ECE571: Advanced Microprocessor Design – Spring 2017
Homework 5

Due: Thursday 23 February 2017, 11:00am

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 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 machine just like in the previous homeworks.

   (a) Run the bzip2 benchmark and measure `instructions:u`, `branches:u`, and `r5301c4:u` (`r5301c4:u`, according to libpfm4, corresponds to the `BR_INST_RETIRED:COND` event which measures only conditional branches).

   ```
   perf stat -e instructions:u,branches:u,r5301c4: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?
      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,r5301c4: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 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 other processors, has no "executed instructions" event but instead we can use the μop (micro operation) events. libpfm4 tells us that `UOPS_RETIRED` is `r5301c2:u` and `UOPS_EXECUTED` is `r5302b1: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 ratios on an ARM64 System**

   For this section you will log into my new NVIDIA Jetson-X1 64-bit ARM board. To do this, when logged into the Haswell machine run:
   `ssh jetson`
   Your password should be the same as it is on the Haswell machine.

   Gather the results using perf.

   (a) Run the bzip2 benchmark and measure `instructions` and `branches`.
   (Remember to first copy the input file to your local directory).

   ```
   cp /opt/ece571/401.bzip2/input.source .
   perf stat -e instructions:u,r10:u,r12:u \
   /opt/ece571/401.bzip2/bzip2 -k -f ./input.source
   ```

   `r10:u` maps to `PC_BRANCH_MIS_PRED` and `r12:u` maps to `PC_BRANCH_PRED`. So the total number of branches is the sum of the two (mispredicted and predicted branches).

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

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

   (a) Calculate the branch miss rate for bzip2. You can do this based on the results you already obtained in the previous question.

6. **Short Answer Questions**

   (a) Does the branch to instruction ratio differ between equake and bzip2 on Haswell? Why might this be?

   (b) Does the branch to instruction ratio differ between bzip2 on the Haswell 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 machine? What might be the source of any differences?

   (d) How do the branch miss-prediction rates compare between bzip2 on Haswell and bzip2 on the ARM64 machine? What different design decisions might have been made between the two machines that affects these results?

(e) A raspberry-pi3 gives the following results for bzip2.

```
 19510081072      instructions:u
  1251257641      branches:u
   220229064      branch-misses:u  #   17.60% of all branches

29.475962689 seconds time elapsed
```

How do these compare to the ARM64 results? Why might the results differ from those on the ARM64?

(f) How do the executed vs retired instruction rates differ between bzip2 and equake on the Haswell 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. (note if you do this, the Linux C library has two random routines, `rand()` and `random()` and one has many extraneous branches while one doesn't).

7. **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.