Objective:

To develop a random test tool which is capable of performing functional and performance tests for a candidate microprocessor against the golden processor.

Motivation:

Modern processors are incredibly complex marvels of engineering that are becoming increasingly hard to evaluate. The random tests for microprocessor is proven to be valuable in the past by detecting the failure of the processor in un-common situations. Simplescalar is a set of tools which perform fast, flexible, and accurate simulation of modern processors that implement the SimpleScalar architecture like PISA. This project is motivated by the Simplescalar's testing capability of the complex modern processors.

The tool set takes binaries compiled for the SimpleScalar architecture and simulates their execution on one of several provided processor simulators. The tool set also includes sample simulators ranging from a fast functional simulator to a detailed, dynamically scheduled processor model that supports non-blocking caches, speculative execution, and state-of-the-art branch prediction. In addition to simulators, the SimpleScalar tool set includes performance visualization tools, statistical analysis resources, and debug and verification infrastructure.

Implementation:

Generating random sequence of instructions and detecting data hazards

isa.py -

- Machine description file which includes the set of instructions supported by the test tool. Instructions are classified based on the PISA ISA encoding.

generator.py - input files: seqstart.txt, seqend.txt output files: seqgen.s, seqhaz.txt

- Generates random assembly sequence of desired length by randomly selecting instructions from the list of instructions in isa.py and stores the sequence in seqgen.s
- Branching out of range is avoided by branching with labels. Each random instruction is given a label. Tool can randomly select a label of the random sequence. The last two instructions in the sequence cannot be Branch instruction and tool supports only forward loops in order to avoid infinite loops. This makes sure branching is within the sequence generated.
- seqhaz.txt reports the data hazards of the generated random sequence. Data hazards are obtained considering no data forwarding.

Testing microprocessor

Simulator- sim-outorder

seqgen.s- output file: reglog.txt, reglog_g.txt, log.txt, log_g.txt

- Golden processor uses the default configuration (-issue:inorder false). Considered a candidate processor which uses different pipeline (-issue:inorder true) compared to golden processor.
- Compiled sequence is run for golden processor and candidate processor, register contents and stats report of the simulation is collected in reglog.txt and log.txt respectively for candidate processor and in reglog_g.txt and log_g.txt for golden processor.

verify.py-

- Verifies the register output and CPI of the candidate processor with the expected values of golden processor and reports the (functional and performance)test result

tester.sh-

- The entire test process is automated using the script tester.sh

Usage:

./tester.sh -l <length of sequence> [-ip]

-l : required option for length

-ip : optional, To indicate the tool to include performance test for processor.

Default – Tool only performs functional test for processor.

References:

- http://www.simplescalar.com/
- http://www.simplescalar.com/docs/users_guide_v2.pdf
- http://www.cse.iitd.ernet.in/~drajeswari/ss_installn.html