

PULP PLATFORM Open Source Hardware, the way it should be!

Low Power Multicore Solutions for Approximation

Luca Benini

lbenini@iis.ee.ethz.ch,luca.benini@unibo.it>













ML Processors from Tiny to Huge

ETH zürich





TinyML Opportunity

[ABI research 21]



TinyML challenge

AI capabilities in the power envelope of an MCU: 10-mW peak (1mW avg)

TinyML Workloads – DNNs (and More)

H Pham 2021(Google) arXiv:2003.10580v3







ETHZürich

5

Energy efficiency @ GOPS is the Challenge

ARM Cortex-M MCUs: M0+, M4, M7 (40LP, typ, 1.1V)*



ETH zürich



*data from ARMs web

The Tunnel: High-Performance vs. Energy-Efficient



- "Classical" core performance scaling trajectory
 - Faster CLK \rightarrow deeper pipeline \rightarrow IPC drops
 - Recover IPC \rightarrow superscalar \rightarrow ILP bottleneck (dependencies)
 - Mitigate ILP bottlenecks \rightarrow 000 \rightarrow huge power, area cost!





ISA extension cost 25 kGE \rightarrow 40 kGE (1.6x), energy efficient if 0.6T_{exec}

RISC-V Instruction Set Architecture



- Started by UC-Berkeley in 2010
- Contract between SW and HW
 - Partitioned into user and privileged spec
 - External Debug
- Standard governed by RISC-V foundation
 - ETHZ is a founding member of the foundation
 - Necessary for the continuity
 - Defines 32, 64 and 128 bit ISA
 - No implementation, just the ISA
 - Different implementations (both open and close source)
- At ETHZ+UNIBO we specialize in efficient implementations of RISC-V cores





HW

RISC-V Foundation Members



RISC-V ISA Baseline and Extensions

- Integer instructions (frozen)
- **E** Reduced number of registers
- Multiplication and Division (frozen)
- A Atomic instructions (frozen)
- F Single-Precision Floating-Point (frozen)

ETH zürich

С

X

- Double-Precision Floating-Point (frozen)
 - Compressed Instructions (frozen)

Non Standard Extensions

- Kept very simple and extendable
 - Wide range of applications from IoT to HPC
- RV + word-width + extensions
 - RV32IMC: 32bit, integer, multiplication, compressed

User specification:

Separated into extensions, only I is mandatory

Privileged Specification (WIP):

- Governs OS functionality: Exceptions, Interrupts
- Virtual Addressing
- Privilege Levels

Baseline + Extensins in a Page

Free & Open 💦 RISC-V Reference Card

| Pas | conneuen. | | | * 0 ** A ** CI IN IN * A & O | _ | A 20 | 10110-110-110-110-1 | | | | | MINASIA ALEAN AND AND AND AND AND AND AND AND AND A | | | | |
|---|---|--|---|--|--|--|---|--|---|--|--|--|---|--|--|--|
| Category | Name | Fint | RV32I Base | +RV{64,128 | Cate | TOTY Name | RV mnemonic | Category | Name | Fmt | RV32M (M | lultiply-Divide) | | +RV(64 | .128) | |
| oads | Load Byte | I LB | rd, rsl, imm | | CSR | Access Atomic R/V | V CSRRW rd.csr.rs1 | Multiply | MULtiply | R | NUL | rd,rsl,rs2 | MUL (W D |) 70 | 1,rsl,rs2 | |
| L. L. | oad Halfword | I LH | rd, rsl, imm | | | Atomic Read & Set B | it CSRRS rd, csr, rs1 | | MULtiply upper Half | R | NULH | rd, rs1, rs2 | | | | |
| | Load Word | I LW | rd, rsl, imm | L{DQ} rd,rsl | .imm | Atomic Read & Clear B | it CERRC rd, csr, rs1 | (M) | ULtiply Half Sign 1 | R | in-peu | D Highlight | | | | |
| Load B | yte Unsigned | I LEU | rd, rs1, imm | | | Attack R/W Int | CSRRWI rd, csr, imm | MU | Ltiply upper Hal | | | | (\mathbf{W}) | | | |
| Load H | Half Unsigned | I LHU | rd, rsl, imm | L{W D}U rd,rs1 | imm Ato | mic Read * A c A V A | I (AYA A, csr, imm | Divide | DIVide | R | DIV | rd,rs1,rs2 | DIVIN D |) a t | 1,751,752 | |
| Stores | Store Byte | S SB | rol, ro2, imm | | Atom | ic Read & Cleur Lit .m. | Losr, imm | | DIVide Unsigned | R | DIVU | rd,rsl,rs2 | | | | |
| St | tore Halfword | S SH | rs1,rs2,imm | | Chan | ge Level Env. Ca | I ECALL | Remainde | er REMainder | R | REM | rd, rsl, rs2 | RENAND |) zi | 1,751,752 | |
| | Store Word | S SW | rsl,rs2,imm | S(D 0) rs1.rs | e.imn E | invironment B a pin | t 🖻 BFAR | | REMainder Unsigned | R | REMIT | ed.es1.es2 | DEMINING | 53. ard | Carso Exercit | |
| Shifts | Shift Loft | R ST.T. | and well we? | ระบงพุโกะ พริ.พร. | 202 | Environme | 10(2 | | | | | | | | | |
| Shift Le | ft Immediate | T ST. | T rd.rs1.shamt | SLLT/WID1 rd.rs1 | shar Trap | Redirect to Supervis | OMETS | Category | Name | Fmt | RV32 | A (Atomic) | | +RV(64 | .128) | 1 |
| 654303 B 828 | Shift Right | R epr. | nd.rel.re2 | SRL/WID1 rd.rol | ra2 Rer | lipst Tian to Hyperkie | ST MR TH | Load | Load Reserved | R | LR-W | rd_rs1 | LR- (DIO | 1 70 | 1.rsl | 1 |
| Shift Righ | ht Immediate | T ISPT. | T rd.rel.shamt | SRLTINID1 rd.rsl | shar Hyper | visor Tran to Superviso | THERS. | Store | Store Conditional | R | SC.W | rd.rsl.rs2 | SC. IDIO |) T(| 1.rsl.rs2 | 1 |
| Shift Rig | ht arithmetic | D RA | rd.rel.re2 | SRA/WID1 rd.rst | rs2 Inter | runt Wait for Internu | N NFT | Swap | SWAP | R | AMOSWAP.W | rd.rsl.rs2 | ANDSWAP | 4D/03 70 | 1.751.752 | 1 |
| Shift Rig | ht Arith Imm | I SRA | I rd.rsl.shamt | SBAL(WD) rd.rsl | shar MMU | Supervisor FENC | E SFENCE. VM rs1 | Add | ADD | B | AMOADD | rd_rsl_rs2 | AMOADD- | ED 03 TO | 1.rsl.rs2 | 1 |
| Arithmeti | ic ADD | ADD | rd.rsl.rs2 | ADD/W[D1 rd.rs1 | T52 | | | Logical | | | NO IN VI | | TOXI R. | VIA B TO | J.rsl.rs2 | 1 |
| AD | D Immediate | ADD | T rd.rsl.imm | ADDL(W D) rd.rsl | . i-mm | | | | AL(0) | ÎÂ | A VAN IN (| | DAL D | A VA TO | f.rsl.rs2 | |
| 11124 | SUBtract | IF SUB | rd.rsl.rs2 | | | | | | OR | R | AMOOR .W | rd.rsl.rs2 | AMOOR / | | f.rsl.rs2 | |
| 1.00 | d Danas Tasas | | and law | Optional Cor | appropriat (| (Chit) Inchruchi | on Extension: RVC | Min /Max | MINIMUM | D | N DECEMBER OF | free free los | KMCM TH | anim | fred red | 1 |
| Add Llog | a upper uniti | 201 | DGy1mm | Catagony Name | East [| DUC DUC | BUI equivalent | rilli/ riak | MAN INCOME | | AND DELENSING | EMpERAgers | K MONTEN | teriati es | 154865486 | |
| Add Upp | er imm to PC | D Non | PC DG, 1mm | Lands Land Marke | CL A SH | NVG | Por equiverent | | MINIMUM Incomed | | AND ADDRESS OF | | ANONITARI | Called as | iyesayese i wali wali | |
| ogical | AMR . | S MOR | IGFISIFISZ | Loads Load Word | CL GADW | Ter, "Ter, "Thus | The rd on domin | | Maximum Unsigned | | AND DELETION OF | supersystem ad and and | ANON DRO | alpiki er | tytestytes | |
| × | AV THUHEODORE | AUR | T TOATATAN | Eoda wora Si | CT CYPHS | os. Tor ^a TRBB | The Tolkelph Then, 4 | | Reasonable Consigned | | Discondition of the | 1.10 p 1 0 2 p 1 0 1 | TANONAKO | ************************************** | 111311132 | |
| | OR | OR | rd,rs1,rs2 | Load Double | CL C.LD | rd',rsl',imm | LD xd', xs1', imm*8 | 10 | тее ориона т | | | | | | | |
| 0 | R Immediate | ORI | rd, rsl, imm | Load Double Si | CI C.LDS | RP md, inn | LD rd, sp, imm*8 | category | Name | FUL | INVOX(FILIQ) | (HE/SE/DE QELLES) | - | #KV(09) | 145 | |
| | AND | AND | rd, rsl, rs2 | Load Quar | CL C.LQ | rd', rsl', imm | LQ rd', rsl', imm*16 | Move | Move from integer | | Nuv. (nj.s) sk | IGyIS1 | FMV- (D) | 5)• x | rd,ra | |
| AN | D Immediate | AND | I rd,rsl,imm | Load Quad SI | CI C.LOS | IP rd,imm | LQ rd, sp, imm*16 | Concernance of the second s | Move to integer | | KVeKe(B S) | JEG _F EBU | FMV _x X _x (| | rd _e rs | |
| Compare | Set < | SLT | rd, rs1, rs2 | Stores Store Word | CS C.SW | rs1',rs2',imm | SW rs1', rs2', imm*4 | Convert | Convert from Int | | CALCENTS OF CONTRACTS | () WE EDJERL | ECSE-(E | 2 P 2}=(| (E(T) Edyca | |
| Set | < Immediate | SLT. | I rd,rsl,imm | Store Word St | CSS C.SWS | IP ISZ,1MM | SW rs2, sp, imm*4 | Conve | it from int Unsigned | | Desire fulgibile | yomu adyest | DOADS-UR | telin Materi | | |
| 50 | rt < Unsigned | K SEE | U EG, FSL, FSZ | Store Double | | ESI', ESZ', 1mm | SD TS1', TS2', 1mm*8 | Contract Contract | Convert to Int | | nerse an anisis | প্ৰাপ্তা কৰাজনাৰ প্ৰাপ্তা কৰাজনাৰ | DESCRIPTION OF | inge gejei | NATES AND A | |
| Set < B | | | The second second | | | CORP. Manager Cold. Manager Cold. | The second s | | | | THE WARD PROPERTY OF THE TAXES | EXTRACTOR DECEMBER | and the second second second | | | |
| | THE PERMIT | DIT. | IO DU/EST/LINE | Store Double St | 000 0.000 | site. Transfe & regiment | an ray'sh'run a | Cor | iverview are origined | H . • | P | alter action | | | | |
| Branches | Branch = | CC BEQ | rs1,rs2,imm | Store Double St | CS C.SQ | rs1',rs2',imm | SQ rs1',rs2',imm*16 | Load | Load | | L.W.D.Q} | rd,rsl,imm | I | | RISC-V Calli | g Convention |
| 3ranches | Branch ≠ | BEQ | rs1,rs2,imm rs1,rs2,imm | Store Quar Store Quar Store Quad Si | CS c.sg | rs1',rs2',imm | SQ rs1',rs2',imm*16 | Load Store | Load Store | | L(W,D,Q) S(W,D,Q) | rd,rsl,inm rsl,rs2,inm | Register | ABI Name | KISC-V Calli Saver | g Convention Description |
| 3ranches | Branch = Branch ≠ Branch < | BEQ | rs1,rs2,imm rs1,rs2,imm rs1,rs2,imm | Store Double Si Store Quad Store Quad Si Arithmetic ADD | es e.se | rs1',rs2',imn | SQ rs1',rs2',imm*16 rs2,sp,imm*16 rd,rd,rs1 | Load Store Arithmeti | Load Store | | L _{L(W,D,Q)} s(W,D,Q) XDD.(s D Q) | rd,rsl,imm rsl,rs2,imm rd,rs1,rs2 | Register | ABI Name Sego | Saver | g Convention Description Hard-wired zero |
| 3ranches | Branch = Branch ≠ Branch < Branch ≥ | BLT BEQ BILT BCE | rs1,rs2,imm rs1,rs2,imm rs1,rs2,imm rs1,rs2,imm rs1,rs2,imm | Store Double Si Store Quad Store Quad Si Arithmetic ADD ADD Word | | | SQ rs1',rs2',inm*16 rs2,sp,inm*16 rd,rd,rs1 ADBW rd,rd,inm | Load Store Arithmeti | Load Store C ADD SUBtract | | L(W,D,Q) S(W,D,Q) ADD. (S D Q) SUB. (S D Q) | rd,rsl,imm rsl,rs2,imm rd,rs1,rs2 rd,rs1,rs2 | Register | ABI Name Sero Ea | Caller | g Convention Description Hard-wired zero Return address |
| Branches | Branch = Branch ≠ Branch < Branch ≥ h < Unsigned | S B BLT | rs1,rs2,imm rs1,rs2,imm rs1,rs2,imm rs1,rs2,imm rs1,rs2,imm U rs1,rs2,imm | Store Double Si Store Quad Si Arithmetic ADD ADD Wore ADD Immediate | CS C.SQ CS C.SQ CS C.ADI CR C.ADI CR C.ADI | n 192, and rs1', rs2', imm press w rd, rs1 rd, imm | SD 152,59,100*5 SQ rs1',rs2',inm*16 rs2,sp,imm*16 D rd,rd,rs1 ADBW rd,rd,inm P rd,rd,inm | Load Store Arithmeti | Load Store ADD SUBtract MULtiply | whal he | L(W,D,Q) S(W,D,Q) ADD, (S D Q) SUB, (S D Q) HUL, (S D Q) | rd,rsl,imm rsl,rsl,imm rd,rsl,rs2 rd,rsl,rs2 rd,rsl,rs2 | Register x0 x1 x2 | ABI Name Sero FR | Caller Caller Callee | g Convention Description Hard-wired zero Return address Stack pointer |
| Branches Brand Brand | s Branch = Branch ≠ Branch < Branch ≥ h < Unsigned h ≥ Unsigned | SB BGE | 10 14,151,100 rs1,rs2,100 rs1,rs2,100 rs1,rs2,100 U rs1,rs2,100 U rs1,rs2,100 | Store Double Si Store Quad Store Quad Si Arithmetic ADD ADD Word ADD Immediate ADD Word I | CS C.SQ CC C i CR C.ADI CI C.ADI | n press mpresse mpresse mrd,rsi fctions | SQ rs2, sby.imm*6 SQ rs2', rs2', imm*16 C) rd, rd, rs1 ADBW rd, rd, imm rd, rd, imm rd, rd, imm rd, rd, imm | Load Store Arithmeti | Load Store C ADD SUBtract MULtiply DIVide | | L(W,D,Q) S(W,D,Q) ADD,(S D Q) SUB.(S D Q) HUL.(S D Q) DIV.(S D Q) | rd,rs1,imm rd,rs1,imm rd,rs1,rs2 rd,rs1,rs2 rd,rs1,rs2 rd,rs1,rs2 rd,rs1,rs2 | Register x0 x1 x2 x2 | ABI Name Sero Ea Sp Sp | Caller Callee | g Convention Description Hard wired zero Return address Stack pointer Global pointer |
| Branches Brand Brand Jump & | s Branch = Branch ≠ Branch < Branch ≥ Branch ≥ h < Unsigned h ≥ Unsigned .ink 384. | SB BLT SB BCE UJ JAL | 10 Fu, ssi, inn rsi, rs2, inn rsi, rs2, inn rsi, rs2, inn rsi, rs2, inn U rsi, rs2, inn U rsi, rs2, inn rd, inn | Store Double ST Store Quad SI Arithmetic ADD ADD Word ADD Immed [*] ht ADD Word I ADD SP Imm [*] 1 | CS C.SQ CC C.ADI CR C.ADI CI C.ADI | npresse ictions | S0 rol', rol', run*6 S0 rol', rol', run*16 F rol, up, imm*16 F rd, rd, rol ADDW rd, rd, imm rd, rd, imm s0, ap, imm*16 | Load Store Arithmeti | Load Store C ADD SUBtract Mültiply DIVide SQuare Root | | L(W,D,Q) S(W,D,Q) ADD.(S D Q) SUB.(S D Q) HUL.(S D Q) HUL.(S D Q) DIV.(S D Q) SQRT.(S D Q) | rd,rs1,inm rs1,rs2,inm rd,rs1,rs2 rd,rs1,rs2 rd,rs1,rs2 rd,rs1,rs2 rd,rs1,rs2 rd,rs1,rs2 rd,rs1,rs2 | Register 20 20 20 20 20 20 20 20 20 20 20 20 20 | ABI Name Sero ER Sp Sp | Caller Caller | g Convention Description Hard-wired zero Return address Stack pointer Global pointer Thread pointer Thread pointer |
| Branches Brand Brand Jump & L Dump & | s Branch = Branch ≠ Branch < Branch < Branch ≥ h < Unsigned h ≥ Unsigned Ink J&L Link Register | SB BLT SB BLT SB BLT SB BLT UJ JAL | 10 10,251,100 rs1,252,imm rs1,252,imm rs1,252,imm U rs1,252,imm U rs1,22,imm rd,100 R rd,751,imm | Store Double Si Store Quat Store Quat Store Quat ADD Word ADD Word I ADD SP Imm * 1 ADD SP Imm * 1 | CR C.ADI | xol', xo2', inm press w rd, ro1 rd, inm ictions rd, inm rd, inm | SD raty, raty, rate raty, raty, rate SQ rati, raty, raty, ratin raty, raty, ratin G raty, raty, ratin raty, raty, ratin ADDW rdy, rat, raty raty, raty, ratin ADDW rdy, raty, ratin raty, | Load Store Arithmeti | Load Store G ADD SUBtract Multiply DIVide SQuare RooT Multiply-ADD | | L(W,D,Q) S(W,D,Q) ADD.(S)D(Q) SUB.(S)D(Q) HUL.(S)D(Q) HUL.(S)D(Q) DIV.(S)D(Q) SQRT.(S)D(Q) FMADD.(S)D(Q) | rd,rs1,imm rs1,rs2,imm rs1,rs2,imm rd,rs1,rs2 rd,rs1,rs2 rd,rs1,rs2 rd,rs1,rs2 rd,rs1,rs2 rd,rs1,rs2 rd,rs1,rs2 rd,rs1,rs2,rs | Register 20 20 20 20 20 20 20 20 20 20 20 20 20 | ABI Name Sero ER Sp Sp tp tp tp tp | Caller Caller Caller | g Convention Description Hard-wired zero Return address Stack pointer Global pointer Thread pointer Thread pointer Temporaries |
| Branches Branch Branch Jump & L Jump & Synch 3 Somethe | s Branch = Branch ≠ Branch ≤ Branch ≤ Branch ≥ h < Unsigned h ≥ Unsigned Link Register Synch thread | SB BEE | <pre>10 Fut, ss1, rma rs1, rs2, imm rs1, rs2, imm rs1, rs2, imm rs1, rs2, imm U rs1, rs2, imm U rs1, rs2, imm rd, imm R rd, rs1, imm CE D r</pre> | Store Double Si Store Quas Store Quas Arithmetic ADD ADD Word ADD Immed ⁻ ht ADD Word I ADD SP Imm ⁺ 1. ADD SP Imm ⁺ 1. Lead Immediat | CR C.ADI CR C.ADI CI C.ADI | A Like , Anni TST , TS2 , Anni APPCSS6 rd, rm APC 10, Anni APC 10, | b) for the phy filmes So rai', ra2', filme'16 So rai', ra2', filme'16 red, rd, rd, rai ADBW rd, rd, film rd, rd, film rd, rd, film b) so rai, rd, film rd, rd, film b) so rai, rd, rai, rai, rai, rai, rai, rai, rai, rai | Load Store Arithmeti Mul-Add | Load Store SUBtract MULtiply DIV/de SQuare RooT Multiply-ADD Multiply-SUBtract | | E{W,D,Q} S{W,D,Q} S{W,D,Q} ADD. (S D Q) ADD. (S D Q) NUL. (S D Q) DIV. (S D Q) DIV. (S D Q) PADD. (S D Q) PADD. (S D Q) PADD. (S D Q) | rd_rrsl_ram rd_rrsl_ram rd_rrsl_ram rd_rrsl_rs2 rd_rrsl_rs2 rd_rrsl_rs2 rd_rrsl_rs2 rd_rrsl_rs2 rd_rrsl_rs2 rd_rrsl_rs2 rd_rrsl_rs2 rd_rrsl_rs2 rd_rrsl_rs2 rd_rrsl_rs2 rd_rrsl_rs2 rd_rrsl_rs2 rd_rrsl_rs2 rd_rrsl_rs2 rd_rrsl_rs2 rd_rrsl_rs2 rd_rrsl_rs2 rd_rs3 rd_rs3 rs2 rd_rs3 rs2 rd_rs3 rs2 rd_rs3 rs2 rd_rs3 rs2 rd_rs3 rs2 rd_rs3 rs2 rd_rs3 rs2 rd_rs3 rs2 rd_rs3 rs2 rs2 rd_rs3 rs2 rd_rs3 rs2 rd_rs3 rs2 rd_rs3 rs2 rd_rs3 rs2 rd_rs3 rs2 rd_rs3 rs2 rd_rs3 rs2 rs3 rs2 rd_rs3 rs3 rs2 rs3 rs3 rs3 rs3 rs3 rs3 rs3 rs3 | Register 30 20 20 20 20 20 20 20 20 20 20 20 20 20 | ABI Name 2010 ER 50 10 to 20-2 s0-2 s0-2 | Caller Caller Caller Caller Caller Caller Caller Caller Caller | g Convention Description Hard-wired zero Return address Stack pointer Global pointer Thread pointer Temporaries Sawed register/frame pointer Sawed register |
| Branches Branch Branch Jump & L Jump & Synch & Synch & | Branch = Branch ≠ Branch ≤ Branch ≤ Branch ≥ h < Unsigned ⊥ink J&L Link Register Synch thread I Instr & Data | SB BEE SB BEE UJ JAL UJ JAL I FEN | <pre>10 Cd, FSI, Ann rS1, rS2, inm rS1, rS2, inm rS1, rS2, inm rS1, rS2, inm U rS1, rS2, inm U rS1, rS2, inm R rd, rS1, inm CE CE CE.I</pre> | Store Double Si Store Quad Si Arithmetic ADD ADD Verra ADD Immed [®] tt ADD SP Imm ^{® 4} ADD SP Imm ^{® 4} Load Immediat Load Upper Imm | CS C.SQ CC C.ADI CR C.ADI CI C.ADI CI C.ADI CI C.ADI CI C.LI CI C.LI | rs1', rs2', inm ppesse rs1', rs2', inm presse rd, rs1 int for a rd, inm rd, inm rd, inm | b) for 2 for 2 for months S0 rat, rat, rat, rime*16 rat, rat, rat, rime*16 rat, rat, rat, rat, rat, rat, rat, rat, | Load Store Arithmeti Mul-Add Negati | Load Store SUBtract MULtiply DIVide SQuare RooT Multiply-XDD Multiply-SUBtract ve Multiply-SUBtract | The state and the | E(W,D,Q) S(W,D,Q) S(W,D,Q) SUB.(S)D(Q) SUB.(S)D(Q) NUL.(S)D(Q) SUW.(S)D(Q) SUW.(S)D(Q) SUW.(S)D(Q) SUW.(S)D(Q) SUW.(S)D(Q) SUW.(S)D(Q) SUW.(S)D(Q) SUW.(S)D(Q) SUW.(S)D(Q) MAUD.(S)D(Q) MAUD.(S)D(Q) MAUD.(S)D(Q) MAUD.(S)D(Q) MAUD.(S)D(Q) MAUD.(S)D(Q) | rd,rsl,rsl,fam rd,rsl,rsl,fam rd,rsl,rsl,rsl rd,rsl,rsl rd,rsl,rsl rd,rsl,rsl rd,rsl,rsl rd,rsl,rsl rd,rsl,rsl rd,rsl,rsl,rsl rd,rsl,rsl,rsl rd,rsl,rsl,rsl rd,rsl,rsl,rsl rd,rsl,rsl,rsl rd,rsl,rsl,rsl rd,rsl,rsl,rsl rd,rsl,rsl,rsl rd,rsl,rsl,rsl rd,rsl,rsl,rsl rd,rsl,rsl rd,rsl,rsl rd,rsl,rsl rd,rsl,rsl rd,rd,rsl rd, | Register 30 20 20 20 20 20 20 20 20 20 20 20 20 20 | ABI Name 2010 FR 50 10 10 10 20 20 20 20 20 20 20 20 20 20 20 20 20 | Caller Caller Caller Caller Caller Caller Callee Callee Callee | g Convention Description Hard-wired zero Return address Stack pointer Global pointer Thread pointer Thread pointer Temporaries Saved register Saved register Saved register |
| Branches Branch Branch Jump & L Jump & Synch & Synch & Synch & | Branch = Branch ≠ Branch ≮ Branch ≤ Branch ≥ h < Unsigned h ≥ Unsigned h ≥ Unsigned Link Register Synch thread Instr & Data System CALL System CALL | BER BER BER BER BER BER BER BER BER BER | <pre>10 10, 10, 10, 10, 10, 10, 10, 10, 10, 1</pre> | Store Datates 3 Store Quar Store Quar Arithmetic ADD ADD Word ADD Word 1 ADD SP Imm ADD SP Imm Load Immediatu Load Upper Imm MWW | CIW C.ADI CI C.ADI | The section of the se | b) ruzyby fume*16 So rul', ru2', fimm*16 ru2, sup, imm*16 ru2, sup, imm*16 ru2, sup, imm*16 ru2, sup, imm*16 ru2, sup, imm*16 ru2, sup, imm*16 ADDI rd, rd, imm ADDI rd, sup, imm*4 ADDI rd, rup, imm*4 ADDI rd, rup, imm*4 Rux rd, imm ADD rd, rup, rup | Load Store Arithmeti Mul-Add Negati Ni | Load Store ADD SUBtract MULtiply DIVide SQuare RooT Multiply-SUBtract ve Multiply-SUBtract ve Multiply-SUBtract cepative Multiply-SUBtract | A ST CARE R. F. H. H. | E.(K,D,Q) S(M,D,Q) S(M,D,Q) S(M,D,Q) SUB.(S) | xd,x32,imm xd,x32,imm xd,x52,imm xd,x52,imm xd,x52,imm xd,x52,imm xd,x52,imm xd,x52,imm xd,x52,imm xd,x52,imm yd,x52,imm yd,x52,imm xd,x52,imm xd,x52,imm yd,x52,imm xd,x52 | Register 20 20 20 20 20 20 20 20 20 20 20 20 20 | ABI Name 2019 ER 50 90 tp t0-2 60/2p 61 a0-1 a0-1 | Caller Caller Caller Callee Callee Callee Callee Callee Callee Caller Caller | g Convention Description Hard wired zero Return address Stack pointer Global pointer Thread pointer Thread pointer Temporaries Saved register/frame pointer Saved register Function arguments/return value Eunction arguments/ |
| Branches Brand Jump & L Jump & Synch 3 Synch 3 | a Branch = Branch ≠ Branch 4 Branch 4 Branch 5 Branch 5 Constant 6 Constant 6 Consta | BER BER BER BER BER BER BER BER BER BER | 10 101, 101, 101, 201 rel, rel, rel2, imm rel, rel2, imm rel, rel2, imm 10 1 rel, rel2, imm 10 rel, rel2, imm 10 rel, rel2, imm 10 rel, rel, imm 10 rel, rel2, imm 10 rel2, rel2, rel2, imm 10 rel2, rel2, | Store Double 3 Store Quad SI Arithmetic ADD ADD Word I ADD SP Imm */ Load Immediat Load Immediat Load Immediat Shife Store Jan | CIUC CADE CADE CADE CADE CIUCA | ne za zana ne zen zen zen zen zen zen zen zen zen z | b) ruzyby rum*16 So rul; ruzyby rum*16 rd; ruzyby rudyrd; rudyrd; ADDW rdyrd, rudyrd; hold rdyrd, rum b) rdyrd, rum | Load Store Arithmeti Mul-Add Negati Ni Sign Tig | ic ADD SUBtract MULtiply DIVide SQuare RooT Multiply-SUBtract Wultiply-SUBtract we Multiply-SUBtract Sign source Factor Sign source | A CONTRACTION OF A CONTRACT | A L(W, D, Q) S(W, D, Q) S(W, D, Q) ADD, (S(D)Q) ADD, (S(D)Q) ADD, (S(D)Q) ADV, (S(D)Q) SQR*, (S(D)Q) MSUB, (S(D)Q) MS | rd_rrs2,imm rd_rrs2,imm rd_rrs1,rs2 rd_rrs1,rs2 rd_rrs1,rs2 rd_rrs1,rs2 rd_rrs1,rs2 rd_rrs1,rs2 rd_rrs1,rs2 rd_rrs1,rs2 rd_rrs1,rs2,rs rd_rrs1,rs2,rs rd_rrs1,rs2,rs rd_rrs1,rs2,rs rd_rrs2,rs2,rs rd_rrs2,rs2,rs rd_rrs2,rs2,rs rd_rrs2,rs2,rs rd_rrs2,rs2,rs rd_rrs2,rs2,rs rd_rrs2,rs2,rs rd_rrs2,rs2,rs rd_rrs2,rs2,rs rd_rrs2,rs2,rs rd_rrs2,rs2,rs rd_rrs2,rs2,rs2,rs rd_rrs2,rs2,rs2,rs rd_rrs2,rs2,rs2,rs rd_rrs2,rs2,rs2,rs rd_rrs2,rs2,rs2,rs rd_rrs2,rs2,rs2,rs rd_rrs2,rs2,rs2,rs rd_rrs2,rs2,rs2,rs rd_rrs2,rs2,rs2,rs rd_rrs2,rs2,rs2,rs2,rs rd_rrs2,rs2,rs2,rs2,rs2,rs2,rs2,rs2,rs2,rs2 | Register 30 30 30 30 30 30 30 30 30 30 30 30 30 | ABI Name 2010 ER 57 67 10-2 10-2 10-2 10-2 10-1 40-1 40-1 40-1 | Caller Caller Caller Caller Caller Caller Caller Caller Caller Caller Caller Caller Caller Caller Caller | g Convention Description Hard wired zero Return address Stack pointer Global pointer Thread pointer Thread pointer Temporaries Saved register/frame pointer Saved register Function arguments/return value Function arguments |
| Branches Branch Branch Jump & L Jump & L Jump & Synch Synch & S Synch & S S Synch & S S Synch & S S S S S S S S S S S S S S S S S S S | s Branch = Branch ≠ Branch ≠ Branch ≥ Branch ≥ Branch ≥ Unsigned ink J&L Link Register Synch thread I Instr & Data I Instr & Data System CALL ystem BREAK ReaD CYCLE | SB BLT SB BLT SB BLT SB BLT SB BGE UJ JAL UJ JAL I FEN I FEN I SCA I SBE I RDC | <pre>No LA, FEI, JAMM ERD, FEZ, intm rel, re2, intm rel, re2, intm rel, re2, intm U rel, re2, intm U rel, re2, intm CE, int CE EAK YCLE rd VCLE rd</pre> | Store Datates 3 Store Quad Store Construction of the | CIV C.ADI CI | mel ven zuna mel ven zen zuna mel ven zen zen zen zen mel zen zen zen zen zen zen mel zen zen zen zen zen zen zen mel zen zen zen zen zen zen zen zen mel zen zen zen zen zen zen zen zen mel zen | SD raty, runn*16 SQ rati, raty, runn*16 G raty, raty, runn*16 G raty, raty, runn*16 ADDW rdy, rdy, runn*16 ADDW rdy, rdy, runn*16 ADDW rdy, raty, runn*16 ADDT rdy, so, runn*16 ADDT rdy, so, runn*16 ADDT rdy, runn*16 BDD rdy, run, run SUB rdy, rdy, run SUB rdy, rdy, run SILI rdy, rdy, run SILI rdy, rdy, run | Load Store Arithmeti Mul-Add Negati Ni Sign Inje | Load Store ADD SUBtract MULICIPL SQuare RooT Multiply-SDBtract Multiply-SDBtract Multiply-SDBtract we Multiply-SDBtract control States SQUARE SUBTRACT Multiply-SDBtract SQUARE SUBTRACT SQUARE SUBTRACT SQUAR | | L(M, D, Q) S(M, D, Q) S(M, D, Q) S(M, D, Q) S(M, D, Q) S(M, Q | rd,rs2,imm rd,rs2,imm rd,rs1,rs2,imm rd,rs1,rs2 rd,rs1,rs2 rd,rs1,rs2 rd,rs1,rs2 rd,rs1,rs2 rd,rs1,rs2 rd,rs1,rs2,rs3) rd,rs1,rs2,rs) rd,rs1,rs2,rs) rd,rs1,rs2,rs) rd,rs1,rs2 rs5 rd,rs1,rs2 rs5 rd,rs1,rs2 rs5 rd,rs1,rs2 rs5 rd,rs1,rs2 rs5 rd,rs1,rs2 rs5 rd,rs1,rs2 rs5 rs5 rs5 rs5 rs5 rs5 rs5 rs5 | Register 30 20 20 20 20 20 20 20 20 20 20 20 20 20 | ABI Name 2010 FR 59 60-2 60-2 60-2 60-2 60-3 62-1 62-1 62-1 62-1 62-1 | Caller Caller Callee Callee Callee Callee Callee Caller Callee Caller Caller Caller Caller | g Convention Description Hard-wired zero Return address Stack pointer Global pointer Thread pointer Thread pointer Saved register/frame pointer Saved register Function arguments/return value Function arguments Saved registers Temocraties |
| Branches Branch Branch Jump & L Jump & Synch Syn | s Branch = Branch ≠ Branch ≤ Branch ≤ Branch ≤ Branch ≥ Branch ≥ Unsigned h ≥ Unsigned ink 38L Link Register Synch thread Linstr & Data System CALL ReaD CYCLE LE upper Half ReaD TIME | BEQ BEQ BEQ BEQ BES BES BES BES BES BES BES BES BES BES | <pre>No Lot, rei, ram rei, rei2, imm rei, rei, imm cB CB . I CL E . I CL E . rei YCLBH rei YME wid</pre> | Store Oak Store Quad SI Arithmetic ADD ADD Word I ADD Word I ADD SP Imm ⁴ . ADD SP Imm ⁴ . Load Immediat Lead Upper Imm Moly Shifts Shift Left Imm Branches Branch-et | CIV C.ADI CI | rel', ren2', imm ppresses rd, ren2', imm rd, ren1' rd, imm rd, ren1 rd, ren1 | b) for 2 for | Load Store Arithmeti Mul-Add Negati Sign Inje N Min/May | Load Store ADD SUBtract Multiply- DIVide SQuare Root Multiply-ADD Multiply-SUBtract ve Multiply-SUBtract settive Multiply-ADD Ct SiGN source Xor SiGN source Xor SiGN source Multiple | | L(M, D, Q) S(M, D, Q) L(M, D, Q) MDL (S D Q) NDL (S D Q) MUL (S D Q) DIV, (S D Q) S(R*, (S D Q) MSUB, (S D Q) MSUB, (S D Q) MSUB, (S D Q) MSUB, (S D Q) S(NX, (S D Q) FSONN, (S D Q) FSONNX, (S D Q) PHYS, (S)D Q | rd_rp3_,imm rd_rp3_,imm rd_rp3_,imm rd_rp3_,im2 rd_rp3_,rp3 rd_rp3_,rp3 rd_rp3_,rp3 rd_rp3_,rp3 rd_rp3_,rp3 rd_rp3_,rp3 rd_rp3_,rp3 rd_rp3_,rp3 rd_rp3_,rp3 rd_rp3,rp3 rd_rp3,rp3 rd_rp3,rp3 rd_rp3,rp3 rd_rp3,rp3 rd_rp3,rp3 rd_rp3,rp3 rd_rp3 r | Register 20 x0 x2 x3 x5 x5 x3 x5 x3 x5 x3 x5 x3 x3 x1 x12 x12 x12 x12 x12 x12 x12 x12 x12 | ABI Name 2010 Fa 50 90 40 40 40 40 40 40 40 40 40 40 40 40 40 | Caller Caller Caller Caller Caller Caller Caller Caller Caller Caller Caller Caller Caller Caller Caller | g Convention Description Hard-wired zero Return address Stack pointer Global pointer Thread pointer Thread pointer Temporaries Saved register/frame pointer Saved register Function arguments/return value Saved registers Temporaries P temporaries |
| Branches Branch Jump & L Jump & L Synch 5 Synch 5 Synch 5 Synch 5 Synch 5 Synch 5 Synch 7 Synch 7 Sync | s Branch = Branch ≠ Branch ≠ Branch ≥ Branch ≥ H < Unsigned h ≥ Unsigned Link Register Synch thread Instr & Data System CALL ystem BREAK ReaD CYCLE LE upper Half ReaD TIME dE unner Half | SB BEQ BEQ BET BET BET BEE BEE BEE BEE BEE BEE BEE | No Lov, rei, Jami rei, reiz, imm rei, reiz, imm rei, reiz, imm rei, reiz, imm rei, reiz, imm V rei, reiz, imm rd, imm R rd, rei, imm CE CE, I LI EAK VCLB rd IME rd IME rd | Store Double 39 Store Quad SI Arithmetic ADD ADD Word I ADD SP Imm* 1 ADD SP Imm* 4 Load Immediat Load Upper Imm *4 SUI Shifts Shift Left Imm Branches Branch= Branches Jump | CIW C.ADI CIW C.ADI CIW C.ADI CIW C.ADI CI C.ADI CI C.ADI CI C.LUJ CI C.LUJ CI C.LUJ CI C.LUJ CI C.SLI CI C.SLI | main and a series and a series of a series | SD raty sby shares SD raty spy imm*16 raty raty raty rates RDDW raty raty raty raty raty raty ration raty | Load Store Arithmeti Mul-Add Negati N Sign Inje N Min/Max | Icadi Store ADD SUBtract MULICIPU DIVIG SQuare Root Muliciply-SUBtract ve Muliciply-SUBtract ve Muliciply-SUBtract ve Muliciply-SUBtract SIGN source legative Muliciply-ADD Ct SIGN source Minimum MAXmum | | 2 2 2 2 2 2 2 2 2 2 2 2 2 2 | rd,rs2,imm rd,rs2,imm rd,rs1,rs2 rd,rs1,rs2 rd,rs1,rs2 rd,rs1,rs2 rd,rs1,rs2 rd,rs1,rs2 rd,rs1,rs2 rd,rs1,rs2 rd,rs1,rs2 rd,rs1,rs2 rd,rs1,rs2 rd,rs1,rs2,rs3 rd,rs1,rs2,rs3 rd,rs1,rs2,rs2 rd,rs1,rs2,rs2 rd,rs1,rs2,rs2 rd,rs1,rs2 | Register 20 x0 x2 x3 x3 x5-7 x6 x3 x3 x3 x3 x3 x3 x3 x3 x3 x3 x3 x3 x3 | ABI Name 2010 Fa 50 40 40 40 40 40 40 40 40 40 40 40 40 40 | Caller Caller Caller Callee Callee Callee Callee Callee Callee Callee Callee Callee Callee Callee Callee Callee Callee Callee Callee | g Convention Description Hard wired zero Return address Stack pointer Global pointer Thread pointer Thread pointer Saved register/frame pointer Saved register Function arguments/return value Function arguments Saved registers Temporaries FP temporaries FP saved registers |
| Branches Branch Jump & L Jump & Synch Sync | s Branch = Branch ≠ Branch ≠ Branch ≥ Branch ≥ Branch ≥ h < Unsigned h ≥ Unsigned inkr J&L Link Register Synch thread I Instr & Data Vystem BREAK ReaD TYCE ReaD TIME ReaD TIME ReaD TIME | SEL BEQ BEQ BET BET BET BET BET BET I SEL BET BET I FEN I SCAI I RDC I RDC I RDC I RDC I RDC I RDC I RDC I RDC I BEQ BEQ BEQ BEQ BEQ BEQ BEQ BEQ BEQ BEQ | No Let, Far, Jam. rsl, rs2, imm. rsl, rs2, imm. rsl, rs2, imm. rsl, rs2, imm. vsl, rs2, imm. R. rd, imm. R. rd, imm. R. rd, imm. CS.I. LL. CLS. rd. VCLS: rd. NME rd. NME rd. NME rd. | Store Quarts Store Quarts Arithmetic ADD ADD Ward I ADD SP Imm * . ADD SP Imm * . Lead Immediat Lead Upper Imm Branches Branched Branches Branched Branches Dranches Jump Jump Jump | CIW C.ADI CI | main and a parameters and a parameters of a pa | b) rudy rudy fumm 16 So rudy rudy rudy imm 16 rudy rdy rudy imm 16 ADDIT rdy rudy imm 16 SULLT rdy rudy imm BKE rudy rudy imm JALK R0 rudy imm JALK R0 rudy imm | Load Store Arithmeti Mul-Add Negati N Sign Inje N Min/Max Compare | Load Store ADD SUBtrack MULtiply DIVide SQuare RooT Multiply-ADD Multiply-SDBtract we Multiply-SDBtract ve Multiply-SDBtract eqative SIGN source Xor SIGN source Nor SIGN source Minimum MAXimum Compare Ripat | | L(M, D, Q) S(M, D, Q) ADD. (S) (D) JDD. (S) (D) JDD. (S) (D) JDD. (S) (D) DIV. (S) (D) DIV. (S) (D) DIV. (S) (D) PARDD. (S) (D) PARD | rd,ro3,rs3,rs3, rs4,rs3,rs3,rs3, rd,rs1,rs3,rs3, rd,rs1,rs3,rs3, rd,rs1,rs3,rs3, rd,rs1,rs3,rs3, rd,rs1,rs3,rs3,rs3, rd,rs1,rs2,rs3,rs3, rd,rs1,rs2,rs3,rs3, rd,rs1,rs2,rs3, rd,rs1,rs2,rs3, rd,rs1,rs2,rs3, rd,rs1,rs2,rs3, rd,rs1,rs3,rs3, rd,rs1,rs3,rs3, rd,rs1,rs3,rs3, rd,rs1,rs3,rs3, rd,rs1,rs3,rs3, rd,rs1,rs3,rs3, rd,rs1,rs3,rs3, rd,rs1,rs3,rs3, rd,rs1,rs3,rs3, rd,rs1,rs3,rs3, rd,rs1,rs3,rs3,rs3, rd,rs1,rs3,rs3,rs3, rd,rs1,rs3,rs3,rs3,rs3,rs3, rd,rs1,rs3,rs3,rs3,rs3,rs3,rs3,rs3,rs3,rs3,rs3 | Register 20 20 20 20 20 20 20 20 20 20 20 20 20 | ABI Name 2010 E1 30 40 40 40 41 40 41 40 41 40 41 40 41 40 41 40 41 40 41 40 41 40 41 40 41 40 41 40 41 40 41 40 40 40 40 40 40 40 40 40 40 40 40 40 | Caller Caller Caller Caller Caller Caller Caller Caller Caller Caller Caller Caller Caller Caller Caller Caller Caller | g Convention Description Hard wired zero Return address Stack pointer Global pointer Thread pointer Temporaries Saved register/frame pointer Saved register Function arguments/return value Function arguments Saved registers Temporaries FP temporaries FP saved registers FP arguments/return values |
| Branchess Branci Jump & L Jump & L Jump & Synch System 5 System 5 System 5 Sounters ReaD TIN ReaD TIN ReaD TIN | s Branch = Branch ≠ Branch ↓ Branch ≥ Branch ≥ Branch ≥ H < Unsigned h ≥ Unsigned h ≥ Unsigned Link Register Synch thread Instr & Data System BREAK ReaD CYCLE LE upper Half ReaD TIME HE upper Half NSTR RETred D unner Half | A BEC SE BEC SE BET BEE BEE BEE BEE BEE BEE BEE | <pre>AD LAW, Extraction Intel, Fast2, intm rel, res2, intm rel, res1, res1, intm rel, res1, res2, intm rel, res1, res3, intm rel, res3, intm r</pre> | Store Double 39 Store Quad 31 Arithmetic ADD ADD Word 1 ADD Word 1 ADD SP Imm 4 ADD SP Imm 4 ADD SP Imm 4 Load Immediat Load Immediat Load Immediat Branches Branch=(Branches Branch=(Branches Imm 8 Jump Aung Amm Jump Register | CIW CADE CIW CADE CI CIW CADE CI CLUE CI CLUE CLUE CLUE CLUE CLUE CLUE CLUE CLUE | rel', res2', imm portesses monoportesses rel, res2', imm rel, res2', imm rel, res2', imm rel, res1', imm rel, res1', imm imm imm rel, res1', imm rel, res1', imm imm | b) first story shares S0 rel', rel', rimm*16 G rel, rel, rel', rimm*16 G rel, rel, rel, rim ADDW rel, rel, rel ADDI rel, sp, imm*16 ADDI rel, sp, imm*16 ADDI rel, rel, rel, rel ADD rel, rel, rel, rel SUB rel, rel, rel, rel SUB rel, rel, rel, rel SELI rel, rel, rel, rel BEQ rol', x0, imm JADA x0, imm JADA x0, imm JAL x0, imm JAL x0, imm JAL x0, rel, 0 | Laad Store Arithmeti Negati Ni Sign Inje N Min/Max Compare | Load Store ADD SUBroad MULtiply Divide SQuare RooT Multiply-SUBtract We Multiply-SUBtract We Multiply-SUBtract We Multiply-SUBtract SIGN source gative Multiply-SUBtract Xor SIGN source Minimum MAXimum Compare Float - Compare Float | | L.(M, D, Q) S(M, D, Q) S(M, D, Q) ADD. (S) D(Q) ADD. (S) D(Q) ADD. (S) D(Q) ADD. (S) D(Q) MDL. (S) D(Q) S(M, Q, Q) MIL. (S) D(Q) S(M, Q, Q) S(M, Q) MUL. (S) D(Q) S(M, Q) S(M, Q) < | rd,rs2,imm rd,rs2,imm xs1,rs2,imm rd,rs1,rs2 rd,rs1,rs2 rd,rs1,rs2 rd,rs1,rs2 rd,rs1,rs2 rd,rs1,rs2 rd,rs1,rs2 rd,rs1,rs2,rs rd,rs1,rs2,rs rd,rs1,rs2,rs rd,rs1,rs2 rd,rs1,r | Register 20 20 22 23 23 25-7 25 25 25 25 25 25 25 25 25 25 25 25 25 | ABI Name 2630 FR 50 60 60 60 60 61 60 61 60 61 60 7 61 61 60 7 61 60 7 61 60 7 60 7 | Caller | g Convention Description Hard wired zero Return address Stack pointer Global pointer Thread pointer Thread pointer Temporaries Saved register/frame pointer Saved register Function arguments/return value Function arguments/return values FP Saved registers Temporaries FP Saved registers FP saved registers FP saved registers FP saved registers FP arguments/return values FP arguments/ |
| Branchess Branci Iump & L Jump & L Synch = Synch = Syn | s Branch = Branch ≠ Branch ≠ Branch ≥ Branch ≥ h < Unsigned Link Register J&L Link Register J&L Link Register J&L Link Register J&L Link Register J&L Link Register J&Synch Thread J Instr & Data System CALL ystem BREAK ReaD TIME ReaD TIME ReaD TIME HE upper Half | BEQ BEQ BEQ BEQ BEC BE BE BEC BEC BEC BEC BEC BEC BEC B | <pre>No Lot, For J Anni rol, For Z, innn rol, roz Z, innn rol, roz Z, innn rol, roz Z, innn U rol, roz Z, innn U rol, roz Z, innn U rol, roz Z, innn CZ CZ, I CZ CZ, I CZ CZ, I CZ CZ, I CZ CZ, I CZ CZ, I CZ CZ, I CZ CZ, I CZ CZ CZ, I CZ CZ CZ, I CZ CZ CZ CZ CZ CZ CZ CZ CZ CZ CZ CZ CZ</pre> | Store Obline 33 Store Quad 33 Arithmetic ADD ADD Word 1 ADD Word 1 ADD SP Imm 4 ADD SP Imm 4 ADD SP Imm 4 ADD SP Imm 4 ADD SP Imm 4 Lead Upper Imm MoV/ SUI Shifts Shift Left Imm Branches Branch=6 Jump B Link J3K Jump B Link Spitk Benizte | CR C.ADI CR C.ADI CI C.ADI CI C.ADI CI C.ADI CI C.ADI CI C.ADI CI C.LI CI C.JI CI CI C.JI CI CI C.JI CI CI C.JI CI CI C.JI CI CI C.JI CI CI CI CI C.JI CI CI C | main market and a series of the series of th | b) for the phy fully fully fully fully fully for the phy fully fully for the phy fully fully for the phy for the phy fully for the phy for phy | Load Store Arithmeti Negati N Sign Inje Min/Max Compare | Load Store G ADD SUBtract MULEply, DIVide SQuare Root Multiply-ADD Multiply-SDBtract we Multiply-SDBtract we Multiply-SDBtract SIGN source gative Multiply-SDBtract SIGN source SIGN source Xor SIGN source Xor SIGN source Compare Float = Compare Float = | TINCE DATES TO SALES | L.(M, D, Q) S(M, D, Q) S(M, D, Q) XD0. (S(D)Q) S(R0, (S(D)Q)) S(R0, (Q)Q) S(R, (S(D)Q)) S(R, (S(D)Q)) S(R, (S(D)Q)) S(R, (S(D)Q)) S(R, (S(D)Q)) S(R, (S(D)Q)) | <pre>rd_r0_iam rd_r0_iam rs_rs_iss_im rd_rs_iss_iss_ rd_rs_iss_ rd_rs_iss_ rd_rs_iss_ rd_rs_iss_ rd_rs_iss_ rd_rs_iss_ rd_rs_iss_ rd_rs_iss_ rd_rs_iss_ rd_rs_iss_ rd_rs_iss_ rd_rs_iss_ rd_rs_iss_ rd_rs_iss_iss_ rd_rs_iss_iss_ rd_rs_iss_iss_ rd_rs_iss_iss_ rd_rs_iss_iss_ rd_rs_iss_iss_iss_ rd_rs_iss_iss_iss_iss_iss_iss_iss_iss_iss_</pre> | Register 20 20 20 20 20 20 20 20 20 20 20 20 20 | ABI Name 2510 FB 50 50 20 40 40 40 40 40 40 40 40 40 40 40 40 40 | Caller | g Convention Description Hard-wired zero Return address Stack pointer Global pointer Thread pointer Thread pointer Saved register/frame pointer Saved register Function arguments/return value Function arguments FP temporaries FP arguments/return values FP arguments |
| Branches Branci Iump & L Jump & L Jump & Synch & Synch | s Branch = Branch + Branch + Branch + Branch + Branch + Branch + Branch + Jack - Synch thread Instr & Data System CALL system BREAK ReaD CYCLE LE upper Half NSTR RETired TR upper Half | BEL BRC BLT BLT BEC BLT BEC BLT BEC BLT BEC BLT BLT BLT BLT BLT BLT BLT BLT BLT BLT | <pre>No Lot, rei, raz, inm rei, raz, inm rei, raz, inm roi, raz, inm voi, raz, inm voi, raz, inm roi, raz, raz, raz, raz, raz, raz, raz, raz</pre> | Store Double 33 Store Quad 31 Arithmetic ADD ADD Word 1 ADD Word 1 ADD SP Imm ⁴ . ADD SP Imm ⁴ . Stifts Shift Left Imm Branches Branch-e Branchet Jump Aunit Left Imm Jump Registe Jump Buink Registe System Erg. ⁴⁰⁰ | CLUE CLUE CR C.ADI CR C.LUI CR CR C | rel', res2', imm ppress rel', res2', imm press rel', res2', imm rel', rel', imm rel', rel', imm rel', rel', imm rel', rel', imm imm imm rel', rel', imm imm rel', rel', rel', imm imm rel', rel', imm | b) rut, rut, rut, rut, mm*16 So rut, rut, rut, rum*16 rut, rd, rut, ADDW rd, rd, run rd, rd, run rut, rd, run ADDW rd, rd, imm*16 ADDW rd, rd, imm*16 ADDW rd, rd, imm*16 ADDW rd, rs, rum*16 ADDW rd, rs, rum*16 ADDW rd, rs, rum*16 ADDW rd, rs, rum*16 ADDW rd, rs, rum*16 SULL rd, rd, run BKE rut, rk0, imm JAL rd, imm JAL rd, imm JAL ra, imm JAL ra, imm JAL ra, imm JAL ra, imm JAL ra, imm JAL ra, imm | Load Store Arithmeti Nut-Add Negati Ni Sign Inje Min/Max Compare | Load Store ADD SUBtract MULtiply- DIVide SQuare Root Multiply-SDBtact we Multiply-SDBtact we Multiply-SDBtact solf SuBtact we Multiply-SDBtact gative Stiffs Source Xor SiGN source Xor SiGN source Xor SiGN source Minimum Compare Float - Compare Float - Compare Float - Compare Float - Suffor Closefly Turue | THE AND REALFIER STATE | L(M,D,Q) S(M,D,Q) ADD.(S(D)Q) JUB.(S(D)Q) JUB.(S(D)Q) JUB.(S(D)Q) JUB.(S(D)Q) HUL.(S(D)Q) HUL.(S(D)Q) HUL.(S(D)Q) HUL.(S(D)Q) HUL.(S(D)Q) HUL.(S(D)Q) HUL.(S(D)Q) HUL.(S(D)Q) HUL.(S(D)Q) JUB.(S(D)Q) | <pre>rd_rc0_inm rd_rc0_inm rd_rc0_inm rd_rc1,rs2 rd_rc1</pre> | Register 20 21 22 23 23 23 23 23 23 23 23 23 | ABI Name 2530 59 99 99 10-2 10/39 40-2 40-4 40-4 40-4 40-4 40-4 40-4 40-4 | Caller | g Convention Description Hard-wired zero Return address Stack pointer Thread pointer Thread pointer Temporaries Saved register/frame pointer Saved register Function arguments/return value Function arguments/return value Function arguments For saved registers FP saved registers FP arguments/return values FP arguments/ FP ar |
| Branches Branch Jump & L Jump & Synch Synch S Synch Sy | Branch ≠ Branch ≠ Branch ≠ Branch ≥ Branch ≥ Branch ≥ Branch ≥ Network = Network = | BEQ BEQ BET BET BET BET BE BET BE BET BE BET BET | No LAN FRIFAMM rel, rel2, imm rel, rel2, imm rel, rel2, imm rel, rel2, imm rel, rel2, imm rd, rel2, imm rd, imm R rd, rel, imm CE CE CE EAK XCLE rd NSTRET rd NSTRET rd | Store Obtaine 33 Store Quad SI Arithmetic ADD ADD Word I ADD Word I ADD SP Imm * 4 Load Immediat Load Immediat Load Immediat Load Immediat Shifts Shift Left Imm Branches Branch=(Branches J Branchat J Jump & Link Registe Jump & Link Registe System Env. BREA | CI CLUE CS C.SQ CADITOR CI C.ADITOR CI C.ADITOR CI C.ADITOR CI C.ADITOR CI C.ADITOR CI C.ADITOR CI C.ADITOR CI C.SUITOR CI C.SUITOR CI C.SUITOR CI C.SUITOR CI C.SUITOR CI C.JAITOR CI C.J | rel', res2', imm population rel, res2', imm rel, imm rel, imm rel, imm rel, imm rel, rel rel, imm rel, rel rel', rel', re | b) for a for | Load Store Arithmeti Negati Sign Inje N Min/Max Compare | Load Store ADD SUBtract MULtply- DIVide SQuare Root Multply-SUBtract Ver Multply-SUBtract Ver Multply-SUBtract SQuare Root Multply-SUBtract SIGN Source SIGN Source SIGN Source Minimum MAXmum Compare Float s Compare Float s Compare Float s Station Classify Type | A CONTRACTOR OF A CONTRACTOR | L(M, D, Q) S(M, C, S) S(M, S) | rd,rs2,imm rd,rs2,imm rd,rs1,rs2 rd,rs1,rs2 rd,rs1,rs2 rd,rs1,rs2 rd,rs1,rs2 rd,rs1,rs2 rd,rs1,rs2 rd,rs1,rs2 rd,rs1,rs2,rs3 rd,rs1,rs2,rs3 rd,rs1,rs2,rs3 rd,rs1,rs2,rs3 rd,rs1,rs2,rs3 rd,rs1,rs2,rs3 rd,rs1,rs2 rd,rs2 rd,rs1,rs2 rd,rs1,rs2 rd,rs1,rs2 | Register 20 30 32 32 35 35 35 35 35 35 35 35 35 35 | ABI Name 2010 50 50 50 20 20 20 20 20 20 20 20 20 20 20 20 50 20 20 50 20 20 50 20 20 50 20 20 50 20 50 50 50 50 50 50 50 50 50 50 50 50 50 | Caller | g Convention Description Hard wired zero Return address Stack pointer Global pointer Thread pointer Thread pointer Saved register/frame pointer Saved registers Function arguments/return value Function arguments Saved registers FP saved registers FP saved registers FP arguments/return values FP arguments FP arguments FP arguments FP temporaries FP te |
| Branches Branci Jump & L Jump & L Jump & L Jump & L Jump & L Synch = 2 Synch Synch = 2 Synch = 2 | Branch ≠ Branch ≠ Branch ≠ Branch ≥ h < Unsigned h ≥ Unsigned h ≥ Unsigned h ≥ Unsigned tink Register Synch thread i Instr & Data System CALL ystam BREAK ReaD TME E upper Half ReaD TME #E upper Half ReaD TME #E upper Half STR REITOR TR upper Half STR BETOR TR upper Half STR BETOR TR upper Half STR BETOR TR upper Half STR BETOR TR upper Half STR BETOR STR BETOR STR BETOR STR | SB BGE SB | No Lot, For J. Amm. rel, re2, imm. rel, re2, imm. rel, re2, imm. rel, re2, imm. rd, jmm. R. rd, rel, imm. CS. J. LL. CS. J. LL. CS. J. LL. CS. J. LL. CS. J. LL. STATES rd. NSTRET rd. NSTRET rd. NSTRET rd. NSTRET rd. NSTRET rd. | Store Obline 33 Store Quad 31 Arithmetic ADD ADD Ward 1 ADD SP Imm * . Load Immediak Load Upper Imm MoVr Shifts Shift Left Imm Branches Branched Branches Jump Jump Jump St. Link J&I Jump & Link Agelste System Env. BREAL | CLUE CLUE CLUE CLUE CLUE CLUE CLUE CLUE | xe1', xe2', imm xe1', xe2', imm DDICESS monocol rd, rsi cdiam rd, imm rd, rsi rd, imm rd, rsi rd, imm rd, rsi imm rd, rsi | SD rat, rst, rimm*16 SG rat, rst, rst, rimm*16 SG rat, rst, rst, rimm*16 SG rat, rst, rst, rst, rst, rst, rst, rst, rs | Load Store Arithmeti Mul-Add Negati N Sign Inje N Min/Max Compare Categoriz Configura | Load Store ADD SUBtrack MULtiply DIVide SQuare Root Multiply-ADD Multiply-SUBtract we Multiply-SUBtract we Multiply-SUBtract we Multiply-SUBtract egative StGN source SiGN source SiGN source Minimum MAXimum Compare Float = Compare Float = | The state of the s | L(M, D, Q) S(M, D, Q) ADD. (S) (D)(2) JUD. (S) (D)(2) JUD. (S) (D)(2) JUD. (S) (D)(2) JUD. (S) (D)(2) BOX. (S) (D)(2) HILL. (S) (D)(2) EXC. (S) (D)(2) HILL. (S) (D) | rd,r03,r03,r03 rd,r03,r03,r03 rd,r03,r03,r03 rd,r03,r03,r03 rd,r03 rd,r03 | Register 20 20 20 25-7 25 25-7 25 25-7 25 25-7 25 25-7 210-21 20-7 20-27 20-21 20-21 210-21 210-21 210-21 210-21 210-21 210-21 210-21 210-21 | 2620 Fa 50 Fb 50 60 50 60 60 60 60 60 60 60 60 60 60 60 60 60 | Caller | g Convention Description Hard wired zero Return address Stack pointer Global pointer Thread pointer Thread pointer Temporaries Saved register/frame pointer Saved register Function arguments/return value Function arguments Saved registers Temporaries FP saved registers FP arguments/return values FP arguments/return values FP arguments FP saved registers FP acude registers FP acude registers FP acude registers FP acude registers FP acude registers FP acude registers FP temporaries FP temporaries |
| Branches Branches Jump & L Jump & L Jump & K Synch S Synch S S Synch S S S S S S S S S S S S S S S S S S S | Branch # Branch 4: Branch 4: Branch 4: Branch 2: Branch 2: Bran | SB BAR BAR BAR BAR BAR BAR BAR BAR BAR BAR | 10 1.0., 2.0.; J.mm. rel, r.s.2, innm. rel, r.s.2, innm. rel, r.s.2, innm. rel, r.s.2, innm. U rel, r.s.2, innm. R. rd, rel, innm. R. rd, rel, innm. R. rd, rel, innm. R. rd, rel, innm. CE CS.I. LLI. F. rel, innm. YCLB. rd NETERE rd NSTREET rd NSTREET rd SSTREET rd 19 1814 2 | Subre Double 39 Store Quad 31 Arithmetic ADD ADD Word 1 ADD Word 1 ADD SP Imm 4 ADD SP Imm 4 ADD SP Imm 4 ADD SP Imm 4 ADD SP Imm 4 Load Immediati Load Immediati Load Immediati Load Immediati Shifts Shift Left Imm Branches Branch=6 Branchet Left Imm Jump ALink 381 Jump & Link 381 Jump | CF C-B2 CF C-B | Test / Junit rest / Junit PDFCSS6 mpredsm rd, imm rd, rs1 | SD 140, 500, 1100, 50 SG 111, 110, 110, 110, 110, 110, 110, 110 | Laad Store Arithmeti Negati Nisign Inje B Min/Max Compare | Load Store ADD SUBtrack MULtiply D1Vide SQuare RooT Multiply-SDBtrack getive Multiply-SDBtrack getive Multiply-SDBtrack getive Multiply-SDBtrack getive StGN source Mikimum MAXimum Compare Float - Compare Float Compare Float Station Classify Type tation Read Status Read Rounding Mode Bead Float | Participal and a service of the serv | L(M, D, Q) S(M, D, Q) ADD. (S) (D) ADD. (S) (D) ADD. (S) (D) ADD. (S) (D) ADD. (S) (D) ADD. (S) (D) S(R, Q) (S) (D) S(R, Q) S(R, Q) S(R | rd,r02,.2mm rd,r02,.2mm rd,r03,.4mm rd,r03,r02 rd,r03 | Register 30 20 22 23 23 25 25 25 25 25 25 25 25 25 25 25 25 25 | 2850 Ea 59 69 60 60 60 60 60 60 60 60 60 60 60 60 60 | Caller | g Convention Description Hard-wired zero Return address Stack pointer Global pointer Thread pointer Temporaries Saved register/frame pointer Saved register Function arguments/return value Function arguments/return value Function arguments FP temporaries FP saved registers FP arguments/return values FP arguments FP arguments FP arguments FP arguments FP temporaries |
| Branches Branches Jump & L Jump & L Jump & K Synch & Synch & Synch & Synch & Synch & Synch & S | S Branch = Branch ≠ Branch ≠ Branch ≥ h < Unsigned h ≥ Unsigned h | C BEL BEL BEL BEL BEL BEL BEL BEL BEL BEL | No Los rei ram rei rez 2 inem rei rei rei rei rei rei rei rei rei rei | Store Double 33 Store Quad 31 Arithmetic ADD ADD Word 1 ADD SP Imm *1. ADD SP Imm *1. ADD SP Imm *1. Load Immediat Load Immediat Load Immediat Shifts Shift Let Imm *1. Shifts Shift Let Branch=(Branches Branch=(Jump & Link Jsk Jump & Link Jsk Jump & Link Registe System Env. BREA 11 8 7 6 0 rd genede | C C C C C C C C C C C C C C C C C C C | 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 | SD 1402 sby shares SQ rai', ra2', simm*16 G rad, ra2', simm*16 ADDW rd, rd, rd, inn Y rd, rd, rai ADDW rd, rd, inn Y rd, rd, rai ADDI rd, rd, inn ADDI rd, rd, inn ADDI rd, rai, so, inn*4 ADDI rd, rai, so, inn*4 ADDI rd, rai, so, inn BBC rd: rx0, so, inn JAL rd, rain JAL rd, inn JAL rd, inn <tr tr=""></tr> | Load Store Arithmeti Negati N Sign Inje N Min/Max Compare Categoriz Configura | Load Store ADD SUBtract MULtiply- DIVide SQuare Root Multiply-SUBtract ve Multiply-SUBtract ve Multiply-SUBtract ve Multiply-SUBtract SIGN source SIGN source Minimum MAXmum Compare Float so Compare Float = Compare Float = Compare Float so Compare Float = Compare Float so Compare Float so Compar | PARTICIPATION DISTRICT RATIONAL | L(M, D, Q) S(M, D, Q) XD0, (S)(D(Q) XD0, (S)(D(Q) XD0, (S)(D(Q) XD0, (S)(D(Q) XD0, (S)(D(Q) D(Q) D(Q) S(R, (S)(D(Q) S(R, (S)(D(Q) S(R, (S)(D(Q) S(R, (S)(D(Q) S(R, (S)(D(Q) S(R, (S)(D(Q) S(R, (S)(Q) S(R, (S)(Q) | <pre>rd_rul_ism rd_rul_ism rs_rul_ism rd_rul_ism rd_rul_ism rd_rul_ism rd_rul_ism rd_rul_ism rd_rul_ism rd_rul_ism rd_rul_ism rd_rul_ism rd_rul_ism rd_rul_rul_ism rd_rul_rul_ism rd_rul_rul_ism rd_rul_rul_</pre> | Register 20 20 22 23 23 25 25 25 25 25 25 25 25 25 25 25 25 25 | 2830 881 Name 2830 89 80 80 80 80 80 80 80 80 80 80 80 80 80 | Saver Saver Caller Callee Callee Callee Callee Caller | g Convention Description Hard wired zero Return address Stack pointer Global pointer Thread pointer Thread pointer Saved register Function arguments/return value Function arguments Saved registers Temporaries FP saved registers FP arguments FP arguments FP arguments FP arguments FP arguments FP temporaries FP temporaries FP temporaries FP temporaries |
| | | | | | | | | | | | | | | | | |
| Branches Branches Jump & L Jump & L Synch S Synch S S Synch S S S S S S S S S S S S S S S S S S S | Branch # Branch # Branch # Branch # Branch # Branch 2 | A BEC BEC BEC BEC BEC BEC BEC BEC | 10 10, 201, 200, 200 rnl, rol2, innn rol1, rol2, innn rol1, rol2, innn rol1, rol2, innn U rol1, rol2, innn R rd, rol1, innn R rd, rol1, innn CE I II rol1, rol2, innn R rd, rol1, innn CE I II rol1 YCLBH rd NETERET rd NETERET rd STAL Function 19 1514 19 1514 19 rol1 19 1514 19 1514 10 1514 11 function | Store Double 33 Store Quad 31 Store Quad 31 Arithmetic ADD ADD Word 1 ADD Word 1 ADD SP Imm ⁴ . Branches Branches Branches Jump & Link Registe System Env. BREAM 11 & 7 & 0 n n 1 & 0 oppde 1 1 & 7 & 0 n 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | CF C C C C C C C C C C C C C C C C C C | xel , years , y | SD 1 m2 y by J lime*0 SD 1 m1 y by J mm*16 ST m1 y rd, rd, rm*16 Fr2, sp, imm*16 Fr2, sp, imm*16 ADDW rd, rd, rm ADDW rd, rd, inn ADDT rd, sp, imm*16 ADDT rd, rd, inn ADDT rd, sp, imm*16 ADDT rd, sp, imm*16 SUB rd, rd, inn ADDT rd, sp, imm*16 SUB rd, yc, inn BRG rn1 ', x0, inn BNE rn1 ', x0, inn JAL x0, inn <t< td=""><td>Load Store Arithmeti Nut-Add Negati Nu Sign Inje Min/Max Compare Configura</td><td>Load Store ADD SUBtract MULtiply- DIVide SQuare Root Multiply-SDBtact we Multiply-SDBtact we Multiply-SDBtact we Multiply-SDBtact we Multiply-SDBtact we Multiply-SDBtact we Multiply-SDBtact we Multiply-SDBtact we Multiply-SDBtact compare Float = Compare Float = Compare</td><td></td><td>L(M,D,Q) S(M,D,Q) ADD.(S(D)Q) JDD.(S(D)Q) JDD.(S(D)Q) JDD.(S(D)Q) JDD.(S(D)Q) HUL.(S(D)Q) HUL.(S(D)Q) HUL.(S(D)Q) HUL.(S(D)Q) HUL.(S(D)Q) HUL.(S(D)Q) HUL.(S(D)Q) HUL.(S(D)Q) HUL.(S(D)Q) JOD.(S(D)Q) JOD.(S(D)Q) JDD.(S(D)Q)</td><td><pre>rd_rv2imm rd_rv2imm rd_rv2imm rd_rv2imm rd_rv2imm rd_rv2imn rd_rv2im2 rd_rv2im2 rd_rv2im2 rd_rv2im2 rd_rv2rv2rv2 rd_rv2rv2rv2 rd_rv2rv2rv2 rd_rv2rv2rv2 rd_rv2rv2rv2 rd_rv2rv2 rd_rv2rv2 rd_rv2rv2 rd_rv2rv2 rd_rv2rv2 rd_rv2rv2 rd_rv2rv2 rd_rv2rv2 rd_rv2rv2 rd_rv2rv2 rd_rv2 r</pre></td><th>Register 30 20 22 23 25 25 25 25 25 25 25 25 25 25 25 25 25</th><td>2850 881 Name 2850 89 89 80 80 80 80 80 80 80 80 80 80 80 80 80</td><td>Caller Caller</td><td>g Convention Description Hard-wired zero Return address Stack pointer Thread pointer Thread pointer Temporaries Saved register/frame pointer Saved register Function arguments/return value Function arguments/return value Saved registers Temporaries FP saved registers FP arguments/return values FP arguments/return values FP arguments/ FP asved registers FP temporaries</td></t<> | Load Store Arithmeti Nut-Add Negati Nu Sign Inje Min/Max Compare Configura | Load Store ADD SUBtract MULtiply- DIVide SQuare Root Multiply-SDBtact we Multiply-SDBtact we Multiply-SDBtact we Multiply-SDBtact we Multiply-SDBtact we Multiply-SDBtact we Multiply-SDBtact we Multiply-SDBtact we Multiply-SDBtact compare Float = Compare | | L(M,D,Q) S(M,D,Q) ADD.(S(D)Q) JDD.(S(D)Q) JDD.(S(D)Q) JDD.(S(D)Q) JDD.(S(D)Q) HUL.(S(D)Q) HUL.(S(D)Q) HUL.(S(D)Q) HUL.(S(D)Q) HUL.(S(D)Q) HUL.(S(D)Q) HUL.(S(D)Q) HUL.(S(D)Q) HUL.(S(D)Q) JOD.(S(D)Q) JOD.(S(D)Q) JDD.(S(D)Q) | <pre>rd_rv2imm rd_rv2imm rd_rv2imm rd_rv2imm rd_rv2imm rd_rv2imn rd_rv2im2 rd_rv2im2 rd_rv2im2 rd_rv2im2 rd_rv2rv2rv2 rd_rv2rv2rv2 rd_rv2rv2rv2 rd_rv2rv2rv2 rd_rv2rv2rv2 rd_rv2rv2 rd_rv2rv2 rd_rv2rv2 rd_rv2rv2 rd_rv2rv2 rd_rv2rv2 rd_rv2rv2 rd_rv2rv2 rd_rv2rv2 rd_rv2rv2 rd_rv2 r</pre> | Register 30 20 22 23 25 25 25 25 25 25 25 25 25 25 25 25 25 | 2850 881 Name 2850 89 89 80 80 80 80 80 80 80 80 80 80 80 80 80 | Caller | g Convention Description Hard-wired zero Return address Stack pointer Thread pointer Thread pointer Temporaries Saved register/frame pointer Saved register Function arguments/return value Function arguments/return value Saved registers Temporaries FP saved registers FP arguments/return values FP arguments/return values FP arguments/ FP asved registers FP temporaries |
| Branches Branches Jump & L Jump & L Jump & L Synch S Synch S S Synch S S S S S S S S S S S S S S S S S S S | s Branch ≠ Branch ≠ Branch ≠ Branch ≥ h < Unsigned h < Unsigned h < Unsigned link Register Synch thread link Register Synch thread link Register System CALL upper Half ReaD YCHE LE upper Half ReaD TIME de upper Half SSTR REFired TR upner Half SSTR REFired IR upner Half SSTR REFired IR upper Half SSTR REFIRE IN SSTR SSTR SSTR SSTR SSTR | A BALL | 10 1.0.7.2.2.7.100. real, res2, innm real, res2, innm real, res2, innm real, res2, innm 0 real, res2, innm 0 real, res2, innm real, real, innm real, innm R rd, real, innm C2 C2 C3 real, innm C4 real, innm C2 C2 C5 I.I. III real VCL25 red NSTRETH red INSTRETH red 19 15 19 15 res1 funct3 res1 funct3 | Store Double 33 Store Quad 31 Arithmetic ADD ADD Word 1 ADD Word 1 ADD SP Imm */ ADD SP Imm */ Load Upper Imn Branches Branch=(Branches Branches Branch=(Branches Branches B | CF CLUE CF CF C | rel ', res2 ', imm PD/PESS mpress rd, imm ICCOMS rd, imm rd, rs1 | SD 1 How y by J lime's SD Tai', rai', rai', rai', rm*16 Tai, rai, rai', rai', rai', rm*16 Tai, rai, rai', rai, rm*16 ADDW rai, rai, rai, rai ADDW rd, rd, rai ADDW rd, rd, rain ADDW rd, rd, rain ADDW rd, rd, rain ADDI rd, rd, rain ADDI rd, rain*4 ADDI rd, rain*4 ADDI rd, rain*4 ADDI rd, rai, rai SUB rd, rai, rai SUB rd, rai, rai SULT rd, rai, rai JAL rai, rai, rai JAL rai, rain JAL rain rain <td>Laad Store Arithmeti Negati Ni Sign Inje N Min/Max Compare Categoriz Configura</td> <td>Load Store ADD SUBtract MULtiply- DIVide SQuare RooT Multiply-SDBtract We Multiply-SDBtract We Multiply-SDBtract We Multiply-SDBtract SiGN source SiGN source Minimum MAXimum Compare Float so Compare Status Read Rounding Mode Read Flags Swap Status Reg Swap Rounding Mode</td> <td>A REAL PROPERTY DISTRICT REAL PROPERTY REAL</td> <td>L(M, D, Q) S(M, D, Q) ADD. (S) D(Q) ADD. (S) D(Q) ADD. (S) D(Q) ADD. (S) D(Q) ADD. (S) D(Q) S(R, (S) D(Q) S(R, (S) D(Q) HNLD, (S) D(Q) HNLD, (S) D(Q) S(R, (S) D(Q) S(R, (S) D(Q)) S(R, (S) D(Q))</td> <td>rd,rs2,imm rd,rs2,imm xs1,rs2,imm rd,rs1,rs2 rd,rs1,rs1 rd,rs1,rs1 rd,rs1,rs1 rd,rs1,rs1 rd,rs1,rs1 rd,rs1,rs1 rd,rs1,rs1 rd,rs1,rs1 rd,rs1,rs1 rd,rs1,rs1 rd,rs1 r</td> <th>Register 250 231 233 235-7 259 230-33 25-7 239-23 25-7 259-9 210-31 252-25 259-9 210-11 252-25 259-9 210-11 252-25 259-33</th> <td>2010 2010 2010 2010 2010 2010 2010 2010</td> <td>Caller Caller</td> <td>g Convention Description Description Hard wired zero Return address Stack pointer Global pointer Thread pointer Thread pointer Tamporaries Saved register/frame pointer Function arguments/return value Function arguments Saved registers Temporaries FP saved registers FP arguments FP saved registers FP temporaries</td> | Laad Store Arithmeti Negati Ni Sign Inje N Min/Max Compare Categoriz Configura | Load Store ADD SUBtract MULtiply- DIVide SQuare RooT Multiply-SDBtract We Multiply-SDBtract We Multiply-SDBtract We Multiply-SDBtract SiGN source SiGN source Minimum MAXimum Compare Float so Compare Status Read Rounding Mode Read Flags Swap Status Reg Swap Rounding Mode | A REAL PROPERTY DISTRICT REAL PROPERTY REAL | L(M, D, Q) S(M, D, Q) ADD. (S) D(Q) ADD. (S) D(Q) ADD. (S) D(Q) ADD. (S) D(Q) ADD. (S) D(Q) S(R, (S) D(Q) S(R, (S) D(Q) HNLD, (S) D(Q) HNLD, (S) D(Q) S(R, (S) D(Q) S(R, (S) D(Q)) S(R, (S) D(Q)) | rd,rs2,imm rd,rs2,imm xs1,rs2,imm rd,rs1,rs2 rd,rs1,rs1 rd,rs1,rs1 rd,rs1,rs1 rd,rs1,rs1 rd,rs1,rs1 rd,rs1,rs1 rd,rs1,rs1 rd,rs1,rs1 rd,rs1,rs1 rd,rs1,rs1 rd,rs1 r | Register 250 231 233 235-7 259 230-33 25-7 239-23 25-7 259-9 210-31 252-25 259-9 210-11 252-25 259-9 210-11 252-25 259-33 | 2010 2010 2010 2010 2010 2010 2010 2010 | Caller | g Convention Description Description Hard wired zero Return address Stack pointer Global pointer Thread pointer Thread pointer Tamporaries Saved register/frame pointer Function arguments/return value Function arguments Saved registers Temporaries FP saved registers FP arguments FP saved registers FP temporaries |
| Branches Branches Jump & L. Jump & L. Jump & L. Synch S Synch S Synch S Synch S Synch S Synch S Synch S Synch S S Sunch S S Synch S S S S S S S S S S S S S S S S S S S | Branch ≠ Branch ≠ Branch ≠ Branch ≥ h < Unsigned h ≥ State system CALL ystam BREAK ReaD TME ReaD TME ReaD TME H upper Half STR RETrad H upper Half STR RETrad S 3 25 25 unsit hmm[1:0] mm[1:0] imm](0.5) | Control of the second s | No. Co., For J., Far.2, Jinm. reb 1, red 2, Jinm. reb 1, red 2, Jinm. reb 1, red 2, Jinm. D = reb 2, Jinm. reb 1, red 1, Jinm. reb 1, red 1, Jinm. reb 1, red 1, There red 1, T | Store Quad SI Store Quad SI Arithmetic ADD ADD Immedia ADD Mord I ADD SP Imm * . Load Immediat Load Immediat Load Upper Imm Moly Shifts Shift Left Imm Branches Branched Branches Branches Jump Jump Jump Registe Jump & Link Registe System Env. BREA 11 & 7 & 0 nd opede nd opede Imm (d) Imm(1) opede | CI C.LUE CI C.LUE CI C.ADI CI C.ADI CI C.ADI CI C.ADI CI C.ADI CI C.LI CI CI C.LI CI C.LI CI CI | Interpretation Interpretation | SD 1 Hoz y By J 1000 - 5 SG 1 rai ', rai ', rinm '16 Proj ray, inm '16 Proj ray, ray, ray Proj ray | Laad Store Arithmeti Mul-Add Negati Ni Sign Inje N Min/Max Compare Categoriz Configura | Load Store ADD SUBtrack MUBLiply SQuare Root Multiply-SDBtract wellutiply-SDBtract wellutiply-SDBtract wellutiply-SDBtract sight source Sight source Sight source Minnum MAXmum Compare Float so Compare Float so Swap Status Reg Bauding Mode Swap Floatss Swap Status Reg Swap Floatss Swap | THE REAL REAL REAL FRANCE | L(M, D, Q) S(M, D, Q) ADO. (S(D(Q)) JUD. (S(D(Q)) JUD. (S(D(Q))) JUD. (S(D(Q)) JUD. (S(D(Q))) FRID. (| rd,r02,200 rd,r02,200 rd,r02,200 rd,r02,200 rd,r03,r02 rd,r03 rd,r03 rd,r03 rd rd rd rd rd rd rd rd rd rd | Register 23 34 25-7 23 25-7 25 25-7 25-1 25-7 25-9 25-9 25-9 25-9 25-9 25-9 25-9 25-9 | 2850 FRAME 2850 FRA 50 50 50 50 50 50 50 50 50 50 50 50 50 | Caller | g Convention Description Hard-wired zero Return address Stack pointer Thread pointer Thread pointer Temporaries/frame pointer Saved register/frame pointer Saved register Function arguments/return value Function arguments Saved registers Temporaries FP saved registers FP arguments/return values FP arguments FP aved registers FP temporaries |
| Branches Branches Jump & L Jump & L Jump & L Jump & L Synch = Synch = | Branch # Branch # Branch # Branch & Branch 2 Branch 2 Bra | Control Contro | 10 1.0., 2.0.7, 2.000 refl, refl, range refl, refl, range refl, refl, range refl, refl, range refl, refl, range refl, refl, range D refl, refl, range refl, refl, range R rd, refl, range refl, refl, range CE refl refl, refl, range CE refl, refl, refl, refl, refl, refl, refl, range refl, refl, refl, refl, refl, range Stat refl, refl | Sobe Dobbe 3 Store Quad SI Arithmetic ADD ADD Word 1 ADD Word 1 ADD SP Imm *. ADD SP Imm *. ADD SP Imm *. ADD SP Imm *. Load Immediate Load Immediate Load Immediate Sitts Smite Left Imm Branches Branch=/ Jump & Link 188 Jump & Link 188 Jump & Link 188 Jump & Link 188 System Env. BREAH 18 7 6 0 rd opcode rd opcode | CL CLUE CALL CL CLUE CLUE CLUE CLUE CLUE CLUE CLUE CLUE | Image Image <thimage< th=""> Image <thi< td=""><td>SD 1 m 2 sby shares SG rai', ra2', simm*16 Tri2, rg2, rg2', simm*16 Tri2, rg2, rg4, rs1 ADDW rd, rd, inn YG 200, rd4, rd1 YG 201, rd4 YG 201, rd4</td><td>Laad Store Arithmeti Niegati Ni Sign Inje N Min/Max Compare Categoriz Configura S Swep</td><td>Load Store ADD SUBtrack MULtiply- DIVide SQuare RooT Multiply-SDBtrack geative Multiply-SDBtrack geative Multiply-SDBtrack geative Multiply-SDBtrack geative Store Subtrack geative Store Subtrack geative Store Subtrack geative Store Subtrack Minisum Maximum Compare Float - Compare Float - Compare Float - Compare Float - Compare Float - Compare Float - Station Classify Type Read Rounding Mode Read Floap Swap Status Reg Swap Flags Swap Flags Rounding Mode Imm Subtrack Team Theorem</td><td>THE REAL REAL REAL FOR THE R. R. P.</td><td>L(M,D,Q) S(M,D,Q) ADD.(S(D)Q)</td><td><pre>rd_rugimm rd_rugimm rd_rugimm rd_rugimm rd_rugimm rd_rugrug_ rd_rugrug rd_rugrug rd_rugrug rd_rugrug rd_rugrug rd_rug_rug rd_rug_rug rd_rug_rug rd_rug_rug rd_rug_rug rd_rug_rug rd_rug_rug rd_rug_rug rd_rug_rug rd_rug rd_rug rd_rug rd_rug rd rug rug rug rug rug rug rug rug rug rug</pre></td><th>Register 23 20 23 23 23 25 7 25 25 25 25 25 25 25 25 25 25 25 25 25</th><td>ABI Name 26250 73 60 60 40-4 40-4 40-4 40-4 40-4 40-4 40-</td><td>Caller Caller</td><td>g Convention Description Hard-wired zero Return address Stack pointer Global pointer Thread pointer Thread pointer Saved register/frame pointer Saved register/frame pointer Function arguments/return value Function arguments FP saved registers FP arguments/return values FP asved registers FP asved registers FP acuments FP temporaries FP temporaries</td></thi<></thimage<> | SD 1 m 2 sby shares SG rai', ra2', simm*16 Tri2, rg2, rg2', simm*16 Tri2, rg2, rg4, rs1 ADDW rd, rd, inn YG 200, rd4, rd1 YG 201, rd4 | Laad Store Arithmeti Niegati Ni Sign Inje N Min/Max Compare Categoriz Configura S Swep | Load Store ADD SUBtrack MULtiply- DIVide SQuare RooT Multiply-SDBtrack geative Multiply-SDBtrack geative Multiply-SDBtrack geative Multiply-SDBtrack geative Store Subtrack geative Store Subtrack geative Store Subtrack geative Store Subtrack Minisum Maximum Compare Float - Compare Float - Compare Float - Compare Float - Compare Float - Compare Float - Station Classify Type Read Rounding Mode Read Floap Swap Status Reg Swap Flags Swap Flags Rounding Mode Imm Subtrack Team Theorem | THE REAL REAL REAL FOR THE R. R. P. | L(M,D,Q) S(M,D,Q) ADD.(S(D)Q) | <pre>rd_rugimm rd_rugimm rd_rugimm rd_rugimm rd_rugimm rd_rugrug_ rd_rugrug rd_rugrug rd_rugrug rd_rugrug rd_rugrug rd_rug_rug rd_rug_rug rd_rug_rug rd_rug_rug rd_rug_rug rd_rug_rug rd_rug_rug rd_rug_rug rd_rug_rug rd_rug rd_rug rd_rug rd_rug rd rug rug rug rug rug rug rug rug rug rug</pre> | Register 23 20 23 23 23 25 7 25 25 25 25 25 25 25 25 25 25 25 25 25 | ABI Name 26250 73 60 60 40-4 40-4 40-4 40-4 40-4 40-4 40- | Caller | g Convention Description Hard-wired zero Return address Stack pointer Global pointer Thread pointer Thread pointer Saved register/frame pointer Saved register/frame pointer Function arguments/return value Function arguments FP saved registers FP arguments/return values FP asved registers FP asved registers FP acuments FP temporaries FP temporaries |
| Branches Branches Jump & Branch Jump & L Synch S Synch S Synch S Synch S Synch S Synch S Synch S S Synch S S Synch S S S S S S S S S S S S S S S S S S S | S Branch = Branch = Branch = Branch = Branch = Branch = Branch ≥ Branch ≥ H A Unsigned h ≥ Unsig | 3 | NO LOA, FUI, FAMI TES1, FES2, imm. TES1, FES2, imm. TES1, FES2, imm. TES1, FES2, imm. U FE1, FES2, imm. U FE1, FES2, imm. CS, FE R rds, FES1, imm. CS, FE CS, FE CS, FE CLAR CS, FE TAL CS, FE TAL CS CLAR CS TAL TAL <td>Store Datalies 3 Store Quad SI Arithmetic ADD ADD Word I ADD Word I ADD SP Imm * 4 ADD SP Imm * 4 Load Immediat Load Immediat Load Immediat Load Immediat Shifts Shift Left Imm Branches Branch=C Branches Branches Branch=C Branches Branches Branches</td> <td>C C C.25 C C C C.25 C C C C C C C C C C C C C C C C C C C</td> <td>xe1 , xe2 , xen xe1 , xe2 , xen mplress mpress mrd, xe1 mrd, xe1 mrd, xe1 mrd, xe1 xd, xe1 xd, xe1 xe1, xe1 xe1, xe1 xe1, xe1 xe1, xe1 xe1, xe1 xe1, xe1 xen xe1, xe1 xen xe1, xe1 xen xen<</td> <td>SD 1 Hoz y By J Hume's SG 1 rol', rol', rimm'16 PC2, sp.jmm*16 PC2, sp.jmm*16 ADDI rd', rd', rdim Pg ap, jmm*16 ADDI rd', rd', rim ADDI rd', sp.jmm*4 ADDI rd', sp.jmm*4 ADDI rd', sp.jmm*4 ADDI rd', rs, jmm*4 ADDI rd', rs, jmm*4 SUB rd', rd', rsn BBE rcl', x0, jmm JAL rd', rd', rsn JAL rd', rsn, jmm JAL ra, imm JAL ra, imm</td> <td>Load Store Arithmeti Negati N Sign Inje N Min/Max Compare Categoriz Configura Sign Inje</td> <td>Load Store ADD SUBtrack Milliply-SDUBtrack SQuare RooT Multiply-SDUBtrack Williply-SDUBtrack very Multiply-SDUBtrack very SUBtrack Substract Substract SQUARE SUBtrack Source Substract Substract Compare Float = Compare Float = Swap Status Reg Swap Status</td> <td>TANK AND A JUNIOR AND AND AND AND AND AND AND AND AND AND</td> <td>L(R, D, Q) S(R, D, Q) XD0, (S) D(Q) XD0, (S) D(Q) XD0, (S) D(Q) XD0, (S) D(Q) XD0, (S) D(Q) D(Q, Q) D(Q, Q) D(Q, Q) D(Q, Q) D(Q, Q) S(R, Q) S(R), (S) D(Q) S(R), (S) D(Q) S(R), (S) D(Q) S(R), (S) D(Q) S(R), (S) D(Q) D(Q, Q) S(R), (S) D(Q) S(R), (S) D(Q) D(Q, Q) S(R), (S) D(Q) S(R), (S) D(Q) S(R),</td> <td>rd_rp3_tam rd_rp3</td> <th>Register 23 22 23 25-7 25 25-7 25-7 25-7 25-27 25-27 25-27 25-27 25-27 25-27 25-27 25-23 25-23 25-23 25-23 25-23</th> <td>ABI Name 2639 Fa 50 60 60 60 60 60 60 60 60 60 60 60 60 60</td> <td>Saver Saver Caller</td> <td>g Convention Description Hard wired zero Return address Stack pointer Global pointer Thread pointer Thread pointer Tamporaries Saved register/frame pointer Saved registers Function arguments Saved registers FP temporaries FP arguments/return values FP arguments FP arguments FP arguments FP saved registers FP arguments FP arguments FP temporaries FP</td> | Store Datalies 3 Store Quad SI Arithmetic ADD ADD Word I ADD Word I ADD SP Imm * 4 ADD SP Imm * 4 Load Immediat Load Immediat Load Immediat Load Immediat Shifts Shift Left Imm Branches Branch=C Branches Branches Branch=C Branches Branches | C C C.25 C C C C.25 C C C C C C C C C C C C C C C C C C C | xe1 , xe2 , xen xe1 , xe2 , xen mplress mpress mrd, xe1 mrd, xe1 mrd, xe1 mrd, xe1 xd, xe1 xd, xe1 xe1, xe1 xe1, xe1 xe1, xe1 xe1, xe1 xe1, xe1 xe1, xe1 xen xe1, xe1 xen xe1, xe1 xen xen< | SD 1 Hoz y By J Hume's SG 1 rol', rol', rimm'16 PC2, sp.jmm*16 PC2, sp.jmm*16 ADDI rd', rd', rdim Pg ap, jmm*16 ADDI rd', rd', rim ADDI rd', sp.jmm*4 ADDI rd', sp.jmm*4 ADDI rd', sp.jmm*4 ADDI rd', rs, jmm*4 ADDI rd', rs, jmm*4 SUB rd', rd', rsn BBE rcl', x0, jmm JAL rd', rd', rsn JAL rd', rsn, jmm JAL ra, imm | Load Store Arithmeti Negati N Sign Inje N Min/Max Compare Categoriz Configura Sign Inje | Load Store ADD SUBtrack Milliply-SDUBtrack SQuare RooT Multiply-SDUBtrack Williply-SDUBtrack very Multiply-SDUBtrack very SUBtrack Substract Substract SQUARE SUBtrack Source Substract Substract Compare Float = Compare Float = Swap Status Reg Swap Status | TANK AND A JUNIOR AND | L(R, D, Q) S(R, D, Q) XD0, (S) D(Q) XD0, (S) D(Q) XD0, (S) D(Q) XD0, (S) D(Q) XD0, (S) D(Q) D(Q, Q) D(Q, Q) D(Q, Q) D(Q, Q) D(Q, Q) S(R, Q) S(R), (S) D(Q) S(R), (S) D(Q) S(R), (S) D(Q) S(R), (S) D(Q) S(R), (S) D(Q) D(Q, Q) S(R), (S) D(Q) S(R), (S) D(Q) D(Q, Q) S(R), (S) D(Q) S(R), | rd_rp3_tam rd_rp3 | Register 23 22 23 25-7 25 25-7 25-7 25-7 25-27 25-27 25-27 25-27 25-27 25-27 25-27 25-23 25-23 25-23 25-23 25-23 | ABI Name 2639 Fa 50 60 60 60 60 60 60 60 60 60 60 60 60 60 | Saver Saver Caller | g Convention Description Hard wired zero Return address Stack pointer Global pointer Thread pointer Thread pointer Tamporaries Saved register/frame pointer Saved registers Function arguments Saved registers FP temporaries FP arguments/return values FP arguments FP arguments FP arguments FP saved registers FP arguments FP arguments FP temporaries FP |

RISC-V Integer Base (RV321/641/1281), privileged, and optional compressed extension (RVC). Registers x1-x31 and the pc are 32 bits wide in RV321, 64 in RV641, and 128 in RV1281 (x0=0). RV641/1281 add 10 instructions for the wider formats. The RVI base of <50 classic integer RISC instructions is required. Every 16-bit RVC instruction matches an existing 32-bit RVI instruction. See risc.org. RMSC-V calling convention and two optional extensions: 10 multiply-drivide instructions (RV32M); 10 optional atomic instructions (RV32A); and 25 floating-point instructions each for single-, double-, and quadruple-precision (RV32E, RV32D, RV32D, RV32D). width matches the widdest precision, and a floating-point control and status register fast. Each larger address adds some instructions: 4 for RVM, 11 for RV3, and 6 each for RVFIDiQ. Using reger notation, (1) means set, so L(10)(0) is both LD and LQ. See risc.org, (821/13 revision)

ETHZürich

RISC-V Architectural State

- There are 32 registers, each 32 / 64 / 128 bits long
 - Named x0 to x31
 - x0 is hard wired to zero
 - There is a standard 'E' extension that uses only 16 registers (RV32E)
- In addition one program counter (PC)
 - Byte based addressing, program counter increments by 4/8/16
 - For floating point operation 32 additional FP registers
- Additional Control Status Registers (CSRs)
 - Encoding for up to 4'096 registers are reserved. Not all are used.



RISC-V Instructions four basic types

- R register to register operations
- operations with immediate/constant values
- S / SB operations with two source registers
- U / UJ operations with large immediate/constant value

| | 31 25 | 24 20 | 19 15 | 14 1: | 11 | 6 0 | |
|---|----------------------------|-----------------------------|-------|--------|---------------------------|--------|--------|
| | funct7 | rs2 | rs1 | funct3 | rd | opcode | R-type |
| | | | | | | | |
| | $\operatorname{imm}[11:$ |)] | rs1 | funct3 | rd | opcode | I-type |
| - | | | | | | | |
| | $\operatorname{imm}[11:5]$ | rs2 | rs1 | funct3 | $\operatorname{imm}[4:0]$ | opcode | S-type |
| | | | | | | | |
| | | $\operatorname{imm}[31:12]$ | | | rd | opcode | U-type |
| / | | | | | | | • |

RISC-V is a load/store architecture

- All operations are on internal registers
 - Can not manipulate data in memory directly
- Load instructions to copy from memory to registers
- R-type or I-type instructions to operate on them
- Store instructions to copy from registers back to memory
- Branch and Jump instructions
- 1/3 ALU utilization if operands are from/to memory (LD, ALU, ST)



ETH zürich

Encoding of the instructions, main groups

- Reserved opcodes for standard extensions
- Rest of opcodes free for custom implementations
- Standard extensions will be frozen/not change in the future

| inst[4:2] | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
|-----------|--------|----------|----------|----------|--------|----------|----------------|------------|
| inst[6:5] | | | | | | | | (> 32b) |
| 00 | LOAD | LOAD-FP | custom-0 | MISC-MEM | OP-IMM | AUIPC | OP-IMM-32 | 48b |
| 01 | STORE | STORE-FP | custom-1 | AMO | OP | LUI | OP-32 | 64b |
| 10 | MADD | MSUB | NMSUB | NMADD | OP-FP | reserved | custom-2/rv128 | 48b |
| 11 | BRANCH | JALR | reserved | JAL | SYSTEM | reserved | custom-3/rv128 | $\geq 80b$ |

Extensibility is integral to RISC-V ISA design!

How to get efficiency: ISA extensions



M. Gautschi et al., "Near-Threshold RISC-V Core With DSP Extensions for Scalable IoT Endpoint Devices," in IEEE TVLSI, Oct. 2017.

Post increment LD/ST

Automatic address update

- Update base register with computed address after the memory access
- ⇒Save instructions to update address register
- Post-increment:
 - Base address serves as memory address

Offset can be stored in:

- Register
- Immediate

c = 0; for(i=0;i<100;i++) c = c + a[i]*b[i];

Original RISCV Auto-incr load/store

addi x4, x0, 64 Lstart : *Ib x2, 0(x10) Ib x3, 0(x12) addi x10, x10, 1 addi x12, x12, 1* bne x2,x3, Lstart addi x4, x0, 64 Lstart : *Ib x2, 0(x10!) Ib x3, 0(x12!)* bne x2,x3, Lstart

⇒save 2 additional instructions to update the read addresses of the operands!

Hardware loops

Hardware loop setup with:

- 3 separate instructions
 - Ip.start, Ip.end, Ip.count, Ip.counti
 - \Rightarrow No restriction on start/end address

Fast setup instructions

lp.setup, lp.setupi

- \Rightarrow Start address= PC + 4
- \Rightarrow End address= start address + offset
- \Rightarrow Counter from immediate/register

Original RISC-V

//initialize counter mv x4, 100 // init accumulator mv x5,0 Lstart: //decrement counter addi x4. x4. -1 //load elements from mem lw x8, 0(x9) lw x10, 0(x11) //update memory pointers add x9, x9, 4 add x11, x11, 4 //mac mul x8, x8, x10 add x5, x5, x8 bne x4, x0, Lstart

c = 0; for(i=0;i<100;i++) c = c + a[i]*b[i];

HW Loop Ext

// init accumulator
mv x5, 0
//set number iterations, start and end of the loop
Ip.setupi 100, Lend
//load elements from mem
lw x8, 0(x9)
lw x10, 0(x11)
//update memory pointers
add x9, x9, 4
add x11, x11, 4
//mac
mul x8, x8, x10
Lend: add x5, x5, x8

No counter and branch overhead!

ETHZUrich



Accumulation on 32 bit data p.mac



Intrinsics: special functions that map directly to inlined DSP instructions.

However, the compiler can already place the p.mac instruction into the above code!



- Directly on the register file
- Pro:
 - Faster access to mac accumulation
 - Single cycle mult/mac
- Cons:
 - Additional read port on the register file
 - used for pre/post increment with register

Xpulp Extentions: packed-SIMD

Remember: DNN inference is OK with low-bitdwidth operands

packed-SIMD extensions

- Make usage of resources the best in performance with little overhead
- Target for embedded systems, RVV is for high performance
- pSIMD in 32bit machines
 - Vectors are either 4 8bits-elements or 2 16bits-elements
- pSIMD instructions

| Computation | add, sub, shift, avg, abs, dot product |
|-------------|--|
| Compare | min, max, compare |
| Manipulate | extract, pack, shuffle |

Xpulp Extensions: packed-SIMD

Same Register-file

• The instruction encode how to interpret the content of the register

| rs1 | 0x03 | 0x02 | 0x01 | 0x00 |
|-----|------|------|------|------|
| rs2 | 0x0D | 0x0C | 0x0B | 0x0A |

| add rD, rs1, rs2 | rD = 0x03020100 + 0x0D0C0B0A |
|--------------------|--|
| add.h rD, rs1, rs2 | rD[0] = 0x0100 + 0x0B0A rD[1] = 0x0302 + 0x0D0C |
| add.b rD, rs1, rs2 | rD[0] = 0x00 + 0x0A rD[1] = 0x01 + 0x0B rD[2] = 0x02 + 0x0C rD[3] = 0x03 + 0x0D |

ALU architecture

- Advanced ALU for Xpulp extensions
- Optimized datapath to reduce resources
- Multiple-adders for round
- Adder followed by shifter for fixed point normalization
 - Clip unit uses one adder as comparator and the main comparator



Expanding SIMD Dot Product



ETH zürich

24

MUL architecture



16x16b with sign selection for short multiplications [with round and normalization]. 5 cycles FSM for higher 64bits (mulh* instructions)

32x32b single cycle MAC/MUL unit

16x16b short parallel dot product

8x8b byte parallel dot product

clock gating to reduce switching activity between the scalar and SIMD multipliers

Reference & Examples on Compiler Builtins

SIMD Instructions of the Xpulp ISA extension

| Computation | add, sub, shift, avg, abs, dot product |
|-------------|--|
| Compare | min, max, compare |
| Manipulate | extract, pack, shuffle |

Dot-product without accumulation between <u>unsigned</u> char vectors (v4u):

S = __builtin_dotup4(A, B); // S = A[0]*B[0] + A[1]*B[1] + A[2]*B[2] + A[3]*B[3], A is v4u, B is v4u

Dot-product without accumulation between signed char vectors (v4s):

S = __builtin_dotsp4(A, B); // S = A[0]*B[0] + A[1]*B[1] + A[2]*B[2] + A[3]*B[3] , A is v4s, B is v4s Also with mixed signs:

S = __builtin_dotusp4(A, B); // S = A[0]*B[0] + A[1]*B[1] + A[2]*B[2] + A[3]*B[3] , A is v4u, B is v4s

Similar builtins without accumulation for short vectors:

S = __builtin_dotup2(A, B); S = __builtin_dotsp2(A, B); S = __builtin_dotusp2(A, B);

All of these are also available with accumulation (over accumulator S):

S = __builtin_sdotup4(A, B, S); S = __builtin_sdotsp4(A, B, S); S = __builtin_sdotusp4(A, B, S);

S = __builtin_sdotup2(A, B, S); S = __builtin_sdotsp2(A, B, S); S = __builtin_sdotusp2(A, B, S);



- The innermost loop has 4x less iterations
 - 4 bytes per matrix are loaded as a 32b word
 - Dot product with accumulation performs in 1 cycle 4 macs

. . . .

| | //iterate #COL/4 |
|------------------------------------|--|
| lp.setup x1,a4, <mark>stop1</mark> | lp.setup x1,a6, <mark>stop1</mark> |
| p.lbu a0,1(a3!) | p.lw a1,4(t1!) //load 4-bytes with post inc |
| p.lbu a1,32(a2!) | p.lw a5,4(t3!) |
| stop1: p.mac a5,a0,a1 | <pre>stop1: pv.sdotsp.b a7,a1,a5 //4 mac</pre> |

. . . .

Extensions at work

for (i = 0; i < 100; i++) d[i] = a[i] + b[i];

| addi x4, x4, -1 <i>addi x12,x12, 1</i> bne x4, x5, Lstart | Lstart | | |
|--|--|---|--|
| add x2, x3, x2 sb x2, 0(x12) | <i>sb x2</i> , <i>0(x12!)</i> bne <i>x4</i> , <i>x5</i> , | | |
| addi x10,x10, 1 | addi x4, x4, -1 add x2, x3, x2 | 0(x12!) | Lend: sw x2, 0(x12!) |
| $\begin{array}{cccc} 1b & x2, & 0(x10) \\ 1b & x3, & 0(x11) \end{array}$ | $\begin{array}{cccc} 1b & x2, & 0(x10!) \\ 1b & x3, & 0(x11!) \end{array}$ | add x2, x3, x2 | pv.add.b x2, x3, |
| mv x5, 0 mv x4, 100 Lstart: | mv x5, 0 mv x4, 100 Lstart: | lp.setupi 100, Lend lb x2, 0(x10!) lb x3, 0(x11!) | <i>lp.setupi</i> 25 , <i>Lend</i> lw x2, 0(x10!) lw x3, 0(x11!) |
| | | | |

ETH ZUrich

Advanced SIMD: Shuffle Instruction

- In order to use the vector unit the elements have to be aligned in the register file
- Shuffle allows to recombine bytes into 1 register: Mask bits
- pv.shuffle2.b rD, rA, rB

rD{3} = (rB[26]==0) ? rA:rD {rB[25:24]} rD{2} = (rB[18]==0) ? rA:rD {rB[17:16]} rD{1} = (rB[10]==0) ? rA:rD {rB[9: 8]} rD{0} = (rB[2]==0) ? rA:rD {rB[1: 0]}

With rX{i} = rX[(i+1)*8-1:i*8]



Shuffle for Direct SIMD Convolution



ETH zürich

Convolution in registers 5x5 convolutional filter

- 7 Sum-of-dot-product
- 4 move
- 1 shuffle
- 3 lw/sw
- ~ 5 control instructions

Significant benefit in reuse of registers and less LD/ST

30



<u>8-bit</u> Convolution example



Never underestimate the importance of registers! How to get "more"?

Achieving 100% dotp Unit Utilization



Hardware for dotp+ld



ETHZUrich

Not only RISC-V: Armv8.1-M

New embedded vector ISA Helium (MVE)

TH züric

- Uses 8 128-bit vector registers (reuses the 32 FP registers)
- ISA enhancements for loops, branches (Low Overhead Branch Extension)
- Instructions for half-precision floating-point support
- Enhancements in debug including performance monitoring unit (PMU) and additional debug support to focus on signal processing application developments.
 - Being able to set a breakpoint which triggers (halts code execution and passes control to the debugger) when a certain count value is reached and being able to set a data watchpoint with a bit mask for data value comparison (for example, for looking for a signal value to be within a certain range).





ETHZÜric

ARM MVE Vectors

- Helium provides a SIMD capability for Cortex-M CPUs: a set of 128-bit registers are provided which can be used to hold, e.g. 16 separate 8-bit values. A single instruction can operate on each value independently (with predication)
- Extension of Arm Thumb
- Helium instructions operate on vectors of elements of the same data type: Int/FP
 - Integer elements may be signed or unsigned 8-, 16-, 32-bit, fixed-point saturating (Q7, Q15, Q31)
 - Floating-point elements may be single (32-bit) or half precision (16-bit).
 - The position of an element in a vector is called *lane*



ARM MVE Vector Execution Model

- MVE permits instruction execution to be interleaved
 - Multiple instructions may overlap in the pipeline execute stage. For example, a Vector Load (VLDR) instruction which reads multiple words from memory into a vector register may execute at the same time as a Vector Multiply (VMUL) instruction which uses that data
- It is up to the CPU hardware designer to decide how many "beats" are executed on each clock cycle (eg. 32-bit datapath vs 64-bit datapath)



Complicates exception handling

D of the VLDR happens after beat A of the VMLA has completed. If memory for beat D triggers a fault, the processor needs to remember that the following instruction was part executed (storing a value which shows which beats have already been executed). If fter exception handling, the program returns to this location, the hardware already knows which beats should not be re-executed

Cortex-M55 Performance

- Performance relative to Cortex-M4
- Major improvements for Q7, FP16 (new datatypes in HW)
- ML benchmark (KWS): MFCC, DNN (2 conv, 3 FC layers), 8-bit (w, act), 80-500KB, accuracy 90%-95%









SoA Quantization Results

üric



Quantizazion of a MobilenetV1_224_1.0 (*)

| Quantization Method | Top1 Accuracy | | | Weight Memory Footprint | | |
|------------------------|---------------|--|------|----------------------------|------------|----|
| Full-Precision | 70.9% | | | 16.27 MB | | |
| INT-8 | 70.1% | | 0.8% | 4.06 MB | 4 x | |
| INT-4 | 66.46% | | 4.4% | 2.35 MB | | 7x |
| Mixed-Precision | 68% | | 2.9% | 2.09 MB 8x | | |

Courtesy of Rusci M. «Example on MobilenetV1_224_1.0.»C

Mixed-precision approach key to meet the memory constraints of tiny devices

Quantized Neural Networks (QNNs) are a natural target for execution on constrained extreme edge platforms.

(*) Rusci M. et al., Memory-Driven Mixed Low Precision Quantization For Enabling Deep Network Inference On Microcontrollers. arXiv preprint arXiv:1905.13082.

Sub-byte operands manipulation

32-bit data load with post increment (one cycle)



Digital computing platforms for near-sensor processing at the extreme edge of the IoT

Mixed Precision SIMD Processor



How to encode all these instructions?

Mixed-Precision Core: New Formats Required

| pv.dotsp.h | pv.dotup.h | pv.dotusp.h | pv.sdotsp .h | pv.sdotup.h | pv.sdotusp.h | dotr | varia | nte |
|------------------------|------------------------|--------------------------|------------------------|------------------------|--------------------------|----------------|-------|--------------------|
| pv.dotsp.b | pv.dotup.b | pv.dotusp.b | pv.sdotsp.b | pv.sdotup.b | pv.sdotusp.b | uoip | | 11.5 |
| pv.dotsp.n | pv.dotup.n | pv.dotusp.n | pv.sdotsp.n | pv.sdotup.n | pv.sdotusp.n | | | |
| pv.dotsp.c | pv.dotup.c | pv.dotusp.c | pv.sdotsp.c | pv.sdotup.c | pv.sdotusp.c | | v hhc | ariante |
| pv.dotsp.m4x2 | pv.dotup.m4x2 | pv.dotusp.m4x2 | pv.sdotsp.m4x2 | pv.sdotup.m4x2 | pv.sdotusp.m4x2 | | auu v | anants |
| pv.dotsp.m8x2 | pv.dotup.m8x2 | pv.dotusp.m8x2 | pv.sdotsp.m8x2 | pv.sdotup.m8x2 | pv.sdotusp.m8x2 | | | |
| pv.dotsp.m8x4 | pv.dotup.m8x4 | pv.dotusp.m8x4 | pv.sdotsp.m8x4 | pv.sdotup.m8x4 | pv.sdotusp.m8x4 | | | ub voriente |
| pv.dotsp.m16x8 | pv.dotup.m16x8 | pv.dotusp.m16x8 | pv.sdotsp.m16x8 | pv.sdotup.m16x8 | pv.sdotusp.m16x8 | | 5 | sud variants |
| pv.dotsp.m16x4 | pv.dotup.m16x4 | pv.dotusp.m16x4 | pv.sdotsp.m16x4 | pv.sdotup.m16x4 | pv.sdotusp.m16x4 | | | |
| pv.dotsp.m16x2 | pv.dotup.m16x2 | pv.dotusp.m16x2 | pv.sdotsp.m16x2 | pv.sdotup.m16x2 | pv.sdotusp.m16x2 | | 6 | avg variants |
| pv.dotsp.sc.h | pv.dotup.sc .h | pv.dotusp.sc .h | pv.sdotsp.sc .h | pv.sdotup.sc .h | pv.sdotusp.sc.h | 8 2 | | |
| pv.dotsp.sc.b | pv.dotup.sc.b | pv.dotusp.sc.b | pv.sdotsp.sc.b | pv.sdotup.sc.b | pv.sdotusp.sc.b | 4 2 | S | shift variants |
| pv.dotsp.sc.c | pv.dotup.sc.c | pv.dotusp.sc.c | pv.sdotsp.sc.c | pv.sdotup.sc.c | pv.sdotusp.sc.c | 2 4 | - | |
| pv.dotsp.sc.n | pv.dotup.sc.n | pv.dotusp.sc.n | pv.sdotsp.sc.n | pv.sdotup.sc.n | pv.sdotusp.sc.n | ×8 | r | nax variants |
| pv.dotsp.sc.m4x2 | pv.dotup.sc.m4x2 | pv.dotusp.sc.m4x2 | pv.sdotsp.sc.m4x2 | pv.sdotup.sc.m4x2 | pv.sdotusp.sc.m4x2 | ×4 | ļ. | |
| pv.dotsp.sc.m8x2 | pv.dotup.sc.m8x2 | pv.dotusp.sc.m8x2 | pv.sdotsp.sc.m8x2 | pv.sdotup.sc.m8x2 | pv.sdotusp.sc.m8x2 | ×2 | r | nin variante |
| pv.dotsp.sc.m8x4 | pv.dotup.sc.m8x4 | pv.dotusp.sc.m8x4 | pv.sdotsp.sc.m8x4 | pv.sdotup.sc.m8x4 | pv.sdotusp.sc.m8x4 | | | |
| pv.dotsp.sc.m16x8 | pv.dotup.sc.m16x8 | pv.dotusp.sc.m16x8 | pv.sdotsp.sc.m16x8 | pv.sdotup.sc.m16x8 | pv.sdotusp.sc.m16x8 | x2 | | ha varianta |
| pv.dotsp.sc.m16x4 | pv.dotup.sc.m16x4 | pv.dotusp.sc.m16x4 | pv.sdotsp.sc.m16x4 | pv.sdotup.sc.m16x4 | pv.sdotusp.sc.m16x4 | x2 | Ċ | |
| pv.dotsp.sc.m16x2 | pv.dotup.sc.m16x2 | pv.dotusp.sc.m16x2 | pv.sdotsp.sc.m16x2 | pv.sdotup.sc.m16x2 | pv.sdotusp.sc.m16x2 | x4 | | |
| pv.dotsp.sci .h | pv.dotup.sci .h | pv.dotusp.sci.h | pv.sdotsp.sci.h | pv.sdotup.sci.h | pv.sdotusp.sci.n | 6x8 (4) | (2 | |
| pv.dotsp.sci.b | pv.dotup.sci.b | pv.dotusp.sci.b | pv.sdotsp.sci.b | pv.sdotup.sci.b | pv.sdotusp.sci.b | 6x4 (8) | (2 | |
| pv.dotsp.sci.c | pv.dotup.sci.c | pv.dotusp.sci.c | pv.sdotsp.sci.c | pv.sdotup.sci.c | pv.sdotusp.sci.c | 6x2 (8) | (4 | |
| pv.dotsp.sci.n | pv.dotup.sci.n | pv.dotusp.sci.n | pv.sdotsp.sci.n | pv.sdotup.sci.n | pv.sdotusp.sci.n | 10 | 5x8 | |
| pv.dotsp.sci.m4x2 | pv.dotup.sci.m4x2 | pv.dotusp.sci.m4x2 | pv.sdotsp.sci.m4x2 | pv.sdotup.sci.m4x2 | pv.sdotusp.sci.m4x2 | 10 | 5x4 | > 500 instructions |
| pv.dotsp.sci.m8x2 | pv.dotup.sci.m8x2 | pv.dotusp.sci.m8x2 | pv.sdotsp.sci.m8x2 | pv.sdotup.sci.m8x2 | pv.sdotusp.sci.m8x2 | 10 | 5x2 | |
| pv.dotsp.sci.m8x4 | pv.dotup.sci.m8x4 | pv.dotusp.sci.m8x4 | pv.sdotsp.sci.m8x4 | pv.sdotup.sci.m8x4 | pv.sdotusp.sci.m8x4 | 1 | | |
| pv.dotsp.sci.m16x8 | pv.dotup.sci.m16x8 | pv.dotusp.sci.m16x8 | pv.sdotsp.sci.m16x8 | pv.sdotup.sci.m16x8 | pv.sdotusp.sci.m16x6 | 4x2) | | |
| pv.dotsp.sci.m16x4 | pv.dotup.sci.m16x4 | pv.dotusp.sci.m16x4 | pv.sdotsp.sci.m16x4 | pv.sdotup.sci.m16x4 | py sdotusp sci m16x2 | 8x2 | | |
| pv.dotsp.sci.m16x2 | pv.dotup.sci.m16x2 | pv.dotusp.sci.m16x2 | pv.sdotsp.sci.m16x2 | pv.suotup.sci.m16x2 | pv.suotusp.sci.m10x2 | Bx4 1 | | |
| pv.dotsp.sci.m | 16x8 pv.dotup.sci.m | n16x8 pv.dotusp.sci.m | 16x8 pv.sdotsp.sci | i.m16x8 pv.sdotup.so | ci.m16x8 pv.sdotusp.sci. | .m16x8 n4 | x2 | |
| pv.dotsp.sci.m | 16x4 pv.dotup.sci.m | n16x4 pv.dotusp.sci.m | 16x4 pv.sdotsp.sci | i.m16x4 pv.sdotup.so | ci.m16x4 pv.sdotusp.sci. | .m16x4 n8 | x2 | |
| pv.dotsp.sci.m | 16x2 pv.dotup.sci.m | n16x2 pv.dotusp.sci.m | 16x2 pv.sdotsp.sci | i.m16x2 pv.sdotup.so | ci.m16x2 pv.sdotusp.sci. | .m16x2 n8 | x4 | |
| | uotspisci.mitoxo pv | .uotup.sci.iii.toxo pris | р | | | . n1 | 6x8 | |
| pv. | dotsp.sci.m16x4 pv | .dotup.sci.m16x4 pv.o | dotusp.sci.m16x4 p | v.sdotsp.sci.m16x4 | pv.sdotup.sci.m16x4 pv. | sdotusp.sci.m1 | 6x4 | |
| | doten sci m16v2 nv | dotup sci m16v2 pv (| lotuspisci m16x2 p | v sdotspisci m16x2 – i | pv.sdotup.sci.m16x2 pv.: | sdotusp.sci.m1 | 6x2 | |

Virtual SIMD Instructions

- Encode operation as a virtual SIMD in the ISA (e.g. sdotsp.v)
- Format specified at runtime by a Control Register (e.g. 4x4)
- 180→18 Instructions needed for SIMD DOTP
 - Potential to avoid code replication for different formats
 - Tiny Overhead on QNN for Switching format

Format switch not frequent in DNN, e.g. every layer.

| pv.dotsp.h | pv.dotup.h | pv.dotusp.h | pv.sdotsp.h | pv.sdotup.h | pv.sdotusp.h |
|--------------------|--------------------|-------------------------|---------------------|-------------------------|--------------------------|
| pv.dotsp.b | pv.dotup.b | pv.dotusp.b | pv.sdotsp.b | pv.sdotup.b | pv.sdotusp.b |
| pv.dotsp.n | pv.dotup.n | pv.dotusp.n | pv.sdotsp.n | pv.sdotup.n | pv.sdotusp.n |
| pv.dotsp.c | pv.dotup.c | pv.dotusp.c | pv.sdotsp.c | pv.sdotup.c | pv.sdotusp.c |
| pv.dotsp.m4x2 | pv.dotup.m4x2 | pv.dotusp.m4x2 | pv.sdotsp.m4x2 | pv.sdotup.m4x2 | pv.sdotusp.m4x2 |
| pv.dotsp.m8x2 | pv.dotup.m8x2 | pv.dotusp.m8x2 | pv.sdotsp.m8x2 | pv.sdotup.m8x2 | pv.sdotusp.m8x2 |
| pv.dotsp.m8x4 | pv.dotup.m8x4 | pv.dotusp.m8x4 | pv.sdotsp.m8x4 | pv.sdotup.m8x4 | pv.sdotusp.m8x4 |
| pv.dotsp.m16x8 | pv.dotup.m16x8 | pv.dotusp.m16x8 | pv.sdotsp.m16x8 | pv.sdotup.m16x8 | pv.sdotusp.m16x8 |
| pv.dotsp.m16x4 | pv.dotup.m16x4 | pv.dotusp.m16x4 | pv.sdotsp.m16x4 | pv.sdotup.m16x4 | pv.sdotusp.m16x4 |
| pv.dotsp.m16x2 | pv.dotup.m16x2 | pv.dotusp.m16x2 | pv.sdotsp.m16x2 | pv.sdotup.m16x2 | pv.sdotusp.m16x2 |
| pv.dotsp.sc.h | pv.dotup.sc.h | pv.dotusp.sc.h | pv.sdotsp.sc.h | pv.sdotup.sc .h | pv.sdotusp.sc.h |
| pv.dotsp.sc.b | pv.dotup.sc.b | pv.dotusp.sc.b | pv.sdotsp.sc.b | pv.sdotup.sc.b | pv.sdotusp.sc.b |
| pv.dotsp.sc.c | pv.dotup.sc.c | pv.dotusp.sc.c | pv.sdotsp.sc.c | pv.sdotup.sc.c | pv.sdotusp.sc.c |
| pv.dotsp.sc.n | pv.dotup.sc.n | pv.dotusp.sc.n | pv.sdotsp.sc.n | pv.sdotup.sc.n | pv.sdotusp.sc.n |
| pv.dotsp.sc.m4x2 | pv.dotup.sc.m4x2 | pv.dotusp.sc.m4x2 | pv.sdotsp.sc.m4x2 | pv.sdotup.sc.m4x2 | pv.sdotusp.sc.m4x2 |
| pv.dotsp.sc.m8x2 | pv.dotup.sc.m8x2 | pv.dotusp.sc.m8x2 | pv.sdotsp.sc.m8x2 | pv.sdotup.sc.m8x2 | pv.sdotusp.sc.m8x2 |
| pv.dotsp.sc.m8x4 | pv.dotup.sc.m8x4 | pv.dotusp.sc.m8x4 | pv.sdotsp.sc.m8x4 | pv.sdotup.sc.m8x4 | pv.sdotusp.sc.m8x4 |
| pv.dotsp.sc.m16x8 | pv.dotup.sc.m16x8 | pv.dotusp.sc.m16x8 | pv.sdotsp.sc.m16x8 | pv.sdotup.sc.m16x8 | pv.sdotusp.sc.m16x8 |
| pv.dotsp.sc.m16x4 | pv.dotup.sc.m16x4 | pv.dotusp.sc.m16x4 | pv.sdotsp.sc.m16x4 | pv.sdotup.sc.m16x4 | pv.sdotusp.sc.m16x4 |
| pv.dotsp.sc.m16x2 | pv.dotup.sc.m16x2 | pv.dotusp.sc.m16x2 | pv.sdotsp.sc.m16x2 | pv.sdotup.sc.m16x2 | pv.sdotusp.sc.m16x2 |
| pv.dotsp.sci.h | pv.dotup.sci.h | pv.dotusp.sci .h | pv.sdotsp.sci.h | pv.sdotup.sci .h | pv.sdotusp.sci .h |
| pv.dotsp.sci.b | pv.dotup.sci.b | pv.dotusp.sci.b | pv.sdotsp.sci.b | pv.sdotup.sci.b | pv.sdotusp.sci.b |
| pv.dotsp.sci.c | pv.dotup.sci.c | pv.dotusp.sci.c | pv.sdotsp.sci.c | pv.sdotup.sci.c | pv.sdotusp.sci.c |
| pv.dotsp.sci.n | pv.dotup.sci.n | pv.dotusp.sci.n | pv.sdotsp.sci.n | pv.sdotup.sci.n | pv.sdotusp.sci.n |
| pv.dotsp.sci.m4x2 | pv.dotup.sci.m4x2 | pv.dotusp.sci.m4x2 | pv.sdotsp.sci.m4x2 | pv.sdotup.sci.m4x2 | pv.sdotusp.sci.m4x2 |
| pv.dotsp.sci.m8x2 | pv.dotup.sci.m8x2 | pv.dotusp.sci.m8x2 | pv.sdotsp.sci.m8x2 | pv.sdotup.sci.m8x2 | pv.sdotusp.sci.m8x2 |
| pv.dotsp.sci.m8x4 | pv.dotup.sci.m8x4 | pv.dotusp.sci.m8x4 | pv.sdotsp.sci.m8x4 | pv.sdotup.sci.m8x4 | pv.sdotusp.sci.m8x4 |
| pv.dotsp.sci.m16x8 | pv.dotup.sci.m16x8 | pv.dotusp.sci.m16x8 | pv.sdotsp.sci.m16x8 | pv.sdotup.sci.m16x8 | pv.sdotusp.sci.m16x8 |
| pv.dotsp.sci.m16x4 | pv.dotup.sci.m16x4 | pv.dotusp.sci.m16x4 | pv.sdotsp.sci.m16x4 | pv.sdotup.sci.m16x4 | pv.sdotusp.sci.m16x4 |
| pv.dotsp.sci.m16x2 | pv.dotup.sci.m16x2 | pv.dotusp.sci.m16x2 | pv.sdotsp.sci.m16x2 | pv.sdotup.sci.m16x2 | pv.sdotusp.sci.m16x2 |
| | | | | | |

pv.dotsp.v pv.dotsp.sc.v pv.dotsp.sci.v pv.dotup.v pv.dotup.sc.v pv.dotup.sci.v pv.dotusp.v pv.dotusp.sc.v pv.dotusp.sci.v pv.sdotsp.v pv.sdotsp.sc.v pv.sdotup.sc.v pv.sdotup.sc.v pv.sdotup.sc.v pv.sdotusp.v pv.sdotusp.sc.v pv.sdotusp.sc.v





Züric

Processor HW extension



- Goal
 - HW support for mixed-precision SIMD instructions;
- Challenge
 - Enormous number of instructions to be encoded in the ISA;
- Solution
 - Status-based execution.

Extended Dot-Product Unit

Multi-Precision Integer Dotp-Unit





Nice – But what about the GOPS? M7: 5.01 CoreMark/MHz-58.5 μW/MHz **Faster+Superscalar is not efficient!** M4: 3.42 CoreMark/MHz-12.26 μW/MHz

ETH zürich

- As VDD decreases, operating speed decreases
- However efficiency increases \rightarrow more work done per Joule
- Until leakage effects start to dominate
- (parMatrixMul2) [MOPs/mW] Put more units in parallel to get performance up and keep them busy with a Efficiency parallel workload

ML is massively parallel and scales well $(P/S \uparrow with NN size)$



Efficiency vs VDD chip01

Multiple RI5CY Cores (1-16)

RISC-V core Core RISC-V core core core

CLUSTER

Low-Latency Shared TCDM

Tightly Coupled Data Memory Mem Logarithmic Interconnect **RISC-V RISC-V RISC-V RISC-V** core core core core **CLUSTER**





World-level bank interleaving «emulates» multiported mem

A. Rahimi, I. Loi, M. R. Kakoee and L. Benini, "A fully-synthesizable single-cycle interconnection network for Shared-L1 processor clusters," *2011 Design, Automation & Test in Europe*, 2011, pp. 1-6.

Fast synchronization and Atomics



CLUSTER

F. Glaser, G. Tagliavini, D. Rossi, G. Haugou, Q. Huang and L. Benini, "Energy-Efficient Hardware-Accelerated Synchronization for Shared-L1-Memory Multiprocessor Clusters," in IEEE TPDS, vol. 32, no. 3, pp. 633-648, 1 March 2021.

ETHZÜrich



Synchronization & Events external cluster CNN acc. DMA Avoid busy waiting! Minimize sw synchro. overhead control MCU Efficient fine-grain parallelization - event signalling execution synchronization - execution control exclusive resources manag. periphs. core N core 0 core 1 Private, per core port \rightarrow single cycle latency 7-5 \rightarrow no contention ShuCH

Results: Barrier



- Fully parallel access to SCU: Barrier cost constant
- Primitive energy cost: Down by up to 30x
- Minimum parallel section for 10% overhead in terms of ...
 - ... cycles: ~100 instead of > 1000 cycles
 - ... energy: \sim 70 instead of > 2000 cycles

PULP for ML (DNNs) Speedup

[Garofalo et al. Philos. Trans. R. Soc 20]

- 8-bit convolution
 - Open source DNN library
- 10x through xPULP
 - Extensions bring real speedup
- Near-linear speedup
 - Scales well for regular workloads
- 75x overall gain



53

8-Cores Cluster + XpulpNN + M&L (22nm)



Addressing Multicore Inefficiencies

Power analysis of a parallel 8-bit x 4-bit convolution



- Reduce unnecessary power consumption (not spent in computation)
- Exploit convolution's instruction and memory data access pattern regularity
 - Increase energy efficiency at low extra-area cost
 - → reconfigurable MIMD/SIMD architecture

The Power of SIMD



- Cores enter in SIMD (VLEM) mode when executing regular kernels (in two clock cycles)
- In SIMD, instruction flow orchestrated only by the MAIN core \rightarrow Less energy
- Cores resume in MIMD mode on divergent branches (..or control tasks) → Flexibility

Broadcasting Share Data



Overhead: many clk cycles to unlock execution in case of concurrent accesses

- Eliminate overhead to access at same address → BROADCAST UNIT
- Misalign static data and stacks to avoid accesses to the same mem bank

Data memory Hierarchy: DMA-based, SW managed



ETH ZUrich

An additional I/O controller for IO, off-chip Memory



All together in VEGA: Extreme Edge IoT Processor

- RISC-V cluster (8cores +1) 614GOPS/W @ 7.6GOPS (8bit DNNs), 79GFLOPS/W @ 1GFLOP (32bit FP appl)
- Multi-precision HWCE(4b/8b/16b) 3×3×3 MACs with normalization / activation: 32.2GOPS and 1.3TOPS/W (8bit)
 1.7 uW cognitive unit for
 - 1.7 µW cognitive unit for autonomous wake-up from retentive sleep mode



D. Rossi et al., "Vega: A Ten-Core SoC for IoT Endnodes With DNN Acceleration and Cognitive Wake-Up From MRAM-Based State-Retentive Sleep Mode," in IEEE Journal of Solid-State Circuits, vol. 57, no. 1, pp. 127-139, Jan. 2022

All together in VEGA: Extreme Edge IoT Processor

- RISC-V cluster (8cores +1) 614GOPS/W @ 7.6GOPS (8bit DNNs), 79GFLOPS/W @ 1GFLOP (32bit FP appl)
- Multi-precision HWCE(4b/8b/16b) 3×3×3 MACs with normalization / activation: 32.2GOPS and 1.3TOPS/W (8bit)
- 1.7 µW cognitive unit for autonomous wake-up from retentive sleep mode
- Fully-on chip DNN inference with 4MB MRAM



| | Technology | 22nm FDSOI |
|--|------------|-------------------|
| | Chip Area | 12mm ² |
| | SRAM | 1.7 MB |
| | MRAM | 4 MB |
| | VDD range | 0.5V - 0.8V |
| | VBB range | 0V - 1.1V |
| | Fr. Range | 32 kHz - 450 MHz |
| | Pow. Range | 1.7 µW - 49.4 mW |





System Scalability... How many processors?



- Performance bottleneck at the core's boundaries (single 32-bit data port)
- Energy efficiency bounded by limited scalability of low-latency local interco
- Custom datapaths to improve throughput and efficiency..

Note: ..but at which cost? See Gianna Paulin's Lecture



Luca Benini, Alessandro Capotondi, Alessandro Ottaviano, Alessio Burrello, Alfio Di Mauro, Andrea Borghesi, Andrea Cossettini, Andreas Kurth, Angelo Garofalo, Antonio Pullini, Arpan Prasad, Bjoern Forsberg, Corrado Bonfanti, Cristian Cioflan, Daniele Palossi, Davide Rossi, Fabio Montagna, Florian Glaser, Florian Zaruba, Francesco Conti, Georg Rutishauser, Germain Haugou, Gianna Paulin, Giuseppe Tagliavini, Hanna Müller, Luca Bertaccini, Luca Valente, Manuel Eggimann, Manuele Rusci, Marco Guermandi, Matheus Cavalcante, Matteo Perotti, Matteo Spallanzani, Michael Rogenmoser, Moritz Scherer, Moritz Schneider, Nazareno Bruschi, Nils Wistoff, Pasquale Davide Schiavone, Paul Scheffler, Philipp Mayer, Robert Balas, Samuel Riedel, Segio Mazzola, Sergei Vostrikov, Simone Benatti, Stefan Mach, Thomas Benz, Thorir Ingolfsson, Tim Fischer, Victor Javier Kartsch Morinigo, Vlad Niculescu, Xiaying Wang, Yichao Zhang, Frank K. Gürkaynak, all our past collaborators and many more that we forgot to mention





http://pulp-platform.org



@pulp_platform