO.html` - GCC-Inline-Assembly-HOWTO
3. `http://asmcourse.cs.msu.ru/` - Архитектура ЭВМ и язык ассемблера