Chapter 2: Case study

Version 5: Assembly code [advanced]

The assembly code of the innermost loop is very straightforward. There are only 3 instructions related to the loop counters. There are 2 memory accesses, 4 operations that permute the elements, and then 8 + 8 useful vector operations (`vminps` and `vaddps`).

```
LOOP:
        vmovaps     (%rdx,%rax), %ymm2
        vmovaps     (%r12,%rax), %ymm3
        addq        $32, %rax
        vpermilps   $177, %ymm2, %ymm0
        cmpq        %rax, %rcx
        vperm2f128  $1, %ymm3, %ymm3, %ymm13
        vaddps      %ymm2, %ymm3, %ymm15
        vpermilps   $78, %ymm3, %ymm14
        vaddps      %ymm0, %ymm3, %ymm3
        vpermilps   $78, %ymm13, %ymm1
        vminps      %ymm15, %ymm11, %ymm11
        vminps      %ymm3, %ymm7, %ymm7
        vaddps      %ymm14, %ymm2, %ymm3
        vaddps      %ymm14, %ymm0, %ymm14
        vminps      %ymm3, %ymm10, %ymm10
        vaddps      %ymm13, %ymm2, %ymm3
        vaddps      %ymm13, %ymm0, %ymm13
        vaddps      %ymm1, %ymm2, %ymm2
        vaddps      %ymm1, %ymm0, %ymm0
        vminps      %ymm14, %ymm6, %ymm6
        vminps      %ymm3, %ymm9, %ymm9
        vminps      %ymm13, %ymm5, %ymm5
        vminps      %ymm2, %ymm8, %ymm8
        vminps      %ymm0, %ymm4, %ymm4
        jne         LOOP
```

Analysis

Let us try to do a bit more careful study of all instructions that we have in the innermost loop and how they might be scheduled by the CPU. Here is a table of the instructions and the execution ports that they could use (again derived from Agner Fog's Instruction tables):

Instruction	Count	Execution ports
`vaddps`	8	0, 1
`vminps`	8	0, 1
`vmovaps`	2	2, 3
`vpermilps`	3	5
`vperm2f128`	1	5
`addq`	1	0, 1, 5, 6
`cmpq`	1	0, 1, 5, 6
`jne`	1	6

If the `vaddps` and `vminps` instructions keep execution ports 0 and 1 busy for 8 clock cycles, we can see that there are plenty of execution ports that the other instructions could use. For example, `vpermilps` and `vperm2f128` could use port 5 for 4 cycles, `vmovaps` could use ports 2 and 3 for 1 cycle, and the remaining operations could use port 6 for 3 cycles. While this is a rather simplified view of the internal workings of the CPU, this suggests that there is a good reason to expect near-100% efficiency: the relevant instructions `vaddps` and `vminps` are the bottleneck here and everything else should be easy to do on the side.

Interactive assembly

Here is the Compiler Explorer version to play with.

- What happens if you change the target architecture to something that does not support `AVX` , e.g., by switching to `-march=x86-64` ?
- Here we provide just the `swap` definitions to experiment with. Can you make the implementation more portable by replacing the intrinsics with an appropriate GCC built-in function? If you do this correctly, the resulting assembly should be the same. Check that this works also for generic `-march=x86-64` architecture.