microRISC

Andrew Schomber

December 31, 2024

Table of Contents

Architecture .... **<3>**
→ Registers .... <3>
→ → General Purpose Registers .... <3>
→ → Program Counter (PC) .... <3>
→ → Zero Register (ZR) .... <3>
→ → CMP Register .... <3>
→ → Link Register (LR) .... <4>
→ Opcode Layout and Control Signals .... <4>
→ Memory Layout .... <4>

Instruction Set .... **<5>**
→ Arithmetic and Logical .... <5>
→ Memory .... <5>
→ Branching .... <5>
→ Other .... <6>

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 opcode is 6 bits wide. Each bit will be a dedicated control signal:

Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5
Control Signal 0	Control Signal 1	Control Signal 2	Control Signal 3	Control Signal 4	Control Signal 5

There are also other control signals implied by the opcode. The first two bits of the opcode are used to determine the type of instruction:

Bit 0-1	Instruction Type
00	Arithmetic/Logic
01	Memory
10	Branching
11	Special

Memory Layout

Data memory: 24-bit wide addresses, 32-bit wide data.

Direct-Mapped Cache: 1 KB (256 lines, 1 word per line, 4 bytes or 32 bits per word), 24-bit address:

Tag	Index	Offset	Valid Bit
16 bits	8 bits	0 bits	1 bit

Instruction Set

Arithmetic and Logical

The following arithmetic and logical operations are supported:

Syntax	Opcode	Rd	Rn	Rm	Unused
ADD Rd, Rn, Rm	000000	5 bits	5 bits	5 bits	11 bits
SUB Rd, Rn, Rm	000001	5 bits	5 bits	5 bits	11 bits
MUL Rd, Rn, Rm	000010	5 bits	5 bits	5 bits	11 bits
DIV Rd, Rn, Rm	000011	5 bits	5 bits	5 bits	11 bits
AND Rd, Rn, Rm	000100	5 bits	5 bits	5 bits	11 bits
ORR Rd, Rn, Rm	000101	5 bits	5 bits	5 bits	11 bits
XOR Rd, Rn, Rm	000110	5 bits	5 bits	5 bits	11 bits
LSL Rd, Rn, Rm	000111	5 bits	5 bits	5 bits	11 bits
LSR Rd, Rn, Rm	001000	5 bits	5 bits	5 bits	11 bits
ASR Rd, Rn, Rm	001001	5 bits	5 bits	5 bits	11 bits

Syntax	Opcode	Rd	Rn	Imm	Unused
ADDI Rd, Rn, Imm	001011	5 bits	5 bits	16 bits	0 bits
SUBI Rd, Rn, Imm	001100	5 bits	5 bits	16 bits	0 bits
MULI Rd, Rn, Imm	001101	5 bits	5 bits	16 bits	0 bits
DIVI Rd, Rn, Imm	001110	5 bits	5 bits	16 bits	0 bits

Syntax	Opcode	Rd	Unused
NEG Rd	001010	5 bits	21 bits

Memory

The following memory instructions are supported:

Syntax	Opcode	Rd	Rn	Rm	Unused
LDR Rd, [Rn, Rm]	010000	5 bits	5 bits	5 bits	11 bits
STR Rd, [Rn, Rm]	010001	5 bits	5 bits	5 bits	11 bits

Syntax	Opcode	Rd	Label	Unused
ADR Rd, Label	010010	5 bits	20 bits	1 bit

Branching

Labels are resolved to 20-bit addresses from the start of the program. The following branching instructions are supported:

Syntax	Opcode	Label	Unused
B Label	100000	20 bits	6 bits
BL Label	100001	20 bits	6 bits

Syntax	Opcode	Label	Unused
BEQ Label	100010	20 bits	6 bits
BNE Label	100011	20 bits	6 bits
BGT Label	100100	20 bits	6 bits
BLT Label	100101	20 bits	6 bits
BGE Label	100110	20 bits	6 bits
BLE Label	100111	20 bits	6 bits

Other

These are other instructions that don't fit under the existing categories:

Syntax	Opcode	Unused
NOP	110000	26 bits
RET	110001	26 bits

Syntax	Opcode	Rd	Rn	Unused
MOV Rd, Rn	110010	5 bits	5 bits	16 bits

Syntax	Opcode	Rd	Imm	Unused
MOVI Rd, Imm	110011	5 bits	16 bits	5 bits

Syntax	Opcode	Rd	Rn	Unused
CMP Rd, Rn	110100	5 bits	5 bits	16 bits

Syntax	Opcode	Rn	Label	Unused
CBZ Rn, Label	110101	5 bits	20 bits	1 bits
CBNZ Rn, Label	110110	5 bits	20 bits	1 bits

There is also support for single line comments and end of line comments:

```
// This is a single line comment
ADD R1, R2, R3 // This is an end of line comment
```