RISC-V Instruction type

RISC-V (Reduced Instruction Set Computing -5) is an open-source instruction set architecture (ISA) that follows a simple and modular design for flexibility and efficiency in computing applications.

Its instruction set is categorized into different types based on their operation and format.

RISC-V has six primary instruction types: **R, I, S, B, U, and J**.

1) **R-Type (Register-Register)**

- Used for arithmetic and logic operations.

2) **I-Type (Immediate)**

- Used for immediate operations, load instructions, and control flow.

3) **S-Type (Store)**

- Used for store instructions.

4) **B-Type (Branch)**

- Used for conditional branch instructions.

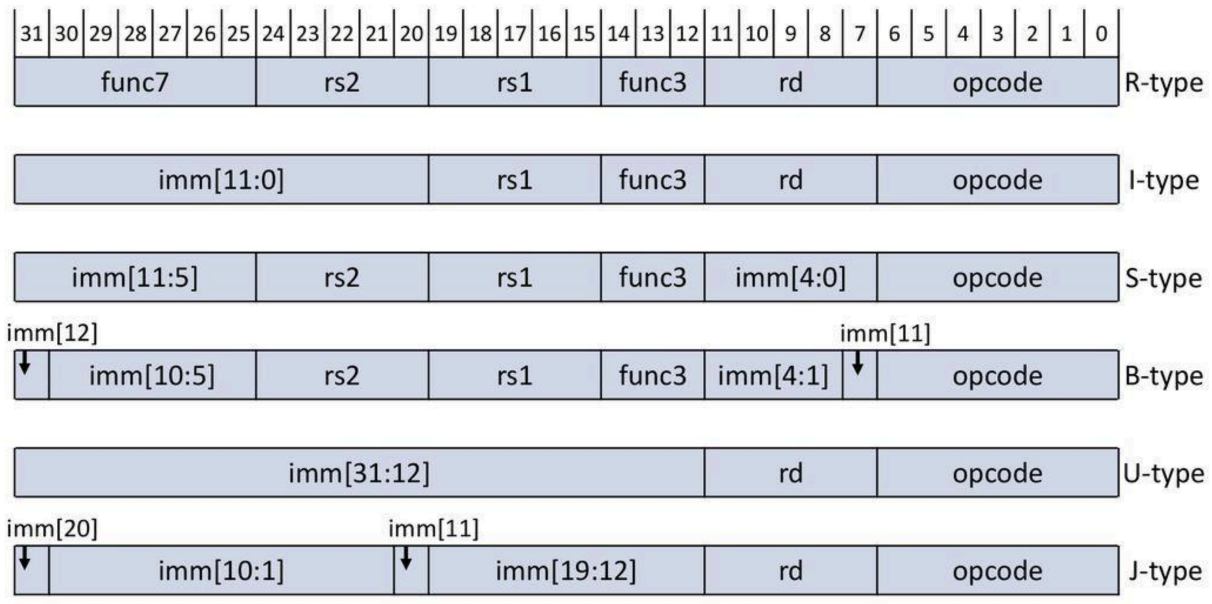
5) **U-Type (Upper Immediate)**

- Used for instructions that load a 20-bit immediate.

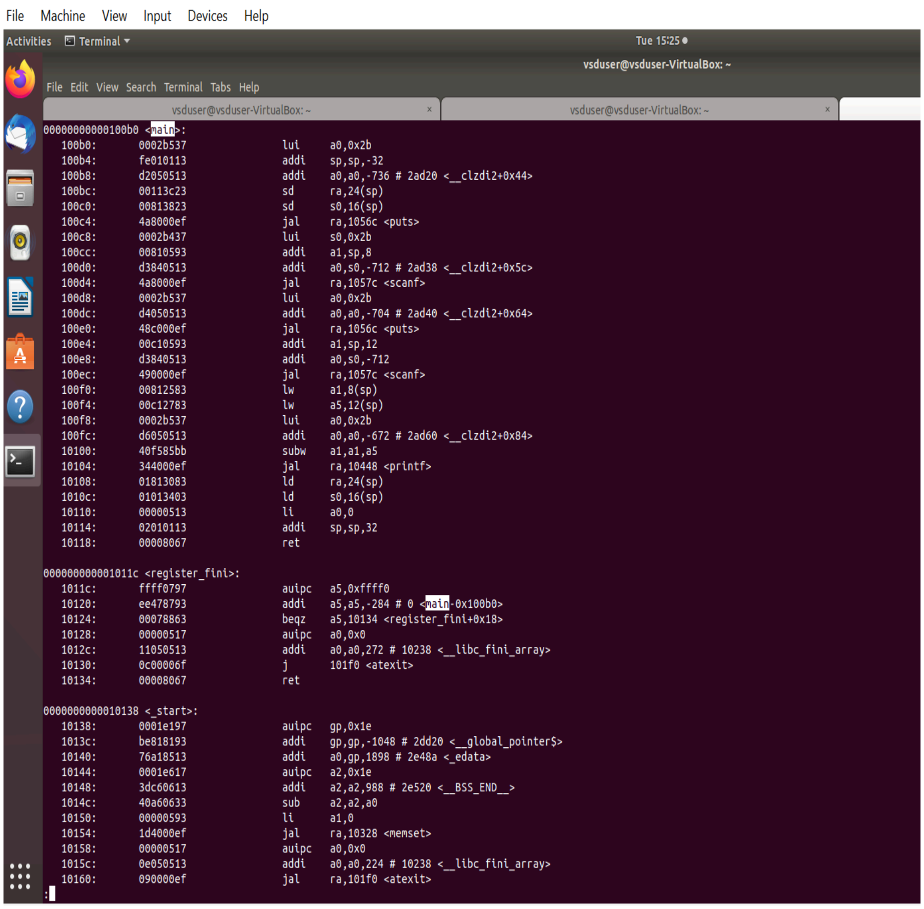
6) **J-Type (Jump)**

- Used for jump instructions.

RISC-V Instruction Formats


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
func7 | rs2 | rs1 | func3 | rd | opcode | R-type
imm[11:0] | rs1 | func3 | rd | opcode | I-type
imm[11:5] | rs2 | rs1 | func3 | imm[4:0] | opcode | S-type
imm[12] | imm[10:5] | rs2 | rs1 | func3 | imm[4:1] | imm[11] | opcode | B-type
imm[31:12] | rd | opcode | U-type
imm[20] | imm[10:1] | imm[11] | imm[19:12] | rd | opcode | J-type

RISCV-objdump of my application code,



```
00000000000100b0 <main>:
   100b0:       0002b537                lui     a0,0x2b
   100b4:       fe010113                addi    sp,sp,-32
   100b8:       d2050513                addi    a0,a0,-736 # 2ad20 <__clzdi2+0x44>
   100bc:       00113c23                sd      ra,24(sp)
   100c0:       00813823                sd      s0,16(sp)
   100c4:       4a8000ef                jal     ra,1056c <puts>
   100c8:       0002b437                lui     s0,0x2b
   100cc:       00810593                addi    a1,sp,8
   100d0:       d3840513                addi    a0,s0,-712 # 2ad38 <__clzdi2+0x5c>
   100d4:       4a8000ef                jal     ra,1057c <scanf>
   100d8:       0002b537                lui     a0,0x2b
   100dc:       d4050513                addi    a0,a0,-704 # 2ad40 <__clzdi2+0x64>
   100e0:       48c000ef                jal     ra,1056c <puts>
   100e4:       00c10593                addi    a1,sp,12
   100e8:       d3840513                addi    a0,s0,-712
   100ec:       490000ef                jal     ra,1057c <scanf>
   100f0:       00812583                lw      a1,8(sp)
   100f4:       00c12783                lw      a5,12(sp)
   100f8:       0002b537                lui     a0,0x2b
   100fc:       d6050513                addi    a0,a0,-672 # 2ad60 <__clzdi2+0x84>
   10100:       40f585bb                subw    a1,a1,a5
   10104:       344000ef                jal     ra,10448 <printf>
   10108:       01813083                ld      ra,24(sp)
   1010c:       01013403                ld      s0,16(sp)
   10110:       00000513                li      a0,0
   10114:       02010113                addi    sp,sp,32
   10118:       00008067                ret

000000000001011c <register_fini>:
   1011c:       ffff0797                auipc   a5,0xffff0
   10120:       ee478793                addi    a5,a5,-284 # 0 <main-0x100b0>
   10124:       00078863                beqz    a5,10134 <register_fini+0x18>
   10128:       00000517                auipc   a0,0x0
   1012c:       11050513                addi    a0,a0,272 # 10238 <__libc_fini_array>
   10130:       0c00006f                j       101f0 <atexit>
   10134:       00008067                ret

0000000000010138 <_start>:
   10138:       0001e197                auipc   gp,0x1e
   1013c:       be818193                addi    gp,gp,-1048 # 2dd20 <__global_pointer$>
   10140:       76a18513                addi    a0,gp,1898 # 2e48a <_edata>
   10144:       0001e617                auipc   a2,0x1e
   10148:       3dc60613                addi    a2,a2,988 # 2e520 <__BSS_END__>
   1014c:       40a60633                sub     a2,a2,a0
   10150:       00000593                li      a1,0
   10154:       1d4000ef                jal     ra,10328 <memset>
   10158:       00000517                auipc   a0,0x0
   1015c:       0e050513                addi    a0,a0,224 # 10238 <__libc_fini_array>
   10160:       090000ef                jal     ra,101f0 <atexit>
:
```

File Machine View Input Devices Help
Activities
Terminal
Tue 15:25
vsduser@vsduser-VirtualBox: ~
File Edit View Search Terminal Tabs Help
vsduser@vsduser-VirtualBox: ~
vsduser@vsduser-VirtualBox: ~

Examples,

1) lui a0,0x2b

Type : U-type

- **Immediate (imm[31:12])** = 0x2B << 12 = 0x2B000

  In binary: 0010 1011 0000 0000 0000 (20 bits)

- **Destination Register (rd)** = a0 = x10

  In binary: 01010 (5 bits)

- **Opcode** = 0110111 (fixed 7-bit opcode)

2) addi sp, sp , -32

Type: I-type

- **Immediate (imm[11:0])** = -32

-32 in signed 12-bit binary: 1111 1100 1110

- **Source Register (rs1)** = sp (x2)

  In binary: 00010 (5 bits)

- **funct3 for ADDI** = 000 (fixed 3-bit code)
- **Destination Register (rd)** = sp (x2)

  In binary: 00010 (5 bits)

- **Opcode for ADDI** = 0010011 (fixed 7-bit opcode)

3) addi a0,a0,-736

Type: I-type

- **Immediate (imm[11:0])** = -736

  -736 in signed 12-bit binary:1000 1100 1010

- **Source Register (rs1)** = a0 (x10)

  In binary: 01010 (5 bits)

- **funct3 for ADDI** = 000 (fixed 3-bit code)
- **Destination Register (rd)** = a0 (x10)

  In binary: 01010 (5 bits)

- **Opcode for ADDI** = 0010011 (fixed 7-bit opcode)

4) sd ra,24(sp)

Type: S-type

- **Immediate (imm[11:0])** = 24

  24 in binary (12-bit unsigned):0000 0010 0100

  imm[11:5] = 0000001

imm[4:0] = 00100

- **Source Register 2 (rs2)** = ra (x1)

  In binary: 00001 (5 bits)

- **Base Register (rs1)** = sp (x2)

  In binary: 00010 (5 bits)

- **funct3 for SD** = 011 (fixed 3-bit code)
- **Opcode for SD** = 0100011 (fixed 7-bit opcode)

5) sd s0,16(sp)

Type: S-type

- **Immediate (imm[11:0])** = 16

  16 in binary (12-bit unsigned):0000 0001 0110

  imm[11:5] = 0000000

  imm[4:0] = 10110

- **Source Register 2 (rs2)** = s0 (x8)

  In binary: 01000 (5 bits)

- **Base Register (rs1)** = sp (x2)

  In binary: 00010 (5 bits)

- **funct3 for SD** = 011 (fixed 3-bit code)
- **Opcode for SD** = 0100011 (fixed 7-bit opcode)

6) jal ra,1056c

Type: J-type (Jump and Link)

- 0x1056C = 0000 0001 0000 0101 0110 1100

Rearrange the 21-bit immediate into J-type format:

imm[20] = 0 (MSB)

imm[10:1] = 1010110110

imm[11] = 0

imm[19:12] = 00010000

- **Destination Register (rd)**=ra (x1)

  In binary:00001 (5 bits)

- **Opcode**: 1101111

7) lui s0,0x2b

Type : U-type

- **Immediate (imm[31:12])** = 0x2B << 12 = 0x2B000

  In binary: 0010 1011 0000 0000 0000 (20 bits)

- **Destination Register (rd)** = s0 (x8)

  In binary: 01000 (5 bits)

- **Opcode** = 0110111 (fixed 7-bit opcode)

8) addi a1,sp,8

Type: I-type

- **Immediate (imm[11:0])** = 8

  In binary: 000000001000

- **Source Register (rs1)** = sp (x2)

  In binary: 00010 (5 bits)

- **funct3 for ADDI** = 000 (fixed 3-bit code)

- **Destination Register (rd)** = a1 (x11)

  In binary: 01011 (5 bits)

- **Opcode for ADDI** = 0010011 (fixed 7-bit opcode)

9) addi a0,s0,-712

   Type: I-type

- **Immediate (imm[11:0])** = -712

  -712 in signed 12-bit binary:1000 1110 1110

- **Source Register (rs1)** = s0 (x8)

  In binary: 01000 (5 bits)

- **funct3 for ADDI** = 000 (fixed 3-bit code)
- **Destination Register (rd)** = a0 (x10)
  In binary: 01010 (5 bits)
- **Opcode for ADDI** = 0010011 (fixed 7-bit opcode)

10) lw a1,8(sp)

   Type: I-type (load instruction)

- **Immediate (imm[11:0])**: 8

  In binary:000000001000

- **Source Register (rs1)**: sp (x2)

  In binary:00010 (5 bits)

- **funct3**: 010
- **Destination Register (rd)**: a1 (x11)

  In binary: 01011

- **Opcode**: 0000011

11) lw a5,12(sp)

Type: I-type (load instruction)

- **Immediate (imm[11:0])**: 12

  In binary: 0000 0000 1100

- **Source Register (rs1)**: sp (x2)

  In binary:00010 (5 bits)

- **funct3**: 010
- **Destination Register (rd)**: a5 (x15)

  In binary: 01111

- **Opcode**: 0000011

12) subw a1,a1,a5

Type**:** R-type

- **funct7 for subw**: 0100000 (7 bits)
- **Source Register 2 (rs2)**: a5 (x15)
  In binary: 01111
- **Source Register 1 (rs1)**: a1 (x11)
  In binary: 01011
- **funct3**: 000 (3 bits)
- **Destination register (rd)**: a1 (x11)
  In binary: 01011
- **Opcode**: 0111011

13) jal ra,10448

Type: J-type (Jump and Link)

- 0x1056C = 0000 0000 0010 1000 1101 0000

Rearrange the 21-bit immediate into J-type format:

imm[20] = 0 (MSB)

imm[10:1] =000 1101 000

imm[11] = 1

imm[19:12] = 0000 0010

- **Destination Register (rd)**=ra (x1)

  In binary:00001 (5 bits)

- **Opcode**: 1101111

14) ld ra,24(sp)

Type: I-Type (Immediate-based Load Instruction)

- **Immediate (imm[11:0])**: 000000011000 (Binary for 24)
- **Source Register (rs1)**: sp (x2 - 00010)
- **funct3**: 011
- **Destination Register (rd)**: ra (x1 - 00001)
- **Opcode**: 0000011

15) ld s0,16(sp)

Type: I-Type

- **Immediate (imm[11:0])**: 000000010000 (Binary for 16)
- **Source Register (rs1)**: sp (x2 - 00010)
- **funct3**: 011
- **Destination Register (rd)**: s0 (x8 - 01000)
- **Opcode**: 0000011