BasicRISCInstTable: System Architect BasicRISC instruction set encoding tables: Level 1 v.1.0

This document contains the encoding infrastructure for the BasicRISC instruction set architecture detailed in the System Architect Level1 Tutorial. The opcode classes are listed in the "Opcode" sheet. For each opcode class that is defined, a separate sheet is present that defines the associated function (func) codes and the associated instructions with their respective mnemonics. Finally, the PseudoInsts tab provides pseudo instruction mnemonics with their equivalent base encodings

Mnemonic	Size (bytes)	Description
UB	1	unsigned byte
UH	2	unsigned halfword
UW	4	unsigned word
UD	8	unsigned doubleword
SB	1	signed byte
SB	2	signed halfword
SW	4	signed word
SD	8	signed doubleword

Opcode	Description	Format
0	Integer Arith	ARITH.IF
1	Comparisons	
2	Memory I/O	
3	Two Operand Arith	
4	Unconditional Branches	
5	UNUSED	
6	UNUSED	
7	UNUSED	
8	UNUSED	
9	Read From Control	ReadCtrl.if
10	Conditional Branches	
11	UNUSED	
12	UNUSED	
13	UNUSED	
14	UNUSED	
15	UNUSED	
16	UNUSED	
17	Move to Control	WriteCtrl.if
18	Call/Rtn Support	
19	UNUSED	
20	UNUSED	
21	UNUSED	
22	UNUSED	
23	UNUSED	
24	UNUSED	
25	UNUSED	UNUSED
26	UNUSED	
27	UNUSED	
28	UNUSED	
29	UNUSED	
30	UNUSED	
31	UNUSED	

	ARITH.IF Format						
Bitfield	[31-25]	[24-20]	[19-15]	[14-10]	[9-5]	[5-0]	**Bitfield**
FieldName	IMM	FUNC	OPC	RT	RB	RA	**FieldName**
Mnemonics							**Description**
ADD RT, RA, RB	0x00	0x00	0x00	Rt	Rb	Ra	Add Ra + Rb; place result in Rt
SUB RT, RA, RB	0x00	0x01	0x00	Rt	Rb	Ra	Subtract Ra - Rb; place result in Rt
MUL RT, RA, RB	0x00	0x02	0x00	Rt	Rb	Ra	Multiply Ra x Rb; place result in Rt
DIV RT, RA, RB	0x00	0x03	0x00	Rt	Rb	Ra	Signed Division Ra / Rb; place result in Rt
DIVU RT, RA, RB	0x00	0x04	0x00	Rt	Rb	Ra	Unsigned Division Ra / Rb; place result in Rt
SLL RT, RA, RB	0x00	0x05	0x00	Rt	Rb	Ra	Shift left logical; Rt = Ra << Rb
SRL RT, RA, RB	0x00	0x06	0x00	Rt	Rb	Ra	Shift right logical; Rt = Ra >> Rb
SRA RT, RA, RB	0x00	0x07	0x00	Rt	Rb	Ra	Arithmetic shift right logical; Rt = Ra >> Rb
AND RT, RA, RB	0x00	0x08	0x00	Rt	Rb	Ra	Bitwise AND Rt = Ra & Rb
OR RT, RA, RB	0x00	0x09	0x00	Rt	Rb	Ra	Bitwise OR Rt = Ra| Rb
NAND RT, RA, Rb	0x00	0x0A	0x00	Rt	Rb	Ra	Bitwise NAND Rt = (~Ra)&(~Rb)
NOR RT, RA, RB	0x00	0x0B	0x00	Rt	Rb	Ra	Bitwise NOR Rt = ~(Ra|Rb)
XOR RT, RA, RB	0x00	0x0C	0x00	Rt	Rb	Ra	Bitwise XOR Rt = Ra^Rb

	ARITH.IF Format						
Bitfield	[31-25]	[24-20]	[19-15]	[14-10]	[9-5]	[5-0]	**Bitfield**
FieldName	IMM	FUNC	OPC	RT	RB	RA	**FieldName**
Mnemonics							**Description**
CMP.NE RT, RA, RB	0x00	0x02	0x01	Rt	Rb	Ra	IF(RA != RB) {RT = 0x02} else {RT=0x00}
CMP.EQ RT, RA, RB	0x00	0x03	0x01	Rt	Rb	Ra	IF(RA == RB) {RT = 0x03} else {RT=0x00}
CMP.GT RT, RA, RB	0x00	0x04	0x01	Rt	Rb	Ra	IF(RA > RB) {RT = 0x04} else {RT=0x00}
CMP.LT RT, RA, RB	0x00	0x05	0x01	Rt	Rb	Ra	IF(RA < RB) {RT = 0x05} else {RT=0x00}
CMP.GTE RT, RA, RB	0x00	0x06	0x01	Rt	Rb	Ra	IF(RA >= RB) {RT = 0x06} else {RT=0x00}
CMP.LTE RT, RA, RB	0x00	0x07	0x01	Rt	Rb	Ra	IF(RA <= RB) {RT = 0x07} else {RT=0x00}

	ARITH.IF Format						
Bitfield	[31-25]	[24-20]	[19-15]	[14-10]	[9-5]	[5-0]	**Bitfield**
FieldName	IMM	FUNC	OPC	RT	RB	RA	**FieldName**
Mnemonics							**Description**
LB RT, IMM(RA)	imm	0x00	0x02	Rt	zero	Ra	Load signed byte Rt = addr(RA+IMM)
LH RT, IMM(RA)	imm	0x01	0x02	Rt	zero	Ra	Load signed halfword Rt = addr(RA+IMM)
LW RT, IMM(RA)	imm	0x02	0x02	Rt	zero	Ra	Load signed word Rt = addr(RA+IMM)
LD RT, IMM(RA)	imm	0x03	0x02	Rt	zero	Ra	Load doubleword Rt = addr(RA+IMM)
SB RA, IMM(RT)	imm	0x04	0x02	Rt	zero	Ra	Store signed byte addr(RT+IMM) = RA
SH RA, IMM(RT)	imm	0x05	0x02	Rt	zero	Ra	Store signed halfword addr(RT+IMM) = RA
SW RA, IMM(RT)	imm	0x06	0x02	Rt	zero	Ra	Store signed word addr(RT+IMM) = RA
SD RA, IMM(RT)	imm	0x07	0x02	Rt	zero	Ra	Store doubleword addr(RT+IMM) = RA
LBU RT, IMM(RA)	imm	0x10	0x02	Rt	zero	Ra	Load unsigned byte Rt = addr(RA+IMM)
LHU RT, IMM(RA)	imm	0x11	0x02	Rt	zero	Ra	Load unsigned halfword Rt = addr(RA+IMM)
LWU RT, IMM(RA)	imm	0x12	0x02	Rt	zero	Ra	Load unsigned word Rt = addr(RA+IMM)
SBU RA, IMM(RT)	imm	0x14	0x02	Rt	zero	Ra	Store unsigned byte addr(RT+IMM) = RA
SHU RA, IMM(RT)	imm	0x15	0x02	Rt	zero	Ra	Store unsigned halfword addr(RT+IMM) = RA
SWU RA, IMM(RT)	imm	0x16	0x02	Rt	zero	Ra	Store unsigned word addr(RT+IMM) = RA

	ARITH.IF Format						
Bitfield	[31-25]	[24-20]	[19-15]	[14-10]	[9-5]	[5-0]	**Bitfield**
FieldName	IMM	FUNC	OPC	RT	RB	RA	**FieldName**
Mnemonics							**Description**
NOT RT, RA	0x00	0x0F	0x03	Rt	zero	Ra	Negation; Rt = !Ra; Rb is R0

	ARITH.IF Format						
Bitfield	[31-25]	[24-20]	[19-15]	[14-10]	[9-5]	[5-0]	**Bitfield**
FieldName	IMM	FUNC	OPC	RT	RB	RA	**FieldName**
Mnemonics							**Description**
BRA RT	0x00	0x00	0x04	Rt	zero	zero	Set PC=RT
BR RT	0x00	0x01	0x04	Rt	zero	zero	Set PC=PC+RT (signed arithmetic)

	ReadCtrl.if						
Bitfield	[31-25]	[24-20]	[19-15]	[14-10]	[9-5]	[5-0]	**Bitfield**
FieldName	IMM	FUNC	OPC	RT	RB	RA	**FieldName**
Mnemonics							**Description**
CADD RT, RA, CTRLB	0x00	0x00	0x09	Rt	Rb	Ra	Add Ra + Rb; place result in Rt

	ReadCtrl.if						
Bitfield	[31-25]	[24-20]	[19-15]	[14-10]	[9-5]	[5-0]	**Bitfield**
FieldName	IMM	FUNC	OPC	RT	RB	RA	**FieldName**
Mnemonics							**Description**
BRAC RT, RA, COND	0x00	0x00	0x0A	Rt	Cond	Ra	if(Ra == Cond) {PC=Rt} else {PC+=0x04}
BRC RT, RA, COND	0x00	0x01	0x0A	Rt	Cond	Ra	if(Ra == Cond) {PC+=Rt} else {PC+=0x04}

NOTE: "Cond" is one of {ne, eq, gt, lt, gte, lte}

	WriteCtrl.if						
Bitfield	[31-25]	[24-20]	[19-15]	[14-10]	[9-5]	[5-0]	**Bitfield**
FieldName	IMM	FUNC	OPC	RT	RB	RA	**FieldName**
Mnemonics							**Description**
LADD CTRL, RA, RB	0x00	0x00	0x17	Rt	Rb	Ra	Add Ra + Rb; place result in Rt

	WriteCtrl.if						
Bitfield	[31-25]	[24-20]	[19-15]	[14-10]	[9-5]	[5-0]	**Bitfield**
FieldName	IMM	FUNC	OPC	RT	RB	RA	**FieldName**
Mnemonics							**Description**
BRR RT	0x00	0x00	0x18	Rt	zero	zero	Set PC=RT; RT must be a control register (RP)

	Bitfield	[31-25]	[24-20]	[19-15]	[14-10]	[9-5]	[5-0]
Base Instruction Mnemonic	**Pseudo Instruction Mnemonic**	IMM	FUNC	OPC	RT	RB	RA
MOV RT, RB	ADD RT, R0, RB	0x00	0x00	0x00	Rt	Rb	zero
MOVCG RT,RB	CADD RT, R0, CTRLB	0x00	0x00	0x09	Rt	Rb	zero
MOVGC RT,RA	LADD CTRL, R0, RB	0x00	0x00	0x17	Rt	Rb	zero