Технологии программирования

Лабораторная работа №4

Тема: Программирование на ассемблере

Задание: Создать приложение, осуществляющее решение индивидуального задания с использованием ассемблерных вставок.

Язык ассемблера — низкоуровневый язык программирования, представляющий собой человеко-понятный способ представления машинных команд. Язык ассемблера является исторически первым из появившихся языков — он возник сразу после непосредственного программирования в машинных командах. Язык ассемблера является абсолютно непереносимым — из-за низкоуровневости, он жестко привязан к системе команд одного процессора (семейства процессоров).

В ассемблере один оператор языка преобразуется в процессе ассемблирования в одну машинную команду.

Из-за непосредственной связи языка ассемблера и системы команд процессора, операторы языка не являются универсальными, и являются тестовыми записями машинных команд. Чаще всего это аббревиатура, обозначающая то или иное машинное действие.

Например, команда пересылки данных (MOVE)

mov eax, 22

В языке ассемблера сначала указывается команды, затем идут операнды в порядке ОПЕРАНД-ПРИЕМНИК, ОПЕРАНД-ИСТОЧНИК (синтаксис Intel). Существует обратный вариант записи операндов (синтаксис AT&T).

Язык ассемблера в настоящее время применяется достаточно редко — существующие языки высокого уровня и их компиляторы достаточно удобны и эффективны. Ассемблерный код применяется в 3 основных случаях:

1. Время выполнения данного участка сверхкритично, и программист применяет ручное написание кода с какими-то неочевидными низкоуровневыми оптимизациями
2. Используются команды и режимы, неизвестные компилятору, скажем MMX, SIMD-расширения
3. Используются системные регистры, возможности и режимы процессора (создание ОС, системного и диагностического ПО)

Для решения задач 1-й и 2-й областей обычно не пишут программу или модуль программы на ассемблере целиком — это неэффективно, с точки зрения трудозатрат (эффективнее написать вручную только самые критичные участки кода, а остальное доверить компилятору — современный среднестатистический компилятор знает ассемблер заметно лучше современного среднестатистического программиста). В таком случае используют технологию ассемблерных вставок — в программу на языке высокого уровня включаются фрагменты на языке низкого уровня (ассемблере). Например, вычисление суммы двух чисел будет выглядеть так:

int Add(int a, int b)

{

\_\_asm {

mov eax, a

mov ebx, b

add eax, ebx

}

}

В данном случае используется синтаксис Intel. Тот же фрагмент на синтаксисе AT&T выглядит так:

int Add(int a, int b)

{

\_\_asm {

movl A, %eax

movl B, %ebx

addl %ebx, %eax

}

}

(Данный пример не будет работать с компилятором MSVC. Но применим, например, для компилятора gcc.)

В ассемблерных вставках вы можете использовать любые инструкции, символьные имена переменных и функций, а также любые регистры, но значения регистров EBP, ESP, а также сегментных регистров должны остаться неизменными на момент выхода из ассемблерной вставки. Для сохранения значения регистров можно использовать пару команд push/pop.

Обратите внимание, что использование ассемблерных вставок запрещено в x64 коде, а также в любом управляемом. Поэтому, для реализации задания вам потребуется создать дополнительный проект неуправляемой библиотеки C++, в которой вы и будете реализовывать функции задания с помощью ассемблерных вставок. Также вам следует помнить о соглашениях именования функции в C++/C и о способах передачи значений параметров в тело функции (соглашениях вызова).

Вариант 1.

1. Найти максимальное число в массиве
2. Подсчитать количество четных чисел в массиве
3. Найти сумму всех чисел в массиве (с учетом переполнения)
4. Сложить два массива поэлементно

Вариант 2

1. Найти минимальное число в массиве
2. Подсчитать количество нулей в массиве
3. Найти произведение всех чисел в массиве (с учетом переполнения)
4. Перемножить два массива поэлементно

Вариант 3

1. Найти число с максимальным количеством единичных разрядов в двоичной записи
2. Подсчитать количество чисел, больших среднего арифметического
3. Найти контрольную сумму массива по модулю 2 (XOR)
4. Поэлементно вычесть один массив из другого

Вариант 4

1. Найти среднее арифметическое массива
2. Подсчитать количество чисел, делящихся на 16
3. Вычислить количество единичных двоичных разрядов в массиве
4. Поэлементно разделить один массив на другой