RISC-V Reference Cheat Sheet

Instructions (a subset)

Name (Format,Op,funct3,funct7)	Syntax	Operation
add (R,51,0,0)	**add rd, rs1, rs2**	reg[rd] = reg[rs1] + reg[rs2]
Add immediate (I,19,0,-)	**addi rd, rs1, imm**	reg[rd] = reg[rs1] + `sext`(imm)
and (R,51,7,0)	**and rd, rs1, rs2**	reg[rd] = reg[rs1] & reg[rs2]
and immediate (I,19,7,-)	**andi rd, rs1, imm**	reg[rd] = reg[rs1] & `sext`(imm)
branch on equal (B,99,0,-)	**beq rs1,rs2,label**	PC=BTA if rs1 == rs2 else PC=PC+4
branch on not equal (B,99,1,-)	**bne rs1,rs2,label**	PC=BTA if rs1 != rs2 else PC=PC+4
divide (R,51,4,1)	**div rd, rs1, rs2**	reg[rd] = reg[rs1] /(reg[rs2] *(signed)*
divide unsigned (R,51,5,1)	**divu rd, rd1, rs2**	reg[rd] = reg[rs1] / reg[rs2] *(unsigned)*
jump-and-link (J,111,-,-)	**jal rd, label**	PC=JTA, reg[rd]=PC_{prev}+4
jump-and-link-register (I,103,0,-)	**jalr rd, rs1, imm**	PC = reg[rs1]+`sext`(imm), reg[rd]=PC_{prev}+4
load byte (I,3,0,-)	**lb rd, imm(rs1)**	reg[rd] = `sext`(mem[rs1 + `sext`(imm)])
load unsigned byte (I,3,4,-)	**lbu rd, imm(rs1)**	reg[rd] = mem[rs1 + `sext`(imm)]
load upper immediate (U,55,-,-)	**lui rd, imm**	reg[rd] = `concat`(imm, ”000000000000”)
load word (I,3,2,-)	**lw rd, imm(rs1)**	reg[rd] = mem[rs1 + `sext`(imm)]
multiply (R,51,0,1)	**mul rd, rs1, rd2**	reg[rd] = reg[rs1] * reg[rs2]
or (R,51,6,0)	**or rd, rs1, rd2**	reg[rd] = reg[rs1] | reg[rs2]
or immediate (I,19,6,-)	**ori rd, rs1, imm**	reg[rd] = reg[rs1] | `sext`(imm)
store byte (S,35,0,-)	**sb rs2, imm(rs1)**	mem[rs1 + `sext`(imm)] = rs2
shift left logical (R,51,1,0)	**sll rd, rs1, rs2**	reg[rd] = reg[rs1] « reg[rs2]
shift left logical immediate (R,19,1,0)	**slli rd, rs1, shamt**	reg[rd] = reg[rs1] « shamt
set less than (R,51,2,0)	**slt rd, rs1, rs2**	reg[rd] = 1 if reg[rs1]<reg[rs2] *(signed)*
set less than immediate (I,19,2,-)	**slti rd, rs1, imm**	reg[rd] = 1 if reg[rs1]<`sext`(imm) *(signed)*
set less than imm. unsigned (I,19,3,-)	**sltiu rd, rs1, (imm**	reg[rd] = 1 if reg[rs1]<`sext`(imm) *(unsigned)*
set less than unsigned (R,51,3,0)	**sltu rd, rs1, rs2**	reg[rd] = 1 if reg[rs1]<reg[rs2] *(unsigned)*
shift right arithmetic (R,51,5,32)	**sra rd, rs1, rs2**	reg[rd] = reg[rs1] » reg[rs2]
shift right arithmetic imm. (R,19,5,32)	**srai rd, rs1, shamt**	reg[rd] = reg[rs1] » shamt
shift right logical (R,51,5,0)	**srl rd, rs1, rs2**	reg[rd] = reg[rs1] » reg[rs2]
shift right logical imm. (R,19,5,0)	**srli rd, rs1, shamt**	reg[rd] = reg[rs1] » shamt
subtract (R,51,0,32)	**sub rd, rs1, rd2**	reg[rd] = reg[rs1] - reg[rs2]
store word (S,35,2,-)	**sw rs2, imm(rs1)**	mem[reg[rs1] + `sext`(timm)] = reg[rs2]
exclusive-or (R,51,4,0)	**xor rd, rs1, rd2**	reg[rd] = reg[rs1] ^ reg[rs2]
exclusive-or imm. (I,19,4,-)	**xori rd, rs1, imm**	reg[rd] = reg[rs1] ^ `sext`(imm)

Registers

Name	#	Usage
zero	0	Always 0
ra	1	Return address
sp	2	Stack pointer
gp	3	Global pointer
tp	4	Thread pointer
t0-t2	5-7	Temporary
s0-s1	8-9	Saved
a0-a7	10-17	Arguments
s2-s11	18-27	Saved
t3-t6	28-31	Temporary

Notes

- All numbers are in the decimal format
- PC_{prev} is the PC prio to the jump
- `sext()` extends and returns a 32-bit 2's-complement value
- `concat()` concatinates two bitstrings into a 32-bit value
- Subscripts of an integer X, e.g., $imm_{4:1|11}$, means a certain bitstring contains the bits 4,3,2,1 followed by bit 11 of X (in the example, imm).
- BTA = PC + signext(imm)
- JTA=`concat`(PC+signext(imm), ”0”)
- *signed* and *unsigned* means that the operands are treated as positive or negative (signed) and only positive(unsigned) respectively
- shamt is an abbreviation for ”shift amount” and decides by how much something is shifted (see R-format)

Contact

By Artur Podobas at the Royal Institute of Technology, Stockholm, Sweden

If you find any errors or want to give feedback, write to podobas@kth.se

Instruction Formats

# bits	*31 ... 25*	*24 … 20*	*19 ... 15*	*14 ... 12*	*11 ... 7*	*6 ... 0*
R-format	func7	rs2 / shamt	rs1	funct3	rd	op
I-format	$imm_{11:0}$		rs1	funct3	rd	op
S-format	$imm_{11:5}$	rs2	rs1	funct3	$imm_{4:0}$	op
B-format	$imm_{12\|10:5}$	rs2	rs1	funct3	$imm_{4:1\|11}$	op
U-format	$imm_{31:12}$				rd	op
J-format	$imm_{20\|10:1\|11\|19:12}$				rd	op