CODASIP URISC

Instruction Set Reference

Version: 4.0.0

Release Date: March 31, 2017

INTEGRATED 3RD PARTY SOFTWARE MODULES AND THEIR LICENSES

All integrated 3rd party software licenses are listed in the document "INTEGRATED 3RD PARTY SOFTWARE MODULES AND THEIR LICENSES" on the Codasip Website.

TABLE OF CONTENTS

1 PREFACE

1.1 About

This document describes the overall architecture of the Codasip uRISC processor, its instruction set and instruction format of each instruction. Specification of syntax, semantics, description of functionality, binary encoding and examples are provided.

1.1.1 Intended Audience

This document is intended for developers using Codasip Studio and working with Codasip uRISC processor. This document can be used for developing applications for Codasip uRISC processor architecture without any prior knowledge of CodAL language.

1.1.2 Release Information

1.0.0 — Initial version

1.1.3 Product Revisions

Codasip uRISC 4.0.0

This reference guide can be used with any version of Codasip Studio.

1.1.4 Typographical Conventions

Table 1: Typographical conventions

Convention	Usage	Example
Capitalised	Standardized terms, defined earlier in the text or in the Glossary	Window, Project
Important	Important text	*Do not forget to ...*
Document ref	Reference to other Codasip and non-Codasip documents	Please refer to the *CodAL Language Reference Manual*.
`Code, filenames etc.`	Code, code values, Unix file names, prompts, etc.	File name `ca_utils.hcodal`
`<`*`abstract name`*`>`	Field for substitution with user data.	*`<project>`*`/model`

Convention	Usage	Example
`keyword`	Inline references to CodAL keywords (lower case).	**`element`**, **`event`**
IDE_word	Inline references to keywords of the IDE (usually starts in upper case)	The **Project Explorer** window
Option→Suboption	Command path, typically starting from the main toolbar.	**File → New → CodAL Project**
`Example`	Examples - typically snippets of code.	`register bit[DATA_W] test;`
`Syntax explained`	Explanation of syntax	`StartSection: "start" "{"`

1.2 References

1.2.1 Other Codasip Documents

Here is a complete list of the documentation for Codasip Studio:

Guides:

Document	Description
Codasip Studio Installation Guide	How to install the Codasip Studio software package.
Codasip Studio User Guide	Detailed guidance on the use of Codasip Studio and the tools that it contains.

Reference Manuals:

Document	Description
CodAL Language Reference Manual	A complete presentation of the CodAL language and how to use it for writing ASIP models.
Codasip Studio Technical Reference Manual	Reference information on Codasip Studio and the tools that it contains.
Codasip Program Descrtiption Model Language Manual	A complete presentation of the PDML language and how to use it for writing constraints for random applications generator.

Codasip Studio Message Reference Manual	A list of Codasip errors, warnings and notes that user can encounter during his work with Codasip Studio with descriptions, explanations, and possible solutions.

Tutorials:

Document	Description
Codasip Studio Quick Start Tutorial	A step-by-step introduction to the essentials of Codasip Studio.
Codasip Instruction Accurate Model Tutorial	A step-by-step introduction to writing Instruction Accurate ASIP models in CodAL.
Codasip Compiler Generation Tutorial	A step-by-step introduction to generating a C/C++ compiler from an Instruction Accurate ASIP model written in CodAL.
Codasip Cycle Accurate Model Tutorial	A step-by-step introduction to writing a Cycle Accurate CodAL model.
Codasip Interrupts and Peripherals Tutorial	A step-by-step introduction to adding external devices to an ASIP CodAL model.
Codasip JTAG Extension Tutorial	A step-by-step introduction to Codasip's JTAG extension.
Codasip SIMD Extension Tutorial	Tutorial showing the implementation of SIMD extensions in the Codasip uRISC
Codasip Custom Components Verification Tutorial	Diescribes proccess of adding manually modified UVM test-bench for a component into the ASIP or the top-level UVM test-bench.

1.2.2 Other References

None.

1.3 Feedback

1.3.1 Feedback on Codasip Products

If you have any comments or suggestions about Codasip products, please contact your supplier or send an email to support@codasip.com. Give:

- The product name
- The product revision or version
- An explanation with as much information as you can provide. Include symptoms and diagnostic procedures if appropriate.

1.3.2 Feedback on this Document

If you have comments on this document, please send an email to feedback@codasip.com. Give:

- The document title and format (pdf, web page, etc)
- The chapter number, page numbers and version to which your comments apply
- A concise explanation of your comments.

Codasip also welcomes general suggestions for additions and improvements.

2 INSTRUCTION SET

The instruction set consists of 32-bit instructions. There are several instructions formats depending on the particular group in instruction set. The following sections describe each instruction format in more detail.

2.1 Special Instructions

There are two special instructions in the Codasip uRISC instruction set. Their instruction format is depicted in the following table:

31 24	23 0
OPC	UNUSED
8	24

Field descriptions:

Field	Description
OPC	Operation code of instruction.
UNUSED	Unused bits, filled with zeros.

2.1.1 NOP

Instruction	**NOP**
Op. code	0x00
Syntax	nop
Semantics	-
Latency	1

31 24	23 0
0x00	0x0
8	24

Example	nop
Description	No operation. Used for inserting wait cycles.

2.1.2 HALT

Instruction	**HALT**
Op. code	0x01

Syntax	halt
Semantics	simulation_stop()
Latency	1

31 24	23 0
0x01	0x0
8	24

Example	halt
Description	This instruction stops the simulation.

2.2 Move Instructions

Instruction set contains one unconditional and two conditional instructions for moving data between registers and two instructions for moving 32-bit constant into register.

31 24	23 19	18 14	13 9	8 0
OPC	DST	SRC_1	SRC_2	UNUSED
8	5	5	5	9

Field descriptions:

Field	**Description**
OPC	Operation code of instruction.
DST	Destination GPR.
SRC_1	First source GPR.
SRC_2	Second source GPR.
UNUSED	Unused bits, filled with zeros.

2.2.1 MOV

Instruction	**MOV**
Op. code	0x04
Syntax	DST = mov SRC_2
Semantics	DST $\leftarrow$ SRC_2
Latency	1

31 24	23 19	18 14	13 9	8 0
0x04	DST	0x0	SRC_2	0x0
8	5	5	5	9

Example	r1 = mov r2
Description	Copies value of source SRC_2 to destination DST register.

2.2.2 MOVZ

Instruction	**MOVZ**
Op. code	0x20
Syntax	DST = movz SRC_1, SRC_2
Semantics	if (SRC_1 == 0) DST $\leftarrow$ SRC_2
Latency	1

31 24	23 19	18 14	13 9	8 0
0x20	DST	SRC_1	SRC_2	0x0
8	5	5	5	9

Example	r1 = movz r2, r3
Description	Copies value of source SRC_2 to destination DST register, if SRC_1 equals zero.

2.2.3 MOVNZ

Instruction	**MOVNZ**
Op. code	0x21
Syntax	DST = movnz SRC_1, SRC_2
Semantics	if (SRC_1 != 0) DST $\leftarrow$ SRC_2
Latency	1

31 24	23 19	18 14	13 9	8 0
0x21	DST	SRC_1	SRC_2	0x0
8	5	5	5	9

Example	r1 = movnz r2, r3
Description	Copies value of source SRC_2 to destination DST register, if SRC_1 does not equal zero.

31 24	23 19	18 14	13 0
OPC	SIMM	DST	SIMM
8	5	5	14

Field descriptions:

Field	Description
OPC	Operation code of instruction.
DST	Destination GPR.
SIMM	Signed immediate.

2.2.4 MOVSI

Instruction	**MOVSI**
Op. code	0x02
Syntax	DST = movsi SIMM
Semantics	DST ←SIMM
Latency	1

31 24	**23 19**	**18 14**	**13 0**
0x02	SIMM	DST	SIMM
8	5	5	14

Example	r1 = movsi 2
Description	Moves 19-bit signed immediate operand to DST register.

2.2.5 MOVHI

Instruction	**MOVHI**
Op. code	0x03
Syntax	DST = movhi SIMM
Semantics	DST ← SIMM[15..0] :: DST[15..0]
Latency	1

31 24	**23 19**	**18 14**	**13 0**
0x03	SIMM	DST	SIMM
8	5	5	14

Example	r1 = movhi 2
Description	Moves 16-bit signed immediate operand to top DST register.

2.3 Arithmetic, Logic and Comparison Instructions

There are several instructions performing typical arithmetic, logic and comparison instructions described in the following subsections.

31 24	23 19	18 14	13 9	8 0
OPC	DST	SRC$_1$	SRC$_2$	UNUSED
8	5	5	5	9

Field descriptions:

Field	Description
OPC	Operation code of instruction.
DST	Destination GPR.
SRC$_1$	First source GPR.
SRC$_2$	Second source GPR.
UNUSED	Unused bits, filled with zeros.

2.3.1 ADD

Instruction	**ADD**
Op. code	0x05
Syntax	DST = add SRC$_1$, SRC$_2$
Semantics	DST $\leftarrow$ SRC$_1$ + SRC$_2$
Latency	1

31 24	23 19	18 14	13 9	8 0
0x05	DST	SRC$_1$	SRC$_2$	0x0
8	5	5	5	9

Example	r1 = add r2, r3
Description	Performs addition of values in SRC$_1$ and SRC$_2$ GPR and stores the result in the DST GPR.

2.3.2 SUB

Instruction	**SUB**
Op. code	0x06
Syntax	DST = sub SRC$_1$, SRC$_2$
Semantics	DST $\leftarrow$ SRC$_1$ - SRC$_2$
Latency	1

31 24	23 19	18 14	13 9	8 0
0x06	DST	SRC_1	SRC_2	0x0
8	5	5	5	9

Example	r1 = sub r2, r3
Description	Performs subtraction of values in SRC_1 and SRC_2 GPR and stores the result in the DST GPR.

2.3.3 MUL

Instruction	**MUL**
Op. code	0x07
Syntax	DST = mul SRC_1, SRC_2
Semantics	MUL $\leftarrow SRC_1 * SRC_2$
Latency	1

31 24	23 19	18 14	13 9	8 0
0x07	DST	SRC_1	SRC_2	0x0
8	5	5	5	9

Example	r1 = mul r2, r3
Description	Performs multiplication of values in SRC_1 and SRC_2 GPR and stores the result in the DST GPR. The result is truncated to 32 bits.

2.3.4 AND

Instruction	**AND**
Op. code	0x08
Syntax	DST = and SRC_1, SRC_2
Semantics	DST $\leftarrow SRC_1$ & SRC_2
Latency	1

31 24	23 19	18 14	13 9	8 0
0x08	DST	SRC_1	SRC_2	0x0
8	5	5	5	9

Example	r1 = and r2, r3
Description	Performs logical AND of values in SRC_1 and SRC_2 GPR and stores the result in the DST GPR.

2.3.5 OR

Instruction	**OR**
Op. code	0x09
Syntax	DST = or SRC_1, SRC_2
Semantics	DST $\leftarrow$ SRC_1 | SRC_2
Latency	1

31 24	23 19	18 14	13 9	8 0
0x9	DST	SRC_1	SRC_2	0x0
8	5	5	5	9

Example	r1 = or r2, r3
Description	Performs logical OR of values in SRC_1 and SRC_2 GPR and stores the result in the DST GPR.

2.3.6 XOR

Instruction	**XOR**
Op. code	0x0A
Syntax	DST= xor SRC_1, SRC_2
Semantics	DST $\leftarrow$ SRC_1 ^ SRC_2
Latency	1

31 24	23 19	18 14	13 9	8 0
0x0A	DST	SRC_1	SRC_2	0x0
8	5	5	5	9

Example	r1 = xor r2, r3
Description	Performs logical exclusive-OR of values in SRC_1 and SRC_2 GPR and stores the result in the DST GPR.

2.3.7 SLL

Instruction	**SLL**
Op. code	0x0B
Syntax	DST= sll SRC_1, SRC_2
Semantics	DST ← SRC_1 << SRC_2[4..0]
Latency	1

31 24	23 19	18 14	13 9	8 0
0x0B	DST	SRC_1	SRC_2	0x0
8	5	5	5	9

Example	r1 = sll r2, r3
Description	Performs logical shift left operation of value in SRC_1, shift length is stored in SRC_2 GPR. Result is stored in the DST GPR.

2.3.8 SRL

Instruction	**SRL**
Op. code	0x0C
Syntax	DST = srl SRC_1, SRC_2
Semantics	DST ← SRC_1 (u) >> SRC_2[4..0]
Latency	1

31 24	23 19	18 14	13 9	8 0
0x0C	DST	SRC_1	SRC_2	0x0
8	5	5	5	9

Example	r1 = srl r2, r3
Description	Performs logical shift right operation of value in SRC_1 with no sign extension, shift length is stored in SRC_2 GPR. Result is stored in the DST GPR.

2.3.9 SRA

Instruction	**SRA**
Op. code	0x0D

Syntax	DST = sra SRC$_1$, SRC$_2$
Semantics	DST ← SRC$_1$ (s) >> SRC$_2$[4..0]
Latency	1

31 24	23 19	18 14	13 9	8 0
0x0D	DST	SRC$_1$	SRC$_2$	0x0
8	5	5	5	9

Example	r1 = sra r2, r3
Description	Performs arithmetic shift right operation of value in SRC$_1$ with sign extension, shift length is stored in SRC$_2$ GPR. Result is stored in the DST GPR.

2.3.10 EQ

Instruction	**EQ**
Op. code	0x1A
Syntax	DST = eq SRC$_1$, SRC$_2$
Semantics	DST ← (SRC$_1$ == SRC$_2$)
Latency	1

31 24	23 19	18 14	13 9	8 0
0x1A	DST	SRC$_1$	SRC$_2$	0x0
8	5	5	5	9

Example	r1 = eq r2, r3
Description	Sets value of DST GPR to non-zero when SRC$_1$ and SRC$_2$ GPR are equal, otherwise DST GPR is set to zero value.

2.3.11 NEQ

Instruction	**NEQ**
Op. code	0x1B
Syntax	DST= neq SRC$_1$,SRC$_2$
Semantics	DST ← (SRC$_1$!= SRC$_2$)
Latency	1

31 24	23 19	18 14	13 9	8 0
0x1B	DST	SRC$_1$	SRC$_2$	0x0
8	5	5	5	9

Example	r1 = neq r2, r3
Description	Sets value of DST GPR to non-zero when SRC$_1$ and SRC$_2$ GPR are not equal, otherwise DST GPR is set to zero value.

2.3.12 SLT

Instruction	**SLT**
Op. code	0x1C
Syntax	DST= slt SRC$_1$,SRC$_2$
Semantics	DST $\leftarrow$ (SRC$_1$ (s)< SRC$_2$)
Latency	1

31 24	23 19	18 14	13 9	8 0
0x1C	DST	SRC$_1$	SRC$_2$	0x0
8	5	5	5	9

Example	r1 = slt r2, r3
Description	Performs signed comparison of values stored in SRC$_1$ GPR and SRC$_2$ GPR. If value in SRC$_1$ GPR is lower than value stored in SRC$_2$ GPR, DST GPR is set to non-zero value. Otherwise it is set to zero.

2.3.13 ULT

Instruction	**ULT**
Op. code	0x1D
Syntax	DST= ult SRC$_1$,SRC$_2$
Semantics	DST $\leftarrow$ (SRC$_1$ (u)< SRC$_2$)
Latency	1

31 24	23 19	18 14	13 9	8 0
0x1D	DST	SRC$_1$	SRC$_2$	0x0
8	5	5	5	9

Example	r1 = ult r2, r3
Description	Performs unsigned comparison of values stored in SRC_1 GPR and SRC_2 GPR. If value in SRC_1 GPR is lower than value stored in SRC_2 GPR, DST GPR is set to non-zero value. Otherwise it is set to zero.

2.3.14 SLE

Instruction	**SLE**
Op. code	0x1E
Syntax	DST = sle SRC_1,SRC_2
Semantics	DST $\leftarrow$ (SRC_1 (s)≤ SRC_2)
Latency	1

31 24	23 19	18 14	13 9	8 0
0x1E	DST	SRC_1	SRC_2	0x0
8	5	5	5	9

Example	r1 = sle r2, r3
Description	Performs signed comparison of values stored in SRC_1 GPR and SRC_2 GPR. If value in SRC_1 GPR is lower than or equal to the value stored in SRC_2 GPR, DST GPR is set to non-zero value. Otherwise it is set to zero.

2.3.15 ULE

Instruction	**ULE**
Op. code	0x1F
Syntax	DST= ule SRC_1,SRC_2
Semantics	DST $\leftarrow$ (SRC_1 (u)≤ SRC_2)
Latency	1

31 24	23 19	18 14	13 9	8 0
0x1F	DST	SRC_1	SRC_2	0x0
8	5	5	5	9

Example	r1 = ule r2, r3
Description	Performs unsigned comparison of values stored in SRC_1 GPR and

SRC$_2$ GPR. If value in SRC$_1$ GPR is lower than or equal to the value stored in SRC$_2$ GPR, DST GPR is set to non-zero value. Otherwise it is set to zero.

2.3.16 ADDI

Instruction	**ADDI**
Op. code	0x24
Syntax	DST = addi SRC$_1$, SIMM
Semantics	DST $\leftarrow$ SRC$_1$ + (int19)SIMM
Latency	1

31 24	23 19	18 14	13 9	8 0
0x24	SIMM	SRC$_1$	DST	SIMM
8	5	5	5	9

Example	r1 = addi r2, 3
Description	19-bit immediate value is sign extended to 32 bits and the result is added with the content of register SRC GPR. Result is written into the DST GPR.

2.4 Load Instructions

This group contains instructions used to load words, halfwords or bytes of data from memory. Load instructions must access aligned addresses when a block larger than one byte is accessed.

31 24	23 19	18 14	13 9	8 0
OPC	SIMM	SRC$_1$	DST	SIMM
8	5	5	5	9

Field descriptions:

Field	Description
OPC	Operation code of instruction.
DST	Destination GPR.
SRC$_1$	Value of source GPR is used as base address.
SIMM	Signed immediate.

2.4.1 LD

Instruction	**LD**
Op. code	0x0E
Syntax	DST = ld [SRC_1+ SIMM]
Semantics	DST ← mem[SRC_1+ (int14)SIMM]
Latency	2

31 24	23 19	18 14	13 9	8 0
0x0E	SIMM	SRC_1	DST	SIMM
8	5	5	5	9

Example	r1 = ld [r2 + 0]
Description	This instruction loads a 32-bit word from memory. The access address must be aligned to 4 bytes, i.e. the lowest 2 bits of must be zeros.

2.4.2 LDHU

Instruction	**LDHU**
Op. code	0x10
Syntax	DST = ldhu [SRC_1+ SIMM]
Semantics	DST ← mem[SRC_1+ (int14)SIMM].subblock(0, 2)
Latency	2

31 24	23 19	18 14	13 9	8 0
0x10	SIMM	SRC_1	DST	SIMM
8	5	5	5	9

Example	r1 = ldhu [r2 + 0]
Description	This instruction loads an unsigned half-word from memory. The access address must be aligned to 2 bytes, i.e. the lowest bit of address must be zero.

2.4.3 LDHS

Instruction	**LDHS**
Op. code	0x0F

Syntax	DST = ldhs [SRC_1+ SIMM]
Semantics	DST ← (int32)mem[SRC_1+ (int14)SIMM].subblock(0, 2)
Latency	2

31 24	23 19	18 14	13 9	8 0
0x0F	SIMM	SRC_1	DST	SIMM
8	5	5	5	9

Example	r1 = ldhs [r2 + 0]
Description	This instruction loads a signed half-word from memory. The access address must be aligned to 2 bytes, i.e. the lowest bit of address must be zero.

2.4.4 LDBU

Instruction	**LDBU**
Op. code	0x12
Syntax	DST = ldbu [SRC_1+ SIMM]
Semantics	DST ← mem[SRC_1+ (int14)SIMM].subblock(0, 1)
Latency	2

31 24	23 19	18 14	13 9	8 0
0x18	SIMM	SRC_1	DST	SIMM
8	5	5	5	9

Example	r1 = ldbu [r2 + 0]
Description	This instruction loads an unsigned byte from memory. The access address does not have to be aligned.

2.4.5 LDBS

Instruction	**LDBS**
Op. code	0x13
Syntax	DST = ldbs [SRC_1+ SIMM]
Semantics	DST ← (int32)mem[SRC_1+ (int14)SIMM].subblock(0, 1)
Latency	2

31 24	23 19	18 14	13 9	8 0
0x13	SIMM	SRC_1	DST	SIMM
8	5	5	5	9

Example	r1 = ldbs [r2 + 0]
Description	This instruction loads a signed byte from memory. The access address does not have to be aligned.

2.5 Store Instructions

This group contains instructions used to store words, half-words or bytes of data to memory.

31 24	23 19	18 14	13 9	8 0
OPC	SIMM	SRC_1	SRC_2	SIMM
8	5	5	5	9

Field descriptions:

Field	Description
OPC	Operation code of instruction.
SRC_1	Source GPR used as base address.
SRC_2	Source GPR from which the value is stored to the memory.
SIMM	Signed immediate.

2.5.1 ST

Instruction	**ST**
Op. code	0x13
Syntax	st SRC_2, [SRC_1 + SIMM]
Semantics	mem[SRC_1+ (int14)SIMM] $\leftarrow$ SRC_2
Latency	1

31 24	23 19	18 14	13 9	8 0
0x13	SIMM	SRC_1	SRC_2	SIMM
8	5	5	5	9

Example	st r1, [r2 + 0]

Description	This instruction stores a word to memory. The access address must be aligned to 4 bytes, i.e. the lowest 2 bits of access address must be zero.

2.5.2 STB

Instruction	**STB**
Op. code	0x15
Syntax	stb SRC_2, [SRC_1 + SIMM]
Semantics	mem[SRC_1+ (int14)SIMM].subblock(0, 1)$\leftarrow$ (uint8)SRC_2
Latency	1

31 24	23 19	18 14	13 9	8 0
0x13	SIMM	SRC_1	SRC_2	SIMM
8	5	5	5	9

Example	stb r1, [r2 + 0]
Description	This instruction stores a byte to memory. The access address does not have to be aligned.

2.5.3 STH

Instruction	**STH**
Op. code	0x14
Syntax	sth SRC_2, [SRC_1 + SIMM]
Semantics	mem[SRC_1+ (int14)SIMM].subblock(0, 2)$\leftarrow$ (uint16)SRC_2
Latency	1

31 24	23 19	18 14	13 9	8 0
0x14	SIMM	SRC_1	SRC_2	SIMM
8	5	5	5	9

Example	sth r1, [r2 + 0]
Description	This instruction stores a half-word to memory. The access address must be aligned to 2 bytes, i.e. the lowest bit of address must be zero.

2.6 Jump and Call instructions

Instructions in this group are used to modify the control flow of the program. These instructions use immediate operands as absolute addresses to modify the content of the program counter.

31 24	23 0
OPC	ABS_ADDR
8	24

Field descriptions:

Field	Description
OPC	Operation code of instruction.
IMM	Unsigned immediate.

2.6.1 JUMP

Instruction	**JUMP**
Op. code	0x16
Syntax	jump
Semantics	PC ← (uint24)ABS_ADDR
Latency	1

31 24	23 0
0x16	ABS_ADDR
8	24

Example	jump $label
Description	New program counter value is set to an absolute address.

2.6.2 CALL

Instruction	**CALL**
Op. code	0x17
Syntax	call ABS_ADDR

Semantics	R3 ← PC + 4;
	PC ← (uint24)ABS_ADDR

Latency	1

31 24	23 0
0x17	ABS_ADDR
8	24

Example call $main

Description Return address is stored to GPR R3. Program counter is set to absolute address and the next instruction to be executed will be fetched from the updated address in the program counter.

31 24	23 14	13 9	8 0
OPC	UNUSED	SRC_2	UNUSED
8	10	5	9

Field descriptions:

Field	Description
OPC	Operation code of instruction.
SRC_2	Source GPR.
UNUSED	Unused bits, filled with zeros.

2.6.3 JUMP

Instruction	**JUMP**
Op. code	0x18
Syntax	jump SRC_2
Semantics	PC $\leftarrow$ SRC_2
Latency	1

31 24	23 14	13 9	8 0
0x18	0x0	SRC	0x0
8	10	5	16

Example jump r1

Description Program counter is set to address stored in register SRC GPR.

2.6.4 CALL

Instruction	**CALL**
Op. code	0x19
Syntax	call SRC_2
Semantics	$R31 \leftarrow PC + 4;$ $PC \leftarrow SRC_2$
Latency	1

31 … 24	23 … 14	13 … 9	8 … 0
0x19	0x0	SRC_2	0x0
8	10	5	16

Example	call r1
Description	Return address is stored to GPR R3. Program counter is set to address stored in SRC_2 GPR and the next instruction to be executed will be fetched from the updated address in the program counter.

31 … 24	23 … 19	18 … 14	13 … 9	8 … 0
OPC	REL_ADDR	SRC_1	UNUSED	REL_ADDR
8	5	5	5	9

Field descriptions:

Field	Description
OPC	Operation code of instruction.
SRC_1	Condition GPR.
UNUSED	Unused bits, filled with zeros.
REL_ADDR	Relative address of jump destination. A signed value of the address of label is read by the assembler. Value of program counter is subtracted and the result is stored in binary coding.

2.6.5 JUMPZ

Instruction	**JUMPZ**
Op. code	0x22
Syntax	jumpz SRC_1, REL_ADDR
Semantics	if $(SRC_1 == 0)\ PC \leftarrow PC + (int14)REL\_ADDR$

Latency 1

31 24	23 19	18 14	13 9	8 0
0x22	REL_ADDR	SRC_1	UNUSED	REL_ADDR
8	5	5	5	9

Example	jumpz r1, $label
Description	If the value of GPR SRC_1 equals zero, the value of relative address is added to current program counter value incremented by 4 and jump is performed. If the condition is not met the program counter is incremented to address of next instruction following the conditional JUMPZ instruction. When instruction is fetched the program counter is immediately incremented to address of the next instruction.

2.6.6 JUMPNZ

Instruction	**JUMPNZ**
Op. code	0x23
Syntax	jumpnz SRC_1, REL_ADDR16
Semantics	if (SRC_1!= 0) PC $\leftarrow$ PC + (int14)REL_ADDR\
Latency	1

31 24	23 19	18 14	13 9	8 0
0x23	REL_ADDR	SRC_1	UNUSED	REL_ADDR
8	5	5	5	9

Example	jumpnz r1, $label
Description	If the value of GPR SRC_1 is other than to zero, the value of relative address is added to the current program counter value incremented by 4 and jump is performed. If the condition is not met the program counter is incremented to the address of the next instruction following the conditional JUMPNZ instruction. When instruction is fetched the program counter is immediately incremented to address of the next instruction.

3 INSTRUCTION SET LISTINGS

OPCODE	INSTRUCTION PARAMETERS				SYNTAX
0x00	UNUSED:24				nop
0x01	UNUSED:24				halt
0x02	SIMM:5	DST:5	SIMM:14		DST = movsi SIMM
0x03	SIMM:5	DST:5	SIMM:14		DST = movhi SIMM
0x04	DST:5	UNUSED:5	SRC$_2$:5	UNUSED:9	DST = mov SRC$_2$
0x05	DST:5	SRC$_1$:5	SRC$_2$:5	UNUSED:9	DST = add SRC$_1$, SRC$_2$
0x06	DST:5	SRC$_1$:5	SRC$_2$:5	UNUSED:9	DST = sub SRC$_1$, SRC$_2$
0x07	DST:5	SRC$_1$:5	SRC$_2$:5	UNUSED:9	DST = mul SRC$_1$, SRC$_2$
0x08	DST:5	SRC$_1$:5	SRC$_2$:5	UNUSED:9	DST = and SRC$_1$, SRC$_2$
0x09	DST:5	SRC$_1$:5	SRC$_2$:5	UNUSED:9	DST = or SRC$_1$, SRC$_2$
0x0A	DST:5	SRC$_1$:5	SRC$_2$:5	UNUSED:9	DST = xor SRC$_1$, SRC$_2$
0x0B	DST:5	SRC$_1$:5	SRC$_2$:5	UNUSED:9	DST = sll SRC$_1$, SRC$_2$
0x0C	DST:5	SRC$_1$:5	SRC$_2$:5	UNUSED:9	DST = srl SRC$_1$, SRC$_2$
0x0D	DST:5	SRC$_1$:5	SRC$_2$:5	UNUSED:9	DST = sra SRC$_1$, SRC$_2$
0x0E	SIMM:5	SRC$_1$:5	DST:5	SIMM:9	DST = ld [SRC$_1$+SIMM]
0x0F	SIMM:5	SRC$_1$:5	DST:5	SIMM:9	DST = ldhs [SRC$_1$+SIMM]
0x10	SIMM:5	SRC$_1$:5	DST:5	SIMM:9	DST = ldhu [SRC$_1$+SIMM]
0x11	SIMM:5	SRC$_1$:5	DST:5	SIMM:9	DST = ldbs [SRC$_1$+SIMM]
0x12	SIMM:5	SRC$_1$:5	DST:5	SIMM:9	DST = ldbu [SRC$_1$+SIMM]
0x13	SIMM:5	SRC$_1$:5	SRC$_2$:5	SIMM:9	st SRC$_2$, [SRC$_1$+SIMM]
0x14	SIMM:5	SRC$_1$:5	SRC$_2$:5	SIMM:9	sth SRC$_2$, [SRC$_1$+SIMM]
0x15	SIMM:5	SRC$_1$:5	SRC$_2$:5	SIMM:9	stb SRC$_2$, [SRC$_1$+SIMM]
0x16	ABS_ADDR:24				jump ABS_ADDR
0x17	ABS_ADDR:24				call ABS_ADDR
0x18	UNUSED:10		SRC$_2$:5	UNUSED:9	jump SRC$_2$
0x19	UNUSED:10		SRC$_2$:5	UNUSED:9	call SRC$_2$
0x1A	DST:5	SRC$_1$:5	SRC$_2$:5	UNUSED:9	DST = eq SRC$_1$, SRC$_2$
0x1B	DST:5	SRC$_1$:5	SRC$_2$:5	UNUSED:9	DST = neq SRC$_1$, SRC$_2$
0x1C	DST:5	SRC$_1$:5	SRC$_2$:5	UNUSED:9	DST = slt SRC$_1$, SRC$_2$
0x1D	DST:5	SRC$_1$:5	SRC$_2$:5	UNUSED:9	DST = ult SRC$_1$, SRC$_2$
0x1E	DST:5	SRC$_1$:5	SRC$_2$:5	UNUSED:9	DST = sle SRC$_1$, SRC$_2$
0x1F	DST:5	SRC$_1$:5	SRC$_2$:5	UNUSED:9	DST = ule SRC$_1$, SRC$_2$
0x20	DST:5	SRC$_1$:5	SRC$_2$:5	UNUSED:9	DST = movz SRC$_1$, SRC$_2$

OPCODE	INSTRUCTION PARAMETERS				SYNTAX
0x21	DST:5	SRC_1:5	SRC_2:5	UNUSED:9	DST = movnz SRC_1, SRC_2
0x22	REL_ADDR:5	SRC_1:5	UNUSED:5	REL_ADDR:9	jumpz SRC_1, REL_ADDR
0x23	REL_ADDR:5	SRC_1:5	UNUSED:5	REL_ADDR:9	jumpnz SRC_1, REL_ADDR
0x24	SIMM:5	SRC_1:5	DST:5	SIMM:9	DST = addi SRC_1, SIMM

4 ANNEX: TABLES, EXAMPLES AND FIGURES

4.1 List of Tables

4.2 List of Examples

4.3 List of Figures