Min Processor: Architecture Specification

Augmented with AADD Rx, Ry as a newly added instruction

Instruction Format:

**operation [15 : 10]**

000 001 : load

000 010 : store

000 011 : test

001 100 : add

010 100 : sub

011 100 : and

000 101 : bz

000 110 : pop

001 110 : push

000 111: aadd

\* static 3-bit alu operation code supplied by instruction decoder is equal to operation[15 : 13]

**ry [3 : 0]**

second operand register no. for register direct (AR) mode

or

register no. of the register involved in memory operand addressing (AI and AB) modes

**Mode [5:4]**

00 : AR (register direct)

01 : AI (register indirect) ry@

10 : AB

(base register + displacement)

(ry+d)@

mode applies to second operand address

**rx [9 : 6]**

first operand register no.

\*\* An instruction that uses AB addressing mode has an instruction extension word which contains the displacement value (a signed 16-bit number)

\*\* First operand is always a register: specified by rx field. The second operand can be a register or a memory location specified by the combination of mode and ry fields

\*\* Only for load instruction first operand is the destination, for all other instructions, second operand is the destination and first operand is a source operand. Additionally, second operand is also the second source operand in instructions using two source operands.

**The execution unit control word is structured using the following control fields:**

asrccntl : a-bus source control

adestcntl : a-bus destination control

bsrccntl : b-bus source control

bdestcntl : b-bus destination control

alucntl : alu function control

memcntl : memory access control

irecntl : irf to ire transfer control

nssel : next state selection control

dbin : direct branch state ID

After going through the final Min flowcharts, following different usages of each control field are noticed and encoded:

asrccntl: pc -> a : 011; t1 -> a : 101; ry -> a : 010; t2 -> a :110; rx -> a : 001; none : 000

bsrccntl: di -> b : 111; ry -> b : 010; t1 -> b : 101; t2 -> b : 110; rx -> b : 001; none : 000

adestcntl: a -> pc : 11; a -> t2 : 01; a -> ry : 10; none : 00

bdestcntl: b -> t2 : 100; b -> pc : 011; b -> rx,t2 : 101; b -> ry,t2 : 110; b -> ry : 010; b -> rx : 001;

none : 000

Pl. note that alu, ao, do are not storage locations: they only use values on bus-a or bus-b as inputs for operations by the alu or memory and are thus selected for those operation by the control fields associated with alu and memory (alucntl, memcntl respectively)

alucntl: a -> alu, +1 -> alu, add-n/x : 001; a -> alu, b -> alu, add-n : 010;

a -> alu, b -> alu, op-s : 110; a -> alu, 0 -> alu, add-s : 100;

a -> alu, -1 -> alu, add-n : 011; none : 000 (computes a+0 but does not load t1 or ccreg)

memcntl: a -> ao, edb -> di (memory read) : 001; a -> ao, edb -> irf (memory read) : 010;

b -> ao, edb -> di (memory read) : 101; b -> ao, a -> do (memory write) : 111;

none : 000

irecntl: irf -> ire :1; otherwise : 0

nssel: direct branch address supplied by the control word to be the next state : 00;

ib output from the instruction decoder to be the next state : 01;

sb output from the instruction decoder to be the next state : 10;

next state for conditional branch computed by bit-stuffing the condition bit (aluout = 0)

\*\* all register and memory reads are asynchronous. All register and memory writes are synchronous

Min processor register set:

Min processor has sixteen 16-bit general purpose registers

Min has four 1-bit condition code registers : Zero, Negative, Carry out, Overflow

Min optimized flowcharts are given in Microprocessor Logic Design (by Nick Tredennick)

On page 48 – 49. These flowcharts have been used for designing of the processor.

The Control word format design uses three main fields:

1. Execution unit and memory control field : which is divided into the following sub-fields: a-bus source control, a-bus destination control, b-bus source control, b-bus destination control, alu function control, memory data transfer control, and ifr to ire transfer control
2. Next state selection control : used by the controller to determine the next state from among the various choices available to it for appropriate sequencing of states
3. Direct branch address : used by the controller as one of the possible choices for the next state

Following sheets illustrates the process of conversion of flowcharts to RTL Verilog code.