Optimization Assignment

CS180 Operating Systems

Optimization Assignment

This assignment relates to implementing an efficient polynomial function.

$$a_0 \times x^0 + a_1 \times x^1 + a_2 \times x^2 + \cdots + a_n \times x^n$$

Here is the naive implementation:

```
/* calculate a0 + a1*x + a2*x^2 + ... + an*x^n */
void poly0(const vec_ptr coefficients, const data_t* x_input, data_t* output)
{
    *output = 0;
    for (long n = 0; n < vec_length(coefficients); n++)
    {
        data_t an;
        get_vec_element(coefficients, n, &an);
        *output += an * (data_t)pow((double)*x_input, (double)n);
    }
}
```

You must optimize this calculation. You must avoid the common optimization blockers and redundant computations. You are to experiment and try out different ways of loop unrolling, accumulators, and reassociation.

Edit the polynomial.c file. Add the new versions there.

To measure its performance, you need to call `add_polynomial_function()` in the provided `register_polynomial_functions()` function.

Create as many versions and combinations of these as you can to find the best one!

To Build and Run

This assignment uses CMake to generate build files.

Open a linux terminal and navigate to the root folder of this assignment.

Run

```
# create a folder called build and generate build files from this source directory
# Tell cmake to generate Release version of the project. This is needed when making makefiles
cmake -DCMAKE_BUILD_TYPE=Release -B build -S .
# create the assignment exe using the build files
# --config Release isn't needed if we have makefiles but if we have XCode/Visual Studio then
#   we need this flag to get a Release exe
cmake --build build --config Release
# run exe
./build/Release/benchmark
```

Note that the assignment can be built and run on Windows and Mac platforms too, but your code will be graded on the Ubuntu Linux platform, so you must make sure your assignment builds and runs well on it.

The measurements won't work on WSL (Windows Subsystem Linux) 1 because it is emulating Linux. WSL 2, however, will work because it implements the Linux kernel.

You need to be running Ubuntu Linux on an actual machine or use WSL 2 on Windows in order to verify that your assignment has no problems.

Grading Rubric

- ☐ [**core**] Submitted a version of the polynomial function that is faster than the naive version.
- ☐ [**core**] Submitted a version of the polynomial function that is fastest you could find from experimentation.
- ☐ [**core**] All versions avoid the `memory aliasing optimization blocker` and eliminate unneeded memory references
- ☐ [**core**] All version remove loop inefficiencies
- ☐ [**core**] All versions avoid `procedure call optimization blocker` by eliminating all external function calls within the loops
- ☐ [**core**] Created a version of polynomial with Nx1 loop unrolling where N is 2 or greater
- ☐ [**core**] Created a version of polynomial with NxM loop unrolling with accumulators where N & M is 2 or greater
- ☐ [**core**] Created a version of polynomial with customized reassociation
- ☐ Created 3 or more versions of Nx1 loop unrolling to see the gains of reducing the bookkeeping overhead of loops
- ☐ Created 3 or more versions of NxN loop unrolling to see the gains of instruction level parallelism and out of order execution
- ☐ Created at least 1 version that combines reassociation with loop unrolling
- ☐ When running the program 3 times, at least one version achieves 1.5 cycles per element or better
- ☐ All source files compile without warnings and without errors.
- ☐ Created a properly filled out `GradingHelper.md` file. Details described below.
- ☐ Correct files submitted. No unnecessary files submitted.
- ☐ **All** Source Code has a proper header comment: Name, Assignment Name, Course Number, Term & Year.

Scores for Exercises will be given as the following:

Score	Assessment
Zero	Nothing turned in at all
Failing	Attempted Something
Rudimentary	Close to meeting core requirements
Satisfactory	Meets all of the core requirements
Good	Clearly meets all requirements
Excellent	High quality, well beyond the requirements

Create a GradingHelper file

Create a **`GradingHelper.md`** markdown file that defines the following.

1. Your name, assignment name/number, course name, term
2. Section describing anything incomplete about the assignment
3. Section describing something you're proud of about the assignment
4. Section that lists the File Name and Line Number for all code requirements
    - one example is enough if there are multiple instances

Submission

Submit your `polynomial.c` and `GradingHelper.md` files on the course site