COS 597C: Assignment 1

Due: September 30, 2010

1. Vectorization using Intel SSE3

The given C program (matrix.c) computes the product of of two matrices, and prints the result. Write a vectorized version of the program, with an aim of maximizing performance with minimal source code changes into matrix-vectorized.c. Use gcc's vector builtins [1] for Intel's Streaming SIMD Extensions 3 [2],to write vectorized code, for a x86-64 processor.
What to submit ?

• matrix-vectorized.c, that generates the same output, as matrix.c. matrix-vectorized.c should ideally run faster than matrix.c

• Report the speedup (measured using time command) and the configuration of the machine on which you ran the code (compiler version – the output of gcc -v and processor info – the output of cat /proc/cpuinfo)

Please do not change the compiler flags in the given Makefile

References

[1] GCC Vector extensions
(http://gcc.gnu.org/onlinedocs/gcc/Vector-Extensions.html,
http://ds9a.nl/gcc-simd/fp-simd-builtins.html,
http://gcc.gnu.org/onlinedocs/gcc-3.4.0/gcc/X86-Built-in-Functions.html)

[2] Intel 64 and IA-32 Architectures Software Developer's Manuals
(http://www.intel.com/products/processor/manuals/)

2. Instruction Level Parallelism

The ILP project will be done using the Trimaran toolchain. You will run several benchmarks against various machine configurations to learn how processor architecture affects benchmark performance. Then, you will use the information you gathered to design a processor.

1. First logon tux machine and run the Trimaran setup script: **source /u/twoh/work/cos597c/trimaran/scripts/envrc.bash**
2. Create a working directory under your home directory, for example, ilp
3. Copy "assignment2.tar.gz" file to your working directory and decompress it.
4. Change the number of clusters to 1 in hpl_pd_elcor_std.hmdes2 file

$def !num_clusters 1

5. Compile the mdes file:
   hc hpl_pd_elcor_std.hmdes2
6. Run a benchmark, mm_dyn for example, using the compiled mdes file:
   tcc –bench mm_dyn –M/u/twoh/ilp/hpl_pd_elcor_std.lmdes2
   (node the lmdes2 suffix on the mdes filename. This is the compiled version of the file that was modified.)

We will use three benchmarks for this assignment:

1. mm_double: Matrix Multiply (uses both float and int)
2. fft: Fast Fourier Transform (uses both float and int)
3. mm_dyn: Matrix Multiply (uses int)

You can make your own configuration by modifying hpl_pd_elcor_std.hmdes2 file. In this assignment, you are allowed to modify

- number of integer/float/memory/branch functional units
  $def !integer_units x
  $def !float_units y
- latency of floating point multiply and divide
  $def !float_multiply_latency x
  $def !float_divide_latency y

Keep number of clusters as 1 for the assignment.

You should find most cost efficient configuration. Cost efficiency can be measured by calculating geometric mean of (total_cycles / design cost) among three benchmarks. total_cycles can be found in PD_STATS file which created under ***{benchmark}*_0/simu_intermediate** directory after running benchmark. Design cost can be calculated with constraints below:

1. You may use up to 6 total functional units (int, float, memory, branch)
2. There is an incremental cost for functional units, as follows:
   a. 4 units are included in the $50 base price
   b. Unit 5 = $10, unit 6 = $20
   c. Floating point unit can be chosen as follows (no miximg):
      i. Slow FPU (adds 1 cycle to mult and div latency) = saves $5 per
      ii. Fast FPU (subs 1 cycle from mult and div latency) = costs $5 per

What to submit?

- Submit a written report in PDF or MS Word format. In the report, describe configuration of each processor you designed, and cost efficiency of the design.
- Submit a **hpl_pd_elcor_std.hmdes2** file for your most cost efficient processor.