**Code Address Registers**

The processor contains 16 code address registers (CA0-CA15). Several of the registers are reserved for predefined purposes. A code address register is used in the formation and storage of code addresses.

|  |  |  |
| --- | --- | --- |
| Reg # |  | Usage |
| 0 | Always Zero | Absolute address formation |
| 1 |  | Subroutine return address |
| 2 |  | This register is available for general use. |
| 3 |  | This register is available for general use. |
| 4 |  | This register is available for general use. |
| 5 |  | This register is available for general use. |
| 6 |  | This register is available for general use. |
| 7 |  | This register is available for general use. |
| 8 |  | This register is available for general use. |
| 9 |  |  |
| 10 |  |  |
| 11 | Catch Link Register | Used by the compiler to link to try/catch handlers. |
| 12 | Exception Table Pointer | This register points to the exception table in memory. |
| 13 | Exceptioned PC | This register is set when an exception occurs |
| 14 | Interrupted PC | This register is automatically set during a hardware interrupt |
| 15 | Program Counter | Relative address formation. |
|  |  |  |

Code address registers may be used to point to a block of code from which the JALR instruction can index into with its 24 bit offset. For instance a register may contain a pointer to a class method jump list; the JALR instruction can then index into this list in order to invoke a method.

The program counter code address register is read-only. The program counter cannot be modified by moving a value to this register.

## Flag Registers

The programming model contains eight flag registers. Flag registers are set by the compare and test instructions and are used for conditional branching.

|  |
| --- |
| f0 |
| f1 |
| f2 |
| f3 |
| f4 |
| f5 |
| f6 |
| f7 |

Each flag register holds four result status bits. These bits are the classic C,V,N, and Z bits.

|  |  |  |  |
| --- | --- | --- | --- |
| N | V | C | Z |

## Status Register (SR)

This register contains bits that control the overall operation of the processor or reflect the processor’s state. Bits are included for interrupt masking, and current operating privilege level. This register is split into two halves with both halves having the same format. The lower half of the register is what determines how the processor works. The upper half of the register maintains a backup copy of the lower half for interrupt processing. There are instructions provided for manipulating the interrupt mask.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 31..16 | 15 | 14 | 13 | 12 | 11..4 | 3..0 |
| same format as 15..0 | Interrupt Mask | Reserved | Reserved | Float Except. Enable |  | Current Privilege Level |
|  | IM | ~ | ~ | FXE | ~ | CPL |

 IM = interrupt mask

Maskable interrupts are disabled when this bit is set.

|  |  |  |  |
| --- | --- | --- | --- |
| 15 11 | 10 8 | 7 0 |  |
| Disp5 | Flg3 | Opcode8 | Bcc |

|  |  |  |  |
| --- | --- | --- | --- |
| 23 11 | 10 8 | 7 0 |  |
| Displacement13 | Flg3 | Opcode8 | LBcc |

|  |  |  |  |
| --- | --- | --- | --- |
| 23 16 | 15 8 | 7 0 |  |
| Disp8 | Ra8 | Opcode8 | BRZ/BRNZ/BRMI/BRPL |

|  |  |  |  |
| --- | --- | --- | --- |
| 15 12 | 11 8 | 7 0 |  |
| Bs4 | Bd4 | Opcode8 | JALR |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 39 16 | 15 12 | 11 8 | 7 0 |  |
| Displacement24 | Bs4 | Bd4 | Opcode8 | JALR |

|  |  |
| --- | --- |
| 7 0 |  |
| Opcode8 | RET |

|  |  |  |  |
| --- | --- | --- | --- |
| 15 8 | 7 3 | 2 0 |  |
| Ra8 | Opcode5 | Flg3 | TST Ft, Ra |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 23 16 | 15 8 | 7 3 | 2 0 |  |
| Rb8 | Ra8 | Opcode5 | Flg3 | CMP Ft, Ra, Rb |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 23 16 | 15 8 | 7 3 | 2 0 |  |
| Imm8 | Ra8 | Opcode5 | Flg3 | CMPI Ft,#i8 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 31 24 | 23 16 | 15 8 | 7 0 |  |
| Imm8 | Rt8 | Ra8 | Opcode8 | SEQ Rt,Ra,#i8 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 31 24 | 23 16 | 15 8 | 7 0 |  |
| Disp8 | Rt8 | Ra8 | Opcode8 | LB Rt,d8(Ra) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 31 24 | 23 16 | 15 8 | 7 0 |  |
| Rt8 | Rb8 | Ra8 | Opcode8 | LBX Rt,(Ra+Rb) |

# Instruction Set Description

A description of the instruction set follows.

## ADD - addition

ADD Rt, Ra, #i8

ADD Rt, Ra, Rb

Instruction Formats:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 048h | Rt8 | Rb8 | Ra8 | 028 | ADD Rt,Ra,Rb |
| Imm8 | Rt8 | Ra8 | 208 | ADD Rt,Ra,#imm |

Operation:

#### Register Immediate Form

Rt = Ra + immediate8

#### Register-Register Form

Rt = Ra + Rb

Notes:

The immediate constant may be extended up to 64 bits with immediate prefix instructions.

This instruction may cause an overflow exception if used with the overflow trap (TRAPV) prefix.

## AND – bitwise logical ‘and’

AND Rt, Ra, #i16

AND Rt, Ra, Rb

Instruction Formats:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 208 | Rt8 | Rb8 | Ra8 | 028 | AND Rt,Ra,Rb |
| Imm8 | Rt8 | Ra8 | 288 | AND Rt,Ra,#imm |

Operation:

#### Register Immediate Form

Rt = Ra & immediate16

#### Register-Register Form

Rt = Ra & Rb

Notes:

The immediate constant may be extended up to 64 bits with immediate prefix instructions.

## ASR – Arithmetic Shift Right

ASR Rt, Ra, #i6

ASR Rt, Ra, Rb

Instruction Formats:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 448 | Rt8 | Rb8 | | Ra8 | 28 | ASR Rt, Ra, Rb |
| 548 | Rt8 | ~ | Imm6 | Ra8 | 28 | ASR Rt, Ra, #i6 |

Operation:

#### Register Immediate Form

Rt = Ra >> immediate6

#### Register-Register Form

Rt = Ra >> Rb

Notes:

Performs an arithmetic shift right, preserving the sign bit of the value.

## Bcc – Branches

Bcc Fa,target\_address

Instruction Formats:

|  |  |  |  |
| --- | --- | --- | --- |
| 15 11 | 10 8 | 7 0 |  |
| Disp5 | Fa3 | 4x8h | Bcc address |

Operation:

If (cond) PC = PC + Disp5

Notes:

A branch is taken to the target address if the condition is true. Branches may branch forwards or backwards up to 16 bytes in range.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | |  |  | |  |
| 40h | BEQ | branch if equal | 48h | BGT | branch if greater than |
| 41h | BNE | branch if not equal | 49h | BLE | branch if less or equal |
| 42h | BVS | branch if overflow set | 4Ah | BGE | branch if greater or equal |
| 43h | BVC | branch if overflow clear | 4Bh | BLT | branch if less than |
| 44h | BMI | branch if negative | 4Ch | BHI | branch if higher |
| 45h | BPL | branch if positive or zero | 4Dh | BLS | branch if lower or same |
| 46h | BRA | branch all the time | 4Eh | BHS | branch if higher or same |
| 47h | BNV | never branch | 4Fh | BLO | branch if lower |

### BRK –Break

|  |
| --- |
| 7 0 |
| 00h |

#### Operation:

<none>

#### Description:

The Break exception is executed.

### CAS – Compare And Swap

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 39 32 | 31 24 | 23 16 | 15 8 | 7 0 |
| Rt | Rc | Rb | Ra | Opcode |
| Rt6 | Rc6 | Rb6 | Ra6 | 97h8 |

#### Operation:

Rt = memory [Ra + offset]

if memory[Ra+offset] = Rb

memory[Ra + offset] = Rc

#### Description:

If the contents of the addressed memory cell is equal to the contents of Rb then a sixty-four bit value is stored to memory from the source register Rc. The original contents of the memory cell are loaded into register Rt. The memory address is the sum of the sign extended offset and register Ra. The memory address must be word aligned. If the operation was successful then Rt and Rb will be the same value. The compare and swap operation is an atomic operation; the bus is locked during the load and potential store operation. This operation assumes that the addressed memory location is part of the volatile region of memory and bypasses the data cache.

#### Assembler:

CAS Rt,Rb,Rc,offset[Ra]

### CLI – Clear Interrupt Mask

|  |
| --- |
| 7 0 |
| Opcode |
| FAh8 |

#### Operation:

im = 0

#### Description:

This instruction is used to enable interrupts.

### CMP Register-Register Compare

CMP Ft,Ra,Rb

|  |  |  |  |
| --- | --- | --- | --- |
| 23 16 | 15 8 | 7 3 | 2 0 |
| Rb8 | Ra8 | 35 | Ft3 |

#### Operation:

Ft = flags of (Ra – Rb)

#### Description:

The register compare instruction compares two registers and sets the flags in the target flag register as a result.

### CMPI Register-Immediate Compare

CMP Ft,Ra,#imm

|  |  |  |  |
| --- | --- | --- | --- |
| 23 16 | 15 8 | 7 3 | 2 0 |
| Imm8 | Ra8 | 25 | Ft3 |

#### Operation:

Ft = flags of (Ra – Rb)

#### Description:

The register immediate compare instruction compares a register to an immediate value and sets the flags in the target flag register as a result.

### IMM64,IMM56,IMM48,IMM40,IMM32,IMM24,IMM16,IMM8

### Immediate Extensions

The immediate extension prefixes are used to extend the immediate constant of the following instruction. The extensions may add from one to eight bytes more to the constant. Most, but not all instructions can accept an immediate prefix.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | | Opcode8 |
| Immediate64 | | | | | | | | F8h |
| Immediate56 | | | | | | | F7h |
| Immediate48 | | | | | | F6h |
| Immediate40 | | | | | F5h |
| Immediate32 | | | | F4h |
| Immediate24 | | | F3h |
| Immediate16 | | F2h |
| Immediate8 | F1h |

## LB – Load Byte with Sign Extend

LB Rt, d(Rn)

LB Rt, o(Ra + Rb)

Instruction Formats:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Disp8 | Rt8 | Ra8 | 808h | LB Rt,d8(Ra) |
| Rt8 | Rb8 | Ra8 | 888h | LB Rt,(Ra+Rb) |

Operation:

#### Register Indirect with Displacement Form

Rt = sign extend(memory[displacement + Ra])

#### Register-Register Form

Rt = sign extend(memory[offset + Ra + Rb])

Notes:

The displacement constant may be extended up to 64 bits with immediate prefix instructions.

## LBcc – Long Branches

LBcc Fa,target\_address

Instruction Formats:

|  |  |  |  |
| --- | --- | --- | --- |
| 23 11 | 10 8 | 7 0 |  |
| Disp13 | Fa3 | 5x8h | Bcc address |

Operation:

PC = PC + Disp13

Notes:

A branch is taken to the target address if the condition is true. Branches may branch forwards or backwards up to 4096 bytes in range.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | |  |  | |  |
| 50h | LBEQ | branch if equal | 58h | LBGT | branch if greater than |
| 51h | LBNE | branch if not equal | 59h | LBLE | branch if less or equal |
| 52h | LBVS | branch if overflow set | 5Ah | LBGE | branch if greater or equal |
| 53h | LBVC | branch if overflow clear | 5Bh | LBLT | branch if less than |
| 54h | LBMI | branch if negative | 5Ch | LBHI | branch if higher |
| 55h | LBPL | branch if positive or zero | 5Dh | LBLS | branch if lower or same |
| 56h | LBRA | branch all the time | 5Eh | LBHS | branch if higher or same |
| 57h | LBNV | never branch | 5Fh | LBLO | branch if lower |

## NOP – No Operation

NOP

Instruction Formats:

|  |  |
| --- | --- |
| EA8 | NOP |

Operation:

none

Notes:

## POP – Pop Register

POP Rt

Instruction Formats:

|  |  |  |
| --- | --- | --- |
| Rt8 | A7h | POP |

Operation:

If Rt <> 0

Rt = memory[SP]

SP = SP + 8

Notes:

This instruction may be used to pop a register from the stack.

## PUSH – Push Register

PUSH Ra

Instruction Formats:

|  |  |  |
| --- | --- | --- |
| Ra8 | A6h | PUSH Ra |

Operation:

SP = SP - 8

memory[SP] = Ra

Notes:

This instruction may be used to push a register onto the stack.

## PUSHC – Push Constant

PUSHC #imm

Instruction Formats:

|  |  |  |
| --- | --- | --- |
| Constant16 | ADh | PUSHC #imm |

Operation:

SP = SP - 8

Memory[sp] = sign extend (immediate16)

Notes:

This instruction pushes a constant value onto the stack. The constant may be extended to 64 bits with a constant prefix instruction.

## ROL – Rotate Left

ROL Rt, Ra, #i6

ROL Rt, Ra, Rb

Instruction Formats:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 41h | Rt8 | Rb8 | | Ra8 | 02h | ROL Rt,Ra,Rb |
| 51h | Rt8 | ~ | Imm6 | Ra8 | 02h | ROL Rt,Ra,#i6 |

Operation:

#### Register Immediate Form

Rt = Ra << immediate6

#### Register-Register Form

Rt = Ra << Rb

Notes:

Most significant bits are rotated into the least significant bits.

## ROR – Rotate Right

ROR Rt, Ra, #i6

ROR Rt, Ra, Rb

Instruction Formats:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 43h | Rt8 | Rb8 | | Ra8 | 02h | ROR Rt,Ra,Rb |
| 53h | Rt8 | ~ | Imm6 | Ra8 | 02h | ROR Rt,Ra,#i6 |

Operation:

#### Register Immediate Form

Rt = Ra >> immediate6

#### Register-Register Form

Rt = Ra >> Rb

Notes:

The least significant bits are rotated to the most significant bits.

## SB – Store Byte

SB Rt, d8(Rn)

SB Rt, (Ra + Rb)

Instruction Formats:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Disp.8 | Rb8 | Ra8 | A08h | SB Rb,d8(Rn) |
| Rc8 | Rb8 | Ra8 | A88h | SB Rc, (Ra+Rb) |

Operation:

#### Register Indirect with Displacement Form

memory8[displacement + Ra] = Rb

#### Register-Register Form

memory[offset + Ra + Rb] = Rc

Notes:

The displacement constant may be extended up to 64 bits with immediate prefix instructions.

## SC – Store Character

SC Rb, d(Rn)

SC Rc, d(Ra + Rb)

Instruction Formats:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Disp8 | Rs8 | Ra8 | A18h | SC Rs,d8(Rn) |
| Rs8 | Rb8 | Ra8 | A98h | SC Rs,(Ra+Rb) |

Operation:

#### Register Indirect with Displacement Form

memory[displacement + Ra] = Rs

#### Register-Register Form

memory[offset + Ra + Rb] = Rs

Notes:

The displacement constant may be extended up to 64 bits with immediate prefix instructions.

## SH – Store Half-Word

SH Rb, d(Ra)

SH Rc, d(Ra + Rb)

Instruction Formats:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Displacement8 | Rb8 | Ra8 | A28h | SH Rb,d8(Ra) |
| Rc8 | Rb8 | Ra8 | AA8h | SH Rc, (Ra+Rb) |

Operation:

#### Register Indirect with Displacement Form

memory[displacement + Ra] = Rb

#### Register-Register Form

memory[offset + Ra + Rb] = Rc

Notes:

The displacement constant may be extended up to 64 bits with immediate prefix instructions.

## SHL – Shift Left

SHL Rt, Ra, #i6

SHL Rt, Ra, Rb

Instruction Formats:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 40h | Rt8 | Rb8 | | Ra8 | 02h | SHL Rt,Ra,Rb |
| 50h | Rt8 | ~ | Imm6 | Ra8 | 02h | SHL Rt,Ra,#i6 |

Operation:

#### Register Immediate Form

Rt = Ra << immediate6

#### Register-Register Form

Rt = Ra << Rb

Notes:

The least significant bits are loaded with zeros.

## SHR – Shift Right

SHR Rt, Ra, #i6

SHR Rt, Ra, Rb

Instruction Formats:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 42h | Rt8 | Rb8 | | Ra8 | 02h | SHR Rt,Ra,Rb |
| 52h | Rt8 | ~ | Imm6 | Ra8 | 02h | SHR Rt,Ra,#i6 |

Operation:

#### Register Immediate Form

Rt = Ra >> immediate6

#### Register-Register Form

Rt = Ra >> Rb

Notes:

The most significant bits are loaded with zeros.

## SW – Store Word

SW Rs, d(Rn)

SW Rs, d(Ra + Rb)

Instruction Formats:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Displacement8 | Rs8 | Ra8 | A38h | SW Rs,d8(Rn) |
| Rs8 | Rb8 | Ra8 | AC8h | SW Rs, (Ra+Rb) |

Operation:

#### Register Indirect with Displacement Form

memory[displacement + Ra] = Rs

#### Register-Register Form

memory[offset + Ra + Rb] = Rs

Notes:

The displacement constant may be extended up to 64 bits with immediate prefix instructions.

## SXB – Sign Extend Byte

SXB Rt, Ra

Instruction Formats:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 08h | Rt8 | Ra8 | 01h | SXB |

Operation:

#### Register Form

Rt = sign extend (Ra)

Notes:

The most significant bits (8 to 63) are loaded with the sign extension of bit 7.

## SXC – Sign Extend Character

SXC Rt, Ra

Instruction Formats:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 09h | Rt8 | Ra8 | 01h | SXC |

Operation:

#### Register Form

Rt = sign extend (Ra)

Notes:

The most significant bits (16 to 63) are loaded with the sign extension of bit 15.

## SXH – Sign Extend Half-Word

SXH Rt, Ra

Instruction Formats:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0Ah | Rt8 | Ra8 | 01h | SXH |

Operation:

#### Register Form

Rt = sign extend (Ra)

Notes:

The most significant bits (32 to 63) are loaded with the sign extension of bit 31.

### TST - Register Test Compare

TST Ft,Ra

Instruction Formats:

|  |  |  |
| --- | --- | --- |
| 15 8 | 7 3 | 2 0 |
| Ra8 | 15 | Ft3 |

#### Operation:

Ft = flags of (Ra – 0)

#### Description:

The register test compare compares a register against the value zero and sets the flags appropriately.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | x0 | x1 | x2 | x3 | x4 | x5 | x6 | x7 | x8 | x9 | xA | xB | xC | xD | xE | xF |  |
| 0x | BRK | {r} | {rr} |  |  |  |  |  | TST Ft,Ra | | | | | | | |  |
| 1x | CMPI Ft,Ra,#i8 | | | | | | | | CMP Ft,Ra,Rb | | | | | | | |  |
| 2x | ADD # | SUB # | MUL # | DIV # | REM # |  |  |  | AND # | OR # | EOR # |  |  |  |  |  |  |
| 3x | SEQ# | SNE# | MULU # | DIVU # | REMU # |  |  |  | SGT# | SLE# | SGE# | SLT# | SHI# | SLS# | SHS# | SLO# |  |
| 4x | BEQ | BNE | BVS | BVC | BMI | BPL | BRA | BRN | BGT | BLE | BGE | BLT | BHI | BLS | BHS | BLO |  |
| 5x | LBEQ | LBNE | LBVS | LBVC | LBMI | LBPL | LBRA | LBRN | LBGT | LBLE | LBGE | LBLT | LBHI | LBLS | LBHS | LBLO |  |
| 6x | JALR | JALR24 | TRAPV | RET |  |  |  |  | BRZ | BRNZ | BRMI | BRPL | DBNZ |  |  |  |  |
| 7x |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 8x | LB | LBU | LC | LCU | LH | LHU | LW | LEA | LBx | LBUx | LCx | LCUx | LHx | LHUx | LWx | LEAx |  |
| 9x |  |  |  |  |  |  |  | CAS |  |  |  |  |  |  |  |  |  |
| Ax | SB | SC | SH | SW | CINV | SWS | PUSH | POP | SBx | SCx | SHx | SWx | CINVx | PUSHC |  |  |  |
| Bx |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Cx |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Dx |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Ex |  |  |  |  |  |  |  |  |  |  | NOP |  |  |  |  |  |  |
| Fx |  | IMM1 | IMM2 | IMM3 | IMM4 | IMM5 | IMM6 | IMM7 | IMM8 |  | CLI |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |