|  |  |  |  |
| --- | --- | --- | --- |
| 23 16 | 15 11 | 10 6 | 5 0 |
| Immediate8 | Ra5 | Rt5 | Opcode6 |

|  |  |  |
| --- | --- | --- |
| 39 11 | 10 6 | 5 0 |
| Immediate29 | Rt5 | Opcode6 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 23 21 | 20 16 | 15 11 | 10 6 | 5 0 |
| Func3 | Rb5 | Ra5 | Rt5 | Opcode6 |

### {R2} Logic Operations

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 2 | 24  SEQ | 25  SNE | 26  SLT | 27  SLE | 28  SLTU | 29  SLEU | 30 | 31 |

### ADD – Addition

**Description:**

Add two source operands and place the result in the target register. Operands are treated as signed twos complement values.

**Instruction Format:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 31 26 | 25 21 | 20 16 | 15 11 | 10 6 | 5 0 |
| 46 | ~5 | Rb5 | Ra5 | Rt5 | 26 |

### ADDI – Add Immediate

**Description:**

Add two source operands and place the result in the target register. The first operand is in register Ra the second is an immediate specified in the instruction. The immediate value is sign extended to the left. Operands are treated as signed twos complement values.

**Instruction Format:**

|  |  |  |  |
| --- | --- | --- | --- |
| 31 16 | 15 11 | 10 6 | 5 0 |
| Immediate16 | Ra5 | Rt5 | 46 |

### ADDIS – Add Immediate Shifted

**Description:**

Add two source operands and place the result in the target register. The first operand is in register Ra the second is an immediate shifted left 16 times specified in the instruction. Operands are treated as signed twos complement values.

**Instruction Format:**

|  |  |  |  |
| --- | --- | --- | --- |
| 31 16 | 15 11 | 10 6 | 5 0 |
| Immediate16 | Ra5 | Rt5 | 36 |

### AND – Bitwise ‘And’

**Description:**

Bitwise ‘and’ two source operands and place the result in the target register.

**Instruction Format:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 31 26 | 25 21 | 20 16 | 15 11 | 10 6 | 5 0 |
| 86 | ~5 | Rb5 | Ra5 | Rt5 | 26 |

### ANDI – Bitwise ‘And’ Immediate

**Description:**

Bitwise ‘And’ two source operands and place the result in the target register. The first operand is in register Ra the second is an immediate padded with one’s on the left, specified in the instruction.

**Instruction Format:**

|  |  |  |  |
| --- | --- | --- | --- |
| 31 16 | 15 11 | 10 6 | 5 0 |
| Immediate16 | Ra5 | Rt5 | 86 |

### ANDIS – Bitwise ‘And’ Immediate Shifted

**Description:**

Bitwise ‘And’ two source operands and place the result in the target register. The first operand is in register Ra the second is an immediate shifted left 16 times and padded with one’s on the right, specified in the instruction. Operands are treated as signed twos complement values.

**Instruction Format:**

|  |  |  |  |
| --- | --- | --- | --- |
| 31 16 | 15 11 | 10 6 | 5 0 |
| Immediate16 | Ra5 | Rt5 | 116 |

### APCIS – Add Immediate Shifted to Program Counter

**Description:**

Add an immediate value to the program counter and place the result in the target register. The first operand is the program counter, the second is an immediate shifted left 16 times specified in the instruction. Operands are treated as signed twos complement values. This instruction is used to form program counter relative addresses.

**Instruction Format:**

|  |  |  |  |
| --- | --- | --- | --- |
| 31 16 | 15 11 | 10 6 | 5 0 |
| Immediate16 | ~5 | Rt5 | 36 |

### ASL – Arithmetic Shift Left

**Description:**

Arithmetic shift left a source operand by a second source operand and place the result in the target register. Both operands are in registers. The least significant bits of the instruction are filled from the ‘F’ bit of the instruction.

**Instruction Format:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 31 26 | 25 | 24 | 23 21 | 20 16 | 15 11 | 10 6 | 5 0 |
| 326 | F | 0 | ~3 | Rb5 | Ra5 | Rt5 | 26 |

### ASLI – Arithmetic Shift Left by Immediate

**Description:**

Arithmetic shift left a source operand by a second source operand and place the result in the target register. The first operand is in a register, the second operand is an immediate constant in the instruction. The least significant bits of the instruction are filled from the ‘F’ bit of the instruction.

**Instruction Format:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 31 26 | 25 | 24 | 23 21 | 20 16 | 15 11 | 10 6 | 5 0 |
| 326 | F | 1 | ~3 | Imm5 | Ra5 | Rt5 | 26 |

### BEQ – Branch if Equal

**Description:**

If Ra and Rb are equal then branch to the target address. The target address is formed as the sum of the program counter and an immediate constant in the instruction.

**Instruction Format:**

|  |  |  |  |
| --- | --- | --- | --- |
| 31 16 | 15 11 | 10 6 | 5 0 |
| Displacement16 | Rb5 | Ra5 | 246 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 31 24 | 23 21 | 20 16 | 15 11 | 10 6 | 5 0 |
| Disp8 | 03 | Rc5 | Rb5 | Ra5 | 306 |

### BRA – Branch

**Description:**

Branch to a target address. The target address is the sum of the program counter and a constant in the instruction.

**Instruction Format:**

|  |  |  |
| --- | --- | --- |
| 31 8 | 7 6 | 5 0 |
| Displacement24 | 02 | 186 |

### BRK – Breakpoint Exception

**Description:**

**Instruction Format:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 31 26 | 25 21 | 20 16 | 15 11 | 10 6 | 5 0 |
| 06 | ~5 | ~5 | ~5 | ~5 | 06 |

### CALL – Call Subroutine

**Description:**

Call a subroutine by storing the program counter in a link register then branching to a target address. The target address is the sum of the program counter and a constant in the instruction.

**Instruction Format:**

|  |  |  |
| --- | --- | --- |
| 31 8 | 7 6 | 5 0 |
| Displacement24 | Lr2 | 186 |

### CMP - Comparison

**Description:**

Compare two source operands and place the result in the target register. The result is a bit vector identifying the relationship between the two source operands as signed and unsigned integers.

**Operation:**

Rt = Ra ? Rb

**Clock Cycles:** 1

**Execution Units:** All Integer ALU’s

**Exceptions:** none

**Notes:**

**Instruction Format:** R2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 31 26 | 25 21 | 20 16 | 15 11 | 10 6 | 5 0 |
| 56 | ~5 | Rb5 | Ra5 | Rt5 | 26 |

|  |  |  |  |
| --- | --- | --- | --- |
| Rt Bit | Mnem. | Meaning | Test |
|  |  | **Integer Compare Results** |  |
| 0 | EQ | = equal | a == b |
| 1 | NE | < > not equal | a <> b |
| 2 | LT | < less than | a < b |
| 3 | LE | <= less than or equal | a <= b |
| 4 | GE | >= greater than or equal | a >= b |
| 5 | GT | > greater than | a > b |
| 6 | BC | Bit clear | !a[b] |
| 7 | BS | Bit set | a[b] |
| 8 | LTZ | < 0 | a < 0 |
| 9 | GTZ | > 0 | a > 0 |
| 10 | LO / CS | < unsigned less than | a < b |
| 11 | LS | <= unsigned less than or equal | a <= b |
| 12 | HS / CC | unsigned greater than or equal | a >= b |
| 13 | HI | unsigned greater than | a > b |
| 14 |  |  |  |
| 15 |  |  |  |

### IRQ – Interrupt Request

**Description:**

Triggers an interrupt request.

**Instruction Format:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 31 26 | 25 | 24 16 | 15 11 | 10 9 | 8 6 | 5 0 |
| 16 | ~ | IRQ Number9 | ~5 | ~2 | Lvl3 | 06 |

### JAL – Jump and Link

**Description:**

Jump to a target address. The target address is a constant in the instruction. The address of the JAL instruction is stored in the target register.

**Instruction Format:**

|  |  |  |  |
| --- | --- | --- | --- |
| 31 16 | 15 11 | 10 6 | 5 0 |
| Immediate16 | Ra5 | Rt5 | 196 |

### OR – Bitwise ‘Or’

**Description:**

Bitwise ‘or’ two source operands and place the result in the target register.

**Instruction Format:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 23 21 | 20 16 | 15 11 | 10 6 | 5 0 |
| 13 | Rb5 | Ra5 | Rt5 | 36 |

### LDB – Load Byte

**Description:**

Load a byte from the memory addressed as the sum of register Ra and a displacement constant encoded in the instruction. The loaded byte is sign extended to the width of the machine.

**Instruction Format:**

|  |  |  |  |
| --- | --- | --- | --- |
| 23 16 | 15 11 | 10 6 | 5 0 |
| Immediate8 | Ra5 | Rt5 | 326 |

### LDBX – Load Byte, Indexed

**Description:**

Load a byte from the memory addressed as the sum of register Ra and scaled register Rb. The loaded byte is sign extended to the width of the machine.

**Instruction Format:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 23 | 22 21 | 20 16 | 15 11 | 10 6 | 5 0 |
| ~ | Sc2 | Rb5 | Ra5 | Rt5 | 406 |

### RTD – Return and Deallocate

**Description:**

Load the PC from LR2 plus an eight-bit offset and add a constant to the SP. The default offset value is four so return to the next instruction after the call instruction is performed.

**Instruction Format:**

|  |  |  |  |
| --- | --- | --- | --- |
| 31 16 | 15 8 | 7 6 | 5 0 |
| Imm16 | Offs8 | Lr2 | 176 |

### RTI – Return from Interrupt

**Description:**

Load the PC from the internal exception stack. Add offset8 to the PC. Usually, offset8 will be zero to return to the interrupted instruction.

**Instruction Format:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 31 26 | 25 16 | 15 8 | 7 6 | 5 0 |
| 46 | ~10 | Offset8 | 0 | 06 |

### RTS – Return from System

**Description:**

Load the PC from the internal exception stack or perform a two-up level return using the internal exception stack. Add offset8 to the PC. Usually, offset8 will be four to return to the next instruction. It may be larger if there are inline parameters.

**Instruction Format:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 31 26 | 25 16 | 15 8 | 7 6 | 5 0 |
| 36 | ~10 | Offset8 | Dep | 06 |

|  |  |
| --- | --- |
| 7 6 | Mnemonic |
| 1 | RTS |
| 2 | RTS2UP |

### SYS – System Call

**Description:**

.

**Instruction Format:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 31 26 | 25 | 24 16 | 15 11 | 10 6 | 5 0 |
| 26 | ~ | Call Number9 | ~5 | ~5 | 06 |

### Major Opcode Group

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0x | 0  {SYS} | 1  APCIS | 2  {R2} | 3  ADDIS | 4  ADDI | 5  CMPI | 6  MULI | 7  DIVI |
|  | 8  ANDI | 9  ORI | 10  EORI | 11  ANDIS | 12  SUBFI | 13  SLTI | 14  MULUI | 15  DIVUI |
| 1x | 16  ERET | 17  RETD | 18  CALL  BRA  JMP | 19  JAL | 20  ORIS | 21  EORIS | 22 | 23 |
|  | 24  BEQ | 25  BNE | 26  BLT | 27  BLE | 28  BLTU | 29  BLEU | 30  BccR | 31  BBS |
| 2x | 32  LDB | 33  LDBU | 34  LDW | 35  LDWU | 36  LDT | 37 | 38  LDA | 39  CACHE |
|  | 40  LDBX | 41  LDBUX | 42  LDWX | 43  LDWUX | 44  LDTX | 45 | 46  LDAX | 47  CACHEX |
| 3x | 48  STB | 49  STW | 50  STT | 51 | 52  STBX | 53  STWX | 54  STTX | 55 |
|  | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 |

### {SYS} System Operations

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0x | 0  BRK | 1  IRQ | 2  SYS | 3  RTS | 4  RTI | 5 | 6 | 7 |
|  | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |

### {R2} Arithmetic Operations

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0x | 0 | 1  {R1} | 2  ADDPC | 3 | 4  ADD | 5  CMP | 6  MUL | 7  DIV |
|  | 8  AND | 9  OR | 10  EOR | 11  ANDN | 12  SUB | 13  SLT | 14  MULU | 15  DIVU |
| 1x | 16  NAND | 17  NOR | 18  ENOR | 19  ORN | 20 | 21 | 22  MULH | 23  MOD |
|  | 24  SEQ | 25  SNE | 26  SLT | 27  SLE | 28  SLTU | 29  SLEU | 30  MULUH | 31  MODU |
| 2x | 32  ASL | 33  LSR | 34  ROL | 35  ROR | 36  ASR | 37 | 38 | 39 |
|  | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
| 3x | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 |
|  | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 |