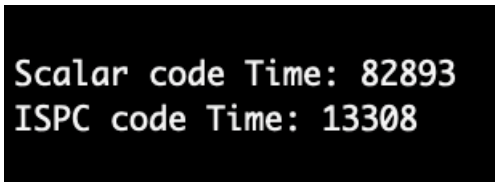
Lab-2 ISPC Programming Assignment

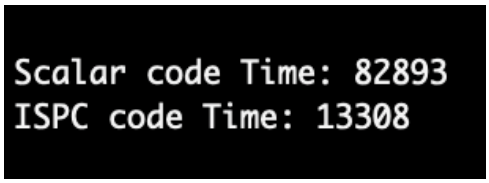
소프트웨어학과 201921085 곽수정

1) Performance comparision: between the **scalar** and **vector** versions

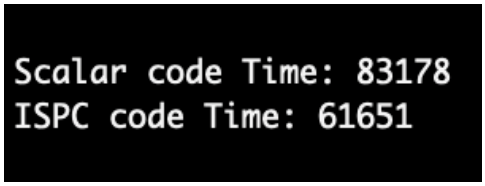


Scalar 방식과 ISPC를 이용한 programming 방식을 비교해본 결과, 여러 instance가 동시에 수행하는 ISPC 코드가 성능이 더 좋았습니다. 같은 instruction에 대해 여러 ALU가 동시에 연산을 수행하기 때문임을 알 수 있었습니다.

2) Performance comparision: between the **interleaved** and **blocked** assignments of program instance



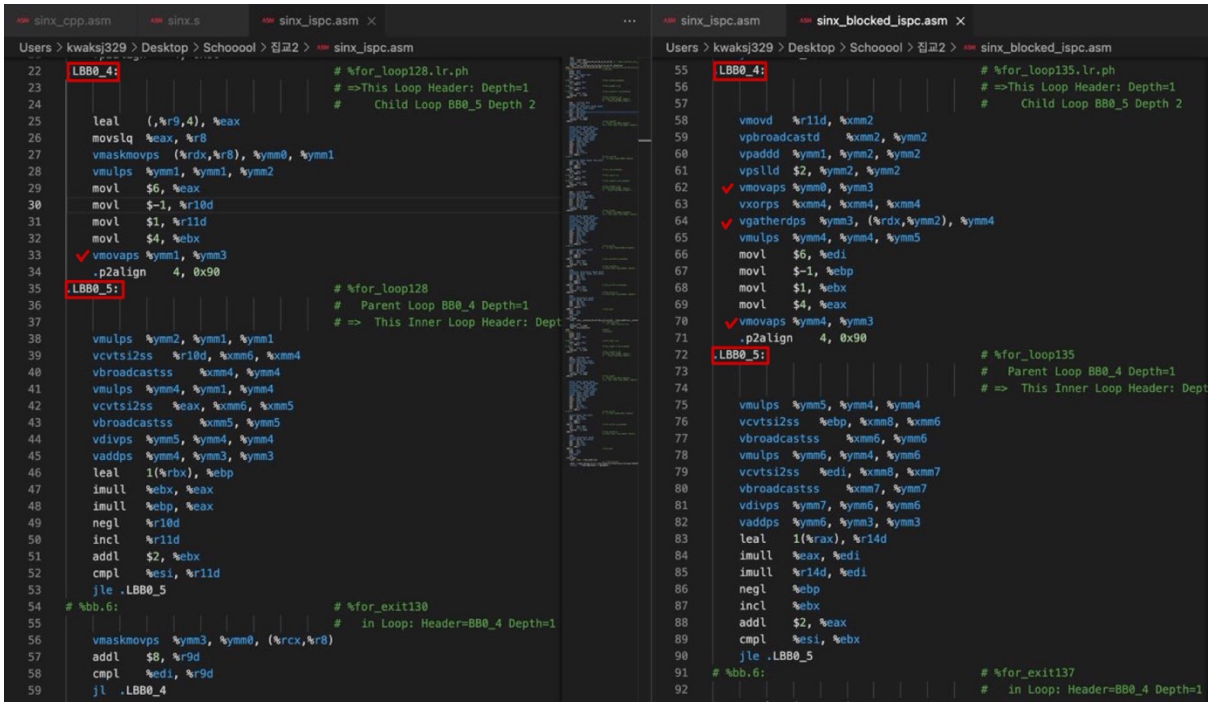
Interleaved assignments



Blocked assignments

같은 프로그램의 loop iterations를 각각 interleaved assignments와 blocked assignment로 실행시켜본 결과 interleaved assignments 방식의 성능이 훨씬 좋은 것을 알 수 있었습니다. Interleaved assignments는 한번에 program count만큼의 instance들이 실행될 때, array의 element들을 한번에 가져와서 instance들이 한꺼번에 실행될 수 있습니다. Blocked assignments에서는 전체 N을 program count로 나눠서 하나의 gang이 수행할 양이 결정됩니다. 따라서 동시에 실행되는 instance의 value들이 contiguous하게 메모리에 위치해 있어서 locality를 가지는 interleaved 방식과는 다르게 blocked 방식에서는 동시에 실행되는 instance들의 data들이 메모리에 non-contiguous하게 위치해있어 더 복잡하고 비용이 많이드는 instruction을 수행하게 됩니다.

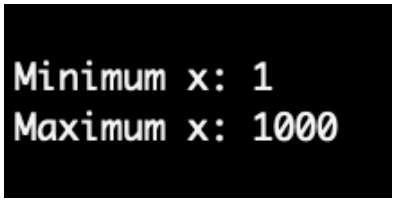
Interleaved와 blocked 방식의 assembly code를 비교해보았을 때도 interleaved는 221줄의 코드를, blocked는 579줄의 코드를 가져 더 많은 양의 instruction을 수행하여 blocked의 성능이 떨어지는 것을 알 수 있었습니다.



가장 outer for문에 대한 interleaved (왼쪽), blocked (오른쪽) assembly code를 살펴보았을 때, 두 코드 다 본격적으로 for문 안의 연산을 하는 LBB0_5의 instruction은 레지스터만 조금 다를 뿐 instruction의 종류와 수는 같은 것을 볼 수 있었습니다. 하지만 for문을 실행하기 이전에 LBB0_4에서 instance를 위한 value들을 가져올 때 interleaved에서보다 blocked에서 실행되는 instruction의 수가 더 많았으며, blocked에서는 packed vector load를 수행하는 vmovaps뿐만 아니라 gather를 위한 vgatherdps instruction을 더 수행해야하기 때문에 interleaved 방식이 더 성능이 좋은 것을 알 수 있었습니다.

Assembly code에서 for문을 시작할 때와 마찬가지로 exit하는 부분에서도 blocked 방식의 instruction이 non-contiguous한 데이터 때문에 훨씬 많은 instruction을 수행해야 했음을 알 수 있었습니다. 따라서 blocked 방식에서 연산을 수행하는 부분보다는 instance를 위한 value를 가져오고, 저장하는 부분에서 많은 instruction이 추가된 것을 볼 수 있었습니다.

3) Finding a **minimum** and **maximum** value in an array



Array의 최소값과 최대값을 찾기 위해 먼저 srand 함수를 사용하여 x 배열에 1부터 1000까지 랜덤한 int형 값을 넣어주었습니다. min_ispc() 함수와 max_ispc() 함수에서 최소, 최대값을 찾기 위해 현재 돌고있는 instance가 가진 최소, 최대 x 값을 reduce_min과 reduce_max 함수를 통해 알아냈습니다. 이를 각각 temp에 담아 이전 cal_min 값보다 temp가 더 작은지 비교하여 더 작다면 cal_min

에 저장하고, 이를 반복하여 가장 작은 값을 xmin 변수에 저장하도록 하였습니다. 최대값의 경우에도 temp에 저장된 값과 이전에 저장한 cal_max와 비교하여 temp가 더 크다면 이를 업데이트해주고, 반복하는 과정을 거쳐 가장 큰 값을 xmax 변수에 저장하도록 하였습니다.

따라서 현재 돌고 있는 instance들의 min/max 값을 함수로 찾아서, array의 최소값과 최대값을 찾을 수 있었습니다.