ЛАБОРАТОРНАЯ РАБОТА №3
ВВЕДЕНИЕ В АРХИТЕКТУРУ x86/x86-64

Цели работы

1. Знакомство с программной архитектурой x86/x86-64.
2. Анализ ассемблерного листинга программы для архитектуры x86/x86-64.

1. КРАТКОЕ ОПИСАНИЕ АРХИТЕКТУРЫ x86

Архитектура x86 в настоящий момент является самой распространенной программной архитектурой настольных и серверных вычислительных систем. Она используется с 1978 года, когда был выпущен 16-разрядный процессор i8086. Впоследствии архитектура была естественным образом расширена до 32-битной (IA-32), а затем и до 64-битной (x64, AMD64, x86-64, IA-32e, EM64T, Intel 64). При каждом очередном расширении сохранялась программная совместимость с предыдущим поколением архитектуры. При этом набор команд расширялся, а новые регистры большего размера включали в качестве своих частей регистры предыдущего поколения архитектуры.

Регистры. Программисту доступны следующие группы регистров.

- регистры общего назначения,
- регистры сопроцессора,
- регистры векторных расширений,
- регистр флагов,
- счетчик команд.

Регистры общего назначения используются для хранения произвольных целочисленных данных. При использовании в большинстве команд в качестве операндов эти регистры полностью взаимозаменяемы, но некоторые команды используют только конкретные регистры. На рис. 1

представлены регистры общего назначения архитектур x86 и x86-64. У каждого регистра отдельно доступна младшая часть размером 8, 16 или 32 бита. Например, младшая часть регистров Rx доступна по именам RxD (32 бита), RxW (16 бит) и RxB (8 бит), где x – число от 8 до 15.

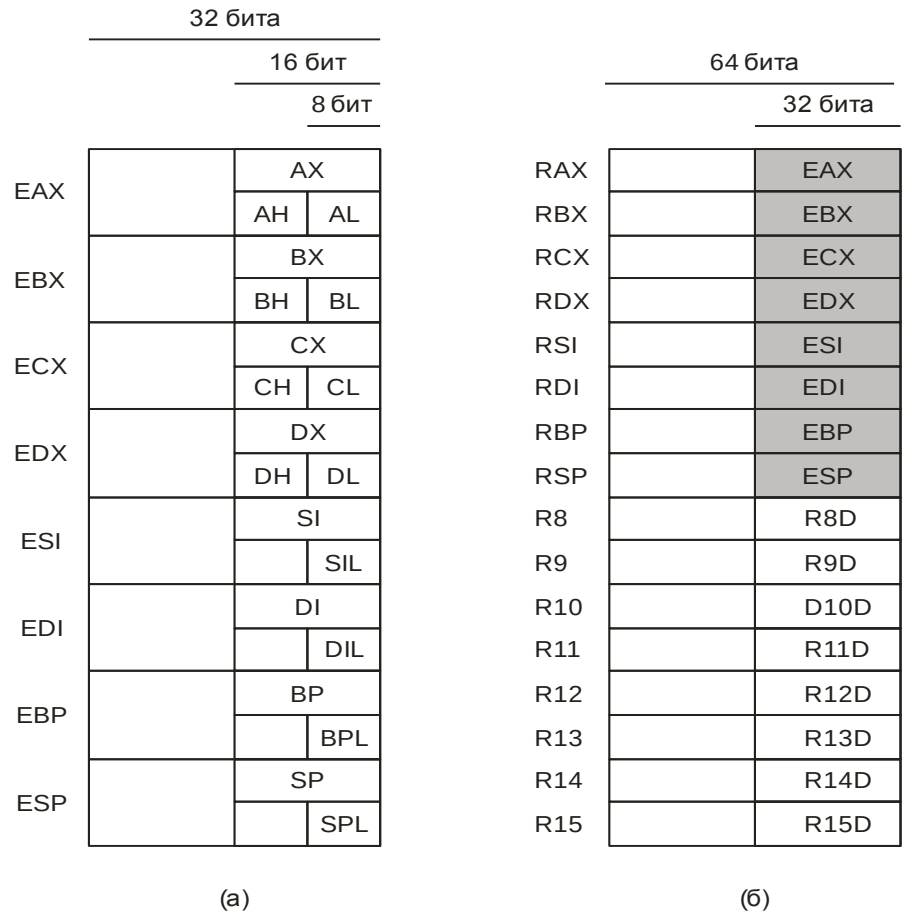

Рис. 1. Регистры общего назначения архитектуры x86 (а) и архитектуры x86-64 (б).

Сопроцессор. Сопроцессор (FPU – Floating Point Unit) предназначен для выполнения операций над вещественными числами. С программной точки зрения сопроцессор содержит несколько управляющих регистров, а также блок из 8 регистров данных разрядностью 80 бит, организованных в стек (рис. 2.8). Номер регистра, являющегося текущей вершиной стека, хранится в специальном поле регистра состояния (указателе вершины стека). Операция push уменьшает значение указателя на 1 и помещает данные в регистр, являющийся новой вершиной стека. Операция pop записывает

данные с вершины стека в память или регистр и увеличивает указатель на 1. Инструкции сопроцессора адресуют регистры либо явно, либо неявно. Неявная адресация (без указания конкретного регистра) подразумевает использование регистров, находящихся на вершине стека. Явная адресация подразумевает указание смещения регистра относительно вершины стека.

Например: st(0) регистр на вершине стека, st(1) – следующий за ним и т.д. записывает данные с вершины стека в память или регистр и увеличивает указатель на 1. Инструкции сопроцессора адресуют регистры либо явно, либо неявно. Неявная адресация (без указания конкретного регистра) подразумевает использование регистров, находящихся на вершине стека. Явная адресация подразумевает указание смещения регистра относительно вершины стека. Например: st(0) регистр на вершине стека, st(1) – следующий за ним и т.д.

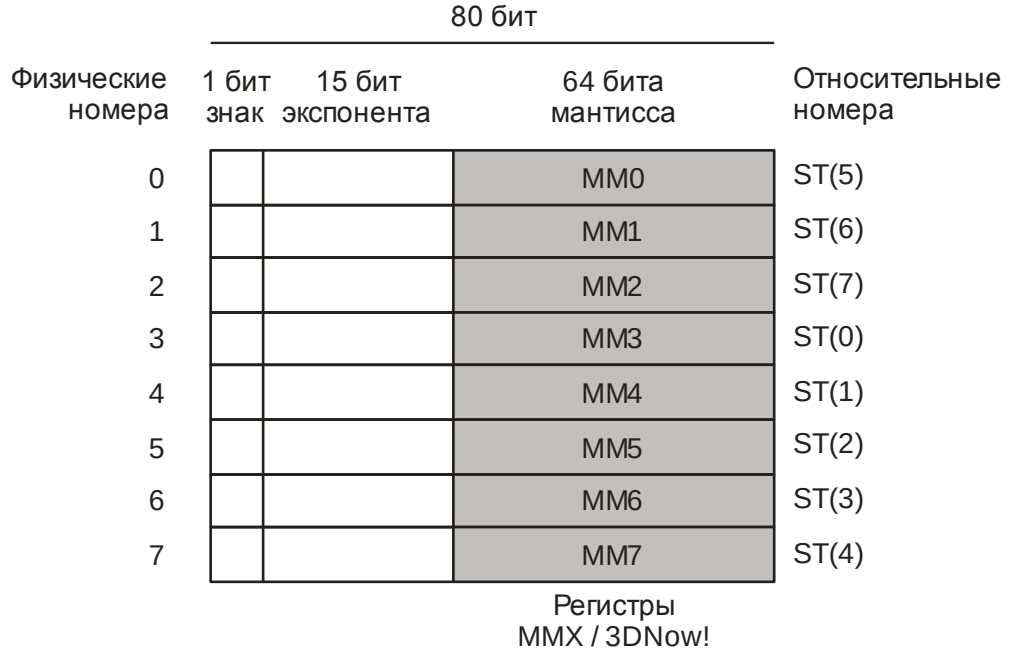

Рис.28. Регистры сопроцессора, MMX и 3DNow!

Векторные расширения. Векторные расширения реализуют технологию SIMD-вычислений.. На рисунке (рис.28) приведены регистры сопроцессора, MMX и 3DNow! Характеристики основных векторных расширений архитектуры x86 / x86-64 представлены в таблице 1.

Табл. 1.

Название	Размер регистра, бит	Число регистров	Имена регистров	Типы элементов данных, размер в битах
MMX	64	8	mm0-mm7	цел.: 8,16,32,64
3DNow!	64	8	mm0-mm7	вещ.: 32
SSE, …	128	8 / 16	xmm0-xmm15	цел.: 8,16,32,64, вещ.: 32, 64
AVX	256	16	ymm0-ymm15	цел.: 8,16,32,64, вещ.: 32, 64

2. ЗАДАНИЕ К ЛАБОРАТОРНОЙ РАБОТЕ

1. Изучить программную архитектуру x86/x86-64:
    - набор регистров,
    - основные арифметико-логические команды,
    - способы адресации памяти,
    - способы передачи управления,
    - работу со стеком,
    - вызов подпрограмм,
    - передачу параметров в подпрограммы и возврат результатов,
    - работу с арифметическим сопроцессором,
    - работу с векторными расширениями.
2. Для программы на языке Си (из лабораторной работы 1) сгенерировать ассемблерные листинги для архитектуры x86 и архитектуры x86-64, используя различные уровни комплексной оптимизации.
3. Проанализировать полученные листинги и сделать следующее.
    - Сопоставьте команды языка Си с машинными командами.
    - Определить размещение переменных языка Си в программах на ассемблере (в каких регистрах, в каких ячейках памяти).
    - Описать и объяснить оптимизационные преобразования, выполненные компилятором.

- Продемонстрировать использование ключевых особенностей архитектур x86 и x86-64 на конкретных участках ассемблерного кода.
- Сравнить различия в программах для архитектуры x86 и архитектуры x86-64.

4. Составить отчет по лабораторной работе. Отчет должен содержать следующее.
    - Титульный лист.
    - Цель лабораторной работы.
    - Полный компилируемый листинг реализованной программы и команды для ее компиляции.
    - Листинг на ассемблере с описаниями назначения команд с точки зрения реализации алгоритма выбранного варианта.
    - Вывод по результатам лабораторной работы.

3. ВАРИАНТЫ ЗАДАНИЙ

Варианты заданий взять из лабораторной работы №1.

4. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Назовите отличия архитектур x86 и x86-64.
2. Почему в программах следует избегать лишних обращений в память?
3. Как зависимости между командами влияют на процесс исполнения программы? Как компилятор и процессор борются с ними?