RISC-V Geneology

Tony Chen
David A. Patterson

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2016-6
http://www.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-6.html

January 24, 2016

Introduction

RISC-V is an open instruction set designed along RISC principles developed originally at UC Berkeley[1] and is now set to become an open industry standard under the governance of the RISC-V Foundation (www.riscv.org). Since the instruction set architecture (ISA) is unrestricted, organizations can share implementations as well as open source compilers and operating systems. Designed for use in custom systems on a chip, RISC-V consists of a base set of instructions called RV32I along with optional extensions for multiply and divide (RV32M), atomic operations (RV32A), single-precision floating point (RV32F), and double-precision floating point (RV32D). The base and these four extensions are collectively called RV32G.

This report discusses the historical precedents of RV32G. We look at 18 prior instruction set architectures, chosen primarily from earlier UC Berkeley RISC architectures and major proprietary RISC instruction sets. Among the 122 instructions in RV32G:

- 6 instructions do not have precedents among the selected instruction sets,
- 98 instructions of the 116 with precedents appear in at least three different instruction sets.

If you are aware of instruction set architectures that are forefathers of RV32G instructions that we list with few or no precedents, please contact an author (pattrsn@cs.berkeley.edu).

Methodology

We consider instructions precedents if the instruction implements the same behavior as the corresponding RISC-V instruction. We label instructions that appear in at least three of the listed instruction set architectures *standard*, call instructions that appear in one or two *infrequent*, and those without precedent *unique*. The table below lists the 18 comparison instruction sets in this report.

Year Published	Instruction Set Architecture	Year Published	Instruction Set Architecture
1964	CDC 6600 [1]	1992	DEC Alpha [12]
1981	RISC I [2] / RISC II [3]	1992	MIPS III [13]
1984	SOAR (RISC III) [4]	1992	IBM PowerPC [14]
1984	Intel i960 [5]	1992	Torrent T0 [15][16][17]
1985	IBM RP3 [6][7]	1994	MIPS IV [18]
1987	ARMv2 [8]	1995	PA-RISC 2.0 [19]
1988	SPUR (RISC IV) [9]	1997	Hitachi SH-4 [20]
1990	DLX [10]	2002	ARMv6 [21]
1990	SPARCv8 [11]	2003	Cray X1 [22]

[1] Waterman, A., Lee, Y., Patterson, D.A. and Asanovic, K., 2011. The RISC-V instruction set manual, volume i: Base user-level ISA. *EECS Department, UC Berkeley, Tech. Rep. UCB/EECS-2011-62.*

Instruction Set Lineage

1964	1981	1984	1984	1985	1987	1988	1990	1990	1992	1992	1992	1992	1994	1995	1997	2002	2003	2011
CDC 6600	RISC I RISC II	SOAR	Intel i960	IBM RP3[1]	ARMv2	SPUR	DLX	SPARCv8	DEC Alpha	MIPS III	IBM PowerPC	Torrent T0[2]	MIPS IV	PA-RISC 2.0	Hitachi SH-4	ARMv6	Cray X1	RISC V
	LDHI						LHI	STHI		LUI		LUI	LUI	LDIL			EXTH	LUI
					ADD[3]											ADD[3]		AUIPC
RJ		CALL	BAL	BALI	BL	JUMP/CALL	JAL	JMPL		JAL	BL	JAL	JAL			BL	J	JAL
RJ		CALL	BAL	BALR	BL	JUMP_REGISTER	JALR	JMPL		JALR	BLR	JALR	JALR	BLR	JSR	BL	J	JALR
EQ	JMPR	SKIP+CALL	BE		BEQ	CMP_BRANCH_LIKELY	BEQ	BICC	BEQ	BEQ	BEQ	BEQ	BEQ	CMPB	BF	BEQ	BZ	BEQ
NE	JMPR	SKIP+CALL	BNE		BNE	CMP_BRANCH_LIKELY	BNE	BICC	BNE	BNE	BNE	BNE	BNE	CMPB	BF	BNE	BN	BNE
LT	JMPR	SKIP+CALL	BL		BLT	CMP_BRANCH_LIKELY		BICC	BLT		BLT			CMPB	BF	BLT	BLT	BLT
GE	JMPR	SKIP+CALL	BGE		BGE	CMP_BRANCH_LIKELY		BICC	BGE		BGE			CMPB	BF	BGE	BGE	BGE
	JMPR	SKIP+CALL	CMPOBL			CMP_BRANCH_LIKELY					BLT			CMPB	BF		BLT	BLTU
	JMPR	SKIP+CALL	CMPOBGE			CMP_BRANCH_LIKELY					BGE			CMPB	BF		BGE	BGEU
	LDBS		LDIB		LDRB		LB	LDSB		LB	LBZ	LB	LB		MOV.B	LDRSB		LB
	LDS	LOADC	LDIS	LHA			LH	LDSH	LDL	LH	LHZ	LH	LH		MOV.W	LDRSH		LH
SAi	LDL	LOAD	LD	L	LDRB	LOAD_32	LW	LD	LDQ	LW	LWZ	LW	LW	LDW	MOV.L	LDR	W [ADDRESS]	LW
	LDBU		LDOB	LC			LBU	LDUB		LBU		LBU	LBU	LDB		LDRB	W, EXTB	LBU
	LDSU		LDOS	LH			LHU	LDUH		LHU	LHA	LHU	LHU	LDH		LDRH	W,EXTH	LHU
	STB		STIB	STC	STRB		SB	STB		SB	STB	SB	SB	STB	MOV.B	STRB	EXTB, W	SB
	STS		STIS	STH			SH	STH	STL	SH	STH	SH	SH	STH	MOV.W	STRH	EXTH, W	SH
SAi	STL	STORE	ST	ST	STR	STORE_32	SW	ST	STQ	SW	STW	SW	SW	STW	MOV.L	STR	[ADDRESS] W	SW
SXi	ADD	ADD		AI	ADD	ADD	ADDI	ADD	ADD	ADDI	ADDI	ADDI	ADDI	ADDI	ADD	ADD	+	ADDI
							SLTI			SLTI		SLTI	SLTI					SLTI
										SLTIU		SLTIU	SLTIU					SLTIU
	XOR	XOR			EOR	XOR	XORI	XOR	XOR	XORI	XORI	XORI	XORI		XOR	EOR	^	XORI
	OR	OR			OR	OR	ORI	OR	BIS	ORI	ORI	ORI	ORI		OR	OR	|	ORI
	AND	AND		NILO	AND	AND	ANDI	AND	AND	ANDI	ANDI	ANDI	ANDI		AND	AND	&	ANDI
LXi	SLL	SLA		SLI	LSL	SLL	SLLI	SLL			SLW				SHLL /SHLL2/8/16	LSL	<<	SLLI
	SRL	SRL		SRI	LSR	SRL	SRLI	SRL			SRW				SHRL /SHRL2/8/16	LSR	>>	SRLI
AXi	SRA	SRA		SARI	ASR	SRA	SRAI	SRA			SRAWI				SHAR	ASR	+>>	SRAI
IXi +	ADD	ADD	ADDI	A	ADD	ADD	ADD	ADD	ADD	ADD	ADDI	FXADD/ADD	ADD	ADD	ADD	ADD	+	ADD
IXi -	SUB/SUBR	SUB	SUBI	S	SUB	SUBTRACT	SUB	SUB	SUB	SUB	SUB	SUB	SUB	SUB	SUB	SUB	-	SUB
LXi	SLL	SLA	SHLI	SL	LSL	SLL	SLL	SLL	SLL	SLL	SLW	SLL	SLL		SHAD	LSL	<<	SLL
							SLT			SLT		SLT	SLT					SLT
										SLTU		SLTU	SLTU					SLTU
BXi	XOR	XOR	XOR	XOR	EOR	XOR	XOR	XOR	XOR	XOR	XORI	XOR	XOR	XOR	XOR	EOR	^	XOR
	SRL	SRL	SHRO	SR	LSR	SRL	SRL	SRL	SRL	SRL	SRW	SRL	SRL		SHRL /SHRL2/8/16	LSR	>>	SRL
BXi	SRA	SRA	SHRI	SAR	ASR	SRA	SRA	SRA	SRA	SRA	SRAW	SRA	SRA		SHAD	ASR	+>>	SRA
BXi	OR	OR	OR	OR	ORR	OR	OR	OR	BIS	ORI	ORI	ORI	ORI	OR	OR	ORR	|	OR
BXi	AND	AND	AND	N	AND	AND	AND	AND	AND	AND	ANDI	AND	AND	AND	AND	AND	&	AND
									MB	SYNC	SYNC	SYNC	SYNC	SYNC			GSYNC	FENCE
									CALL_PAL IMB		ISYNC			FIC			LSYNC	FENCE.I
		TRAP	CALLS	SVC	SWI	CALL_KERNEL	TRAP	TRAP	CALL_PAL GENTRAP	SYSCALL	SC	SYSCALL	SYSCALL	B,GATE	TRAPA	SWI	SYSCALL	SCALL
			MARK						CALL_PAL BPT	BREAK		BREAK	BREAK	BREAK	BRK	BKPT	BREAK	SBREAK
								RDASR	RPCC								AI CK	RDCYCLE
																		RDCYCLEH
								RDASR			MFSPR						AI CK	RDTIME
											MFSPR							RDTIMEH
								RDASR										RDINSTRET
																		RDINSTRETH
			MULI		MUL		MULT	SMUL	MUL	MULT	MULLW	MULT/FXMUL	MULT		MUL.L	MUL	*	MUL
								SMUL		MULT	MULHW	MULT/FXMUL	MULT			SMULL	HI*	MULH
												FXMUL						MULHSU
			MULO					UMUL	UMULH	MULTU	MULHWU	MULTU/FXMUL	MULTU	XMPYU		UMULL		MULHU
			DIVI				DIV	SDIV		DIV	DIVW	DIV	DIV				/	DIV
			DIVO				DIVU	UDIV		DIVU	DIVWU	DIVU	DIVU					DIVU
			REMO							DIVU			DIVU					REMU
			SYNLD					LDSTUB	LDL_L	LL	LWARX	LL	LL			LDREX		LR.W
				FETCH & STORE				LDSTUB	STL_C	SC	STWCX	SC	SC			STREX		SC.W
								SWAP								SWP	ACSWAP	AMOSWAP.W
			ATADD	FETCH & ADD													AADD	AMOADD.W
																	AFAX	AMOXOR.W
				FETCH & AND													AFAX	AMOAND.W
				FETCH & OR													AFAX	AMOOR.W
				FETCH & MIN														AMOMIN.W
				FETCH & MAX														AMOMAX.W
																		AMOMINU.W
																		AMOMAXU.W
SAi				WTFR	LDF	LOAD_SINGLE	LF	LDF	LDS	LWC1	LFS	LWC1	LWC1	FLDW	FMOV	FLDS	S [ADDRESS]	FLW
SAi				RDFR	STF	STORE_SINGLE	SF	STF	STS	SWC1	STFS	SWC1	SWC1	FSTW	FMOV	FSTS	[ADDRESS] S	FSW
											FMADDS		MADD.S	FMPYFADD	FMAC	FMACS		FMADD.S
											FMSUBS		MSUB.S			FMSCS		FMSUB.S
											FNMSUBS		NMSUB.S	FMPYSUB		FNMSCS		FNMSUB.S
											FNMADDS		NMADD.S	FMPYNFADD		FNMACS		FNMADD.S
FXi +			ADDR	ADDS	ADF	FADD	ADDF	FADDs	ADDS	ADD.S	FADDS	ADD.S	ADD.S	FADD	FADD	FADDS	+	FADD.S
FXi -				SUBS	SUF	FSUB	SUBF	FSUBs	SUBS	SUB.S	FSUBS	SUB.S	SUB.S	FSUB	FSUB	FSUBS	-	FSUB.S
FXi *			MULR	MULS	MUF	FMUL	MULTF	FMULs	MULS	MUL.S	FMULS	MUL.S	MUL.S	FMPY	FMUL	FMULS	*	FMUL.S
FXi /			DIVR	DIVS	DVF	FDIV	DIVF	FDIVs	DIVS	DIV.S	FDIVS	DIV.S	DIV.S	FDIV	FDIV	FDIVS	/	FDIV.S
			SQRTR		SQT			FSQRTs		SQRT.S		SQRT.S	SQRT.S	FSQRT	FSQRT	FSQRTS	SQRT	FSQRT.S
			CPYSRE[4]						CPYS								CPYS	FSGNJ.S
			CPYRSRE[4]			FNEGATE			CPYSN									FSGNJN.S
																		FSGNJX.S
											FSEL[5]							FMIN.S
											FSEL[5]							FMAX.S
			CVTRI	FSW/RSW/TSW	FIX	EXTENDED_TO_INT[6]	CVTF2I	FsTOi		CVT.W.S	FCTIW	CVT.W.S	CVT.W.S	FCNV	FTRC	FTOSIS	W, S	FCVT.W.S
														FCNV		FTOUIS		FCVT.WU.S
					RFS	FROM_FPU	MOVFP2I			MFC1	FMR	MFC1	MFC1			FMRS		FMV.X.S
			CMPR		CMF		EQF	FCMPs		C.EQ.S	FCMPO	C.EQ.S	C.EQ.S	FCMP	FCMPEQ		==	FEQ.S
			CMPR		CMF		LTF	FCMPs		C.LT.S	FCMPO	C.LT.S	C.LT.S	FCMP	FCMPGT		<	FLT.S
					CMF		LEF	FCMPs		C.LE.S	FCMPU	C.LE.S	C.LE.S	FCMP			<=	FLE.S
			CLASSR															FCLASS.S
				CWS	FLT	INT_TO_EXTENDED[6]	CVTI2F	FiTOs	CVTQS	CVT.S.W		CVT.S.W	CVT.S.W	FCNV	FLOAT	FITOS	W, S	FCVT.S.W
														FCNV		FUITOS		FCVT.S.WU
					WFS	TO_FPU	MOVI2FP				FMR				FSTS	FMSR		FMV.S.X
				RDSTR/WTSTR				STFSR /LDFSR	MF_FPCR /MT_FPCR	CFC1/CTC1		CFC1/CTC1	CFC1/CTC1	FLDWX /FSTWX			AI CK/CI AK	FRCSR
								STFSR	MF_FPCR	CFC1		CFC1	CFC1	FLDWX				FRRM
				RDSTR				STFSR	MF_FPCR	CFC1		CFC1	CFC1	FLDWX			AI CK	FRFLAGS
								LDFSR	MT_FPCR	CTC1		CTC1	CTC1	FSTWX			ROUND/TRUNC /CEIL/FLOOR	FSRMI
				WTSTR				LDFSR	MT_FPCR	CTC1		CTC1	CTC1	FSTWX			CI AK	FSFLAGSI
SAi				WTFR		LOAD_DOUBLE	LD	LDDF	LDT	LDC1	LFD	LDC1	LDC1	FLDD	FMOV	FLDD	D [ADDRESS]	FLD
SAi				RDFR		STORE_DOUBLE	SD	STDF	STT	SDC1	STFD	SDC1	SDC1	FSTD	FMOV	FSTD	[ADDRESS] D	FSD
											FMADD		MADD.D	FMPYFADD		FMACD		FMADD.D
											FMSUB		MSUB.D			FMSCD		FMSUB.D
											FNMSUB		NMSUB.D	FMPYSUB		FNMSCD		FNMSUB.D
											FNMADD		NMADD.D	FMPYNFADD		FNMACD		FNMADD.D
DXi +			ADDR	ADDL	ADF	FADD	ADDD	FADDd	ADDT	ADD.D	FADD	ADD.D	ADD.D	FADD	FADD	FADDD	+	FADD.D
DXi -			SUBR	SUBL	SUF	FSUB	SUBD	FSUBd	SUBT	SUB.D	FSUB	SUB.D	SUB.D	FSUB	FSUB	FSUBD	-	FSUB.D
DXi *			MULR	MULL	MUF	FMUL	MULTD	FMULd	MULT	MUL.D	FMUL	MUL.D	MUL.D	FMPY	FMUL	FMULD	*	FMUL.D
DXi /			DIVR	DIVL	DVF	FDIV	DIVD	FDIVd	DIVT	DIV.D	FDIV	DIV.D	DIV.D	FDIV	FDIV	FDIVD	/	FDIV.D
			SQRTRL		SQT			FSQRTd		SQRT.D		SQRT.D	SQRT.D	FSQRT	FSQRT	FSQRTD	SQRT	FSQRT.D
			CPYSRE[4]						CPYS								CPYS	FSGNJ.D
			CPYRSRE[4]						CPYSN									FSGNJN.D
																		FSGNJX.D
											FSEL[5]							FMIN.D
											FSEL[5]							FMAX.D
				CLS			CVTD2F	FdTOs	CVTTS	CVT.S.D		CVT.S.D	CVT.S.D	FCNV	FCNVDS	FCVTDS	S, D	FCVT.S.D
				CSL			CVTF2D	FsTOd		CVT.D.S		CVT.D.S	CVT.D.S	FCNV	FCNVSD	FCVTSD	D, S	FCVT.D.S
			CMPRL		CMFE		EQD	FCMP	CMPTEQ	C.EQ.D		C.EQ.D	C.EQ.D	FCMP	FCMPEQ		==	FEQ.D
			CMPRL		CMFE		LTD	FCMP	CMPTLT	C.LT.D		C.LT.D	C.LT.D	FCMP	FCMPGT		<	FLT.D
					CMFE		LED	FCMP	CMPTLE	C.LE.D		C.LE.D	C.LE.D	FCMP			<=	FLE.D
			CLASSRL															FCLASS.D
				FLW/RLW/TLW	FIX	EXTENDED_TO_INT[6]	CVTD2I	FdTOi	CVTTQ	CVT.W.D	FCTIW	CVT.W.D	CVT.W.D	FCNV	FTRC	FTOSID	W, D	FCVT.W.D
														FCNV		FTOUID		FCVT.WU.D
				CWL	FLT	EXTENDED_TO_INT[6]	CVTI2D	FiTOd	CVTQT	CVT.D.W		CVT.D.W	CVT.D.W	FCNV	FLOAT	FSITOD	D, W	FCVT.D.W
														FCNV		FUITOD		FCVT.D.WU

Notes:

1. IBM RP3 augments the IBM RT/PC architecture with fetch-and-op functions.
2. Torrent T0 supplements the MIPS II instruction set.
3. Specific instruction: ADD Rx, R15, 8_bit_immediate &07. R15 is the PC. The 12 bit immediate is split into 8 bit immediate and 4 bit "position", which can specify any odd bit starting position. &07 bit position starting bit 25 for 24-bit PC.
4. CPYSRE and CPYRSRE operate on the extended format, so SETBIT and CLRBIT instructions must be used before operating upon single and double precision floating points
5. To perform FMIN, use FSUB fs, fa, fb then FSEL fx, fs, fb, fa. To perform FMAX, use FSUB fs, fa, fb then FSEL fx, fs, fa, fb. Registers fa and fb are the register contents being compared.
6. INT_TO_EXTENDED requires an EXTENDED_TO_SINGLE or EXTENDED_TO_DOUBLE to convert from integer to single or double and converting floating point to integer requires SINGLE_TO_EXTENDED or DOUBLE_TO_EXTENDED followed by EXTENDED_TO_INT.

Results

The table on the prior page lists all the RV32G instructions in the rightmost column, with the 18 earlier ISAs in the columns to the left. The relevant precedents of an RV32G instruction are listed in each row. Some instruction sets contain multiple instructions that implement similar behavior to a RISC-V instruction, so we list them all using a "/" to separate them

Standard RV32G Instructions

The following 98 of the 122 RV32G instructions are considered standard instructions, common to almost all instruction set architectures.

1. LUI
2. JAL
3. JALR
4. BEQ
5. BNE
6. BLT
7. BGE
8. BLTU
9. BGEU
10. LB
11. LH
12. LW
13. LBU
14. LHU
15. SB
16. SH
17. SW
18. ADDI
19. SLTI
20. SLTIU
21. XORI
22. ORI
23. ANDI
24. SLLI
25. SRLI
26. SRAI
27. ADD
28. SUB
29. SLL
30. SLT
31. SLTU
32. XOR
33. SRL
34. SRA
35. OR
36. AND
37. FENCE
38. FENCE.I
39. SCALL
40. SBREAK
41. RDCYCLE
42. RDTIME
43. RDINSTRET
44. MUL
45. MULH
46. MULHU
47. DIV
48. DIVU
49. REMU
50. LR.W
51. SC.W
52. AMOSWAP.W
53. AMOADD.W
54. FLW
55. FSW
56. FMADD.S
57. FMSUB.S
58. FNMSUB.S
59. FNMADD.S
60. FADD.S
61. FSUB.S
62. FMUL.S
63. FDIV.S
64. FSQRT.S
65. FSGNJ.S
66. FSGNJN.S
67. FCVT.W.S
68. FMV.X.S
69. FEQ.S
70. FLT.S
71. FLE.S
72. FCVT.S.W
73. FMV.S.X
74. FRCSR
75. FRRM
76. FRFLAGS
77. FSRMI
78. FSFLAGSI
79. FLD
80. FSD
81. FMADD.D
82. FMSUB.D
83. FNMSUB.D
84. FNMADD.D
85. FADD.D
86. FSUB.D
87. FMUL.D
88. FDIV.D
89. FSQRT.D
90. FSGNJ.D
91. FSGNJN.D
92. FCVT.S.D
93. FCVT.D.S
94. FEQ.D
95. FLT.D
96. FLE.D
97. FCVT.W.D
98. FCVT.D.W

Infrequent RV32G Instructions

Here are six categories containing 18 RV32G instructions that have one or two precedents.

- **AUIPC - Add Upper Immediate to PC**

The ARM instruction set contains a versatile ADD instruction which can shift and add an immediate to a register. Register R15 has been the program counter register as early as ARMv2.

- **RDTIMEH - write a 32-bit register with the value from bits 63-32 of the counter containing the the wall clock time (TIME).**

Move From Time Base Upper (MFTBU) in PowerPC is the precedent.

- **MULHSU - returns the upper 32 bits of the 64-bit product for signed×unsigned integer operand multiplication.**

The precedent instruction was FXMUL from the Torrent computer.

- **FMAX.{S/D}/FMIN.{S/D}/FCLASS.{S/D} - FMIN.S/D and FMAX.S/D write the smaller or larger of rs1 and rs2 to rd. FCLASS.S/D examines the value in rs1 and writes to integer register rd a 10-bit mask that indicates the Fl. Pt. number class.**

The instructions are recommended in the IEEE 754-1985 standard, but were not required until the IEEE 754-2008 revision. IBM PowerPC implements FMAX and FMIN using its FSEL instruction. The Intel i960 implemented the recommended FCLASS instruction.

- **FCVT.WU.{S/D},FCVT.{S/D}.WU - convert floating point to unsigned integers and unsigned integers to floating point.**

PA-RISC 2.0 implemented these instructions in 1995, but many instruction sets emulate these instructions as shown below:

```
max = 0x80000000
if (f < max)  u = float2int(f);
   else  u = max + float2int(f - max);
```

- **AMO{AND/OR/XOR/MIN/MAX}.W - atomic memory operation (AMO) instructions perform read-modify-write operations for synchronization.**

IBM RP3 implemented fetch-and-AND, fetch-and-OR, fetch-and-MIN, and fetch-and-MAX. RP3 is based on the IBM RT/PC instruction set. AMOXOR appeared only in the Cray X1 architecture.

Unique RV32G Instructions

Here are the three categories containing the six RV32G instructions that have no known precedents.

- **AMO{MINU/MAXU}.W - AMO instructions perform read-modify-write operations.**

The unsigned versions of minimum and maximum (AMOMINU and AMOMAXU) do not appear in any of the listed instruction sets.

- **RD{CYCLEH/INSTRETH} - write a 32-bit register with the value from bits 63-32 of the counter containing the number of clock cycles (CYCLE) or the number of instructions retired (INSTRET).**

The RISC-V performance counters are 64 bits wide, but RV32G is a 32-bit ISA. To accommodate these counters, they must be read in 32-bit chunks. Performance counters are often implementation specific, so many instruction sets only provide methods to supply these operations.

- **FSGNJX.{S/D} - takes all bits except the sign bit from operand rs1, with the sign bit set from the XOR of the sign bits of operands rs1 and rs2.**

An instruction that has no precedent in the listed ISAs, FSGNJX can be used to perform the floating point pseudo-op, FABS, which takes the absolute value of the floating point number in a source register and stores it in the destination register.

Conclusion

Half of the infrequent floating-point instructions were suggested but not required floating-point operations in the IEEE 754-1985 standard, thus did appear in many instruction sets. In the future, we will try to find more precedents for the 18 infrequent instructions and the 6 unique ones. If you have suggestions, please contact an author (pattrsn@cs.berkeley.edu).

Bibliography

[1] *Control Data 6400/6500/6600 Computer Systems Reference Manual*, 4th ed., Control Data Corp., St. Paul, MN, Feb. 1967 [Online]. Available: http://www.ygdes.com/CDC/cdc6600.html

[2] C. H. Séquin and D. A. Patterson, "Design and Implementation of RISC I," University of California, Berkeley, Berkeley, CA, Tech. Rep. UCB/CSD-82-106, Oct. 1982, [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/1982/5449.html

[3] R. W. Sherburne et al., "A 32-Bit NMOS Microprocessor with a Large Register File," *IEEE J. Solid-State Circuits*, vol. SC-19, no. 5, pp. 682-689, Oct. 1984.

[4] A. D. Samples et al., "SOAR Architecture," University of California, Berkeley, Berkeley, CA, Tech. Rep. UCB/CSD-85-226, 1985, [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/1985/5940.html

[5] *80960 Programmer's Reference Manual*, Intel Corp., Santa Clara, CA, 1988, [Online]. Available: http:// www.bitsavers.org

[6] G. S. Almasi and S. L. Harvey, "RP3," in International Specialist Seminar on the Design and Application of Parallel Digital Processors, Lisbon, Portugal, 1988, pp. 67-73.

[7] W. C. Brantley et al., "The RP3 Processor-Memory Element," in Proc. 1985 International Conf. on Parallel Processing, 1985, pp. 782-789.

[8] P. J. Cockerell, *ARM Assembly Language Programming*. Hemel Hempstead, England: Computer Concepts Ltd., 1987.

[9] M. D. Hill et al., “SPUR: A VLSI Multiprocessor Workstation,” University of California, Berkeley, Berkeley, CA, Tech. Rep. UCB/CSD-86-273, Dec. 1985, [Online]. Available: http://ww w.eecs.berkeley.edu/Pubs/TechRpts/1985/6083.html

[10] J. L. Hennessy and D. A. Patterson, “The DLX Architecture,” in *Computer Architecture: A Quantitative Approach*. San Mateo, CA: Morgan Kaufmann Publishers, Inc., 1990, pp. 160-167.

[11] *The SPARC Architecture Manual Version 8*, SPARC International, Inc., Menlo Park, CA, 1992, [Online]. Available: http://www.gaisler.com/doc/sparcv8.pdf

[12] *Alpha Architecture Reference Manual*, Digital Equipment Corp., Burlington, MA, 1992, [Online]. Available: http://www.bitsavers.org

[13] *MIPS R4000 Microprocessor User's Manual*, 2nd ed., MIPS Technologies, Inc., Mountain View, CA, 1994, [Online]. Available: http://groups.csail.mit.edu/cag/raw/documents/R4400_Uman_book_Ed2.pdf

[14] *The PowerPC Architecture: A Specification for a New Family of RISC Processors*, 2nd ed., International Business Machines, Inc., Austin, TX, 1994, [Online]. Available: http://www.bitsavers.org

[15] K. Asanovic and J. Beck, “Torrent Architecture Manual,” UC Berkeley, Tech. Rep. UCB/CSD-97-930, Jan. 1997.www.eecs.ber keley.edu/Pubs/TechRpts/1997/5413.html

[16] *R6000 CPU Chip Spec*, MIPS Computer Systems, Inc., Sunnyvale, CA, 1987, [Online]. Available: https://www.linux-mips.org/wiki/R6000

[17] *R6010 CPU Floating-Point Controller*, MIPS Computer Systems, Inc., Sunnyvale, CA, 1987, [Online]. Available: https://www.linux-mips.org/wiki/R6000

[18] *MIPS IV Instruction Set*, 3rd ed., MIPS Technologies, Inc., Mountain View, CA, 1995, [Online]. Available: http://www.cs.cmu.edu/afs/cs/academic/class/15740-f97/public/doc/mips-isa.pdf

[19] *PA-RISC 2.0*, Hewlett-Packard Company, 1995, [Online]. Available: https://parisc.wiki.kernel.org /index.php/Technical_Documentation

[20] *SH-4 CPU Core Architecture*, STMicroelectronics and Hitatchi, Ltd., 2002, [Online]. Available: http://www.st.com/st-web-ui/static/active/en/resource/technical/document/user_manual/CD00147165.pdf

[21] *ARM Architecture Reference Manual*, Rev. I, ARM, Ltd., Cambridge, England, 2005.

[22] *Cray Assembly Language (CAL) for Cray X1 Systems Reference Manual*, Cray Inc., 2003, [Online]. Available: http://docs.cray.com/books/S-2314-51