

Frost32 Instruction Set

- General Purpose Registers (32-bit):
 - `zero` (always zero), `u0` , `u1` , `u2` , `u3` , `u4` , `u5` , `u6` , `u7` , `u8` , `u9` , `u10` , `temp` (assembler temporary (but can be used otherwise)), `lr` (upon any call instruction, return address stored here), `fp` (recommended for use as the frame pointer), `sp` (recommended for use as the stack pointer)
- Special Purpose Registers (32-bit)
 - `pc` (program counter), `ireta` (interrupt return address) `idsta` (interrupt destination address)
- Special Purpose Registers (1-bit)
 - `ie` (interrupt enable)
- Opcode Group: 0b0000
 - Encoding: 0000 aaaa bbbb cccc 0000 0000 0000 0000
 - `a` : rA
 - `b` : rB
 - `c` : rC
 - `o` : opcode
 - Instructions:
 - **add** rA, rB, rC
 - Opcode: 0x0
 - **sub** rA, rB, rC
 - Opcode: 0x1
 - **sltu** rA, rB, rC
 - Opcode: 0x2
 - **slts** rA, rB, rC
 - Opcode: 0x3
 - **sgtu** rA, rB, rC
 - Opcode: 0x4
 - **sgts** rA, rB, rC
 - Opcode: 0x5
 - **mul** rA, rB, rC
 - Opcode: 0x6
 - **and** rA, rB, rC
 - Opcode: 0x7
 - **orr** rA, rB, rC
 - Opcode: 0x8
 - **xor** rA, rB, rC
 - Opcode: 0x9
 - **nor** rA, rB, rC
 - Opcode: 0xa
 - **lsl** rA, rB, rC

- Opcode: 0xb
 - **lsl** rA, rB, rC
 - Opcode: 0xc
 - **asr** rA, rB, rC
 - Opcode: 0xd
- Opcode Group: 0b0001
 - Encoding: 0001 aaaa bbbb oooo iiii iiii iiii iiii
 - a : rA
 - b : rB
 - o : opcode
 - i : 16-bit immediate
 - Instructions:
 - **addi** rA, rB, imm16
 - Opcode: 0x0
 - **subi** rA, rB, imm16
 - Opcode: 0x1
 - **sltui** rA, rB, imm16
 - Opcode: 0x2
 - **sltsi** rA, rB, simm16
 - Opcode: 0x3
 - **sgtui** rA, rB, imm16
 - Opcode: 0x4
 - **sgtsi** rA, rB, simm16
 - Opcode: 0x5
 - **muli** rA, rB, imm16
 - Opcode: 0x6
 - **andi** rA, rB, imm16
 - Opcode: 0x7
 - **orri** rA, rB, imm16
 - Opcode: 0x8
 - **xori** rA, rB, imm16
 - Opcode: 0x9
 - **nori** rA, rB, imm16
 - Opcode: 0xa
 - **lsli** rA, rB, imm16
 - Opcode: 0xb
 - **lsri** rA, rB, imm16
 - Opcode: 0xc
 - **asri** rA, rB, imm16
 - Opcode: 0xd
 - **addsi** rA, pc, simm16
 - Opcode: 0xe
 - **cpyhi** rA, imm16
 - Opcode: 0xf
- Opcode Group: 0b0010
 - Encoding: 0010 aaaa bbbb oooo iiii iiii iiii iiii
 - a : rA
 - b : rB

- o : opcode
- i : 16-bit immediate
- Instructions:
 - **bne** rA, rB, offset16
 - Opcode: 0x0
 - **beq** rA, rB, offset16
 - Opcode: 0x1
 - **bltu** rA, rB, offset16
 - Opcode: 0x2
 - **bgeu** rA, rB, offset16
 - Opcode: 0x3
 - **bleu** rA, rB, offset16
 - Opcode: 0x4
 - **bgtu** rA, rB, offset16
 - Opcode: 0x5
 - **blts** rA, rB, offset16
 - Opcode: 0x6
 - **bges** rA, rB, offset16
 - Opcode: 0x7
 - **bles** rA, rB, offset16
 - Opcode: 0x8
 - **bgts** rA, rB, offset16
 - Opcode: 0x9
- Opcode Group: 0b0011
 - Encoding: 0011 aaaa bbbb cccc 0000 0000 0000 0000
 - a : rA
 - b : rB
 - c : rC
 - o : opcode
 - Instructions:
 - **jne** rA, rB, rC
 - Opcode: 0x0
 - **jeq** rA, rB, rC
 - Opcode: 0x1
 - **jltu** rA, rB, rC
 - Opcode: 0x2
 - **jgeu** rA, rB, rC
 - Opcode: 0x3
 - **jleu** rA, rB, rC
 - Opcode: 0x4
 - **jgtu** rA, rB, rC
 - Opcode: 0x5
 - **jlt** rA, rB, rC
 - Opcode: 0x6
 - **jges** rA, rB, rC
 - Opcode: 0x7
 - **jles** rA, rB, rC
 - Opcode: 0x8
 - **jgts** rA, rB, rC
 - Opcode: 0x9

- Opcode Group: 0b0100

- Encoding: 0100 aaaa bbbb cccc 0000 0000 0000 0000

- a : rA
 - b : rB
 - c : rC
 - o : opcode

- Instructions:

- **cne** rA, rB, rC
 - Opcode: 0x0
 - **ceq** rA, rB, rC
 - Opcode: 0x1
 - **cltu** rA, rB, rC
 - Opcode: 0x2
 - **cgeu** rA, rB, rC
 - Opcode: 0x3
 - **cleu** rA, rB, rC
 - Opcode: 0x4
 - **cgts** rA, rB, rC
 - Opcode: 0x5
 - **clts** rA, rB, rC
 - Opcode: 0x6
 - **cges** rA, rB, rC
 - Opcode: 0x7
 - **cles** rA, rB, rC
 - Opcode: 0x8
 - **cgts** rA, rB, rC
 - Opcode: 0x9

- Opcode Group: 0b0101

- Encoding: 0101 aaaa bbbb cccc iiii iiii iiii 0000

- a : rA
 - b : rB
 - c : rC
 - i : sign-extended 12-bit immediate
 - o : opcode

- Instructions:

- **ldr** rA, [rB, rC]
 - Opcode: 0b0000
 - **ldh** rA, [rB, rC]
 - Opcode: 0b0001
 - **ldsh** rA, [rB, rC]
 - Opcode: 0b0010
 - **ldb** rA, [rB, rC]
 - Opcode: 0b0011
 - **ldsb** rA, [rB, rC]
 - Opcode: 0b0100
 - **str** rA, [rB, rC]
 - Opcode: 0b0101
 - **sth** rA, [rB, rC]

- Opcode: 0b0110
 - **stb** rA, [rB, rC]
 - Opcode: 0b0111
 - **ldri** rA, [rB, simm12]
 - Opcode: 0b1000
 - **ldhi** rA, [rB, simm12]
 - Opcode: 0b1001
 - **ldshi** rA, [rB, simm12]
 - Opcode: 0b1010
 - **ldbi** rA, [rB, simm12]
 - Opcode: 0b1011
 - **ldsbi** rA, [rB, simm12]
 - Opcode: 0b1100
 - **stri** rA, [rB, simm12]
 - Opcode: 0b1101
 - **sthi** rA, [rB, simm12]
 - Opcode: 0b1110
 - **stbi** rA, [rB, simm12]
 - Opcode: 0b1111
- Opcode Group: 0b0110
 - Encoding: 0110 aaaa bbbb cccc 0000 0000 0000 0000
 - a : rA
 - b : rB
 - c : rC
 - o ; opcode
 - Instructions:
 - **ei**
 - Note: Enable interrupts
 - Opcode: 0x0
 - **di**
 - Note: Disable interrupts
 - Opcode: 0x1
 - **cpy** ireta, rA
 - Opcode: 0x2
 - **cpy** rA, ireta
 - Opcode: 0x3
 - **cpy** idsta, rA
 - Opcode: 0x4
 - **cpy** rA, idsta
 - Opcode: 0x5
 - **reti**
 - Note: Enable interrupts and change the program counter to the value contained in ireta
 - Opcode: 0x6
 - Pseudo Instructions:
 - **inv** rA, rB
 - Encoded as nor rA, rB, zero
 - **invi** rA, imm16
 - Encoded as nori rA, zero, imm16

- **cpy** rA, rB
 - Encoded as `add rA, rB, zero`
- **cpy** rA, pc
 - Encoded as `addsi rA, pc, 0`
- **cpyi** rA, imm16
 - Encoded as `addi rA, zero, imm16`
- **cpya** rA, imm32
 - Copy absolute (32-bit immediate)
 - Encoded as
 - `addi rA, zero, (imm32 & 0xffff)`
 - `cpyhi rA, (imm32 >> 16)`
- **bra** simm16
 - Unconditional relative branch
 - Encoded as `beq zero, zero, simm16`
- **jmp** rC
 - Unconditional jump to address in register
 - Encoded as `jeq zero, zero, rC`
- **call** rC
 - Unconditional call to address in register
 - Encoded as `ceq zero, zero, rC`
- **jmpa** imm32
 - Jump absolute (to directly encoded address)
 - Encoded as
 - `cpya temp, imm32`
 - `jmp temp`
- **calla** imm32
 - Call absolute (to directly encoded address)
 - Encoded as
 - `cpya temp, imm32`
 - `call temp`
- **jmpane** rA, rB, imm32
 - Conditional jump absolute (to directly encoded address)
 - Encoded as
 - `cpya temp, imm32`
 - `jne rA, rB temp`
- **jmpaeq** rA, rB, imm32
 - Conditional jump absolute (to directly encoded address)
 - Encoded as
 - `cpya temp, imm32`
 - `jeq rA, rB temp`
- **callane** rA, rB, imm32
 - Conditional call absolute (to directly encoded address)
 - Encoded as
 - `cpya temp, imm32`
 - `callne rA, rB, temp`
- **callaeq** rA, rB, imm32
 - Conditional call absolute (to directly encoded address)
 - Encoded as
 - `cpya temp, imm32`
 - `calleq rA, rB, temp`

- **inc rA**
 - Encoded as `addi rA, rA, 1`
- **dec rA**
 - Encoded as `subi rA, rA, 1`
- **alu_op_three_regs rA, rB**
 - Encoded as `alu_op_three_regs rA, rA, rB`
- **alu_op_two_regs_one_immediate rA, imm16**
 - Encoded as `alu_op_two_regs_one_immediate rA, rA, imm16`