Computer Architectures

2nd part labs – lab 3

Using SimpleScalar

1) Launch the SimpleScalar virtual machine or use the virtualization system
2) running SPEC95 programs

the SPEC95 programs (*anagram*, *compress95*, *go*, *cc1*, and *perl*), the programs' inputs, and the trace of the programs are placed in:

```
~/simplescalar/BenchMarks_Little$
```

a. run the 5 binary files and check the obtained results by executing:

- anagram

```
student@student-laptop:~/simplescalar$
./simplesim-3.0/sim-outorder
BenchMarks_Little/Programs/anagram.ss
BenchMarks_Little/Input/words <
BenchMarks_Little/Input/anagram.in 2>
BenchMarks_Little/Results/anagram.trace >
BenchMarks_Little/Results/anagram.out
```

- go

```
student@student-laptop:~/simplescalar$
./simplesim-3.0/sim-outorder

BenchMarks_Little/Programs/go.ss 50 9
BenchMarks_Little/Input/2stone9.in 2>
BenchMarks_Little/Results/go.trace >
BenchMarks_Little/Results/go.out
```

- cc1

```
student@student-laptop:~/simplescalar$
./simplesim-3.0/sim-outorder
BenchMarks_Little/Programs/cc1.ss -O
BenchMarks_Little/Input/1stmt.i 2>
BenchMarks_Little/Results/cc1.trace
```

- perl

```
student@student-laptop:~/simplescalar$
./simplesim-3.0/sim-outorder
BenchMarks_Little/Programs/perl.ss <
BenchMarks_Little/Input/perl-tests.pl 2>
BenchMarks_Little/Results/perl.trace >
BenchMarks_Little/Results/perl.out
```

- compress95:

```
student@student-laptop:~/simplescalar$
./simplesim-3.0/sim-outorder
BenchMarks_Little/Programs/compress95.ss <
BenchMarks_Little/Input/compress95.in 2>
BenchMarks_Little/Results/compress95.trace >
BenchMarks_Little/Results/compress95.out
```

3) benchmarks behavior

execute the SPEC95 programs using the configuration you found as the best one in the previous lab (row 7 of the table in the exercise No 6)

a. check the program output, and collect for every program the following statistics parameters:

- `sim_num_insn` `// total number of instructions committed`
- `sim_cycle` `// total simulation time in cycles`
- `sim_IPC` `// instructions per cycle`
- `sim_CPI` `// cycles per instruction`

b. collect additional information related to the branch instructions behavior in the SPEC95 programs by checking:

- `bpred_bimod.misses` `// total number of misses`
- `bpred_bimod.addr_hits` `//total number of address-predicted hits (in BTB)`
- `bpred_bimod.dir_hits` `// total number of direction-predicted hits (includes addr_hits)`

4) comparing BPUs behavior

execute the SPEC95 programs using the simulator default configuration. Then, modify the BPU as detailed in the following table and compare the prediction rate results (missprediction rate)

a. use the following hints, that refer to the simulator option `bpred`, in order to modify the processor BPU.

- *nottaken*

for setting a not taken strategy:

```
-student@student-laptop:~/simplescalar$ ./simplesim-3.0/sim-outorder -bpred nottaken test-math
```

- *bimod* is a 2-bit BHT in simplescalar

  for setting a 2-bit BHT of 1024 entries:

  ```
  -student@student-laptop:~/simplescalar$ ./simplesim-3.0/sim-outorder -bpred bimod -bpred:bimod 1024 test-math
  ```

- *btb* in simplescalar it requires to use the bimod predictor:

  for setting a BTB of 1024 entries without associativity:

  ```
  -student@student-laptop:~/simplescalar$ ./simplesim-3.0/sim-outorder -bpred bimod -bpred:btb 1024 1 test-math
  ```

Progrs / BPU	Not Taken / Taken	BTB 1 entry	BTB 256 entries	BTB 512 entries	BTB 1024 entries	BHT 1 entry	BHT 256 entries	BHT 512 entries	BHT 1024 entries
compress95									
go									
anagram									
cc1									
perl									

Table 1: Branch prediction rate for different BPU configurations