SPEC95 Benchmarks for SimpleScalar

This quarter, we will support four SPEC95 benchmarks on the SimpleScalar simulator: *compress*, *cc1*, *go*, and *perl*. All benchmark programs are stored in /cse/courses/cse471/06sp/simplescalar/benchmarks/SPEC95, and the compiled binaries have the extension *.ss*. Inputs to the benchmarks are stored in /cse/courses/cse471/06sp/simplescalar/inputs/SPEC95.

compress:

This benchmark generates an in-memory buffer of data, compresses it to another in-memory buffer, then decompresses it.. It uses the compression algorithm from an old UNIX utility of the same name. Its input file, *compress.in*, specifies how large a random buffer to generate and seeds the generation process. The version in the inputs directory operates on a 136000 byte file, and causes around 400 million instructions to execute.

Usage: compress.ss <compress.in

cc1:

This benchmark consists of an old C compiler. It compiles an already pre-processed input file, producing an assembly file as output. Its input file, cc1.in, is a preprocessed C file for a short C program. The benchmark executes around 120 million instructions. The parameter "-o /dev/null" causes the compiler not to write its results to a file, since you won't really need them.

Usage: cc1.ss cc1.in –o /dev/null

go:

An AI algorithm for playing the game of Go. The input file for this benchmark is 2stone9. It runs for 550 million instructions.

Usage: go.ss 50 9 2stone9.in

perl:

This benchmark is an old perl interpreter. Its inputs are perl scripts and their respective inputs. The scripts available to run in the perl interpreter are *primes.pl*, and *charcount*. *Primes.pl* checks a list of numbers and identifies the primes via a brute-force algorithm. Its inputs are called *primes.in*, *primes.in.small* or *primes.in.sm*. *charcount* counts the number of characters in a file. Its inputs are *all_gre_words* or *gre_words.a*.

Run with input *primes.in.small*, the prime verifier runs for only around 8 million instructions, and with primes.in.sm for around 1.2 million. These are very short running times, and may be dominated by startup and cleanup execution, rather than the benchmark's main loop. The input *primes.in* runs for 7.1 billion instructions, and takes substantially longer to execute as a result. You can pretty easily alter the running time of this benchmark by adding or removing lines from primes.in – more lines can be found in primes.in.big.

Run with input *all_gre_words*, *charcount* runs for 65 million instructions.

Usage: perl.ss primes.pl primes.in
Usage: perl.ss charcount all_gre_words