



BRAD DEVICES

Built in Africa, for the World

BradISA V1

Instruction Set Architecture Specification

Revision 1.0 — Kinetic Gen1 — July 2026

Confidential — Brad Devices Proprietary

Contents

1	Introduction	3
1.1	Data Model	3
1.2	Pipeline Models	3
2	Register File	4
2.1	ABI Convention	4
2.2	Calling Convention	4
3	Instruction Formats	5
3.1	RRR Format — Register-to-Register ALU	5
3.2	RI Format — Register-Immediate	5
3.3	BR Format — Branch	5
3.4	JMP Format — Unconditional Jump / Call	6
3.5	RET — Return	6
4	Instruction Set	7
4.1	ALU Operations	7
4.2	Memory Operations	7
4.3	Conditional Branches	7
4.4	Call / Return	7
5	Instruction Encoding Diagrams	8
5.1	RRR Format (ALU)	8
5.2	RI Format (ADDI, LDW, STW)	8
5.3	BR Format (BZ, BNZ)	8
5.4	JMP Format (JMP, CALL)	8
5.5	RET	8
6	Exception and Interrupt Model	9
6.1	Machine Status Registers	9
6.2	Status Register Flags	9
7	Performance Characteristics	10
7.1	IPC Estimates	10
7.2	Dhrystone Estimates	10
7.3	Pipeline Stage Descriptions	10
7.4	Vector Extension (VSET)	10

1 Introduction

BradISA V1 (Kinetic Gen1) is a 32-bit fixed-length RISC instruction set architecture designed for the Brad Devices BradCore processor family. It provides 16 general-purpose registers, 16 opcodes encoded in a 4-bit primary opcode field, and three primary instruction formats.

1.1 Data Model

- **Word size:** 32 bits (4 bytes)
- **Address space:** 32-bit byte-addressed
- **Page size:** 4096 bytes (4 KiB)
- **Endianness:** Little-endian
- **Instruction length:** 32 bits, fixed

1.2 Pipeline Models

BradISA defines two pipeline models:

Table 1: Pipeline model comparison

Property	Phoenix	Falcon
Pipeline depth	10 stages	5 stages
Issue width	Octa-issue	Single-issue
Execution order	Out-of-order	In-order
ROB size	128 entries	—
Typical IPC	2.2	0.9
Peak IPC	5.0	1.0
Target DMIPS/MHz	5.5	2.2
Target frequency	3.0–5.2 GHz	2.4–3.6 GHz

2 Register File

BradISA V1 provides 16 32-bit general-purpose registers, numbered **r0** through **r15**. Register **r0** is hardwired to zero; writes to **r0** are silently discarded.

2.1 ABI Convention

Table 2: Application Binary Interface register map

Register	ABI Name	Purpose
r0	zero	Hardwired zero
r1	a0	Argument #0 / return value
r2	a1	Argument #1
r3	a2	Argument #2
r4	a3	Argument #3
r5	t0	Temporary #0
r6	t1	Temporary #1
r7	t2	Temporary #2
r8	s0	Callee-saved #0
r9	s1	Callee-saved #1
r10	s2	Callee-saved #2
r11	s3	Callee-saved #3
r12	s4	Callee-saved #4
r13	sp	Stack pointer
r14	lr	Link register
r15	pc	Program counter (read-only)

2.2 Calling Convention

- Arguments 0–3 are passed in **r1-r4**. Additional arguments are passed on the stack.
- Return value is placed in **r1**.
- Callee-saved registers **r8-r12** must be preserved across function calls.
- The link register **r14** holds the return address on **CALL**.
- The stack grows downward; **r13** (**sp**) must point to the last used stack slot.

3 Instruction Formats

All BradISA V1 instructions are 32 bits wide. The three primary formats are:

Table 3: Instruction format field layout

Format	Bits [31:28]	[27:24]	[23:20]	[19:16]	[15:0]
RRR	opcode	rd	rs1	rs2	—
RI	opcode	rd	rs1	—	immediate
BR	opcode	—	rs1	—	offset ($\times 4$)
JMP	opcode	—	—	—	offset ($\times 4$)

3.1 RRR Format — Register-to-Register ALU

Used by all eight ALU instructions (ADD, SUB, MUL, AND, OR, XOR, SHL, SHR).

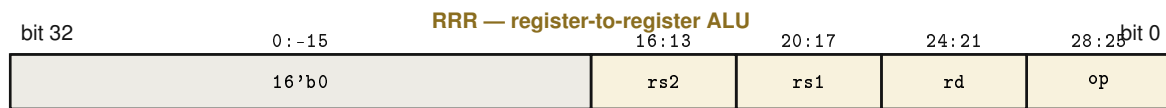


Figure 1: RRR format — 32-bit register-to-register encoding (gold = field, grey = reserved/zero)

$$\text{instruction}[31:0] = \{\text{op}[3:0], \text{rd}[3:0], \text{rs1}[3:0], \text{rs2}[3:0], 16'b0\}$$

3.2 RI Format — Register-Immediate

Used by ADDI, LDW, and STW.



Figure 2: RI format — register-immediate encoding (sign-extended immediate)

$$\text{instruction}[31:0] = \{\text{op}[3:0], \text{rd}[3:0], \text{rs1}[3:0], \text{imm}[15:0]\}$$

The 16-bit immediate is sign-extended to 32 bits before use.

3.3 BR Format — Branch

Used by BZ and BNZ.

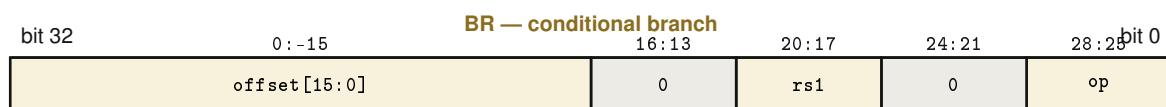


Figure 3: BR format — 16-bit word offset (“ $\times 4$ ” at encode time, ± 128 KiB range)

$$\text{instruction}[31:0] = \{\text{op}[3:0], 4'b0, \text{rs1}[3:0], 4'b0, \text{offset}[15:0]\}$$

The branch offset is encoded in **words** (multiplied by 4 at encode time). At execution, the target address is:

$$\text{PC} = \text{insn_pc} + 4 + \text{sign_extend}(\text{offset}[15:0])$$

Branch offsets are therefore ± 128 KiB from the instruction.

3.4 JMP Format — Unconditional Jump / Call

Used by JMP and CALL.

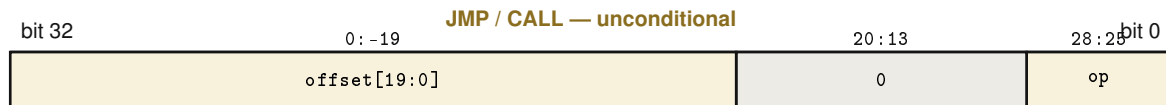


Figure 4: JMP/CALL format — 20-bit word offset (“×4” at encode time, ±4 MiB range)

$$\text{instruction}[31:0] = \{\text{op}[3:0], 8'b0, \text{offset}[19:0]\}$$

The jump offset is in words (×4). Target address:

$$\text{PC} = \text{insn_pc} + 4 + \text{sign_extend}(\text{offset}[19:0])$$

Jump range is ±4 MiB.

3.5 RET — Return

Fixed encoding:

$$\text{RET} = 32'hF0000000$$

Branches to the address in **r14** (**lr**). Does not modify **lr**.

4 Instruction Set

Table 4: Complete BradISA V1 instruction set

Opcode	Mnemonic	Format	Operation
0x0	ADD	RRR	$rd = rs1 + rs2$
0x1	SUB	RRR	$rd = rs1 - rs2$
0x2	MUL	RRR	$rd = rs1 \times rs2$ (low 32 bits)
0x3	AND	RRR	$rd = rs1 \& rs2$
0x4	OR	RRR	$rd = rs1 rs2$
0x5	XOR	RRR	$rd = rs1 \wedge rs2$
0x6	SHL	RRR	$rd = rs1 \ll rs2[4:0]$
0x7	SHR	RRR	$rd = rs1 \gg rs2[4:0]$ (logical)
0x8	ADDI	RI	$rd = rs1 + sext(imm16)$
0x9	LDW	RI	$rd = mem[rs1 + sext(imm16)]$
0xA	STW	RI	$mem[rs1 + sext(imm16)] = rs2$
0xB	BZ	BR	if ($rs1 == 0$) $PC = insn_pc + 4 + sext(offset \times 4)$
0xC	BNZ	BR	if ($rs1 \neq 0$) $PC = insn_pc + 4 + sext(offset \times 4)$
0xD	JMP	JMP	$PC = insn_pc + 4 + sext(offset \times 4)$
0xE	CALL	JMP	$lr = insn_pc + 4$; $PC = target$
0xF	RET	—	$PC = lr$

4.1 ALU Operations

All ALU operations (opcodes 0x0–0x7) use the RRR format. The second source operand is a register (**rs2**), not an immediate. **MUL** produces the low 32 bits of the product. **SHL** and **SHR** use only the bottom 5 bits of **rs2** as the shift amount (0–31).

4.2 Memory Operations

LDW loads a 32-bit word from byte address $rs1 + sext(imm16)$ into **rd**. **STW** stores the value of **rs2** to the same effective address. Addresses must be 4-byte aligned; an unaligned access raises an exception.

4.3 Conditional Branches

BZ and **BNZ** test **rs1** against zero. The branch offset is a 16-bit signed word offset ($\times 4$) with a range of ± 128 KiB. The branch target PC-relative computation uses the address of the instruction itself, not the next instruction:

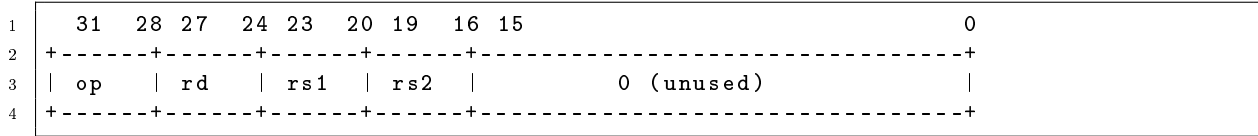
$$target = insn_pc + 4 + (offset \times 4)$$

4.4 Call / Return

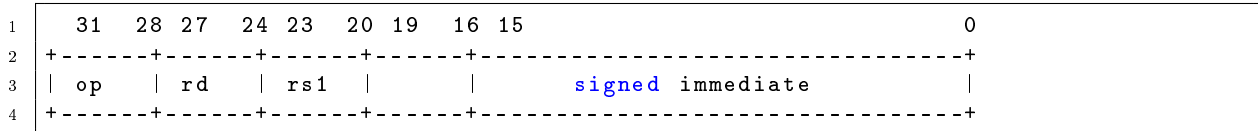
CALL saves the return address ($insn_pc + 4$) into the link register **r14** (**lr**) and then jumps to the target. **RET** jumps unconditionally to **lr**, providing a simple leaf-function call/return mechanism.

5 Instruction Encoding Diagrams

5.1 RRR Format (ALU)

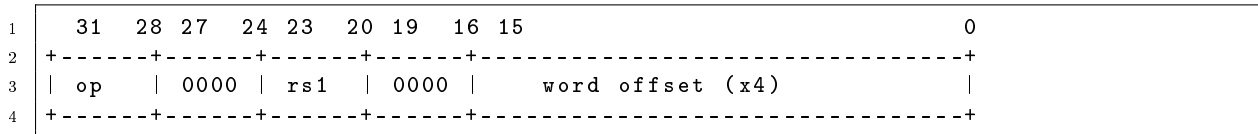


5.2 RI Format (ADDI, LDW, STW)

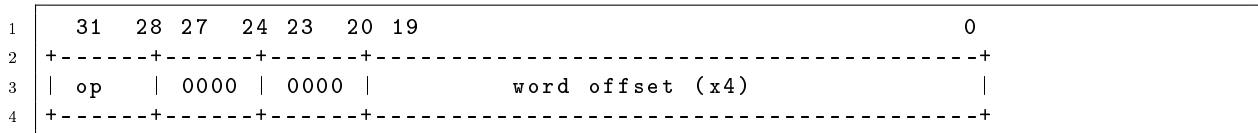


Note: STW ignores **rd** (uses **rs2** as store data).

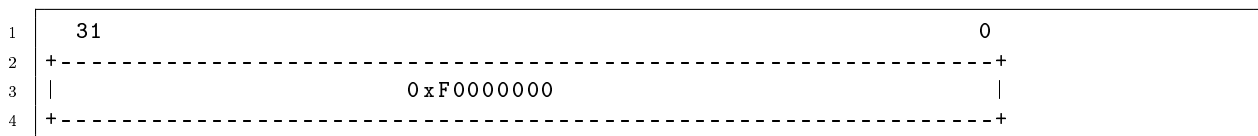
5.3 BR Format (BZ, BNZ)



5.4 JMP Format (JMP, CALL)



5.5 RET



6 Exception and Interrupt Model

BradISA V1 supports a vectored exception model with a fixed vector table at the base of the address space.

Table 5: Exception vector addresses

Vector address	Exception	Cause code
0x00000000	Reset / cold boot	—
0x00000004	Undefined instruction	1
0x00000008	Page fault	2
0x0000000C	Unaligned access	3
0x00000010	Privilege violation	4
0x00000014	System call (ECALL)	5
0x00000018	Timer interrupt	6
0x00000020	External IRQ 0	7
0x00000024	External IRQ 1	7
0x00000028	External IRQ 2	7
0x0000002C	External IRQ 3	7

6.1 Machine Status Registers

Table 6: MSR index map

Index	Mnemonic	Description
0	STATUS	Processor status flags
1	CAUSE	Cause of last exception
2	EPC	Exception program counter
3	EAR	Exception address register
4	PAGE_BASE	Page table base address
5	TICK	Cycle count timer
6	CORE_ID	Core identifier
7	CLUSTER_ID	Cluster identifier

6.2 Status Register Flags

Bit	Flag	Description
0	IE	Interrupt enable
1	SVC	Supervisor mode
2	HALTED	Core halted
3	WFE	Wait for event
8	EXC_PEND	Exception pending

7 Performance Characteristics

7.1 IPC Estimates

- **Phoenix (10-stage OoO):** 2.2 IPC typical, 5.0 IPC peak
- **Falcon (5-stage in-order):** 0.9 IPC typical, 1.0 IPC peak

7.2 Dhrystone Estimates

- **Phoenix:** ~5.5 DMIPS/MHz
- At 5.2 GHz: ~28,600 DMIPS
- **Falcon:** ~2.2 DMIPS/MHz
- At 3.6 GHz: ~7,920 DMIPS

7.3 Pipeline Stage Descriptions

Phoenix (10-stage)

1. **Fetch1** — Instruction TLB access, next-PC selection
2. **Fetch2** — Instruction cache read
3. **Decode** — Instruction decode, register rename
4. **Rename** — Register map read, free-list access
5. **Dispatch** — Insert into issue queue / ROB
6. **Issue** — Wakeup, select, read physical register file
7. **Execute** — ALU / AGU / branch resolution
8. **Memory** — Data cache access (load/store)
9. **Writeback** — Result write to physical register file
10. **Commit** — Retire from ROB, update architectural state

Falcon (5-stage)

1. **Fetch1** — Instruction fetch, PC increment
2. **Fetch2** — Instruction cache / memory read
3. **Decode** — Instruction decode, register file read
4. **Execute** — ALU / AGU / branch resolution, writeback
5. **Writeback** — Register file write

7.4 Vector Extension (VSET)

BradISA V1 has no floating-point or SIMD facility; both are provided by the BradISA V2 Vector Extension (VSET), specified in the companion document **BradISA V2 Extensions**. VSET adds 32×256 -bit vector registers (**v0–v31**), elementwise integer/FP operations, vector memory, reductions, and converts, encoded under primary opcode **0xF** with **0xF0000000** retained as **RET** for backward compatibility. Enable via MSR 23 (**VSTATUS.VE**).

Revision History

Revision	Date	Changes
1.0	July 2026	Initial specification — Kinetic Gen1
1.1	July 2026	Phoenix upgraded to octa-issue OoO (ROB 128), IPC 2.2/5.0, DMIPS/MHz 5.5; Falcon DMIPS/MHz aligned to 2.2
1.2	July 2026	Cross-reference BradISA V2 Vector Extension (VSET), opcode 0xF, VSTATUS MSR 23