RISC_V uvod

- Razvoj koncepta računara sa smanjenim skupom instrukcija (RISC) se generalno pripisuje projektima na Univerzitetu Kalifornije, Berkliju i Univerzitetu Stanford ranih 1980.
- Najnovija generacija arhitekture skupa instrukcija RISC iz UC Berkli se zove RISC-V (izgovara se kao "risk-pet") i izdvojen je u neprofitnu organizaciju fondacije (***www.riscv.org***) u pokušaju da se stvori osnova za otvorenu arhitekturu skupa instrukcija i hardver otvorenog koda.
- Arhitektura RISC-V skupa instrukcija (ISA) je još uvek dovoljno nova !

RISC-V Instruction Set Architecture

- RISC-V arhitektura skupa instrukcija (ISA) je u stravi porodica od četiri odvojena, ali povezana, osnovna seta instrukcija
- .Ova četiri ISA-a imaju ili različitu širinu registrara ili različit broj registara, ali su povezani jer svi koriste isto enkodiranje instrukcija za veći deo osnovnog skupa instrukcija.

RISC-V Instruction Set Architecture

- Četiri osnovna skupa instrukcija su:
- RV32I (32-bitni osnovni celobrojni ISA),
- RV32E (32-bitni Base Integer Embedded ISA),
- RV64I (64-bit Base Integer ISA) i
- RV128I (128-bitni osnovni ceo broj ISA).
- Od njih, samo RV32I i RV64I su trenutno zamrznuti.
- RV32I sadrži 32 32-bitna registra i 32-bitni adresni prostor.

RISC-V Instruction Set Architecture

- Četiri osnovna skupa instrukcija su:
- RV32I (32-bitni osnovni celobrojni ISA),
- RV32E (32-bitni Base Integer Embedded ISA),
- RV64I (64-bit Base Integer ISA) i
- RV128I (128-bitni osnovni ceo broj ISA).
- Od njih, samo RV32I i RV64I su trenutno zamrznuti.
- RV32I sadrži 32 32-bitna registra i 32-bitni adresni prostor.

RISC-V Instruction Set Architecture

- RV64I sadrži 64-bitne registre i pravlja 64-bitnim adresama.
- Sva instrukcije koja postoje u RV32I takođe su prisutne u RV64I, osim što ova instrukcije sada rade na 64-bitnim podacima, a ne na 32-bitnim.
- To znači da isti kod može da se pokrene na obe arhitekture, ali će rezultati biti različiti.
- RV64I sadrži posebne instrukcije za izvođenje 32-bitnih operacija koje će vratiti isti rezultat kao RV32I u donju reč registra, sa proširenjem predznaka u gornjoj reči registra.

RISC-V Instruction Set Architecture

- RV128I sadrži 128-bitne registre i upravljati 128-bitnim adresama.
- Ova verzija RISC-V ISA ima iste probleme u odnosu na RV64I kao i ISA64 za RV32I.

RISC-V Instruction Set Architecture

- RISC-V memorija je litle-endian.
- Stvarna širina memorijske magistrale smatrase detaljem implementacije i sirina se moze prilagodjavati zahtevima aplikacija.
- Numerisanje bajtova za litle-endianom prikazano je na sledecem slajdu
- Prikazana je 32-bitnu širinu podataka.

RISC-V Instruction Set Architecture

Bits	31:24	23:16	15:8	7:0
Bytes	byte 3	byte 2	byte 1	byte 0
Halfwords	halfword 1		halfword 0	
Word	word			

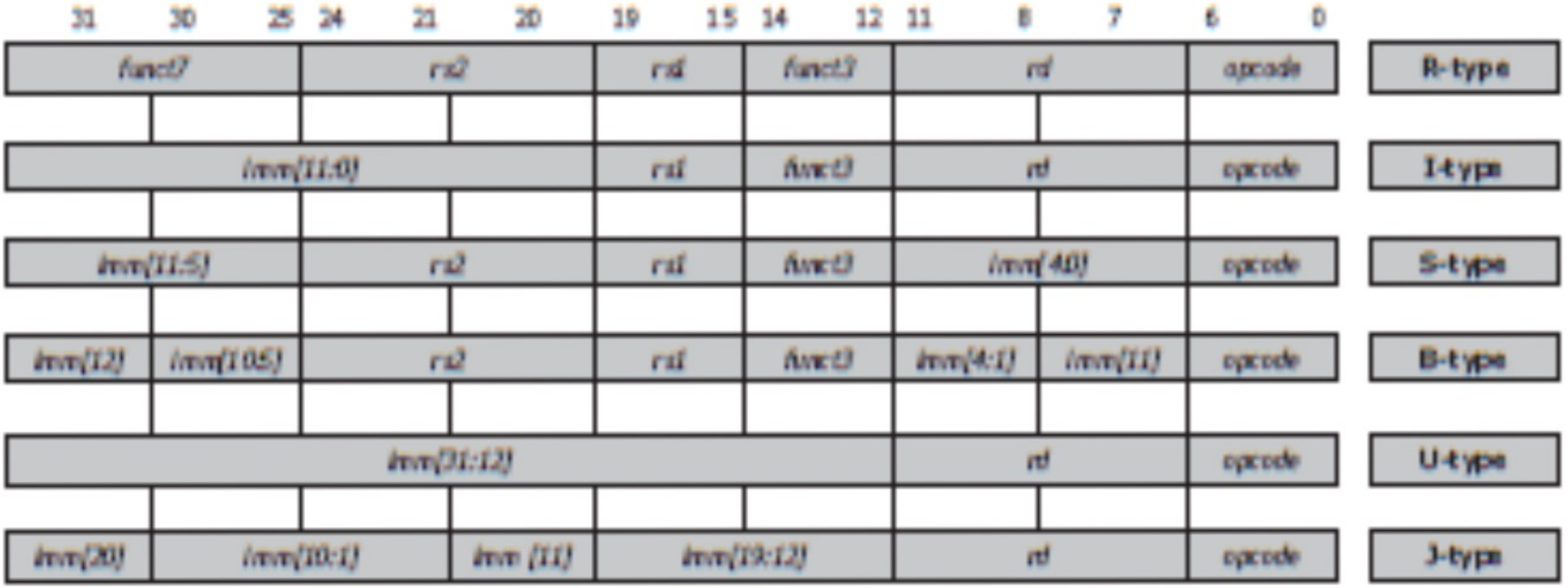
RISC-V Instruction Set Architecture

- Svi osnovni RISC-V ISA koriste fiksnu 32-bitnu veličinu instrukcija i sve instrukcije moraju biti poravnate po rečima u memoriji.
- To znači da su dva najmanje značajna bita adrese instrukcije moraju biti nula ili ce se generisati izuzetak, Address Misaligned exception!

RISC-V format instrukcija

- Skoro sve trenutno definisane RISC-V instrukcije koriste jednu od šest osnovnih 32-bitnih formata instrukcija, što značajno pojednostavljuje dekodiranje instrukcija.
- Na sledecem slajdu prikazan je ovaj format instrukcija.

RISC-V format instrukcija



RISC-V format instrukcija

- Format instrukcija tipa R se koristi za operacije Registar-Registar.
- Sa ovim formatom instrukcija čita dva izvorna operanda iz registara i upisuje rezultat operacije u drugi registar.
- Izvorni operandi su nepromenjeni osim ako se jedan od izvornih registara koristi kao odredište.

RISC-V format instrukcija

- Format instrukcija tipa I se koristi za operacije Register-Immediate.
- Ovim formatom instrukcije čitaju jedan izvorni operand iz registra, uzimaju drugi operand iz neposrednog polja u kodu operacije i upisuju rezultat operacije u registar.
- Izvorni operand je nepromenjen osim ako se izvorni registar takođe koristi kao odredište.
- Formatom instrukcija tipa I, neposredni operand je dugačak 12 bita i proširen je predznakom na puna 32 bita za koriscenje u operaciji.
- Ovo daje opseg od –2048 do +2047 za trenutne podatke.

RISC-V format instrukcija

- Format instrukcija tipa S se koristi isključivo za Store operacije skaldistenja i neposredne podaci su uvek pomeranje adrese.
- Kao i kod drugih neposrednih podataka, 12-bitni pomeraj je uvek sa proširenim znakom, dajući opseg odstupanja od –2048 do +2047 od osnovne registarske adrese.

RISC-V format instrukcija

- Format instrukcija tipa B se koristi isključivo za instrukcije grananja,
- 12-bitni neposredni podaci su opet pomak adrese.
- U ovom formatu pomak u instrukciji je proširen znak, a zatim pomeren ulevo za jedan bit da bi se garantovao da je paran, dajući opseg od –212 do +212 – 2 za grananje.

RISC-V format instrukcija

- Format instrukcija tipa U se koristi za instrukcije koje zahtevaju šire trenutne podatke.
- U ovom formatu neposredni operandi su široki dvadeset bita i ispunjavaju najvažnije bitove 32-bitne reči, sa donjih 12 bitova reči postavljenih na nulu.

RISC-V format instrukcija

- Format instrukcija tipa J se koristi isključivo za jednu vrstu instrukcija skoka, i to 12-bitni neposredni podaci su opet pomeraj adrese.
-

RISC-V registar set

- Polja za izbor registra rd, rs1 i rs2 su kodirana kao što je prikazano u tabeli na sledecem slajdu.
- Registri se oznaceni od x0 do x31, pri čemu je registar x0 ugrađen da sadrži samo read only zero!.
- Postavljanje x0 na nulu omogućava brojne pseudo-instrukcije koje su pogodne za programiranje na asembleru.

rd, rs1, rs2 encoding	Register Name	Register ABI Name	Register ABI Description
00000	x0	zero	Hard-wired zero
00001	x1	ra	Return address
00010	x2	sp	Stack pointer
00011	x3	gp	Global pointer
00100	x4	tp	Thread pointer
00101	x5	t0	Temporary register 0
00110	x6	t1	Temporary register 1
00111	x7	t2	Temporary register 2
01000	x8	s0/fp	Saved reg 0/Frame pointer
01001	x9	s1	Saved register 1
01010	x10	a0	Function return value 0
01011	x11	a1	Function return value 1
01100	x12	a2	Function argument 2
01101	x13	a3	Function argument 3
01110	x14	a4	Function argument 4
01111	x15	a5	Function argument 5
10000	x16	a6	Function argument 6
10001	x17	a7	Function argument 7
10010	x18	s2	Saved register 2
10011	x19	s3	Saved register 3
10100	x20	s4	Saved register 4
10101	x21	s5	Saved register 5
10110	x22	s6	Saved register 6
10111	x23	s7	Saved register 7
11000	x24	s8	Saved register 8
11001	x25	s9	Saved register 9
11010	x26	s10	Saved register 10
11011	x27	s11	Saved register 11
11100	x28	t3	Temporary register 3
11101	x29	t4	Temporary register 4
11110	x30	t5	Temporary register 5
11111	x31	t6	Temporary register 6

RISC-V base integer instruction set

- RV32I Base Integer set instrukcija sadrži samo četrdeset instrukcija.
- Ovih četrdeset instrukcija su apsolutni minimum za bilo koju implementaciju RISC-V i dovoljni su za emulaciju skoro svih trenutnih standardnih ekstenzija RISC-V.
- Lista ovih četrdeset instrukcija prikazana je na sledecem slajdu zajedno sa kodom operacije
- Ova tabela je organizovana po tipu koda operacije.

Assembly Language	Opcode	Type
Defined Illegal	1111111_11111_11111_111_11111_1111111	R
ADD *rd, rs1, rs2*	0000000_ttttt_sssss_000_ddddd_0110011	R
SUB *rd, rs1, rs2*	0100000_ttttt_sssss_000_ddddd_0110011	R
SLL *rd, rs1, rs2*	0000000_ttttt_sssss_001_ddddd_0110011	R
SLT *rd, rs1, rs2*	0000000_ttttt_sssss_010_ddddd_0110011	R
SLTU *rd, rs1, rs2*	0000000_ttttt_sssss_011_ddddd_0110011	R
XOR *rd, rs1, rs2*	0000000_ttttt_sssss_100_ddddd_0110011	R
SRL *rd, rs1, rs2*	0000000_ttttt_sssss_101_ddddd_0110011	R
SRA *rd, rs1, rs2*	0100000_ttttt_sssss_101_ddddd_0110011	R
OR *rd, rs1, rs2*	0000000_ttttt_sssss_110_ddddd_0110011	R
AND *rd, rs1, rs2*	0000000_ttttt_sssss_111_ddddd_0110011	R
LB *rd, offset(rs1)*	mmmmmnn_mmmnn_sssss_000_ddddd_0000011	I
LH *rd, offset(rs1)*	mmmmmnn_mmmnn_sssss_001_ddddd_0000011	I
LW *rd, offset(rs1)*	mmmmmnn_mmmnn_sssss_010_ddddd_0000011	I
LBU *rd, offset(rs1)*	mmmmmnn_mmmnn_sssss_100_ddddd_0000011	I
LHU *rd, offset(rs1)*	mmmmmnn_mmmnn_sssss_101_ddddd_0000011	I
ADDI *rd, rs1, imm*	mmmmmnn_mmmnn_sssss_000_ddddd_0010011	I
SLLI *rd, rs1, shamt*	0000000_shamt_sssss_001_ddddd_0010011	I
SLTI *rd, rs1, imm*	mmmmmnn_mmmnn_sssss_010_ddddd_0010011	I
SLTIU *rd, rs1, imm*	mmmmmnn_mmmnn_sssss_011_ddddd_0010011	I
XORI *rd, rs1, imm*	mmmmmnn_mmmnn_sssss_100_ddddd_0010011	I
SRLI *rd, rs1, shamt*	0000000_shamt_sssss_101_ddddd_0010011	I
SRAI *rd, rs1, shamt*	0100000_shamt_sssss_101_ddddd_0010011	I
ORI *rd, rs1, imm*	mmmmmnn_mmmnn_sssss_110_ddddd_0010011	I
ANDI *rd, rs1, imm*	mmmmmnn_mmmnn_sssss_111_ddddd_0010011	I
JALR *rd, offset(rs1)*	mmmmmnn_mmmnn_sssss_000_ddddd_1100111	I
FENCE	fmodior_wiorw_00000_000_00000_0001111	I
ECALL	0000000_00000_00000_000_00000_1110011	I
EBREAK	0000000_00001_00000_000_00000_1110011	I
SB *rs2, offset(rs1)*	mmmmmnn_ttttt_sssss_000_mnnnn_0100011	S
SH *rs2, offset(rs1)*	mmmmmnn_ttttt_sssss_001_mnnnn_0100011	S
SW *rs2, offset(rs1)*	mmmmmnn_ttttt_sssss_010_mnnnn_0100011	S
BEQ *rs1, rs2, offset*	nmmmmnn_ttttt_sssss_000_mnnnn_1100011	B
BNE *rs1, rs2, offset*	nmmmmnn_ttttt_sssss_001_mnnnn_1100011	B
BLT *rs1, rs2, offset*	nmmmmnn_ttttt_sssss_100_mnnnn_1100011	B
BGE *rs1, rs2, offset*	nmmmmnn_ttttt_sssss_101_mnnnn_1100011	B
BLTU *rs1, rs2, offset*	nmmmmnn_ttttt_sssss_110_mnnnn_1100011	B
BGEU *rs1, rs2, offset*	nmmmmnn_ttttt_sssss_111_mnnnn_1100011	B
AUIPC *rd, imm*	mmmmmnn_mmmnn_mmmnn_mmm_ddddd_0010111	U
LUI *rd, imm*	mmmmmnn_mmmnn_mmmnn_mmm_ddddd_0110111	U
JAL *rd, offset*	mmmmmnn_mmmnn_mmmnn_mmm_ddddd_1101111	J