|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 | | | | |

**Zor**: “Objective RISC” Extension 2

Unprivileged:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 31 |  |  |  |  |  | 25 | 24 |  |  |  | 20 | 19 |  |  |  | 15 | 14 |  | 12 | 11 |  |  |  | 7 | 6 |  |  |  |  |  | 0 |  |  |  |  | |  |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | zero | | | | | rs1 | | | | | 0 | 0 | 0 | rd | | | | | 0 | 0 | 0 | 1 | 0 | 1 | 1 | ALC | | | | R | |
| size[13:2] | | | | | | | | | | | | sp | | | | | 0 | 1 | 0 | rd | | | | | 0 | 0 | 0 | 1 | 0 | 1 | 1 | ALCI | | | | I | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | zero | | | | | rs1 | | | | | 0 | 0 | 1 | rd | | | | | 0 | 0 | 0 | 1 | 0 | 1 | 1 | ALC.D | | | | R | |
| size[13:2] | | | | | | | | | | | | sp | | | | | 0 | 1 | 1 | rd | | | | | 0 | 0 | 0 | 1 | 0 | 1 | 1 | ALCI.D | | | | I | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | zero | | | | | rs1 | | | | | 1 | 0 | 0 | rd | | | | | 0 | 0 | 0 | 1 | 0 | 1 | 1 | QSZ | | | | R | |
|  | | | | | | |  |  |  |  |  |  | | | | |  |  |  |  | | | | |  |  |  |  |  |  |  |  | | | |  | |

Machine Mode:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 31 |  |  |  |  | 26 | 25 | 24 |  |  |  | 20 | 19 |  |  |  | 15 | 14 |  | 12 | 11 |  |  |  | 7 | 6 |  |  |  |  |  | 0 |  |  |  |  | |  |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | rs2 | | | | | rs1 | | | | | 0 | 0 | 0 | rd | | | | | 1 | 1 | 1 | 0 | 0 | 1 | 1 | DTP | | | | R | |
| 1 | 0 | 1 | 1 | 1 | 1 | 1 | zero | | | | | rs1 | | | | | 0 | 0 | 0 | rd | | | | | 1 | 1 | 1 | 0 | 0 | 1 | 1 | PTD | | | | R | |
| 1 | 1 | 0 | 1 | 1 | 1 | 1 | zero | | | | | rs1 | | | | | 0 | 0 | 0 | rd | | | | | 1 | 1 | 1 | 0 | 0 | 1 | 1 | ITD | | | | R | |
| 1 | 1 | 1 | 0 | 1 | 1 | 1 | zero | | | | | rs1 | | | | | 0 | 0 | 0 | rd | | | | | 1 | 1 | 1 | 0 | 0 | 1 | 1 | LW.X | | | | R | |
| 1 | 1 | 1 | 1 | 0 | 1 | 1 | zero | | | | | rs1 | | | | | 0 | 0 | 0 | rd | | | | | 1 | 1 | 1 | 0 | 0 | 1 | 1 | SW.X | | | | R | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | | | |  |  |  |  |  |  |  |  | | | |  | |

Misc:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| reg | alias | reg | alias |  | pseudo-instruction | implemented as |
| x0 | zero | x16 | a6 |  | push imm | alci sp, imm |
| x1 | ra | x17 | a7 |  | pop | lw/d sp, 4(sp) |
| x2 | sp | x18 | s2 |  |  |  |
| x3 | gp (got) | x19 | s3 |  |  |  |
| x4 | tp | x20 | s4 |  |  |  |
| x5 | t0 | x21 | s5 |  |  |  |
| x6 | t1 | x22 | s6 |  |  |  |
| x7 | t2 | x23 | s7 |  |  |  |
| x8 | s0 | x24 | s8 |  |  |  |
| x9 | s1 | x25 | s9 |  |  |  |
| x10 | a0 | x26 | s10 |  |  |  |
| x11 | a1 | x27 | s11 |  |  |  |
| x12 | a2 | x28 | t3 |  |  |  |
| x13 | a3 | x29 | t4 |  |  |  |
| x14 | a4 | x30 | t5 |  |  |  |
| x15 | a5 | x31 | t6 |  |  |  |

**OBJECTS**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | *Generic Header* | | | | | | | | | | | | | | | |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  | 31 | 30 | 29 | 28 | 27 | 26 |  |  |  |  | 5 | 4 | 3 | 2 | 1 | 0 |  |  |
|  |  | gc | | r | w | d | zero | | | | | | 1 | 1 | 1 | 1 | 1 |  |  |
|  |  | λ | | | | | | | | | | | | | c | f | 0 |  |  |
|  |  | gc: reserved bits for garbage collection  r: readable  w: writable  d: data only (no pointers allowed)  λ: λength of this object  f: is stack frame object?  c: color of stack frame if f = 1, else don’t care | | | | | | | | | | | | | | | |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | | Ordinary | | | | | | | | | | | | | | | | | | | | |  |  | |
|  |  | |  | | | | | | | | | | | | | | | | | | | | |  |  | |
|  |  | | 31 | 30 | | 29 | | 28 | | 27 | | 26 |  |  |  |  | 5 | | 4 | 3 | 2 | 1 | 0 |  |  | |
| *♦* |  | gc | | | r | | w | | d | | zero | | | | | | | 1 | | 1 | 1 | 1 | 1 |  |  |
|  |  | λ(30:2) | | | | | | | | | | | | | | | | | | | c | f | 0 |  |  |
|  |  |  | | | | | | | | | | | | | | | | | | | | | |  | 0 |
|  |  | ● ● ● | | | | | | | | | | | | | | | | | | | | | |  |  |
|  |  |  | | | | | | | | | | | | | | | | | | | | | |  | λ-4 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | | Executable | | | | | | | | | | | | | | | | | | | |  |  | |
|  |  | |  | | | | | | | | | | | | | | | | | | | |  |  | |
|  |  | | 31 | 30 | | 29 | | 28 | | 27 | | 26 |  |  |  |  | 5 | 4 | 3 | 2 | 1 | 0 |  |  | |
| *♦* |  | gc | | | 0 | | 0 | | 1 | | zero | | | | | | | 1 | 1 | 1 | 1 | 1 |  |  |
|  |  | λ(30:2) | | | | | | | | | | | | | | | | | | 0 | 0 | 0 |  |  |
|  |  | got | | | | | | | | | | | | | | | | | 0 | 1 | 1 | 1 |  |  |
|  |  | *instructions* | | | | | | | | | | | | | | | | | | | | |  | 0 |
|  |  | ● ● ● | | | | | | | | | | | | | | | | | | | | |  |  |
|  |  | *instructions* | | | | | | | | | | | | | | | | | | | | |  | λ-4 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | | Immediate (Primitive) | | | | | | | | | | | | | | | |  |  | |
|  |  | |  | | | | | | | | | | | | | | | |  |  | |
|  |  | | 31 | 30 | 29 | 28 | 27 | 26 |  |  |  |  |  | 4 | 3 | 2 | 1 | 0 |  |  | |
| *♦* |  | integer(30:0) | | | | | | | | | | | | | | | | 0 |  | 0 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | | Immediate (Pointer) | | | | | | | | | | | | | | | |  |  | |
|  |  | |  | | | | | | | | | | | | | | | |  |  | |
|  |  | | 31 | 30 | 29 | 28 | 27 | 26 |  |  |  |  |  | 4 | 3 | 2 | 1 | 0 |  |  | |
| *♦* |  | ptr(31:4) | | | | | | | | | | | | x | x | 1 | 1 | 1 |  | 0 |
|  |  | index(30:0) | | | | | | | | | | | | | | | | 0 |  | 4 |

|  |  |
| --- | --- |
| **Invariant 1:** | data-only objects which are not readable and not writable are implicitly executable |
| **Invariant 3:** | immediates can only be stored in 2^29-byte distance of its parent object-header |

**POINTERS & DATA  
(inside objects)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 31 |  | 1 | 0 |
| immediate (prim) pointer: | b | distance(30:2) | 0 | 1 |
|  | bit 31 of integer = b | | | |
| addr(immediate)-addr(object-header) = distance | | | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 31 | |  | 3 | 2 | 1 | 0 |
| immediate (ptr) pointer: | b | distance(30:3) | | | 0 | 1 | 1 |
|  | bit 31 of index = b | | | | | | |
| addr(immediate)-addr(object-header) = distance | | | | | | | |

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

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

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

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | T |  | 31 | |  | | | | | | | | | | 4 | 3 | 2 | | 1 | 0 |  | 30 |  | 0 |  | 29 |  | 0 |
| **data** |  | 0 |  | value(31:0) | | | | | | | | | | | | | | | | | |  | *zero* | | |  | *zero* | | |
|  |  |  |  |  |  | |  |  |  |  |  |  |  |  |  | |  |  |  | |  |  |  | | |  |  | | |
| **pointer** |  | 1 |  | ptr(31:4) | | | | | | | | | | | | | r | w | | d | 1 |  | index(30:0) | | |  | λ(31:2) | | |
|  |  |  |  |  | | | | | | | | | | | | |  |  | |  |  |  |  | | |  |  | | |
| **sp** |  | 1 |  | ptr(31:4) | | | | | | | | | | | | | 1 | 1 | | 0 | c |  | index(30:0) | | |  | λ(31:2) | | |
|  |  |  |  |  | | | | | | | | | | | | |  |  | |  |  |  |  | | |  |  | | |
| **ra** |  | 1 |  | ptr(31:4) | | | | | | | | | | | | | r | w | | 1 | c |  | index(30:0) | | |  | λ(31:2) | | |
|  |  |  |  |  | | | | | | | | | | | | |  |  | |  |  |  |  | | |  |  | | |
| **gp** |  | 1 |  | ptr(31:4) | | | | | | | | | | | | | 1 | 0 | | 0 | 1 |  | zero! | | |  | λ(31:2) | | |
|  |  |  |  |  | | | | | | | | | | | | | | | | | |  |  | | |  |  | | |
| **io pointer** |  | 1 |  | device | | | | | | | | | | | | | 1 | 1 | 1 | | 1 |  | index(30:0) | | |  | λ(31:2) | | |

Flags/Tags:  
**r** read access, **w** write access, **d** data only

**Microarchitectural Registers:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | T |  | 31 |  | 4 | 3 | 2 | 1 | 0 |  | 31 |  | 0 |  | Flags | | | |  | 29 |  | 0 |
| **alc-params (ap)** |  |  |  | ptr(31:0) | | | | | | |  |  | | |  |  |  |  |  |  |  | | |

**OBJECT INITIALIZATION**

#### Deprecated yet again ;-;

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **CLEAR ON ALC** | | | | | |  | **INIT WITH DEFAULTS** | | | | | |  | **DISALLOWED LAZY INIT** | | | | | |  | **ALLOWED LAZY INIT** | | | | | | |
|  | | | | | |  |  | | | | | |  |  | | | | | |  |  | | | | | |
| struct foo {  bool a = false;  int array[100];  int c = 0;  } bar; | | | | | |  | struct foo {  bool a = true;  int array[100];  int c = 1234;  } bar;  …  //forall  bar->array[i] = 7; | | | | | |  | struct foo {  bool a = true;  int array[100];  int c = 1234;  } bar;  …  bar->array = routine1(); | | | | | |  | struct foo {  bool a = true;  int array[100];  int c = 1234;  } bar;  …  bar->array = routine1(); | | | | | |
|  | | | | | |  |  | | | | | |  |  | | | | | |  |  | | | | | |
| routine0:  alci s0, 12  .init0:  clr s0  leb t0, s0  beq zero,t0,.init0  alci t1, 400  .init1:  clr t1, zero  leb t0, t1  beq zero,t0,.init1  sw t1, 4(s0) | | | | | |  | routine0:  alci s0, 12  li t1, 1  sb t1, 0(s0)  alci t2, 400  sw t2, 4(s0)  li t3, 1234  sw t3, 8(s0)  li t4, 7  .init1:  clr t2, t4  leb t5, t2  beq zero,t5,.init1 | | | | | |  | routine0:  alci s0, 12  li t1, 1  sb t1, 0(s0)  li t1, 1234  sw t1, 8(s0)  ...  jal ra, routine1  sw a0, 4(s0) | | | | | |  | routine0:  alci s0, 12  li t1, 1  sb t1, 0(s0)  li t1, 1234  sw t1, 4(s0)  ...  jal ra, routine1  sw a0, 8(s0) | | | | | |
|  | | | | | |  |  | | | | | |  |  | | | | | |  |  | | | | | |
|  | | | | | |  |  | | | | | |  |  | | | | | |  |  | | | | | |
|  | 12 | | | |  |  |  | 12 | | | |  |  |  | 12 | | | |  |  |  | 12 | | | |  |
|  | 0 | 0 | 0 | 0 | 0 |  |  | 0 | 0 | 0 | 1 |  |  |  | 0 | 0 | 0 | 1 |  |  |  | 0 | 0 | 0 | 1 |  |
|  | ptr | | | | 4 |  |  | ptr | | | | 4 |  |  | ptr | | | | 4 |  |  | 1234 | | | | 4 |
| 0 | | | | 8 |  |  | 1234 | | | | 8 |  |  | 1234 | | | | 8 |  |  | ptr | | | | 8 |
|  | | | |  |  |  |  | | | | 12 |  |  |  | | | | 12 |  |  |  | | | | 12 |
| 400 | | | |  |  |  | 400 | | | |  |  |  | 400 | | | |  |  |  | 400 | | | |  |
|  | 0 | | | | 0 |  |  | 7 | | | | 0 |  |  | a | | | | 0 |  |  | a | | | | 0 |
|  | 0 | | | | 4 |  |  | 7 | | | | 4 |  |  | b | | | | 4 |  |  | b | | | | 4 |
|  | ● ● ● | | | |  |  |  | ● ● ● | | | |  |  |  | ● ● ● | | | |  |  |  | ● ● ● | | | |  |
|  | 0 | | | | 392 |  |  | 7 | | | | 392 |  |  | y | | | | 392 |  |  | y | | | | 392 |
|  | 0 | | | | 396 |  |  | 7 | | | | 396 |  |  | z | | | | 396 |  |  | z | | | | 396 |
|  |  | | | |  |  |  |  | | | |  |  |  |  | | | |  |  |  |  | | | |  |
|  |  | | | |  |  |  |  | | | |  |  |  |  | | | |  |  |  |  | | | |  |
|  |  | | | |  |  |  |  | | | |  |  |  |  | | | |  |  |  |  | | | |  |
|  |  | | | |  |  |  |  | | | |  |  |  |  | | | |  |  |  |  | | | |  |
|  |  | | | |  |  |  |  | | | |  |  |  |  | | | |  |  |  |  | | | |  |
|  |  | | | |  |  |  |  | | | |  |  |  |  | | | |  |  |  |  | | | |  |
|  |  | | | |  |  |  |  | | | |  |  |  |  | | | |  |  |  |  | | | |  |
|  |  | | | |  |  |  |  | | | |  |  |  |  | | | |  |  |  |  | | | |  |
|  |  | | | |  |  |  |  | | | |  |  |  |  | | | |  |  |  |  | | | |  |
|  |  | | | |  |  |  |  | | | |  |  |  |  | | | |  |  |  |  | | | |  |
|  |  | | | |  |  |  |  | | | |  |  |  |  | | | |  |  |  |  | | | |  |
|  |  | | | |  |  |  |  | | | |  |  |  |  | | | |  |  |  |  | | | |  |
|  |  | | | |  |  |  |  | | | |  |  |  |  | | | |  |  |  |  | | | |  |
|  |  | | | |  |  |  |  | | | |  |  |  |  | | | |  |  |  |  | | | |  |
|  |  | | | |  |  |  |  | | | |  |  |  |  | | | |  |  |  |  | | | |  |
|  |  | | | |  |  |  |  | | | |  |  |  |  | | | |  |  |  |  | | | |  |

**CODE SEGMENTATION**

|  |  |
| --- | --- |
| **Executable-GOT linking** | |
|  |  |
| |  |  |  | | --- | --- | --- | |  | *Executable* |  | |  | λ = 140 |  | |  | got-ptr |  | | 0 | ##Code## |  | |  | ● ● ● |  | | 28 | lw t2, 16(gp) |  | |  | ● ● ● |  | | 44 | auipc ra, 0x0 |  | | 48 | jalr ra, 220(ra) |  | |  | ● ● ● |  | |  | ##PLT## |  | | 100 | .pltR: |  | |  | ● ● ● |  | | 120 | .plt0: |  | |  | ● ● ● |  | | 130 | .plt1: |  | |  | ● ● ● |  | | 13C |  | λ-4 | |  |  |  | | |  |  |  |  | | --- | --- | --- | --- | |  | *Global Offset Table* | | | |  | ro | λ = 14 |  | | 0 | *resolver* | |  | | 4 | *link map* | |  | | 8 | *printf* | |  | | C | *scanf* | |  | | 10 | *static str* | | λ-4 | |  |  | |  | |

Code-Objects can only be exited via the procedure linkage table (plt) which uses entries of the global offset table (got). Upon creation of the code-object or on the first try to exit the code object, the supervisor puts the pointer to the target code-object into the global offset table, if (and only if) the source code-object is allowed to jump to that target.

**User Mode Instructions (Single Cycle)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Instruction** | **rd** | **rs1** | **rs2** | **cr** | **imm** | **Notes** |
| lui | rd | --- | --- | - | imm |  |
| auipc | rd | --- | --- | - | imm |  |
| jal | rd | --- | sp | ● | imm |  |
|  |  |  |  |  |  |  |
| bcc | --- | rs1 | rs2 | - | imm |  |
|  |  |  |  |  |  |  |
| arithi | rd | rs1 | --- | - | imm |  |
| arith | rd | rs1 | rs2 | - | --- |  |
|  |  |  |  |  |  |  |
| lb/bu/h/hu | rd | rs1 | --- | ● | imm |  |
| sb/h/w | --- | rs1 | rs2 | ● | imm |  |
|  |  |  |  |  |  |  |
| alc | rd | rs1 | sp | ● | --- | *if rd = sp, also stores sp to index 4* |
| alci | rd | --- | sp | ● | imm | *if rd = sp, also stores sp to index 4* |
| alc.d | rd | rs1 | --- | - | --- | *zero, ra, gp and sp forbidden for rd* |
| alci.d | rd | --- | --- | - | imm | *zero, ra, gp and sp forbidden for rd* |
| qsz | rd | rs1 | --- | - | --- |  |
|  |  |  |  |  |  |  |

**User Mode Instructions (Multi Cycle)**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Instruction** | | **rd** | **rs1** | **rs2** | **cr** | **imm** | **Decision** |
| **jalr** | | *rd* | *rs1* | *---* | *-* | *imm* |  |
| A | jalr | rd | rs1 | sp | ● | imm | *always* |
| A | lgt | got | rs1 | --- | - | --- | *always (instead of nop)* |
|  | |  |  |  |  |  |  |
| **lw** | | *---* | *rs1* | *rs2* | *-* | *imm* |  |
| A | lw | rd | rs1 | --- | ● | imm | *always* |
| a | lw | rd | rs1 | --- | ● | imm | *needed if rdata is immediate pointer* |
|  | |  |  |  |  |  |  |

**Machine Mode Instructions:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Instruction** | **rd** | **rs1** | **rs2** | **cr** | **imm** | **Notes** |
| 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* |
| lw.x | rd | rs1 | --- | - | --- | *load data from linear memory* |
| sw.x | rd | rs1 | rs2 | - | --- | *store data to linear memory* |

# 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  ... |
|  |