1 Overview of the AMD64 Architecture

1.1 Introduction

The AMD64 architecture is a simple yet powerful 64-bit, backward-compatible extension of the industry-standard (legacy) x86 architecture. It adds 64-bit addressing and expands register resources to support higher performance for recompiled 64-bit programs, while supporting legacy 16-bit and 32-bit applications and operating systems without modification or recompilation. It is the architectural basis on which new processors can provide seamless, high-performance support for both the vast body of existing software and 64-bit software required for higher-performance applications.

The need for a 64-bit x86 architecture is driven by applications that address large amounts of virtual and physical memory, such as high-performance servers, database management systems, and CAD tools. These applications benefit from both 64-bit addresses and an increased number of registers. The small number of registers available in the legacy x86 architecture limits performance in computationintensive
applications. Increasing the number of registers provides a performance boost to many such applications.

1.1.1 AMD64 Features

The AMD64 architecture includes these features:

• Register Extensions (see Figure 1-1 on page 2):
- 8 additional general-purpose registers (GPRs).
- All 16 GPRs are 64 bits wide.
- 8 additional YMM/XMM registers.
- Uniform byte-register addressing for all GPRs.
- An instruction prefix (REX) accesses the extended registers.

• Long Mode (see Table 1-1 on page 2):
- Up to 64 bits of virtual address.
- 64-bit instruction pointer (RIP).
- Instruction-pointer-relative data-addressing mode.
- Flat address space.

1.1.2 Registers

Table 1-2 compares the register and stack resources available to application software, by operating mode. The left set of columns shows the legacy x86 resources, which are available in the AMD64 architecture's legacy and compatibility modes. The right set of columns shows the comparable resources in 64-bit mode. Gray shading indicates differences between the modes. These register differences (not including stack-width difference) represent the *register extensions* shown in Figure 1-1.

As Table 1-2 shows, the legacy x86 architecture (called *legacy mode* in the AMD64 architecture) supports eight GPRs. In reality, however, the general use of at least four registers (EBP, ESI, EDI, and ESP) is compromised because they serve special purposes when executing many instructions. The AMD64 architecture's addition of eight GPRs—and the increased width of these registers from 32 bits to 64 bits—allows compilers to substantially improve software performance. Compilers have more flexibility in using registers to hold variables. Compilers can also minimize memory traffic—and thus boost performance—by localizing work within the GPRs.

1.1.3 Instruction Set

The AMD64 architecture supports the full legacy x86 instruction set, with additional instructions to support long mode (see Table 1-1 on page 2 for a summary of operating modes). The applicationprogramming
instructions are organized into four subsets, as follows:

• *General-Purpose Instructions*—These are the basic x86 integer instructions used in virtually all programs. Most of these instructions load, store, or operate on data located in the general-purpose registers (GPRs) or memory. Some of the instructions alter sequential program flow by branching to other program locations.

• *Streaming SIMD Extensions Instructions (SSE)*—These instructions load, store, or operate on data located primarily in the YMM/XMM registers. 128-bit media instructions operate on the lower half of the YMM registers. SSE instructions perform integer and floating-point operations on

vector (packed) and scalar data types. Because the vector instructions can independently and simultaneously perform a single operation on multiple sets of data, they are called *singleinstruction, multiple-data* (SIMD) instructions. They are useful for high-performance media and scientific applications that operate on blocks of data.

• *Multimedia Extension Instructions*—These include the MMX™ technology and AMD 3DNow!™ technology instructions. These instructions load, store, or operate on data located primarily in the 64-bit MMX registers which are mapped onto the 80-bit x87 floating-point registers. Like the SSE instructions, they perform integer and floating-point operations on vector (packed) and scalar data types. These instructions are useful in media applications that do not require high precision. Multimedia Extension Instructions use saturating mathematical operations that do not generate operation exceptions. AMD has de-emphasized the use of 3DNow! instructions, which have been superceded by their more efficient SSE counterparts. Relevant recommendations are provided in Chapter 5, "64-Bit Media Programming" on page 233, and in the *AMD64 Programmer's Manual Volume 4: 64-Bit Media and x87 Floating-Point Instructions*.

• *x87 Floating-Point Instructions*—These are the floating-point instructions used in legacy x87 applications. They load, store, or operate on data located in the 80-bit x87 registers.

Some of these application-programming instructions bridge two or more of the above subsets. For example, there are instructions that move data between the general-purpose registers and the YMM/XMM or MMX registers, and many of the integer vector (packed) instructions can operate on either YMM/XMM or MMX registers, although not simultaneously. If instructions bridge two or more subsets, their descriptions are repeated in all subsets to which they apply.

1.1.4 Media Instructions

Media applications—such as image processing, music synthesis, speech recognition, full-motion video, and 3D graphics rendering—share certain characteristics:

• They process large amounts of data.

• They often perform the same sequence of operations repeatedly across the data.

• The data are often represented as small quantities, such as 8 bits for pixel values, 16 bits for audio samples, and 32 bits for object coordinates in floating-point format.

SSE and MMX instructions are designed to accelerate these applications. The instructions use a form of vector (or packed) parallel processing known as single-instruction, multiple data (SIMD) processing. This vector technology has the following characteristics:

• A single register can hold multiple independent pieces of data. For example, a single YMM register can hold 32 8-bit integer data elements, or eight 32-bit single-precision floating-point data elements.

• The vector instructions can operate on all data elements in a register, independently and simultaneously. For example, a PADDB instruction operating on byte elements of two vector operands in 128-bit XMM registers performs 16 simultaneous additions and returns 16 independent results in a single operation.

Перевод
1 обзор архитектуры AMD64
1.1 Введение
Архитектура AMD64 - простое все же мощное 64-разрядное, совместимое расширение промышленного стандарта (устаревшей) x86 архитектуры. Оно добавляет 64-разрядную адресацию и Расширяет ресурсы регистра
чтобы поддерживать более высокую производительность для перекомпилированных 64-разрядных программ, при поддержке 16-разрядных устаревших и 32-
разрядных приложений и операционные системы без модификации или перекомпиляции. Это - архитектурное
основание, на котором новые процессоры могут предоставить цельную, высокоэффективную поддержку для обеих: обширного количества существующего программного обеспечения и 64-разрядного программного обеспечения, требуемого более высокой производительности в приложениях.
Потребность в 64-разрядной x86 архитектуре вызвана приложениями, которые адресуют большие Количества виртуальной
и физической памяти, такие как высокоэффективные серверы, системы управления базами данных и CAD
инструменты. Эти приложения извлекают выгоду и из 64-разрядных адресов и из увеличенного числа регистров. Небольшое количество регистров, доступных в устаревшей x86 архитектура, ограничивает производительность в приложениях с интенсивным вычислениями. Увеличение числа регистров обеспечивает повышение производительности большинства таких
приложения.
1.1.1 Функции AMD64
Архитектура AMD64 включает эти функции:
• Расширения регистров(см. рисунок 1-1 на странице 2):
- 8 дополнительных регистров общего назначения (GPRs).
- Все 16 GPRs шириной 64 бита .
- 8 дополнительных регистров YMM/XMM.
- Универсальный байт-регистр, адресующийся для всех GPRs.
- Префикс инструкции (REX) получающие доступ к расширенным регистрам.
• Длинный Режим (см. Таблицу 1-1 на странице 2):
- До 64 битов виртуального адресации.
- 64-разрядный указатель команд (RIP).
- Относительный способ адресации данных.
- Область простой адресации.
1.1.2 Регистры
Таблица 1-2 сравнивает регистр и ресурсы стека, доступные прикладному программному обеспечению, по режиму работы. Левый набор столбцов показывает наследованные x86 ресурсы, которые доступны в AMD64 архитектуре режимы:
наследованния и эмуляции. Правый набор столбцов показывает сопоставимые
ресурсы в 64-разрядном режиме. Серая штриховка указывает различия между режимами. Эти различия регистров (не включая различие ширины стека) представляют расширения регистра, показанные в
Рисунок 1-1.
Поскольку Таблица 1-2 показывает, устаревшая x86 архитектура (названный устаревшим режимом в архитектуре AMD64)
поддерживает восемь GPRs. В действительности, однако, общее использование по крайней мере четырех регистров (EBP, ESI, EDI, и
ESP), поставлено под угрозу, потому что они служат особому назначению при выполнении многих инструкций.
Добавление архитектуры AMD64 восьми GPRs — и увеличенная ширина этих регистров от 32 битов

к 64 битам — позволяет компиляторам существенно улучшать производительность программного обеспечения. У компиляторов есть больше
гибкость в использовании регистров, чтобы содержать переменные. Компиляторы могут также минимизировать трафик памяти — и таким образом
повысить производительность — локализуя работу в GPRs.

1.1.3 Система команд

Архитектура AMD64 поддерживает полный набор инструкций устарешвей x86 системы команд + дополнительные инструкции для
поддержки длинного режима (см. Таблицу 1-1 на странице 2

Инструкция прикладного программирования организованы в четыре подмножества, а именно:

• Команды общего назначения- Это основные целые инструкции x86 используются практически во всех программах. Большинство из этих инструкций загружается, хранится, или действуют над данными, расположенными в регистрах общего назначения(GPRS)
или памяти. Некоторые из инструкций изменяют последовательный ход ветвления программы в других местах программы.

• Потоковое Расширение процессора SIMD (SSE) — Эти инструкции загружаются, хранятся, или оперируют над данными
расположенными прежде всего в регистрах YMM/XMM. 128-разрядные медиа инструкции воздействуют на младшую
половина регистров YMM. Инструкции SSE преобразуют целочисленные и операции с плавающей точкой в
векторные (упакованные) и скалярные типы данных. Поскольку векторные инструкции могут независимо и
одновременно выполните одиночный поток команд,над множественным поток данных, их вызывают singleinstruction,
инструкции множественных данных (SIMD) . Они полезны для высокоэффективных носителей и научных приложений, которые работают с блоками данных.

• Инструкции Мультимедийного расширения — Они включают технологию MMX™ и AMD 3DNow! ™ технологические инструкции. Эти инструкции загружается, хранятся, или оперируют над данными, расположенными прежде всего в
64-разрядные регистры MMX, которые отображены на 80-разрядные x87 регистры с плавающей точкой. Как и SSE инструкции, они преобразуют целочисленные и операции с плавающей точкой в векторные (упакованные) и скалярные типы данных. Эти инструкции полезны в медиа приложениях, которые не требуют высокой точности.

Инструкций Мультимедийного расширения, Используются насыщающиеся математические операции, которые не генерируют
исключения. AMD преуменьшил роль использования 3DNow! инструкции, которые были замененный их более эффективными дубликатами SSE. В соответствующих рекомендациях приведены
Глава 5, “64-Bit Media Programming” на странице 233, и в Руководстве Программиста AMD64 Volume 4: 64-Bit Media and x87 Floating-Point Instructions.

• x87 Инструкции С плавающей точкой — Это инструкции с плавающей точкой, используемые в устаревших87
приложения. Они загружают, хранят или воздействуют на данные, расположенные в 80-разрядных регистрах x87.

Некоторые из этих инструкций прикладного программирования соединяют мостом два или больше из вышеупомянутых подмножеств. Для
примера, есть инструкции, которые перемещают данные между регистрами общего назначения и YMM/XMM или регистры MMX и многие целочисленные векторные инструкции (упакованные) могут воздействовать на YMM/XMM или регистры MMX, несмотря на то, но не одновременно. Если инструкции соединяют мостом два или больше
подмножества, их описания повторены во всех подмножествах, к которым они применяются.

1.1.4 Медиа Инструкции

Медиа Приложения — такие как обработка изображений, музыкальный синтез, распознавание речи, бесперебойно воспроизводимое
видео и 3D рендеринг графики — имеют определенные общие характеристики:

• Они обрабатывают большие объемы данных.
• Они часто выполняют ту же последовательность операций неоднократно через данные.
• Данные часто представляются как небольшие количества, такие как 8 битов для пиксельных значений, 16 битов для аудио
образцы, и 32 бита для координат объекта в формате с плавающей точкой.

SSE и инструкции MMX разработаны, чтобы ускорить эти приложения. Инструкции используют форму векторной (или упакованной) параллельной обработки, известная как одиночный поток команд, множественный поток данных (SIMD). У этой векторной технологии есть следующие характеристики:

• Один регистр может содержать Множество независимых частей данных. Например, единственный YMM
регистр может содержать 32 8-разрядных элемента целочисленных данных или восемь 32-разрядных элементов данных с плавающей точкой одинарной точности.
• Векторные инструкции могут воздействовать на все элементы данных в регистре, независимо и одновременно. Например, инструкция PADDB, воздействующая на элементы байта двух векторных операндов в 128-разрядных регистрах XMM выполняют одновременно 16 сложений и возвращаются 16
независимые результаты за одну операцию.

The article deals with - technical description of the amd64 architecture

As the title implies Overview of the AMD64 Architecture the article describes features of a amd64 system

A mention should be made about General-Purpose Register's wide in amd64 architectureIt

1) **The text deals with** different facts concerning the existing of black holes.

2**) As the title implies** "Black Holes" **the article describes** some discoveries of the American scientists.

3**) A mention should be made about** the comparison of different types of black holes.

4) **Much attention is given to** stellar black holes.

5) **It should be noted** that a lot of scientists are interested in black holes.

6) **The text gives valuable information on/about** the event horizon.

7) **The fact that** there is a lot of doubt about the existing of black holes **is stressed in the text.**

8**) A lot of recent data are given in the article.**

9) **Modern equipment is mentioned in the article.**

10) **The text is of interest to** the wide range of readers/to the second-year students.

11) **The text is of great help to** future astronomers.