ECE571: Advanced Microprocessor Design – Fall 2024
Homework 4

Due: Friday 4 October 2024, 12:00pm

Create a document that contains the data and answers described in the sections below. A .pdf or .txt file is preferred but I can accept MS Office or Libreoffice format if necessary.

1. **Branch ratios on x86 Haswell-EP Machine**

   We will determine exactly how often branches happen in some benchmarks. In class I said traditionally computer architects say that one in five instructions is a branch. We will find out if that is true.

   For this section, log into the Haswell-EP machine just like in the previous homeworks.

   (a) Run the bzip2 benchmark and measure `instructions:u`, `branches:u`, and `br_instr_retired.conditional:u`

   ```
   perf stat -e instructions:u,branches:u,br_inst_retired.conditional:u \
   /opt/ece571/401.bzip2/bzip2 -k -f ./input.source
   ```

      i. What are the total number of instructions, branches, and conditional branches?
      ii. What is the ratio of branches to total instructions? (something like 1:5)
      iii. What is the ratio of conditional branches to total instructions?

   (b) Now do the same test with the equake_l benchmark (note that's an L not a 1).

   ```
   perf stat -e instructions:u,branches:u,br_inst_retired.conditional:u \
   /opt/ece571/equake_l.specomp/equake_l < \
   /opt/ece571/equake_l.specomp/inp.in
   ```

      i. What are the total number of instructions, branches, and conditional branches?
      ii. What is the ratio of branches to total instructions?
      iii. What is the ratio of conditional branches to total instructions?

2. **Branch miss rate on x86 Haswell Machine**

   Now we will calculate the branch miss rates for the benchmarks.

   (a) For the bzip2 benchmark measure `branches:u` and `branch-misses:u`. Calculate the branch miss rate $\frac{branch_{miss}}{branch_{total}} * 100$ (hint, you'll notice perf does this for you when you measure these instructions).

   (b) Also calculate the branch miss rate for equake_l.

3. **Speculative execution on x86 Haswell Machine**

   We learned that if a branch prediction fails, then the instructions down the wrong path have to be kicked out of the pipeline. We can see how often this occurs by looking at the difference between "executed" instructions (ones that made it to the execute stage) versus "retired" instructions (ones that actually finished and were written back).

   Haswell, unlike some other processors, has no "executed instructions" event but instead we can use the μop (micro operation) events `uops_retired.all:u` and `uops_executed.core:u`.

   (a) Find out what percentage of instructions were executed but not retired with bzip2.

   (b) Find out what percentage of instructions were executed but not retired with equake_l.

4. **Branch miss rate on x86 AMD Epyc Machine**

   Now log into an AMD EPYC machine and see how the branch miss rate compares.

   From the haswell-ep machine, you can type `ssh epyc` and use the same password you use on haswell-ep and it should let you log in.

   Remember to first copy the input file to your local directory.
   `cp /opt/ece571/401.bzip2/input.source .`

   (a) For the bzip2 benchmark measure `branches:u` and `branch-misses:u`. Calculate the branch miss rate $\frac{branch_{miss}}{branch_{total}} * 100$ (hint, you'll notice perf does this for you when you measure these instructions).

   (b) Also calculate the branch miss rate for equake_l.

5. **Branch ratios on an ARM64 System**

   For this section you will log into an Ampere ARM64 server. To do this, when logged into the Haswell-EP machine run:
   `ssh ampere`
   Your password should be the same as it is on the Haswell-EP machine.

   Gather the results using perf.

   (a) Run the bzip2 benchmark and measure `inst_retired:u` and `br_retired:u`.
   (Remember to first copy the input file to your local directory).
   `cp /opt/ece571/401.bzip2/input.source .`

   What are the total number of instructions and branches? What is the ratio of branches to total instructions?

   (b) Repeat this, but for `equake_l`

6. **Branch miss rate on ARM64 System**

   (a) For the bzip2 benchmark measure `br_retired:u`, `br_pred:u`, `br_mis_pred:u` and `br_mis_pred_retired:u`. The first is total branches retired, the second is predictable branches (which includes speculative branches), the next is mispredicted total branches, and finally mispredicted retired branches. You can see a lot of branch prediction happens down speculative paths, so it's an interesting question how to calculate branch prediction rates. For this homework lets calculate it based on br_pred vs br_mis_pred.

   (b) Also calculate the branch miss rate for equake_l.

7. **Short Answer Questions**

   (a) Does the branch to instruction ratio differ between equake and bzip2 on Haswell-EP? What might cause this? How could you test to see if that's actually the cause?

   (b) Does the branch to instruction ratio differ between bzip2 on the Haswell-EP machine and bzip2 on the ARM64 machine? Why might this be?

   (c) How do the branch miss-prediction rates compare between bzip2 and equake on the Haswell-EP machine? What might be the source of any differences?

   (d) How do the branch miss-prediction rates compare between bzip2 on Haswell-EP and bzip2 on the ARM64 machine?

   (e) The additional measurements below were gathered on a relatively new Raptor Lake system from Intel, which is approximately 10 years newer than the Haswell EP. We get the following results (note: this computer has hybrid cores that complicate the measurements, the reported results here were run with "taskset" used to limit the run to the "core" or "power" cores.

```
bzip2 results:
     3,049,155,475      cpu_core/branches:u/
       189,708,000      cpu_core/branch-misses:u/
        2.06s elapsed

equake_l taskset --cpu-list 0-15 results
    89,219,153,632      cpu_core/branches:u/
       337,060,421      cpu_core/branch-misses:u/
      27.665239912 seconds time elapsed
```

   Are the branch prediction results for `bzip2` and `equake_l` better than those on the much older haswell-ep machine?

   (f) How do the executed vs retired instruction rates differ between bzip2 and equake on the Haswell-EP machine? What implications might this have about the power efficiency of the two benchmarks?

   (g) Imagine you wanted to write a benchmark to validate the branch prediction performance counters on a system. What kind of short benchmark could you write that would give you a 50% miss-predict rate?

   Optionally write such a small example program, test it out, and report your results.

8. **Submitting your work.**

   - Create the document containing the data as well as answers to the questions asked.
   - Please make sure your name appears in the document.
   - e-mail the file to me by the homework deadline.