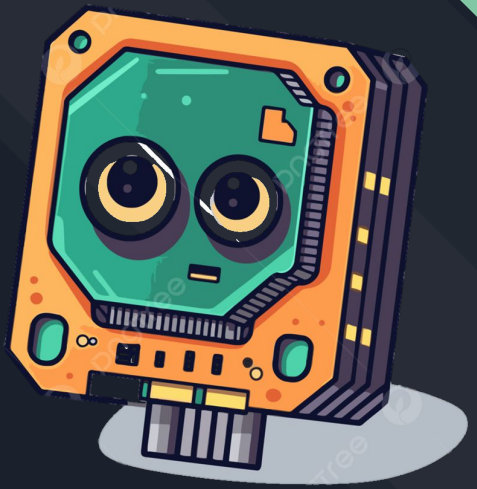
ISA Design Proposal: GiggleFlop



Will Lillis
Pranav Shekar

Summary

- General purpose
- 32 bit word size
- Three register banks
  - General Purpose
  - Floating Point
  - Status/Flags
- Clean sheet design
- Supported data types
  - Ints (unsigned/signed)
  - Floats
  - Bytes
- Additional special features

Parameters

- 32 registers
  - 1 status/flag register
  - 1 PC register
  - Split rest between float and general purpose registers
- 3 operands per instruction max
- Fetch paradigm - single instruction per word
- Unified instruction and data memory
- Byte addressing
- 12 bits in address range (4k words)

Instruction Types/Addressing Modes

- Control Instructions
  - Branch on flag
  - Conditional jump
    - Based on flag registers (equal/not equal, overflow, etc)
- Subroutine jumps/returns
  - Largely handle with calling conventions
  - Store PC in return register, jump indirect using return register to return
  - Designated general purpose register
- Halt instruction to exit
- Addressing modes supported
  - Register direct - simple operations
  - Register indirect base + address - good for arrays
- Load and store
  - Support 8, 16, and 32 bits

Sample Instruction Formats

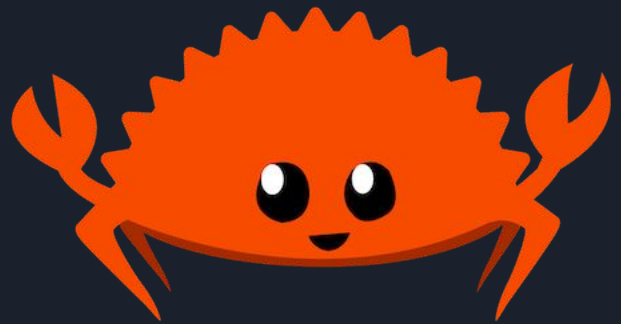
- Operations supported
  - Basic arithmetic operations for int/float
    - Add, subtract, multiply, divide, modulus
  - Basic bit operations
    - Bit shifting, XOR, AND, OR, etc.
  - Comparisons
- 4 Types
  - General operations
  - Signed integer operations
  - Unsigned integer operations
  - Floating point operations
- 0-3 operands
  - 5 bits to specify instruction
  - Max allowable width for immediates
- Ex. add integer
  - ADDI, R3, R1, R2
  - Instruction(ADDI), result register(R3), registers to add (R1/R2)

Special Features

- Branch prediction
- Configurable cache
  - Number of levels
  - Size
- Debugging capabilities
  - Breakpoints
  - Run/Step Execution
  - Serialization/ Deserialization of machine state for hot loading
  - “Watchvalues” if time allows

Project Management

- Version control and code storage via Git and Github
- Developed in Rust
  - egui crate for GUI
  - serde crate for saving/ reloading machine state
- Timeline as detailed in class



Questions?