Black Widow

(c) 2022 Robert Finch

## Introduction

The BlackWidow ISA and core is the author’s attempt at a VLIW processing core. The issue width is at least three instructions wide.

# Programming Model

## Datapath

The data-path is 128 bits wide.

*The primary motivation for an 128-bit data path is to support 128-bit decimal floating-point.*

## Instructions

Instructions are a fixed 40-bits in size. More detail on instructions is given in the instruction set description section below.

## General Purpose Registers – GPRs

The ISA provides for 64 general purpose registers. R0 always contains the value zero. The GPRs store either integer or floating-point data. This is referred to as a *unified* register file.

## Predicate Registers

The ISA has 64 predicate registers P0 to P63. A predicate is specified for every instruction and the instruction will execute only if the predicate is true. Predicate zero is always false. Predicate one is always true.

## Floating-Point Format

The floating-point format is an 80-bit extended precision decimal format with the following layout:

|  |  |  |
| --- | --- | --- |
| 79 | 78 70 | 69 0 |
| Sign  1 bit | Exponent  9 bits | Significand  70 bits |

The sign bit is zero for negative, or one for positive numbers

The exponent is a power of ten exponent represented as a binary number, the exponent range is -255 to +256. The exponent bias value is 255.

The significand is a group of seven, ten-bit densely-packed-decimal values. This provides 21 significant digits. There are no hidden bits. There is one whole digit before the decimal point.

Infinity is represented as an exponent of all ones and a zero significand.

Nans are represented as an exponent of all ones and a non-zero significand.

# Instruction Set Description

## Instruction Formats

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  | | |  |  |  |  |  |
| B | Opcode6 | Func6 | | ~3 | | Rb6 | Ra6 | Rt6 | Pr6 |  |
| B | Opcode6 | Immediate15 | | | | | Ra6 | Rt6 | Pr6 |  |
| B | Opcode6 | Op3 | Immediate12 | | | | Ra6 | Rt6 | Pr6 | SETI / SETUI |
| B | Opcode6 | Op3 | Pt26 | | Pt16 | | Rb6 | Ra6 | Pr6 | CMP |
| B | Opcode6 | Op3 | Pt26 | | Pt16 | | Imm6 | Ra6 | Pr6 | CMPI |
| B | Opcode6 | Displacement27 | | | | | | | Pr6 | BRA / BSR |
| B | Opcode6 | Immediate27 | | | | | | | Pr6 | CON1/2/3 |

## Opcode Maps

### Major Opcode6

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0x | 0  SYS | 1  MOD | 2  {R2} | 3  {FLT} | 4  ADDI | 5  SUBFI | 6 | 7 |
| 8  ANDI | 9  ORI | 10  XORI | 11 | 12  MULI | 13  BRA | 14  BSR | 15  BMR |
| 1x | 16  CMP | 17  CMPU | 18  FCMP | 19  CMP | 20  CMPI | 21  CMPUI | 22  FCMPI | 23  CMPI |
|  | 24  FDP | 25  FFDP | 26 | 27 | 28  SETI | 29  SETUI | 30  FSETI | 31 |
| 2x | 32  LDB | 33  LDBU | 34  LDW | 35  LDWU | 36  LDT | 37  LDTU | 38  LDO | 39  LDOU |
|  | 40 | 41 | 42  LDH | 43  LDHR | 44 | 45 | 46 | 47 |
| 3x | 48  STB | 49  STW | 50  STT | 51  STO | 52 | 53  STH | 54  STHC | 55 |
|  | 56 | 57 | 58  CON4 | 59  CON5 | 60  NOP | 61  CON1 | 62  CON2 | 63  CON3 |

### Opcode 2, Major Func6

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0x | 0  SLL | 1  SRL | 2  SRA | 3 | 4  ADD | 5  SUB | 6 | 7 |
| 8  AND | 9  OR | 10  XOR | 11 | 12  MUL | 13  JMP | 14 | 15 |
| 1x | 16  NAND | 17  NOR | 18  XNOR | 19 | 20 | 21 | 22 | 23 |
|  | 24 | 25 | 26 | 27 | 28  SET | 29  SETU | 30  FSET | 31 |
| 2x | 32  LDBX | 33  LDBUX | 34  LDWX | 35  LDWUX | 36  LDTX | 37  LDTUX | 38  LDOX | 39  LDOUX |
|  | 40 | 41 | 42  LDHX | 43  LDHRX | 44 | 45 | 46 | 47  LDCHK |
| 3x | 48  STBX | 49  STWX | 50  STTX | 51  STOX | 52 | 53  STHX | 54  STHCX | 55 |
|  | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63  REG |

## ALU Operations

### ADD – Addition

**Description:**

Add two source operands and place the result in the target register. The immediate constant is sign extended to the machine width. The operands are treated as signed two-complement numbers.

**Instruction Formats:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 39 | 38 33 | 32 27 | 26 24 | 23 18 | 17 12 | 11 6 | 5 0 |
| B | 26 | 46 | ~3 | Rb6 | Ra6 | Rt6 | Pred6 |
| B | 46 | Imm15 | | | Ra6 | Rt6 | Pred6 |

**Exceptions:** none

**Execution Units:** All ALUs

### AND – Bitwise And

**Description:**

Bitwise and two source operands and place the result in the target register. The immediate constant is sign extended to the machine width.

**Instruction Formats:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 39 | 38 33 | 32 27 | 26 24 | 23 18 | 17 12 | 11 6 | 5 0 |
| B | 26 | 86 | ~3 | Rb6 | Ra6 | Rt6 | Pred6 |
| B | 86 | Imm15 | | | Ra6 | Rt6 | Pred6 |

**Exceptions:** none

**Execution Units:** All ALUs

### CMP – Compare

**Description:**

Compare two source operands for a specified relationship. Set Pt1 to true and Pt2 to false if the relationship is true. Otherwise set Pt1 false and Pt2 true.

**Instruction Formats:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 39 | 38 33 | 32 30 | 29 24 | 23 18 | 17 12 | 11 6 | 5 0 |
| B | Opcode6 | Op3 | Pt26 | Pt16 | Rb6 | Ra6 | Pr6 |
| B | Opcode6 | Op3 | Pt26 | Pt16 | Imm6 | Ra6 | Pr6 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Op3 |  |  | Opcode6 |  |
| 0 | LT |  | 16 | CMP |
| 1 | GE |  | 17 | CMPU |
| 2 | LE |  | 18 | FCMP |
| 3 | GT |  |  |  |
| 4 | EQ |  | 20 | CMPI |
| 5 | NE |  | 21 | CMPUI |
| 6 | BC |  | 22 | FCMPI |
| 7 | BS |  |  |  |

**Exceptions:** none

**Execution Units:** All ALUs

### CSR – Control and Special Register Access

**Description:**

Read or write CSR registers.

**Instruction Formats:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 39 35 | 34 30 | 2928 | 2726 | 25 16 | 15 11 | 10 6 | 5 0 |
| 25 | 155 | Op2 | M2 | CSR10 | Ra5 | Rt5 | Pred6 |

|  |  |
| --- | --- |
| Op2 | Operation |
| 0 | Read CSR |
| 1 | Write CSR |
| 2 | Set CSR bits |
| 3 | Clear CSR bits |

### MUL – Multiply

**Description:**

Multiply two source operands and place the result in the target register. The immediate constant is sign extended to the machine width. The operands are treated as signed two-complement numbers. If the REG prefix is used then the high order product bits are available in the specified target register of the REG prefix.

**Instruction Formats:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 39 | 38 33 | 32 27 | 26 24 | 23 18 | 17 12 | 11 6 | 5 0 |
| B | 26 | 126 | ~3 | Rb6 | Ra6 | Rt6 | Pred6 |
| B | 126 | Imm15 | | | Ra6 | Rt6 | Pred6 |

|  |  |
| --- | --- |
| Op3 | Operation Performed |
| 0 | signed multiply |
| 1 | unsigned multiply |
| 2 | signed by unsigned multiply |
| 3 |  |
| 4 | (a \* b) + (c \* d) signed dot product, requires reg prefix |
| 5 | -((a \*b) + (c \* d)) negate signed dot product |
| 6 | (a \* b) – (c \* d) |
| 7 | -((a \* b) – (c \* d)) |

**Exceptions:** none

**Execution Units:** ALU #0 Only

### OR – Bitwise Or

**Description:**

Bitwise or two source operands and place the result in the target register. The immediate constant is sign extended to the machine width.

**Instruction Formats:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 39 | 38 33 | 32 27 | 26 24 | 23 18 | 17 12 | 11 6 | 5 0 |
| B | 26 | 96 | ~3 | Rb6 | Ra6 | Rt6 | Pred6 |
| B | 96 | Imm15 | | | Ra6 | Rt6 | Pred6 |

**Exceptions:** none

**Execution Units:** All ALUs

### SLL – Shift Left Logical

**Description:**

Shift the value in Ra to the left by the number of bits specified in the second source operand. Zeros are shifted into the least significant bits.

If the extended precision ‘e’ bit is set in the instruction then bits from the Ra register of the following REG instruction will be shifted into the low order bits. Bits shifted out of the high order bits will be placed in the Rt register of the following REG instruction.

**Instruction Formats:**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 39 | 38 33 | 32 27 | 26 | 25 | 24 | 23 18 | 17 12 | 11 6 | 5 0 |
| B | 26 | 06 | 0 | e | ~ | Rb6 | Ra6 | Rt6 | Pr6 |
| B | 26 | 06 | 1 | e | Imm7 | | Ra6 | Rt6 | Pr6 |

**Exceptions:** none

**Execution Units:** All ALUs

### SRA – Shift Right Arithmetic

**Description:**

Shift the value in Ra to the right by the number of bits specified in the second source operand. The sign bit is preserved in the most significant bit.

**Instruction Formats:**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 39 | 38 33 | 32 27 | 26 | 25 | 24 | 23 18 | 17 12 | 11 6 | 5 0 |
| B | 26 | 26 | 0 | e | ~ | Rb6 | Ra6 | Rt6 | Pr6 |
| B | 26 | 26 | 1 | e | Imm7 | | Ra6 | Rt6 | Pr6 |

**Exceptions:** none

**Execution Units:** ALU #0 Only

### SRL – Shift Right Logical

**Description:**

Shift the value in Ra to the right by the number of bits specified in the second source operand. Zeros are shifted into the most significant bits.

**Instruction Formats:**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 39 | 38 33 | 32 27 | 26 | 25 | 24 | 23 18 | 17 12 | 11 6 | 5 0 |
| B | 26 | 16 | 0 | e | ~ | Rb6 | Ra6 | Rt6 | Pr6 |
| B | 26 | 16 | 1 | e | Imm7 | | Ra6 | Rt6 | Pr6 |

**Exceptions:** none

**Execution Units:** ALU #0 Only

### SUBF – Subtract From

**Description:**

Subtract source operand A from operand B and place the result in the target register.

**Instruction Formats:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 39 | 38 33 | 32 27 | 26 24 | 23 18 | 17 12 | 11 6 | 5 0 |
| B | 26 | 56 | ~3 | Rb6 | Ra6 | Rt6 | Pred6 |
| B | 56 | Imm15 | | | Ra6 | Rt6 | Pred6 |

**Exceptions:** none

**Execution Units:** All ALUs

### XOR – Bitwise Exclusive Or

**Description:**

Bitwise exclusive or two source operands and place the result in the target register. The immediate constant is sign extended to the machine width.

**Instruction Formats:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 39 | 38 33 | 32 27 | 26 24 | 23 18 | 17 12 | 11 6 | 5 0 |
| B | 26 | 106 | ~3 | Rb6 | Ra6 | Rt6 | Pred6 |
| B | 106 | Imm15 | | | Ra6 | Rt6 | Pred6 |

**Exceptions:** none

**Execution Units:** All ALUs

## Program Flow Control Operations

### BMR – Branch to Milli-code Routine

**Description:**

Branches to the target address if the qualifying predicate is true. The target address is the sum of a displacement specified in the instruction and the current instruction pointer. The address of the next instruction is stored in register R2.

**Instruction Format:**

|  |  |  |  |
| --- | --- | --- | --- |
| 39 | 38 33 | 34 6 | 5 0 |
| B | 156 | Displacement27 | Pr6 |

**Operation:**

if (Pr)

R2 = next IP

IP = IP + Displacement

### BRA – Branch

**Description:**

Branches to the target address if the qualifying predicate is true. The target address is the sum of a displacement specified in the instruction and the current instruction pointer.

**Instruction Format:**

|  |  |  |  |
| --- | --- | --- | --- |
| 39 | 38 33 | 34 6 | 5 0 |
| B | 136 | Displacement27 | Pr6 |

**Operation:**

if (Pr)

IP = IP + Displacement

### BSR – Branch to Subroutine

**Description:**

Branches to the target address if the qualifying predicate is true. The target address is the sum of a displacement specified in the instruction and the current instruction pointer. The address of the next instruction is stored in register R1.

**Instruction Format:**

|  |  |  |  |
| --- | --- | --- | --- |
| 39 | 38 33 | 34 6 | 5 0 |
| B | 146 | Displacement27 | Pr6 |

**Operation:**

if (Pr)

R1 = next IP

IP = IP + Displacement

### JMR – Jump to Register

**Description:**

This instruction performs a jump to a target address specified in register Ra. The return address is stored in register Rt. This instruction may be used to return from a subroutine.

**Instruction Formats:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| B | 26 | 136 | ~9 | Ra6 | Rt6 | Pr6 |

**Exceptions:** none

**Execution Units:** Branch

### NOP – No Operation

**Description:**

No operation is performed by this instruction.

**Instruction Format:**

|  |  |  |  |
| --- | --- | --- | --- |
| 39 | 38 33 | 34 6 | 5 0 |
| B | 06 | ~27 | 06 |

**Operation:** none

**Exceptions:** none

**Execution Units:** Branch

## Memory Operations

### LDB – Load Byte

**Description:**

Load a byte from memory and sign extend the value to the machine width. The address is either the sum of register Ra and Rb, or the sum of Ra and an immediate value.

**Instruction Formats:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 39 | 38 33 | 32 27 | 26 24 | 23 18 | 17 12 | 11 6 | 5 0 |
| B | 26 | 326 | ~3 | Rb6 | Ra6 | Rt6 | Pred6 |
| B | 326 | Imm15 | | | Ra6 | Rt6 | Pred6 |

**Exceptions:** none

**Execution Units:** All Memory

### LDBU – Load Unsigned Byte

**Description:**

Load a byte from memory and zero extend the value to the machine width. The address is either the sum of register Ra and Rb, or the sum of Ra and an immediate value.

**Instruction Formats:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 39 | 38 33 | 32 27 | 26 24 | 23 18 | 17 12 | 11 6 | 5 0 |
| B | 26 | 336 | ~3 | Rb6 | Ra6 | Rt6 | Pred6 |
| B | 336 | Imm15 | | | Ra6 | Rt6 | Pred6 |

**Exceptions:** none

**Execution Units:** All Memory

### LDD – Load Deci

**Description:**

Load a deci-byte value from memory and sign extend the value to the machine width. The address is either the sum of register Ra and Rb, or the sum of Ra and an immediate value.

**Instruction Formats:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 39 | 38 33 | 32 27 | 26 24 | 23 18 | 17 12 | 11 6 | 5 0 |
| B | 26 | 426 | ~3 | Rb6 | Ra6 | Rt6 | Pred6 |
| B | 426 | Imm15 | | | Ra6 | Rt6 | Pred6 |

**Exceptions:** none

**Execution Units:** All Memory

### LDO – Load Octa

**Description:**

Load a octa from memory and sign extend the value to the machine width. The address is either the sum of register Ra and Rb, or the sum of Ra and an immediate value.

**Instruction Formats:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 39 | 38 33 | 32 27 | 26 24 | 23 18 | 17 12 | 11 6 | 5 0 |
| B | 26 | 386 | ~3 | Rb6 | Ra6 | Rt6 | Pred6 |
| B | 386 | Imm15 | | | Ra6 | Rt6 | Pred6 |

**Exceptions:** none

**Execution Units:** All Memory

### LDOU – Load Octa Unsigned

**Description:**

Load an octa from memory and zero extend the value to the machine width. The address is either the sum of register Ra and Rb, or the sum of Ra and an immediate value.

**Instruction Formats:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 39 | 38 33 | 32 27 | 26 24 | 23 18 | 17 12 | 11 6 | 5 0 |
| B | 26 | 396 | ~3 | Rb6 | Ra6 | Rt6 | Pred6 |
| B | 396 | Imm15 | | | Ra6 | Rt6 | Pred6 |

**Exceptions:** none

**Execution Units:** All Memory

### LDP – Load Penta

**Description:**

Load a penta-byte value from memory and sign extend the value to the machine width. The address is either the sum of register Ra and Rb, or the sum of Ra and an immediate value.

**Instruction Formats:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 39 | 38 33 | 32 27 | 26 24 | 23 18 | 17 12 | 11 6 | 5 0 |
| B | 26 | 406 | ~3 | Rb6 | Ra6 | Rt6 | Pred6 |
| B | 406 | Imm15 | | | Ra6 | Rt6 | Pred6 |

**Exceptions:** none

**Execution Units:** All Memory

### LDPU – Load Penta Unsigned

**Description:**

Load a penta-byte value from memory and zero extend the value to the machine width. The address is either the sum of register Ra and Rb, or the sum of Ra and an immediate value.

**Instruction Formats:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 39 | 38 33 | 32 27 | 26 24 | 23 18 | 17 12 | 11 6 | 5 0 |
| B | 26 | 416 | ~3 | Rb6 | Ra6 | Rt6 | Pred6 |
| B | 416 | Imm15 | | | Ra6 | Rt6 | Pred6 |

**Exceptions:** none

**Execution Units:** All Memory

### LDT – Load Tetra

**Description:**

Load a tetra from memory and sign extend the value to the machine width. The address is either the sum of register Ra and Rb, or the sum of Ra and an immediate value.

**Instruction Formats:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 39 | 38 33 | 32 27 | 26 24 | 23 18 | 17 12 | 11 6 | 5 0 |
| B | 26 | 366 | ~3 | Rb6 | Ra6 | Rt6 | Pred6 |
| B | 366 | Imm15 | | | Ra6 | Rt6 | Pred6 |

**Exceptions:** none

**Execution Units:** All Memory

### LDTU – Load Unsigned Tetra

**Description:**

Load a tetra from memory and zero extend the value to the machine width. The address is either the sum of register Ra and Rb, or the sum of Ra and an immediate value.

**Instruction Formats:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 39 | 38 33 | 32 27 | 26 24 | 23 18 | 17 12 | 11 6 | 5 0 |
| B | 26 | 376 | ~3 | Rb6 | Ra6 | Rt6 | Pred6 |
| B | 376 | Imm15 | | | Ra6 | Rt6 | Pred6 |

**Exceptions:** none

**Execution Units:** All Memory

### LDW – Load Wyde

**Description:**

Load a wyde from memory and sign extend the value to the machine width. The address is either the sum of register Ra and Rb, or the sum of Ra and an immediate value.

**Instruction Formats:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 39 | 38 33 | 32 27 | 26 24 | 23 18 | 17 12 | 11 6 | 5 0 |
| B | 26 | 346 | ~3 | Rb6 | Ra6 | Rt6 | Pred6 |
| B | 346 | Imm15 | | | Ra6 | Rt6 | Pred6 |

**Exceptions:** none

**Execution Units:** All Memory

### LDWU – Load Unsigned Wyde

**Description:**

Load a wyde from memory and zero extend the value to the machine width. The address is either the sum of register Ra and Rb, or the sum of Ra and an immediate value.

**Instruction Formats:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 39 | 38 33 | 32 27 | 26 24 | 23 18 | 17 12 | 11 6 | 5 0 |
| B | 26 | 356 | ~3 | Rb6 | Ra6 | Rt6 | Pred6 |
| B | 356 | Imm15 | | | Ra6 | Rt6 | Pred6 |

**Exceptions:** none

**Execution Units:** All Memory

### STB – Store Byte

**Description:**

Store a byte to memory from register Rs. The address is either the sum of register Ra and Rc, or the sum of Ra and an immediate value. Rc must be specified using the REG prefix.

**Instruction Formats:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 39 | 38 33 | 32 27 | 26 24 | 23 18 | 17 12 | 11 6 | 5 0 |
| B | 26 | 486 | ~3 | Rb6 | Ra6 | Rs6 | Pred6 |
| B | 486 | Imm15 | | | Ra6 | Rs6 | Pred6 |

**Exceptions:** none

**Execution Units:** Memory #0

### STD – Store Deci

**Description:**

Store a deci-byte to memory from register Rs. The address is either the sum of register Ra and Rc, or the sum of Ra and an immediate value. Rc must be specified using the REG prefix.

**Instruction Formats:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 39 | 38 33 | 32 27 | 26 24 | 23 18 | 17 12 | 11 6 | 5 0 |
| B | 26 | 536 | ~3 | Rb6 | Ra6 | Rs6 | Pred6 |
| B | 536 | Imm15 | | | Ra6 | Rs6 | Pred6 |

**Exceptions:** none

**Execution Units:** Memory #0

### STO – Store Octa

**Description:**

Store an octa to memory from register Rs. The address is either the sum of register Ra and Rc, or the sum of Ra and an immediate value. Rc must be specified using the REG prefix.

**Instruction Formats:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 39 | 38 33 | 32 27 | 26 24 | 23 18 | 17 12 | 11 6 | 5 0 |
| B | 26 | 516 | ~3 | Rb6 | Ra6 | Rs6 | Pred6 |
| B | 516 | Imm15 | | | Ra6 | Rs6 | Pred6 |

**Exceptions:** none

**Execution Units:** Memory #0

### STP – Store Penta

**Description:**

Store a penta-byte to memory from register Rs. The address is either the sum of register Ra and Rc, or the sum of Ra and an immediate value. Rc must be specified using the REG prefix.

**Instruction Formats:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 39 | 38 33 | 32 27 | 26 24 | 23 18 | 17 12 | 11 6 | 5 0 |
| B | 26 | 526 | ~3 | Rb6 | Ra6 | Rs6 | Pred6 |
| B | 526 | Imm15 | | | Ra6 | Rs6 | Pred6 |

**Exceptions:** none

**Execution Units:** Memory #0

### STT – Store Tetra

**Description:**

Store a tetra to memory from register Rs. The address is either the sum of register Ra and Rc, or the sum of Ra and an immediate value. Rc must be specified using the REG prefix.

**Instruction Formats:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 39 | 38 33 | 32 27 | 26 24 | 23 18 | 17 12 | 11 6 | 5 0 |
| B | 26 | 506 | ~3 | Rb6 | Ra6 | Rs6 | Pred6 |
| B | 506 | Imm15 | | | Ra6 | Rs6 | Pred6 |

**Exceptions:** none

**Execution Units:** Memory #0

### STW – Store Wyde

**Description:**

Store a wyde to memory from register Rs. The address is either the sum of register Ra and Rc, or the sum of Ra and an immediate value. Rc must be specified using the REG prefix.

**Instruction Formats:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 39 | 38 33 | 32 27 | 26 24 | 23 18 | 17 12 | 11 6 | 5 0 |
| B | 26 | 496 | ~3 | Rb6 | Ra6 | Rs6 | Pred6 |
| B | 496 | Imm15 | | | Ra6 | Rs6 | Pred6 |

**Exceptions:** none

**Execution Units:** Memory #0

## Special Operations

### CARRY – Carry Modifier

**Description:**

This instruction indicates where there are carries into and out of following instructions for up to eight instructions.

**Instruction Formats:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 25 | 305 | ~3 | IO7 | IO6 | IO5 | IO4 | IO3 | IO2 | IO1 | IO0 | Rt5 | Pred6 |

|  |  |
| --- | --- |
| IO |  |
| 0 | no carries |
| 1 | input carry |
| 2 | output carry |
| 3 | input and output carry |

**Exceptions:** none

**Execution Units:** All ALUs

### CON – Constant Prefix

**Description:**

Add constant bits to the following instruction. Constants are extended from bit six of the constant. There may be multiple constant prefixes present to extend the constant up to 80 bits.

*The compare instruction has only a six-bit immediate field. So that the constant prefix may be used with the compare instruction it extends from bit six. The constant prefix is simplified by fixing which bit it extends from. It would be too much hardware for the value to vary which bit is extended according to the instruction.*

*Since constants are never executed the predicate for them is always zero.*

*Breaks should not be inserted in the middle of constants for proper operation.*

**Instruction Formats:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 39 | 38 33 | 34 6 | | 5 0 |
| B | 616 | Immediate32..6 | | 06 |
| B | 626 | Immediate59..33 | | 06 |
| B | 636 | ~8 | Immediate79..60 | 06 |

**Exceptions:** none

**Execution Units:** All Decoders

### REG – Fetch Registers

**Description:**

This instruction fetches two more operands to be used in the following instruction and provides an additional target register. This instruction is used for indexed stores, multiply and add and fused dot product instructions.

**Instruction Format:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 39 | 38 33 | 32 27 | 26 24 | 23 18 | 17 12 | 11 6 | 5 0 |
| B | 26 | 636 | ~3 | Rb6 | Ra6 | Rt6 | Pred6 |
| B | 636 | Imm15 | | | Ra6 | Rt6 | Pred6 |