Thor Guide

[Type the document subtitle]

[Type the abstract of the document here. The abstract is typically a short summary of the contents of the document. Type the abstract of the document here. The abstract is typically a short summary of the contents of the document.]

2013

robfinch@Finitron.ca

Finitron

1/1/2013

Table of Contents

[Programming Model 4](#_Toc371591960)

[General Registers 4](#_Toc371591961)

[Predicates 5](#_Toc371591962)

[Predicate Conditions 5](#_Toc371591963)

[Instruction Formats 6](#_Toc371591964)

[RR - Register-Register 6](#_Toc371591965)

[RI - Register-Immediate 6](#_Toc371591966)

[CMP Register-Register Compare 6](#_Toc371591967)

[CMPI Register-Immediate Compare 6](#_Toc371591968)

[TST - Register Test Compare 6](#_Toc371591969)

[CTRL- Control 6](#_Toc371591970)

[BR - Relative Branch 6](#_Toc371591971)

[BRK/NOP 6](#_Toc371591972)

[JSR - Jump To Subroutine 6](#_Toc371591973)

[Instruction Set 8](#_Toc371591974)

[2ADDU - Register-Register 8](#_Toc371591975)

[2ADDUI - Register-Immediate 8](#_Toc371591976)

[4ADDU - Register-Register 9](#_Toc371591977)

[4ADDUI - Register-Immediate 9](#_Toc371591978)

[8ADDU - Register-Register 10](#_Toc371591979)

[8ADDUI - Register-Register 10](#_Toc371591980)

[16ADDU - Register-Register 11](#_Toc371591981)

[16ADDUI - Register-Register 11](#_Toc371591982)

[ADD - Register-Register 12](#_Toc371591983)

[ADDI - Register-Immediate 12](#_Toc371591984)

[ADDU - Register-Register 12](#_Toc371591985)

[ADDUI - Register-Immediate 12](#_Toc371591986)

[AND - Register-Register 13](#_Toc371591987)

[ANDI - Register-Immediate 13](#_Toc371591988)

[BR - Relative Branch 14](#_Toc371591989)

[BRK –Break 15](#_Toc371591990)

[BSR - Branch To Subroutine 16](#_Toc371591991)

[CLI – Clear Interrupt Mask 17](#_Toc371591992)

[CMP Register-Register Compare 18](#_Toc371591993)

[CMPI Register-Immediate Compare 18](#_Toc371591994)

[EOR - Register-Register 19](#_Toc371591995)

[EORI - Register-Immediate 19](#_Toc371591996)

[IMM64,IMM56,IMM48,IMM40,IMM32,IMM24,IMM16 20](#_Toc371591997)

[Immediate Extensions 20](#_Toc371591998)

[INT –Interrupt 21](#_Toc371591999)

[JMP - Jump To Address 22](#_Toc371592000)

[JSR - Jump To Subroutine Instruction 22](#_Toc371592001)

[LOOP – Loop Branch 23](#_Toc371592002)

[MFSPR – Special Register-Register 24](#_Toc371592003)

[MTSPR –Register-Special Register 24](#_Toc371592004)

[NOP – No Operation 25](#_Toc371592005)

[OR - Register-Register 26](#_Toc371592006)

[ORI - Register-Immediate 26](#_Toc371592007)

[RTE – Return From Exception Routine 27](#_Toc371592008)

[*Description* 27](#_Toc371592009)

[RTI – Return From Interrupt Routine 27](#_Toc371592010)

[*Description* 27](#_Toc371592011)

[RTS – Return From Subroutine 28](#_Toc371592012)

[*Description* 28](#_Toc371592013)

[SEI – Set Interrupt Mask 29](#_Toc371592014)

[STSB – Store String Byte 29](#_Toc371592015)

[STSW – Store String Word 29](#_Toc371592016)

[SUB - Register-Register 30](#_Toc371592017)

[SUBI - Register-Immediate 30](#_Toc371592018)

[SYNC – Synchronize Memory 31](#_Toc371592019)

[SYS –Call system routine 32](#_Toc371592020)

[TST - Register Test Compare 33](#_Toc371592021)

# Programming Model

## General Registers

There are 256 general purpose registers. General purpose registers are 64 bits wide.

Register #0 is always zero.

**Branch Registers**

The processor contains 16 branch registers (BR0-BR15). Several of branch registers are reserved for special purposes.

|  |  |  |
| --- | --- | --- |
| Reg # |  | Usage |
| 0 | Always Zero |  |
| 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 |  |  |
| 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 |  |
|  |  |  |

## Predicates

The processor contains 16 predicate registers each of which hold three flags. These flags are set as the result of a compare operation. The flags represent equality (eq) signed less than (lt) and unsigned less than (ltu). It should be noted that the compare instruction can’t overflow.

|  |  |  |  |
| --- | --- | --- | --- |
| 3 | 2 | 1 | 0 |
| ~ | ltu | lt | eq |

All instructions are executed conditionally determined by the value of a predicate register. The special predicate 00 executes the break vector.

### Predicate Conditions

|  |  |  |  |
| --- | --- | --- | --- |
| Cond. |  | Test |  |
| 0 | PF | 0 | Always false – Instructions predicated with condition zero never execute regardless of the predicate register contents. This is used for extended immediate values as well. |
| 1 | PT | 1 | Always True – The instruction predicated with an always true condition always executes regardless of the predicate register contents. |
| 2 | PEQ | eq | Equal – instruction executes if the predicate register equal flag is set |
| 3 | PNE | !eq | Not Equal – instruction executes if the predicate register equal flag is clear |
| 4 | PLE | lt|eq | Less or Equal – predicate less or equal flag is set |
| 5 | PGT | !(lt|eq) | greater than |
| 6 | PGE | !lt | greater or equal |
| 7 | PLT | lt | less than |
| 8 | PLEU | ltu|eq | unsigned less or equal |
| 9 | PGTU | !(ltu|eq) | unsigned greater than |
| 10 | PGEU | !ltu | unsigned greater or equal |
| 11 | PLTU | ltu | unsigned less than |
|  |  |  |  |

## Instruction Formats

Instructions vary in length from one to eight bytes. There are only a couple of single byte instructions consisting of only a predicate.

All instruction sequences begin with a predicate byte that determines the conditions under which the instruction executes. With the exception of special predicate values, the next field in the instruction is always the opcode byte. All opcodes may be preceded by an extended constant value.

### RR - Register-Register

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 39 32 | 31 24 | 23 16 | 15 8 | 7 0 | |
| Rt | Rb | Ra | Opcode | Predicate | |
| Rt8 | Rb8 | Ra8 | Opcode8 | Pn4 | Pc4 |

### RI - Register-Immediate

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 39 32 | 31 24 | 23 16 | 15 8 | 7 0 | |
| Immediate | Rt | Ra | Opcode | Predicate | |
| Immediate7..0 | Rt8 | Ra8 | Opcode8 | Pn4 | Pc4 |

### CMP Register-Register Compare

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 31 24 | 23 16 | 15 12 | 11 8 | 7 0 | |
| Rb | Ra | Opc | Pt | Predicate | |
| Rb8 | Ra8 | 14 | Pt4 | Pn4 | Pc4 |

### CMPI Register-Immediate Compare

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 31 24 | | 23 16 | | 15 12 | | 11 8 | | 7 0 | |
| Immed | | Ra | | Opc | | Pt | | Predicate | |
| Immed8 | | Ra8 | | 24 | | Pt4 | | Pn4 | Pc4 |
| TST - Register Test Compare | | | | | | | |  | |  | CTRL- Control | | |
| 23 16 | 15 12 | | 11 8 | | 7 0 | | |  | |  | 15 8 | 7 0 | |
| Ra | Opc | | Pt | | Predicate | | |  | |  | Opcode | Predicate | |
| Ra8 | 04 | | Pt4 | | Pn4 | | Pc4 |  | |  | Opcode8 | Pn4 | Pc4 |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| BR - Relative Branch | | | | |  |  | | BRK/NOP | |
| 23 16 | 15 8 | | 7 0 | |  |  | | 7 0 | |
| Disp. | Opc | Disp. | Predicate | |  |  | | Predicate | |
| Disp7..0 | 34 | D11..8 | Pn4 | Pc4 |  | |  | 0/14 | 04 |

### JSR - Jump To Subroutine

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Disp. | Br | Brt | Opcode | Predicate | |
| Offset7..0 | Br4 | Brt4 | Opcode8 | Pn4 | Pc4 |

# Instruction Set

### 2ADDU - Register-Register

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 39 32 | 31 24 | 23 16 | 15 8 | 7 0 | |
| Rt | Rb | Ra | Opcode | Predicate | |
| Rt8 | Rb8 | Ra8 | 40h8 | Pn4 | Pc4 |

#### Operation:

Rt = Ra \* 2 + Rb

#### Description:

Multiply Ra by two and add Rb and place the sum in the target register. This instruction will never cause an overflow exception.

### 2ADDUI - Register-Immediate

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 39 32 | 31 24 | 23 16 | 15 8 | 7 0 | |
| Immed | Rb | Ra | Opcode | Predicate | |
| Immed8 | Rb8 | Ra8 | 6Bh8 | Pn4 | Pc4 |

#### Operation:

Rt = Ra \* 2 + immediate

#### Description:

Multiply Ra by two and add immediate and place the sum in the target register. This instruction will never cause an overflow exception.

### 4ADDU - Register-Register

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 39 32 | 31 24 | 23 16 | 15 8 | 7 0 | |
| Rt | Rb | Ra | Opcode | Predicate | |
| Rt8 | Rb8 | Ra8 | 57h8 | Pn4 | Pc4 |

#### Operation:

Rt = Ra \* 4 + Rb

#### Description:

Multiply Ra by four and add Rb and place the sum in the target register. This instruction will never cause an overflow exception.

### 4ADDUI - Register-Immediate

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 39 32 | 31 24 | 23 16 | 15 8 | 7 0 | |
| Immed | Rb | Ra | Opcode | Predicate | |
| Immed8 | Rb8 | Ra8 | 6Ch8 | Pn4 | Pc4 |

#### Operation:

Rt = Ra \* 4 + immediate

#### Description:

Multiply Ra by four and add immediate and place the sum in the target register. This instruction will never cause an overflow exception.

### 8ADDU - Register-Register

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 39 32 | 31 24 | 23 16 | 15 8 | 7 0 | |
| Rt | Rb | Ra | Opcode | Predicate | |
| Rt8 | Rb8 | Ra8 | 64h8 | Pn4 | Pc4 |

#### Operation:

Rt = Ra \* 8 + Rb

#### Description:

Multiply Ra by eight and add Rb and place the sum in the target register. This instruction will never cause an overflow exception.

### 8ADDUI - Register-Register

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 39 32 | 31 24 | 23 16 | 15 8 | 7 0 | |
| Immed | Rb | Ra | Opcode | Predicate | |
| Immed8 | Rb8 | Ra8 | 6Dh8 | Pn4 | Pc4 |

#### Operation:

Rt = Ra \* 8 + immediate

#### Description:

Multiply Ra by eight and add immediate and place the sum in the target register. This instruction will never cause an overflow exception.

### 16ADDU - Register-Register

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 39 32 | 31 24 | 23 16 | 15 8 | 7 0 | |
| Rt | Rb | Ra | Opcode | Predicate | |
| Rt8 | Rb8 | Ra8 | 65h8 | Pn4 | Pc4 |

#### Operation:

Rt = Ra \* 16 + Rb

#### Description:

Multiply Ra by sixteen and add Rb and place the sum in the target register. This instruction will never cause an overflow exception.

### 16ADDUI - Register-Register

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 39 32 | 31 24 | 23 16 | 15 8 | 7 0 | |
| Immed | Rb | Ra | Opcode | Predicate | |
| Immed8 | Rb8 | Ra8 | 6Eh8 | Pn4 | Pc4 |

#### Operation:

Rt = Ra \* 16 + immediate

#### Description:

Multiply Ra by sixteen and add immediate and place the sum in the target register. This instruction will never cause an overflow exception.

### ADD - Register-Register

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 39 32 | 31 24 | 23 16 | 15 8 | 7 0 | |
| Rt | Rb | Ra | Opcode | Predicate | |
| Rt8 | Rb8 | Ra8 | 40h8 | Pn4 | Pc4 |

#### Operation:

Rt = Ra + Rb

#### Description:

Add two registers and place the sum in the target register. This instruction may cause an overflow exception.

### ADDI - Register-Immediate

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 39 32 | 31 24 | 23 16 | 15 8 | 7 0 | |
| Immediate | Rt | Ra | Opcode | Predicate | |
| Immediate7..0 | Rt8 | Ra8 | 48h8 | Pn4 | Pc4 |

#### Operation:

Rt = Ra + immediate

#### Description:

Add a register and immediate value and place the sum in the target register. This instruction may cause an overflow exception.

### ADDU - Register-Register

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 39 32 | 31 24 | 23 16 | 15 8 | 7 0 | |
| Rt | Rb | Ra | Opcode | Predicate | |
| Rt8 | Rb8 | Ra8 | 44h8 | Pn4 | Pc4 |

#### Operation:

Rt = Ra + Rb

#### Description:

This instruction will never cause any exceptions.

### ADDUI - Register-Immediate

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 39 32 | 31 24 | 23 16 | 15 8 | 7 0 | |
| Immediate | Rt | Ra | Opcode | Predicate | |
| Immediate7..0 | Rt8 | Ra8 | 4Ch8 | Pn4 | Pc4 |

### AND - Register-Register

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 39 32 | 31 24 | 23 16 | 15 8 | 7 0 | |
| Rt | Rb | Ra | Opcode | Predicate | |
| Rt8 | Rb8 | Ra8 | 50h8 | Pn4 | Pc4 |

#### Operation:

Rt = Ra & Rb

#### Description:

Locgially and’s two registers and places the result in a target register.

### ANDI - Register-Immediate

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 39 32 | 31 24 | 23 16 | 15 8 | 7 0 | |
| Immediate | Rt | Ra | Opcode | Predicate | |
| Immediate7..0 | Rt8 | Ra8 | 53h8 | Pn4 | Pc4 |

### BR - Relative Branch

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 23 16 | 15 8 | | 7 0 | |
| Disp. | Op | Disp | Predicate | |
| Disp7..0 | 3h4 | D11..8 | Pn4 | Pc4 |

#### Operation:

pc <= pc + displacement

#### Description:

A branch is made relative to the address of the branch instruction.

* The displacement field cannot be extended with an immediate constant prefix. Branches are executed immediately in the ifetch stage of the processor before it is known if there is a prefix present.

### BRK –Break

|  |  |
| --- | --- |
| 7 0 | |
| Predicate | |
| 04 | 04 |

#### Operation:

<none>

#### Description:

This instruction contains only a predicate byte. The Break exception is executed.

### BSR - Branch To Subroutine

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Disp. | Br | Brt | Opcode | Predicate | |
| Offset7..0 | 154 | Brt4 | A2h8 | Pn4 | Pc4 |

#### Description:

This is an alternate mnemonic for the JSR instruction. Normally the BSR instruction is used with an immediate predicate constant in order to extend the address range of the jump. A jump is made to the sum of the sign extended offset supplied in the offset field of the instruction and the specified branch register Br.

The subroutine return address is stored in a branch register specified in the Brt field of the instruction.

### CLI – Clear Interrupt Mask

|  |  |  |
| --- | --- | --- |
| 15 8 | 7 0 | |
| Opcode | Predicate | |
| F0h8 | Pn4 | Pc4 |

#### Operation:

im = 0

#### Description:

This instruction is used to enable interrupts.

### CMP Register-Register Compare

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 31 24 | 23 16 | 15 12 | 11 8 | 7 0 | |
| Rb | Ra | Opc | Pt | Predicate | |
| Rb8 | Ra8 | 14 | Pt4 | Pn4 | Pc4 |

### CMPI Register-Immediate Compare

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 31 24 | 23 16 | 15 12 | 11 8 | 7 0 | |
| Immed | Ra | Opc | Pt | Predicate | |
| Immed8 | Ra8 | 24 | Pt4 | Pn4 | Pc4 |

#### Operation:

if signed Ra < signed immediate

P.lt = true

else

P.lt = false

if unsigned Ra < unsigned immediate

P.ltu = true

else

P.ltu = false

if Ra = immediate

P.eq = true

else

P.eq = false

#### Description:

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

### EOR - Register-Register

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 39 32 | 31 24 | 23 16 | 15 8 | 7 0 | |
| Rt | Rb | Ra | Opcode | Predicate | |
| Rt8 | Rb8 | Ra8 | 52h8 | Pn4 | Pc4 |

#### Operation:

Rt = Ra ^ Rb

#### Description:

Logically exclusive or register with register and place result in target register.

### EORI - Register-Immediate

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 39 32 | 31 24 | 23 16 | 15 8 | 7 0 | |
| Immediate | Rt | Ra | Opcode | Predicate | |
| Immediate7..0 | Rt8 | Ra8 | 55h8 | Pn4 | Pc4 |

#### Operation:

Rt = Ra ^ immediate

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

### Immediate Extensions

The immediate extension predicates are used to extend the immediate constant of the following instruction. The extensions may add from one to seven bytes more to the constant. Most, but not all instructions can accept a predicated immediate.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Immediate | | | | | | | Predicate | |
| Immediate63..8 | | | | | | | 84 | 04 |
| Immediate55..8 | | | | | | 74 | 04 |
| Immediate47..8 | | | | | 64 | 04 |
| Immediate39..8 | | | | 54 | 04 |
| Immediate31..8 | | | 44 | 04 |
| Immediate23..8 | | 34 | 04 |
| Immediate15..8 | 24 | 04 |

### INT –Interrupt

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Disp. | Br | Brt | Opcode | Predicate | |
| Offset7..0 | Ch4 | Eh4 | A6h8 | Pn4 | Pc4 |

#### Description:

This instruction calls a system function located as the sum of the offset times 16 plus branch register 12. The return address is stored in the IPC register (Branch register #14).

The offset field of this instruction cannot be extended.

Note that this instruction is automatically invoked for hardware interrupt processing. This instruction would not normally be used by software. The return address stored is the address of the interrupt instruction, not the address of the next instruction. To call system routines use the SYS instruction.

### JMP - Jump To Address

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Disp. | Br | Brt | Opcode | Predicate | |
| Offset7..0 | Br4 | 04 | A2h8 | Pn4 | Pc4 |

#### Operation:

pc = Br[n] + offset

#### Description:

This is an alternate mnemonic for the JSR instruction.

Normally the JMP instruction is used with an immediate predicate constant in order to extend the address range of the jump. A jump is made to the sum of the sign extended offset supplied in the offset field of the instruction and the specified branch register Br.

### JSR - Jump To Subroutine Instruction

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Disp. | Br | Brt | Opcode | Predicate | |
| Offset7..0 | Br4 | Brt4 | A2h8 | Pn4 | Pc4 |

#### Operation:

Br[t] = pc

pc = Br[n] + offset

#### Description:

Normally the JSR instruction is used with an immediate predicate constant in order to extend the address range of the jump. A jump is made to the sum of the sign extended offset supplied in the offset field of the instruction and the specified branch register Br.

The subroutine return address is stored in a branch register specified in the Brt field of the instruction. Typically branch register #1 is used.

### LB – Load Byte

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 39 32 | 31 24 | 23 16 | 15 8 | 7 0 | |
| Offset | Rt | Ra | Opcode | Predicate | |
| Offset7..0 | Rt8 | Ra8 | 80h8 | Pn4 | Pc4 |

#### Operation:

Rt = sign extend (mem[Ra])

#### Description:

A eight bit value is loaded from memory and sign extended, then placed in the target register. The memory address is the sum of the offset and register Ra.

### LBU – Load Byte Unsigned

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 39 32 | 31 24 | 23 16 | 15 8 | 7 0 | |
| Offset | Rt | Ra | Opcode | Predicate | |
| Offset7..0 | Rt8 | Ra8 | 81h8 | Pn4 | Pc4 |

#### Operation:

Rt = zero extend (mem[Ra])

#### Description:

A eight bit value is loaded from memory and zero extended, then placed in the target register. The memory address is the sum of the offset and register Ra.

### LC – Load Character

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 39 32 | 31 24 | 23 16 | 15 8 | 7 0 | |
| Offset | Rt | Ra | Opcode | Predicate | |
| Offset7..0 | Rt8 | Ra8 | 82h8 | Pn4 | Pc4 |

#### Operation:

Rt = sign extend (mem[Ra])

#### Description:

A sixteen bit value is loaded from memory and sign extended, then placed in the target register. The memory address is the sum of the offset and register Ra. The memory address must be character aligned.

### LCU – Load Character Unsigned

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 39 32 | 31 24 | 23 16 | 15 8 | 7 0 | |
| Offset | Rt | Ra | Opcode | Predicate | |
| Offset7..0 | Rt8 | Ra8 | 83h8 | Pn4 | Pc4 |

#### Operation:

Rt = zero extend (mem[Ra])

#### Description:

A sixteen bit value is loaded from memory and zero extended, then placed in the target register. The memory address is the sum of the offset and register Ra. The memory address must be character aligned.

### LDI - Load-Immediate

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 31 24 | 23 16 | 15 8 | 7 0 | |
| Immediate | Rt | Opcode | Predicate | |
| Immediate7..0 | Rt8 | 6Fh8 | Pn4 | Pc4 |

#### Operation:

Rt = immediate

#### Description:

This instruction loads an immediate constant into a register. The immediate constant may be extended by using an immediate prefix instruction.

### LH – Load Half-Word

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 39 32 | 31 24 | 23 16 | 15 8 | 7 0 | |
| Offset | Rt | Ra | Opcode | Predicate | |
| Offset7..0 | Rt8 | Ra8 | 84h8 | Pn4 | Pc4 |

#### Operation:

Rt = sign extend (mem[Ra])

#### Description:

A thirty-two bit value is loaded from memory and sign extended, then placed in the target register. The memory address is the sum of the offset and register Ra. The memory address must be half-word aligned.

### LHU – Load Half-word Unsigned

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 39 32 | 31 24 | 23 16 | 15 8 | 7 0 | |
| Offset | Rt | Ra | Opcode | Predicate | |
| Offset7..0 | Rt8 | Ra8 | 85h8 | Pn4 | Pc4 |

#### Operation:

Rt = zero extend (mem[Ra])

#### Description:

A thirty-two bit value is loaded from memory and zero extended, then placed in the target register. The memory address is the sum of the offset and register Ra. The memory address must be half-word aligned.

### LW – Load Word

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 39 32 | 31 24 | 23 16 | 15 8 | 7 0 | |
| Offset | Rt | Ra | Opcode | Predicate | |
| Offset7..0 | Rt8 | Ra8 | 86h8 | Pn4 | Pc4 |

#### Operation:

Rt = mem[Ra]

#### Description:

A sixty-four bit value is loaded from memory and placed in the target register. The memory address is the sum of the offset and register Ra. The memory address must be word aligned.

### LOOP – Loop Branch

|  |  |  |  |
| --- | --- | --- | --- |
| 23 16 | 15 8 | 7 0 | |
| Disp. | Opcode | Predicate | |
| Disp7..0 | A4h8 | Pn4 | Pc4 |

#### Operation:

If lc <> 0

pc <= pc + displacement

lc = lc - 1

#### Description:

A branch is made relative to the current value of the program counter if the loop count register is non-zero. The loop count register is decremented by this instruction.

### MFSPR – Special Register-Register

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 31 24 | 23 16 | 15 8 | 7 0 | |
| Rt | Spr | Opcode | Predicate | |
| Rt8 | Spr8 | A8h8 | Pn4 | Pc4 |

#### Operation:

Rt = Spr[n]

#### Special Purpose Registers

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Reg # | R/W |  |  |  |
| 0 | R | MID | Machine ID |  |
| 1 | R | FEAT | Features |  |
| 2 | R | TICK | Tick count |  |
| 3 | RW | LC | Loop Counter |  |
| 4 | RW | PREGS | Predicate register array |  |
|  |  |  |  |  |

### MTSPR –Register-Special Register

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 31 24 | 23 16 | 15 8 | 7 0 | |
| Spr | Ra | Opcode | Predicate | |
| Spr8 | Ra8 | A9h8 | Pn4 | Pc4 |

#### Operation:

Spr[n] = Ra

### NEG - Negate Register

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 31 24 | 23 16 | 15 8 | 7 0 | |
| Rt | Ra | Opcode | Predicate | |
| Rt8 | Ra8 | 70h8 | Pn4 | Pc4 |

#### Operation:

Rt = - Ra

#### Description:

This instruction negates a register and places the result in a target register.

### NOP – No Operation

|  |  |
| --- | --- |
| 7 0 | |
| Predicate | |
| 14 | 04 |

#### Operation:

<none>

#### Description:

This instruction contains only a predicate byte.

### OR - Register-Register

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 39 32 | 31 24 | 23 16 | 15 8 | 7 0 | |
| Rt | Rb | Ra | Opcode | Predicate | |
| Rt8 | Rb8 | Ra8 | 51h8 | Pn4 | Pc4 |

#### Operation:

Rt = Ra | Rb

#### Description:

Logically inclusively or two registers and place the result in the target register.

### ORI - Register-Immediate

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 39 32 | 31 24 | 23 16 | 15 8 | 7 0 | |
| Immediate | Rt | Ra | Opcode | Predicate | |
| Immediate7..0 | Rt8 | Ra8 | 54h8 | Pn4 | Pc4 |

#### Operation:

Rt = Ra | imm

### RTE – Return From Exception Routine

|  |  |  |
| --- | --- | --- |
| Opcode | Predicate | |
| F3h8 | Pn4 | Pc4 |

#### Operation**:**

pc = Br[13]

Flags = FlagsBackup

#### **Description**:

The program counter is loaded with the value contained in branch register #13 which is the EPC register.

### RTI – Return From Interrupt Routine

|  |  |  |
| --- | --- | --- |
| Opcode | Predicate | |
| F4h8 | Pn4 | Pc4 |

#### Operation:

pc = Br[14]

Flags = FlagsBackup

Flags.im = 0

#### **Description**:

The program counter is loaded with the value contained in branch register #14 which is the IPC register.

### RTS – Return From Subroutine

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Br | ~ | Opcode | Predicate | |
| Br4 | 04 | A3h8 | Pn4 | Pc4 |

#### Operation:

pc = Br[N]

#### **Description**:

The program counter is loaded with the value contained in the specified branch register.

#### Short Form:

|  |  |
| --- | --- |
| Predicate | |
| 14 | 14 |

#### Operation:

pc = Br[1]

#### **Description**:

The program counter is loaded with the value contained in branch register #1.

### SB – Store Byte

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 39 32 | 31 24 | 23 16 | 15 8 | 7 0 | |
| Offset | Rt | Ra | Opcode | Predicate | |
| Offset7..0 | Rt8 | Ra8 | 90h8 | Pn4 | Pc4 |

#### Operation:

memory[Ra] = Rb[7..0]

#### Description:

An eight bit value is stored to memory from the source register Rb. The memory address is the sum of the offset and register Ra.

### SC – Store Character

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 39 32 | 31 24 | 23 16 | 15 8 | 7 0 | |
| Offset | Rt | Ra | Opcode | Predicate | |
| Offset7..0 | Rt8 | Ra8 | 91h8 | Pn4 | Pc4 |

#### Operation:

memory[Ra] = Rb[15..0]

#### Description:

A sixteen bit value is stored to memory from the source register Rb. The memory address is the sum of the offset and register Ra. The memory address must be character aligned.

### SEI – Set Interrupt Mask

|  |  |  |
| --- | --- | --- |
| 15 8 | 7 0 | |
| Opcode | Predicate | |
| F1h8 | Pn4 | Pc4 |

#### Operation:

im = 1

#### Description:

The interrupt mask is set, disabling maskable interrupts.

### SH – Store Half-word

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 39 32 | 31 24 | 23 16 | 15 8 | 7 0 | |
| Offset | Rt | Ra | Opcode | Predicate | |
| Offset7..0 | Rt8 | Ra8 | 92h8 | Pn4 | Pc4 |

#### Operation:

memory[Ra] = Rb[31..0]

#### Description:

A thirty-two bit value is stored to memory from the source register Rb. The memory address is the sum of the offset and register Ra. The memory address must be half-word aligned.

### STSB – Store String Byte

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 31 24 | 23 16 | 15 8 | 7 0 | |
| Rb | Ra | Opcode | Predicate | |
| Rb8 | Ra8 | 9Ah8 | Pn4 | Pc4 |

#### Operation:

If LC <> 0

mem[Ra] = Rb[7:0]

Ra = Ra + 1

LC = LC – 1

### STSW – Store String Word

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 31 24 | 23 16 | 15 8 | 7 0 | |
| Rb | Ra | Opcode | Predicate | |
| Rb8 | Ra8 | 9Ah8 | Pn4 | Pc4 |

#### Operation:

If LC <> 0

mem[Ra] = Rb

Ra = Ra + 8

LC = LC – 1

### SW – Store Word

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 39 32 | 31 24 | 23 16 | 15 8 | 7 0 | |
| Offset | Rt | Ra | Opcode | Predicate | |
| Offset7..0 | Rt8 | Ra8 | 93h8 | Pn4 | Pc4 |

#### Operation:

memory[Ra] = Rb

#### Description:

A sixty-four bit value is stored to memory from the source register Rb. The memory address is the sum of the offset and register Ra. The memory address must be word aligned.

### SUB - Register-Register

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 39 32 | 31 24 | 23 16 | 15 8 | 7 0 | |
| Rt | Rb | Ra | Opcode | Predicate | |
| Rt8 | Rb8 | Ra8 | 418 | Pn4 | Pc4 |

#### Operation:

Rt = Ra - Rb

#### Description:

This instruction subtracts one register from another and places the result into a third register. This instruction may cause an overflow exception.

### SUBI - Register-Immediate

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 39 32 | 31 24 | 23 16 | 15 8 | 7 0 | |
| Immediate | Rt | Ra | Opcode | Predicate | |
| Immediate7..0 | Rt8 | Ra8 | 49h8 | Pn4 | Pc4 |

#### Operation:

Rt = Ra – Imm

#### Description:

This instruction subtracts an immediate value from a register and places the result into a register. This instruction may cause an overflow exception.

### SUBU - Register-Register

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 39 32 | 31 24 | 23 16 | 15 8 | 7 0 | |
| Rt | Rb | Ra | Opcode | Predicate | |
| Rt8 | Rb8 | Ra8 | 458 | Pn4 | Pc4 |

#### Operation:

Rt = Ra - Rb

#### Description:

This instruction subtracts one register from another and places the result into a third register.

### SUBUI - Register-Immediate

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 39 32 | 31 24 | 23 16 | 15 8 | 7 0 | |
| Immediate | Rt | Ra | Opcode | Predicate | |
| Immediate7..0 | Rt8 | Ra8 | 4Dh8 | Pn4 | Pc4 |

#### Operation:

Rt = Ra – Imm

#### Description:

This instruction subtracts an immediate value from a register and places the result into a register.

### SYNC – Synchronize Memory

|  |  |  |
| --- | --- | --- |
| 15 8 | 7 0 | |
| Opcode | Predicate | |
| F8h8 | Pn4 | Pc4 |

#### Operation:

im = 1

#### Description:

All memory accesses before the SYNC command are completed before execution continues.

### SYS –Call system routine

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Disp. | Br | Brt | Opcode | Predicate | |
| Offset7..0 | Ch4 | Dh4 | A5h8 | Pn4 | Pc4 |

#### Description:

This instruction calls a system function located as the sum of the offset times 16 plus branch register 12. The return address is stored in the EPC register (Branch register #13).

### TST - Register Test Compare

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 23 16 | 15 12 | 11 8 | 7 0 | |
| Ra | Opc | Pt | Predicate | |
| Ra8 | 04 | Pt4 | Pn4 | Pc4 |

#### Operation:

if Ra < 0

Pt.lt = 1

else

Pt.lt = 0

if Ra = 0

Pt.eq = 1

else

Pt.eq = 0

Pt.ltu = 0

#### Description:

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