## Initial Parsing to AST

### File -> AST

Different sections represent multiple paths. Each path converts some sort of raw file to an AST.

|  |  |
| --- | --- |
| Text Reader |  |
| Tokenizer |  |
| AST Creator |  |
| ID Checker |  |
| Type Checker |  |

|  |  |
| --- | --- |
| CIL Parser | Converting stack based CIL to an AST. Assume no need to check types, etc. at this stage. Also resolves dependencies (i.e. mscorlib) |

## AST transformations

### AST -> AST

All of the stages below are visitor based transformations; AST -> AST.

|  |  |
| --- | --- |
| Desugarer |  |
| Common Expression Grouper | Analyzes expression trees used in statements and finds common paths that can be shared between calls, then moves them to some other area. |
| Funroller | Does what one would expect |

## General ISA translation

### AST -> ISA Representation

|  |  |
| --- | --- |
| Order instructions | Expand expressions to be calculated step by step |
| Labeller | Converts structure to linear labels  Constructors allocate memory and zero fields  Generates generic instantiations |
| ISA Converter | Generates representational instructions and extracts data |
| Internal Implementation resolver | Resolves internal implementation to proper calls, whether they be external calls (syscalls or dll calls) or just inserting a function to be called |
| Register Allocator | duh |

## Instruction Emitter

### ISA Representation -> Address Independent Instructions

|  |  |
| --- | --- |
| ISA Optimizer | This is possibly optional? |
| ISA Emitter | Emit the actual opcodes |
| External Call Resolver |  |

## Executable Stage

### Address Independent Instructions -> Binary

|  |  |
| --- | --- |
| Addresser | Handles virtual addressing and pdata |
| Physical addresser |  |
| Executable writer |  |