LiteRISC

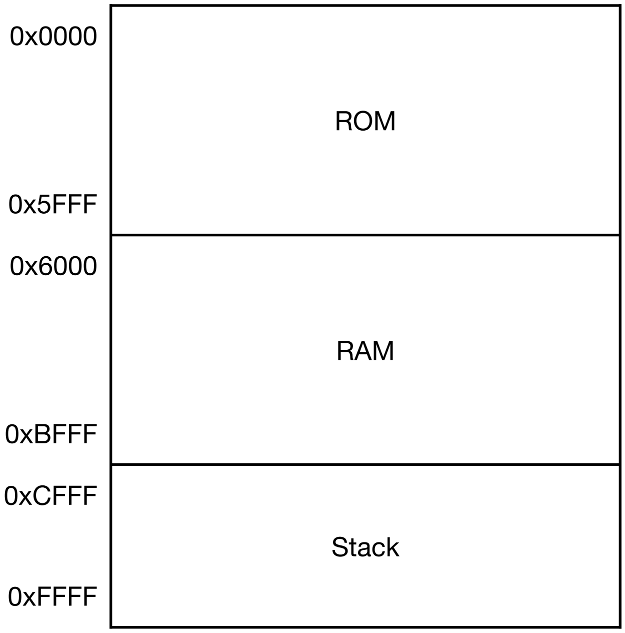
Instruction Set Architecture

March 2025
v1.0

Alan Bui
Department of Electrical
and Computer Engineering
The University of Texas at Austin

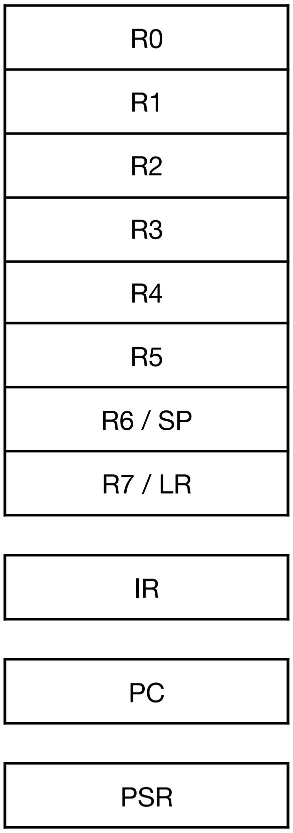
alan.bui@utexas.edu

Memory Map



16-bit Addressability

Registers



Register Formats

LiteRISC has ten 16-bit registers, where

- R0 – R5: General Purpose Registers
- R6: Stack Pointer Register
- R7: Link Register
- IR: Instruction Register
- PC: Program Counter
- PSR: Stores Condition Code Flags N, Z, and P

A Note on Procedure Call Standards.

LiteRISC follows a modified version of the ARM Architecture Procedure Call Standard (AAPCS) adjusted for 16-bit architectures, where

- R0 – R2: Procedure Input
- R0: Procedure Output
- R3 – R5 must be preserved between function calls.

LiteRISC

Instruction Reference

Unconditional Branch BRR

Opcode	Ext	Operands	Operation	Assembly Instruction
`0000 (0x0)`	–	imm12	PC ← PC + imm12	**`BRR Label`**

Machine Code

`0000`	**`imm12`**
15 – 12	*11 – 0*

Description
The BRR instruction

Condition Code Flags
The BRR instruction does not modify any Condition Code Flags.

Conditional Branch with Equality **BEQ/BNE**

Opcode	Ext	Operands	Operation	Assembly Instruction
`0001 (0x1)`	`0`	imm11	PC ← PC + imm11	**`BEQ Label`**
`0001 (0x1)`	`1`	imm11	PC ← PC + imm11	**`BNE Label`**

Machine Code

`0001`	`0`	**`imm11`**
15 – 12	*11*	*10 – 0*

`0001`	`1`	**`imm11`**
15 – 12	*11*	*10 – 0*

Description

The BEQ/BNE instructions

- `Ext == 00`: T
- `Ext == 01`: T

Condition Code Flags

The BEQ/BNE instructions do not modify any Condition Code Flags.

Conditional Branch with Comparison **BGT/BLT**

Opcode	Ext	Operands	Operation	Assembly Instruction
`0010 (0x2)`	`0`	imm11	PC ← PC + imm11	**`BEQ Label`**
`0010 (0x2)`	`1`	imm11	PC ← PC + imm11	**`BNE Label`**

Machine Code

`0010`	`0`	**`imm11`**
15 – 12	*11*	*10 – 0*

`0010`	`1`	**`imm11`**
15 – 12	*11*	*10 – 0*

Description

The BGT/BLT instructions

- `Ext == 00`: T
- `Ext == 01`: T

Condition Code Flags

The BGT/BLT instructions do not modify any Condition Code Flags.

Load Register LDR

Opcode	Ext	Operands	Operation	Assembly Instruction
`0011 (0x3)`	`00`	DR, SR	DR ← [SR]	**`LDR DR, [SR]`**
`0011 (0x3)`	`01`	DR, SR1, SR2	DR ← [SR1 + SR2]	**`LDR DR, [SR1, SR2]`**
`0011 (0x3)`	`10`	DR, SR, imm4	DR ← [SR + imm4]	**`LDR DR, [SR, #imm4]`**
`0011 (0x3)`	`11`	DR, [Label]	DR ← [Label]	**`LDR DR, =Label`**

Machine Code

0011	**00**	**-**	**SR**	**DR**
15 – 12	*11 – 10*	*9 – 6*	*5 – 3*	*2 – 0*

0011	**01**	**-**	**SR2**	**SR1**	**DR**
15 – 12	*11 – 10*	*9*	*8 – 6*	*5 – 3*	*2 – 0*

0011	**10**	**imm4**	**SR**	**DR**
15 – 12	*11 – 10*	*9 – 6*	*5 – 3*	*2 – 0*

0011	**11**	**imm7**	**DR**
15 – 12	*11 – 10*	*9 – 3*	*2 – 0*

Description

The LDR instruction loads a value from memory or a specified address into a register, depending on the Extension bits, where

- `Ext == 00`: T
- `Ext == 01`: T
- `Ext == 10`: T
- `Ext == 11`: T

Condition Code Flags

The LDR instruction does not modify any Condition Code Flags.

Store Register STR

Opcode	Ext	Operands	Operation	Assembly Instruction
`0100 (0x4)`	`00`	DR, SR	[DR] ← SR	**`STR SR, [DR]`**
`0100 (0x4)`	`01`	DR, SR1, SR2	[DR + SR2] ← SR1	**`STR SR1, [DR, SR2]`**
`0100 (0x4)`	`10`	DR, SR, imm4	[DR + imm4] ← SR	**`STR SR, [DR, #imm4]`**
`0100 (0x4)`	`11`	–	–	-

Machine Code

0100	00	-	SR	DR
15 – 12	*11 – 10*	*9 – 6*	*5 – 3*	*2 – 0*

0100	01	-	SR2	SR1	DR
15 – 12	*11 – 10*	*9*	*8 – 6*	*5 – 3*	*2 – 0*

0100	10	imm4	SR	DR
15 – 12	*11 – 10*	*9 – 6*	*5 – 3*	*2 – 0*

Description

The STR instruction

- `Ext == 00`: T
- `Ext == 01`: T
- `Ext == 10`: T
- `Ext == 11`: Undefined.

Condition Code Flags

The STR instruction does not modify any Condition Code Flags.

Move **MOV**

Opcode	**Ext**	**Operands**	**Operation**	**Assembly Instruction**
`0101 (0x5)`	`0`	DR, SR	DR ← SR	**`MOV DR, SR`**
`0101 (0x5)`	`1`	DR, imm8	DR ← imm8	**`MOV DR, #imm8`**

Machine Code

`0101`	`0`	`-`	`SR`	`DR`
15 – 12	*11*	*10 – 6*	*5 – 3*	*2 – 0*

`0101`	`1`	`imm8`	`DR`
15 – 12	*11*	*10 – 3*	*2 – 0*

Description

The MOV instruction

- `Ext == 0`: T
- `Ext == 1`: T

Condition Code Flags

The MOV instruction will modify the Condition Code Flags N, Z, and P, according to the sign of the result of subtraction.

Bitwise Logical And **AND**

Opcode	Ext	Operands	Operation	Assembly Instruction
`0110 (0x6)`	`00`	DR, SR	DR ← DR & SR	**`AND DR, SR`**
`0110 (0x6)`	`01`	DR, imm7	DR ← DR & imm7	**`AND DR, #imm7`**
`0110 (0x6)`	`10`	DR, SR1, SR2	DR ← SR1 & SR2	**`AND DR, SR1, SR2`**
`0110 (0x6)`	`11`	DR, SR, imm4	DR ← SR & imm4	**`AND DR, SR, #imm4`**

Machine Code

0110	00	-	SR	DR
15 – 12	*11 – 10*	*9 – 6*	*5 – 3*	*2 – 0*

0110	01	imm7	DR
15 – 12	*11 – 10*	*9 – 3*	*2 – 0*

0110	10	-	SR2	SR1	DR
15 – 12	*11 – 10*	*9*	*8 – 6*	*5 – 3*	*2 – 0*

0110	11	imm4	SR	DR
15 – 12	*11 – 10*	*9 – 6*	*5 – 3*	*2 – 0*

Description

The AND instruction

- `Ext == 00`: T
- `Ext == 01`: T
- `Ext == 10`: T
- `Ext == 11`: T

Condition Code Flags

The AND instruction will modify the Condition Code Flags N, Z, and P, according to the new value in the destination register.

Bitwise Logical Or **ORR**

Opcode	Ext	Operands	Operation	Assembly Instruction
`0111 (0x7)`	`00`	DR, SR	DR ← DR | SR	**`ORR DR, SR`**
`0111 (0x7)`	`01`	DR, imm7	DR ← DR | imm7	**`ORR DR, #imm7`**
`0111 (0x7)`	`10`	DR, SR1, SR2	DR ← SR1 | SR2	**`ORR DR, SR1, SR2`**
`0111 (0x7)`	`11`	DR, SR, imm4	DR ← SR | imm4	**`ORR DR, SR, #imm4`**

Machine Code

`0111`	`00`	`-`	**`SR`**	**`DR`**
15 – 12	*11 – 10*	*9 – 6*	*5 – 3*	*2 – 0*

`0111`	`01`	**`imm7`**	**`DR`**
15 – 12	*11 – 10*	*9 – 3*	*2 – 0*

`0111`	`10`	`-`	**`SR2`**	**`SR1`**	**`DR`**
15 – 12	*11 – 10*	*9*	*8 – 6*	*5 – 3*	*2 – 0*

`0111`	`11`	**`imm4`**	**`SR`**	**`DR`**
15 – 12	*11 – 10*	*9 – 6*	*5 – 3*	*2 – 0*

Description

The ORR instruction

- `Ext == 00`: T
- `Ext == 01`: T
- `Ext == 10`: T
- `Ext == 11`: T

Condition Code Flags

The ORR instruction will modify the Condition Code Flags N, Z, and P, according to the new value in the destination register.

Bitwise Logical Not

NOT

Opcode	Ext	Operands	Operation	Assembly Instruction
`1000 (0x8)`	-	DR	DR ← ~DR	`NOT DR`

Machine Code

`1000`	`-`	`DR`
15 – 12	*11 – 3*	*2 – 0*

Description

The NOT instruction

Condition Code Flags

The NOT instruction will modify the Condition Code Flags N, Z, and P, according to the new value in the destination register.

Addition ADD

Opcode	Ext	Operands	Operation	Assembly Instruction
`1001 (0x9)`	`00`	DR, SR	DR ← DR + SR	`ADD DR, SR`
`1001 (0x9)`	`01`	DR, imm7	DR ← DR + imm7	`ADD DR, #imm7`
`1001 (0x9)`	`10`	DR, SR1, SR2	DR ← SR1 + SR2	`ADD DR, SR1, SR2`
`1001 (0x9)`	`11`	DR, SR, imm4	DR ← SR + imm4	`ADD DR, SR, #imm4`

Machine Code

1001	**00**	**-**	**SR**	**DR**
15 – 12	*11 – 10*	*9 – 6*	*5 – 3*	*2 – 0*

1001	**01**	**imm7**	**DR**
15 – 12	*11 – 10*	*9 – 3*	*2 – 0*

1001	**10**	**-**	**SR2**	**SR1**	**DR**
15 – 12	*11 – 10*	*9*	*8 – 6*	*5 – 3*	*2 – 0*

1001	**11**	**imm4**	**SR**	**DR**
15 – 12	*11 – 10*	*9 – 6*	*5 – 3*	*2 – 0*

Description

The ADD instruction

- `Ext == 00`: T
- `Ext == 01`: T
- `Ext == 10`: T
- `Ext == 11`: T

Condition Code Flags

The ADD instruction will modify the Condition Code Flags N, Z, and P, according to the new value in the destination register.

Subtraction **SUB**

Opcode	Ext	Operands	Operation	Assembly Instruction
`1010 (0xA)`	`00`	DR, SR	DR ← DR – SR	`SUB DR, SR`
`1010 (0xA)`	`01`	DR, imm7	DR ← DR – imm7	`SUB DR, #imm7`
`1010 (0xA)`	`10`	DR, SR1, SR2	DR ← SR1 – SR2	`SUB DR, SR1, SR2`
`1010 (0xA)`	`11`	DR, SR, imm4	DR ← SR – imm4	`SUB DR, SR, #imm4`

Machine Code

1010	00	-	SR	DR
15 – 12	*11 – 10*	*9 – 6*	*5 – 3*	*2 – 0*

1010	01	imm7	DR
15 – 12	*11 – 10*	*9 – 3*	*2 – 0*

1010	10	-	SR2	SR1	DR
15 – 12	*11 – 10*	*9*	*8 – 6*	*5 – 3*	*2 – 0*

1010	11	imm4	SR	DR
15 – 12	*11 – 10*	*9 – 6*	*5 – 3*	*2 – 0*

Description

The SUB instruction

- `Ext == 00`: T
- `Ext == 01`: T
- `Ext == 10`: T
- `Ext == 11`: T

Condition Code Flags

The SUB instruction will modify the Condition Code Flags N, Z, and P, according to the new value in the destination register.

Compare **CMP**

Opcode	Ext	Operands	Operation	Assembly Instruction
`1011 (0xB)`	`0`	DR, SR	NZP ⇐ DR – SR	`CMP DR, SR`
`1011 (0xB)`	`1`	DR, imm7	NZP ⇐ DR – imm8	`CMP DR, #imm8`

Machine Code

`1011`	`0`	`-`	`SR`	`DR`
15 – 12	*11*	*10 – 6*	*5 – 3*	*2 – 0*

`1011`	`1`	`imm8`	`DR`
15 – 12	*11*	*10 – 3*	*2 – 0*

Description

The CMP instruction

- `Ext == 0`: T
- `Ext == 1`: T

Condition Code Flags

The CMP instruction will modify the Condition Code Flags N, Z, and P, according to the sign of the result of subtraction.

Push Register onto Stack **PSH**

Opcode	Ext	Operands	Operation	Assembly Instruction
`1100 (0xC)`	`-`	SR	SP – 1 [SP] ← SR	**`PSH SR`**

Machine Code

`1100`	`-`	**`SR`**	`-`
15 – 12	*11 – 6*	*5 – 3*	*2 – 0*

Description

The PSH instruction decrements the stack pointer and stores the source register in the address of the updated stack pointer.

Condition Code Flags

The PSH instruction does not modify any Condition Code Flags.

Pop Register from Stack **POP**

Opcode	Ext	Operands	Operation	Assembly Instruction
`1101 (0xD)`	-	DR	DR ← [SP] SP + 1	`POP DR`

Machine Code

`1101`	`-`	`DR`
15 – 12	*11 – 3*	*2 – 0*

Description

The POP instruction loads the contents of the memory at the address of the stack pointer, then increments the stack pointer.

Condition Code Flags

The POP instruction does not modify any Condition Code Flags.

Jump

JMP

Opcode	Ext	Operands	Operation	Assembly Instruction
`1110 (0xE)`	-	DR	PC ← [DR + imm9]	`JMP DR, #imm9`

Machine Code

`1110`	`imm9`	`DR`
15 – 12	*11 – 3*	*2 – 0*

Description

The JMP instruction

Condition Code Flags

The JMP instruction does not modify any Condition Code Flags.

Halt Program Execution HLT

Opcode	Ext	Operands	Operation	Assembly Instruction
`1111 (0xF)`	`-`	–	Halts Program	`HLT`

Machine Code

1111	-
15 – 12	*11 – 0*

Description

The HALT instruction

Condition Code Flags

The HALT instruction does not modify any Condition Code Flags.