LoongArch ELF ABI specification

Register Convention

Table 1. Integer Register Convention

Name	Alias	Meaning	Preserved across calls
`$r0`	`$zero`	Constant zero	—
`$r1`	`$ra`	Return address	No
`$r2`	`$tp`	Thread pointer	--(Unallocatable)
`$r3`	`$sp`	Stack pointer	Yes
`$r4-$r11`	`$a0-$a7`	Argument registers	No
`$r4-$r5`	`$v0-$v1`	Return value	No
`$r12-$r20`	`$t0-$t8`	Temp registers	No
`$r21`	–	Reserved	--(Unallocatable)
`$r22`	`$fp/$s9`	Frame pointer/Saved register	Yes
`$r23-$r31`	`$s0-$s8`	Saved register	Yes

Table 2. Floating-point Register Convention

Name	Alias	Meaning	Preserved across calls
`$f0-$f7`	`$fa0-$fa7`	Argument registers	No
`$f0-$f1`	`$fv0-$fv7`	Return value	No
`$f8-$f23`	`$ft0-$ft15`	Temp registers	No
`$f24-$f31`	`$fs0-$fs7`	Saved register	Yes

Type Size and Alignment

Table 3. LP64 ABI

Scalar type	Size(Bytes)	Alignment(Bytes)
`bool/_Bool`	1	1
`unsigned/signed char`	1	1
`unsigned/signed short`	2	2
`unsigned/signed int`	4	4
`unsigned/signed long`	8	8

Scalar type	Size(Bytes)	Alignment(Bytes)
`unsigned/signed long long`	8	8
`pointer`	8	8
`float`	4	4
`double`	8	8
`long double`	16	16

char is signed.

ELF Object Files

EI_CLASS: File class.

EI_CLASS	Value	Description
`ELFCLASS32`	1	32-bit objects
`ELFCLASS64`	2	64-bit objects

e_machine: Identifies the machine.

LoongArch (258)

e_flags: Identifies the ABIs of this ELF file.

ABIs	Value	Description
`lp32`	0x1	soft float
`Reserved`	0x2	—
`lp64`	0x3	64bit default ABI
`lp32f`	0x4	single float
`lp32d`	0x5	double float
`Reserved`	0x6--	—

ABI Version:

ABI Version:	Value	Description
`v0`	0	Stack operands base relocation type.
`v1`	1	Another relocation type IF needed.
`–`	2--	Reserved.

Relocations

Table 4. ELF Relocation types

Enum	ELF reloc type	Usage	Detail
0	R_LARCH_NONE		
1	R_LARCH_32	Runtime address resolving	`*(int32_t *) PC = RtAddr + A`
2	R_LARCH_64	Runtime address resolving	`*(int64_t *) PC = RtAddr + A`
3	R_LARCH_RELATIVE	Runtime fixup for load-address	`(void *) PC = B + A`
4	R_LARCH_COPY	Runtime memory copy in executable	`memcpy (PC, RtAddr, sizeof (sym))`
5	R_LARCH_JUMP_SLOT	Runtime PLT supporting	*implementation-defined*

Enum	ELF reloc type	Usage	Detail
6	R_LARCH_TLS_DTPMOD32	Runtime relocation for TLS-GD	*(int32_t *) PC = ID of module defining sym
7	R_LARCH_TLS_DTPMOD64	Runtime relocation for TLS-GD	*(int64_t *) PC = ID of module defining sym
8	R_LARCH_TLS_DTPREL32	Runtime relocation for TLS-GD	*(int32_t *) PC = DTV-relative offset for sym
9	R_LARCH_TLS_DTPREL64	Runtime relocation for TLS-GD	*(int64_t *) PC = DTV-relative offset for sym

Enum	ELF reloc type	Usage	Detail
10	R_LARCH_TLS_TPREL32	Runtime relocation for TLE-IE	`*(int32_t *) PC = T`
11	R_LARCH_TLS_TPREL64	Runtime relocation for TLE-IE	`*(int64_t *) PC = T`
12	R_LARCH_IRELATIVE	Runtime local indirect function resolving	`(void *) PC = (void )()( (B + A)) ()`
… Reserved for dynamic linker.			
20	R_LARCH_MARK_LA	Mark la.abs	Load absolute address for static link.

Enum	ELF reloc type	Usage	Detail
21	R_LARCH_MARK_PCREL	Mark external label branch	Access PC relative address for static link.
22	R_LARCH_SOP_PUSH_PCREL	Push PC-relative offset	`push (S - PC + A)`
23	R_LARCH_SOP_PUSH_ABSOLUTE	Push constant or absolute address	`push (S + A)`
24	R_LARCH_SOP_PUSH_DUP	Duplicate stack top	`opr1 = pop (), push (opr1), push (opr1)`

Enum	ELF reloc type	Usage	Detail
25	R_LARCH_SOP_PUSH_GPREL	Push for access GOT entry	push (G)
26	R_LARCH_SOP_PUSH_TLS_TPREL	Push for TLS-LE	push (T)
27	R_LARCH_SOP_PUSH_TLS_GOT	Push for TLS-IE	push (IE)

Enum	ELF reloc type	Usage	Detail
28	R_LARCH_SOP_PUSH_TLS_GD	Push for TLS-GD	`push (GD)`
29	R_LARCH_SOP_PUSH_PLT_PCREL	Push for external function calling	`push (PLT - PC)`
30	R_LARCH_SOP_ASSERT	Assert stack top	`assert (pop ())`
31	R_LARCH_SOP_NOT	Stack top operation	`push (!pop ())`

Enum	ELF reloc type	Usage	Detail
32	R_LARCH_SOP_SUB	Stack top operation	`opr2 = pop (), opr1 = pop (), push (opr1 - opr2)`
33	R_LARCH_SOP_SL	Stack top operation	`opr2 = pop (), opr1 = pop (), push (opr1 << opr2)`
34	R_LARCH_SOP_SR	Stack top operation	`opr2 = pop (), opr1 = pop (), push (opr1 >> opr2)`
35	R_LARCH_SOP_ADD	Stack top operation	`opr2 = pop (), opr1 = pop (), push (opr1 + opr2)`
36	R_LARCH_SOP_AND	Stack top operation	`opr2 = pop (), opr1 = pop (), push (opr1 & opr2)`

Enum	ELF reloc type	Usage	Detail
37	R_LARCH_SOP_IF_ELSE	Stack top operation	`opr3 = pop (), opr2 = pop (), opr1 = pop (), push (opr1 ? opr2 : opr3)`
38	R_LARCH_SOP_POP_32_S_10_5	Instruction imm-field relocation	`opr1 = pop (), (*(uint32_t *) PC) [14 … 10] = opr1 [4 … 0]` with check 5-bit signed overflow
39	R_LARCH_SOP_POP_32_U_10_12	Instruction imm-field relocation	`opr1 = pop (), (*(uint32_t *) PC) [21 … 10] = opr1 [11 … 0]` with check 12-bit unsigned overflow

Enum	ELF reloc type	Usage	Detail
40	R_LARCH_SOP_POP_32_S_10_12	Instruction imm-field relocation	`opr1 = pop (), (*(uint32_t *) PC) [21 … 10] = opr1 [11 … 0]` with check 12-bit signed overflow
41	R_LARCH_SOP_POP_32_S_10_16	Instruction imm-field relocation	`opr1 = pop (), (*(uint32_t *) PC) [25 … 10] = opr1 [15 … 0]` with check 16-bit signed overflow
42	R_LARCH_SOP_POP_32_S_10_16_S2	Instruction imm-field relocation	`opr1 = pop (), (*(uint32_t *) PC) [25 … 10] = opr1 [17 … 2]` with check 18-bit signed overflow and 4-bit aligned

Enum	ELF reloc type	Usage	Detail
43	R_LARCH_SOP_POP_32_S_5_20	Instruction imm-field relocation	`opr1 = pop (), (*(uint32_t *) PC) [24 … 5] = opr1 [19 … 0]` with check 20-bit signed overflow
44	R_LARCH_SOP_POP_32_S_0_5_10_16_S2	Instruction imm-field relocation	`opr1 = pop (), ((uint32_t *) PC) [4 … 0] = opr1 [22 … 18], ((uint32_t *) PC) [25 … 10] = opr1 [17 … 2]` with check 23-bit signed overflow and 4-bit aligned

Enum	ELF reloc type	Usage	Detail
45	R_LARCH_SOP_POP_32_S_0_10_10_16_S2	Instruction imm-field relocation	`opr1 = pop (), ((uint32_t *) PC) [9 … 0] = opr1 [27 … 18], ((uint32_t *) PC) [25 … 10] = opr1 [17 … 2]` with check 28-bit signed overflow and 4-bit aligned
46	R_LARCH_SOP_POP_32_U	Instruction fixup	`(*(uint32_t *) PC) = pop ()` with check 32-bit unsigned overflow
47	R_LARCH_ADD8	8-bit in-place addition	`*(int8_t *) PC += S + A`
48	R_LARCH_ADD16	16-bit in-place addition	`*(int16_t *) PC += S + A`

Enum	ELF reloc type	Usage	Detail
49	R_LARCH_ADD24	24-bit in-place addition	`*(int24_t *) PC += S + A`
50	R_LARCH_ADD32	32-bit in-place addition	`*(int32_t *) PC += S + A`
51	R_LARCH_ADD64	64-bit in-place addition	`*(int64_t *) PC += S + A`
52	R_LARCH_SUB8	8-bit in-place subtraction	`*(int8_t *) PC -= S + A`
53	R_LARCH_SUB16	16-bit in-place subtraction	`*(int16_t *) PC -= S + A`
54	R_LARCH_SUB24	24-bit in-place subtraction	`*(int24_t *) PC -= S + A`

Enum	ELF reloc type	Usage	Detail
55	R_LARCH_SUB32	32-bit in-place subtraction	`*(int32_t *) PC -= S + A`
56	R_LARCH_SUB64	64-bit in-place subtraction	`*(int64_t *) PC -= S + A`
57	R_LARCH_GNU_VTINHERIT	GNU C++ vtable hierarchy	
58	R_LARCH_GNU_VTENTRY	GNU C++ vtable member usage	