# microRISC

Andrew Schomber

December 12, 2025

# Table of Contents

# Architecture

## Registers

### General Purpose Registers

The general purpose registers are used to store data and perform arithmetic operations. They are named R0-R29 and are 32 bits wide. R30 is reserved for the stack pointer (SP) and R31 is reserved for the comparison register (CMP).

| Register | Binary Representation | Register | Binary Representation |
|---|---|---|---|
| R0 (ZR) | 00000 | R1 | 00001 |
| R2 | 00010 | R3 | 00011 |
| R4 | 00100 | R5 | 00101 |
| R6 | 00110 | R7 | 00111 |
| R8 | 01000 | R9 | 01001 |
| R10 | 01010 | R11 | 01011 |
| R12 | 01100 | R13 | 01101 |
| R14 | 01110 | R15 | 01111 |
| R16 | 10000 | R17 | 10001 |
| R18 | 10010 | R19 | 10011 |
| R20 | 10100 | R21 | 10101 |
| R22 | 10110 | R23 | 10111 |
| R24 | 11000 | R25 | 11001 |
| R26 | 11010 | R27 | 11011 |
| R28 | 11100 | R29 | 11101 |
| R30 (CMP) | 11110 | R31 (LR) | 11111 |

### Program Counter (PC)

The program counter register keeps track of the current instruction being executed. It is automatically incremented after each instruction is executed. It can not be directly accessed or modified by the programmer.

### Zero Register (ZR)

The zero register is a special-purpose register that always contains the value zero. It is used to simplify certain arithmetic operations and comparisons. You can reference it using the ZR keyword, or R0.

| Register | Binary Representation |
|---|---|
| ZR | 00000 |

### CMP Register

The CMP register is used to store the result of a comparison operation. It is set by the CMP instruction, which subtracts the second operand from the

first operand and sets the CMP register based on the result. Reference it using the CMP keyword (its an operation and a register), or R30.

| Register | Binary Representation |
|---|---|
| CMP | 111110 |

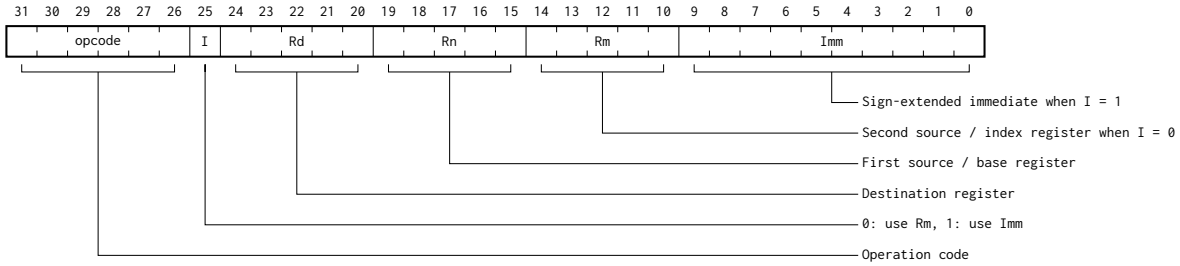
**Link Register (LR)**

The link register is used to store the return address of a function call. When a function is called, the address of the next instruction is stored in the link register. When the function returns, the program counter is set to the value of the link register to resume execution. You can reference it using the LR keyword, or R31.

| Register | Binary Representation |
|---|---|
| LR | 11111 |

## Opcode Layout and Control Signals

Every instruction is 32 bits wide. The top 6 bits form the opcode and drive the main control signals. Bit 25 is the I flag, which selects between a register operand and an immediate field for ALU and memory instructions.




## Memory Layout

The microRISC architecture uses separate instruction and data memories. Both memories are word-addressed and store 32-bit words.

**Instruction Memory**

| Property | Value | Notes |
|---|---|---|
| Address width | 24 bits | Word-addressed; one word per address |
| Word width | 32 bits | Each instruction is 32 bits |

**Data Memory**

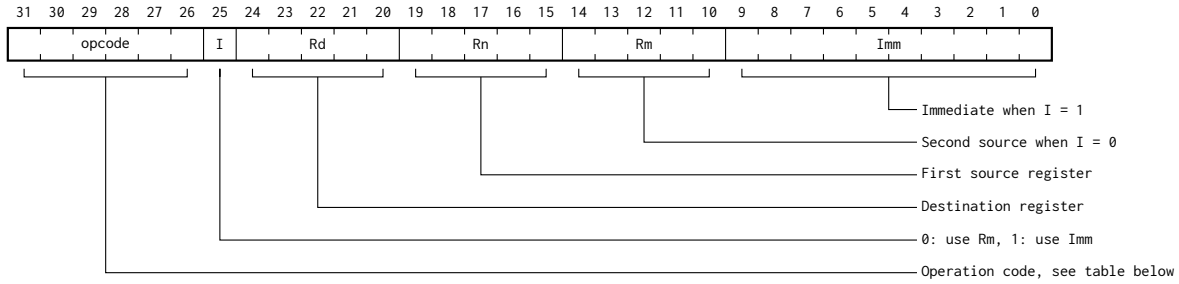
| Property | Value | Notes |
|---|---|---|
| Address width | 24 bits | Word-addressed; one word per address |
| Word width | 32 bits | Matches general-purpose register size |

The LDR and STR instructions operate on 32-bit words and use a base register plus an optional register or immediate offset to form the 24-bit data address.

# Instruction Set

## Arithmetic and Logical

The ALU instructions share one encoding each. The I flag selects whether the second operand comes from a register Rm or from an immediate value Imm.

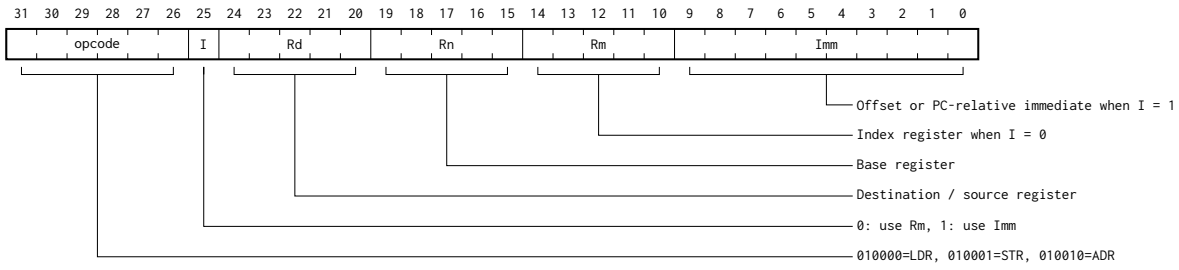



| Syntax | Opcode | I | Notes |
|---|---|---|---|
| ADD Rd, Rn, Rm / Imm | 000000 | 0 or 1 | Add Rn and Rm or Imm into Rd |
| SUB Rd, Rn, Rm / Imm | 000001 | 0 or 1 | Subtract Rm or Imm from Rn into Rd |
| MUL Rd, Rn, Rm / Imm | 000010 | 0 or 1 | Multiply Rn by Rm or Imm into Rd |
| DIV Rd, Rn, Rm / Imm | 000011 | 0 or 1 | Divide Rn by Rm or Imm into Rd |
| AND Rd, Rn, Rm / Imm | 000100 | 0 or 1 | Bitwise AND of Rn and Rm or Imm |
| ORR Rd, Rn, Rm / Imm | 000101 | 0 or 1 | Bitwise OR of Rn and Rm or Imm |
| XOR Rd, Rn, Rm / Imm | 000110 | 0 or 1 | Bitwise XOR of Rn and Rm or Imm |
| LSL Rd, Rn, Rm / Imm | 000111 | 0 or 1 | Logical shift left Rn by Rm or Imm |
| LSR Rd, Rn, Rm / Imm | 001000 | 0 or 1 | Logical shift right Rn by Rm or Imm |
| ASR Rd, Rn, Rm / Imm | 001001 | 0 or 1 | Arithmetic shift right Rn by Rm or Imm |

| Syntax | Opcode | Notes |
|---|---|---|
| NEG Rd | 001010 | Unary negate; encoded as an ALU op using the Rn field internally |

## Memory

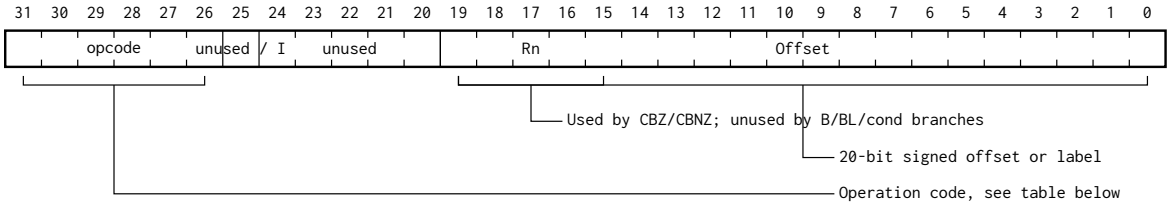
Load and store instructions use either a register index or an immediate offset from a base register. The I flag selects between the two.



| Syntax | Opcode | I | Notes |
|---|---|---|---|
| LDR Rd, Rn, Rm / Imm | 010000 | 0 or 1 | Load word from Rn + Rm or Imm into Rd |
| STR Rd, Rn, Rm / Imm | 010001 | 0 or 1 | Store word from Rd to Rn + Rm or Imm |
| ADR Rd, Label | 010010 | 1 | PC-relative address of label in Imm field |

## Branching

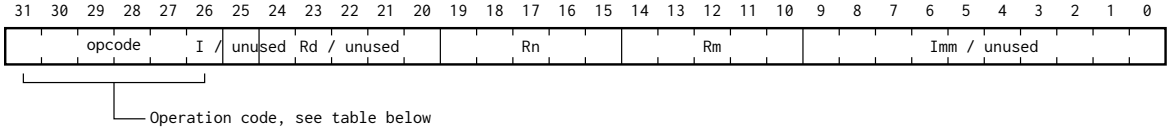
Branches use a 20-bit signed offset from the current PC. For conditional branches based on the CMP register, the CMP register is set by the CMP instruction. CBZ and CBNZ test a general-purpose register directly.




| Syntax | Opcode | Notes |
|---|---|---|
| B Label | 100000 | Unconditional branch with 20-bit signed offset |
| BL Label | 100001 | Branch with link; LR gets return address |
| BEQ Label | 100010 | Branch if CMP indicates equal |
| BNE Label | 100011 | Branch if CMP indicates not equal |
| BGT Label | 100100 | Branch if greater than |
| BLT Label | 100101 | Branch if less than |
| BGE Label | 100110 | Branch if greater or equal |
| BLE Label | 100111 | Branch if less or equal |

| Syntax | Opcode | Notes |
|---|---|---|
| CBZ Rn, Label | 110101 | Branch if Rn == 0 |
| CBNZ Rn, Label | 110110 | Branch if Rn != 0 |

## Other




| Syntax | Opcode | Notes |
|---|---|---|
| NOP | 110000 | No operation |
| RET | 110001 | Return to address in LR |

| Syntax | Opcode | Notes |
| --- | --- | --- |
| MOV Rd, Rn / Imm | 110010 | Move register or immediate value into Rd |
| CMP Rn, Rm / Imm | 110100 | Subtract Rm or Imm from Rn and update CMP register |