ISA Specification

James Fletcher

December 13, 2020

Contents

Abstract

1 Summary

A fictitious RISC Store-Load 32-bit CPU architecture specification for an out-of-order, speculative and superscalar CPU simulator. The CPU has 16 general-purpose 32-bit registers. The instruction set is based upon RISC-V and MIPS. Register 0 is wired to zero value and any writes to register 0 will be silently ignored.

2 Instruction Formats

There are 3 different formats of instructions, all 32 bits in size, as outlined below. All have a 7 bit opcode field placed at the MSB [1]. A key is also provided to describe field names.

[1]MSB - Most Significant Byte

2.1 Field Key

Field Name	Description
Op	6 Operation
Src1	Source 1 Register
Src2	Source 2 Register
Dst	Destination Register
Variable	Depends on operation
Constant	A constant value

2.2 Instruction Format 1 - A

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Op	0	0	0	0	0	0	0	0	0	0	0	0	Dst	Src1	Src2

2.3 Instruction Format 2 - B

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Op	Variable	Src1	Src2

2.4 Instruction Format 3 - C

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Op	Constant	Dst

3 Instruction List

The table below gives an overview of the instructions supported by the processor as well as an example of what the instruction would look like. Further on, each instruction's pseudo-code implementation is provided to help aid understanding.

Op	Instruction	I Form	Example	Description
Load Instructions - ”Var” is memory offset constant				
0h	lw Src1, Src2, Var	B	lw r1, r2, 0	Load Word
1h	lwu Src1, Src2, Var	B	lwu r1, r2, 8	Load Word Unsigned
2h	lb Src1, Src2, Var	B	lb r1, r2, -4	Load Byte
3h	lbu Src1, Src2, Var	B	lbu r3, r2, 4	Load Byte Unsigned
4h	lh Src1, Src2, Var	B	lh r1, r15, 2	Load Half-Word
5h	lhu Src1, Src2, Var	B	lhu r5, r8, 0	Load Half-Word Unsigned
Store Instructions - ”Var” is memory offset constant				
6h	sw Src1, Src2, Var	B	sw r2, r1, -4	Store Word
7h	sb Src1, Src2, Var	B	sb r2, r1, 2	Store Byte
8h	sh Src1, Src2, Var	B	sh r2, r1, 0	Load Half-Word
Load Immediate Instructions				
9h	ldi Src1, Const	C	ldi r1, 1024	Load const to reg
Ah	ldhi Src1, Const	C	ldhi r3, 0	Load const to upper 11-bits of reg

Comparison Instructions - Var is a const value				
Bh	slt Dst, Src1, Src2	A	slt r5, r8, r9	Set Dst if Src1 $<$ Src2
Ch	sltu Dst, Src1, Src2	A	sltu r5, r8, r9	Set Dst if Src1 $<$ Src2
Dh	slti Src1, Src2, Var	B	slti r5, r8, -2	Set Src1 if Src2 $<$ Var
Eh	sltiu Src1, Src2, Var	B	sltiu r5, r8, 5	Set Src1 if Src2 $<$ Var
Control Flow Instructions				
Fh	beq Src1, Src2, Var	B	beq r5, r8, -2	Branch if Src1 = Src2 to PC += Var
10h	bne Src1, Src2, Var	B	bne r5, r8, -2	Branch if Src1 ! = Src2 to PC += Var
11h	blt Src1, Src2, Var	B	blt r5, r8, -2	Branch if Src1 $<$ Src2 to PC += Var
12h	bge Src1, Src2, Var	B	bge r5, r8, -2	Branch if Src1 $>$ Src2 to PC += Var
13h	bltu Src1, Src2, Var	B	bltu r5, r8, -2	Branch if Src1 $<$ Src2 to PC += Var
14h	bgeu Src1, Src2, Var	B	bgeu r5, r8, -2	Branch if Src1 $>$ Src2 to PC += Var
15h	jal Dst, Var	C	jal r5, r8, -2	Dst = PC + 4, PC += Var
16h	jalr Src1, Src2, Var	B	jalr r5, r8, -2	Src1 = PC + 4, PC = Src2 + Var
17h	j Const	C	j label	PC += Const
18h	jr Dst, Var	B	jr r2, 0	PC = Dst + Const
Arithmetic Instructions				
19h	add Dst, Src1, Src2	A	jr r2, 0	PC = Dst + Const
1Ah	addi Dst, Src1, Const	B	jr r2, 0	PC = Dst + Const
1Bh	sub Dst, Src1, Src2	A	jr r2, 0	PC = Dst + Const
1Ch	subi Dst, Src1, Const	B	jr r2, 0	PC = Dst + Const
1Dh	mul Dst, Src1, Src2	A	jr r2, 0	PC = Dst + Const
1Eh	mulh Dst, Src1, Src2	A	jr r2, 0	PC = Dst + Const
1Fh	mulhsu Dst, Src1, Src2	A	jr r2, 0	PC = Dst + Const
20h	mulhu Dst, Src1, Src2	A	jr r2, 0	PC = Dst + Const
21h	div Dst, Src1, Src2	A	jr r2, 0	PC = Dst + Const
22h	divu Dst, Src1, Src2	A	jr r2, 0	PC = Dst + Const
23h	rem Dst, Src1, Src2	A	jr r2, 0	PC = Dst + Const
24h	remu Dst, Src1, Src2	A	jr r2, 0	PC = Dst + Const
Logic Instructions				
25h	and Dst, Src1, Src2	A	jr r2, 0	PC = Dst + Const
26h	or Dst, Src1, Src2	A	jr r2, 0	PC = Dst + Const
27h	xor Dst, Src1, Src2	A	jr r2, 0	PC = Dst + Const
28h	andi Dst, Src1, Const	B	jr r2, 0	PC = Dst + Const
29h	ori Dst, Src1, Const	B	jr r2, 0	PC = Dst + Const
2Ah	xori Dst, Src1, Const	B	jr r2, 0	PC = Dst + Const
2Bh	srl Dst, Src1, Src2	A	jr r2, 0	PC = Dst + Const
2Ch	srli Dst, Src1, Const	B	jr r2, 0	PC = Dst + Const
2Dh	sll Dst, Src1, Src2	A	jr r2, 0	PC = Dst + Const
2Eh	slli Dst, Src1, Const	B	jr r2, 0	PC = Dst + Const
2Fh	srai Dst, Src1, Const	B	jr r2, 0	PC = Dst + Const
30h	sra Dst, Src1, Src2	A	jr r2, 0	PC = Dst + Const
Experimental Instructions				

XXh	pushb Dst, Src1	A	cpy r1, r2, r3	mem[Dst] = Src1[0:7], Dst+=1
XXh	pushh Dst, Src1	A	cpy r1, r2, r3	mem[Dst] = Src1[0:15], Dst+=2
XXh	push Dst, Src1	A	cpy r1, r2, r3	mem[Dst] = Src1, Dst+=4
XXh	popb Dst, Src1	A	cpy r1, r2, r3	Dst-=1, Src1[0:7]=mem[Dst]
XXh	poph Dst, Src1	A	cpy r1, r2, r3	Dst-=2, Src1[0:15]=mem[Dst]
XXh	pop Dst, Src1	A	cpy r1, r2, r3	Dst-=4, Src1=mem[Dst]

4 Memory

This defines how memory is interacted with and is currently not decided on. Total Store Ordering? Weak Memory Model? Strong Memory Model?