RV32I Reference Card

CS-173 Fundamentals of Digital Systems

12th June 2024

1 Assembler directives

Directive	Effect
`.text`	Store subsequent instructions at next available address in *text* segment
`.data`	Store subsequent items at next available address in *data* segment
`.asciiz`	Store string followed by null-terminator in `.data` segment
`.byte`	Store listed values as 8-bit bytes
`.word`	Store listed values as 32-bit words

2 Registers

Register	Mnemonic	Description
`x0`	`zero`	Hard-wired zero
`x1`	`ra`	Return Address
`x2`	`sp`	Stack Pointer
`x3`	`gp`	Global Pointer
`x4`	`tp`	Thread Pointer
`x5`	`t0`	Temporary/alternate link register
`x6–7`	`t1–2`	Temporaries
`x8`	`s0/fp`	Saved register/Frame Pointer
`x9`	`s1`	Saved register
`x10–11`	`a0–1`	Function Arguments/return values
`x12–17`	`a2–7`	Function Arguments
`x18–27`	`s2–11`	Saved registers
`x28–31`	`t3–6`	Temporaries
`pc`		Program counter

3 Instruction types

	31 25	24 20	19 15	14 12	11 7	6 0	
R	funct7	rs2	rs1	funct3	rd	opcode	Register-Register
I	imm[11:0]		rs1	funct3	rd	opcode	Register-Immediate
I	funct7	imm[4:0]	rs1	funct3	rd	opcode	Register-Immediate Shift
S	imm[11:5]	rs2	rs1	funct3	imm[4:0]	opcode	Store
B	imm[12|10:5]	rs2	rs1	funct3	imm[4:1|11]	opcode	Branch
U	imm[31:12]				rd	opcode	Upper Immediate
J	imm[20|10:1|11|19:12]				rd	opcode	Jump

4 Instructions

Instruction		Pseudocode	Type	funct7	funct3	opcode
Shift						
sll	rd,rs1,rs2	rd $\leftarrow$ rs1 $\ll$ rs2	R	0x00	0x1	0x33
slli	rd,rs1,imm	rd $\leftarrow$ rs1 $\ll$ imm	I	0x00	0x1	0x13
srl	rd,rs1,rs2	rd $\leftarrow$ rs1 $\gg_u$ rs2	R	0x00	0x5	0x33
srli	rd,rs1,imm	rd $\leftarrow$ rs1 $\gg_u$ imm	I	0x00	0x5	0x13
sra	rd,rs1,rs2	rd $\leftarrow$ rs1 $\gg_s$ rs2	R	0x20	0x5	0x33
srai	rd,rs1,imm	rd $\leftarrow$ rs1 $\gg_s$ imm	I	0x20	0x5	0x13
Arithmetic						
add	rd,rs1,rs2	rd $\leftarrow$ rs1 + rs2	R	0x00	0x0	0x33
addi	rd,rs1,imm	rd $\leftarrow$ rs1 + sext(imm)	I		0x0	0x13
sub	rd,rs1,rs2	rd $\leftarrow$ rs1 $-$ rs2	R	0x20	0x0	0x33
lui	rd,imm	rd $\leftarrow$ imm $\ll$ 12	U			0x37
auipc	rd,imm	rd $\leftarrow$ pc + (imm $\ll$ 12)	U			0x17
Logical						
xor	rd,rs1,rs2	rd $\leftarrow$ rs1 $\wedge$ rs2	R	0x00	0x4	0x33
xori	rd,rs1,imm	rd $\leftarrow$ rs1 $\wedge$ sext(imm)	I		0x4	0x13
or	rd,rs1,rs2	rd $\leftarrow$ rs1 $\mid$ rs2	R	0x00	0x6	0x33
ori	rd,rs1,imm	rd $\leftarrow$ rs1 $\mid$ sext(imm)	I		0x6	0x13
and	rd,rs1,rs2	rd $\leftarrow$ rs1 & rs2	R	0x00	0x7	0x33
andi	rd,rs1,imm	rd $\leftarrow$ rs1 & sext(imm)	I		0x7	0x13
Compare						
slt	rd,rs1,rs2	rd $\leftarrow$ rs1 $<_s$ rs2	R	0x00	0x2	0x33
slti	rd,rs1,imm	rd $\leftarrow$ rs1 $<_s$ sext(imm)	I		0x2	0x13
sltu	rd,rs1,rs2	rd $\leftarrow$ rs1 $<_u$ rs2	R	0x00	0x3	0x33
sltiu	rd,rs1,imm	rd $\leftarrow$ rs1 $<_u$ sext(imm)	I		0x3	0x13
Branch						
beq	rs1,rs2,imm	pc $\leftarrow$ pc + sext(imm $\ll$ 1), if rs1 = rs2	B		0x0	0x63
bne	rs1,rs2,imm	pc $\leftarrow$ pc + sext(imm $\ll$ 1), if rs1 $\neq$ rs2	B		0x1	0x63
blt	rs1,rs2,imm	pc $\leftarrow$ pc + sext(imm $\ll$ 1), if rs1 $<_s$ rs2	B		0x4	0x63
bge	rs1,rs2,imm	pc $\leftarrow$ pc + sext(imm $\ll$ 1), if rs1 $\geq_s$ rs2	B		0x5	0x63
bltu	rs1,rs2,imm	pc $\leftarrow$ pc + sext(imm $\ll$ 1), if rs1 $<_u$ rs2	B		0x6	0x63
bgeu	rs1,rs2,imm	pc $\leftarrow$ pc + sext(imm $\ll$ 1), if rs1 $\geq_u$ rs2	B		0x7	0x63
Jump and link						
jal	rd,imm	rd $\leftarrow$ pc + 4	J			0x6F
		pc $\leftarrow$ pc + sext(imm $\ll$ 1)				
jalr	rd,rs1,imm	rd $\leftarrow$ pc + 4	I		0x0	0x67
		pc $\leftarrow$ (rs1 + sext(imm)) & ($\sim$1)				
Load						
lb	rd,imm(rs1)	rd $\leftarrow$ sext(mem[rs1 + sext(imm)][7 : 0])	I		0x0	0x03
lbu	rd,imm(rs1)	rd $\leftarrow$ zext(mem[rs1 + sext(imm)][7 : 0])	I		0x4	0x03
lh	rd,imm(rs1)	rd $\leftarrow$ sext(mem[rs1 + sext(imm)][15 : 0])	I		0x1	0x03
lhu	rd,imm(rs1)	rd $\leftarrow$ zext(mem[rs1 + sext(imm)][15 : 0])	I		0x5	0x03
lw	rd,imm(rs1)	rd $\leftarrow$ mem[rs1 + sext(imm)]	I		0x2	0x03
Store						
sb	rs2,imm(rs1)	mem[rs1 + sext(imm)] $\leftarrow$ rs2[7 : 0]	S		0x0	0x23
sh	rs2,imm(rs1)	mem[rs1 + sext(imm)] $\leftarrow$ rs2[15 : 0]	S		0x1	0x23
sw	rs2,imm(rs1)	mem[rs1 + sext(imm)] $\leftarrow$ rs2	S		0x2	0x23

Pseudoinstruction		Translation	Meaning
li	rd,imm	*Various*	Load immediate
la	rd,imm	auipc rd,imm[31:12]	Load address
		addi rd,rd,imm[11:0]	
j	imm	jal x0,imm	Jump
nop		addi x0,x0,0	No operation

$\sim$	Bitwise NOT	$\ll$	Left shift	mem	Memory access
&	Bitwise AND	$\gg$	Right shift	sext	Sign extend
$\mid$	Bitwise OR	op_s	Signed *op*eration	zext	Zero extend
$\wedge$	Bitwise XOR	op_u	Unsigned *op*eration		