RV32I/RV32E: ALU Behavior by Instruction

Sunday, October 27, 2024 12:17 PM

NOTE: Overflow checking is done with a separate instruction as RISC-V does not have a status register.
NOTE: All instructions are done at the instruction word size unless otherwise noted.
NOTE: All operands are treated as signed unless otherwise noted.
NOTE: While RV32I and RV32E instruction implementation are the same, RV32E only has 16 registers instead of the typical 32 registers that the base integer ISA has.

- Immediate Instructions *(note that all immediate values are sign extended unless otherwise noted)*
    - **ADDI**: rd <= rs1 + *imm*
    - **SLTI**: rd <= (rs1 < *imm*)?1'b1:1'b0
    - **SLTIU**: rd <= (rs1 < *imm*)?1'b1:1'b0 (*NOTE: Sign extended, but not treated as signed)*
    - **ANDI**: rd <= rs1 & *imm*
    - **ORI**: rd <= rs1 | *imm*
    - **XORI**: rd <= rs1 ^ *imm*
    - Pseudoinstructions
        - **NOT**: rd <= rs1 ^ -1 (12'hFFF, sign extended) *(NOTE: Assembler pseudoinstruction)*
- Shift Instructions
    - **SLLI**: rd <= rs1 << *imm*[4:0], *imm*[30] == 0
    - **SRLI**: rd <= rs1 >> *imm*[4:0], *imm*[30] == 0 (Logical shift, do not sign extend)
    - **SRAI**: rd <= rs1 >> *imm*[4:0], *imm*[30] == 1
    - **SLL**: rd <= rs1 << rs2[4:0]
    - **SRL**: rd <= rs1 >> rs2[4:0] (Logical shift, do not sign extend)
    - **SRA**: rd <= rs1 >> rs2[4:0] (Arithmetic shift, sign extend)
- Upper Immediate Instructions
    - **LUI**: rd <= {*imm*[31:12], 12'h000}
    - **AUIPC**: rd <= pc + {*imm*[31:12], 12'h000}
- Register-Register Instructions
    - **ADD**: rd <= rs2 + rs1
    - **SUB**: rd <= rs2 - rs1
    - **SLT**: rd <= (rs1 < rs2)?1'b1:1'b0
    - **SLTU**: rd <= (rs1 < rs2)?1'b1:1'b0 *(NOTE: Values treated as unsigned)*
    - Pseudoinstructions
        - **SNEZ**: rd <= (x0 < rs2)?1'b1:1'b0 *(NOTE: Assembler pseudoinstruction)*
        - **NOP**: x0 <= x0 + 12'h000 *(NOTE: Assembler pseudoinstruction)*
- Control Transfer Instructions
    - Exceptions raised if:
        - Destination address is not aligned to a 4-byte boundary (Instruction-Address-Misaligned) on the branch instruction
        - The target causes an access fault (access-fault exception) or a page fault (page-fault exception) on the target instruction
    - Unconditional Jumps
        - **JAL**: rd <= pc + 4, pc <= pc + *offset*[20:1] (*NOTE: Offset is signed and sign extended*)
        - **JALR**: rd <= pc + 4, pc = rs1 + *offset*[11:0]; pc[0] `= 1'b0; (*NOTE: Offset is signed and sign extended*)
        - Return-address stack prediction hints
            - (rd != x1 or x5) and (rd != x1 or x5): Do nothing
            - (rd != x1 or x5) and (rd == x1 or x5): Pop
            - (rd == x1 or x5) and (rd != x1 or x5): Push

            - (rd == x1 or x5) and (rd == x1 or x5) and (rd != rs1): Pop, then push
            - (rd == x1 or x5) and (rd == x1 or x5) and (rd == rs1): Push
    - Conditional Branches (revise to match MSP430 behavior)
        - **BEQ**: if (rs2 - rs1 == 0) pc <= pc + *offset*[12:1]
        - **BNE**: if (rs2 - rs1 != 0) pc <= pc + *offset*[12:1]
        - **BLT**: if (rs2 - rs1 > 0) pc <= pc + *offset*[12:1]
        - **BLTU**: {c, res} <= rs2-rs1; if (!c && res > 0) pc <= pc + *offset*[12:1]
        - **BGE**: if (rs1 - rs2 >= 0) pc <= pc + *offset*[12:1]
        - **BGEU**: {c, res} <= rs1-rs2; if ((!c && res > 0) || (res == 0) pc <= pc + *offset*[12:1]
        - Pseudoinstructions
            - **BLE**: if (rs2 - rs1 >= 0) pc <= pc + *offset*[12:1] *(NOTE: Assembler pseudoinstruction)*
            - **BLEU**: {c, res} <= rs2-rs1; if ((!c && res > 0) || (res == 0) pc <= pc + *offset*[12:1] *(NOTE: Assembler pseudoinstruction)*
            - **BGT**: if (rs1 - rs2 > 0) pc <= pc + *offset*[12:1] *(NOTE: Assembler pseudoinstruction)*
            - **BGTU**: {c, res} <= rs1-rs2; if (!c && res > 0) pc <= pc + *offset*[12:1] *(NOTE: Assembler pseudoinstruction)*

- ALU Flag behaviors
    - C: set if carry out on operation
    - N: set if result is negative
    - V: set if positive and positive results in negative (and reverse)
- Load and Store Instructions
    - **LW**: rd <= *imm*[11:0](rs1) (32-bits signed)
    - **LH**: rd <= *imm*[11:0](rs1) (16-bits signed)
    - **LHU**: rd <= *imm*[11:0](rs1) (16-bits unsigned)
    - **LB**: rd <= *imm*[11:0](rs1) (8-bits signed)
    - **LBU**: rd <= *imm*[11:0](rs1) (8-bits unsigned)
    - **SW**: *imm*[11:0](rs1) <= rs2 (32-bits signed)
    - **SH**: *imm*[11:0](rs1) <= rs2 (16-bits signed)
    - **SB**: *imm*[11:0](rs1) <= rs2 (8-bits signed)
- Memory Ordering Instructions
    - **FENCE**: Pauses all memory (and memory mapped I/O) accesses until previous load/store operations done by the processor are complete based on specified flags.
- Environment (System) Call and Breakpoint Instructions
    - **ECALL**: Makes service request to execution environment (such as an operating system). Not fully implemented in the unprivileged architecture.
    - **EBREAK**: Cedes control to a debugging environment. Handled by flag to control register. Must be cleared from an external signal. Not fully implemented in the unprivileged architecture.
- Hint Instructions
    - These instructions use existing instructions with rd == x0. However, as this is a real-time processor without speculative execution (as of present) branch hints are not implemented.