|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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**

Ordinary

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 31 | 30 | 29 | | 28 |  | 2 | 1 | 0 |
| gc[[1]](#footnote-1) | | w[[2]](#footnote-2) | size(28:2) | | | | 0 | 0 |
|  | | | | | | | | |
|  | | | | | | | | |
| ... | | | | | | | | |
|  | | | | | | | | |
|  | | | | | | | | |

Frame

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 31 | 30 | 29 |  | 6 | | 5 | 4 | 3 | 2 | 1 | 0 |
| gc | | key(23:0) | | | r[[3]](#footnote-3) | | 1 | 1 | 1 | 1 | 1 |
| 00 | | old\_key | | | 0 | | 1 | 1 | 1 | 1 | 1 |
| ra-ptr? | | | | | | | | | | | |
| fp-eop! | | | | | | | | | | | |
| ra-ix! | | | | | | | | | | | |
| fp-ptr! | | | | | | | | | | | |
|  | | | | | | | | | | | |
| ... | | | | | | | | | | | |
|  | | | | | | | | | | | |

Data only

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 31 | 30 | 29 | | 28 |  | 2 | 1 | 0 |
| gc | | w | size(28:2) | | | | 0 | 1 |
|  | | | | | | | | |
|  | | | | | | | | |
| ... | | | | | | | | |
|  | | | | | | | | |
|  | | | | | | | | |

Code

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 31 |  | 2 | 1 | 0 |
| eoc(30:1) | | | 1 | 1 |
| eop(30:1) | | | 1 | 1 |
|  | | | | |
| ... | | | | |
|  | | | | |

Immediate (Primitive)

|  |  |  |
| --- | --- | --- |
| 31 |  | 0 |
| integer | | |

Immediate (Pointer)

|  |  |  |
| --- | --- | --- |
| 31 |  | 0 |
| ptr | | |
| ix | | |
| attr | | |

**POINTERS & DATA  
(in memory)**

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

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 31 |  | 3 | 2 | 1 | 0 |
| ord./code/d.o.-ptr: | ptr(31:4) | | 0 | 0 | 1 | 1 |

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

(*immediate (ptr) pointers* shall never be present in the register-file. pc pointers shall never be stored to memory, except in the hidden ra-ptr spot of stack-frames)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 31 |  | | 5 | 4 | 3 | 2 | 1 | 0 |
| io pointer: | dev | | size | g | 0 | 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**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | T |  | 31 |  | 0 |  |  |  |  |
| **data** | 0 |  | value(31:0) | | |  | alc\_addr |  | alc\_lim |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | T |  | 31 |  | 4 | 3 | 2 | 1 | 0 |  | 31 |  | 0 |  | 31 | 30 | 29 |  | 2 | 1 | 0 |
| **ordinary pointer** | 1 |  | ptr(31:4) | | | 0 | 0 | 0 | 0 |  | index(31:0) | | |  | 0 | 0 | size(29:2) | | | 0 | 0 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | T |  | 31 |  | 4 | 3 | 2 | 1 | 0 |  | 31 |  | 0 |  | 31 | 30 |  |  |  | 1 | 0 |
| **code pointer** | 1 |  | ptr(31:4) | | | 0 | 1 | 0 | 0 |  | eop(31:0) | | |  | 0 | eoc(30:1) | | | | | 0 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | T |  | 31 |  | 4 | 3 | 2 | 1 | 0 |  | 31 |  | 0 |  | 31 | 30 |  |  |  | 1 | 0 |
| **pc pointer** | 1 |  | ptr(31:4) | | | 1 | 0 | 0 | 0 |  | index(31:0) | | |  | c | eoc(30:1) | | | | | 0 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | T |  | 31 |  | 4 | 3 | 2 | 1 | 0 |  | 31 |  | 0 |  | 31 | 30 |  | 0 |
| **sp/fp** | 1 |  | base-ptr(31:4) | | | 0 | 0 | 0 | 1 |  | index(31:0) | | |  | c | eop(30:0) | | |

contents of sp (x2) and fp (x8) may be moved to another register, but stack-frames may only be allocated using sp and the public area may only be increased by operations on sp. Contents of the public area of past frames may only be accessed using fp.   
highest valid address for memory access using fp-types: fp(eop)  
lowest valid address for memory access using fp-types: sp

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | T |  | 31 |  | 4 | 3 | 2 | 1 | 0 |  | 31 |  | 0 |  | 31 |  |  |  |  |  | 0 |
| **copies of sp/fp** | 1 |  | base-ptr(31:4) | | | 0 | 0 | 1 | 0 |  | index(31:0) | | |  | key | | | | | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | T |  | 31 |  | 4 | 3 | 2 | 1 | 0 |  | 31 |  | 0 |  | 31 | 30 | 29 |  | 2 | 1 | 0 |
| **io pointer** | 1 |  | dev(27:0) | | | 1 | 1 | 0 | 0 |  | index(31:0) | | |  | g | size(29:2) | | | | 0 | 0 |

**FRAME OPERATIONS**

Dangling references are tracked by a key associated with registers containing pointers on stack frames. When such a register is supposed to be stored to memory, it will always be emitted into an immediate pointer, so the key-attribute of such pointers is not lost.  
Contents on a stack frame may only be accessed (apart from sp and fp) via a special stack pointers. These stack pointers are composed of a (unmodifiable) base pointer of the stack frame and a (also unmodifiable) index to where the local data is stored. The header field of a stack frame contains a key, which identifies the stack frames age. Only if the base pointer and the key of the register match the base pointer and the key it tries to load/store to, the access is granted. Otherwise, a dangling reference exception is thrown.  
The key is realized by a simple “pop counter”. With every deallocation operation of a stack frame (header), the pop counter is increased. It can only be decreased by the garbage collector, after a successful rearranging sweep over all stack frames. If the pop counter overflows, a stack overflow exception is thrown.

**Example: trying to load from a dangling reference**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ❶ | 3 Stack frames with keys and pointers on their content | ❷ | The last stack frame gets deallocated – s2 is dangling | ❸ | A new stack frame is allocated |
| |  |  |  | | --- | --- | --- | | 0 | 0 |  | | 4 | --- |  | |  | ● ● ● |  | | 10 |  |  | | 14 |  |  | | 18 |  | ◂a5(0,8,0) | |  | *padding* |  | | 20 | 3 |  | | 24 | --- |  | |  | ● ● ● |  | | 30 |  |  | | 34 |  |  | | 38 |  |  | | 3C |  | ◂t2(20,C,3) | | 40 | 3 |  | | 44 | --- |  | |  | ● ● ● |  | | 50 |  |  | | 54 |  |  | |  | *padding* |  | | 60 | 3 |  | | 64 | --- |  | |  | ● ● ● |  | | 70 |  |  | | 74 |  |  | | 78 |  | ◂s2(60,8,3) | | | |  |  |  | | --- | --- | --- | | 0 | 0 |  | | 4 | --- |  | |  | ● ● ● |  | | 10 |  |  | | 14 |  |  | | 18 |  | ◂a5(0,8,0) | |  | *padding* |  | | 20 | 3 |  | | 24 | --- |  | |  | ● ● ● |  | | 30 |  |  | | 34 |  |  | | 38 |  |  | | 3C |  | ◂t2(20,C,3) | | 40 | 3 |  | | 44 | --- |  | |  | ● ● ● |  | | 50 |  |  | | 54 |  |  | |  | *padding* |  | |  |  |  | |  |  |  | |  |  |  | |  |  |  | |  |  |  | |  |  | ◂s2(60,8,3) | | | |  |  |  | | --- | --- | --- | | 0 | 0 |  | | 4 | --- |  | |  | ● ● ● |  | | 10 |  |  | | 14 |  |  | | 18 |  | ◂a5(0,8,0) | |  | *padding* |  | | 20 | 3 |  | | 24 | --- |  | |  | ● ● ● |  | | 30 |  |  | | 34 |  |  | | 38 |  |  | | 3C |  | ◂t2(20,C,3) | | 40 | 3 |  | | 44 | --- |  | |  | ● ● ● |  | | 50 |  |  | | 54 |  |  | |  | *padding* |  | | 60 | 4 |  | | 64 | --- |  | |  | ● ● ● |  | | 70 |  |  | | 74 |  |  | | 78 |  | ◂s2(60,8,3) | | 7C |  | ◂t5(60,C,4) | | |
|  | |  | |  | |
| lw t0, 0(s2) and sw t0, 0(s2) would first load address 24 and compare its key with the key stored at that address. In this case, the keys would match and the load/store operation at memory address 2C can be operated. | | lw t0, 0(s2) and sw t0, 0(s2) would first load address 24 and compare its key with the key stored at that address. In this case, memory address 24 does not contain a key anymore, so the match is not successful, and an exception is thrown. | | lw t0, 0(s2) and sw t0, 0(s2) would first load address 24 and compare its key with the key stored at that address. In this case, the key in memory does not match the key of the register, which also causes an exception. | |

**Example: garbage collector freeing stack frame keys (work in progress)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ❶ | 4 Stack frames with keys and pointers on their content | ❷ | First Cycle | ❸ | Second Cycle |
| |  |  |  | | --- | --- | --- | | 0 | 0 |  | | 4 | --- |  | |  | ● ● ● |  | | 10 |  |  | | 14 |  |  | | 18 |  | ◂a5(0,8,0) | |  | *padding* |  | | 20 | 3 |  | | 24 | --- |  | |  | ● ● ● |  | | 30 |  |  | | 34 |  |  | | 38 |  |  | | 3C |  | ◂t2(20,C,3) | | 40 | 3 |  | | 44 | --- |  | |  | ● ● ● |  | | 50 |  |  | | 54 |  |  | |  | *padding* |  | | 60 | 4 |  | | 64 | --- |  | |  | ● ● ● |  | | 70 |  |  | | 74 |  |  | | 78 |  | ◂s2(60,8,3) | | 7C |  | ◂t5(60,C,4) | | | |  |  |  | | --- | --- | --- | | 0 | 0 |  | | 4 | --- |  | |  | ● ● ● |  | | 10 |  |  | | 14 |  |  | | 18 |  | ◂a5(0,8,0) | |  | *padding* |  | | 20 | 1 |  | | 24 | 3 |  | |  | ● ● ● |  | | 30 |  |  | | 34 |  |  | | 38 |  |  | | 3C |  | ◂t2(20,C,1) | | 40 | 1 |  | | 44 | 3 |  | |  | ● ● ● |  | | 50 |  |  | | 54 |  |  | |  | *padding* |  | | 60 | 4 |  | | 64 | --- |  | |  | ● ● ● |  | | 70 |  |  | | 74 |  |  | | 78 |  | ◂s2(60,8,1) | | 7C |  | ◂t5(60,C,4) | | | |  |  |  | | --- | --- | --- | | 0 | 0 |  | | 4 | --- |  | |  | ● ● ● |  | | 10 |  |  | | 14 |  |  | | 18 |  | ◂a5(0,8,0) | |  | *padding* |  | | 20 | 1 |  | | 24 | --- |  | |  | ● ● ● |  | | 30 |  |  | | 34 |  |  | | 38 |  |  | | 3C |  | ◂t2(20,C,1) | | 40 | 1 |  | | 44 | --- |  | |  | ● ● ● |  | | 50 |  |  | | 54 |  |  | |  | *padding* |  | | 60 | 2 |  | | 64 | 4 |  | |  | ● ● ● |  | | 70 |  |  | | 74 |  |  | | 78 |  | ◂s2(60,8,1) | | 7C |  | ◂t5(60,C,2) | | |
|  | |  | |  | |
| In this scenario, the stack frames with keys 3 and 4 can be bumped up to keys 1 and 2 respectively, to free up keys for future allocations. | | In a first iteration, the garbage collector would notice the available space between frame 0 and frame 3. As a consequence, the garbage collector would re-assign the lowest possible key to stack frames 3 and subsequently update all pointers with key 3 to key 1. While this collection cycle is in progress, keys 1 and 3 are both valid for this stack frame. This is marked by the gc-bit in the key field of the frame being set. After the cycle finished, the gc-bit will be cleared again and only key 1 will be valid from then on. | | In the second iteration, the garbage collector would notice the available space between frame 1 and 4. Just as the first iteration, the collector would bump key 4 and all its pointers to key 2. This process continues, until the end of stack is reached. If that happens, the current value of the counter csr is subtracted by the difference of the last frames original key and the last frames new key.  E.g. frame 4 was the last frame on stack and the csr had a value of 7, then the csr will be updated to 5. | |

# 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 4 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, 4(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

|  |
| --- |
| #GLOBAL OFFSET TABLE#  .got.plt  0: \_dl\_runtime\_resolve(?)  4: \_link\_map(?)  8: offset0  12: offset1  13: offset2  ...  #PROCEDURE LINKAGE TABLE#  .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 # offset of .plt entry in .got  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: ... |

**CODE SEGMENTATION**

Code is segmented into objects. We differentiate between two types of code:

**Executable**: Code that is self-contained and is not dependent on external libraries. The only way for program execution to leave an executable object (by itself) is via a system-call.

**Relocatable**: Code that is dynamically linked into a sort of operating system. These programs are able to call external functions. It is the responsibility of the supervisor to ensure that a reloc-object cannot call external functions it does not have access to and it only calls external functions at their designated entry points.

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 1 | size |  |
| 4 | *instr* | |  |
| 8 | *instr* | |  |
| C | *instr* | |  |
|  | ● ● ● | |  |
| XX | *instr* | |  |
| XX+4 | *instr* | | ◂size |

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 0 | size |  |
| 4 | eoc | |  |
| 8 | got | |  |
| C | *instr* | |  |
| 10 | *instr* | |  |
| 14 | *instr* | |  |
|  | ● ● ● | |  |
| YY-4 | *instr* | | ◂eoc |
| YY | *plt* | |  |
|  | ● ● ● | |  |
| XX | *plt* | |  |
| XX+4 | *plt* | | ◂size |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Instruction** | | **rd** | **rs1** | **rs2** | **cr** | **imm** | **Notes/Decoder Decision** |
| lui | | rd | --- | --- | - | imm |  |
| auipc | | rd | --- | --- | - | imm |  |
| jal | | rd | --- | sp | ● | imm |  |
| jalr | | rd | rs1 | sp | ● | imm |  |
| bcc | | --- | rs1 | rs2 | - | imm |  |
| lb/bu/h/hu/w | | rd | rs1 | --- | ● | --- |  |
|  | |  |  |  |  |  |  |
| **sb/h/w** | | *---* | *rs1* | *rs2* | *-* | *imm* |  |
| A | loadmux | rs2 | rs1 | rs2 | ● | imm | *if sb and imm(0) = 1 or sh and imm(1) = 1* |
| A | sb\_m/h\_m | rs2 | rs1 | rs2 | ● | imm |
| B | 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 | --- | - | --- |  |
| lgt | | rd | --- | --- | - | --- | *load global offset table* |
|  | |  |  |  |  |  |  |

1. reserved for garbage collector. [↑](#footnote-ref-1)
2. if bit is 0, object cannot be written to. [↑](#footnote-ref-2)
3. set to 1 if the ra-ptr field contains a valid pointer. [↑](#footnote-ref-3)