Κάρτα δεδομένων αναφοράς RISC-V 1. Τραβήξτε κατά μήκος τη διάτρηση για να αφαιρέσετε την κάρτα 2. Διπλώστε την κάρτα κατακόρυφα στη μέση

①

RISC-V Δεδομένα αναφοράς

Βασικές ακέραιες εντολές του RV32I, σε αλφαβητική σειρά

Μνημονικό	Μορφή	Όνομα	Περιγραφή (σε Verilog)	Σημείωση
add	R	ADD	R[rd] = R[rs1] + R[rs2]	
addi	I	ADD Immediate	R[rd] = R[rs1] + imm	
and	R	AND	R[rd] = R[rs1] & R[rs2]	
andi	I	AND Immediate	R[rd] = R[rs1] & imm	
auipc	U	Add Upper Immediate to PC	R[rd] = PC + {imm, 12'b0}	
beq	SB	Branch EQual	if(R[rs1]==R[rs2]) PC=PC+{imm,1b'0}	
bge	SB	Branch Greater than or Equal	if(R[rs1]>=R[rs2]) PC=PC+{imm,1b'0}	
bgeu	SB	Branch ≥ Unsigned	if(R[rs1]>=R[rs2]) PC=PC+{imm,1b'0}	2)
blt	SB	Branch Less Than	if(R[rs1]<R[rs2) PC=PC+{imm,1b'0}	
bltu	SB	Branch Less Than Unsigned	if(R[rs1]<R[rs2) PC=PC+{imm,1b'0}	2)
bne	SB	Branch Not Equal	if(R[rs1]!=R[rs2) PC=PC+{imm,1b'0}	
csrrc	I	Cont./Stat.RegRead&Clear	R[rd] = CSR;CSR = CSR & ~R[rs1]	
csrrci	I	Cont./Stat.RegRead&Clear Imm	R[rd] = CSR;CSR = CSR & ~imm	
csrrs	I	Cont./Stat.RegRead&Set	R[rd] = CSR; CSR = CSR | R[rs1]	
csrrsi	I	Cont./Stat.RegRead&Set Imm	R[rd] = CSR; CSR = CSR | imm	
csrrw	I	Cont./Stat.RegRead&Write	R[rd] = CSR; CSR = R[rs1]	
csrrwi	I	Cont./Stat.Reg Read&Write Imm	R[rd] = CSR; CSR = imm	
ebreak	I	Environment BREAK	Transfer control to debugger	
ecall	I	Environment CALL	Transfer control to operating system	
fence	I	Synch thread	Synchronizes threads	
fence.i	I	Synch Instr & Data	Synchronizes writes to instruction stream	
jal	UJ	Jump & Link	R[rd] = PC+4; PC = PC + {imm,1b'0}	
jalr	I	Jump & Link Register	R[rd] = PC+4; PC = R[rs1]+imm	
lb	I	Load Byte	R[rd] = {24'bM[](7),M[R[rs1]+imm](7:0)}	3) 4)
lbu	I	Load Byte Unsigned	R[rd] = {24'b0,M[R[rs1]+imm](7:0)}	
lh	I	Load Halfword	R[rd] = {16'bM[](15),M[R[rs1]+imm](15:0)}	
lhu	I	Load Halfword Unsigned	R[rd] = {16'b0,M[R[rs1]+imm](15:0)}	4)
lui	U	Load Upper Immediate	R[rd] = {imm, 12'b0}	
lw	I	Load Word	R[rd] = {M[R[rs1]+imm](31:0)}	
or	R	OR	R[rd] = R[rs1] | R[rs2]	
ori	I	OR Immediate	R[rd] = R[rs1] | imm	4)
sb	S	Store Byte	M[R[rs1]+imm](7:0) = R[rs2](7:0)	
sh	S	Store Halfword	M[R[rs1]+imm](15:0) = R[rs2](15:0)	
sll	R	Shift Left	R[rd] = R[rs1] << R[rs2]	
slli	I	Shift Left Immediate	R[rd] = R[rs1] << imm	
slt	R	Set Less Than	R[rd] = (R[rs1] < R[rs2]) ? 1 : 0	
slti	I	Set Less Than Immediate	R[rd] = (R[rs1] < imm) ? 1 : 0	
sltiu	I	Set < Immediate Unsigned	R[rd] = (R[rs1] < imm) ? 1 : 0	
sltu	R	Set Less Than Unsigned	R[rd] = (R[rs1] < R[rs2]) ? 1 : 0	
sra	R	Shift Right Arithmetic	R[rd] = R[rs1] >> R[rs2]	
srai	I	Shift Right Arith Imm	R[rd] = R[rs1] >> imm	2)
srl	R	Shift Right (Word)	R[rd] = R[rs1] >> R[rs2]	2)
srli	I	Shift Right Immediate	R[rd] = R[rs1] >> imm	5)
sub,subw	R	SUBtract (Word)	R[rd] = R[rs1] – R[rs2]	5)
sw	S	Store Word	M[R[rs1]+imm](31:0) = R[rs2](31:0)	
xor	R	XOR	R[rd] = R[rs1] ^ R[rs2]	
xori	I	XOR Immediate	R[rd] = R[rs1] ^ imm	

Σημειώσεις: 1) Η λειτουργία εκτελείται εξ ορισμού με απρόσημους ακεραίους (αντί για συμπλήρωμα ως προς 2)

2) Το λιγότερο σημαντικό bit της διεύθυνσης διακλάδωσης της jalr παίρνει τιμή 0

3) Οι (προσημασμένες) εντολές load επεκτείνουν το bit προσήμου των δεδομένων ώστε να γεμίσει ο καταχωρητής 32 bit

4) Με επανάληψη του bit προσήμου για να συμπληρωθούν τα αριστερότερα bit του αποτελέσματος κατά τη δεξιά ολίσθηση

5) Πολλαπλασιασμός με έναν τελεστέο προσημασμένο και έναν απρόσημο

6) Η έκδοση single εκτελεί λειτουργία απλής ακρίβειας χρησιμοποιώντας τα 32 δεξιότερα bit ενός καταχωρητή F των 64 bit

7) Κατηγοριοποίηση των εγγραφών με μια μάσκα 10 bit που δείχνει ποιες ιδιότητες ισχύουν (π.χ. –άπειρο, –0, +0, +άπειρο, μη κανονικοποιημένος, …)

8) Ατομική λειτουργία μνήμης: Τίποτε άλλο δεν μπορεί να παρεμβληθεί μεταξύ της ανάγνωσης και της εγγραφής της θέσης μνήμης

Το άμεσο πεδίο υφίσταται επέκταση προσήμου στη RISC-V

②

ΣΥΝΟΛΟ ΕΝΤΟΛΩΝ ΑΡΙΘΜΗΤΙΚΟΥ ΠΥΡΗΝΑ

Επέκταση Πολλαπλασιασμού RV64M

Μνημονικό	Μορφή	Όνομα	Περιγραφή (σε Verilog)	Σημείωση
mul	R	MULtiply	R[rd] = (R[rs1] * R[rs2])(63:0)	
mulh	R	MULtiply High	R[rd] = (R[rs1] * R[rs2])(127:64)	
mulhsu	R	MULtiply High Unsigned	R[rd] = (R[rs1] * R[rs2])(127:64)	2)
mulhu	R	MULtiply upper Half Unsigned	R[rd] = (R[rs1] * R[rs2])(127:64)	6)
div	R	DIVide	R[rd] = (R[rs1] / R[rs2])	
divu	R	DIVide Unsigned	R[rd] = (R[rs1] / R[rs2])	2)
rem	R	REMainder	R[rd] = (R[rs1] % R[rs2])	
remu	R	REMainder Unsigned	R[rd] = (R[rs1] % R[rs2])	2)

Επεκτάσεις κινητής υποδιαστολής RV64F και RV64D

fld,flw	I	Load (Word)	F[rd] = M[R[rs1]+imm]	
fsd,fsw	S	Store (Word)	M[R[rs1]+imm] = F[rd]	
fadd.s,fadd.d	R	ADD	F[rd] = F[rs1] + F[rs2]	7)
fsub.s,fsub.d	R	SUBtract	F[rd] = F[rs1] – F[rs2]	7)
fmul.s,fmul.d	R	MULtiply	F[rd] = F[rs1] * F[rs2]	7)
fdiv.s,fdiv.d	R	DIVide	F[rd] = F[rs1] / F[rs2]	7)
fsqrt.s,fsqrt.d	R	SQuare RooT	F[rd] = sqrt(F[rs1])	7)
fmadd.s,fmadd.d	R	Multiply-ADD	F[rd] = F[rs1] * F[rs2] + F[rs3]	7)
fmsub.s,fmsub.d	R	Multiply-SUBtract	F[rd] = F[rs1] * F[rs2] - F[rs3]	7)
fmnsub.s,fmnsub.d	R	Negative Multiply-ADD	F[rd] = –(F[rs1] * F[rs2] – F[rs3])	7)
fmnadd.s,fmnadd.d	R	Negative Multiply-SUBtract	F[rd] = –(F[rs1] * F[rs2] + F[rs3])	7)
fsgnj.s,fsgnj.d	R	SiGN source	F[rd] = { F[rs2]<63>,F[rs1]<62:0>}	7)
fsgnjn.s,fsgnjn.d	R	Negative SiGN source	F[rd] = { (~F[rs2]<63>), F[rs1]<62:0>}	7)
fsgnjx.s,fsgnjx.d	R	Xor SiGN source	F[rd] = {F[rs2]<63>^F[rs1]<63>, F[rs1]<62:0>}	7)
fmin.s,fmin.d	R	MINimum	F[rd] = (F[rs1] < F[rs2]) ? F[rs1] : F[rs2]	7)
fmax.s,fmax.d	R	MAXimum	F[rd] = (F[rs1] > F[rs2]) ? F[rs1] : F[rs2]	7)
feq.s,feq.d	R	Compare Float EQual	R[rd] = (F[rs1]== F[rs2]) ? 1 : 0	7)
flt.s,flt.d	R	Compare Float Less Than	R[rd] = (F[rs1]< F[rs2]) ? 1 : 0	7)
fle.s,fle.d	R	Compare Float Less than or =	R[rd] = (F[rs1]<= F[rs2]) ? 1 : 0	7)
fclass.s,fclass.d	R	Classify Type	R[rd] = class(F[rs1])	7,8)
fmv.s.x,fmv.d.x	R	Move from Integer	F[rd] = R[rs1]	7)
fmv.x.s,fmv.x.d	R	Move to Integer	R[rd] = F[rs1]	7)
fcvt.d.s	R	Convert from SP to DP	F[rd] = single(F[rs1])	
fcvt.s.d	R	Convert from DP to SP	F[rd] = double(F[rs1])	
fcvt.s.w,fcvt.d.w	R	Convert from 32b Integer	F[rd] = float(R[rs1](31:0))	7)
fcvt.s.l,fcvt.d.l	R	Convert from 64b Integer	F[rd] = float(R[rs1](63:0))	7)
fcvt.s.wu,fcvt.d.wu	R	Convert from 32b Int Unsigned	F[rd] = float(R[rs1](31:0))	2,7)
fcvt.s.lu,fcvt.d.lu	R	Convert from 64b Int Unsigned	F[rd] = float(R[rs1](63:0))	2,7)
fcvt.w.s,fcvt.w.d	R	Convert to 32b Integer	R[rd](31:0) = integer(F[rs1])	7)
fcvt.l.s,fcvt.l.d	R	Convert to 64b Integer	R[rd](63:0) = integer(F[rs1])	7)
fcvt.wu.s,fcvt.wu.d	R	Convert to 32b Int Unsigned	R[rd](31:0) = integer(F[rs1])	2,7)
fcvt.lu.s,fcvt.lu.d	R	Convert to 64b Int Unsigned	R[rd](63:0) = integer(F[rs1])	2,7)

Επέκταση ατομικών (αδιαίρετων) εντολών RV64A

amoadd.w,amoadd.d	R	ADD	R[rd] = M[R[rs1]], M[R[rs1]] = M[R[rs1]] + R[rs2]	9)
amoand.w,amoand.d	R	AND	R[rd] = M[R[rs1]], M[R[rs1]] = M[R[rs1]] & R[rs2]	9)
amomax.w,amomax.d	R	MAXimum	R[rd] = M[R[rs1]], if (R[rs2] > M[R[rs1]]) M[R[rs1]] = R[rs2]	9)
amomaxu.w,amomaxu.d	R	MAXimum Unsigned	R[rd] = M[R[rs1]], if (R[rs2] > M[R[rs1]]) M[R[rs1]] = R[rs2]	2,9)
amomin.w,amomin.d	R	MINimum	R[rd] = M[R[rs1]], if (R[rs2] < M[R[rs1]]) M[R[rs1]] = R[rs2]	9)
amominu.w,amominu.d	R	MINimum Unsigned	R[rd] = M[R[rs1]], if (R[rs2] < M[R[rs1]]) M[R[rs1]] = R[rs2]	2,9)
amoor.w,amoor.d	R	OR	R[rd] = M[R[rs1]], M[R[rs1]] = M[R[rs1]] | R[rs2]	9)
amoswap.w,amoswap.d	R	SWAP	R[rd] = M[R[rs1]], M[R[rs1]] = R[rs2]	9)
amoxor.w,amoxor.d	R	XOR	R[rd] = M[R[rs1]], M[R[rs1]] = M[R[rs1]] ^ R[rs2]	9)
lr.w,lr.d	R	Load Reserved	R[rd] = M[R[rs1]], reservation on M[R[rs1]]	
sc.w,sc.d	R	Store Conditional	if reserved, M[R[rs1]] = R[rs2], R[rd] = 0; else R[rd] = 1	

ΜΟΡΦΕΣ ΕΝΤΟΛΩΝ ΠΥΡΗΝΑ

	31 27	26 25	24 20	19 15	14 12	11 7	6 0
R	funct7		rs2	rs1	funct3	rd	Opcode
I	imm[11:0]			rs1	funct3	rd	Opcode
S	imm[11:5]		rs2	rs1	funct3	imm[4:0]	opcode
SB	imm[12|10:5]		rs2	rs1	funct3	imm[4:1|11]	opcode
U	imm[31:12]					rd	opcode
UJ	imm[20|10:1|11|19:12]					rd	opcode

ΨΕΥΔΟΕΝΤΟΛΕΣ ③

Μνημονικό	Όνομα	Περιγραφή	Χρήσεις
beqz	Branch = zero	if(R[rs1]==0) PC=PC+{imm,1b'0}	beq
bnez	Branch ≠ zero	if(R[rs1]!=0) PC=PC+{imm,1b'0}	bne
fabs.s,fabs.d	Absolute Value	F[rd] = (F[rs1]< 0) ? –F[rs1] : F[rs1]	fsgnx
fmv.s,fmv.d	FP Move	F[rd] = F[rs1]	fsgnj
fneg.s,fneg.d	FP negate	F[rd] = –F[rs1]	fsgnjn
j	Jump	PC = {imm,1b'0}	jal
jr	Jump register	PC = R[rs1]	jalr
la	Load address	R[rd] = address	auipc
li	Load imm	R[rd] = imm	addi
mv	Move	R[rd] = R[rs1]	addi
neg	Negate	R[rd] = –R[rs1]	sub
nop	No operation	R[0] = R[0]	addi
not	Not	R[rd] = ~R[rs1]	xori
ret	Return	PC = R[1]	jalr
seqz	Set = zero	R[rd] = (R[rs1]== 0) ? 1 : 0	sltiu
snez	Set ≠ zero	R[rd] = (R[rs1]!= 0) ? 1 : 0	sltu

ΚΩΔΙΚΟΙ ΛΕΙΤΟΥΡΓΙΑΣ (OPCODES) ΣΕ ΑΡΙΘΜΗΤΙΚΗ ΣΕΙΡΑ

Μνημονικό	Μορφή	OPCODE	Λειτουργία3	Λειτουργία7 ή IMM	Δεκαεξαδικά
lb	I	0000011	000		03/0
lh	I	0000011	001		03/1
lw	I	0000011	010		03/2
lbu	I	0000011	100		03/4
lhu	I	0000011	101		03/5
fence	I	0001111	000		0F/0
fence.i	I	0001111	001		0F/1
addi	I	0010011	000		13/0
slli	I	0010011	001	0000000	13/1/00
slti	I	0010011	010		13/2
sltiu	I	0010011	011		13/3
xori	I	0010011	100		13/4
srli	I	0010011	101	0000000	13/5/00
srai	I	0010011	101	0100000	13/5/20
ori	I	0010011	110		13/6
andi	I	0010011	111		13/7
auipc	U	0010111			17
sb	S	0100011	000		23/0
sh	S	0100011	001		23/1
sw	S	0100011	010		23/2
add	R	0110011	000	0000000	33/0/00
sub	R	0110011	000	0100000	33/0/20
sll	R	0110011	001	0000000	33/1/00
slt	R	0110011	010	0000000	33/2/00
sltu	R	0110011	011	0000000	33/3/00
xor	R	0110011	100	0000000	33/4/00
srl	R	0110011	101	0000000	33/5/00
sra	R	0110011	101	0100000	33/5/20
or	R	0110011	110	0000000	33/6/00
and	R	0110011	111	0000000	33/7/00
lui	U	0110111			37
beq	SB	1100011	000		63/0
bne	SB	1100011	001		63/1
blt	SB	1100011	100		63/4
bge	SB	1100011	101		63/5
bltu	SB	1100011	110		63/6
bgeu	SB	1100011	111		63/7
jalr	I	1100111	000		67/0
jal	UJ	1101111			6F
ecall	I	1110011	000	000000000000	73/0/000
ebreak	I	1110011	000	000000000001	73/0/001
CSRRW	I	1110011	001		73/1
CSRRS	I	1110011	010		73/2
CSRRC	I	1110011	011		73/3
CSRRWI	I	1110011	101		73/5
CSRRSI	I	1110011	110		73/6
CSRRCI	I	1110011	111		73/7

ΟΝΟΜΑ, ΧΡΗΣΗ, ΣΥΜΒΑΣΗ ΚΛΗΣΗΣ ΚΑΤΑΧΩΡΗΤΩΝ ④

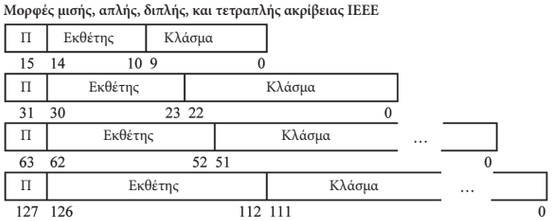
ΚΑΤΑΧΩΡΗΤΗΣ	ΟΝΟΜΑ	ΧΡΗΣΗ	ΑΠΟΘΗΚΕΥΕΤΑΙ ΑΠΟ
x0	zero	Η σταθερή τιμή 0	Δ/Ε
x1	ra	Διεύθυνση επιστροφής	Καλούσα συνάρ./Λειτ.
x2	sp	Δείκτης στοίβας	Καλούμενη συνάρ./Λειτ.
x3	gp	Καθολικός δείκτης	--
x4	tp	Δείκτης νήματος	--
x5–x7	t0–t2	Προσωρινοί καταχωρητές	Καλούσα συνάρ./Λειτ.
x8	s0/fp	Αποθηκευμένος καταχωρητής/ δείκτης πλαισίου	Καλούμενη συνάρ./Λειτ.
x9	s1	Αποθηκευμένος καταχωρητής	Καλούμενη συνάρ./Λειτ.
x10–x11	a0–a1	Ορίσματα συνάρτησης/ επιστρεφόμενες τιμές	Καλούσα συνάρ./Λειτ.
x12–x17	a2–a7	Ορίσματα συνάρτησης	Καλούσα συνάρ./Λειτ.
x18–x27	s2–s11	Αποθηκευμένοι καταχωρητές	Καλούμενη συνάρ./Λειτ.
x28–x31	t3–t6	Προσωρινοί καταχωρητές	Καλούσα συνάρ./Λειτ.
f0–f7	ft0–ft7	Προσωρινοί καταχωρητές κιν. υποδ.	Καλούσα συνάρ./Λειτ.
f8–f9	fs0–fs1	Αποθηκευμένοι καταχωρητές κιν. υποδ.	Καλούμενη συνάρ./Λειτ.
f10–f11	fa0–fa1	Ορίσματα συνάρτησης/ επιστρεφόμενες τιμές κιν. υποδ.	Καλούσα συνάρ./Λειτ.
f12–f17	fa2–fa7	Ορίσματα συνάρτησης κιν. υποδ.	Καλούσα συνάρ./Λειτ.
f18–f27	fs2–fs11	Αποθηκευμένοι καταχωρητές κιν. υποδ.	Καλούμενη συνάρ./Λειτ.
f28–f31	ft8–ft11	R[rd] = R[rs1] + R[rs2]	Καλούσα συνάρ./Λειτ.

ΠΡΟΤΥΠΟ ΚΙΝΗΤΗΣ ΥΠΟΔΙΑΣΤΟΛΗΣ IEEE 754

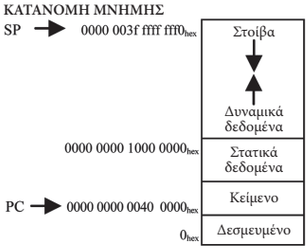
$$(-1)^{\Pi} \times (1 + \text{Κλάσμα}) \times 2^{(\text{Εκθέτης} - \text{Πόλωση})}$$

όπου πόλωση μισής ακρίβειας = 15, πόλωση απλής ακρίβειας = 127,
πόλωση διπλής ακρίβειας = 1023, πόλωση τετραπλής ακρίβειας = 16383

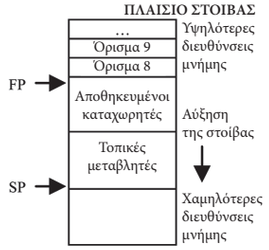
Μορφές μισής, απλής, διπλής, και τετραπλής ακρίβειας IEEE



ΚΑΤΑΝΟΜΗ ΜΝΗΜΗΣ



ΠΛΑΙΣΙΟ ΣΤΟΙΒΑΣ



ΠΡΟΘΕΜΑΤΑ ΜΕΓΕΘΟΥΣ ΚΑΙ ΣΥΜΒΟΛΑ

ΜΕΓΕΘΟΣ	ΠΡΟΘΕΜΑ	ΣΥΜΒΟΛΟ	ΜΕΓΕΘΟΣ	ΠΡΟΘΕΜΑ	ΣΥΜΒΟΛΟ
1000^1	Kilo-	K	2^{10}	Kibi-	Ki
1000^2	Mega-	M	2^{20}	Mebi-	Mi
1000^3	Giga-	G	2^{30}	Gibi-	Gi
1000^4	Tera-	T	2^{40}	Tebi-	Ti
1000^5	Peta-	P	2^{50}	Pebi-	Pi
1000^6	Exa-	E	2^{60}	Exbi-	Ei
1000^7	Zetta-	Z	2^{70}	Zebi-	Zi
1000^8	Yotta-	Y	2^{80}	Yobi-	Yi
1000^9	Ronna-	R	2^{90}	Robi-	Ri
1000^{10}	Quecca-	Q	2^{100}	Quebi-	Qi
1000^{-1}	milli-	m	1000^{-5}	femto-	f
1000^{-2}	micro-	μ	1000^{-6}	atto-	a
1000^{-3}	nano-	n	1000^{-7}	zepto-	z
1000^{-4}	pico-	p	1000^{-8}	yocto-	y
			1000^{-9}	ronto-	r
			1000^{-10}	quecto-	q

Κάρτα δεδομένων αναφοράς RISC-V 1. Τραβήξτε κατά μήκος τη διάτρηση για να αφαιρέσετε την κάρτα 2. Διπλώστε την κάρτα κατακόρυφα στη μέση