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

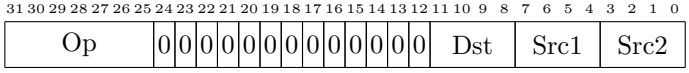
There are 3 different formats of instructions, all 32 bits in size, as outlined below. All have a 7 bit opcode field placed at the MSB [1]. A key is also provided to describe field names. The instruction format isn't optimal however it is easy to encode and understand which is the primary objective.

[1] MSB - Most Significant Byte

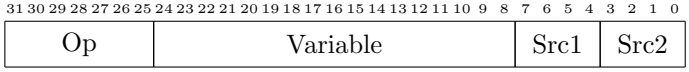
2.1 Field Key

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

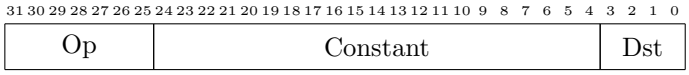
2.2 Instruction Format 1 - A



2.3 Instruction Format 2 - B



2.4 Instruction Format 3 - C



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. The cycle count for each instruction may not be 100% accurate due to external events like branch misses, memory latency and other quirks of instructions, like multiplying by zero. Instruction names that end in a "u" are unsigned operations else they are assumed signed.

Op	Instruction	C	I Form	Example	Description
Load Instructions - "Var" is memory offset constant					
0h	lw Src1, Src2, Var	1	B	lw r1, r2, 0	Load Word
2h	lb Src1, Src2, Var	1	B	lb r1, r2, -4	Load Byte
3h	lbu Src1, Src2, Var	1	B	lbu r3, r2, 4	Load Byte Unsigned
4h	lh Src1, Src2, Var	1	B	lh r1, r15, 2	Load Half-Word
5h	lhu Src1, Src2, Var	1	B	lhu r5, r8, 0	Load Half-Word Unsigned
Store Instructions - "Var" is memory offset constant					
6h	sw Src1, Src2, Var	1	B	sw r2, r1, -4	Store Word
7h	sb Src1, Src2, Var	1	B	sb r2, r1, 2	Store Byte
8h	sh Src1, Src2, Var	1	B	sh r2, r1, 0	Load Half-Word

Load Immediate Instructions					
9h	ldi Src1, Const	1	C	ldi r1, 1024	Load const to reg
Ah	ldhi Src1, Const	1	C	ldhi r3, 0	Load const to upper 11-bits of reg
Comparison Instructions - Var is a const value					
Bh	slt Dst, Src1, Src2	2	A	slt r5, r8, r9	Set Dst if Src1 < Src2
Ch	sltu Dst, Src1, Src2	2	A	sltu r5, r8, r9	Set Dst if Src1 < Src2
Dh	slti Src1, Src2, Var	2	B	slti r5, r8, -2	Set Src1 if Src2 < Var
Eh	sltiu Src1, Src2, Var	2	B	sltiu r5, r8, 5	Set Src1 if Src2 < Var
Control Flow Instructions					
Fh	beq Src1, Src2, Var	3	B	beq r5, r8, -2	Branch if Src1 = Src2 to PC += Var
10h	bne Src1, Src2, Var	3	B	bne r5, r8, -2	Branch if Src1 ! = Src2 to PC += Var
11h	blt Src1, Src2, Var	3	B	blt r5, r8, -2	Branch if Src1 < Src2 to PC += Var
12h	bge Src1, Src2, Var	3	B	bge r5, r8, -2	Branch if Src1 > Src2 to PC += Var
13h	bltu Src1, Src2, Var	3	B	bltu r5, r8, -2	Branch if Src1 < Src2 to PC += Var
14h	bgeu Src1, Src2, Var	3	B	bgeu r5, r8, -2	Branch if Src1 > Src2 to PC += Var
15h	jal Dst, Var	2	C	jal r5, r8, -2	Dst = PC + 4, PC += Var
16h	jalr Src1, Src2, Var	3	B	jalr r5, r8, -2	Src1 = PC + 4, PC = Src2 + Var
17h	j Const	1	C	j label	PC += Const
18h	jr Dst, Var	2	B	jr r2, 0	PC = Dst + Const
Arithmetic Instructions					
19h	add Dst, Src1, Src2	1	A	add r1, r2, r3	Dst = Src1+Src2
1Ah	addi Dst, Src1, Const	1	B	addi r1, r2, 3	Dst = Src1+Const
1Bh	sub Dst, Src1, Src2	1	A	sub r1, r2, r3	Dst=Src1-Src2
1Ch	subi Dst, Src1, Const	1	B	subi r1, r2, 2	Dst=Src1-Const
1Dh	mul Dst, Src1, Src2	7	A	mul r1, r2, r3	Dst = Src1*Src2
1Eh	mulh Dst, Src1, Src2	7	A	mulh r1, r2, r3	Dst = (Src1*Src2)>>32
1Fh	mulhsu Dst, Src1, Src2	7	A	mulhsu r1, r2, r3	Dst = (Src1*Src2)>>32
20h	mulhu Dst, Src1, Src2	7	A	mulhu r1, r2, r3	Dst = (Src1*Src2)>>32
21h	div Dst, Src1, Src2	12	A	div r1, r2, r3	Dst = Src1/Src2
22h	divu Dst, Src1, Src2	12	A	divu r1, r2, r3	Dst = Src1/Src2
23h	rem Dst, Src1, Src2	12	A	rem r1, r2, r3	Dst = Src1%Src2
24h	remu Dst, Src1, Src2	12	A	remu r1, r2, r3	Dst = Src1%Src2
Logic Instructions					
25h	and Dst, Src1, Src2	1	A	and r1, r2, r3	Dst = Src1&Src2
26h	or Dst, Src1, Src2	1	A	or r1, r2, r3	Dst = Src1|Src2

27h	xor Dst, Src1, Src2	1	A	xor r1, r2, r3	Dst = Src1^Src2
28h	andi Dst, Src1, Const	1	B	andi r1, r2, 1	Dst = Src1&Const
29h	ori Dst, Src1, Const	1	B	ori r1, r2, 2	Dst = Src1|Const
2Ah	xori Dst, Src1, Const	1	B	xori r1, r2, r3	Dst = Src1^Const
2Bh	srl Dst, Src1, Src2	1	A	srl r1, r2, r3	Dst = Src1>>Src2
2Ch	srli Dst, Src1, Const	1	B	srli r1, r2, 31	Dst = Src1>>Const
2Dh	sll Dst, Src1, Src2	1	A	sll r1, r2, r3	Dst = Src1<<Src2
2Eh	slli Dst, Src1, Const	1	B	slli r1, r2, 2	Dst = Src1<<Const
2Fh	srai Dst, Src1, Const	1	B	srai r1, r2, 1	Dst = Src1<<<Const
30h	sra Dst, Src1, Src2	1	A	sra r1, r2, r3	Dst = Src1<<<Src2
Experimental Instructions					
XXh	pushb Dst, Src1	2	A	pushb r1, r2	mem[Dst] = Src1[0:7], Dst+=1
XXh	pushh Dst, Src1	2	A	pushh r1, r2	mem[Dst] = Src1[0:15], Dst+=2
XXh	push Dst, Src1	2	A	push r1, r2	mem[Dst] = Src1, Dst+=4
XXh	popb Dst, Src1	2	A	popb r1, r2	Dst-=1, Src1[0:7]=mem[Dst]
XXh	poph Dst, Src1	2	A	poph r1, r2	Dst-=2, Src1[0:15]=mem[Dst]
XXh	pop Dst, Src1	2	A	pop r1, r2	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?