**Risc-V Emulator Documentation  
  
A table with numbers and letters

Description automatically generated  
instruction types**

**R-Type:** 3 register operations. No immediate.

**I-Type:** Immediate operations.

**S-Type:** store operations.

**B-Type:** conditional branch instructions

**U-Type**: upper immediate instructions, used for large immediates.

**J-Type**: general jump instructions.

**Instruction list:**

* ***Lui:***Load upper immediate to rd, like the ‘mov’ command in x86.   
  U-Type
* ***Auipc:***  
  Add upper immediate and to pc and store in rd.   
  PC does not get changed.   
  Used to build pc-relative addresses.  
  U-Type
* ***Jal:***Jump and link  
  Save pc + 4 in rd.  
  Add immediate to pc and store in pc.  
  J-Type  
  Bits are not in order to reduce number of multiplexers needed

\***multiplexer** – a device that can receive multiple input signals and synthesize a single output signal in a recoverable manner for each input signal

* ***Jalr:***

Jump and link register  
  
  
save pc + 4 in rd (should use x0 if result is not useful, like in ret from subroutine).

Add immediate to rs1, set LSB (least significant bit) to 0 add to pc.  
I-Type

* ***Cjumps:***
  + **BNE** – JNE
  + **BEQ** – JE
  + **BLT** – JL
  + **BGE** – JGE
  + **BLTU** – JB (unsigned)
  + **BGEU** – JAE (unsigned)

B-Type

* ***Alui***arithmetic and logical immediate instructions:
* **ADDI**:

Addi immediate

* **Slti**:  
  set less than immediate – places 1 in rd if rs1 < sign-extended immediate. Else 0. \*used with bne in loops, instead of blt and etc.
* **Sltiu**:  
  set less than immediate unsigned – places 1 in rd if rs1 < immediate. Else 0.  
  \*used with bne in loops, instead of blt and etc.
* **XORI**:  
  xori immediate
* **ORI**:

Ori immediate

* **Andi**:

Andi immediate

* **Slli**:

Shift left

* **Srli**:

Shift right

* **Srai**:

Arithmetic right shift

(there is not difference between arithmetic shift left and regular shift left)

I-Type

* ***ALU*** arithmetic and logical instructions:
* **ADD**:

Add

* **Slt**:  
  set less than – places 1 in rd if rs1 < rs2. Else 0.   
  \*used with bne in loops, instead of blt and etc.
* **Sltu**:  
  set less than unsigned – places 1 in rd if rs1 < rs2 (unsigned). Else 0.  
  \*used with bne in loops, instead of blt and etc.
* **XOR**:  
  xori
* **OR**:

Ori

* **And**:

And

* **Sll**:

Shift left

* **Srl**:

Shift right

* **Sra**:

Arithmetic right shift

(there is not difference between arithmetic shift left and regular shift left)

R-Type

* ***Load***  
  load from memory:  
    
  \*all extend to 32-bits
* **LB**:

Load byte

* **LBU:**  
  Load byte unsigned
* **LH**:  
  load word
* **LHU:**  
  load word unsigned
* **LW**:  
  load double word.

I-Type

* ***Store***  
  store in memory:  
  store from the lower bits of rs2 to memory
* **LB**:

Load byte

* **LBU:**  
  Load byte unsigned
* **LH**:  
  load word
* **LHU:**  
  load word unsigned
* **LW**:  
  load double word.

S-Type

* ***Ecall***system syscalls. Vaguely defined by RISC-V purposefully.   
  \***A7** – syscall number  
  **\*a0-a5** – args (zeroed by os if not used)  
  **\*return value – a0**  
  syscall table:  
  https://jborza.com/post/2021-05-11-riscv-linux-syscalls/