Лабораторная работа 4 (0100 = 4)

Использование ассемблерных вставок в программах на C++. Команды пересылки

Версия 2024 г. — засчитывается только в 2024 г. Актуальная версия — в `https://gitlab.com/illinc/gnu-asm`

Цель работы: научиться вставлять в программы на языке высокого уровня ассемблерные фрагменты. Ознакомиться с командами пересылки данных.

Данная лабораторная работа, как и все предыдущие и последующие, должна собираться **компилятором из коллекции GCC** (для C++ — компилятор `g++`, для чистого C — `gcc`). Среда Microsoft Visual Studio не поддерживает других компиляторов, кроме Microsoft, и не может быть использована. Задания выполняются в виде ассемблерных вставок в программу на C/C++.

Штрафы: -1 балл за одно пропущенное обязательное задание, $-\frac{1}{2}$ балла за каждую некорректную секцию перезаписываемых элементов (clobbers).

Задание на лабораторную работу

Задание Л4.з1. Как в задании Л1.з4, создайте массивы Ms из 16-битных целых чисел, Ml из 32-битных целых чисел, Mq из 64-битных целых чисел (длина и начальные значения аналогичны Л1.з4).

Реализуйте для каждого массива M вставку, записывающую непосредственное значение 16 в $M[i]$ для заданного $i \in [0, N)$ с использованием команды mov, где выражение $M[i]$ является выходным параметром вставки *в памяти.* Так как оба операнда mov здесь не имеют определённого размера (непосредственное значение и память), необходимо указывать для mov суффикс размера: $movw$, $movl$, $movq$.

Здесь и далее все целочисленные массивы до и после изменения выводите в шестнадцатеричном представлении.

Задание Л4.з2. Реализуйте для одного из массивов M (по варианту согласно таблице Л4.1) вставку, записывающую непосредственное (-1) в $M[i]$, где адрес начала массива M и индекс i передаются как входные параметры *в регистрах.*

Варианты целочисленного массива M

Таблица Л4.1

$(№ - 1)\%3 + 1$	**Вариант**
1	Ms
2	Ml
3	Mq

Используйте компоненты эффективного адреса $(Base, Index, 2^{Scale})$. Разрядность компонент $Base$ и $Index$ должна быть одинаковой, поэтому для переносимости вставки необходимо объявить переменную i не как int (4 байта как для 32-, так и для 64-битного режимов), а как $size\_t$ (размер равен размеру указателя).

Задание Л4.з3. Реализуйте вставку, записывающую непосредственное значение `0xBB` в заданный байт $Mq[i]$ (по варианту согласно таблице Л4.2; младший байт считайте нулевым) с использованием одной команды mov ($movb$) и всех компонент эффективного адреса $Disp(Base, Index, 2^{Scale})$; адрес начала массива Mq и индекс i передаются как входные параметры *в регистрах.*

Варианты перезаписываемого байта $Mq[i]$ для Л4.з3

Таблица Л4.2

$(№ - 1)\%5 + 1$	**Вариант**
1	Первый байт после младшего
2	Третий байт
3	Четвёртый байт
4	Шестой байт
5	Седьмой байт (старший байт 64-битного $Mq[i]$)

Задание Л4.з4. Реализуйте вставку, записывающую в $M[i]$ значение x (M по варианту согласно таблице Л4.1; размер переменной x равен размеру элемента M), где значение x передаётся как входной параметр *в памяти,* M и i — как входные параметры *в регистрах.*

Так как команда x86 не может адресовать два операнда в памяти, прямая пересылка $x \to M[i]$ невозможна; используйте промежуточный регистр (таблица Л4.3).

Задание Л4.з5. Реализуйте вставку, записывающую в $M[i]$ значение x аналогично Л4.з4, но во вставку передаётся адрес $\&x$.

Задание Л4.з6. Реализуйте вставку, рассчитывающую для целочисленных x и y значения $z = x + y$ и $w = x - y$ при помощи команд add и sub. Разрядность указана в таблице Л4.4; переменные x, y, z, w передаются во вставку как параметры (z и w — выходные, x и y — входные).

Задание Л4.з7. Определите, доступны ли на выбранной платформе расширения AVX и SSE, используя команду $cpuid$ или документацию на процессор.

Как в задании Л1.з4, создайте массивы Mfl из 64-битных чисел с плавающей запятой и Mfs из 32-битных чисел с плавающей запятой.

Реализуйте вставку, записывающую в $M[i]$ значение x с плавающей запятой аналогично Л4.з4 (M по варианту согласно таблице Л4.5; размер переменной x равен

Варианты временного РОН

Таблица Л4.3

$(№ - 1)\%7 + 1$	**Вариант**
1	Регистр A ($al/ax/eax/rax$)
2	Регистр C ($cl/cx/ecx/rcx$)
3	Регистр D ($dl/dx/edx/rdx$)
4	Регистр si ($sil/si/esi/rsi$)
5	Регистр di ($dil/di/edi/rdi$)
6	Регистр $r8$ ($r8b/r8w/r8d/r8$) на 64-битной платформе, A на 32
7	Регистр $r9$ ($r9b/r9w/r9d/r9$) на 64-битной платформе, C на 32

Варианты разрядности x, y, z, w

Таблица Л4.4

$(№ - 1)\%2 + 1$	**Вариант**
1	64 бита
2	16 бит

размеру элемента M; x, M и i — параметры вставки), используя команды AVX $vmovsd/vmovss$ или их SSE-аналоги $movsd/movss$. Используйте промежуточ-

Варианты массива M из значений с плавающей запятой

Таблица Л4.5

$(№ - 1)\%2 + 1$	**Вариант**
1	Mfl
2	Mfs

ный регистр xmmi, где номер регистра $i \in [0, 5]$ рассчитывается как $(№ - 1)\%6$ (по варианту).

Задание Л4.з8. Реализуйте вставку, записывающую в $M[i]$ значение с плавающей запятой, равное целочисленному значению x. Преобразование целочисленного x к нужному виду выполните при помощи команд AVX $vcvtsi2sd/vcvtsi2ss$ или их SSE-аналогов $cvtsi2sd/cvtsi2ss$.

Л4.1. Дополнительные бонусные и штрафные баллы

−3 балла за утечку памяти (выделенные, но не освобождённые блоки динамической памяти).

Л4.2. Ссылки на теоретические сведения

4.3. Ассемблерные вставки в код C++
5.2. Основные команды

Л4.3. Вопросы

1. Какие вы знаете регистры общего назначения x86 и x86-64?
2. Какие вы знаете xmm-регистры x86 и x86-64?
3. Каким ключевым словом открывается ассемблерная вставка?
4. Где описываются выходные параметры ассемблерных вставок расширенного синтаксиса GCC? Что означают символы `=`, `=&`, `+` в начале строки ограничений выходного параметра?
5. Где описываются входные параметры?
6. Где указывается список перезаписываемых [clobbers] во вставке регистров (кроме параметров)? Какая строка соответствуют изменению флагов $flags$? Какая строка соответствуют изменению памяти (кроме параметров)?
7. Где описываются метки ЯВУ, на которые может быть передано управление из вставки?
8. Какое ключевое слово нужно указать после `asm`, чтобы запретить компилятору оптимизировать вставку?
9. Какое ключевое слово нужно указать после `asm`, чтобы показать, что управление из вставки может передаваться на ту или иную метку C/C++ (зато у этой вставки нет выходных параметров)?
10. Для чего используется команда mov и команды AVX/SSE $*mov*$?