Part C

2.3.1 Analysis

In this part, our task is to modify ncopy.ys and pipe-full.hcl with the goal of making ncopy.ys run as fast as possible. The difficult point lies in how to find the extra overhead of the pipeline and how to avoid it in a proper way.

Our optimization steps are as follows.

Add iaddl Instruction to pipe-full.hcl

We extended the processor to support a new instruction: iaddl like what we have done in part B. In this way, we avoided extra steps to save a constant in a register. After optimizing the program by adding the instruction iaddl, our CPE test reached 13.96.

4-Way Loop Unrolling

Since predicting loops takes a lot of time, we choose to perform “loop unrolling” to minimize this overhead. “4-Way Loop Unrolling” is to do 4 loops each time and update the relevant data every 4 loops. When the length is less than 4, we change to the remaining part which is still in a loop way. In this way, our CPE test reached 11.28. Therefore, we can see “loop unrolling” is an efficient way for pipeline optimization.

10-Way Loop Unrolling

After 4-Way Loop Unrolling, we consider the more way we unroll the loops, the better performance we will have. So, we tried 10-Way Loop Unrolling and the implementation is the same as the last step. However, our CPE test only reached 11.21. Performance improved, but not significantly.

Increase the Number of Registers

We noticed that there exists stall between reading the val from the src and testing if the val is less than zero in each loop. After unrolling the loop, we can use two registers to store the val from src. So in each loop, the val we test has already been read in the last loop. In this way, our CPE test reached 10.51, which is a significant improvement.

Combine 10-Way Loop Unrolling and 4-Way Loop Unrolling

When taking CPE test, it can be seen that when the input is small, the performance of 10-way loop unrolling is not that useful. Thus, we have to optimize the remaining part. Taking the 4-way loop unrolling we tried before into account, we choose to change the remaining part to another loop unrolling. Fortunately, our CPE test reached 10.16.

2.3.3

Evaluation

Our code for ncopy.ys runs correctly with YIS and our pipe-full.hcl passes all tests in y86-code and ptest. Here are all the test screenshots.

PartC: regression test(with iaddl included)

PartC::benchmark test

PartC:correctness test

PartC:CPE test

Note: our CPE test has reached 10.16 and have a score of 56.1. Though we didn’t score a full mark, but we …

As our optimization steps in the Analysis section，our modifications of ncopy.ys are as follows.

Add iaddl

Use iaddl to avoid using a register to save a constant while changing the value in a register like count++, len--, src++ and so on.

Loop unrolling. Combine 10-way and 4-way

First, we enter 10-way loop unrolling part.

We test whether len (%edx) is less than 10

If so, go to Remainloop part which is 4-way loop unrolling

Otherwise, we loop 10 times and in the end we enter Npos10 part in which we update the data of src (%ebx) and dst (%ecx) and test whether len (%edx) is less than 10 again to choose whether take another 10-way loop.

The 4-way loop unrolling in the Remainloop part is the same as 10-way loop unrolling.

We test whether len (%edx) is less than 4

If so, go to Remain part which is traditional loop part.

Otherwise, we loop 4 times and in the end we enter Npos4 part in which we update the data of src (%ebx) and dst (%ecx) and test whether len (%edx) is less than 4 again to choose whether take another 4-way loop.

Last part is Remain, a traditional loop part.

We update the data of src (%ebx) and dst (%ecx) and test len (%edx) in every loop.

Increase the Number of Registers

Two registers (%esi and %edi) are used alternately for each loop section. In every loop, one store the current val and the other read the val we need to test in the next loop.

Conclusion

In this project, our group successfully completed the three part.

In Part A, we transferred three functions about linked list in example.c into Y86 code with the basic knowledge of the Y86 assembly language

In Part B, we added the iaddl instruction and the leave instruction to Y86’s sequential design by modifying the HCL file after a deep exploring into the stages of these two instructions.

In Part C, we improved and optimized the performance of the pipeline processor with proper ways including adding instruction, using loop unrolling, adding register and changing the instruction order.

The problems encountered in the process and our achievements are as follows

Problems

[In this part you can list the obstacles you met during the project, and better add how you overcome them if you have made it.]

Get familiar with new language and tools. Since Y86 assembly language is s new language for us. Before we start out project, we have to learn about the basic language knowledge including gramma and instruction meaning.

Take care of the use of stack, registers, and variables. Assembly language operate on registers and memory directly, it is critical to take care of all these.

Take care of the readability of our codes. Assembly language is less readable than high-level language, so we added detailed notes in our codes.

Get the meaning of new instruction and figure out how to implement it. In the process we learned from CS:APP to further explore the instruction implementation.

Analyze the pipeline performance and relevant factors. Explore and design proper ways to optimize it. As our optimize process described in Part C is not a smooth ride, and in the end we did not score a full mark in CPE test. It shows that our pipeline still has room for optimization. If there is an opportunity, we will continue to explore different ways to optimize pipeline performance.

Achievements

[In this part you can list the strength of your project solution, like the performance improvement, coding readability, partner cooperation and so on. You can also write what you have learned if you like.]

In the process of project, both two group members all made contributions to the project and report part, which is a good cooperation. Besides, we both have found it a very interesting and meaningful project which helped us know better about the implementation of a pipelined Y86 processor.