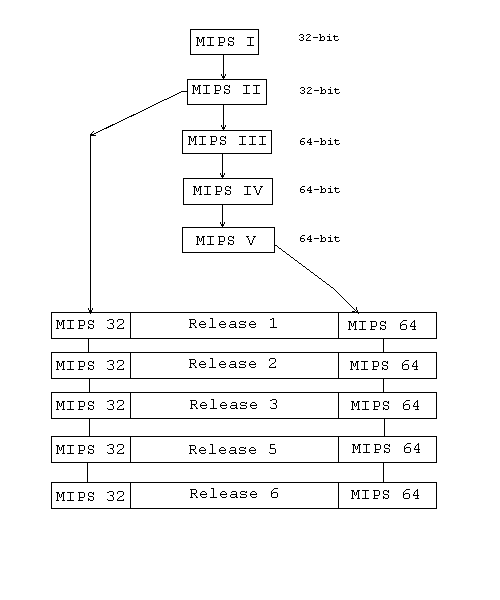
# Программная архитектура MIPS 32/64 Release 6

## Документация и источники:

|  |  |  |  |
| --- | --- | --- | --- |
| URL | Наименование | Описание | Примечания |
| https://s3-eu-west-1.amazonaws.com/downloads-mips/documents/MD00086-2B-MIPS32BIS-AFP-6.06.pdf | MIPS 32 Architecture for Programmers Vol. 2A (MD00086), Revision 6.06 | Описание набора инструкций MIPS 32 | Инструкции в основном отсюда |
| https://s3-eu-west-1.amazonaws.com/downloads-mips/documents/MD00087-2B-MIPS64BIS-AFP-6.06.pdf | MIPS 64 Architecture for Programmers Vol. 2A (MD00087), Revision 6.06 | Описание набора инструкций MIPS 64 |  |

## Варианты архитектуры (дерево развития)



## Некоторые расширения архитектуры

* MIPS 16 – 16-битное расширение для MIPS 32 и 64
* MDMX – улучшение работы с графикой (MIPS 64 only)
* MIPS-3D – инструкции для задач 3D-графики
* MIPS-SIMD – набор SIMD инструкций

## Модель памяти

#### Регистры

* 32 регистра общего назначения
  + r0 – const “0”
  + r31 – хранение последнего адреса перехода в некоторых инструкциях
* HI, LO – для операций целочисленного умножения / деления
* PC – счётчик команд

Кроме того, по стандарту определены следующие компоненты FPU:

* 32 регистра FPU
* 5 регистров управления FPU (строго 32-bit на всех архитектурах)

Все регистры имеют разрядность 32 бит в MIPS 32 и 64 бит в MIPS 64. Регистры 64-bit FPU могут работать в разрядности 32 бита.

#### Память

Используется единое адресное пространство для данных и стека. Контроль стека осуществляется вручную с использованием любого из регистров.

## Набор инструкций

#### Overview

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | Инструкции передачи данных | | CFC1 | | CFC2 | | CTC | | CTC2 | | LB | | LBE | | LBU | | LBUE | | LDC1 | | LDC2 | | LH | | LHE | | LHU | | LHUE | | LL | | LLE | | LLWP | | LLWPE | | LSA | | LUI | | LW | | LWC1 | | LWC2 | | LWE | | LWPC | | MFC0 | | MFC1 | | MFC2 | | MFHC0 | | MFHC1 | | MFHC2 | | MOV.fmt | | MTC0 | | MTC1 | | MTC2 | | MTHC0 | | MTHC1 | | MTHC2 | | NAL | | RHDWR | | RDPGPR | | SB | | SBE | | SC | | SCE | | SCWP | | SCWPE | | SDC1 | | SDC2 | | SH | | SHE | | SW | | SWE | | SWC1 | | SWC2 | | WRPGPR | | |  |  | | --- | --- | | Инструкции АЛУ | Инструкции сопроцессоров | | ADD | ABS.fmt | | ADDI | ADD.fmt | | ADDIU | CEIL.L.fmt | | ADDIUPC | CEIL.W.fmt | | ADDU | CLASS.fmt | | AND | CMP.condn.fmt | | ANDI | COP2 | | AUI | CVT.D.fmt | | AUIPC | CVT.L.fmt | | CRC32B/H/W | CVT.S.fmt | | CRC32CB/H/W | CVT.W.fmt | | DIV | DIV.fmt | | MOD | FLOOR.L.fmt | | DIVU | FLOOR.W.fmt | | MODU | MADDF.fmt | | MUL | MSUBF.fmt | | MUH | MAX.fmt | | MULU | MIN.fmt | | MUHU | MAXA.fmt | | NOP | MINA.fmt | | NOR | MUL.fmt | | OR | NEG.fmt | | ORI | RECIP.fmt | | ROTR | RINT.fmt | | ROTR | ROUND.L.fmt | | SELEQZ | ROUND.W.fmt | | SELNEZ | RSQRT.fmt | | SLL | SEL.fmt | | SLLV | SELEQZ.fmt | | SLT | SELNEZ.fmt | | SLTI | SQRT.fmt | | SLTIU | SUB.fmt | | SLTU | TRUNC.w.fmt | | SRA | TRUNC.w.fmt | | SRAV |  | | SRL |  | | SRLV |  | | SSNOP |  | | SUB |  | | SUBU |  | | XOR |  | | XORI |  | | |  | | --- | | Инструкции передачи управления | | B | | BAL | | BALC | | BC | | BC1EQZ | | BC1NEZ | | BC2EQZ | | BC2NEZ | | BEQ | | BGEZ | | BLEZALC | | BGEZALC | | BLTZALC | | BGTZALC | | BEQZALC | | BNEZALC | | BEQC | | BNEC | | BLTC | | BGEC | | BLTUC | | BGEUC | | BGTC | | BLEC | | BGTUC | | BLEUC | | BLTZC | | BLEZC | | BGEZC | | BGTZC | | BEQZC | | BNEZC | | BGTZ | | BLEZ | | BLTZ | | BNE | | BOVC | | BNVC | | J | | JAL | | JALR | | JALR.HB | | JIALC | | JIC | | JR | | JR.HB | | |  |  | | --- | --- | | Инструкции работы с кэшем | Инструкции исключений | | CACHE | BREAK | | CACHEE | DERET | | GINVI | DI | | GINVT | DVP | | PREF | EHB | | PREFE | EI | | TLBINV | ERET | | TLBINVF | ERETNC | | TLBP | EVP | | TLBR | PAUSE | | TLBWI | SDBBP | | TLBWR | SIGRIE | |  | SYNC | |  | SYNCI | |  | SYSCALL | |  | TEQ | |  | TGE | |  | TGEU | |  | TLT | |  | TLTU | |  | TNE | |  | WAIT | | |  | | --- | | Прочие инструкции работы с данными | | ALIGN | | ALUIPC | | BITSWAP | | CLO | | CLZ | | EXT | | INS | | SEB | | SEH | | WSBH |  |  |  | | --- | --- | | Только в MIPS 64 | | | DADD | DMFC0 | | DADDIU | DMFC1 | | DADDU | DMFC2 | | DCLO | DMTC0 | | DCLZ | DMTC1 | | DEXT | DMTC2 | | DEXTM | DROTR | | DEXTU | DROTR32 | | DINS | DROTRV | | DINSM | DSBH | | DINSU | DSHD | | DMOD | DSLL | | DDIV | DSLL32 | | DMODU | DSLLV | | DDIVU | DSRA | | DSRA32 | LD | | DSRAV | LLD | | DSRL | LLDP | | DSRL32 | DLSA | | DSRLV | LWU | | DSUB | LWUPC | | DSUBU | SCD | |  | SD | |

#### Инструкции перемещения данных

|  |  |  |  |
| --- | --- | --- | --- |
| Инструкция | Кодировка | Описание | Примечания |
| CFC1 | opcode: 010001  “00010”  5 bit: rt  5 bit: fs  11 bit: “0” | rt = fs  fs – регистр состояния COP1 (32-bit) |  |
| CFC2 | opcode: 010010  “00010”  5 bit: rt  16 bit: reg | rt = reg  reg – регистр состояния COP2 |  |
| CTC1 | opcode: 010001  “00110”  5 bit: rt  5 bit: fs  11 bit: “0” | fs = rt  fs – регистр состояния COP1 (32-bit) |  |
| CTC2 | opcode: 010010  “00110”  5 bit: rt  16 bit: reg | reg = rt  reg – регистр состояния COP2 |  |
| LB | opcode: 100000  5 bit: base  5 bit: rt  16 bit: offset | Загрузить в rt байт по адресу в base и смещению offset и расширить знаком |  |
| LBE | opcode: 011111  5 bit: base  5 bit: rt  9 bit: offset  “0”  operation: 101100 | Загрузить в rt байт по адресу в base и смещению offset и расширить знаком  При расчёте адреса используется другая схема с использованием кэшей. |  |
| LBU | opcode: 100100  5 bit: base  5 bit: rt  16 bit: offset | Загрузить в rt байт по адресу в base и смещению offset |  |
| LBUE | opcode: 011111  5 bit: base  5 bit: rt  9 bit: offset  “0”  operation: 101000 | Загрузить в rt байт по адресу в base и смещению offset.  При расчёте адреса используется другая схема с использованием кэшей. |  |
| LDC1 | opcode: 110101  5 bit: base  5 bit: rt  16 bit: offset | Загрузить двойное слово по адресу в base и смещению offset в регистр ft FPU |  |
| LDC2 | opcode: 010010  5 bit: 01110  5 bit: rt  5 bit: base  11 bit: offset | Загрузить двойное слово в регистр rt COP2 |  |
| LH | opcode: 100001  5 bit: base  5 bit: rt  16 bit: offset | Загрузить в rt 16 бит по адресу в base и смещению offset и расширить знаком |  |
| LHE | opcode: 011111  5 bit: base  5 bit: rt  9 bit: offset  “0”  operation: 101101 | Загрузить в rt 16 бит по адресу в base и смещению offset и расширить знаком  При расчёте адреса используется другая схема с использованием кэшей. |  |
| LHU | opcode: 100101  5 bit: base  5 bit: rt  16 bit: offset | Загрузить в rt 16 бит по адресу в base и смещению offset |  |
| LHUE | opcode: 011111  5 bit: base  5 bit: rt  9 bit: offset  “0”  operation: 101001 | Загрузить в rt 16 бит по адресу в base и смещению offset.  При расчёте адреса используется другая схема с использованием кэшей. |  |
| LL | opcode: 011111  5 bit: base  5 bit: rt  9 bit: offset  “0”  operation: 110110 | Загрузить в rt 16 бит по адресу в base и смещению offset.  Позволяет проводить атомарные операции read-modify-write. |  |
| LLE | opcode: 011111  5 bit: base  5 bit: rt  9 bit: offset  “0”  operation: 101110 | Загрузить в rt 16 бит по адресу в base и смещению offset.  Позволяет проводить атомарные операции read-modify-write.  При расчёте адреса используется другая схема с использованием кэшей. |  |
| LLWP | opcode: 011111  5 bit: base  5 bit: rt  5 bit: rd  “00001”  operation: 110110 | Загрузить пару смежных слов из памяти с адреса в base в rt и rd  При расчёте адреса используется другая схема с использованием кэшей. |  |
| LLWPE | opcode: 011111  5 bit: base  5 bit: rt  5 bit: rd  “00001”  operation: 101110 | Загрузить пару смежных слов из памяти с адреса в base в rt и rd  При расчёте адреса используется другая схема с использованием кэшей. |  |
| LSA | opcode: 000000  5 bit: rs  5 bit: rt  5 bit: rd  “000”  2 bit: sa  operation: 000101 | rd = rs << (sa+1) + rt |  |
| LUI | opcode: 001111  “00000”  5 bit: rt  16 bit: value | Загрузить value в верхнюю часть rt | Идиома для AUI r0, rt, value |
| LW | opcode: 100011  5 bit: base  5 bit: rt  16 bit: offset | Загрузить в rt 32 бита по адресу в base и смещению offset и расширить знаком |  |
| LWC1 | opcode: 110001  5 bit: base  5 bit: rt  16 bit: offset | Загрузить 32 бита по адресу в base и смещению offset в регистр ft FPU |  |
| LWC2 | opcode: 010010  5 bit: 01010  5 bit: rt  5 bit: base  11 bit: offset | Загрузить 32 бита в регистр rt COP2 |  |
| LWE | opcode: 011111  5 bit: base  5 bit: rt  9 bit: offset  “0”  operation: 101111 | Загрузить в rt 32 бита по адресу в base и смещению offset и расширить знаком.  При расчёте адреса используется другая схема с использованием кэшей. |  |
| LWPC | opcode: 111011  5 bit: rs  “01”  19 bit: offset | Загрузиь в rs слово по адресу PC + sign\_extend(offset << 2) |  |
| MFC0 | opcode: 010000  “00000”  5 bit: rt  5 bit: rd  8 bit: “0”  3 bit: sel | Загрузить в rt содержимое регистра COP0, определяемого через rd и sel. |  |
| MFC1 | opcode: 010001  “00000”  5 bit: rt  5 bit: fs  11 bit: “0” | Загрузить в rt содержимое регистра fs FPU (COP1). |  |
| MFC2 | opcode: 010010  “00000”  5 bit: rt  16 bit: reg | Загрузить в rt содержимое регистра reg FPU (COP2). |  |
| MFHC0 | opcode: 010000  “00010”  5 bit: rt  5 bit: rd  8 bit: “0”  3 bit: sel | Загрузить в rt содержимое верхних 32 бит регистра COP0, определяемого через rd и sel. |  |
| MFHC1 | opcode: 010001  “00011”  5 bit: rt  5 bit: fs  11 bit: “0” | Загрузить в rt содержимое верхних 32 бит регистра fs FPU (COP1). |  |
| MFHC2 | opcode: 010010  “00011”  5 bit: rt  16 bit: reg | Загрузить в rt содержимое верхних 32 бит регистра reg FPU (COP2). |  |
| MOV.fmt | opcode: 010001  5 bit: fmt  5 bit: “0”  5 bit: fs  5 bit: fd  “000110” | fd = fs  fd, fs – регистры FPU (COP1) |  |
| MTC0 | opcode: 010000  “00100”  5 bit: rt  5 bit: rd  8 bit: “0”  3 bit: sel | Загрузить в регистр COP0, определяемый через rd и sel, содержимое rt |  |
| MTC1 | opcode: 010001  “00100”  5 bit: rt  5 bit: fs  11 bit: “0” | Загрузить в регистр fs FPU (COP1) содержимое rt. |  |
| MTC2 | opcode: 010010  “00100”  5 bit: rt  16 bit: reg | Загрузить в регистр reg FPU (COP2) содержимое rt. |  |
| MTHC0 | opcode: 010000  “00110”  5 bit: rt  5 bit: rd  8 bit: “0”  3 bit: sel | Загрузить в верхние 32 бита регистра COP0, определяемого через rd и sel, содержимое rt |  |
| MTHC1 | opcode: 010001  “00110”  5 bit: rt  5 bit: fs  11 bit: “0” | Загрузить в верхние 32 бита регистра fs FPU (COP1) содержимое rt. |  |
| MTHC2 | opcode: 010010  “00011”  5 bit: rt  16 bit: reg | Загрузить содержимое rt в верхние 32 бита регистра reg FPU (COP2). |  |
| NAL | opcode: 000001  5 bit: “0”  “10000”  16 bit: offset | Запись в r31 значения PC  До Release 6 – идиома команды BLTZAL (убрана в Release 6) | **Deprecated, планируется к удалению.** |
| RHDWR | opcode: 011111  5 bit: “0”  5 bit: rt  5 bit: rd  “00”  3 bit: sel  operation: “111011” | Записать в rt значение специальных регистров (см. т. 5.4) по индексу rd, под-индексу sel |  |
| RDPGPR | opcode: 010000  “01010”  5 bit: rt  5 bit: rd  11 bit: “0” | Загрузить в rd значение из предшествующего набора теневых регистров по индексу rt |  |
| SB | opcode: 101000  5 bit: base  5 bit: rt  16 bit: offset | Загрузить в память по адресу в base и смещению offset младший байт из rt. |  |
| SBE | opcode: 011111  5 bit: base  5 bit: rt  9 bit: offset  “0”  operation: 011100 | Загрузить в память по адресу в base и смещению offset младший байт из rt. Использовать схему подсчёта адреса в командах EVA |  |
| SC | opcode: 011111  5 bit: base  5 bit: rt  9 bit: offset  “0”  operation: 100110 | Если бит LL установлен, то сохранить слово из rt по адресу в base и смещению offset и записать в rt значение бита LL.  Сбросить бит LL.  Парная LL команда. |  |
| SCE | opcode: 011111  5 bit: base  5 bit: rt  9 bit: offset  “0”  operation: 011110 | Если бит LL установлен, то сохранить слово из rt по адресу в base и смещению offset и записать в rt значение бита LL.  Сбросить бит LL.  Парная LLE команда.  Использовать схему подсчёта адреса в командах EVA |  |
| SCWP | opcode: 011111  5 bit: base  5 bit: rt  5 bit: rd  “00001”  operation: 100110 | Если бит LL установлен, то сохранить слово из rd и rt по адресу в base и смещению offset и записать в rt значение бита LL.  Сбросить бит LL.  Парная LLWP команда. |  |
| SCWPE | opcode: 011111  5 bit: base  5 bit: rt  5 bit: rd  “00001”  operation: 011110 | Если бит LL установлен, то сохранить слова из rd и rt по адресу в base и смещению offset и записать в rt значение бита LL.  Сбросить бит LL.  Парная LLWPE команда.  Использовать схему подсчёта адреса в командах EVA |  |
| SDC1 | opcode: 111101  5 bit: base  5 bit: rt  16 bit: offset | Загрузить в память по адресу в base и смещению offset содержимое регистра ft FPU (COP1) |  |
| SDC2 | opcode: 010010  “01111”  5 bit: rt  5 bit: base  11 bit: offset | Загрузить в память по адресу в base и смещению offset содержимое регистра rt COP2 |  |
| SH | opcode: 101001  5 bit: base  5 bit: rt  16 bit: offset | Загрузить в память по адресу в base и смещению offset младшие 2 байта из rt. |  |
| SHE | opcode: 011111  5 bit: base  5 bit: rt  9 bit: offset  “0”  operation: 011101 | Загрузить в память по адресу в base и смещению offset младшие 2 байта из rt. Использовать схему подсчёта адреса в командах EVA |  |
| SW | opcode: 101011  5 bit: base  5 bit: rt  16 bit: offset | Загрузить в память по адресу в base и смещению offset 32 бита из rt. |  |
| SWE | opcode: 011111  5 bit: base  5 bit: rt  9 bit: offset  “0”  operation: 011111 | Загрузить в память по адресу в base и смещению offset 32 бита из rt. Использовать схему подсчёта адреса в командах EVA |  |
| SWC1 | opcode: 111001  5 bit: base  5 bit: rt  16 bit: offset | Загрузить в память по адресу в base и смещению offset 32 бита из rt FPU |  |
| SWC2 | opcode: 010010  “01011”  5 bit: rt  5 bit: base  11 bit: offset | Загрузить в память по адресу в base и смещению offset 32 бита из rt COP2 |  |
| WRPGPR | opcode: 010000  “01110”  5 bit: rt  5 bit: rd  11 bit: “0” | Загрузить в rd из предшествующего набора теневых регистров значение rt |  |

#### Инструкции работы с кэшем

|  |  |  |  |
| --- | --- | --- | --- |
| Инструкция | Кодировка | Описание | Примечания |
| CACHE | opcode: 011111  5 bit: base  5 bit: op  9 bit: offset  “0”  operation: 100101 | 1. address = [base] + offset 2. В зависимости от операции (см. т. 3.{3-5}) |  |
| CACHEE | opcode: 011111  5 bit: base  5 bit: op  9 bit: offset  “0”  operation: 011011 | 1. address = [base] + offset 2. В зависимости от операции (см. т. 3.{3-5}) |  |
| GINVI | opcode: 011111  5 bit: rs  13 bit: “0”  “00”  “111101” | Сбросить все кэши инструкций.  Если rs != 0, то сбросить только кэш по индексу в младших битах rs. |  |
| GINVT | opcode: 011111  5 bit: rs  11 bit: “0”  2 bit: type  “10”  “111101” | Сбросить все TLB. Если rs != 0, то сбросить только TLB по индексу в младших битах rs. |  |
| PREF | opcode: 011111  5 bit: rs  5 bit: hint  9 bit: offset  “0”  operation: “110101” | Загрузить данные в кэш из паямти по адресу в base и смещению offset (знаковому) | hint: см. т. 5.2 |
| PREFE | opcode: 011111  5 bit: rs  5 bit: hint  9 bit: offset  “0”  operation: “100011” | Загрузить данные в кэш из паямти по адресу в base и смещению offset (знаковому).  Используется другая схема подсчёта адреса, аналогичная командам EVA (постфикс “E”) | ыва |
| TLBINV | opcode: 010000  “1”  19 bit: “0”  operation: “000011” | Сбросить TLB по ASID и регистру index |  |
| TLBINVF | opcode: 010000  “1”  19 bit: “0”  operation: “000100” | Сбросить TLB по регистру index |  |
| TLBP | opcode: 010000  “1”  19 bit: “0”  operation: “001000” | Поиск в TLB. Результат в регистре index |  |
| TLBR | opcode: 010000  “1”  19 bit: “0”  operation: “000001” | Чтение из TLB. Результат в регистрах EntryHI, EntryLo0, EntryLo1, PageMask |  |
| TLBWI | opcode: 010000  “1”  19 bit: “0”  operation: “000010” | Запись в TLB по адресу в Index. EntryHI, EntryLo0, EntryLo1, PageMask |  |
| TLBWR | opcode: 010000  “1”  19 bit: “0”  operation: “000110” | Запись в TLB по адресу, зависящему от реализации. EntryHI, EntryLo0, EntryLo1, PageMask |  |

#### Инструкции АЛУ

|  |  |  |  |
| --- | --- | --- | --- |
| Инструкция | Кодировка | Описание | Примечания |
| ADD | opcode: 000000  5 bit: rs  5 bit: rt  5 bit: rd  5 bit: 00000  operation: 100000 | rd = rs + rt | Может вызвать исключение переполнения; rd в этом случае не изменяется |
| ADDIU | opcode: 001001  5 bit: rs  5 bit: rt  16 bit: value (signed short) | rt = rs + value |  |
| ADDIUPC | opcode: 111011  5 bit: rs  “00”  19 bit: value | rs = PC + sign-extend (value << 2) |  |
| ADDU | opcode: 000000  5 bit: rs  5 bit: rt  5 bit: rd  5 bit: “00000”  operation: 100001 | rd = rs + rt |  |
| AND | opcode: 000000  5 bit: rs  5 bit: rt  5 bit: rd  “00000”  operation: 100100 | rd =rs & rt |  |
| ANDI | opcode: 001100  5 bit: rs  5 bit: rt  16 bit: value | rt = rs & zero\_extend(value) |  |
| AUI | opcode: 001111  5 bit: rs  5 bit: rt  16 bit: value (signed short) | rt = rs + sign\_extend(immediate << 16)  (Сложение константы со старшими 16 битами rs) |  |
| AUIPC | opcode: 111011  5 bit: rs  “11110”  16 bit: value (signed short) | rs = PC + immediate << 16 |  |
| CRC32B/H/W | opcode: 011111  5 bit: rs  5 bit: rt  “00000000”  2 bit: size (00-byte, 01- halfword, 10-word)  operation: 001111 | Сгенерировать CRC из rs и rt; результат в rt. Константа для CRC – 0xEDB88320 |  |
| CRC32CB/H/W | opcode: 011111  5 bit: rs  5 bit: rt  “00000001”  2 bit: size (00-byte, 01- halfword, 10-word)  operation: 001111 | Сгенерировать CRC из rs и rt; результат в rt. Константа для CRC – 0x82F63B78 |  |
| DIV | opcode: 000000  5 bit: rs  5 bit: rt  5 bit: rd  “00010”  operation: 011010 | rd = rs / rt  Целочисленное деление. |  |
| MOD | opcode: 000000  5 bit: rs  5 bit: rt  5 bit: rd  “00011”  operation: 011010 | rd = rs mod rt  Целочисленный остаток от деления.  Знак остатка такой же, как у делимого. |  |
| DIVU | opcode: 000000  5 bit: rs  5 bit: rt  5 bit: rd  “00010”  operation: 011011 | rd = rs / rt  Целочисленное деление.  Беззнаковая операция. |  |
| MODU | opcode: 000000  5 bit: rs  5 bit: rt  5 bit: rd  “00011”  operation: 011011 | rd = rs mod rt  Целочисленный остаток от деления.  Беззнаковая операция. |  |
| MUL | opcode: 000000  5 bit: rs  5 bit: rt  5 bit: rd  “00010”  operation: 011000 | rd = (rs \* rt)[0:31] |  |
| MUH | opcode: 000000  5 bit: rs  5 bit: rt  5 bit: rd  “00011”  operation: 011000 | rd = (rs \* rt)[32:63] |  |
| MULU | opcode: 000000  5 bit: rs  5 bit: rt  5 bit: rd  “00010”  operation: 011001 | rd = (rs \* rt)[0:31]  Unsigned-операция |  |
| MUHU | opcode: 000000  5 bit: rs  5 bit: rt  5 bit: rd  “00011”  operation: 011001 | rd = (rs \* rt)[32:63]  Unsigned-операция |  |
| NOP | 32 bit: “0” | No-OPeration | = SLL r0, r0, 0 |
| NOR | opcode: 000000  5 bit: rs  5 bit: rt  5 bit: rd  “00000”  operation: 100111 | rd = rs nor rt |  |
| OR | opcode: 000000  5 bit: rs  5 bit: rt  5 bit: rd  “00000”  operation: 100101 | rd = rs or rt |  |
| ORI | opcode: 001101  5 bit: rs  5 bit: rt  16 bit: value | rs = rt or value |  |
| ROTR | opcode: 000000  “00001”  5 bit: rt  5 bit: rd  5 bit: sa  operation: 000010 | Циклический сдвиг rt на sa. Результат в rd |  |
| ROTRV | opcode: 000000  5 bit: rs  5 bit: rt  5 bit: rd  “00001”  operation: 000110 | Циклический сдвиг rt на младшие 5 бит rs. Результат в rd |  |
| SELEQZ | opcode: 000000  5 bit: rs  5 bit: rt  5 bit: rd  “00000”  operation: 110101 | rd = rt ? 0 : rs |  |
| SELNEZ | opcode: 000000  5 bit: rs  5 bit: rt  5 bit: rd  “00000”  operation: 110111 | rd = rt ? rs : 0 |  |
| SLL | opcode: 000000  “00000”  5 bit: rt  5 bit: rd  5 bit: sa  operation: 000000 | rd = rt << sa |  |
| SLLV | opcode: 000000  5 bit: rs  5 bit: rt  5 bit: rd  “00000”  operation: 000100 | rd = rt << rs |  |
| SLT | opcode: 000000  5 bit: rs  5 bit: rt  5 bit: rd  “00000”  operation: 101010 | rd = (rs < rt) ? 1 : 0 |  |
| SLTI | opcode: 001010  5 bit: rs  5 bit: rt  16 bit: value | rd = (rs < sign\_extend(value)) ? 1 : 0 |  |
| SLTIU | opcode: 001011  5 bit: rs  5 bit: rt  16 bit: value | rd = (rs < sign\_extend(value)) ? 1 : 0  rs считать unsigned |  |
| SLTU | opcode: 000000  5 bit: rs  5 bit: rt  5 bit: rd  “00000”  operation: 101011 | rd = (rs < rt) ? 1 : 0  rs, rt считать unsigned |  |
| SRA | opcode: 000000  “00000”  5 bit: rt  5 bit: rd  5 bit: sa  operation: 000011 |  |  |
| SRAV | opcode: 000000  5 bit: rs  5 bit: rt  5 bit: rd  “00000”  operation: 000111 |  |  |
| SRL | opcode: 000000  “00000”  5 bit: rt  5 bit: rd  5 bit: sa  operation: 000010 |  |  |
| SRLV | opcode: 000000  5 bit: rs  5 bit: rt  5 bit: rd  “00000”  operation: 000110 |  |  |
| SSNOP | 21 bit: “0”  “00001”  6 bit: “0” | SuperScalar No-Operation | = SSL r0, r0, 1  deprecated |
| SUB | opcode: 000000  5 bit: rs  5 bit: rt  5 bit: rd  5 bit: 00000  operation: 100010 | rd = rs - rt |  |
| SUBU | opcode: 000000  5 bit: rs  5 bit: rt  5 bit: rd  5 bit: 00000  operation: 100011 | rd = rs - rt |  |
| XOR | opcode: 000000  5 bit: rs  5 bit: rt  5 bit: rd  “00000”  operation: 100110 | rd = rs xor rt |  |
| XORI | opcode: 001110  5 bit: rs  5 bit: rt  16 bit: value | rs = rt xor value |  |

Примечания:

* «U» в мнемонике инструкции обозначет «unsigned» ⬄ «Не вызывает исключения переполнения»

#### Инструкции передачи управления

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Инструкция | Кодировка | | Описание | | Примечания |
| B | opcode: 000100  “00000 00000”  16 bit: offset | | Безусловный переход на offset << 2 | | Идиома ассемблера MIPS.  Эквивалентна  “BEQ r0,r0,offset” |
| BAL | opcode: 000001  “00000”  operation: “10001”  16 bit: offset | | Поместить адрес возврата в R31 и перейти на offset << 2 | |  |
| BALC | opcode: 111010  26 bit: offest | | Поместить адрес возврата в R31 и перейти на offset << 2 | |  |
| BC | opcode: 110010  26 bit: offest | | Перейти на offset << 2 | |  |
| BC1EQZ | opcode: 010001  “01001”  5 bit: ft  16 bit: offset | | Если ft[0] == 0, перейти на offset << 2 | | ft – регистр COP1 |
| BC1NEZ | opcode: 010001  “01101”  5 bit: ft  16 bit: offset | | Если ft[0] != 0, перейти на offset << 2 | | ft – регистр COP1 |
| BC2EQZ | opcode: 010010  “01001”  5 bit: ct  16 bit: offset | | Если COP2Cond[ct] == 0, перейти на offset << 2 | | COP2Cond – состояние COP2, зависит от реализации COP2 |
| BC2NEZ | opcode: 010010  “01101”  5 bit: ct  16 bit: offset | | Если COP2Cond[ct] != 0, перейти на offset << 2 | |
| BEQ | opcode: 000100  5 bit: rs  5 bit: rt  16 bit: offset | | Если rs == rt, перейти на offset << 2 | |  |
| BGEZ | opcode: 000001  5 bit: rs  “00001”  16 bit: offset | | Если rs >= 0, перейти на offset << 2 | |  |
| BLEZALC | opcode: 000110  “00000”  5 bit: rt  16 bit: offset | | Если rt <=0, поместить адрес возврата в R31 и перейти на offset << 2 | |  |
| BGEZALC | opcode: 000110  10 bit: rs, rt (rs == rt != “00000”)  16 bit: offset | | Если rt >=0, поместить адрес возврата в R31 и перейти на offset << 2 | |  |
| BLTZALC | opcode: 000111  10 bit: rs, rt (rs == rt != “00000”)  16 bit: offset | | Если rt <0, поместить адрес возврата в R31 и перейти на offset << 2 | |  |
| BGTZALC | opcode: 000111  “00000”  5 bit: rt  16 bit: offset | | Если rt > 0, поместить адрес возврата в R31 и перейти на offset << 2 | |  |
| BEQZALC | opcode: 001000  5 bit: rs  “00001”  16 bit: offset | | Если rt == 0, поместить адрес возврата в R31 и перейти на offset << 2 | |  |
| BNEZALC | opcode: 011000  5 bit: rs  “00001”  16 bit: offset | | Если rt == 0, поместить адрес возврата в R31 и перейти на offset << 2 | |  |
| BEQC | opcode: 001000  10 bit: rs, rt (rs < rt != 0)  16 bit: offset | | Если rs == rt, перейти на offset << 2 | |  |
| BNEC | opcode: 011000  10 bit: rs, rt (rs < rt != 0)  16 bit: offset | | Если rs != rt, перейти на offset << 2 | |  |
| BLTC | opcode: 010111  10 bit: rs, rt (rs != rt != 0)  16 bit: offset | | Если rs < rt, перейти на offset << 2 | |  |
| BGEC | opcode: 010110  10 bit: rs, rt (rs != rt != 0)  16 bit: offset | | Если rs >= rt, перейти на offset << 2 | |  |
| BLTUC | opcode: 000111  10 bit: rs, rt (rs != rt != 0)  16 bit: offset | | Если rs < rt, перейти на offset << 2. rs и rt интерпретировать как unsigned | |  |
| BGEUC | opcode: 000110  10 bit: rs, rt (rs != rt != 0)  16 bit: offset | | Если rs >= rt, перейти на offset << 2. rs и rt интерпретировать как unsigned | |  |
| BGTC | Идиома для BLTC (обратный порядок операндов) | | | | |
| BLEC | Идиома для BGEC (обратный порядок операндов) | | | | |
| BGTUC | Идиома для BLTUC (обратный порядок операндов) | | | | |
| BLEUC | Идиома для BGEUC (обратный порядок операндов) | | | | |
| BLTZC | opcode: 010111  10 bit: rs, rt (rs == rt != 0)  16 bit: offset | Если rt < 0, перейти на offset << 2 | |  | |
| BLEZC | opcode: 010110  “00000”  5 bit: rt (!= 0)  16 bit: offset | Если rt <= 0, перейти на offset << 2 | |  | |
| BGEZC | opcode: 010110  10 bit: rs, rt (rs == rt != 0)  16 bit: offset | Если rt >= 0, перейти на offset << 2 | |  | |
| BGTZC | opcode: 010111  “00000”  5 bit: rt  16 bit: offset | Если rt > 0, перейти на offset << 2 | |  | |
| BEQZC | opcode: 110110  5 bit: rs  21 bit: offset | Если rs == 0, перейти на offset << 2 | |  | |
| BNEZC | opcode: 111110  5 bit: rs  21 bit: offset | Если rs != 0, перейти на offset << 2 | |  | |
| BGTZ | opcode: 000111  5 bit: rs  “00000”  16 bit: offset | Если rs > 0, перейти на offset << 2 | |  | |
| BLEZ | opcode: 000110  5 bit: rs  “00000”  16 bit: offset | Если rs <= 0, перейти на offset << 2 | |  | |
| BLTZ | opcode: 000001  5 bit: rs  “00000”  16 bit: offset | Если rs < 0, перейти на offset << 2 | |  | |
| BNE | opcode: 000101  5 bit: rs  5 bit: rt  16 bit: offset | Если rs != rt, перейти на offset << 2 | |  | |
| BOVC | opcode: 001000  5 bit: rs (rs >= rt)  5 bit: rt  16 bit: offset | Если есть переполнение, перейти на offset << 2 | |  | |
| BNVC | opcode: 011000  5 bit: rs (rs >= rt)  5 bit: rt  16 bit: offset | Если нет переполнения, перейти на offset << 2 | |  | |
| J | opcode: 000010  26 bit: offset | Перейти на offset << 2 | |  | |
| JAL | opcode: 000011  26 bit: offset | Поместить адрес возврата в R31 и перейти на offset << 2 | |  | |
| JALR | opcode: 000000  5 bit: rs  5 bit: “0”  5 bit: rd  5 bit: hint  operation: 001001 | Поместить адрес возврата в rd и перейти на адрес в rs | |  | |
| JALR.HB | opcode: 000000  5 bit: rs  5 bit: “0”  5 bit: rd  5 bit: “1” + hint  operation: 001001 | Поместить адрес возврата в rd и перейти на адрес в rs. Добавить execution hazard barrier после выполнения. | |  | |
| JIALC | opcode: 111110  5 bit: “0”  5 bit: rt  16 bit: value | Поместить адрес возврата в R31 и перейти на (rt + sign\_extend(offset) ) | |  | |
| JIC | opcode: 110110  5 bit: “0”  5 bit: rt  16 bit: value | Перейти на (rt + sign\_extend(offset) ) | |  | |
| JR | opcode: 000000  5 bit: rs  5 bit: “0”  5 bit: “0”  5 bit: hint  operation: 001001 | Перейти на адрес в rs | |  | |
| JR.HB | opcode: 000000  5 bit: rs  5 bit: “0”  5 bit: “0”  5 bit: “1” + hint  operation: 001001 | Перейти на адрес в rs. Добавить execution hazard barrier после выполнения. | |  | |

Примечания:

* Инструкции передачи управления не должны идти сразу за инструкциями перехода, если за этими инструкциями следует branch delay slot. Согласно MIPS 32&64 Release 6, при исполнении такой инструкции происходит Reserved Instruction Exception.
* Обычные переходы имеют branch delay slot и исполняются после команды, следующей за командой перехода. Адрес возврата – адрес *второй* инструкции после команды перехода.
* «Компактные» переходы (мнемоники оканчиваются на «C») исполняются без слота задержки. Адрес возврата – адрес *следующей* инструкции после команды перехода.

#### Инструкции исключений

|  |  |  |  |
| --- | --- | --- | --- |
| Инструкция | Кодировка | Описание | Примечания |
| BREAK | opcode: 000000  20 bit: data  operation: 001101 | Немедленная безусловная передача управления обработчику исключений | 20 бит могут быть использованы под параметры |
| DERET | opcode: 010000  “1”  19 bit: “0”  6 bit: “011111” | Возврат из обработчика исключений. | Адрес возврата – регистр DRPC |
| DI | opcode: 010000  “01011”  5 bit: rt  “01100”  “00000”  “0” “00” “000” | Сохранить значение регистра состояния (status) в rt и сбросить бит IE |  |
| DVP | opcode: 010000  “01011”  5 bit: rt  “00000”  “00000”  “1” “00” “000” | Отключить все виртуальные исполнители, кроме получившего инструкцию.  (тёмная магия) |  |
| EHB | opcode: 000000  “00000”  “00000”  “00000”  “00011”  “000000” | Execution Hazard Barrier  Остановить выполнение, пока не будут разрешены конфликты конвейера.  (тёмная магия конвейеров) | Технически это “SLL r0, r0, 3” |
| EI | opcode: 010000  “01011”  5 bit: rt  “01100”  “00000”  “1” “00” “000” | Сохранить значение регистра состояния (status) в rt и установить бит IE |  |
| ERET | opcode: 010000  “1”  19 bit: “0”  6 bit: “011000” | Возврат из прерывания |  |
| ERETNC | opcode: 010000  “1”  18 bit: “0”  “1”  6 bit: “011000” | Возврат из прерывания  Не сбрасывает бит LL |  |
| EVP | opcode: 010000  “01011”  5 bit: rt  “00000”  “00000”  “0” “00” “100” | Включить все виртуальные исполнители, кроме получившего инструкцию.  (тёмная магия) |  |
| PAUSE | 21 bit: 0  “00101”  6 bit:”0” | Ждать, пока LL-бит не будет сброшен |  |
| SDBBP | opcode: 000000  20 bit: code  operation: 001110 | Брейкпойнт. Вызывает debug breakpoint exception |  |
| SIGRIE | opcode: 000001  “00000”  “10111”  16 bit: code | Reserved Instruction Exception |  |
| SYNC | 21 bit: “0”  5 bit: stype (см. т. 5.5)  operation: “001111” | Для синхронизации доступа к памяти |  |
| SYNCI | opcode: 000001  5 bit: base  “11111”  16 bit: offset | Для синхронизации кэшей |  |
| SYSCALL | opcode: 000000  20 bit: code  operation: 001100 | Системный вызов |  |
| TEQ | opcode: 000000  5 bit: rs  5 bit: rt  10 bit: code  operation: 110100 | if rs == rt then TrapException() |  |
| TGE | opcode: 000000  5 bit: rs  5 bit: rt  10 bit: code  operation: 110000 | if rs >= rt then TrapException() |  |
| TGEU | opcode: 000000  5 bit: rs  5 bit: rt  10 bit: code  operation: 110001 | if rs >= rt then TrapException()  rs, rt считать unsigned |  |
| TLT | opcode: 000000  5 bit: rs  5 bit: rt  10 bit: code  operation: 110010 | if rs < rt then TrapException() |  |
| TLTU | opcode: 000000  5 bit: rs  5 bit: rt  10 bit: code  operation: 110011 | if rs < rt then TrapException()  rs, rt считать unsigned |  |
| TNE | opcode: 000000  5 bit: rs  5 bit: rt  10 bit: code  operation: 110110 | if rs != rt then TrapException() |  |
| WAIT | opcode: 000000  “1”  19 bit: code  operation: 100000 | Ждать события |  |

#### Инструкции сопроцессоров

|  |  |  |  |
| --- | --- | --- | --- |
| Инструкция | Кодировка | Описание | Примечания |
| ABS.fmt | opcode: 010001  5 bit: fmt  5 bit: 00000  5 bit: fs  5 bit: fd  operation: 000101 | fd = abs(fs) |  |
| ADD.fmt | opcode: 010001  5 bit: fmt  5 bit: ft  5 bit: fs  5 bit: fd  operation: 000000 | ft = fs + fd |  |
| CEIL.L.fmt | opcode: 010001  5 bit: fmt  5 bit: 00000  5 bit: fs  5 bit: fd  operation: 001010 | Преобразовать в 64-битный fixed-point формат и округлить в большую сторону значение в fs. Результат в fd (должен соответствовать 64-битному fixed-point формату) |  |
| CEIL.W.fmt | opcode: 010001  5 bit: fmt  5 bit: 00000  5 bit: fs  5 bit: fd  operation: 001110 | Преобразовать в 32-битный формат и округлить в большую сторону значение в fs. Результат в fd (должен соответствовать 32-битному fixed-point формату) |  |
| CLASS.fmt | opcode: 010001  5 bit: fmt  5 bit: 00000  5 bit: fs  5 bit: fd  operation: 011011 | Помещает 10-bit маску, отражающую состояние fs в fd. | Только для форматов S и D |
| CMP.condn.fmt | opcode: 010001  “10101”  5 bit: ft  5 bit: fs  5 bit: fd  “0”  5 bit: condn | fd = fs condn ft.  Сравнение согласно полю condn (см. т. 3.9) |  |
| COP2 | opcode: 010010  “1”  25 bit: operation | Операция operation COP2  COP2-dependent |  |
| CVT.D.fmt | opcode: 010001  5 bit: fmt  5 bit: 00000  5 bit: fs  5 bit: fd  operation: 100001 | Преобразовать в 64-битный floating-point формат и округлить соответственно настройкам сопроцессора значение в fs. Результат в fd (должен соответствовать 64-битному floating-point формату) | Округление настраивается в FCRS |
| CVT.L.fmt | opcode: 010001  5 bit: fmt  5 bit: 00000  5 bit: fs  5 bit: fd  operation: 100101 | Преобразовать в 64-битный fixed-point формат и округлить соответственно настройкам сопроцессора значение в fs. Результат в fd (должен соответствовать 64-битному fixed-point формату) |  |
| CVT.S.fmt | opcode: 010001  5 bit: fmt  5 bit: 00000  5 bit: fs  5 bit: fd  operation: 100000 | Преобразовать в 32-битный floating-point формат и округлить соответственно настройкам сопроцессора значение в fs. Результат в fd (должен соответствовать 32-битному floating-point формату) |  |
| CVT.W.fmt | opcode: 010001  5 bit: fmt  5 bit: 00000  5 bit: fs  5 bit: fd  operation: 100100 | Преобразовать в 32-битный fixed-point формат и округлить соответственно настройкам сопроцессора значение в fs. Результат в fd (должен соответствовать 32-битному fixed-point формату) |  |
| DIV.fmt | opcode: 010001  5 bit: fmt  5 bit: ft  5 bit: fs  5 bit: fd  operation: 000011 | fd = fs / ft |  |
| FLOOR.L.fmt | opcode: 010001  5 bit: fmt  5 bit: 00000  5 bit: fs  5 bit: fd  operation: 001011 | Преобразовать в 64-битный fixed-point формат и округлить в меньшую сторону значение в fs. Результат в fd (должен соответствовать 64-битному fixed-point формату) |  |
| FLOOR.W.fmt | opcode: 010001  5 bit: fmt  5 bit: 00000  5 bit: fs  5 bit: fd  operation: 001111 | Преобразовать в 32-битный формат и округлить в меньшую сторону значение в fs. Результат в fd (должен соответствовать 32-битному fixed-point формату) |  |
| MADDF.fmt | opcode: 010001  5 bit: fmt  5 bit: ft  5 bit: fs  5 bit: fd  operation: 011000 | fd += fs\*ft |  |
| MSUBF.fmt | opcode: 010001  5 bit: fmt  5 bit: ft  5 bit: fs  5 bit: fd  operation: 011001 | fd -= fs\*ft |  |
| MAX.fmt | opcode: 010001  5 bit: fmt  5 bit: ft  5 bit: fs  5 bit: fd  operation: 011110 | fd = max(fs, ft) |  |
| MIN.fmt | opcode: 010001  5 bit: fmt  5 bit: ft  5 bit: fs  5 bit: fd  operation: 011100 | fd = min(fs,ft) |  |
| MAXA.fmt | opcode: 010001  5 bit: fmt  5 bit: ft  5 bit: fs  5 bit: fd  operation: 011111 | fd = max(abs(fs), abs(ft)) |  |
| MINA.fmt | opcode: 010001  5 bit: fmt  5 bit: ft  5 bit: fs  5 bit: fd  operation: 011101 | fd = max(abs(fs), abs(ft)) |  |
| MUL.fmt | opcode: 010001  5 bit: fmt  5 bit: ft  5 bit: fs  5 bit: fd  operation: 000010 | fd = fs \* ft |  |
| NEG.fmt | opcode: 010001  5 bit: fmt  5 bit: “0”  5 bit: fs  5 bit: fd  operation: 000111 | fd = -fs |  |
| RECIP.fmt | opcode: 010001  5 bit: fmt  5 bit: “0”  5 bit: fs  5 bit: fd  operation: 010101 | fd = 1.0 / fs |  |
| RINT.fmt | opcode: 010001  5 bit: fmt  5 bit: “0”  5 bit: fs  5 bit: fd  operation: 011010 | Записать в fd целочисленное округление значения в fs |  |
| ROUND.L.fmt | opcode: 010001  5 bit: fmt  5 bit: “0”  5 bit: fs  5 bit: fd  operation: 001000 | Преобразовать fs в 64-битный формат с фиксированной точкой и округлить к ближайшему. Результат в fd. |  |
| ROUND.W.fmt | opcode: 010001  5 bit: fmt  5 bit: “0”  5 bit: fs  5 bit: fd  operation: 001100 | Преобразовать fs в 32-битный формат с фиксированной точкой и округлить к ближайшему. Результат в fd. |  |
| RSQRT.fmt | opcode: 010001  5 bit: fmt  5 bit: “0”  5 bit: fs  5 bit: fd  operation: 010110 | fd = 1.0 / sqrt(fs) |  |
| SEL.fmt | opcode: 010001  5 bit: fmt (s / d)  5 bit: ft  5 bit: fs  5 bit: fd  operation: 010000 | fd = fd[0] ? ft : fs |  |
| SELEQZ.fmt | opcode: 010001  5 bit: fmt (s / d)  5 bit: ft  5 bit: fs  5 bit: fd  operation: 010100 | fd = ft[0] ? 0 : fs |  |
| SELNEZ.fmt | opcode: 010001  5 bit: fmt (s / d)  5 bit: ft  5 bit: fs  5 bit: fd  operation: 010111 | fd = ft[0] ? fs : 0 |  |
| SQRT.fmt | opcode: 010001  5 bit: fmt (s / d)  “00000”  5 bit: fs  5 bit: fd  operation: 000100 | fd = sqrt(fs) |  |
| SUB.fmt | opcode: 010001  5 bit: fmt  5 bit: ft  5 bit: fs  5 bit: fd  operation: 000001 | fd = fs - ft |  |
| TRUNC.L.fmt | opcode: 010001  5 bit: fmt  5 bit: ft  5 bit: fs  5 bit: fd  operation: 001001 | Преобразовать fs в 64-битный формат с фиксированной точкой и округлить к меньшему. Результат в fd. |  |
| TRUNC.W.fmt | opcode: 010001  5 bit: fmt  5 bit: ft  5 bit: fs  5 bit: fd  operation: 001101 | Преобразовать fs в 32-битный формат с фиксированной точкой и округлить к меньшему. Результат в fd. |  |

Примечания:

* Необходимо соответствие операндов fmt, если в описании и примечаниях не указано иное. Иначе результат операции «unpredictable».
* fmt принимает следующие значения:
  + S – 10000
  + D – 10001
  + W – 10100
  + L – 10101

#### Инструкции работы с данными

|  |  |  |  |
| --- | --- | --- | --- |
| Инструкция | Кодировка | Описание | Примечания |
| ALIGN | opcode: 011111  5 bit: rs  5 bit : rt  5 bit: rd  “010”  2 bit: bp  operation: 100000 | Конкатенировать значения rt и rs и записать в rd 32 бита с позиции bp  rd = rt << (8\*bp) || rs >> GPRLEN-(8\*bp) | GPRLEN – размер РОН.  bp = 0 не рекомендуется, т.к. эквивалентно перемещению из rt в rd. |
| ALUIPC | opcode: 111011  5 bit: rs  “11111”  16 bit: value | rs = 0x0ffff & PC + sign\_extend(value << 16) |  |
| BITSWAP | opcode: 011111  “00000”  5 bit: rt  5 bit: rd  “00000”  operation: 100000 | Для всех байтов rt записать в rd их «перевёрнутые» копии. Порядок байтов оставить неизменным. |  |
| CLO | opcode: 000000  5 bit: rs  “00000”  5 bit: rd  “00001”  operation: 010001 | Подсчёт старших единиц в rs. Результат в rd. Если rs = 0xFFFF, то результат равен 32. |  |
| CLZ | opcode: 000000  5 bit: rs  “00000”  5 bit: rd  “00001”  operation: 010000 | Подсчёт старших нулей в rs. Результат в rd. Если rs = 0x0000, то результат равен 32. |  |
| EXT | opcode: 011111  5 bit: rs  5 bit: rt  5 bit: size  5 bit: pos  operation: 000000 | rt = rs[pos:size+1]  Получить срез rs с pos по size+1 |  |
| INS | opcode: 011111  5 bit: rs  5 bit: rt  5 bit: size  5 bit: pos  operation: 000100 | Вставить младшие size бит rs на pos в rt. |  |
| SEB | opcode: 011111  “00000”  5 bit: rt  5 bit: rd  “10000”  operation: 000100 | Расширить знаковым битом младший байт rt. Результат в rd. |  |
| SEH | opcode: 011111  “00000”  5 bit: rt  5 bit: rd  “11000”  operation: 000100 | Расширить знаковым битом младшие 16 бит rt. Результат в rd. |  |
| WSBH | opcode: 011111  “00000”  5 bit: rt  5 bit: rd  “00010”  operation: 100000 | rd = rt[23:16], rt[31:24], rt[0:7], rt[8:15] |  |

#### Инструкции, присутствующие только в MIPS64

|  |  |  |  |
| --- | --- | --- | --- |
| Инструкция | Кодировка | Описание | Примечания |
| DADD | opcode: 000000  5 bit: rs  5 bit: rt  5 bit: rd  “00000”  operation: 101100 | rd = rs + rt |  |
| DADDIU | opcode: 011001  5 bit: rs  5 bit: rt  16 bit: rd | rt = rs + sign\_extend(immediate) |  |
| DADDU | opcode: 000000  5 bit: rs  5 bit: rt  5 bit: rd  “00000”  operation: 101101 | rd = rs + rt | Unsigned операция |
| DCLO | opcode: 000000  5 bit: rs  5 bit: “0”  5 bit: rd  “00001”  operation: 010011 | Посчитать к-во 1 с самого старшего бита до первого 0 |  |
| DCLZ | opcode: 000000  5 bit: rs  5 bit: “0”  5 bit: rd  “00001”  operation: 010010 | Посчитать к-во 0 с самого старшего бита до первой 1 |  |
| DEXT | opcode: 011111  5 bit: rs  5 bit: rt  5 bit: size  5 bit: pos  operation: 000011 | rt = rs[pos:size+1]  Получить срез rs с pos по size+1 | Получает срез младших 32 бит |
| DEXTM | opcode: 011111  5 bit: rs  5 bit: rt  5 bit: size  5 bit: pos  operation: 000001 | rt = rs[pos:size+1]  Получить срез rs с pos по size+1 | Получает срез любых средних 32 бит |
| DEXTU | opcode: 011111  5 bit: rs  5 bit: rt  5 bit: size  5 bit: pos  operation: 000010 | rt = rs[pos:size+1]  Получить срез rs с pos по size+1 | Получает срез старших 32 бит |
| DINS | opcode: 011111  5 bit: rs  5 bit: rt  5 bit: size  5 bit: pos  operation: 000111 | Записать в rt срез rs с pos по size+1 | Запись в младшие 32 бита |
| DINSM | opcode: 011111  5 bit: rs  5 bit: rt  5 bit: size  5 bit: pos  operation: 000101 | Записать в rt срез rs с pos по size+1 | Запись в средние 32 бита |
| DINSU | opcode: 011111  5 bit: rs  5 bit: rt  5 bit: size  5 bit: pos  operation: 000110 | Записать в rt срез rs с pos по size+1 | Запись в старшие 32 бита |
| DMOD | opcode: 000000  5 bit: rs  5 bit: rt  5 bit: rd  “00011”  operation: 011110 | rd = rs mod rt |  |
| DDIV | opcode: 000000  5 bit: rs  5 bit: rt  5 bit: rd  “00010”  operation: 011110 | rd = rs / rt |  |
| DMODU | opcode: 000000  5 bit: rs  5 bit: rt  5 bit: rd  “00011”  operation: 011111 | rd = rs mod rt | Unsigned операции |
| DDIVU | opcode: 000000  5 bit: rs  5 bit: rt  5 bit: rd  “00010”  operation: 011111 | rd = rs / rt |
| DMFC0 | opcode: 010000  “00001”  5 bit: rt  5 bit: rd  8 bit: “0”  3 bit: sel | Загрузить в rt содержимое регистра COP0, определяемого через rd и sel. |  |
| DMFC1 | opcode: 010001  “00001”  5 bit: rt  5 bit: fs  11 bit: “0” | Загрузить в rt содержимое регистра fs FPU (COP1). |  |
| DMFC2 | opcode: 010010  “00001”  5 bit: rt  16 bit: reg | Загрузить в rt содержимое регистра reg FPU (COP2). |  |
| DMTC0 | opcode: 010000  “00101”  5 bit: rt  5 bit: rd  8 bit: “0”  3 bit: sel | Загрузить в регистр COP0, определяемый через rd и sel, содержимое rt |  |
| DMTC1 | opcode: 010001  “00101”  5 bit: rt  5 bit: fs  11 bit: “0” | Загрузить в регистр fs FPU (COP1) содержимое rt. |  |
| DMTC2 | opcode: 010010  “00101”  5 bit: rt  16 bit: reg | Загрузить в регистр reg FPU (COP2) содержимое rt. |  |
| DROTR | opcode: 000000  “00001”  5 bit: rt  5 bit: rd  5 bit: sa  operation: 111010 | Циклический сдвиг rt на sa.  Результат в rd. |  |
| DROTR32 | opcode: 000000  “00001”  5 bit: rt  5 bit: rd  5 bit: sa  operation: 111110 | Циклический сдвиг rt на (sa + 32).  Результат в rd. |  |
| DROTRV | opcode: 000000  5 bit: rs  5 bit: rt  5 bit: rd  “00001”  operation: 010110 | Циклический сдвиг rt на значение младших 6 бит в rs.  Результат в rd. |  |
| DSBH | opcode: 011111  “00000”  5 bit: rt  5 bit: rd  “00010”  operation: 100100 | В каждых 16 бит rt поменять местами позиции байт. Результат в rd. |  |
| DSHD | opcode: 011111  “00000”  5 bit: rt  5 bit: rd  “00101”  operation: 100100 | В каждых 32 бит rt поменять местами позиции 16-битных частей. Результат в rd. |  |
| DSLL | opcode: 000000  “00000”  5 bit: rt  5 bit: rd  5 bit: sa  operation: 111000 | rd = rt << sa |  |
| DSLL32 | opcode: 000000  “00000”  5 bit: rt  5 bit: rd  5 bit: sa  operation: 111100 | rd = rt << (sa + 32) |  |
| DSLLV | opcode: 000000  5 bit: rs  5 bit: rt  5 bit: rd  “00000”  operation: 010100 | rd = rt << rs |  |
| DSRA | opcode: 000000  “00000”  5 bit: rt  5 bit: rd  5 bit: sa  operation: 111011 | rd = rt >> sa  Сдвиг арифметический (знаковый бит сохраняется) |  |
| DSRA32 | opcode: 000000  “00000”  5 bit: rt  5 bit: rd  5 bit: sa  operation: 111111 | rd = rt >> (sa+32)  Сдвиг арифметический (знаковый бит сохраняется) |  |
| DSRAV | opcode: 000000  5 bit: rs  5 bit: rt  5 bit: rd  “00000”  operation: 010111 | rd = rt >> rs  Сдвиг арифметический (знаковый бит сохраняется) |  |
| DSRL | opcode: 000000  “00000”  5 bit: rt  5 bit: rd  5 bit: sa  operation: 111010 | rd = rt >> sa |  |
| DSRL32 | opcode: 000000  “00000”  5 bit: rt  5 bit: rd  5 bit: sa  operation: 111110 | rd = rt >> (sa+32) |  |
| DSRLV | opcode: 000000  5 bit: rs  5 bit: rt  5 bit: rd  “00000”  operation: 010110 | rd = rt >> rs |  |
| DSUB | opcode: 000000  5 bit: rs  5 bit: rt  5 bit: rd  “00000”  operation: 101110 | rd = rs – rt |  |
| DSUBU | opcode: 000000  5 bit: rs  5 bit: rt  5 bit: rd  “00000”  operation: 101111 | rd = rs – rt  Unsigned операция |  |
| LD | opcode: 110111  5 bit: base  5 bit: rt  16 bit: offset | Загрузить в rt значение по адресу в base со смещением offset |  |
| LLD | opcode: 011111  5 bit: base  5 bit: rt  9 bit: offset  “0”  operation: 110111 | Аналогично LLW |  |
| LLDP | opcode: 011111  5 bit: base  5 bit: rt  5 bit: rd  “0000”  “1”  operation: 110111 | Аналогично LLWP |  |
| DLSA | opcode: 000000  5 bit: base  5 bit: rt  5 bit: rd  “000”  2 but: sa  operation: 010101 | Аналогично LSA |  |
| LWU | opcode: 100111  5 bit: base  5 bit: rt  16 bit: offset | Загрузить в rt 32-битное значение по адресу в base со смещением offset как беззнаковое значение |  |
| LWUPC | opcode: 110111  5 bit: base  2 bit: “10”  19 bit: offset | Аналогично LWPC, но 32-битное значение интерпретируется как беззнаковое |  |
| SCD | opcode: 011111  5 bit: base  5 bit: rt  9 bit: offset  “0”  operation: 100111 | Парное LLD |  |
| SD | opcode: 110111  5 bit: base  5 bit: rt  16 bit: offset | Загрузить по адресу в base со смещением offset значение в rt |  |