Table of Contents

[Overview 1](#_Toc36010820)

[Programming Model 1](#_Toc36010821)

[General Registers 1](#_Toc36010822)

[Code Address Registers 2](#_Toc36010823)

[Predicates 3](#_Toc36010824)

[Predicate Conditions 3](#_Toc36010825)

[Compiler Usage 4](#_Toc36010826)

[Instructions 5](#_Toc36010827)

[Instruction Pointer 5](#_Toc36010828)

[Instruction Set Description 7](#_Toc36010829)

[MOVI – Move Immediate 7](#_Toc36010830)

# Overview

Thor2020 is a powerful 64-bit superscalar processor that represents a generational refinement of processor architecture. The processor contains 64, 64-bit general-purpose integer registers. Thor2020 uses fixed length instructions three packed into a 128-bit bundle and handles 8, 16, 32, and 64-bit data within a 64-bit address space.

## Programming Model

### General Registers

There are 64 general-purpose registers. General purpose registers are 64 bits wide. The general registers may hold integer or floating-point values.

Register #0 is always zero or +0.0.

A screenshot of a cell phone

Description automatically generated

### Code Address Registers

The processor contains eight code address registers (C0-C7). 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 | Catch Link Register | Used by the compiler to link to try/catch handlers. |
| 6 | Exceptioned PC | This register is automatically set during a hardware interrupt or exception. |
| 7 | Program Counter | Relative address formation. This register is read-only. |

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

The presence of multiple code address registers allows multi-level return addresses to be used for performance. Leaf routines may use C1 as the return address. Next to leaf routines may use C2, etc. So that memory operations are avoided when implementing subroutine call and return.

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

## Predicates

The processor features predicated execution of all instructions. Whether or not an instruction is executed depends on the contents of a predicate register and the predicate condition specified in the predicate byte. There are 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).

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

All instructions are executed conditionally determined by the value of a predicate register.

### 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. The false predicate byte for instructions is 90h. |
| 1 | PT | 1 | Always True – The instruction predicated with an always true condition always executes regardless of the predicate register contents. The always true predicate byte is 01h. Other true predicates are instruction short-forms. |
| 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  POR | !ltu | unsigned greater or equal  Ordered for floating point |
| 11 | PLTU  PUN | ltu | unsigned less than  Unordered for floating point |
| 12 |  |  |  |
| 13 |  |  |  |
| 14 |  |  |  |
| 15 |  |  |  |

### Compiler Usage

The compiler uses predicate register #15 to conditionally move TRUE / FALSE values to a register when evaluating a logical operation.

Predicate registers beginning with P0 and incrementing are applied for use as the control flow nesting level increases. The compiler does not support control flow nesting more than 14 levels in a single subroutine. Predicate registers beginning with P14 and decrementing are used in the evaluation of the hook operator. Care must be taken such that the number of predicate registers in use does not exceed the number available.

|  |  |  |
| --- | --- | --- |
| Pred. | Usage |  |
| P0 | control flow level 0 |  |
| P1 | control flow nesting level 1 |  |
| P2 | control flow nesting level 2 |  |
| … |  |  |
| Pn | control flow nesting level n (n not to exceed 14) |  |
| … |  |  |
| P12 | third hook operator in an expression |  |
| P13 | second hook operator in an expression |  |
| P14 | first hook operator in an expression |  |
| P15 | conditionally moves TRUE/FALSE for logical expressions |  |

## Instructions

Thor2020 uses instruction bundles. Three 41-bit instructions are placed in a 128-bit bundle. The remaining bits of the bundle are for control. Instructions are effectively executed in the order slot0, slot1, then slot2. Multiple instructions may be executed in the same clock cycle. Stop bits in the control part of the bundle indicate when multiple cycles are required to resolve dependencies.

|  |  |  |  |
| --- | --- | --- | --- |
| 127 123 | 122 82 | 81 41 | 40 0 |
| Control | Slot2 | Slot1 | Slot0 |

|  |  |
| --- | --- |
| Control | Description |
| 123 | stop after instruction zero |
| 124 | stop after instruction one |
| 125 | stop after instruction two |
| 126 | reserved |
| 127 | reserved |

## Instruction Pointer

The instruction pointer is always bundle (128-bit) aligned. Jumps and branches target a 128-bit aligned address. The first slot of a bundle is always targeted. Since branch targets must be bundle aligned it may be necessary for the assembler / compiler to output NOP instructions.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | |  |  |  | | | | Opcode8 | | Pred4 | Pc4 |  |
| Immediate14 | | | | Ra6 | Rt6 | | | | 47 | | Pr4 | Pc4 | ADD # |
| Immediate14 | | | | Ra6 | Rt6 | | | | 87 | | Pr4 | Pc4 | AND # |
| Immediate14 | | | | Ra6 | Rt6 | | | | 97 | | Pr4 | Pc4 | OR # |
| Immediate14 | | | | Ra6 | Rt6 | | | | 107 | | Pr4 | Pc4 | EOR # |
| Disp13..0 | | | | Ra6 | Rt6 | | | | 53 | Func4 | Pr4 | Pc4 | LDx |
| Disp13..6 | | | Rb6 | Ra6 | Disp5..0 | | | | 63 | Func4 | Pr4 | Pc4 | STx |
| Immediate14 | | | | Ra6 | ~2 | | Pt4 | | 67 | | Pr4 | Pc4 | CMP # |
| Immediate14 | | | | Ra6 | ~2 | | Pt4 | | 77 | | Pr4 | Pc4 | BIT # |
|  | | | |  |  | | | |  | |  |  |  |
|  |  | |  |  |  | | | |  | |  |  |  |
| 47 | | ~ | Rb6 | Ra6 | Rt6 | | | | 27 | | Pr4 | Pc4 | ADD |
| 57 | | ~ | Rb6 | Ra6 | Rt6 | | | | 27 | | Pr4 | Pc4 | SUB |
| 87 | | ~ | Rb6 | Ra6 | Rt6 | | | | 27 | | Pr4 | Pc4 | AND |
| 97 | | ~ | Rb6 | Ra6 | Rt6 | | | | 27 | | Pr4 | Pc4 | OR |
| 107 | | ~ | Rb6 | Ra6 | Rt6 | | | | 27 | | Pr4 | Pc4 | EOR |
| 117 | |  | Rb6 | Ra6 | Rt6 | | | | 27 | | Pr4 | Pc4 | ANDCM |
| 127 | |  | Rb6 | Ra6 | Rt6 | | | | 27 | | Pr4 | Pc4 | NAND |
| 137 | |  | Rb6 | Ra6 | Rt6 | | | | 27 | | Pr4 | Pc4 | NOR |
| 147 | |  | Rb6 | Ra6 | Rt6 | | | | 27 | | Pr4 | Pc4 | ENOR |
| 63 | | ~ | Rb6 | Ra6 | ~2 | | Pt4 | | 27 | | Pr4 | Pc4 | CMP |
| Immediate22..3 | | | | | Rt6 | | | | 84 | I2..0 | Pr4 | Pc4 | MOVI |
| Address23..4 | | | | | | Ca3 | | Ct3 | 787 | | Pr4 | Pc4 | JMP |
| Address23..4 | | | | | | Ca3 | | Ct3 | 797 | | Pr4 | Pc4 | JML |

# Instruction Set Description

## Control Flow Instructions

### JML - Jump to Long Address

**Description:**

A jump is made to the sum of the zero extended offset supplied in the offset field of the instruction and the specified code address register Ca.

A subroutine return address may be stored in a code address register specified in the Ct field of the instruction. Typically, code address register #1 is used.

This instruction is unusual in that it occupies the first two slots of a bundle. A full 64-bit address may be formed.

**Instruction Format**:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 127 123 | 122 82 | 81 41 | 40 0 | |
| Control | Slot2 | Address63..24 | Address23..4 | Other |

**Clock cycles**: 0.66 (two slots are occupied).

**Execution Unit:** FCU

**Operation:**

C[t] = ip + 16

ip = C[n] + offset

**Exceptions**: none

### JMP - Jump to Address

**Description:**

A jump is made to the sum of the zero extended offset supplied in the offset field of the instruction and the specified code address register Ca.

A subroutine return address may be stored in a code address register specified in the Ct field of the instruction. Typically, code address register #1 is used.

**Instruction Formats: JMP**

**Clock Cycles:** 0.33

**Execution Units:** All ALU’s

**Operation:**

Cr[t] = pc

pc = Cr[n] + offset

**Exceptions**: none

## Arithmetic / Logical

### ADD - Register-Register

**Description:**

Add two registers or a register and an immediate value and place the sum in the target register. The register form of the instruction may cause an overflow exception is enabled.

**Instruction Format: R2, RI**

**Clock Cycles:** 0.33

**Execution Unit:** AllALU’s

**Operation:**

Rt = Ra + Rb

**Exceptions**: integer overflow

### AND - Register-Register

**Description:**

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

**Instruction Format: R2**

**Clock Cycles:** 0.33

**Execution Unit:** AllALU’s

**Operation:**

Rt = Ra & Rb

**Exceptions:** none

## MOVI – Move Immediate

Description:

The immediate move instruction is unusual in that it occupies two slots of the bundle as shown below. The movi instruction may be placed only in the first two instruction slots.

Instruction Format:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 127 123 | 122 82 | 81 41 | 40 0 | |
| Control | Slot2 | Immediate63..23 | Immed22..3 | Other |

Clock cycles: 0.66 (two slots are occupied).