Formulas to rederive and understand

- Number of accumulators for reduction based on latencies and throughput
- MMM block size based on cache size
- compiler optimizations
    - scalar replacement

Instruction Level Parallelism: Number of accumulators for reduction

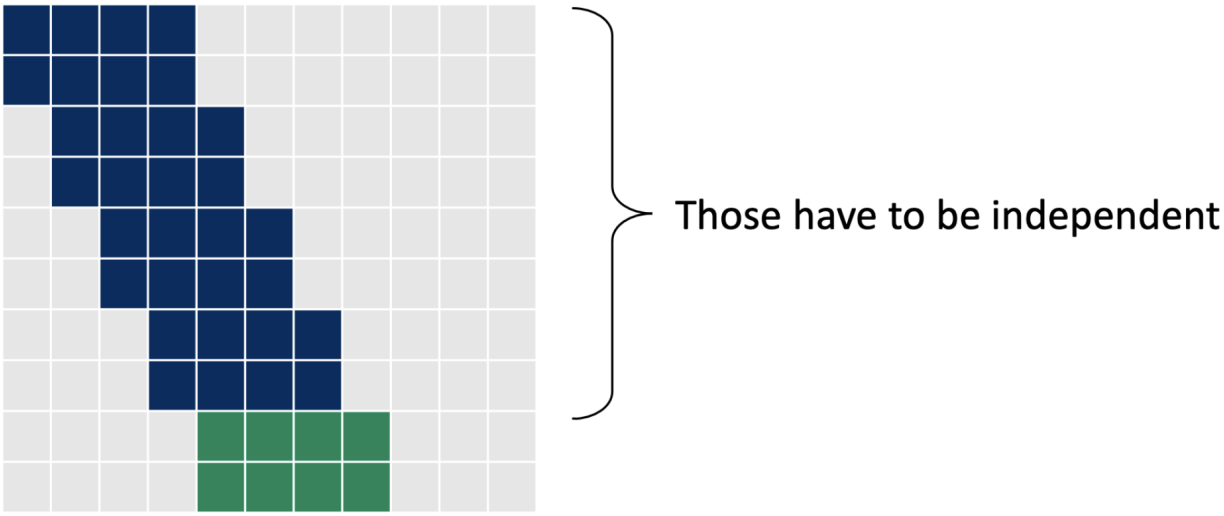
Reduction code:

```
double reduce(const double* v, const size_t n) {
    double sum = 0.0;
    for (int i = 0; i < n; ++i) {
        sum = sum + v[i];
    }
    return sum;
}
```

Skylake microarchitecture floating point add

- Throughput: 2 flops/cycle (given by the number of ports supporting instruction)
- Latency = 4 cycles

Why 8?



Based on this insight: K = #accumulators
= ceil(latency / cycles per issue)
= ceil(latency * throughput)

Here (FP Mult): K = ceil(4 / 0.5) = ceil(4 * 2) = 8

Number of accumulators = ceil(latency * throughput) = ceil(4 * 2) = 8
Explanation: The processor can issue 2 flops / cycle and each flop requires 4 cycles to complete. In order to achieve best performance, we need to issue flop instructions each cycle. Since the first instruction will take 4 cycles to complete, we need to create separate accumulators for each issued flop instruction during this period in order to use ILP. Thus for 4 cycles, we issue 2 flop instruction and require 8 accumulators in total. After the 4 cycles the same accumulators can be reused since the previous operation has terminated.

Compiler Optimizations

- **Code Motion** (e.g., Loop-invariant code motion): Remove redundant computations
  - Pull computation out of loop
- **Strength Reduction**: Replace costly operations with simpler ones
  - Power by bit shift

- **Scalar Replacement**: Copy memory variables that are reused into local variables
  - Prevents memory aliasing issues

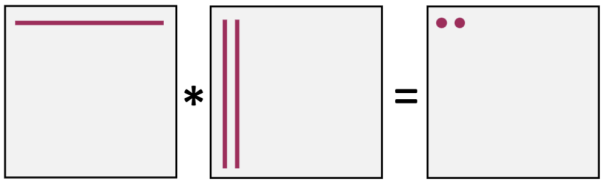
Tile Size for GEMM

Cache Miss Analysis MMM

C = A*B, all n x n

Assumptions: cache size γ << n, cache block: 8 doubles, only 1 cache, row-major order

Triple loop:

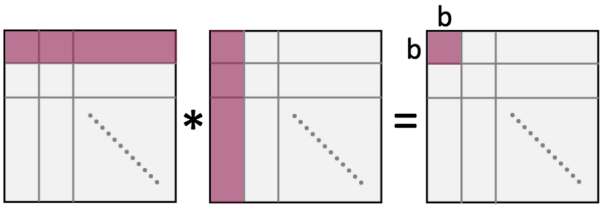


1. entry: n/8 + n = 9n/8 cache misses

2. entry: same

Total: $n^2 * 9n/8 = 9n^3/8$

Blocked (six-fold loop): block size b, 8 divides b



1. block: nb/8 + nb/8 = nb/4 cache misses

2. block: same

Total: $(n/b)^2 * nb/4 = n^3/(4b)$

How to choose b?

The above analysis assumes that the multiplication of b x b blocks can be done with only compulsory misses. This is achieved with $3b^2 \leq \gamma$.

b = sqrt(γ/3) which yields about sqrt(3)/(4*sqrt(γ)) * n^3 cache misses, a gain of ≈ 2.6*sqrt(γ)

I(n) = O(sqrt(γ))