

ECE574: Cluster Computing – Homework 4

SIMD/Vector Math

Due: Friday 2 October 2026, 5:00pm

1. Background

- In this homework we will take the sobel code from and parallelize some of the math using SIMD intrinsics.

2. Setup

- For this assignment, log into the same Haswell-EP machine we used in previous homeworks. As a reminder, use the username handed out in class and ssh in like this

```
ssh -p 2131 username@weaver-lab.eece.maine.edu
```
- Download the code template from the webpage. You can do this directly via

```
wget https://web.eece.maine.edu/~vweaver/classes/ece574/ece574_hw4_code.tar.gz
```


to avoid the hassle of copying it back and forth.
- Decompress the code

```
tar -xzvf ece574_hw4_code.tar.gz
```
- Run make to compile the code.

3. Baseline Measurements (3 points)

The `sobel_serial.s` file has some working code that does the sobel convolution from HW#3. It's not the most optimized code but we will use it for comparison purposes.

This code has also been instrumented with PAPI to show cycles `PAPI_TOT_CYC` and retired instructions `PAPI_TOT_INS`.

- (a) Run the `sobel_serial` program on the `space_station_hires.jpg` input.
- (b) Report in the README the number of cycles, instructions, and the IPC (instructions per cycle) for this baseline. There are three metrics reported, for `sobelx`, `sobely`, and the combine, report all of them.

4. gcc Auto-vectorized code (3 points)

The gcc compiler has some ability to use SIMD instructions. By default on x86_64 this is limited because it can only assume SSE2 support as baseline. Also traditionally compilers struggle with vectorizing (adding SIMD instructions) to code.

To enable SIMD instructions in gcc, you have to specify at least `-O3` optimization level. To get AVX2 enabled you can either specify `-mavx` or else let the compiler know what machine you plan to run the code on. You can do this with `-march=native` or in our case we just specify `-march=haswell`

The Makefile provided with the homework builds a copy of `sobel_serial.c` called `sobel_haswell.c` with these options enabled, so it can generate AVX2 code where appropriate. If you are curious you can use `--objdump-disassemble-all` on the executable to see what instructions were added.

- (a) Run the `sobel_haswell` program on the `space_station_hires.jpg`
- (b) Report in the README the number of cycles, instructions, and the IPC (instructions per cycle) for this baseline

- (c) Calculate the speedup against the baseline using the CYCLES results
- (d) Did you see much improvement in speed?

5. Hand-written SIMD code (4 points)

It is possible to hand-write SIMD code. Using inline assembly language would be a pain, so most compilers support Intel's SIMD intrinsics, which let you write SIMD code in a C like fashion which is slightly easier.

As discussed in class, there are an immense amount of these instructions which makes writing code like this difficult. You can find a summary of these intrinsics here:

<https://www.intel.com/content/www/us/en/docs/intrinsics-guide/index.html>

I have provided some code to try out, `sobel_simd.c`, that has attempts at hand-optimizing the routines with intrinsics.

- (a) Run the `sobel_simd` program on the `space_station_hires.jpg`
- (b) Report the number of cycles, instructions, and the IPC (instructions per cycle) for this baseline
- (c) Calculate the speedup against the baseline using the CYCLES results
- (d) Was the convolution code much faster?
- (e) Was the combine code much faster? [the combine code was attempting to combine 8 values at once, so the theoretical maximum speedup would be 8]

6. Extra-credit (optional) (1 points)

Try to see if you can optimize the SIMD code to be any faster.

Work on the `sobel_extra.c` file and see if you can make it faster. If you did get an improvement, make a note in the README and say what you did.

7. Submitting your work.

- Be sure to edit the README to include your name, as well as the timing results, and any notes you want to add about your something cool.
- Run `make submit` and it should create a file called `hw04_submit.tar.gz`.
- Hint: To copy the file from the server to your local machine you can use `scp`. On Linux/MacOS on your local machine you can do something like:

```
scp -P2131 ece574-0@weaver-lab.eece.maine.edu:hw04_submit.tar.gz .
```

you will have to adjust your username. Note it's capital P for the port option. After the colon you can put the relative path from your home directory. The `.` by itself at the end means copy to the current directory on the machine you're running it on.
- e-mail the file to me by the homework deadline.