ISA Design Documentation

Marko Ljubičić

Ben Nagel

Keoki Daley

Logan Ropelato

CS 3710

Fall 2017

Table of Contents

Section 1: Overview

We propose a simple, Reduced instruction set Computer (RisC) inspired processor design that utilizes a compact instruction set which can be appended with additional bits of data to further specify desired instructions.

The main driving force or idea behind this architecture is that with a relatively small set of 'core' instructions, it will be fairly straightforward to implement while still retaining a degree of flexibility by allowing for the utilization of additional bits of data that take the form of Opcode.

Section 2: Instruction List

Our processor will be able to execute up to 16 base instructions:

I. ADD: Add

II. ADDi: Add immediate

III. SUB: Subtract

IV. LW: Load Word

V. SW: Store Word

VI. LI: Load Immediate

VII. CMP: Compare

VIII. B: Branch

IX. J: Jump

X. SLL: Shift Logical Left

XI. SLR: Shift Logical Right

XII. AND

XIII. NAND

XIV. OR

XV. NOT

In the cases of Branch and Jump, additional bits of Op-code can be appended to the instructions that will specify how the instruction is to be used (i.e, Branch if not equal) .

We will also be making use of 16 registers, each 24 bits in length to encompass both the 16 bit data values and 24 bit address values which will need to be processed. Register 0 will always contain the value of 0.

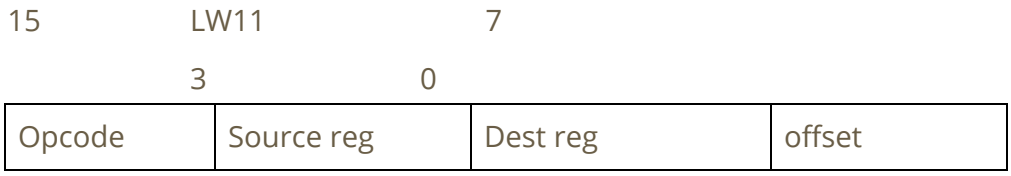
Section 3: Instruction Organization

Example Instructions

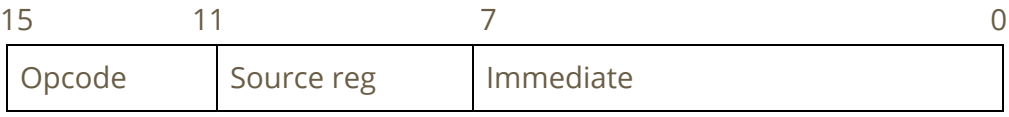
Operation	Opcode	Type	Sample Code	Action
add	0000	ALU	add $rs, $rd	$rd = $rs + $rd
addi	0001	IMM	addi $rs, $imm	$rd = $rd + imm
sub	0010	ALU	sub $rs, $rd	$rd = $rd - $rs
lw	0011	ADDR	lw $rs, (addr)	$rs = (addr)
sw	0100	ADDR	sw $rs, (addr)	(addr) = $rs
li	0101	IMM	li $rs, imm	$rs = imm
cmp	0110	ALU	cmp $rs, $rd	$rd - $rs, set appropriate flags
br	0111	ADDR	br $rs, (addr)	Branch (op-conditional) to addr
jmp	1000	ADDR	Jmp (addr)	Jump to addr
sll	1001	ALU	sll $rd, offset	$rd = $rd << offset
slr	1010	ALU	slr $rd, offset	$rd = $rd >> offset
and	1011	ALU	and $rs, $rd	$rd = $rd & $rs
nand	1100	ALU	nand $rs, $rd	$rd = $rd ~& $rs
or	1101	ALU	or $rs, $rd	$rd = $rd | $rs
not	1110	ALU	not $rs, $rd	$rd = ~$rs

Instruction Layout

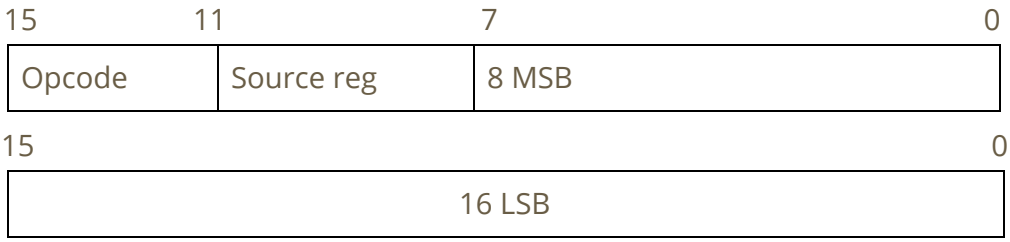
ALU:



IMM:



ADDR:



Section 4: Memory Layout

Memory Address Offset (32Kb)	Hexidecimal Offset	Characterization
0	0x0000	Drawable Region (128x64)
8Kb	0x2000	Glyph's
9Kb	0x2400	Program Memory (Stack, Heap)
15Kb	0x3b00	Memory-Mapped I.O.
15.5Kb	0x3C8C	Instructions

Section 5: Example Loop

Loop Code in Psuedo-C

```
sum = 0;
for (i = 0; i < 100; i++)
    sum += a[i];
```

Corresponds to:

```
li %i, 0                ; i = 0
li %sum, 0              ; sum = 0
li %tmp, 0              ; tmp = 0
li %tmp1, 0             ; tmp1 = 0
Loop:
    sll  %tmp, %i, 2    ; tmp = i * 4 (interger)
    add  %tmp, %a       ; tmp = tmp + &a
    lw   %tmp1, %tmp    ; load value @ address tmp into tmp1
    add  %sum, %tmp1    ; sum += tmp
    addi %i, 1          ; i += 1
    li   %cmp, 100      ; load 100 into cmp
    cmp  %i, %cmp       ; compare i < 100
    br   %ZF, Loop      ; branch if Zero Flag set
```