CAHPv3 Instruction Set Specification

Naoki Matsumoto, Ryotaro Banno, Kotaro Matsuoka2020/09/21

1 Programmers' Model for CAHPv3 ISA

1.1 Integer Registers

15	0
x0 / ra(return address)	
x1 / sp(stack pointer)	
x2 / fp(frame pointer)	
x3 / s0	
x4 / s1	
x5 / s2	
x6 / s3	
x7 / s4	
x8 / a0	
x9 / a1	
x10 / a2	
x11 / a3	
x12 / a4	
x13 / a5	
x14 / t0	
x15 / t1	
16	
15	0
pc	
16	

Figure 1: CAHPv3 integer register state.

2 24-bit Length Instructions

2.1 Integer Computational Instructions

Integer Register-Register Instructions

23	20 19 16	6 15 12	11 8	3 7 0
0000	rs2	rs1	rd	opcode
4	4	4	4	8
	src2	$\operatorname{src}1$	dest	ADD
	src2	$\operatorname{src}1$	dest	SUB
	src2	$\operatorname{src}1$	dest	AND
	src2	$\operatorname{src}1$	dest	XOR
	src2	$\operatorname{src}1$	dest	OR
	src2	$\operatorname{src}1$	dest	LSL
	src2	$\operatorname{src}1$	dest	LSR
	src2	$\operatorname{src}1$	dest	ASR

Instruction	Opcode	Formula	Description
ADD rd, rs1, rs2	00000001	$rd \leftarrow rs1 + rs2$	Add rs1 value to rs2 value, and store to rd.
SUB rd, rs1, rs2	00001001	$rd \leftarrow rs1 - rs2$	Subtract rs2 value from rs1 value, and store to rd.
AND rd, rs1, rs2	00010001	$rd \leftarrow rs1 \wedge rs2$	Do bit and between rs1 value and rs2 value, and store to rd.
XOR rd, rs1, rs2	00011001	$rd \leftarrow rs1 \oplus rs2$	Do bit xor between rs1 value and rs2 value, and store to rd.
OR rd, rs1, rs2	00100001	$rd \leftarrow rs1 \vee rs2$	Do bit or between rs1 value and rs2 value, and store to rd.
LSL rd, rs1, rs2	00101001	$rd \leftarrow rs1 \ll rs2$	Do left logical shift rs1 value with width rs2 value, and store to rd.
LSR rd, rs1, rs2	00110001	$rd \leftarrow rs1 \gg rs2$	Do right logical shift rs1 value with width rs2 value, and store to rd.
ASR rd, rs1, rs2	00111001	$rd \leftarrow rs1 \ggg rs2$	Do right arithmetic shift rs1 value with width rs2 value, and store to rd.

${\bf Integer\ Register\text{-}Immediate\ Instructions}$

23	16 15	12 11	=	8 7	6 5	0
simm10[7:0	rs	1	$^{\mathrm{rd}}$	simm10[9:8]	opcode	
8	4		4	2	6	
simm10[7:0])] sr	$^{\mathrm{c}}$	dest	simm10[9:8]	ADDI	
simm10[7:0])] sr	$^{\mathrm{c}}$	dest	simm10[9:8]	ANDI	
simm10[7:0])] sr	$^{\mathrm{c}}$	dest	simm10[9:8]	XORI	
simm10[7:0])] sr	$^{\mathrm{c}}$	dest	simm10[9:8]	ORI	
simm10[7:0]	00	00	dest	simm10[9:8]	$_{ m LI}$	

Instruction	Opcode	Formula	Description
ADDI rd, rs1, simm10	000011	$rd \leftarrow rs1 + simm10$	Add rs1 value to simm10, and store to rd.
ANDI rd, rs1, simm10	010011	$rd \leftarrow rs1 \wedge simm10$	Do bit and between rs1 value and simm10, and store to rd.
XORI rd, rs1, simm10	011011	$rd \leftarrow rs1 \oplus simm10$	Do bit xor between rs1 value and simm10, and store to rd.
ORI rd, rs1, simm10	100011	$rd \leftarrow rs1 \vee simm10$	Do bit or between rs1 value and simm10, and store to rd.
LI rd, simm10	110101	$rd \leftarrow simm10$	Load simm10, and store to rd.

23	20 19	16 15 12	2 11	8 7 0
0000	uimm4[3:0]	rs1	rd	opcode
4	4	4	4	8
	uimm4[3:0]	src	dest	LSLI
	uimm4[3:0]	src	dest	LSRI
	uimm4[3:0]	src	dest	ASRI

Instruction	Opcode	Formula	Description
LSLI rd, rs1, uimm4	00101011	$rd \leftarrow rs1 \ll uimm4$	Do left logical shift rs1 value with width uimm4, and store to rd.
LSRI rd, rs1, uimm4	00110011	$rd \leftarrow rs1 \gg uimm4$	Do right logical shift rs1 value with width uimm4, and store to rd.
ASRI rd, rs1, uimm4	00111011	$rd \leftarrow rs1 \ggg uimm4$	Do right arithmetic shift rs1 value with width uimm4, and store to rd.

2.2 Conditional Branches

23	16 1	5 1	2 11	8 7	6 5	0
simm10	[7:0]	rs1	rs2	simm10[9:8]	opcode	
8		4	4	2	6	
$\operatorname{simm}1$	0[7:0]	$\operatorname{src}1$	src2	simm10[9:8]	BEQ	
$\operatorname{simm}1$	0[7:0]	$\operatorname{src}1$	src2	simm10[9:8]	BNE	
$\operatorname{simm}1$	0[7:0]	$\operatorname{src}1$	src2	simm10[9:8]	BLT	
$\operatorname{simm}1$	0[7:0]	$\operatorname{src}1$	src2	simm10[9:8]	BLTU	
$\operatorname{simm}1$	0[7:0]	$\operatorname{src}1$	src2	simm10[9:8]	BLE	
$\operatorname{simm}1$	0[7:0]	$\operatorname{src}1$	src2	simm10[9:8]	BLEU	

Instruction	Opcode	Description
BEQ rs1, rs2, simm10	001111	If rs1 value is equal to rs2 value, jump to PC + simm10.
BNE rs1, rs2, simm10	101111	If rs1 value is not equal to rs2 value, jump to PC + simm10.
BLT rs1, rs2, simm10	110111	If rs1 signed value is less than rs2 one, jump to PC + simm10.
BLTU rs1, rs2, simm10	010111	If rs1 unsigned value is less than rs2 one, jump to PC + simm10.
BLE rs1, rs2, simm10	111111	If rs1 signed value is equal or less than rs2 one, jump to PC + simm10.
BLEU rs1, rs2, simm10	011111	If rs1 unsigned value is equal or less than rs2 one, jump to PC + simm10.

2.3 Load and Store Instructions

23	16	15 15	2 11	8 7	6 5	0
simm10[7:0]	rs	rd	simm10	[9:8] opco	ode
8		4	4	2	6	
simm10	[7:0]	src	dest	simm10	[9:8] LV	V
simm10	[7:0]	src	dest	simm10	[9:8] LI	3
simm10	[7:0]	src	dest	simm10	[9:8] LB	U

Instruction	Opcode	Formula	Description
LW rd gimm10(rg)	010101	$rd \leftarrow [rs + simm10]$	Load word value with address rs value $+ simm10$,
LW rd, simm10(rs)	010101	$ra \leftarrow [rs + simm10]$	and store to rd.
LB rd, simm10(rs)	100101	$rd \leftarrow [rs + simm10]$	Load byte value with address rs value + simm10,
LB rd, simmio(rs)	100101		and store sign extended one to rd.
LBU rd, simm10(rs)	000101	$rd \leftarrow [rs + simm10]$	Load byte value with address rs value + simm10,
LDC 1d, SIIIIII10(18)	000101	$[10 \leftarrow [1s + simm10]]$	and store to rd without sign extension.

23 16	5 15 12	2 11 8	3 7	6 5	0
simm10[7:0]	rd	rs	simm10[9:8]	opcode	
8	4	4	2	6	
simm10[7:0]	dest	src	simm10[9:8]	SW	
simm10[7:0]	dest	src	simm10[9:8]	$_{ m SB}$	

Instruction Opcode Formula		Formula	Description	
SW rs, simm10(rd)	$001101 [rd + simm10] \leftarrow 1$		Load value from rs, and store word value to address	
	001101		rs value + simm10.	
SB rs, simm10(rd)	110101	$[rd + simm10] \leftarrow rs$	Load value from rs, and store value[7:0] to address	
SD is, similito(Id)	110101		rs value $+ simm10$.	

3 16-bit Length Instructions

3.1 Integer Computational Instructions

Integer Register-Register Instructions

15	12 11	8 7	0
rs	rd	opcode	
4	4	8	
src	dest	MOV	
src	dest	ADD2	
src	dest	SUB2	
src	dest	AND2	
src	dest	XOR2	
src	dest	OR2	
src	dest	LSL2	
src	dest	LSR2	
src	dest	ASR2	

Instruction	Opcode	Formula	Description
MOV rd, rs	11000000	$rd \leftarrow rs$	Load rs value, and store to rd.
ADD2 rd, rs	10000000	$rd \leftarrow rd + rs$	Add rs1 value to rd value, and store to rd.
SUB2 rd, rs	10001000	$rd \leftarrow rd - rs$	Subtract rd value from rs value, and store to rd.
AND2 rd, rs	10010000	$rd \leftarrow rd \wedge rs$	Do bit and between rd value and rs value, and store to rd.
XOR2 rd, rs	10011000	$rd \leftarrow rd \oplus rs$	Do bit xor between rd value and rs value, and store to rd.
OR2 rd, rs	10100000	$rd \leftarrow rd \vee rs$	Do bit or between rd value and rs value, and store to rd.
LSL2 rd, rs	10101000	$rd \leftarrow rd \ll rs$	Do left logical shift rd value with width rs value, and store
LSL2 Id, IS	10101000	1d ← 1d ≪ 1s	to rd.
LSR2 rd, rs	10110000	$rd \leftarrow rd \gg rs$	Do right logical shift rd value with width rs value, and store
LD102 10, 15	10110000	14 \ 14 // 15	to rd.
ASR2 rd, rs	10111000	$rd \leftarrow rd \gg rs$	Do right arithmetic shift rd value with width rs value, and
A5112 10, 18	101111000	1 u ← 1 u ≫ rs	store to rd.

${\bf Integer\ Register\text{-}Immediate\ Instructions}$

15	12 11	8 7	6 5		0
simm6[3:0]	rd	simme	3[5:4]	opcode	
4	4	2		6	
simm6[3:0]	dest	$\operatorname{simm} 6$	6[5:4]	ADDI2	
simm6[3:0]	dest	$\operatorname{simm} 6$	3[5:4]	ANDI2	
simm6[3:0]	dest	$\operatorname{simm} 6$	3[5:4]	LSI	
simm6[3:0]	dest	$\operatorname{simm} 6$	6[5:4]	LUI	

Instruction	Opcode	Formula	Description		
ADDI2 rd, simm6	000010	$rd \leftarrow rd + simm6$	Add rd value to simm6, and store to rd.		
ANDI2 rd, simm6	010010	10010 $rd \leftarrow rd \land simm6$ Do bit and between rd value and simm6, and			
			to rd.		
LSI rd, simm6	110100	$rd \leftarrow simm6$	Load simm6, and store to rd.		
LUI rd, simm6	000100	rd ((gimm6 // 10)	Load simm6 with left logical shift with width 10,		
LOT Id, Sillillo	000100	$rd \leftarrow (simm6 \ll 10)$	and store to rd.		

15	12 11	8	3 7	0
uimm4[3:0]	rd	opcode	
4	·	4	8	
uimm4	[3:0]	dest	LSLI2	
uimm4	[3:0]	dest	LSRI2	
uimm4	[3:0]	dest	ASRI2	

Instruction	Opcode	Formula	Description		
LSLI2 rd_uimm4	00101010	$rd \leftarrow rd \ll uimm4$	Do left logical shift rd value with width uimm4,		
ESEIZ IG, unimit	00101010	ra v ra « ammir	and store to rd.		
I CDI2 rd uimm4	00110010	nd / nd > nimm1	Do right logical shift rd value with width uimm4,		
LSIGIZ IG, ullilli14	$[2 \text{ rd, uimm4} \mid 00110010 \mid \text{rd} \leftarrow \text{rd}]$		and store to rd.		
ACDI2 nd uimm4	00111010	$rd \leftarrow rd \ggg uimm4$	Do right arithmetic shift rd value with width		
ASK12 rd, ullilli14			uimm4, and store to rd.		

3.2 Unconditional Jumps

15	12 11	3 7 0
0000	rs	opcode
4	4	8
	src	JALR
	src	$_{ m JR}$

Instruction	Opcode	Formula	Description	
JALR rs	00010110	$ \begin{array}{c} ra \leftarrow PC + 2 \\ PC \leftarrow rs \end{array} $	Jump to rs value, and store $PC + 2$ to return address register ra.	
JR rs	00000110	$PC \leftarrow rs$	Jump to rs value.	

15	5 4	0
simm11[10:0]	opcode	
11	5	
simm11[10:0]	$_{ m JS}$	
simm11[10:0]	JSAL	

Instruction	Opcode	Formula	Description
JS simm11	01110	$PC \leftarrow PC + simm11$	Jump to $PC + simm11$.
JSAL simm11	11110	$ \begin{array}{c} \operatorname{ra} \leftarrow \operatorname{PC} + 2 \\ \operatorname{PC} \leftarrow \operatorname{PC} + \operatorname{simm} 11 \end{array} $	Jump to $PC + simm11$, and store $PC + 2$ to return address register ra.

3.3 Load and Store Instructions

15	$12 \ 11$	8	3 7	6 5	0
uimm7[4:1]		rd	uimm7[6:5]	(pcode
11		5			
uimm7[4:1]		dest	uimm7[6:5]]	LWSP

Instruction	Opcode	Formula	Description	
LWSP rd, uimm7(sp)	010100	$rd \leftarrow [sp + uimm7]$	Load word value with address stack pointer register value + uimm7, and store to rd.	

15	12 11	8 7	6	5 5	0
uimm7[4:1]	rs		uimm7[6:5]	opcode	
11	5				
$\operatorname{uimm7}[4:1]$	sre	С	$\operatorname{uimm7}[6:5]$	SWSP	

Instruction	Opcode	Formula	Description		
SWSP rs, uimm7(sp)	011100	$[\mathrm{sp} + \mathrm{uimm7}] \leftarrow \mathrm{rs}$	Store rs value to uimm8 with address stack pointer register value + uimm7.		

4 Instructions List