Simulator

Chetan Mistry

October 21, 2019

Contents

1 About

The Simulator is modelled off the MIPS ISA. As such it has a similar Instruction Set which is described below.

1.1 Register Table

Continuing to model off the MIPS ISA, the register table is the same as MIPS which is described below:

Register Number	Conventional Name	Usage
$0	$zero	Hardwired to 0
$1	$at	Reserved for pseudo-instructions
$2-$3	$v0,$v1	Return Values from functions
$4-$7	$a0-$a3	Arguments to functions - not preserved by subprograms
$8-$15	$t0-$t7	Temporary Data - not preserved by subprograms
$16-$23	$s0-$s7	Saved Registers - preserved by subprograms
$24-$25	$t8-$t9	More temporary registers - not preserved by subprograms

Continued on next page

Continued from previous page

Register Number	Conventional Name	Usage
$26-$27	$k0 - $k1	Reserved for kernel. DO NOT USE
$28	$gp	Global Area Pointer (base of global data segment)
$29	$sp	Stack Pointer
$30	$fp	Frame Pointer
$31	$ra	Return Address
$f0-$f3		Floating Point Return values
$f4-$f10		Temporary Registers - not preserved by subprograms
$f12-$f14		First two arguments to subprograms - not preserved by subprograms
$f16-$f18		More temporary registers - not preserved by subprograms
$f20-$f30		Saved registers - preserved by subprograms

1.2 Instruction Set

Instruction	Description
ADD	Add (w/Overflow)
ADDI	Add immediate (w/Overflow)
ADDIU	Add immediate unsigned (wo/Overflow)
ADDU	Add unsigned (no Overflow)
AND	Bitwise AND
ANDI	Bitwise AND immediate
BEQ	Branch on Equal
BGEZ	Branch on greater than or equal to zero
BGEZAL	Branch on greater than or equal to zero and link
BGTZ	Branch on greater than zero
BLEZ	Branch on less than or equal to zero
BLTZ	Branch on less than zero
BLTZAL	Branch on less than zero and link
BNE	Branch on not equal
DIV	Divide
DIVU	Divide Unsigned
J	Unconditional Jump
JAL	Jump and link
JR	Jump to address in Register

Continued on next page

Continued from previous page

Instruction	Description
LB	Load Byte
LUI	Load Upper Immediate
LW	Load Word
MULT	Multiply
MULTU	Multiply Unsigned
NOOP	No operation
NOR	Bitwise NOR
OR	Bitwise OR
ORI	Bitwise OR Immediate
SB	Store Byte
SLL	Shift Left Logical
SLLV	Shift Left Logical Variable
SLT	Set on Less Than (signed)
SLTI	Set on Less Than Immediate (signed)
SLTIU	Set on Less Than Immediate (unsigned)
SLTU	Set on Less Than Unsigned
SRA	Shift Right Arithmetic
SRL	Shift Right Logical
SRLV	Shift Right Logical Variable
SUB	Subtract
SUBU	Subtract Unsigned
SW	Store WOrd
SYSCALL	System Call
XOR	Bitwise XOR
XORI	Bitwise XOR Immediate
MFHI	Move from $HI
MFLO	Move from $LO
LI	Load Immediate

1.2.1 Instruction Encoding

1. R-Type Instructions

Instruction	Syntax	Opcode/Funct(hex)
SLL	$rd = $rt « shamt	0/00
SRL	$rd = $rt » shamt	0/02
JR	PC=$rs	0/08
MULT	${HI,LO} = $rs * $rt	0/10

Continued on next page

Continued from previous page

Instruction	Syntax	Opcode/Funct(hex)
DIV	$LO = $rs/$rt, $HI = $rs % $rt	0/1a
ADD	$rd = $rs + $rt	0/20
SUB	$rd = $rs - $rt	0/22
AND	$rd = $rs & $rt	0/24
OR	$rd = $rs ‘or‘ $rt	0/25
NOR	$rd = ~($rs ‘or‘ $rt)	0/27
SLT	$rd = ($rs<$rt)? 1:0	0/2a

2 Test Programs\

2.1 Livermore Loops:

```
ADD s0 s1 s2
```