COL216 Assignment 3 Report

Navneel Singhal, Akash S

January 2020

Implementation details

We developed a simple processor implementing a subset of the MIPS instruction set (add, sub, sll, srl, sw, lw).

Specifications of processor

1. We have two constants which store the memory range parameters - `memzero` and `memmax`.
2. The machine instructions are loaded into the memory from index 0 to `memzero` - 1.
3. The data memory is stored from `memzero` to `memmax`.
4. The memory is loaded using an input file
5. The input format for the file is as follows :
   (a) The first few lines contain the program, line by line in MIPS instruction format as a 32-bit binary string.
   (b) The line that signals the end of the program body should be a string of 32 0s.
   (c) The lines with index `memzero` $+ x$ contain the value stored in the x^{th} data memory location, as a 32-bit binary string.
6. Invalid instructions are ignored.
7. We assume that any register that is used has to be initialised in the processor before being used (i.e. the registers are not initialised to some value by default, and the values need to be loaded manually as li isn't implemented).
8. Each instruction is executed in a clock cycle (clock period 1ns in simulation)

Testing

Testing is an important part of programming. It verifies the correctness of the code. We have divided our testing in two parts : corner case testing and random testing.

Corner case testing

Corner case testing tests case which could cause an error in our processor, these errors are either resolved or are mentioned in our specifications.

(a) Number of registers out of bounds - We have used 32 registers in our register file. The register number is specified by a 5-bit value in the MIPS instructions set (rs, rt, rd). So there cannot be number of registers out of bounds as any value specified by the 5-bit value has to lie between 0 and 31.

(b) Shifting left and right can be only done for unsigned numbers correctly. If a negative number is shifted then only binary shifting is done (i.e. negative number can become positive or positive number can become negative)

(c) Overriding register values also works (i.e. add \$t1, \$t1, \$t2 works, the value of t1 is overridden)

(d) There is a limit for number of instructions (as there can only be `memzero` instructions including the program terminating indication line consisting of all 0s), as instructions are stored in memory from 0 to `memzero` - 1.

(e) There is also a limit for number of data memory (which is `memmax - memzero` + 1).

Random Testing

Random testing allows use to test our processor for a large number of instructions which was not possible in corner cases testing.

(a) We have written a C++ code which generates a random set of MIPS instructions, and another C++ code which converts the MIPS instruction to machine code. Then the machine code is fed into our input file, and the processor is simulated. The results of the simulation are checked correspondingly with the random set of MIPS instruction.