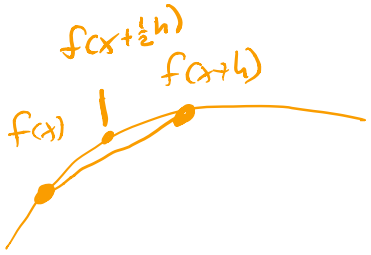
Week 6 Day 1: vectorization

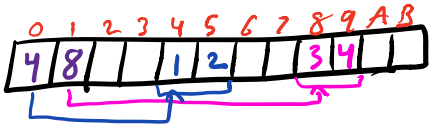
Notes from before:

- Issue on Quiz 5.2 (5.1?)
- Projects
- FD vs CD for search

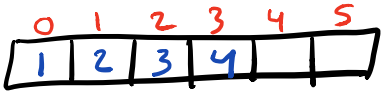


Memory Layout:

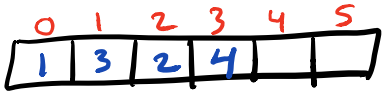
$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$



↑ Vector of vectors (c++)



Row-major "C"-style (C, Python)



Column-major "F"-style (Fortran, matlab)

Note:

$$A^T = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$$

Also note:

$A[row, column] \Rightarrow A = \begin{bmatrix} A_{0,0} & A_{0,1} \\ A_{1,0} & A_{1,1} \end{bmatrix}$

but this is y, x in a graph!

C-style

$A[a, b]$

↑ slow ↑ rapid

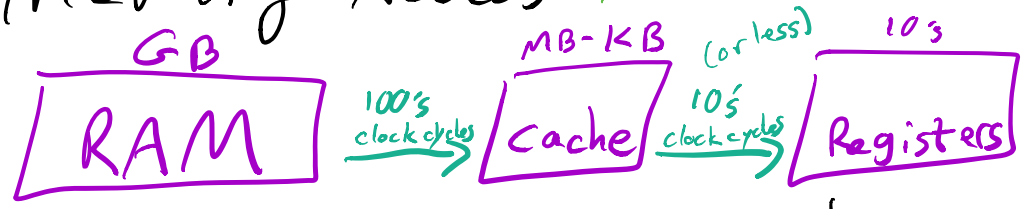
F-style

$A[a, b]$

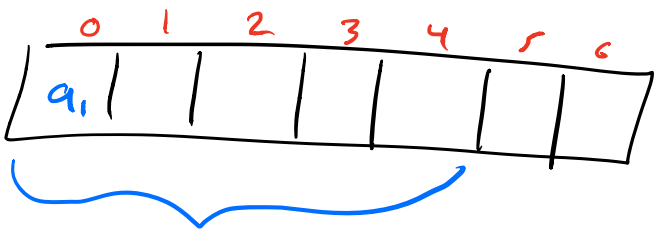
↑ rapid ↑ slow

Memory Access

note: several levels



Computers love sequential access!



Computers tend to load wide batches of memory to your cache

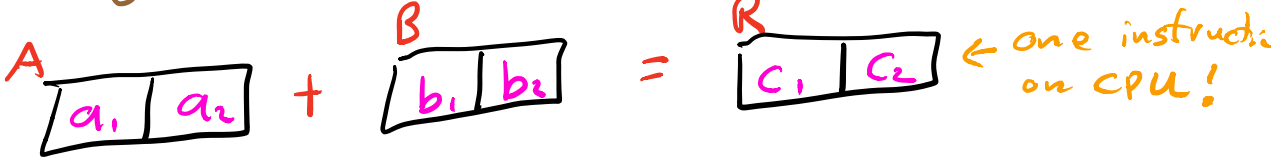
RAM → Cache

asking for 0-4 is now much faster, but 5+ is still slow

Hardware vectorization

Cache → Registers

Modern hardware can compute truly in parallel on some registers!



You might be able to take advantage of this, but only with a proper data layout.

	32 bit	64 bit	width
SSE	4	X	128 bits
SSE2	4	2	128 bits
AVX(1+2)	8	4	256 bits
AVX512	16	8	512 bits

Take away message: if you avoid explicit loops, you'll be ready for vectorization!

Note: From now on, we'll mostly use the term "vectorization" in the MATLAB sense, for Array-at-a-time programming.

But at least now you know what real vectorization is!