|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |  |  |  |  |  |
| funct7 | | | | | | | rs2 | | | | | rs1 | | | | | funct3 | | | rd | | | | | opcode | | | | | | | R-type | | | | |
| imm[11:0] | | | | | | | | | | | | rs1 | | | | | funct3 | | | rd | | | | | opcode | | | | | | | I-type | | | | |
| imm[11:5] | | | | | | | rs2 | | | | | rs1 | | | | | funct3 | | | imm[4:0] | | | | | opcode | | | | | | | S-type | | | | |
| imm[12|10:5] | | | | | | | rs2 | | | | | rs1 | | | | | funct3 | | | rd | | | | | opcode | | | | | | | B-type | | | | |
| imm[31:12] | | | | | | | | | | | | | | | | | | | | rd | | | | | opcode | | | | | | | U-type | | | | |
| imm[20|10:1|11|19:12] | | | | | | | | | | | | | | | | | | | | rd | | | | | opcode | | | | | | | J-type | | | | |

**Zbb**: “Basic bit-manipulation” Extension

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 31 |  |  |  |  |  | 25 | 24 |  |  |  | 20 | 19 |  |  |  | 15 | 14 |  | 12 | 11 |  |  |  | 7 | 6 |  |  |  |  |  | 0 |  |  |  |  |  |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | rs2 | | | | | rs1 | | | | | 1 | 1 | 1 | rd | | | | | 0 | 1 | 1 | 0 | 0 | 1 | 1 | ANDN | | | | |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | rs2 | | | | | rs1 | | | | | 1 | 1 | 0 | rd | | | | | 0 | 1 | 1 | 0 | 0 | 1 | 1 | ORN | | | | |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | rs2 | | | | | rs1 | | | | | 1 | 0 | 0 | rd | | | | | 0 | 1 | 1 | 0 | 0 | 1 | 1 | XNOR | | | | |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | rs1 | | | | | 0 | 0 | 1 | rd | | | | | 0 | 0 | 1 | 0 | 0 | 1 | 1 | CLZ | | | | |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | rs1 | | | | | 0 | 0 | 1 | rd | | | | | 0 | 0 | 1 | 0 | 0 | 1 | 1 | CTZ | | | | |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | rs1 | | | | | 0 | 0 | 1 | rd | | | | | 0 | 0 | 1 | 0 | 0 | 1 | 1 | CPOP | | | | |
| 0 | 0 | 0 | 0 | 1 | 0 | 1 | rs2 | | | | | rs1 | | | | | 1 | 1 | 0 | rd | | | | | 0 | 1 | 1 | 0 | 0 | 1 | 1 | MAX | | | | |
| 0 | 0 | 0 | 0 | 1 | 0 | 1 | rs2 | | | | | rs1 | | | | | 1 | 1 | 1 | rd | | | | | 0 | 1 | 1 | 0 | 0 | 1 | 1 | MAXU | | | | |
| 0 | 0 | 0 | 0 | 1 | 0 | 1 | rs2 | | | | | rs1 | | | | | 1 | 0 | 0 | rd | | | | | 0 | 1 | 1 | 0 | 0 | 1 | 1 | MIN | | | | |
| 0 | 0 | 0 | 0 | 1 | 0 | 1 | rs2 | | | | | rs1 | | | | | 1 | 0 | 1 | rd | | | | | 0 | 1 | 1 | 0 | 0 | 1 | 1 | MINU | | | | |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | rs1 | | | | | 0 | 0 | 1 | rd | | | | | 0 | 0 | 1 | 0 | 0 | 1 | 1 | SEXT.B | | | | |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | rs1 | | | | | 0 | 0 | 1 | rd | | | | | 0 | 0 | 1 | 0 | 0 | 1 | 1 | SEXT.H | | | | |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | rs1 | | | | | 1 | 0 | 0 | rd | | | | | 0 | 1 | 1 | 0 | 0 | 1 | 1 | ZEXT.H | | | | |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | rs2 | | | | | rs1 | | | | | 0 | 0 | 1 | rd | | | | | 0 | 1 | 1 | 0 | 0 | 1 | 1 | ROL | | | | |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | rs2 | | | | | rs1 | | | | | 1 | 0 | 1 | rd | | | | | 0 | 1 | 1 | 0 | 0 | 1 | 1 | ROR | | | | |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | shamt | | | | | rs1 | | | | | 1 | 0 | 1 | rd | | | | | 0 | 0 | 1 | 0 | 0 | 1 | 1 | RORI | | | | |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | rs1 | | | | | 1 | 0 | 1 | rd | | | | | 0 | 0 | 1 | 0 | 0 | 1 | 1 | ORC.B | | | | |
| 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | rs1 | | | | | 1 | 0 | 1 | rd | | | | | 0 | 0 | 1 | 0 | 0 | 1 | 1 | REV8 | | | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |  |  |  |  |  |
| funct7 | | | | | | | rs2 | | | | | rs1 | | | | | funct3 | | | rd | | | | | opcode | | | | | | | R-type | | | | |
| imm[11:0] | | | | | | | | | | | | rs1 | | | | | funct3 | | | rd | | | | | opcode | | | | | | | I-type | | | | |
| imm[11:5] | | | | | | | rs2 | | | | | rs1 | | | | | funct3 | | | imm[4:0] | | | | | opcode | | | | | | | S-type | | | | |
| imm[12|10:5] | | | | | | | rs2 | | | | | rs1 | | | | | funct3 | | | rd | | | | | opcode | | | | | | | B-type | | | | |
| imm[31:12] | | | | | | | | | | | | | | | | | | | | rd | | | | | opcode | | | | | | | U-type | | | | |
| imm[20|10:1|11|19:12] | | | | | | | | | | | | | | | | | | | | rd | | | | | opcode | | | | | | | J-type | | | | |

**Zri**: “Load/Store indirect with Index” Extension

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 31 |  |  |  |  |  | 25 | 24 |  |  |  | 20 | 19 |  |  |  | 15 | 14 |  | 12 | 11 |  |  |  | 7 | 6 |  |  |  |  |  | 0 |  |  |  |  |  |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | rs2 | | | | | rs1 | | | | | 1 | 1 | 1 | rd | | | | | 0 | 0 | 0 | 0 | 0 | 1 | 1 | LB.R | | | | |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | rs2 | | | | | rs1 | | | | | 1 | 1 | 1 | rd | | | | | 0 | 0 | 0 | 0 | 0 | 1 | 1 | LH.R | | | | |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | rs2 | | | | | rs1 | | | | | 1 | 1 | 1 | rd | | | | | 0 | 0 | 0 | 0 | 0 | 1 | 1 | LW.R | | | | |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | rs2 | | | | | rs1 | | | | | 1 | 1 | 1 | rd | | | | | 0 | 0 | 0 | 0 | 0 | 1 | 1 | LBU.R | | | | |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | rs2 | | | | | rs1 | | | | | 1 | 1 | 1 | rd | | | | | 0 | 0 | 0 | 0 | 0 | 1 | 1 | LHU.R | | | | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | rs3 | | | | | rs1 | | | | | 1 | 1 | 1 | rs2 | | | | | 0 | 1 | 0 | 0 | 0 | 1 | 1 | SB.R | | | | |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | rs3 | | | | | rs1 | | | | | 1 | 1 | 1 | rs2 | | | | | 0 | 1 | 0 | 0 | 0 | 1 | 1 | SH.R | | | | |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | rs3 | | | | | rs1 | | | | | 1 | 1 | 1 | rs2 | | | | | 0 | 1 | 0 | 0 | 0 | 1 | 1 | SW.R | | | | |

|  |  |
| --- | --- |
| lb | rd, rs2(rs1) |
| lh | rd, rs2(rs1) |
| lw | rd, rs2(rs1) |
| lbu | rd, rs2(rs1) |
| lhu | rd, rs2(rs1) |
| sb | rs2, rs3(rs1) |
| sh | rs2, rs3(rs1) |
| sw | rs2, rs3(rs1) |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |  |  |  |  |  |
| funct7 | | | | | | | rs2 | | | | | rs1 | | | | | funct3 | | | rd | | | | | opcode | | | | | | | R-type | | | | |
| imm[11:0] | | | | | | | | | | | | rs1 | | | | | funct3 | | | rd | | | | | opcode | | | | | | | I-type | | | | |
| imm[11:5] | | | | | | | rs2 | | | | | rs1 | | | | | funct3 | | | imm[4:0] | | | | | opcode | | | | | | | S-type | | | | |
| imm[12|10:5] | | | | | | | rs2 | | | | | rs1 | | | | | funct3 | | | rd | | | | | opcode | | | | | | | B-type | | | | |
| imm[31:12] | | | | | | | | | | | | | | | | | | | | rd | | | | | opcode | | | | | | | U-type | | | | |
| imm[20|10:1|11|19:12] | | | | | | | | | | | | | | | | | | | | rd | | | | | opcode | | | | | | | J-type | | | | |

**Zor**: “Objective RISC” Extension

Unprivileged:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 31 |  |  |  |  |  | 25 | 24 |  |  |  | 20 | 19 |  |  |  | 15 | 14 |  | 12 | 11 |  |  |  | 7 | 6 |  |  |  |  |  | 0 |  |  |  |  | |  |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | rs2 | | | | | rs1 | | | | | 0 | 0 | 0 | rs3 | | | | | 0 | 0 | 0 | 1 | 0 | 1 | 1 | SP.R | | | | R | |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | rs2 | | | | | rs1 | | | | | 0 | 0 | 0 | rd | | | | | 0 | 0 | 0 | 1 | 0 | 1 | 1 | LP.R | | | | R | |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | index[4:0] | | | | | frame | | | | | 0 | 0 | 0 | rs1 | | | | | 0 | 0 | 0 | 1 | 0 | 1 | 1 | SV | | | | R | |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | index[4:0] | | | | | frame | | | | | 0 | 0 | 0 | rd | | | | | 0 | 0 | 0 | 1 | 0 | 1 | 1 | RST | | | | R | |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | zero | | | | | rs1 | | | | | 0 | 0 | 0 | rd | | | | | 0 | 0 | 0 | 1 | 0 | 1 | 1 | QDTB | | | | R | |
| 0 | 0 | 0 | 0 | 1 | 0 | 1 | zero | | | | | rs1 | | | | | 0 | 0 | 0 | rd | | | | | 0 | 0 | 0 | 1 | 0 | 1 | 1 | QDTH | | | | R | |
| 0 | 0 | 0 | 0 | 1 | 1 | 0 | zero | | | | | rs1 | | | | | 0 | 0 | 0 | rd | | | | | 0 | 0 | 0 | 1 | 0 | 1 | 1 | QDTW | | | | R | |
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | zero | | | | | rs1 | | | | | 0 | 0 | 0 | rd | | | | | 0 | 0 | 0 | 1 | 0 | 1 | 1 | QDTD | | | | R | |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | zero | | | | | rs1 | | | | | 0 | 0 | 0 | rd | | | | | 0 | 0 | 0 | 1 | 0 | 1 | 1 | QPI | | | | R | |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | zero | | | | | zero | | | | | 0 | 0 | 0 | rd | | | | | 0 | 0 | 0 | 1 | 0 | 1 | 1 | GCP | | | | R | |
| 0 | 0 | 0 | 1 | 1 | 0 | 0 | zero | | | | | frame | | | | | 0 | 0 | 0 | frame | | | | | 0 | 0 | 0 | 1 | 0 | 1 | 1 | POP | | | | R | |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | zero | | | | | zero | | | | | 0 | 0 | 0 | zero | | | | | 0 | 0 | 0 | 1 | 0 | 1 | 1 | RTLIB | | | | R | |
| 0 | 0 | 1 | 0 | 0 | 1 | 0 | zero | | | | | zero | | | | | 0 | 0 | 0 | zero | | | | | 0 | 0 | 0 | 1 | 0 | 1 | 1 | CPFC | | | | R | |
| 0 | 0 | 1 | 0 | 0 | 1 | 1 | zero | | | | | zero | | | | | 0 | 0 | 0 | zero | | | | | 0 | 0 | 0 | 1 | 0 | 1 | 1 | CHECK | | | | R | |
| imm[11:5] | | | | | | | rs2 | | | | | rs1 | | | | | 0 | 0 | 1 | imm[4:0] | | | | | 0 | 0 | 0 | 1 | 0 | 1 | 1 | SP | | | | S | |
| imm[11:0] | | | | | | | | | | | | rs1 | | | | | 0 | 1 | 0 | rd | | | | | 0 | 0 | 0 | 1 | 0 | 1 | 1 | LP | | | | I | |
| imm[11:0] | | | | | | | | | | | | rs1 | | | | | 0 | 1 | 1 | ra | | | | | 0 | 0 | 0 | 1 | 0 | 1 | 1 | JLIB | | | | I | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | rs2 | | | | | rs1 | | | | | 1 | 0 | 0 | rd | | | | | 0 | 0 | 0 | 1 | 0 | 1 | 1 | ALC | | | | R | |
| pi[11:0] | | | | | | | | | | | | rs1 | | | | | 1 | 0 | 1 | rd | | | | | 0 | 0 | 0 | 1 | 0 | 1 | 1 | ALCI.P | | | | I | |
| dt[11:0] | | | | | | | | | | | | rs1 | | | | | 1 | 1 | 0 | rd | | | | | 0 | 0 | 0 | 1 | 0 | 1 | 1 | ALCI.D | | | | I | |
| dt[6:0] | | | | | | | 0 | 0 | 0 | 0 | 0 | rd | | | | | 1 | 1 | 1 | pi[4:0] | | | | | 0 | 0 | 0 | 1 | 0 | 1 | 1 | ALCI | | | | S | |
| dt[6:0] | | | | | | | 0 | 0 | 0 | 1 | 0 | frame | | | | | 1 | 1 | 1 | pi[4:0] | | | | | 0 | 0 | 0 | 1 | 0 | 1 | 1 | PUSHG | | | | S | |
| dt[6:0] | | | | | | | 0 | 0 | 0 | 1 | 1 | frame | | | | | 1 | 1 | 1 | pi[4:0] | | | | | 0 | 0 | 0 | 1 | 0 | 1 | 1 | PUSH | | | | S | |

Machine Mode:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 31 |  |  |  |  | 26 | 25 | 24 |  |  |  | 20 | 19 |  |  |  | 15 | 14 |  | 12 | 11 |  |  |  | 7 | 6 |  |  |  |  |  | 0 |  |  |  |  | |  |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | rd | | | | | 1 | 1 | 1 | 0 | 0 | 1 | 1 | ALCB | | | | R | |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | rs2 | | | | | rs1 | | | | | 0 | 0 | 0 | rd | | | | | 1 | 1 | 1 | 0 | 0 | 1 | 1 | CIOP | | | | R | |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | rs1 | | | | | 0 | 0 | 0 | rd | | | | | 1 | 1 | 1 | 0 | 0 | 1 | 1 | CCP | | | | R | |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | rs1 | | | | | 0 | 0 | 0 | rd | | | | | 1 | 1 | 1 | 0 | 0 | 1 | 1 | RPR | | | | R | |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | rs1 | | | | | 0 | 0 | 0 | rd | | | | | 1 | 1 | 1 | 0 | 0 | 1 | 1 | QPIR | | | | R | |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | rs1 | | | | | 0 | 0 | 0 | rd | | | | | 1 | 1 | 1 | 0 | 0 | 1 | 1 | QDTR | | | | R | |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | rs1 | | | | | 0 | 0 | 0 | rd | | | | | 1 | 1 | 1 | 0 | 0 | 1 | 1 | QPTR | | | | R | |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | rd | | | | | 1 | 1 | 1 | 0 | 0 | 1 | 1 | SEAL | | | | R | |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | rd | | | | | 1 | 1 | 1 | 0 | 0 | 1 | 1 | UNSL | | | | R | |

Misc:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| reg | alias | reg | alias |  | pseudo-instruction | implemented as |
| x0 | zero | x16 | a6 |  | lcp rd, imm(rs1) | lp rd, imm(rs1) |
| x1 | ra ~~rix~~ | x17 | a7 |  |  | sp x0, imm(rs1) |
| x2 | frame | x18 | s2 |  | lcp.r rd, imm(rs1) | lp.r rd, rs2(rs1) |
| x3 | ~~rcd/~~root/core | x19 | s3 |  |  | sp.r x0, rs2(rs1) |
| x4 | ctxt | x20 | s4 |  | scp rs2, imm(rs1) | sp rs2, imm(rs1) |
| x5 | t0 | x21 | s5 |  |  | addi rs2, x0,0 |
| x6 | t1 | x22 | s6 |  | scp.r rs2, rs3(rs1) | sp.r rs2, rs3(rs1) |
| x7 | t2 | x23 | s7 |  |  | addi rs2, x0,0 |
| x8 | s0 | x24 | s8 |  | pusht pi,dt | alci frame, pi,dt |
| x9 | s1 | x25 | s9 |  |  |  |
| x10 | a0 | x26 | s10/bm |  |  |  |
| x11 | a1 | x27 | cnst |  |  |  |
| x12 | a2 | x28 | t3 |  |  |  |
| x13 | a3 | x29 | t4 |  |  |  |
| x14 | a4 | x30 | t5 |  |  |  |
| x15 | a5 | x31 | t6 |  |  |  |

Implementation:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Instruction** | **rdst** | **rdat** | **rptr** | **raux** | **imm** |
| sb/h/w | zero | ra.rix | rs1 | rs2 | imm |
| lb/bu/h/hu/w | rd | --- | rs1 | ra | imm |
| sp | zero | ra.rix | rs1 | rs2 | imm |
| lp | rd | --- | rs1 | ra | imm |
| sb/h/w.r | zero | rs3 | rs1 (≠ frame) | rs2 | --- |
| lb/bu/h/hu/w.r | rd | rs2 | rs1 (≠ frame) | --- | --- |
| sp.r | zero | rs3 | rs1 (≠ frame) | rs2 | --- |
| lp.r | rd | rs2 | rs1 (≠ frame) | --- | --- |
| sv | zero | ra.rix | frame | rs1 | index |
| rst | rd | ra.rix | frame | bm | index |
| qdtx |  |  |  |  |  |
| qpi |  |  |  |  |  |
| gcp |  |  |  |  |  |
| pop | frame | ra.rix | frame | --- | --- |
| jlib | ra | frame | rs1 | ra | imm |
| jal | rd | frame | --- | ra | imm |
| jr | rd | frame | rs1 | ra | imm |
| rtlib | ra | ra.rix | ra | frame | --- |
| alc | rd (≠ frame) | rs1 | alc\_params | rs2 | --- |
| alci.p | rd (≠ frame) | rs1 | alc\_params | --- | pi |
| alci.d | rd (≠ frame) | rs1 | alc\_params | --- | dt |
| alci | rd | ra.rix | alc\_params | frame | pi & dt |
| pushg | rd | ra.rix | alc\_params | frame | pi & dt |
| push | rd | ra.rix | alc\_params | frame | pi & dt |
| alcb |  |  |  |  |  |
| ciop | rd | rs1 | --- | rs2 | --- |
| rpr |  |  |  |  |  |
| qpir |  |  |  |  |  |
| qdtr |  |  |  |  |  |
| qptr |  |  |  |  |  |
| seal |  |  |  |  |  |
| unsl |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 31 | 30 | 29 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 3 | 2 | 1 | 0 |
| ra.rix | lib entry | rix(30:1) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | color |
| frame | frame(31:3) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 1 | 0 | color |
| pi | uini | pi(30:2) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | bumper/gc | gc |
| dt | rc | ri | dt(29:0) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

|  |  |  |
| --- | --- | --- |
| instruction | condition | action |
| jlib | ra.rix(color) != frame(color) target ptr != ra.rcd | set ra.rix(lib entry), toggle rix(color) |
| jal ra, … or jr ra, … | ra.rix(color) != frame(color) | clear ra.rix(lib entry), toggle rix(color) |
| pushx | ra.rix(color) = frame(color) | toggle frame(color) |
| pop | ra.rix(color) != frame(color) | toggle frame(color) |
| jr …, 0(ra) | ra.rix(color) = frame(color) | toggle ra.rix(color) if ra.rix(lib entry) = 1 do cross code-object return else stay in this code-object |

**OBJECTS**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | *Generic Header* | | | | | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |  | 2 | 1 | 0 |
|  |  | gc | | s | r | w | x | d | λ(24:2) | | | f | i |
|  |  | gc: reserved bits for garbage collection  s: only accessible in supervisor mode (or higher)  r: readable  w: writable  x: executable  d: data only (no pointers allowed)  λ: length of this object  f: stack frame object  i: boxed immediate | | | | | | | | | | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | | Ordinary | | | | | | | | | | | | | | | | | | | | |
|  |  | |  | | | | | | | | | | | | | | | | | | | | |
|  |  | | 31 | 30 | | 29 | | 28 | | 27 | | 26 | | 25 | | 24 |  | 2 | | 1 | | 0 | |
| *♦-4* |  | gc | | | s | | r | | w | | x | | d | | λ(24:2) | | | | 0 | | 0 | |
| *♦+0* |  |  | | | | | | | | | | | | | | | | | | | | |
|  |  | ● ● ● | | | | | | | | | | | | | | | | | | | | |
| *♦+λ-4* |  |  | | | | | | | | | | | | | | | | | | | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | | Executable | | | | | | | | | | | | | | | | | | | | |
|  |  | |  | | | | | | | | | | | | | | | | | | | | |
|  |  | | 31 | 30 | | 29 | | 28 | | 27 | | 26 | | 25 | | 24 |  | 2 | | 1 | | 0 | |
| *♦-16* |  | gc | | | s | | r | | w | | 1 | | 1 | | λ(24:2) | | | | 0 | | 0 | |
| *♦-12* |  | got (virtual index) | | | | | | | | | | | | | | | | | | | | |
| *♦-8* |  | got (pointer) | | | | | | | | | | | | | | | | | | | | |
| *♦-4* |  | got (λ) | | | | | | | | | | | | | | | | | | | | |
| *♦+0* |  |  | | | | | | | | | | | | | | | | | | | | |
|  |  | ● ● ● | | | | | | | | | | | | | | | | | | | | |
| *♦+λ-4* |  |  | | | | | | | | | | | | | | | | | | | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Frame | | | | | | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  | |  |  |  |
|  |  | 31 | 30 | 29 | 28 | 27 | 26 |  | 4 | 3 | | 2 | 1 | 0 |
| *♦-16* |  | gc | | s | 1 | 1 | λ(26:4) | | | | ■ | 0 | 1 | 0 |
| *♦-12* |  | ra-ptr (if ■ = 1) | | | | | | | | | | | | |
| *♦-8* |  | fp-ptr! | | | | | | | | | | | | |
| *♦-4* |  | fp-eop! | | | | | | | | | | | | |
| *♦+0* |  |  | | | | | | | | | | | | |
|  |  | ● ● ● | | | | | | | | | | | | |
|  |  |  | | | | | | | | | | | | |
| *♦+λ-8* |  | fp-ix! | | | | | | | | | | | | |
| *♦+λ-4* |  | ra-ix! | | | | | | | | | | | | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Immediate (Primitive) | | | | | | |
|  |  |  |  |  |  |  |  |  |
|  |  | 31 | 30 | 29 |  | 2 | 1 | 0 |
| *♦+0* |  | gc | | zero | | | 0 | 1 |
| *♦+4* |  | integer(31:0) | | | | | | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Immediate (Pointer) | | | | | | |
|  |  |  |  |  |  |  |  |  |
|  |  | 31 | 30 | 29 |  | 2 | 1 | 0 |
| *♦+0* |  | gc | | zero | | | 1 | 1 |
| *♦+4* |  | ptr(31:0) | | | | | | |
| *♦+8* |  | ix(31:0) | | | | | | |
| *♦+12* |  | attr(31:0) | | | | | | |

**POINTERS & DATA  
(in memory)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 31 |  | 1 | 0 |
| immediate (prim) pointer: | ptr(31:2) | | 0 | 1 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 31 |  | 3 | 2 | 1 | 0 |
| ordinary pointer | ptr(31:4) | | 0 | 0 | 1 | 1 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 31 |  | 3 | 2 | 1 | 0 |
| immediate (ptr) pointer: | ptr(31:4) | | 0 | 1 | 1 | 1 |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 31 |  | | 5 | 4 | 3 | 2 | 1 | 0 |
| io pointer: | device | | λ | | g | 1 | 1 | 1 | 1 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 32 | 31 | 25 | 24 | | 17 | 16 | | 9 | 8 | | 1 | 0 |
| Small Data (w): | 31 | int(30:0) | | | | | | | | | | | 0 |
| Small Data (h): | 15 | h1(14:0) | | | | | | h0(15:0) | | | | | 0 |
| Small Data (b): | 7 | b3 | | | b2 | | | b1 | | | b0 | | 0 |

Allocate immediate primitive if:

* sw and rs(30) ≠ rs(31)
* sh at h1 and rs(14) ≠ rs(15)
* sb at b3 and (rs(7) = 1 or rs < 0)

**REGISTER FILE & PIPELINE**

**Architectural Registers (x0-x31):**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | T |  | 31 |  | 4 | 3 | 2 | 1 | 0 |  | 31 |  | 0 |  | Flags | | | | |  | 24 |  |  |  |  | 0 |
| **data** |  | 0 |  | value(31:0) | | | | | | |  | alc\_addr | | |  | 0 | 0 | 0 | 0 | 0 |  | zero | | | | | |
|  |  |  |  |  | | | | | | |  |  | | |  |  |  |  |  |  |  |  | | | | | |
| **ordinary pointer** |  | 1 |  | ptr(31:4) | | | 0 | 0 | 0 | 0 |  | index(31:0) | | |  | s | r | w | x | d |  | λ(26:2) | | | | | |
|  |  |  |  |  | | |  |  |  |  |  |  | | |  |  |  |  |  |  |  |  | | | | | |
| **frame-type** |  | 1 |  | ptr(31:4) | | | ■ | c | 1 | 1 |  | index(31:0) | | |  | s | 1 | 1 | 0 | 0 |  | λ(26:2) | | | | | |
|  |  |  |  |  | | |  |  |  |  |  |  | | |  |  |  |  |  |  |  |  | | | | | |
| **ra** |  | 1 |  | ptr(31:4) | | | 0 | c | 1 | 1 |  | index(31:0) | | |  | s | r | w | x | d |  | λ(26:2) | | | | | |
|  |  |  |  |  | | |  |  |  |  |  |  | | |  |  |  |  |  |  |  |  | | | | | |
| **io pointer** |  | 1 |  | ptr(31:4) | | | 0 | 0 | 0 | 1 |  | index(31:0) | | |  | s | r | w | 0 | 1 |  | λ(26:2) | | | | | |

**Microarchitectural Registers:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | T |  | 31 |  | 4 | 3 | 2 | 1 | 0 |  | 31 |  | 0 |  | Flags | | | | |  | 24 |  |  |  |  | 0 |
| **got** |  | 1 |  | ptr(31:0) | | | | | | |  |  | | |  | 0 | 1 | 0 | 0 | 0 |  | λ(26:2) | | | | | |
|  |  |  |  |  | | | | | | |  |  | | |  |  |  |  |  |  |  |  | | | | | |
| **scratch (scr)** |  |  |  |  | | |  |  |  |  |  |  | | |  |  |  |  |  |  |  |  | | | | | |
|  |  |  |  |  | | |  |  |  |  |  |  | | |  |  |  |  |  |  |  |  | | | | | |
| **frame scratch (fs)** |  | 1 |  | past-frame-ptr | | | 0 | 0 | 0 | 0 |  | current-public (cp) | | |  |  |  |  |  |  |  | past-publ. (pp) | | | | | |
|  |  |  |  |  | | |  |  |  |  |  |  | | |  |  |  |  |  |  |  |  | | | | | |
|  |  |  |  |  | | |  |  |  |  |  |  | | |  |  |  |  |  |  |  |  | | | | | |
|  |  |  |  |  | | |  |  |  |  |  |  | | |  |  |  |  |  |  |  |  | | | | | |
|  |  |  |  |  | | |  |  |  |  |  |  | | |  |  |  |  |  |  |  |  | | | | | |

**FRAME OPERATIONS**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ❶ | | addi sp, sp,-32 |  | | addi sp, sp,-16 |  | addi sp, sp,-16 | |
| ❷ | | sw ra, 28(sp) |  | | sw ra, 12(sp) |  | sw ra, 12(sp) | |
| ❸ | | sw fp, 24(sp) | ❼ | | sw fp, 8(sp) | ❾ | sw fp, 8(sp) | |
|  | | sw s1, 20(sp) |  | | addi fp, sp,16 |  | addi fp, sp,16 | |
|  | | sw s2, 16(sp) | ❽ | | lw t0, 0(fp) |  | ● ● ● | |
| ❹ | | addi fp, sp,32 | ❽ | | lw t1, 4(fp) | ❿ | lw fp, 8(sp) | |
| ❺ | | sw t0, -20(fp) |  | | jal ra, label2 |  |  | |
| ❺ | | sw a0, -24(fp) |  | |  |  |  | |
| ❻ | | sw a4, 4(sp) |  | |  |  |  | |
| ❻ | | sw a5, 0(sp) |  | |  |  |  | |
|  | | jal ra, label1 |  | |  |  |  | |
|  | |  |  | |  |  |  | |
|  | |  |  | |  |  |  | |
| |  |  |  | | --- | --- | --- | |  | *Frame 1* | ❶ | |  |  |  | |  | ra-ptr | ❷ | |  | fp-ptr | ❸ | |  | fs-pp = 0 | ❸ | | 0 | a5 | ❻◂sp(0) ◂fs | | 4 | a4 | ❻ | | 8 | a0 | ❺ | | 12 | t0 | ❺ | | 16 | s2 |  | | 20 | s1 |  | | 24 | fp-ix | ❸ | | 28 | ra-ix | ❷ | |  |  | ◂fp(32) | | | | |  |  |  | | --- | --- | --- | |  | *Frame 2* |  | |  |  |  | |  | --- |  | |  | fp-ptr | ❼ | |  | fs-pp = 0 | ❼ | | 0 |  | ◂sp(0) ◂fs | | 4 |  |  | | 8 | fp-ix | ❼ | | 12 | ra-ix |  | |  |  | ◂fp(16) | | | | |  |  |  | | --- | --- | --- | |  | *Frame 3* |  | |  |  |  | |  | --- |  | |  | fp-ptr | ❾❿ | |  | fs-pp = 8 | ❾❿ | | 0 |  | ◂sp(0) | | 4 |  |  | | 8 | fp-ix | ❾❿ | | 12 | ra-ix |  | |  |  | ◂fp(16) | | | |
|  | | |  | | |  | | |
|  | | |  | | |  | | |
| ❶ | allocate a new stack frame object via addi on sp | | ❽ | When an access via fp is detected, where the index would be too big for the current stack frame (fp(index)+offset > fp(λ)), then fs is used for the memory access instead. The index for the fs access calculates as: fs(offset) = fp(index)+offset-fp(λ). if fs(offset) > fs(pp), an out of bounds exception is thrown. | | ❿ | | *How does fs-pp get restored?* |
| ❷ | store return index in designated “ra-spot” (aka. last element of frame). If ra was produced by a cross-object jump, ra-ptr is stored into the frame header. | |
| ❸ | store index of frame pointer (fp/s0) into designated “fp-spot” (aka. second to last element of frame). also stores the current value of fs (past-public) into the fp-pp spot. also stores the base-pointer of the frame-object into the fp-ptr spot. also writes fs(cp) to fs(pp), set fs(cp) = 0 and fs(ptr) = fp(ptr). | |
| ❹ | overwrite fp with current stack frame pointer | |  |  | |  | |  |
| ❺ | sw-instructions using fp shorten the current-public-space. If the store address is lower than the current-public index, the index is decremented accordingly. | |  |  | |  | |  |
| ❻ | sw-instructions using sp widen the current public-space. If the store address is higher than the current-public index and the store was executed using an argument register (a0-a7), the index is incremented accordingly. | |  |  | |  | |  |
|  | | |  | | |  | | |

**CODE SEGMENTATION**

Invariant: first element of Executable Objects is always a pointer to its Global Offset Table(?)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | | 31 | 30 | | 29 | | 28 | | 27 | | 26 | | 25 | | 24 |  | 2 | | 1 | | 0 | |
| *♦-4* |  | gc | | | s | | r | | w | | 1 | | 1 | | λ(24:2) | | | | 0 | | 0 | |
| *♦+0* |  | GOT | | | | | | | | | | | | | | | | | | | | |
| *♦+4* |  | INSTRUCTIONS | | | | | | | | | | | | | | | | | | | | |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  | PROCEDURE LINKAGE TABLE | | | | | | | | | | | | | | | | | | | | |
| *♦+λ-4* |  |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |  | 2 | 1 | 0 |
| gc | | 0 | 1 | 0 | 0 | 0 | λ(24:2) | | | 0 | 0 |
| resolver | | | | | | | | | | | |
| link map | | | | | | | | | | | |
| index | | | | | | | | | | | |
| index | | | | | | | | | | | |
| ● ● ● | | | | | | | | | | | |
| index | | | | | | | | | | | |

**User Mode Instructions**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Instruction** | | **rd** | **rs1** | **rs2** | **cr** | **imm** | **Decoder Decision** |
| lui | | rd | --- | --- | - | imm |  |
| auipc | | rd | --- | --- | - | imm |  |
| jal | | rd | --- | sp | ● | imm |  |
|  | |  |  |  |  |  |  |
| **jalr** | | *rd* | *rs1* | *---* | *-* | *imm* |  |
| A | jalr | rd | rs1 | sp | ● | imm | *always* |
| A | lgt | got | rs1 | --- | - | --- | *always (instead of nop)* |
|  | |  |  |  |  |  |  |
| bcc | | --- | rs1 | rs2 | - | imm |  |
|  | |  |  |  |  |  |  |
| **lb/bu/h/hu/w** | | *rd* | *rs1* | *---* | *-* | *---* |  |
| A | lb/bu/h/hu/w | rd | rs1 | fs | ● | --- | *if rs1 = fp or rs1 = sp* |
| B | lb/bu/h/hu/w | rd | rs1 | got | ● | --- | *otherwise* |
|  | |  |  |  |  |  |  |
| **sb/h/w** | | *---* | *rs1* | *rs2* | *-* | *imm* |  |
| A | loadmux | scr | rs1 | rs2 | ● | imm | *if sb and imm(0) = 1 or sh and imm(1) = 1 WRONG* |
| A1 | sb\_m/h\_m | --- | rs1 | scr | ● | imm |
| A2 | sb\_m/h\_m | fs | rs1 | scr | ● | imm | *if rs1 = sp* |
| B | sb/h/w | fs | rs1 | rs2 | ● | imm | *if rs1 = sp* |
| C | sb/h/w | --- | rs1 | rs2 | ● | imm | *otherwise* |
|  | |  |  |  |  |  |  |
| **addi** | | *rd* | *rs1* | *---* | *-* | *imm* |  |
| A | push | sp | sp | --- | ● | imm | *if rd = sp and rs1 = sp and imm > 0* |
| B | pop | sp | sp | --- | - | imm | *if rd = sp and rs1 = sp and imm < 0* |
| C | addi | rd | rs1 | --- | - | imm | *otherwise* |
|  |  |  |  |  |  |  |  |
| arithi | | rd | rs1 | --- | - | imm |  |
| arith | | rd | rs1 | rs2 | - | --- |  |
|  | |  |  |  |  |  |  |
| alc | | rd | rs1 | alc\_params | - | --- |  |
| alci | | rd | --- | alc\_params | - | imm |  |
| alc.d | | rd | rs1 | alc\_params | - | --- |  |
| alci.d | | rd | --- | alc\_params | - | imm |  |
| qsz | | rd | rs1 | --- | - | --- |  |
|  | |  |  |  |  |  |  |
|  | |  |  |  |  |  |  |

**Supervisor Mode Instructions:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Instruction** | **rd** | **rs1** | **rs2** | **cr** | **imm** | **Notes** |
| sb/h/w.r | --- | rs1 | rs2 | - | imm | *“store raw”, allows stores at any point in memory. Uses rs1 as base-ptr* |
| lb/bu/h/hu/w.r | rd | rs1 | --- | - | --- | *“load raw”, same as store raw* |
| dtp | rd | rs1 | --- | - | --- | *“data to pointer”, creates a pointer from data* |
| ptd | rd | rs1 | --- | - | --- | *“pointer to data”, extracts base address of pointer as data* |
| itd | rd | rs1 | --- | - | --- | *“index to data”, extracts index of pointer as data* |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

Problem: we only know if we need to box an immediate in execute. How do we handle instructions, which split into multiple nano-instructions in execute?

# DOKUMENTATION: ELF-FILES

“Executable and Linkable Format”-Files bestehen mindestens aus einem Header, einer “Program Header Table” und einer “Section Header Table”. Im Header werden Informationen über das ELF-File selbst gespeichert, wie z.B. die Prozessorarchitektur, für welche das Programm kompiliert wurde und die Positionen der PHT und der SHT in Relation zum File-Anfang. In einem Program Header werden Informationen gespeichert, die dem Betriebssystem angeben, wie viele und welche Arten von virtuellen Seiten für dieses Programm benötigt werden. In einem Section Header wird angegeben, in welche Einzelteile das Programm zerlegt wurde und ob noch mehr Informationen über das Programm im ELF-File zu finden sind (z.B. für relocatable Programme).

## Daten

Statische Daten werden von einem Compiler über Assemblerdirektiven immer so in die .data bzw. .rodata Sektionen abgelegt, sodass sie in der Symboltabelle des ELF-Files immer als Objekt mit seiner Größe eindeutig erkennbar sind.

|  |  |
| --- | --- |
| //C-Code  static char stringA[] = "hello world!"; | //C-Code  static const char stringB[] = "hello world!"; |
|  |  |
|  |  |
| #Resultierender Assembly-Code  .data  .type stringA, @object  stringA: .asciz "hello world!"  .size stringA, .-stringA | #Resultierender Assembly-Code  .rodata  .type stringB, @object  stringB: .asciz "hello world!"  .size stringB, .-stringB |
|  |  |
|  |  |
| //Section Header Table im erzeugten ELF-File  Section Headers:  [Nr] Name Type Address Offset Size EntSize Flags Link Info Align  ...  [ 5] .data PROGBITS 00002010 000003b4 0000000d 00000000 WA 0 0 4  [ 6] .rodata PROGBITS 00002020 000003c4 0000000d 00000000 A 0 0 4  ...  //Symbol Table im erzeugten ELF-File  Symbol table '.symtab' contains 60 entries:  Num: Value Size Type Bind Vis Ndx Name  ...  49: 00000000 13 OBJECT LOCAL DEFAULT 5 stringA  50: 00000000 13 OBJECT LOCAL DEFAULT 6 stringB  ... | |

Ein Zugriff auf solche statischen Daten kann in executables und muss in relocatables über die Global Offset Table (GOT) stattfinden. Angenommen ein Programm läge an der physikalischen Adresse 0x0 und seine zugehörige GOT an der Adresse 0x1000 und am Offset 8 der GOT stünde die Adresse für das Symbol stringA, dann würde mit folgenden Assembly befehlen auf diesen Eintrag zugegriffen werden.

|  |
| --- |
| auipc   t2, 0x1    # R\_RISCV\_GOT\_HI20 (symbol), R\_RISCV\_RELAX      lw      t2, 8(t2)  # R\_RISCV\_PCREL\_LO12\_I (auipc), R\_RISCV\_RELAX |

In einer executable können die Immediates für diese Befehlssequenz direkt befüllt werden, da der Abstand des Programms zur GOT schon beim Kompilieren des Programms bekannt ist. Bei einem relocatable Programm belässt der Compiler diese Immediates mit 0 und markiert die Befehle in der „Relocation Section“ als unaufgelöst. Sowohl die GOT als auch die .data oder .rodata Sektionen können vom Betriebssystem beim Laden des Programms an beliebige Stellen im Speicher platziert werden. Sind alle Sektionen platziert, kann der Dynamische Linker anhand der Tags der Einträge in der Relocation Section herausfinden, wie er die Immediates für die aufzulösenden Symbole zu berechnen hat. R\_RISCV\_GOT\_HI20 z.B. bedeutet, dass für diese Instruktion die obersten 20 Bits der Differenz aus Position der Instruktion und Position der GOT benötigt. Die Relax Tags sollen anzeigen, dass es je nach Positionierung möglich sein könnte, eine der beiden Instruktionen zu sparen falls z.B. Instruktion und GOT nah genug beieinander liegen.

## Code

Bla bla bla Procedure Linkage Table

|  |
| --- |
| #PROCEDURE LINKAGE TABLE#  00000080 <.plt>:  .plt  .pltR: auipc t2, %pcrel\_hi(.got.plt)  sub t1, t1, t3 # t1 = difference between caller and .pltR + 12  lw t3, %pcrel\_lo(.pltR)(t2) # t3 = addr(\_dl\_runtime\_resolve)  addi t1, t1, -44 # subtract size of .pltR (32) and jalr offset in caller (12)  addi t0, t2, %pcrel\_lo(.pltR) # t0 = start of .got  srli t1, t1, 2 # index of .plt entry in .got.plt  lw t0, 4(t0) # link map  jr t3  .plt0: auipc t3, %pcrel\_hi(functionA@.got.plt)  lw t3, %pcrel\_lo(.plt0)(t3)  jalr t1, t3  nop  .plt1: auipc t3, %pcrel\_hi(functionB@.got.plt)  lw t3, %pcrel\_lo(.plt1)(t3)  jalr t1, t3  nop  .plt2: ...  #GLOBAL OFFSET TABLE#  000010ac <.got.plt>:  .got.plt  .word 0xffffffff #to be filled with address of the dynamic resolver  .word 0x00000000 #to be filled with pointer to “link map”  .word 0x00000080 #func entry 0  .word 0x00000080 #func entry 1  .word 0x00000080 #func entry 2  ... |
|  |