Instruction Set

Register Architecture

- **R0-R15**: General purpose registers (16 x 16-bit)
- **PC**: Program Counter
- **STATUS**: Status flags (validation result, zero flag)
- **TBASE**: Tally table base address register

Instruction Summary Table

Mnemonic	Opcode	Format	Description
`ADD`	0000	R-type	Add two registers
`SUB`	0001	R-type	Subtract two registers
`AND`	0010	R-type	Bitwise AND
`OR`	0011	R-type	Bitwise OR
`XOR`	0100	R-type	Bitwise XOR
`NOT`	0101	R-type	Bitwise NOT
`LOAD`	0110	I-type	Load from memory to register
`STORE`	0111	I-type	Store register to memory
`LDI`	1000	I-type	Load immediate value
`NEWC`	1001	R-type	Register candidate ID (initialize tally row)
`NEWD`	1010	R-type	Register district ID (initialize district column)
`ADDT`	1011	R-type	Add `value` votes to `candID` from `distID` , update total
`OUTT`	1100	R-type	Output total tally for `candID` to 7-segment display
`FLIP`	1101	R-type	Flip all bits in register (tag validation)
`SWAP`	1110	R-type	Swap bit segments between registers (tag validation)
`SHIFT`	1111	R-type	Rotate left shift register (tag validation)
`BXOR`	0001	R-type	XOR all blocks together (tag validation)
`INSW`	0010	I-type	Read from switches to register
`OUTLED`	0011	I-type	Output register to LEDs
`OUT7SEG`	0100	I-type	Output register to 7-segment display
`CMP`	0101	R-type	Compare two registers, set STATUS flags
`JMP`	0110	J-type	Unconditional jump

Mnemonic	Opcode	Format	Description
`JEQ`	0111	J-type	Jump if equal (STATUS.Z = 1)
`JNE`	1000	J-type	Jump if not equal
`JVAL`	1001	J-type	Jump if tag validation passed
`NOP`	1010	—	No operation (for hazard handling)
`HALT`	1011	—	Stop processor execution (halts pipeline)

Instruction Formats

R-Type

Used for arithmetic, logic, validation, tallying, and display.

```
[4-bit opcode][4-bit rd][4-bit rs1][4-bit rs2/imm4]
```

I-Type

Used for memory access and I/O operations.

```
[4-bit opcode][4-bit rd][8-bit immediate/address]
```

J-Type

Used for control flow.

```
[4-bit opcode][12-bit address]
```

Instruction Details

Basic ALU Operations

`ADD rd, rs1, rs2`

- **Opcode**: `0000`
- **Format**: R-type
- **Description**: Adds rs1 and rs2, stores result in rd. Sets zero flag if result is 0.

`SUB rd, rs1, rs2`

- **Opcode**: `0001`
- **Format**: R-type

- **Description**: Subtracts rs2 from rs1, stores result in rd. Sets zero flag if result is 0.

`AND rd, rs1, rs2`

- **Opcode**: `0010`
- **Format**: R-type
- **Description**: Performs bitwise AND of rs1 and rs2, stores result in rd.

`OR rd, rs1, rs2`

- **Opcode**: `0011`
- **Format**: R-type
- **Description**: Performs bitwise OR of rs1 and rs2, stores result in rd.

`XOR rd, rs1, rs2`

- **Opcode**: `0100`
- **Format**: R-type
- **Description**: Performs bitwise XOR of rs1 and rs2, stores result in rd.

`NOT rd, rs1`

- **Opcode**: `0101`
- **Format**: R-type
- **Description**: Performs bitwise NOT of rs1, stores result in rd.

`CMP rs1, rs2`

- **Opcode**: `0101`
- **Format**: R-type
- **Description**: Compares rs1 and rs2, sets STATUS flags (zero flag if equal).

Memory and Data Operations

`LOAD rd, addr`

- **Opcode**: `0110`
- **Format**: I-type
- **Description**: Loads data from memory address into register rd. Executed in MEM stage.

`STORE rs, addr`

- **Opcode**: `0111`
- **Format**: I-type
- **Description**: Stores register rs to memory address. Executed in MEM stage.

`LDI rd, imm8`

- **Opcode**: `1000`
- **Format**: I-type
- **Description**: Loads 8-bit immediate value into register rd.

Election-Specific Operations

`NEWC rd, candID`

- **Opcode**: `1001`
- **Format**: R-type
- **Description**: Initializes a new candidate's tally row. Stores table offset in rd. Executed in WB stage.

`NEWD rd, distID`

- **Opcode**: `1010`
- **Format**: R-type
- **Description**: Initializes district column for vote tracking. Stores offset in rd. Executed in WB.

`ADDT distID, candID, value`

- **Opcode**: `1011`
- **Format**: R-type
- **Description**: Adds `value` votes from `distID` to `candID`. Updates district entry and total tally in WB.

`OUTT candID`

- **Opcode**: `1100`
- **Format**: R-type
- **Description**: Reads and displays total vote tally for a candidate on 7-segment. Output happens in MEM.

Tag Validation Operations (for Member 3)

`FLIP rd, rs`

- **Opcode**: `1101`
- **Format**: R-type
- **Description**: Flips all bits in register rs, stores result in rd. Part of tag validation process.

`SWAP rd, rs, pos, size`

- **Opcode**: `1110`
- **Format**: R-type
- **Description**: Swaps bit segments within register. pos = starting position, size = segment size.

`SHIFT rd, rs, amount`

- **Opcode**: `1111`
- **Format**: R-type
- **Description**: Rotate-left-shifts register rs by amount bits, stores in rd.

`BXOR rd, rs1, rs2`

- **Opcode**: `0001`

- **Format**: R-type
- **Description**: XORs all blocks together for final tag generation. Sets validation flag.

I/O Operations (for FPGA - Member 5)

`INSW rd`

- **Opcode**: `0010`
- **Format**: I-type
- **Description**: Reads 16-bit value from switches into register rd.

`OUTLED rs`

- **Opcode**: `0011`
- **Format**: I-type
- **Description**: Outputs register rs to LED array for debugging/status.

`OUT7SEG rs`

- **Opcode**: `0100`
- **Format**: I-type
- **Description**: Outputs register rs to 7-segment display for vote totals.

Control Flow Operations

`JMP addr`

- **Opcode**: `0110`
- **Format**: J-type
- **Description**: Unconditional jump to address. Updates PC in EX stage.

`JEQ addr`

- **Opcode**: `0111`
- **Format**: J-type
- **Description**: Jump to address if STATUS.Z flag is set (equal comparison result).

`JNE addr`

- **Opcode**: `1000`
- **Format**: J-type
- **Description**: Jump to address if STATUS.Z flag is clear (not equal).

`JVAL addr`

- **Opcode**: `1001`
- **Format**: J-type
- **Description**: Jump to address if tag validation flag is set (valid tag).

System Operations

`NOP`

- **Opcode**: `1010`
- **Format**: —
- **Description**: No operation. Used to resolve hazards or insert delays.

`HALT`

- **Opcode**: `1011`
- **Format**: —
- **Description**: Halts processor. Ends simulation or FPGA run.

Implementation Notes for Team Members

For Member 2 (Datapath Design):

- ALU needs to support: ADD, SUB, AND, OR, XOR, NOT operations
- Register file: 16 registers, 16-bit wide
- Memory interface for LOAD/STORE operations
- Special registers: PC, STATUS, TBASE

For Member 3 (Tag Validation Logic):

- Implement FLIP, SWAP, SHIFT, BXOR as separate modules
- SWAP operation needs bit position and size parameters
- BXOR sets validation flag in STATUS register
- Make operations configurable for different tag sizes

For Member 4 (Testing):

- Test cases should use basic ALU ops to build complex operations
- Create records with known tags using tag generation sequence
- Test different record sizes by adjusting bit extraction masks

For Member 5 (FPGA):

- INSW maps to 16 switches on FPGA board
- OUT7SEG drives 7-segment display
- OUTLED drives LED array
- Button inputs can be read via additional I/O instructions if needed