Lecture 03 (Introduction to Optimisations)

1 Ways to Compile

1. Do nothing (aka interpreter)
2. Substitute input program into the interpreter's implementation - evaluate each statement into something

2 Local Optimisations

1. Evaluate each statement only once, perform jumps to evaluated statement (which is cached)
2. Caching and reuse of generated code

3 Modes of Compilation

1. Static (Ahead-of-Time)
2. Dynamic (Just-in-Time)

3.1 Static

1. No runtime cost
2. Compilation might be slow
3. Not enough information available during compilation

3.2 Dynamic

1. Runtime information during compilation
2. Only relevant portions are compiled
3. Runtime compilation cost is high
4. Multi core processors compile in parallel - advantages »> cost

4 Global Optimisations

1. Span multiple statements and optimise them into a single statement

   i. lazy programmer
   ii. low level syntax is richer
   iii. some optimisations enable more optimisations
2. Order of optimisations also matters (left shift vs mutiply+add) - phase ordering problem

Compiler optimisation is undecidable problem

5 Why Complex ISAs?

1. Transistors can operte in parallel
2. This parallelism is of the order of $O(billion)$

6 Out of Order Super Scalar Processor

1. Executes multiple instructions in parallel
2. Has a dedicated stage to figure out dependencies
3. Instruction level parallelism

7 AOT vs JIT

1. JIT will use more resources during runtime
2. Startup time is quite high in comparison
3. JIT will need to compile every time the program is run: sounds like resource wastage