Frost64 Utility CPU

Andrew Clark

July 1, 2020

1 Table of Contents

Contents

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
  - `u0`: index `0x1`
  - `u1`: index `0x2`
  - `u2`: index `0x3`
  - `u3`: index `0x4`
  - `u4`: index `0x5`
  - `u5`: index `0x6`
  - `u6`: index `0x7`
  - `u7`: index `0x8`
  - `u8`: index `0x9`
  - `u9`: index `0xa`
  - `u10`: index `0xb`
  - `u11`: index `0xc`
  - `lr`: index `0xd`, link register
  - `fp`: index `0xe`, frame pointer
  - `sp`: index `0xf`, stack pointer

  There are a few special-purpose registers:

  - `pc`: index `0x0`, program counter
  - `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

3 Instructions

3.1 Group 0 Instructions

- Encoding: `0ooo aaaa iiii iiii`
  - `o`: opcode
  - `a`: general-purpose register `rA`
  - `i`: 8-bit unsigned immediate `imm8` or 8-bit signed immediate `simm8`

- Instruction List:

  1. `cpyi rA, imm8`
     - This instruction sets `rA` to the immediate value.
  2. `cpyi1 rA, imm8`
     - This instruction sets byte 1 of `rA` to the immediate value.
  3. `cpyi2 rA, imm8`
     - This instruction sets byte 2 of `rA` to the immediate value.
  4. `cpyi3 rA, imm8`
     - This instruction sets byte 3 (msb) of `rA` to the immediate value.
  5. `addsi pc, rA, simm8`
     - This instruction writes `(rA + to_s32(simm8)) << 1` into `pc`
  6. `addsi rA, pc, simm8`
     - This instruction writes `pc + to_s32(imm8)` into `rA`
  7. `bz rA, simm8`
     - Relative branch if `rA` is *zero*.
  8. `bnz rA, simm8`
     - Relative branch if `rA` is *non-zero*.

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. `sltui rA, imm5`
  2. `sltsi rA, simm5`
  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. `orr 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, fp*
  15. *add rA, rB, sp*
  16. *add rA, rB, pc*

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, rB`
   - Copy `rB` to `rA`
10. `cpy rA, sB`
    - Copy `sB` to `rA`
11. `cpy sA, rB`
    - Copy `rB` 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:
  1. `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:
  1. `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:
  1. `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:
  1. `add rA, rB, rC`