Lab 8 - Microbenchmarks and SIMD Instructions
NERS/ENGR 570 Fall 2020

Oct. 23rd, 2020
DUE BY: Nov. 6th, 2020

Exercise 1 (60 pts) - Membench

1. Obtain the source code and Makefile for membench at:
   `https://github.com/bkochuna/membench.git`

2. Build and run the benchmark on a Great Lakes ***login node***
   a. `make membench`
   b. `make membench.out`
   c. `make membench.png`

3. Examine the results.

4. Modify the `CACHE_MAX` (units are in words) value in `membench.c` to make the largest array 1 GB. Then repeat step 2 for a ***compute node***.

5. Examine the resulting image and try to identify the size of the caches, cache line sizes, and the access times. Annotate the figure like slide 10 of the Lab presentation. You are not required to identify the TLB page sizes or entries (but you are welcome to give it a shot).
   a. *Note:* it may be useful to check the system hardware to determine what the cache sizes and line sizes are. (see lecture 2 slides, or be creative)
   b. In annotating the figure, do this with whatever method is easiest for you. PowerPoint may be good, but there are certainly other ways to do this.

Exercise 2 (40 pts) - SIMD Instructions and In-line assembly

Perform the following each for the Great Lakes compute node.

1. Verify you are on a machine with a processor that supportsAVX or AVX2 instructions
    a. Describe how you did this

2. Check if the GNU compiler version supports AVX or AVX2.
    a. Describe how you did this.

3. Check if the Intel compiler version supports AVX or AVX2.
    a. Describe how you did this.

4. Provided the hardware supports AVX or AVX2 instructions, for the naive matrix multiply you wrote in Lab 4 have the compiler print out the assembly for your `mydgemm` routine (not the whole program).
    a. See lecture 4, or consult the internet on how to do this (you can also use goldbot.org if you wish, but the compiler version should be consistent with the one on Great Lakes)
    b. *Note*: it may be beneficial to put just the minimal amount of code (e.g. a single function or subroutine) in its own file and use this to generate the assembly.
    c. Describe what command you used to get the assembly output, and provide the loops for your naive matrix-matrix multiply.

5. Examine the assembly and determine if AVX or AVX2 instructions are being generated. A list of the AVX2 instructions and registers may be found here: https://en.wikipedia.org/wiki/Advanced_Vector_Extensions http://www.intel.com/content/dam/www/public/us/en/documents/manuals/64-ia-32-architectures-software-developer-vol-1-manual.pdf (Chapter 14)
    a. What are the assembly instructions specific to the matrix-matrix multiply operations?

6. If AVX or AVX2 instructions are not being used, try modifying compiler optimization flags to give you AVX or AVX2 SIMD instructions. Specifically, you want to try and have the compiler give you fused multiply add instructions and to use the 128-bit (or 256-bit) registers and load and store operations.

    What compiler flags did you end up using that produced the AVX or AVX2 instructions?

7. If you are certain you have the correct compiler options and still aren't getting AVX instructions, try modifying your source code.
    Describe what modifications, if any, you had to make to your source.
8. What AVX or AVX2 instructions do you get? Does your mydgemm run faster than before?

Extra-credit (25 pts) - In-line assembly

Write in-line assembly in C or C++ that uses AVX2 instructions for matrix vector multiply.

In delivering the extra credit include your source code in your lab deliverable.

Deliverables

For exercise 1:

1. An annotated figure for the membench results of a Great Lakes compute node
2. The hardware specifications of the machine, e.g.
    a. the processor type and clock speed
    b. Number of levels of cache and their respective sizes and line sizes.
    c. The cache associativity (if you can find it)
3. A discussion of the results.

For exercise 2: include your answers to the questions and any source code snippets

Provide everything in a single pdf to be submitted to canvas.