**Программная архитектура Intel Architecture**

Оглавление

[Модель x86 2](#_Toc1569365)

[Варианты архитектуры 2](#_Toc1569366)

[Основные расширения 3](#_Toc1569367)

[Регистровая структура 4](#_Toc1569368)

[Регистры 4](#_Toc1569369)

[Регистры x87 FPU 5](#_Toc1569370)

[Модель памяти 6](#_Toc1569371)

[Типы аппаратно поддерживаемых данных 6](#_Toc1569372)

[Формат машинных команд 8](#_Toc1569373)

[Базовая система команд 8](#_Toc1569374)

[Модель AMD64 12](#_Toc1569375)

[Варианты архитектуры 12](#_Toc1569376)

[Основные различия AMD64 и Intel64(EMT64) 12](#_Toc1569377)

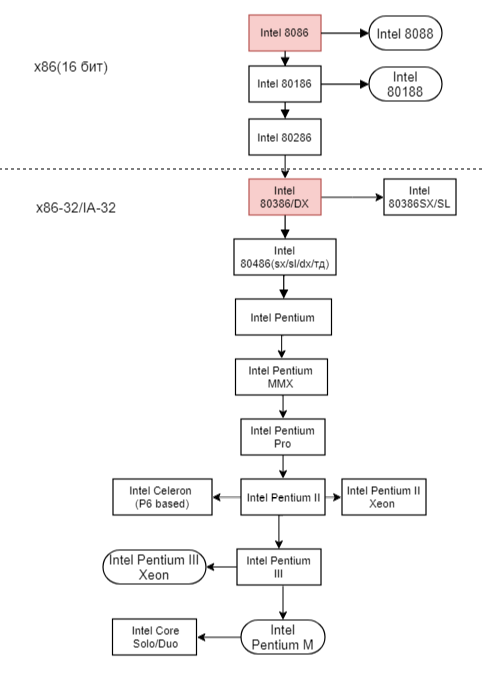
[Архитектурные особенности 13](#_Toc1569378)

[Инструкции введенные в AMD64 15](#_Toc1569379)

[Документация и источники 15](#_Toc1569380)

# Модель x86

## Варианты архитектуры



### Основные расширения

|  |  |
| --- | --- |
| Название | Описание |
|  |  | MMX | Multimedia Extensions – SIMD набор инструкций, представленный в процессоре Pentium MMX. Предназначен для ускорения обработки аудио и видеоданных. Режим использует регистры сопроцессора. |
| SSE | Streaming SIMD extensions – расширение, добавленное в линейке процессоров Pentium III. Убирает минусы MMX(регистры сопроца и работа только с целыми числами), добавляя 8(в х86-64 -16) 128-битных регистров XMM и упакованный 128-битный тип данных |
| SSE2 | Появилось в Pentium 4.Расширяет набор SSE с целью вытеснить MMX. Данный набор добавляет 144 команды к SSE, в котором только 70 команд. |
| SSE3 | Добавляет 13 инструкций, наиболее заметное изменение — возможность горизонтальной работы с регистрами |
| SSSE3 | Добавляются 16 уникальных команд, работающих с упакованными целыми. Каждая из них может работать как с 64-битными (MMX), так и с 128-битными (XMM) регистрами |
| SSE4(SSE4.1, SSE4.2, ATA) | Доступны в процессорах с МА Nehalem. Состоит из 54 инструкций(ускорение движения в видеокартах, быстрое чтение из памяти, упрощение векторизации) |
| AVX | XMM регистры увеличиваются до 256 бит(YMM). Набор инструкций AVX содержит в себе аналоги 128-битных [SSE](https://ru.wikipedia.org/wiki/SSE) инструкций для вещественных чисел. 128-битные AVX-инструкции сохраняют прочие преимущества AVX, такие, как новая схема кодирования, трехоперандный синтаксис и не выровненный доступ к памяти. |
| AVX2 | Дальнейшее развитие AVX. Целочисленные команды SSE начинают работать с 256-битными регистрами. |
| FMA | Fused Multiply-Add - это набор опциональных 128- и 256-битных SIMD-инструкций для архитектур x86 и x86-64, предназначенный для выполнения операции умножения-сложения над числами в формате с плавающей запятой.  Существует два варианта расширения: FMA4(AMD) и FMA3(Intel), они не совместимы |
| AES | Целью данного расширения является ускорение приложений, использующих шифрование по алгоритму AES |
| MPX | Bound-instructions and registers |
| CX8 | Поддерживается инструкция CMPXCHG8B |
| SEP | Поддерживаются инструкции SYSENTER и SYSEXIT |
| CMOV | Поддерживаются инструкции условной пересылки данных |
| PCLMUL | Поддержка инструкции PCLMUL |
| MON | Поддержка инструкции MONITOR/MWAIT |
| CX16 | Поддержка инструкции CMPXCHG16B |
| POPCNT | Поддержка инструкции POPCNT |
| MOVBE | Поддержка инструкции MOVBE |

## Регистровая структура

### Регистры

|  |  |  |  |
| --- | --- | --- | --- |
| Функциональная группа | Имя | Описание | Примечания |
| Общего назначения | EAX | Аккумулятор, используется в арифметических инструкциях |  |
| EBX | По умолчанию используется как базовая компонента эффективного адреса операнда, находящегося в памяти |  |
| ECX | Используется для циклов |  |
| EDX | Хранит данные во время i/o операций |  |
| ESI | По умолчанию используется для задания индексной компоненты ЕА, а также для адресации элементов строки-источника в командах обработки строк |  |
| EDI | По умолчанию используется аналогично SI для задания индексной компоненты ЕА, а также для адресации элементов строки-приемника в командах обработки строк |  |
| EBP | По умолчанию используется как базовая компонента эффективного адреса операнда в памяти, как EВХ |  |
| ESP | По умолчанию используется для адресации вершины стека |  |
| Контроля потока управления | EIP | Счетчик команд, содержит адрес следующей выполняемой инструкции |  |
| Состояния | EFLAGS | Отражает текущее состояние процессора |  |
| Сегментные регистры | CS | Содержит адрес сегмента с машинными командами, к которому имеет доступ микропроцессор |  |
| DS | Хранит адрес сегмента данных текущей программы |  |
| SS | Содержит адрес сегмента стека |  |
| ES | Хранят адреса дополнительных сегментов |  |
| FS |  |
| GS |  |
| Управление памятью | GDTR | Содержит 32-битный базовый адрес и 16-битный предел глобальной таблицы дескрипторов(GDT) |  |
| LDTR | Содержит только селектор дескриптора локальной таблицы дескрипторов(LDT) |  |
| IDTR | -||- таблицы дескрипторов прерываний |  |
| TR  (task register) | Содержит селектор дескриптора сегмента состояния задачи (TSS).Сам TSS автоматом загружается в скрытую часть TR |  |
| Управляющие регситры | CR0 | Управление режимами и различными искл. ситуациями |  |
| CR1 | Зарезервирован, при обращении выбросится исключение |  |
| CR2 | Если включен механизм стр. адресации и генерируется исключение, то CR2 будет содержать линейный адрес, поступление которого в блок стр. адресации вызвало эту ошибку |  |
| CR3 | Содержит 20 старших бит физ. адреса каталога страниц |  |
| CR4 | Обеспечивает вкл/выкл поддержки различных режимов и доп. возможностей процессора | Перед программированием битов регистра стоит проверить наличие режима с помощью команды CPUID |
| CR5 - CR7 | Зарезервированы, как и cr1 |  |
| Отладка | DR0 – DR5 | Содержат линейные адреса точек остнова |  |
| DR6 | Регистр состояния отладки |  |
| DR7 | Регистр управления отладкой |  |

### Регистры x87 FPU

|  |  |  |
| --- | --- | --- |
| Функциональная группа | Имя | Описание |
| Регистры данных | (R0 -R7) | 80-битные регистры, к которым нельзя обратится по имени. Значение в ней хранятся в формате с двойной расширенной точностью и плавающей точкой. Эти восемь регистров рассматриваются как стек, вершина которого называется ST, а более глубокие элементы — ST(1), ST(2) и так далее до ST(7).  К регистрам R0 – R7 нельзя обращаться напрямую, по именам, но если процессор поддерживает расширение ММХ, то мантиссы, находящиеся в этих регистрах, становятся доступны, как ММ0 – ММ7. |
| Регистр управления | CR | Содержит биты отвечающие за способ округления, точность, реакцию на бесконечность. Так же содержит в себе битовую маску исключений. |
| Регистр состояний | SR | 16 – битный, показывает текущее состояние FPU. |
| Контроля потока управления | FIP  & FDP | Содержат адрес последней выполненной команды (кроме FINIT, FCLEX, FLDCW, FSTCW, FSTSW, FSTSWAX, FSTENV, FLDENV, FSAVE, FRSTOR и FWAIT) и адрес ее операнда соответственно и используются в обработчиках исключений для анализа вызвавшей его команды. |
| Регистр тэгов | TW | Содержит восемь пар бит, описывающих содержание каждого регистра данных, — биты 15 – 14 описывают регистр R7, 13 – 12 — R6 и т.д. Если пара бит (тег) равна 11, соответствующий регистр пуст. 00 означает, что регистр содержит число, 01 — ноль, 10 — нечисло, бесконечность, денормализованное число, неподдерживаемое число. |
| Опкод | FOP | 11 битный, содержит опкод последней не управляющей инструкции |

## Модель памяти

При использовании средств управления памятью процессора, программы напрямую не обращаются к физической памяти. Вместо этого они получают доступ к памяти с помощью одной из трех моделей памяти: плоской, сегментной или в режиме реального адреса:

•**Плоская модель** — для программы память является единым и непрерывным адресным пространством. Этот пространство называется линейным адресным пространством. Код, данные и стеки содержатся в этом адресном пространстве. Линейное адресное пространство является байт-адресуемым. Адрес для любого байта в линейном адресном пространстве называется линейным адресом.

•**Сегментная модель** — механизм адресации, обеспечивающий существование нескольких независимых адресных пространств как в пределах одной задачи, так и в системе в целом для защиты задач от взаимного влияния. Память для программы делится на непрерывные области памяти, называемые ***сегментами***.

Сама программа может обращаться только к данным, которые находятся в этих сегментах. Операционная система размещает сегменты программы в ОП по определенным физическим адресам, а значения этих адресов записывает в определенные места, в зависимости от режима работы микропроцессора:

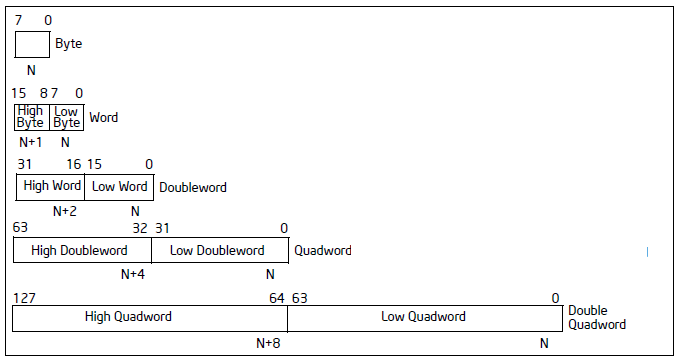
* в реальном режиме адреса помещаются непосредственно в сегментные регистры
* в защищенном режиме - в специальную системную дескрипторную таблицу (Элементом дескрипторной таблицы является дескриптор сегмента. Каждый сегмент имеет дескриптор сегмента - 8 байт. Существует три дескрипторные таблицы. Адрес каждой таблицы записывается в специальный системный регистр).

Таким образом, для обращения к конкретному физическому адресу ОП необходимо определить адрес начала сегмента и смещение внутри сегмента.

• **Модель режима реальной адресации** — это модель памяти для процессора Intel 8086. Она поддерживается для обеспечения совместимости с существующими программами, написанными для работы на процессоре Intel 8086. В real-address mode используется специфическая реализация сегментированной памяти, в которой линейное адресное пространство для программы и операционный системы состоят из массива сегментов размером до 64 Кбайт каждый. Максимальный размер линейного адресного пространства в режиме реальной адресации 220 байт.

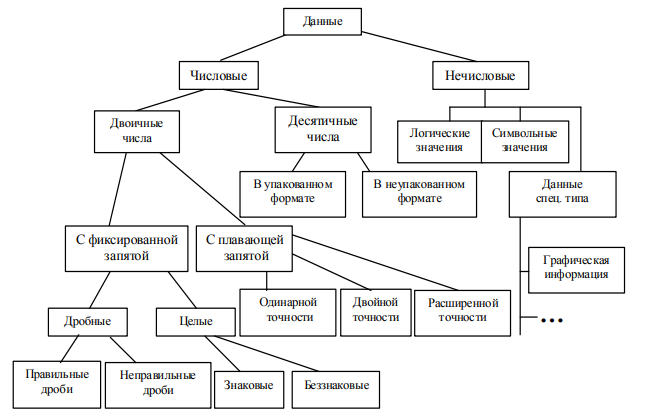
## Типы аппаратно поддерживаемых данных

Фундаментальные типы данных архитектур IA-32 и Intel64:

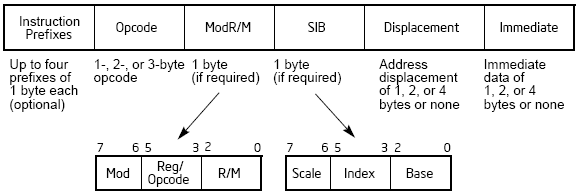


Большинство команд оперируют этими данными без какой-либо дополнительной их типизации. Учетверенное слово было введено в i486, двойное учетверенное — в Pentium III с расширением SSE. В памяти операнды хранятся побайтно.

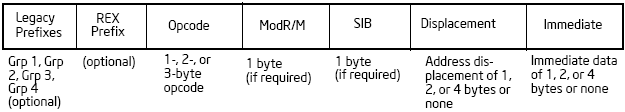
Кроме размера некоторые команды требуют дополнительных признаков классификации операндов. Арифметические команды оперируют числами, которые, в свою очередь, могут быть следующих типов:



## Формат машинных команд



**Формат команды в 64-битном режиме:**



## Базовая система команд

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| арифметические | | | | | | | | | | |
| двухоперандные | | | | однооперандные | | | | | | |
| аддитивные | мультипликативные | | | расширение форматов | инкремент  декремент | | изм.  знака | коррекция | | |
| ADD  ADC  ADCX  ADOX  SUB  SBB  CMP  NPG  CMPSB  CMPSW | беззнаковые | | знаковые | CBW  CWD | INC  DEC | NEG | | BCD  (десятичная) | | ASCII-  коррекция |
| MUL  DIV | IMUL  IDIV | | DAA  DAS | AAA  AAS  AAM  AAD | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| передачи данных и адресов | | | | | |
| общей передачи | условной передачи | стековые передачи | ввод/вывод | передачи адреса | передача флагов |
| MOV  MOVSX  MOVZX  BSWAP  CMPXCHG  CMPXCHG8B  CWD/CDQ  CBW/CWD  XCHG  XLAT | CMOVE/CMOVZ  CMOVNE/CMOVNZ  CMOVA/CMOVNBE  CMOVAE/CMOVNB  CMOVB/CMOVNAE  CMOVBE/CMOVNA  CMOVG/CMOVLNE  CMOVGE/CMOVNL  CMOVL/CMOVNGE  CMOVLE/CMOVNG  CMOVC  CMOVNC  CMOVO  CMOVNO  CMOVS  CMOVNS  CMOVP/CMOVPE  CMOVNP/CMOVPO | PUSH/PUSHA  POP/POPA  PPUSH | IN  OUT  INS/INSB/INSW/INSD  OUTS/OUTSB/ OUTSW/ OUTSD | LEA  LDS  LES  LDS  LES  LFS  LGS  LSS | LAHF  SAHF  PUSHF/PUSHFD  POPF/POPFD |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| битовые инструкции | | | | | | | |
| логические | | сдвиги | | | | установки по условию | сравнение |
| двухоперндные | однооперндные | арифметические | логические | циклические | | SETE/SETZ  SETNE/SETNZ  SETA/SETNBE  SETAE/SETNB/SETNC  SETB/SETNAE/SETC  SETBE/SETNA  SETG/SETNLE  SETGE/SETNL  SETL/SETNGE  SETLE/SETNG  SETS/SETNS  SETO  SETNO  SETPE/SETP SETPO/SETNP | BT  BTS  BTR  BTC  BSF  BSR |
| AND  OR  XOR  TEST | NOT | SAR  SAL | SHR/SHL  SHRD/SHLD | без CF | c CF |
| ROR  ROL | RCR  RCL |

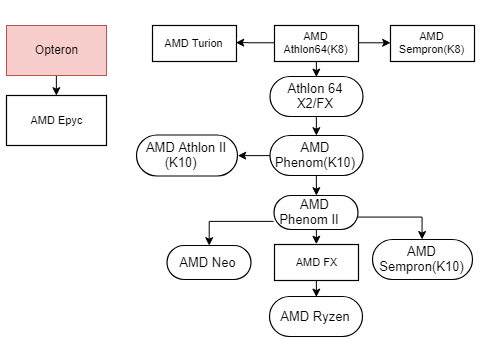
|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| управления программой | | | | | | | | | | |
| передача управления | | | | | | | | | программные прерывания | манипуляция флагами |
| переходы | | | | | циклы | вызовы | highLvl | возвраты |
| безусловные | условные | | | | LOOP  LOOPE/LOOPZ  LOOPNE/LOOPNZ | CALL | ENTER  LEAVE | RET  RETN  RETF | INT n  INT(INT3)  INTO  IRET | CLC  STC  CMC  CLD  STD  CLI  STI |
| JMP | знаковые | беззнаковые | по флагам | по счетчику |
| JG(JNG)  JL(JNL)  JGE(JNGE)  JLE(JNLE) | JA(JNA)  JB(JNB)  JAE(JNAE)  JBE(JNBE) | JO(JNO)  JC(JNC)  JZ(JNZ)  JS(JNS)  JP(JNP)  JE(JNE)  JPO  JPE | JCXZ  JECX |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| строковые инструкции | | | | |
| передачи | сравнения | передачи в EAX | хранения | повтора |
| MOVS  MOVSB  MOVSD  MOVSW | CMPS  CMPSB/CMPSW/CMPSD  SCAS  SCASB/SCASW/SCASD | LODS  LODSB  LODSW  LODSD | STOS  STOSB  STOSW  STOSD | REP  REPE/REPZ  REPNE/REPNZ |

|  |  |  |  |
| --- | --- | --- | --- |
| разное | | | |
| random number | extended state save/restore | BMI1, BMI2 | NOP  BOUND  UD2  XLAT/XLATB  CPUID  PREFETCHW  PREFETCHWT1  CLFLUSH  CLFLUSHOPT  CRC32  POPCNT |
| RDRAND  RDSEED | XSAVE  XSAVEC  XSAVEOPY  SRSTOR  XGETBV | ANDN  BEXTR  BLSI  BLSMSK  BLSR  BZHI  LZCNT  MULX  PDEP  PEXT  RORX  SARX  SHLX  SHRX  TZCNT |

# Модель AMD64

## Варианты архитектуры



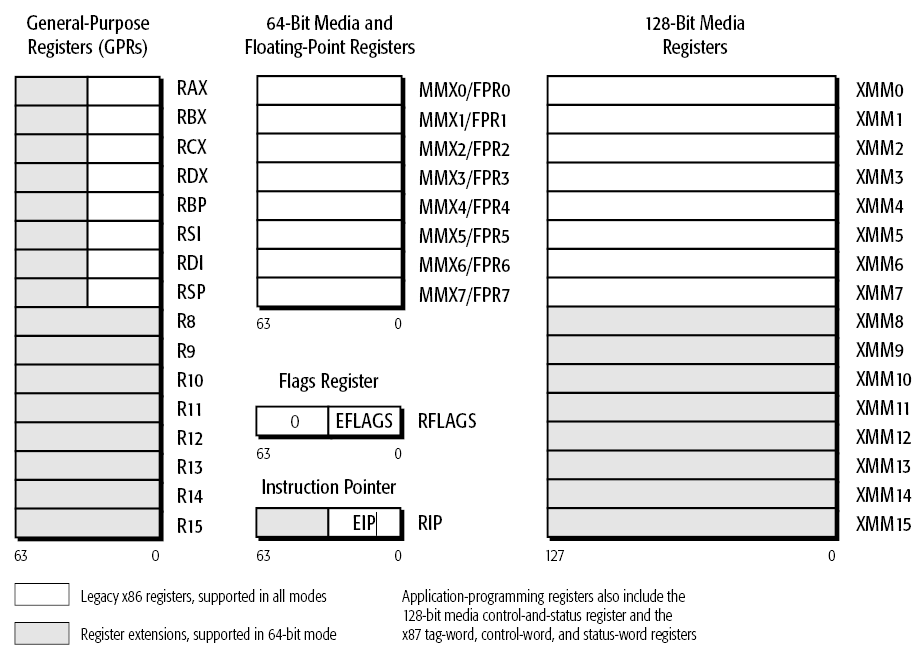
## Основные различия AMD64 и Intel64(EMT64)

* AMD64 поддерживает инструкции “3DNow!” и прочие.
* Intel64 не имеет возможности сохранять и восстанавливать уменьшенную версию состояния FPU (с использованием инструкций FXSAVE и FXRSTOR)
* В Intel64 упразднены некоторые MSR, которые присутствуют в AMD64.Например: SYSCFG, TOP\_MEM, TOP\_MEM2.
* Intel64 поддерживает инструкции MONITOR и MWAIT, используемые операционной системой для улучшения работы гиперпоточности
* Intel64 допускает использование SYSCALL/SYSRET только в 64-битном режиме(не compatibility mode) и SYSENTER/SYSEXIT поддерживаются в обоих режимах. В AMD64 SYSENTER/SYSEXIT не поддерживаются в Long mode.
* В 64-битном режиме переходы с префиксом 0×66 ведут себя по разному. Intel 64 игнорирует этот префикс: инструкция имеет 32-битный sign extended смещение, и указатель инструкции не усекается. AMD64 использует 16-битное поле смещения в инструкции, и очищает верхние 48 битов указателя инструкции
* Последние версии процессоров на базе AMD64 вводят поддержку сегментации для упрощения виртуализации

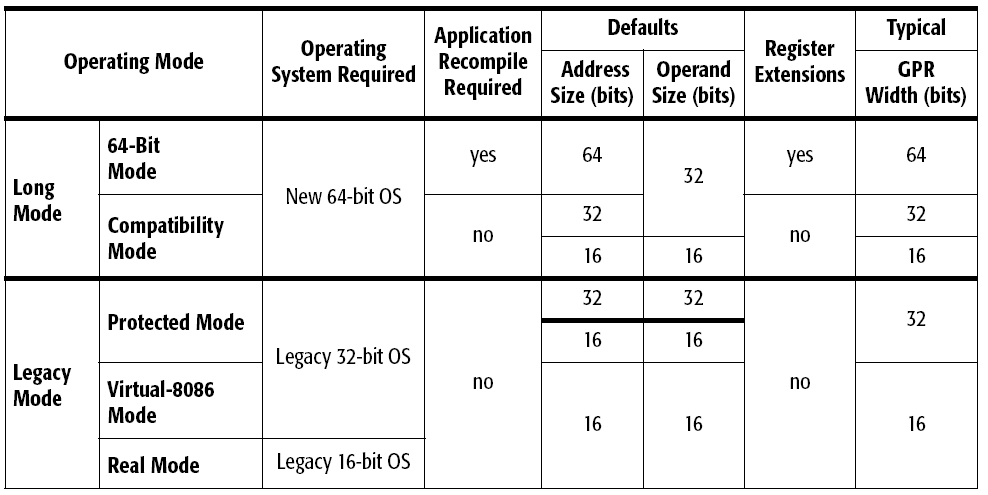
## Архитектурные особенности

Архитектура x86-64 вводит две новые особенности:

* Все регистры общего назначения расширены до 64 бит, и все арифметические и логические операции могут напрямую работать с 64-битными значениями
* Дополнительные регистры: количество GPRs увеличено с 8 до 16(+ r8…r15)
* Количество XMM (SSE) регистров увеличено с 8 до 16
* Большее виртуально адресное пространство: архитектура AMD64 вводит 64-битный формат для виртуального адреса, из которых ныне используются 48 младших бит, что позволяет использовать до 256TB (248) виртуального адресного пространства
* Большее физическое адресно пространство: первая реализация AMD64 поддерживает 40-битные физические адреса, что позволяет адресовать до 1TB оперативной памяти. Нынешние версии поддерживают 48-битные адреса
* Удаленные возможности: в 64-битном режиме используется только плоское адресное пространство, соответственно сегментная адресация, ”task state switch” механизм и “Virtual 8086 mode” остаются реализованными в “legacy mode”. Так же были удалены некоторые инструкции: сохранение/удаление сегментных регистров и регистров общего назначения в стэк, десятичная арифметика и инструкции BOUND, INTO.



Измененные регистры



Режимы работы

## Инструкции введенные в AMD64

|  |  |
| --- | --- |
| **Инструкция** | **Описание** |
| CDQE | Sign extend EAX into RAX |
| CQO | Sign extend RAX into RDX:RAX |
| CMPSQ | CoMPare String Quadword |
| CMPXCHG16B | CoMPare and eXCHanGe 16 Bytes |
| IRETQ | 64-bit Return from Interrupt |
| JRCXZ | Jump if RCX is zero |
| LODSQ | LoaD String Quadword |
| MOVSXD | MOV with Sign Extend 32-bit to 64-bit |
| POPFQ | POP RFLAGS Register |
| PUSHFQ | PUSH RFLAGS Register |
| RDTSCP | ReaD Time Stamp Counter and Processor ID |
| SCASQ | SCAn String Quadword |
| STOSQ | STOre String Quadword |
| SWAPGS | Exchange GS base with KernelGSBase MSR |

# Документация и источники

|  |  |  |  |
| --- | --- | --- | --- |
| URL | Имя | Описание | Примечания |
| [intel.com/.../intel-sdm](https://software.intel.com/en-us/articles/intel-sdm) | Intel® 64 and IA-32 Architectures Software Developer Manuals | Основная полная документация по всем аспектам архитектур IA32 и IA32e | Модель памяти и инструкции x86 отсюда |
| <https://developer.amd.com/resources/developer-guides-manuals/> | Developer Guides, Manuals & ISA Documents for  AMD processors | Документация по AMD64 | Выполнена лучше Intel manual’a |
| http://ref.x86asm.net/coder32.html |  |  |  |
| <http://sandpile.org> |  |  |  |