

Frost64 Utility CPU

Andrew Clark

July 2, 2020

1 Table of Contents

Contents

1	Table of Contents	1
2	Registers, Main Widths, etc.	2
3	Instructions	3
3.1	Group 0 Instructions	3
3.2	Group 1 Instructions	4
3.3	Group 2 Instructions	4
3.4	Group 3 Instructions	5
3.5	Group 4 Instructions	6
3.6	Group 5 Instructions	6
3.7	Group 6 Instructions	6
3.8	Group 7 Instructions	6

2 Registers, Main Widths, etc.

- The main width of the processor is 32-bit, contrary to what the name of the processor suggests. Addresses are 32-bit.
- General Purpose Registers (32-bit) There are sixteen general-purpose registers:
 - zero: index 0x0, always zero
 - ra: index 0x1
 - rb: index 0x2
 - rc: index 0x3
 - rd: index 0x4
 - re: index 0x5
 - rf: index 0x6
 - rg: index 0x7
 - rh: index 0x8
 - ri: index 0x9
 - rj: index 0xa
 - rk: index 0xb
 - rl: index 0xc
 - lr: index 0xd, link register
 - fp: index 0xe, frame pointer
 - sp: index 0xf, stack pointer (one for interrupts mode, one for user mode)

There are a few special-purpose registers:

- pc: index 0x0, program counter
 - * Note that writes into this register forcibly have the LSB set to 0b0.
- ira: index 0x1, interrupt return address
- ida: index 0x2, interrupt destination address
- inp: index 0x3, interrupt pin (which interrupt pin was asserted)
- ie: index 0x4, interrupt enable
- usp: index 0x5, mirror of user-mode stack pointer
- isp: index 0x6, mirror of interrupts-mode stack pointer (writes to this register are ignored when in user-mode)

3 Instructions

3.1 Group 0 Instructions

- Encoding (non-pre): 0ooo aaaa iiii iiii
 - o: opcode (non-0b000)
 - a: general-purpose register rA
 - i: 8-bit unsigned immediate imm8 or 8-bit signed immediate simm8
- Encoding (pre): 0000 iiii iiii iiii
 - i: 12-bit partial immediate, imm12
- Instruction List:
 1. pre imm12
 - This instruction sets the immediate value of the following instruction to `imm12 << 12 | next_instr_immediate`.
 - Multiples of this instruction can be chained together to create even larger immediates.
 - For instructions that use a sign-extended immediate, the final immediate will be sign extended starting from whatever the most significant bit of the final immediate is. For example, `pre imm12; addsi rA, simm8` will sign extend the 20-bit value, `(imm12 << 12) | simm8`, to 32-bit before performing the operation.
 - This instruction, while in effect, cannot be interrupted.
 - When an instruction that doesn't use an immediate follows this one, `pre` acts as a nop.
 2. cpyi rA, imm8
 - This instruction sets rA to the immediate value.
 3. addsi rA, simm8
 - This instruction writes `rA + to_s32(simm8)` into rA.
 4. sltui rA, imm8
 5. sltsi rA, simm8
 6. addsi rA, pc, simm8
 7. bz rA, simm8
 - Relative branch if rA is *zero*.
 - Note that the branch offset is treated as `simm8 << 1`.
 8. bnz rA, simm8
 - Relative branch if rA is *non-zero*.
 - Note that the branch offset is treated as `simm8 << 1`.

3.2 Group 1 Instructions

- Encoding: 1000 aaaa oooi iiii
 - a: general-purpose register rA or special-purpose register sA
 - o: opcode
 - i: 5-bit unsigned immediate imm5 or 5-bit signed immediate simm5
- Instruction List:
 1. andi rA, imm5
 2. ori rA, imm5
 3. xorsi rA, simm5
 4. xorsi sA, simm5
 5. lsli rA, imm5
 6. lsri rA, imm5
 7. asri rA, imm5
 8. *Reserved for future expansion.*

3.3 Group 2 Instructions

- Encoding: 1001 aaaa bbbb oooo
 - a: general-purpose register rA
 - b: general-purpose register rB
 - o: opcode
- Instruction List:
 1. add rA, rB
 2. sub rA, rB
 3. sltu rA, rB
 4. slts rA, rB
 5. and rA, rB
 6. or rA, rB
 7. xor rA, rB
 8. lsl rA, rB
 9. lsr rA, rB
 10. asr rA, rB
 11. mul rA, rB
 12. udiv rA, rB
 13. sdiv rA, rB
 14. add rA, rB, pc
 15. *Reserved for future expansion.*
 16. *Reserved for future expansion.*

3.4 Group 3 Instructions

- Encoding: 1010 aaaa bbbb oooo
 - a: general-purpose register *rA* or special-purpose register *sA*
 - b: general-purpose register *rB* or special-purpose register *sB*
 - o: opcode
- Instruction List:
 1. *ldb rA, [rB]*
 2. *ldsb rA, [rB]*
 3. *stb rA, [rB]*
 4. *ldh rA, [rB]*
 5. *ldsh rA, [rB]*
 6. *sth rA, [rB]*
 7. *ldr sA, [rB, sp]*
 8. *str sA, [rB, sp]*
 9. *cpy rA, sB*
 - Copy *sB* to *rA*
 10. *cpy sA, rB*
 - Copy *rB* to *sA*
 11. *cpy sA, sB*
 - Copy *sB* to *sA*
 12. *reti*
 - Return from an interrupt, enabling interrupts (by setting the low bit of *ie* to 0b1) and jumping to the program counter value at *ira*.
 13. *zeb rA, rB*
 - Zero extend the low 8 bits of *rB* and write the result into *rA*.
 14. *zeh rA, rB*
 - Zero extend the low 16 bits of *rB* and write the result into *rA*.
 15. *seb rA, rB*
 - Sign extend the low 8 bits of *rB* and write the result into *rA*.
 16. *seh rA, rB*
 - Sign extend the low 16 bits of *rB* and write the result into *rA*.

3.5 Group 4 Instructions

- Encoding: 1011 *iiii* *iiii* *iiii*
 - *i*: 12-bit signed immediate, *simm12*
- Instruction List:
 - l. bl *simm12*
 - This instruction sets *lr* to *pc* + 2 and jumps to the address *pc* + (*to_s32(simm12) << 1*).

3.6 Group 5 Instructions

- Encoding: 1100 *aaaa* *bbbb* *cccc*
 - *a*: general-purpose register *rA*
 - *b*: general-purpose register *rB*
 - *c*: general-purpose register *rC*
- Instruction List:
 - l. ldr *rA*, [*rB*, *rC*]

3.7 Group 6 Instructions

- Encoding: 1101 *aaaa* *bbbb* *cccc*
 - *a*: general-purpose register *rA*
 - *b*: general-purpose register *rB*
 - *c*: general-purpose register *rC*
- Instruction List:
 - l. str *rA*, [*rB*, *rC*]

3.8 Group 7 Instructions

- Encoding: 1110 *aaaa* *bbbb* *cccc*
 - *a*: general-purpose register *rA*
 - *b*: general-purpose register *rB*
 - *c*: general-purpose register *rC*
- Instruction List:
 - l. add *rA*, *rB*, *rC*