Black Widow

Introduction

The BlackWidow ISA and core is the author's attempt at a WEX 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 a 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	$Opcode_6$	$Func_6$	$\sim_3$	Rb_6	Ra_6	Rt_6	Pr_6	
B	$Opcode_6$	$Immediate_{15}$			Ra_6	Rt_6	Pr_6	
B	$Opcode_6$	Op_3	$Immediate_{12}$		Ra_6	Rt_6	Pr_6	SETI / SETUI
B	$Opcode_6$	Op_3	$Pt2_6$	$Pt1_6$	Rb_6	Ra_6	Pr_6	CMP
B	$Opcode_6$	Op_3	$Pt2_6$	$Pt1_6$	Imm_6	Ra_6	Pr_6	CMPI
B	$Opcode_6$	$Displacement_{27}$					Pr_6	BRA / BSR
B	$Opcode_6$	$Immediate_{27}$					Pr_6	CON1/2/3

Opcode Maps

Major Opcode$_6$

	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 BNZ	14 BZ	15 BSR
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	59 CON4	60 NOP	61 CON1	62 CON2	63 CON3

Opcode 2, Major Func$_6$

	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, ADDI – 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	2_6	4_6	$\sim_3$	Rc_6	Rb_6	Ra_6	Rt_6
B	4_6	Imm_{21}				Ra_6	Rt_6

Exceptions: none

Execution Units: All ALUs

AND, ANDI – 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	2_6	8_6	$\sim_3$	Rc_6	Rb_6	Ra_6	Rt_6
B	8_6	Imm_{21}				Ra_6	Rt_6

Exceptions: none

Execution Units: All ALUs

CMP,CMPI – Compare

Description:

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

Instruction Formats:

39	38 33	32 30	29 24	23 18	17 12	11 6	5 0
B	$Opcode_6$	Op_3	$\sim_6$	$\sim_6$	Rb_6	Ra_6	Rt_6
B	$Opcode_6$	Op_3	Imm_{18}			Ra_6	Rt_6

Op_3	
0	LT
1	GE
2	LE
3	GT
4	EQ
5	NE
6	BC
7	BS

$Opcode_6$	
16	CMP
17	CMPU
18	FCMP
20	CMPI
21	CMPUI
22	FCMPI

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
2_5	15_5	Op_2	M_2	CSR_{10}	Ra_5	Rt_5	$Pred_6$

Op_2	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	2_6	12_6	$\sim_3$	Rc_6	Rb_6	Ra_6	Rt_6
B	12_6	Imm_{21}				Ra_6	Rt_6

Op_3	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	2_6	9_6	$\sim_3$	Rc_6	Rb_6	Ra_6	Rt_6
B	9_6	Imm_{21}				Ra_6	Rt_6

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	2_6	0_6	0	e	~	$\sim_6$	Rb_6	Ra_6	Rt_6
B	2_6	0_6	1	e	$\sim_6$		$Immed_7$	Ra_6	Rt_6

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	2_6	2_6	0	e	~	$\sim_6$	Rb_6	Ra_6	Rt_6
B	2_6	2_6	1	e	$\sim_6$		$Immed_7$	Ra_6	Rt_6

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	2_6	1_6	0	e	~	$\sim_6$	Rb_6	Ra_6	Rt_6
B	2_6	1_6	1	e	$\sim_6$		$Immed_7$	Ra_6	Rt_6

Exceptions: none

Execution Units: ALU #0 Only

SUB – Subtract

Description:

Subtract source operand B from operand A 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	2_6	5_6	$\sim_3$	Rc_6	Rb_6	Ra_6	Rt_6

Exceptions: none

Execution Units: All ALUs

SUBFI – 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	5_6	Imm_{21}	Ra_6	Rt_6

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	2_6	10_6	$\sim_3$	Rc_6	Rb_6	Ra_6	Rt_6
B	10_6	Imm_{21}				Ra_6	Rt_6

Exceptions: none

Execution Units: All ALUs

Program Flow Control Operations

BNZ – Branch if Non-Zero

Description:

Branches to the target address if Ra is non-zero. 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	13_6	$Displacement_{27}$	Ra_6

Operation:

```
if (Ra)
        IP = IP + Displacement
```

BRA – Branch

Description:

Branches to the target address. 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	14_6	$Displacement_{27}$	0_6

Operation:

```
if (Ra)
        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	15_6	$Displacement_{27}$	Rt_6

Operation:

Rt = next IP
IP = IP + Displacement

BZ – Branch if Zero

Description:

Branches to the target address if Ra is zero. 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	14_6	$Displacement_{27}$	Ra_6

Operation:

if (Ra)
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	2_6	13_6	$\sim_9$	$\sim_6$	Ra_6	Rt_6

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	60_6	$\sim_{33}$

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	2_6	32_6	$\sim_3$	Rc_6	Rb_6	Ra_6	Rt_6
B	32_6	Imm_{21}				Ra_6	Rt_6

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	2_6	33_6	$\sim_3$	Rc_6	Rb_6	Ra_6	Rt_6
B	33_6	Imm_{21}				Ra_6	Rt_6

Exceptions: none

Execution Units: All Memory

LDH – Load Hexi

Description:

Load a hexi-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	2_6	42_6	$\sim_3$	Rc_6	Rb_6	Ra_6	Rt_6
B	42_6	Imm_{21}				Ra_6	Rt_6

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	2_6	38_6	$\sim_3$	Rc_6	Rb_6	Ra_6	Rt_6
B	38_6	Imm_{21}				Ra_6	Rt_6

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	2_6	39_6	$\sim_3$	Rc_6	Rb_6	Ra_6	Rt_6
B	39_6	Imm_{21}				Ra_6	Rt_6

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	2_6	36_6	$\sim_3$	Rc_6	Rb_6	Ra_6	Rt_6
B	36_6	Imm_{21}				Ra_6	Rt_6

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	2_6	37_6	$\sim_3$	Rc_6	Rb_6	Ra_6	Rt_6
B	37_6	Imm_{21}				Ra_6	Rt_6

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	2_6	34_6	$\sim_3$	Rc_6	Rb_6	Ra_6	Rt_6
B	34_6	Imm_{21}				Ra_6	Rt_6

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	2_6	35_6	$\sim_3$	Rc_6	Rb_6	Ra_6	Rt_6
B	35_6	Imm_{21}				Ra_6	Rt_6

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	2_6	48_6	$\sim_3$	Rc_6	Rb_6	Ra_6	Rs_6
B	48_6	Imm_{21}				Ra_6	Rs_6

Exceptions: none

Execution Units: Memory #0

STH – Store Hexi

Description:

Store a hexi-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	2_6	53_6	$\sim_3$	Rc_6	Rb_6	Ra_6	Rs_6
B	53_6	Imm_{21}				Ra_6	Rs_6

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	2_6	51_6	$\sim_3$	Rc_6	Rb_6	Ra_6	Rs_6
B	51_6	Imm_{21}				Ra_6	Rs_6

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	2_6	50_6	$\sim_3$	Rc_6	Rb_6	Ra_6	Rs_6
B	50_6	Imm_{21}				Ra_6	Rs_6

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	2_6	49_6	$\sim_3$	Rc_6	Rb_6	Ra_6	Rs_6
B	49_6	Imm_{21}				Ra_6	Rs_6

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:

2_5	30_5	$\sim_3$	IO_7	IO_6	IO_5	IO_4	IO_3	IO_2	IO_1	IO_0	Rt_5	$Pred_6$

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 18 of the constant. There may be multiple constant prefixes present to extend the constant up to 128 bits.

The compare instruction has only a 18-bit immediate field. So that the constant prefix may be used with the compare instruction it extends from bit 18. 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.

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

Instruction Formats:

39	38 33	34 6 5 0
B	61_6	$Immediate_{50..18}$
B	62_6	$Immediate_{84..51}$
B	63_6	$Immediate_{117..85}$
B	59_6	$Immediate_{150..118}$

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	2_6	63_6	$\sim_3$	Rb_6	Ra_6	Rt_6	$Pred_6$
B	63_6	Imm_{15}			Ra_6	Rt_6	$Pred_6$