CLC - Vector Processing Simulator

Colt Uhlenhopp & Jack Utzerath

Grand Canyon University

CST-307

Professor Citro

4/26/2023

Vector Processing Simulator

Overview:

In future systems, we expect to see heterogeneous computing platforms constructed out of heterogeneous CPUs. We have begun to see some appear in the embedded processing market in systems that contain both floating point DSPs and microcontroller CPUs in a multichip module package. Assume that you have three classes of CPU:

CPU A: A moderate speed multi-core CPU (with a floating point unit) that can execute multiple instructions per cycle.

CPU B: A fast single-core integer CPU (i.e., no floating point unit) that can execute a single instruction per cycle.

CPU C: A slow vector CPU (with floating point capability) that can execute multiple copies of the same instruction per cycle.

Assume that our processors run at the following frequencies:

CPU A can execute 2 instructions per cycle, CPU B can execute 1 instruction per cycle, and CPU C can execute 8 instructions (though the same instruction) per cycle. Assume all operations can complete execution in a single cycle of latency without any hazards.

All three CPUs have the ability to perform integer arithmetic, though CPU B cannot perform floating point arithmetic. CPU A and B have an instruction set similar to a MIPS processor. CPU C can only perform floating point add and subtract operations, as well as memory loads and stores. Assume all CPUs have access to shared memory and that synchronization has zero cost.

The task at hand is to compare two matrices X and Y that each contain 1024 × 1024 floating point elements. The output should be a count of the number indices where the value in X was larger or equal to the value in Y.

1. **Describe how you would partition the problem on the 3 different CPUs to obtain the best performance.**

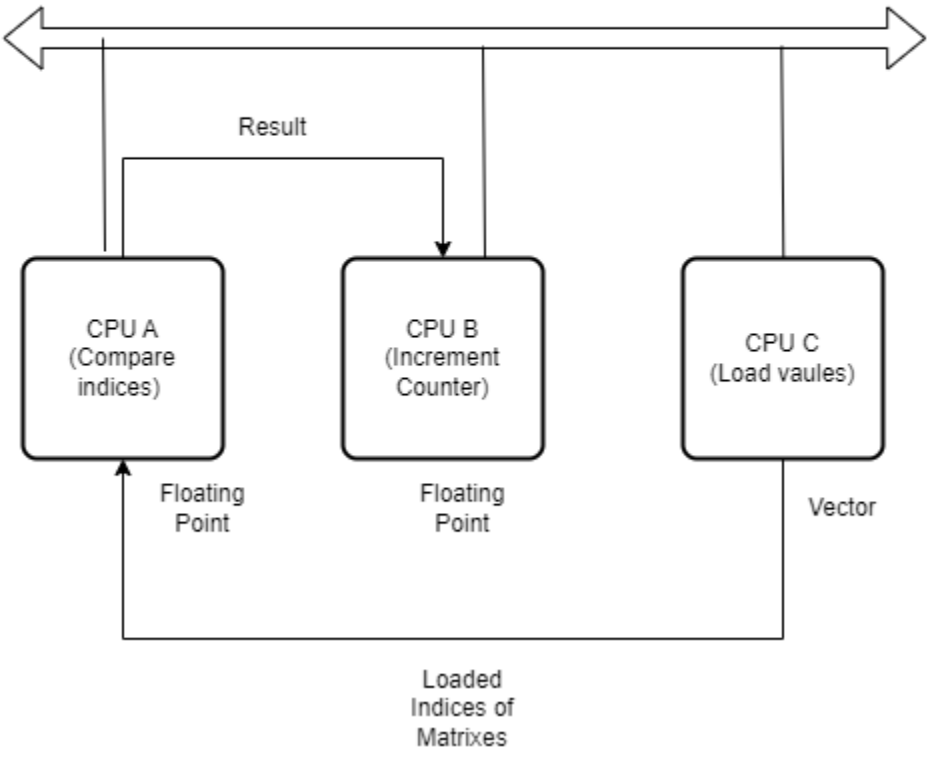
To partition the problem using the 3 different CPUs in order to obtain the best performance, we can split up the amount of work into 3 parts, one for each CPU. One processor will load the values into another CPU, one processor will compare the values of X and Y and store the result, and one processor will count how many indices had values of X larger than Y (determined by the previous processor). Due to the constraints of the processors, only certain processors can perform specific tasks. To determine which processors will take on which task, we need to determine the capability of each processor doing the tasks, and if two processors are capable of doing the task, then determine which one is most efficient at it.

To find the CPU that loads in the floating numbers to be compared, the CPU has to be able to handle floating points. This automatically rules out CPU B since it cannot handle floating numbers. That leaves CPU A and C. CPU A loads 2 instructions per cycle, and CPU C can load 8 instructions per cycle. However, CPU C can only load 8 instructions if they are the same instructions and this CPU is 4 times slower than CPU A . Based on the next task of comparing the values, we know that CPU C has to be chosen to complete this task.

The comparison between the two values is the easiest task to give to a CPU. CPU B cannot deal with floating numbers and CPU C can only do addition and subtraction instructions. So this task is given to CPU A to complete.

This leaves the counter to CPU B. Since this CPU doesn’t work with floating points, it makes sense for this processor to complete this simple increment task. CPU C will load the values, CPU A will compare and store the result, then CPU B increments if need be.

Jack Utzerath
Vector Processing Simulator



2. **What kind of instruction would you add to the vector CPU C to obtain better performance?**

To obtain better performance of the vector processor, an instruction that should be added to CPU C is the following:

If (x[i][j] > y[i][j]){
Count++; }

This is an increment instruction. Instead of CPU B being the processor that will count how many indices had values of X larger than Y, we can instead add this increment instruction to CPU C. The instructions of CPU C may be limited, but because it is able to perform addition and subtraction operations, this instruction would work perfectly. By adding this instruction, it will increase the performance because now it is able to increment a count along with loading in values in the same cycle. CPU B can only execute 1 instruction per cycle compared to CPU C's 8

instructions per cycle. Although CPU B is a fast single core processor, CPU C can simply do the processor's job in the same instruction cycle as it does its own job of loading the floating point values. By using the CPU C’s parallel pipelining to our advantage, this will increase the performance.