PRCO - Processor Documentation

PRCO304 - Processor Documentation

Ben Lancaster
February 13, 2018

Revision History

Table 1: Document revisions.

Date	Version	Changes
13/02/2018	1.10	Add Control and Pipeline section.
04/02/2018	1.00	Initial revision. Processor introduction. Initial ISA. Initial Register definitions.

Table of Contents

1 PRCO Processor

The PRCO processor is a soft-microprocessor design targeted for general purpose computing and co-processing.

1.1 Features

- Small, embeddable, Verilog core.
- 16-bit RISC instruction set.
- 16-bit register, ALU, and IO, bus widths.
- 12+12 general purpose IO inputs and outputs.
- 9 special IO pins.
  - 4 PWM pins.
  - 2 RS232 pins.
  - 3 SPI pins.

2 PRCO Architecture

2.1 Registers

PRCO has a total of 6 addressable, read and write, registers. These registers are identified by letters A through F.

2.1.1 General Purpose Registers

Registers A through D are designed for general purpose use and are safe to store user values over the run-time of the processor.

Table 2: General purpose registers.

Registers	Bits	Description
A through D	15:0	4 General purpose registers

Instructions that require a destination register, such as CMP, can reference any register (even special registers if that is your requirement). For the CMP instruction as an example, the processor will put the result of the comparison instruction in the destination register, overwriting any value present in that register.

2.1.2 Special Registers

Registers E and F are special registers within the processor. The processor cannot guarantee that a value written or read in these registers will persist over the run-time of the processor. Erroneously writing to these registers may severely affect program and processor behaviour.

Even though all registers can be used at the will of the programmer, it is recommended to isolate a few registers to provide special features, such as RAM stack management, interrupts, and IO multiplexing.

Table 3: Special registers.

Registers	Bits	Description
E	15:0	RAM Stack pointer
F	15:0	RAM Base pointer

2.2 Control and Pipelining

The PRCO processor employs primitive control and pipeline strategies to provide reliable instruction performance, optimisation, and delay and fault handling.

Each module within the processor is connected to the previous and next module by control signals.

2.3 Interrupts and Exceptions

3 PRCO Instruction Set Architecture

This section describes instructions available on the PRCO processor.

The following instruction definitions use the following letters to describe values: X for any value; 0 for all zeros; 1 for all ones; Imm8 for unsigned 8-bit immediate; Simm5 for signed 5-bit immediate.

3.1 Timings

As the processor does not employ instruction pipelining techniques, but instead uses control signals to individually turn on sub-processes on the CPU. These sub-processes do not happen in parallel.

For instructions that do not require a RAM read/write request, the RAM stage of the control sequence is skipped reducing the instruction cycle by 1 CPU clock for that instruction.

The fastest instruction in terms of CPU cycles is the NOP instruction. The greatest number of cycles for an instruction includes all RAM read/write request operations, such as the LW and SW instructions (see section 3.2).

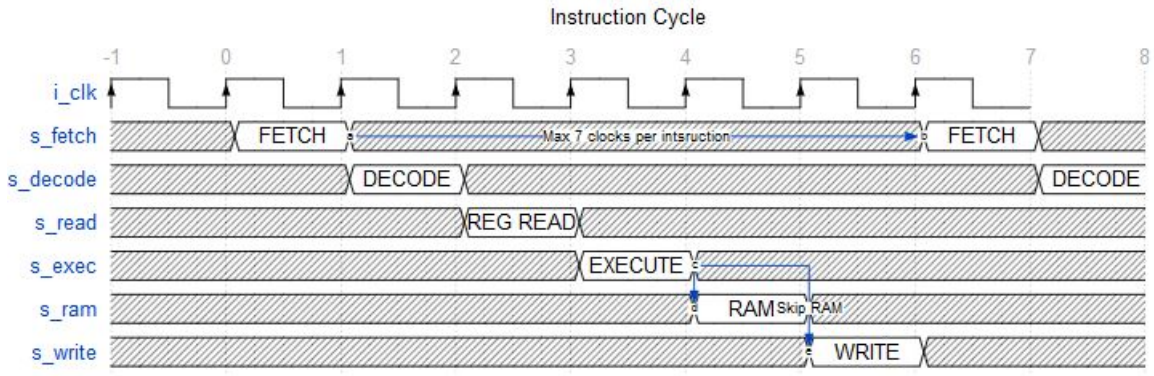

Figure 1: The Discrete Fourier Transform definition (**?**).

3.2 General Instructions

The term, general instruction, is given to instructions that are common to primitive operations such as arithmetic and comparison instructions.

3.2.1 NOP

Description The NOP instruction performs no action for 1 instruction cycle (see section 3.1).

Assembly NOP

Pseudocode

Registers altered

Clock cycles 2 (FETCH, DECODE)

15:11	10:0
00000	X

3.2.2 LW - Load Word

Description Copies a 16-bit word from RAM to a register.

Assembly LW

Pseudocode Rd <= RAM[Ra + Simm5]

Registers altered Rd

Clock cycles 6 (FETCH, DECODE, READ, EXECUTE, RAM, WRITE)

15:11	10:8	7:5	4:0
00001	Rd	Ra	Simm5

3.2.3 MOVR

Description The MOVR instruction copies a 16-bit register value to another register.

Assembly MOVR %Ra, %Rd

Pseudocode Rd <= Ra

Registers altered Rd

Clock cycles 5 (FETCH, DECODE, READ, EXECUTE, WRITE)

15:11	10:8	7:5	4:0
00011	Rd	Ra	X

3.2.4 MOVI

Description The MOVR instruction copies a 16-bit register value to another register.

Assembly MOVR %Ra, %Rd

Pseudocode Rd <= Ra

Registers altered Rd

Clock cycles 5 (FETCH, DECODE, READ, EXECUTE, WRITE)

15:11	10:8	7:0
00100	Rd	Imm8

3.2.5 ADD

Description The ADD instruction adds an immediate value to a destination register, Rd.

Assembly ADDI $255, %Rd

Pseudocode Rd <= Rd + Imm8

Registers altered Rd

Clock cycles 5 (FETCH, DECODE, READ, EXEC, WRITE)

15:11	10:8	7:5	4:0
01000	Rd	Ra	X

3.2.6 ADDI

Description The ADD instruction adds an immediate value to a destination register, Rd.

Assembly ADDI $255, %Rd

Pseudocode Rd <= Rd + Imm8

Registers altered Rd

Clock cycles 5 (FETCH, DECODE, READ, EXEC, WRITE)

15:11	10:8	7:0
01001	Rd	Imm8

3.2.7 SUBI

Description The SUB instruction subtracts an immediate value from a destination register, Rd.

Assembly SUBI $255, %Rd

Pseudocode Rd <= Rd - Imm8

Registers altered Rd

Clock cycles 5 (FETCH, DECODE, READ, EXEC, WRITE)

15:11	10:8	7:0
01001	Rd	Imm8

3.2.8 CMP

Description Sets register, Rd, to the value of Ra - Rb.

Assembly CMP %Ra,

Pseudocode Rd <= CMP(Ra, Rd)

Registers altered Rd

Clock cycles 5 (FETCH, DECODE, READ, EXEC, WRITE)

15:12	11:9	8:6	5:3	2:0
0003	Rd	Ra	Rb	X

3.3 Special Instructions

4 Compiler

4.1