RVisio RISC-V RV32IM podsetnik

Ime i prezime studenta: _____ Potpis studenta: _____

Potpis dežurnog: _____ Datum ispita: _____

Oznake

OZNAKA	ZNAČENJE
`imm`	konstanta iz instrukcije, uvek proširena znakom na 32 bita (osim shamt i uimm, 0..31)
`sext / zext`	proširenje znakom / nulama
`pc`	adresa instrukcije koja se izvršava

Aritmetika

INSTRUKCIJA	TIP	OPIS	ZNAČENJE
`add rd, rs1, rs2`	R	sabiranje	`rd = rs1 + rs2`
`sub rd, rs1, rs2`	R	oduzimanje	`rd = rs1 − rs2`
`addi rd, rs1, imm`	I	sabiranje sa konstantom	`rd = rs1 + imm`
`slt rd, rs1, rs2`	R	manje od, označeno	`rd = (rs1 < rs2) ? 1 : 0`
`slti rd, rs1, imm`	I	manje od konstante, označeno	`rd = (rs1 < imm) ? 1 : 0`
`sltu rd, rs1, rs2`	R	manje od, neoznačeno	`rd = (rs1 < rs2) ? 1 : 0`
`sltiu rd, rs1, imm`	I	manje od konstante, neoznačeno (imm se prvo proširi znakom)	`rd = (rs1 < imm) ? 1 : 0`
`lui rd, imm20`	U	gornjih 20 bita	`rd = imm20 << 12`
`auipc rd, imm20`	U	PC + gornjih 20 bita	`rd = pc + (imm20 << 12)`

Logika i pomeranje

INSTRUKCIJA	TIP	OPIS	ZNAČENJE
`and rd, rs1, rs2`	R	AND	`rd = rs1 & rs2`
`or rd, rs1, rs2`	R	OR	`rd = rs1 | rs2`
`xor rd, rs1, rs2`	R	XOR	`rd = rs1 ^ rs2`
`andi rd, rs1, imm`	I	AND sa konstantom	`rd = rs1 & imm`
`ori rd, rs1, imm`	I	OR sa konstantom	`rd = rs1 | imm`
`xori rd, rs1, imm`	I	XOR sa konstantom	`rd = rs1 ^ imm`
`sll rd, rs1, rs2`	R	logičko pomeranje levo	`rd = rs1 << rs2[4:0]`
`srl rd, rs1, rs2`	R	logičko pomeranje desno (ulaze nule)	`rd = rs1 >> rs2[4:0]`
`sra rd, rs1, rs2`	R	aritmetičko pomeranje desno (ponavlja se znak)	`rd = rs1 >> rs2[4:0]`
`slli rd, rs1, shamt`	I	pomeranje levo za konstantu	`rd = rs1 << shamt`
`srli rd, rs1, shamt`	I	logičko desno za konstantu	`rd = rs1 >> shamt`
`srai rd, rs1, shamt`	I	aritmetičko desno za konstantu	`rd = rs1 >> shamt`

shamt je neoznačena konstanta 0..31.

Čitanje i upis u memoriju

INSTRUKCIJA	TIP	OPIS	ZNAČENJE
`lw rd, imm(rs1)`	I	čitanje reči (32 b)	`rd = mem32[rs1 + imm]`
`lh rd, imm(rs1)`	I	čitanje polureči, označeno	`rd = sext(mem16[rs1 + imm])`
`lhu rd, imm(rs1)`	I	čitanje polureči, neoznačeno	`rd = zext(mem16[rs1 + imm])`
`lb rd, imm(rs1)`	I	čitanje bajta, označeno	`rd = sext(mem8[rs1 + imm])`
`lbu rd, imm(rs1)`	I	čitanje bajta, neoznačeno	`rd = zext(mem8[rs1 + imm])`
`sw rs2, imm(rs1)`	S	upis reči	`mem32[rs1 + imm] = rs2`
`sh rs2, imm(rs1)`	S	upis polureči	`mem16[rs1 + imm] = rs2[15:0]`
`sb rs2, imm(rs1)`	S	upis bajta	`mem8[rs1 + imm] = rs2[7:0]`

sext = proširenje znakom (bit znaka se kopira u više bitove), zext = proširenje nulama.

Grananja i skokovi

INSTRUKCIJA	TIP	OPIS	ZNAČENJE
`beq rs1, rs2, imm`	B	skok ako je jednako	`if (rs1 == rs2) pc = pc + imm`
`bne rs1, rs2, imm`	B	skok ako nije jednako	`if (rs1 != rs2) pc = pc + imm`
`blt rs1, rs2, imm`	B	skok ako je manje, označeno	`if (rs1 < rs2) pc = pc + imm`
`bge rs1, rs2, imm`	B	skok ako je veće ili jednako, označeno	`if (rs1 >= rs2) pc = pc + imm`
`bltu rs1, rs2, imm`	B	skok ako je manje, neoznačeno	`if (rs1 < rs2) pc = pc + imm`
`bgeu rs1, rs2, imm`	B	skok ako je veće ili jednako, neoznačeno	`if (rs1 >= rs2) pc = pc + imm`
`jal rd, imm`	J	skok i povratna adresa	`rd = pc + 4; pc = pc + imm`
`jalr rd, imm(rs1)`	I	skok na adresu iz registra	`rd = pc + 4; pc = (rs1 + imm) & ~1`

pc je adresa instrukcije skoka, a imm je pomeraj u bajtovima (uvek paran): ±4 KiB za grananja, ±1 MiB za jal. U asembleru se umesto imm najčešće piše labela ili adresa, a asembler sam izračuna imm = adresa − pc (npr. bne x3, x0, -24).

RISC-V RV32IM podsetnik

Ime i prezime studenta: _____ Potpis studenta: _____

Potpis dežurnog: _____ Datum ispita: _____

Množenje i deljenje (M)

INSTRUKCIJA	TIP	OPIS	ZNAČENJE
`mul rd, rs1, rs2`	R	proizvod, nižih 32 bita	`rd = (rs1 × rs2)[31:0]`
`mulh rd, rs1, rs2`	R	viših 32 bita, označeno × označeno	`rd = (rs1 × rs2)[63:32]`
`mulhsu rd, rs1, rs2`	R	viših 32 bita, označeno × neoznačeno	`rd = (rs1 × rs2)[63:32]`
`mulhu rd, rs1, rs2`	R	viših 32 bita, neoznačeno × neoznačeno	`rd = (rs1 × rs2)[63:32]`
`div rd, rs1, rs2`	R	deljenje, označeno	`rd = rs1 / rs2`
`divu rd, rs1, rs2`	R	deljenje, neoznačeno	`rd = rs1 / rs2`
`rem rd, rs1, rs2`	R	ostatak, označeno	`rd = rs1 % rs2`
`remu rd, rs1, rs2`	R	ostatak, neoznačeno	`rd = rs1 % rs2`

Samo jezgra sa M ekstenzijom (RV32IM); FemtoRV32 Quark je RV32I. Deljenje nulom ne izaziva izuzetak: div daje -1, rem daje rs1.

Sistem, prekidi i CSR

INSTRUKCIJA	TIP	OPIS	ZNAČENJE
`csrrw rd, csr, rs1`	I	zameni CSR	`rd = csr; csr = rs1`
`csrrs rd, csr, rs1`	I	postavi bitove	`rd = csr; csr |= rs1`
`csrrc rd, csr, rs1`	I	obriši bitove	`rd = csr; csr &= ~rs1`
`csrrwi / csrrsi / csrrci`	I	isto, sa konstantom 0..31	`rd = csr; csr op= uimm5`
`mret`	I	povratak iz prekida	`pc = mepc`
`ecall`	I	poziv okruženja	`izuzetak (mcause 11)`
`ebreak`	I	prekidna tačka; u simulatoru kraj programa	`halt`
`fence`	I	redosled pristupa memoriji	`nop u simulatoru`

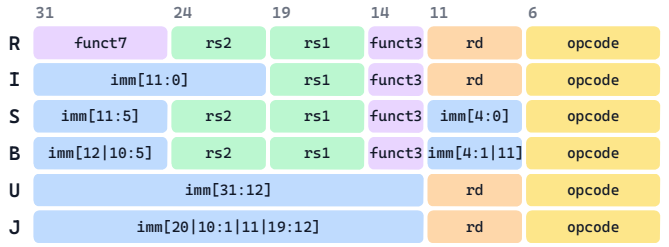
Imena CSR-ova: mstatus, misa, mie, mtvec, mscratch, mepc, mcause, mtval, mip, cycle, cycleh, time, instret, instreth.

Pseudo-instrukcije

PSEUDO	OPIS	ASEMBLER PRAVI
`nop`	ništa	`addi zero, zero, 0`
`li rd, imm`	-2048..2047	`addi rd, zero, imm`
`li rd, imm`	ostale konstante	`lui rd, %hi(imm)` `addi rd, rd, %lo(imm)`
`la rd, symbol`	učitaj adresu	`auipc rd, %hi(sym-pc)` `addi rd, rd, %lo(sym-pc)`
`mv rd, rs`	kopiraj registar	`addi rd, rs, 0`
`not rd, rs`	komplement	`xori rd, rs, -1`
`neg rd, rs`	promena znaka	`sub rd, zero, rs`
`seqz rd, rs`	rd = (rs == 0)	`sltiu rd, rs, 1`
`snez rd, rs`	rd = (rs != 0)	`sltu rd, zero, rs`
`sltz rd, rs`	rd = (rs < 0)	`slt rd, rs, zero`
`sgtz rd, rs`	rd = (rs > 0)	`slt rd, zero, rs`
`beqz rs, label`	rs == 0	`beq rs, zero, label`
`bnez rs, label`	rs != 0	`bne rs, zero, label`
`bltz rs, label`	rs < 0	`blt rs, zero, label`
`bgez rs, label`	rs >= 0	`bge rs, zero, label`
`blez rs, label`	rs <= 0	`bge zero, rs, label`
`bgtz rs, label`	rs > 0	`blt zero, rs, label`
`bgt rs1, rs2, label`	rs1 > rs2	`blt rs2, rs1, label`
`ble rs1, rs2, label`	rs1 <= rs2	`bge rs2, rs1, label`
`bgtu rs1, rs2, label`	rs1 > rs2, bez znaka	`bltu rs2, rs1, label`
`bleu rs1, rs2, label`	rs1 <= rs2, bez znaka	`bgeu rs2, rs1, label`
`j label`	bezuslovni skok	`jal zero, label`
`jr rs`	skok na rs	`jalr zero, 0(rs)`
`call label`	poziv	`jal ra, label`
`tail label`	poziv bez povratka	`jal zero, label`
`ret`	povratak	`jalr zero, 0(ra)`
`csrr rd, csr`	pročitaj CSR	`csrrs rd, csr, zero`
`csrw csr, rs`	upiši CSR	`csrrw zero, csr, rs`
`csrs csr, rs`	postavi bitove	`csrrs zero, csr, rs`
`csrc csr, rs`	obriši bitove	`csrrc zero, csr, rs`
`rdcycle rd`	broj taktova	`csrrs rd, cycle, zero`

Pseudo-instrukcije nisu prave instrukcije procesora: asembler ih zamenjuje jednom ili dve prave (vidi panel Program, kolona sa mašinskim kodom).

Formati instrukcija (32 bita)



Konstanta (imm) se uvek proširuje znakom; bit 31 instrukcije je uvek njen znak. U B i J formatu bit 0 adrese je uvek 0, pa se ne čuva.

Registri i ABI

REGISTAR	ABI	ULOGA	ČUVA
`x0`	`zero`	uvek 0 (upis se ignoriše)	
`x1`	`ra`	povratna adresa	pozivalac
`x2`	`sp`	pokazivač steka	pozvani
`x3`	`gp`	globalni pokazivač	
`x4`	`tp`	pokazivač niti	
`x5–x7`	`t0–t2`	privremeni	pozivalac
`x8`	`s0 / fp`	sačuvan / pokazivač okvira	pozvani
`x9`	`s1`	sačuvan	pozvani
`x10–x11`	`a0–a1`	argumenti / povratna vrednost	pozivalac
`x12–x17`	`a2–a7`	argumenti	pozivalac
`x18–x27`	`s2–s11`	sačuvani	pozvani
`x28–x31`	`t3–t6`	privremeni	pozivalac

„Čuva pozvani“: potprogram mora da vrati staru vrednost pre ret (sačuva je na steku). „Čuva pozivalac“: posle call vrednost više ne važi.

Direktive asemblera

DIREKTIVA	ZNAČENJE
`.text / .data / .rodata / .bss`	sekcija koda ili podataka
`.word / .half / .byte v, …`	reč (4 B), polureč (2 B), bajt
`.ascii / .asciz "tekst"`	tekst (asciz dodaje 0 na kraju)
`.space / .zero n`	n bajtova nula
`.align n / .balign n`	poravnanje na 2^n / n bajtova
`.equ ime, v`	konstanta
`%hi(sym) / %lo(sym)`	gornjih 20 / donjih 12 bita adrese