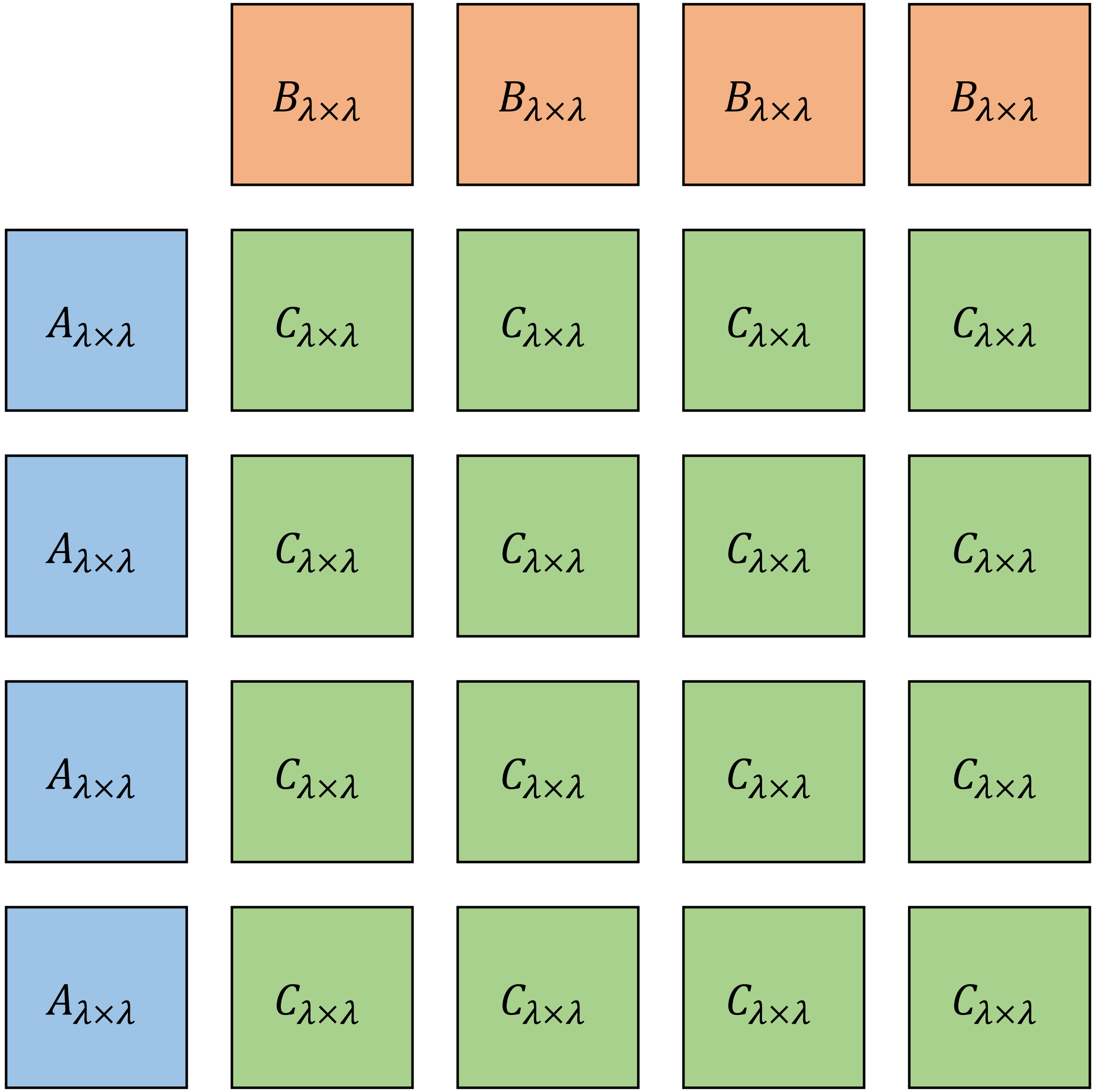
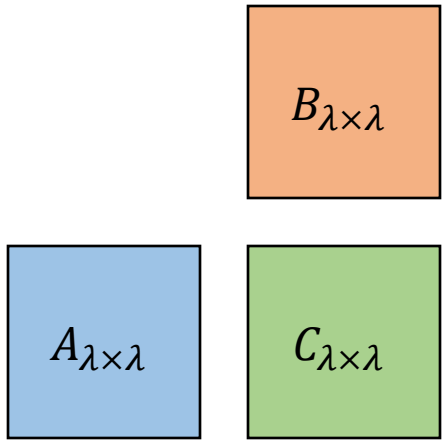
Option A: 1 Matrix per vector register

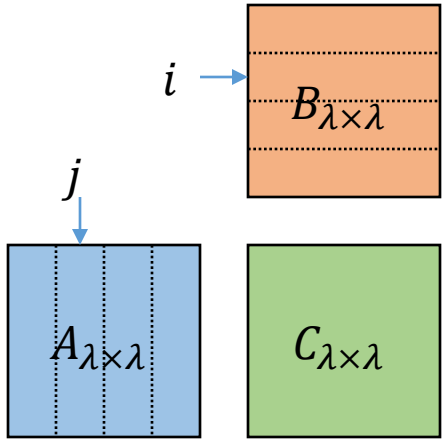


- An L-word vector register holds an $\lambda \times \lambda$ matrix, $\lambda = \sqrt{L}$
- 16 registers hold a $4\lambda \times 4\lambda$ panel of $\boldsymbol{C}$
- 4 registers hold a $4\lambda \times \lambda$ panel of A
- 4 registers hold a $\lambda \times 4\lambda$ panel of B
- We compute $\boldsymbol{C}_{4\lambda\times 4\lambda} \leftarrow \boldsymbol{A}_{4\lambda\times\lambda} \times \boldsymbol{B}_{\lambda\times 4\lambda} + \boldsymbol{C}_{4\lambda\times 4\lambda}$
- Total of $4\lambda \times 4\lambda \times \lambda = 16\lambda^3$ multiply-adds
- Minimum time = $\lambda\Delta$ cycles
- Maximum computation rate $R = \frac{16\lambda^3}{\lambda 4} = 4\lambda^2 = 4L$
- This is the upper bound with 16 registers for $\boldsymbol{C}$
- Total of $8\lambda^2$ elements loaded ($^{8\lambda}/_{\Delta}$ words/cycle)
- Computational intensity $\eta = \frac{16\lambda^3}{8\lambda^2} = 2\lambda$ madds/word
- This works for $L = 4$, 16, 64, … words
- Single- and double-precision are incompatible

Option A: Compute instructions

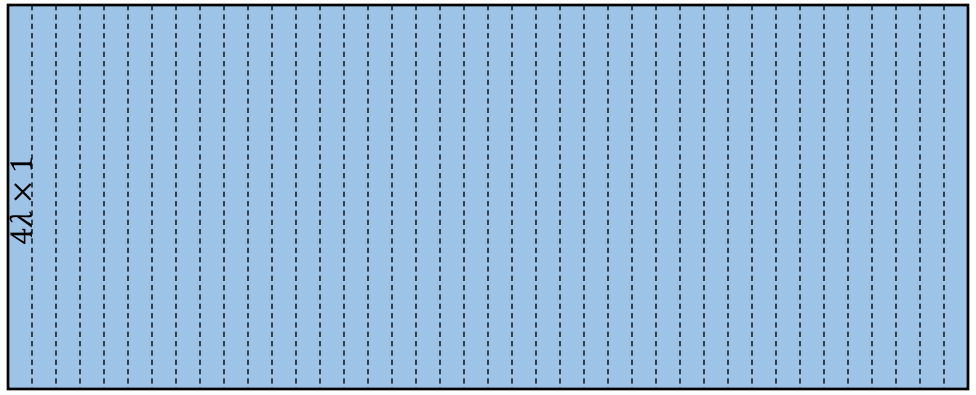


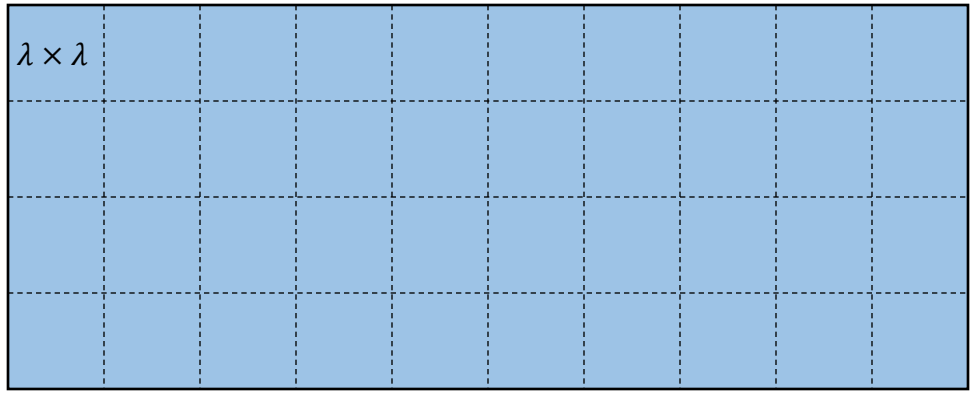
- rank-λ update (matrix multiply)
  - $\boldsymbol{C}_{\lambda\times\lambda} \leftarrow \pm\boldsymbol{A}_{\lambda\times\lambda} \times \boldsymbol{B}_{\lambda\times\lambda} \pm \boldsymbol{C}_{\lambda\times\lambda}$
  - Computations: λ^3 madds/instruction
  - Latency: $\lambda\Delta$
  - Must dispatch/issue 4 computational instructions/operations every λ cycles to achieve maximum computation rate ($4L$ madds/cycle)



- rank-1 update (outer product)
  - $\boldsymbol{C}_{\lambda\times\lambda} \leftarrow \pm\boldsymbol{A}^{j} \times \boldsymbol{B}_{i} \pm \boldsymbol{C}_{\lambda\times\lambda}$ (usually $i = j$)
  - Computations: λ^2 madds/instruction
  - Latency: Δ
  - Must dispatch/issue 4 computational instructions/operations every cycle to achieve maximum computation rate ($4L$ madds/cycle)
  - rank-λ update can be cracked into λ rank-1 updates

Option A: Software impact





- Before the compute kernel is executed, the input matrices are *packed* to optimize streaming performance
- $\boldsymbol{A}$ panels are typically formatted as column-major matrices of shape $4\lambda \times K$
- $\boldsymbol{B}$ panels are typically formatted as row-major matrices of shape $K \times 4\lambda$
- For Option A to work, both $\boldsymbol{A}$ and $\boldsymbol{B}$ must be packed into $\lambda \times \lambda$ blocks
- Both the compute and packing kernels must be modified to support matrix operations
- $\boldsymbol{C}$ panel must also be reformatted for load/store