# Notes on DoD

## General Notes:

* Watch polymorphic procedures from Jonathan blow
* OOP encourages AoS (Array of Struct) pattern which is said to be slow

## Research:

* Streaming prefetch bonus and general CPU bonuses
* How to analyze compiler
  + Check mike acton gdc unity talk

## Mike Acton talk:

* Where there is one there many (solve the most common problem first)
* The more context you have the better the solution, don’t throw away data or constraints that you need
* OOP confuses Maintenance with Understanding properties of data at the benefit of very little bonus to maintenance the cost is it makes it hard to solve performance problems

## Post Hand in:

### AoS vs SoA

* Maybe try to apply parallelization to AoS vs SoA example to see if there is a bigger boost to one memory access pattern than another
* Research stack/heap
* Mess around with Cpp
* Experiments
  + Mess around with the iterations
  + Make the same experiment with objects and figure out if there is a measurable difference between thing being stored on the stack vs the heap
  + Introduce more/less data/variables
  + Introduce different kinds of data
  + Introduce operations together with the data (which is typically against the data-oriented philosophy)

### OOP vs DoD example (Stocks)

* Make everything into structs
* Only use flat linear arrays and see if separating huge chunks into separate arrays yield better results
* Look at code metrics
* Make unit tests to show difference

### Disposition

* Introduce the subject
  + Why is it interesting, and what led me to it?
    - End user
  + How I wanted to approach it
    - I really wanted to be able to look at the compiler, why that wasn’t feasible
* Mike Acton GDC Unity
* SoA vs AoS
  + I had some correspondence with a youtuber who made one of the video examples in my references and he said it gave him very similar and trivial performance boost of max 5 %
  + Intel compiler (-03) optimization
  + SIMD and vectorization
  + Take away: compilers are bad?
* Since I couldn’t look under the hood properly, I just tried to apply the basic principles to make a DoD version of Pers Stock example
* Show the project
  + Structs/No classes (stack instead of heap), don’t think it’s properly implemented
  + Linear data structures (basically only arrays), most of the performance boost
  + Data and operation/function/method separation
  + Minimal states (Look for bools)
  + Metrics: DOD example was better by 6 points but…
* Conclusion
  + C# is not suited well for data-oriented design because of garbage collection and other runtime Blackbox systems
  + Because everything needs to be in a class or struct it’s hard to organize code the way I wanted to
  + Maintainability….Metrics
    - Nobody didn’t finish a project because writing the code took too long
  + Leaving me with a big question: How do compilers work?

## References

### Mike Acton Talk:

<https://www.youtube.com/watch?v=rX0ItVEVjHc>

<https://www.youtube.com/watch?v=p65Yt20pw0g>

### Stoyan Nikolov Talk:

<https://www.youtube.com/watch?v=yy8jQgmhbAU>

### Jonathan Blow Talks:

<https://www.youtube.com/watch?v=TH9VCN6UkyQ&list=PLmV5I2fxaiCKfxMBrNsU1kgKJXD3PkyxO>

### What Is CPU Cache? Article:

<https://www.makeuseof.com/tag/what-is-cpu-cache/>

### Data-Oriented Design (Or Why You Might Be Shooting Yourself in The Foot With OOP):

<http://gamesfromwithin.com/data-oriented-design>

### Forum Posts:

<https://stackoverflow.com/questions/29192679/memory-layout-optimization-in-c-sharp>

<https://stackoverflow.com/questions/56255090/does-c-sharp-allocate-array-of-structs-as-structure-of-arrays-in-memory>

### AoS vs Soa:

<https://www.youtube.com/watch?v=ScvpoiTbMKc>

<https://www.youtube.com/watch?v=Dj3lkfIEUyA&t>