中国科学技术大学
University of Science and Technology of China

.NET 编译的中间表示

黄业琦，任俊屹，魏剑宇

组长：王瑞凯，wrk15835@mail.ustc.edu.cn
指导老师：张昱，yuzhang@ustc.edu.cn

2019.11.20

报告目录

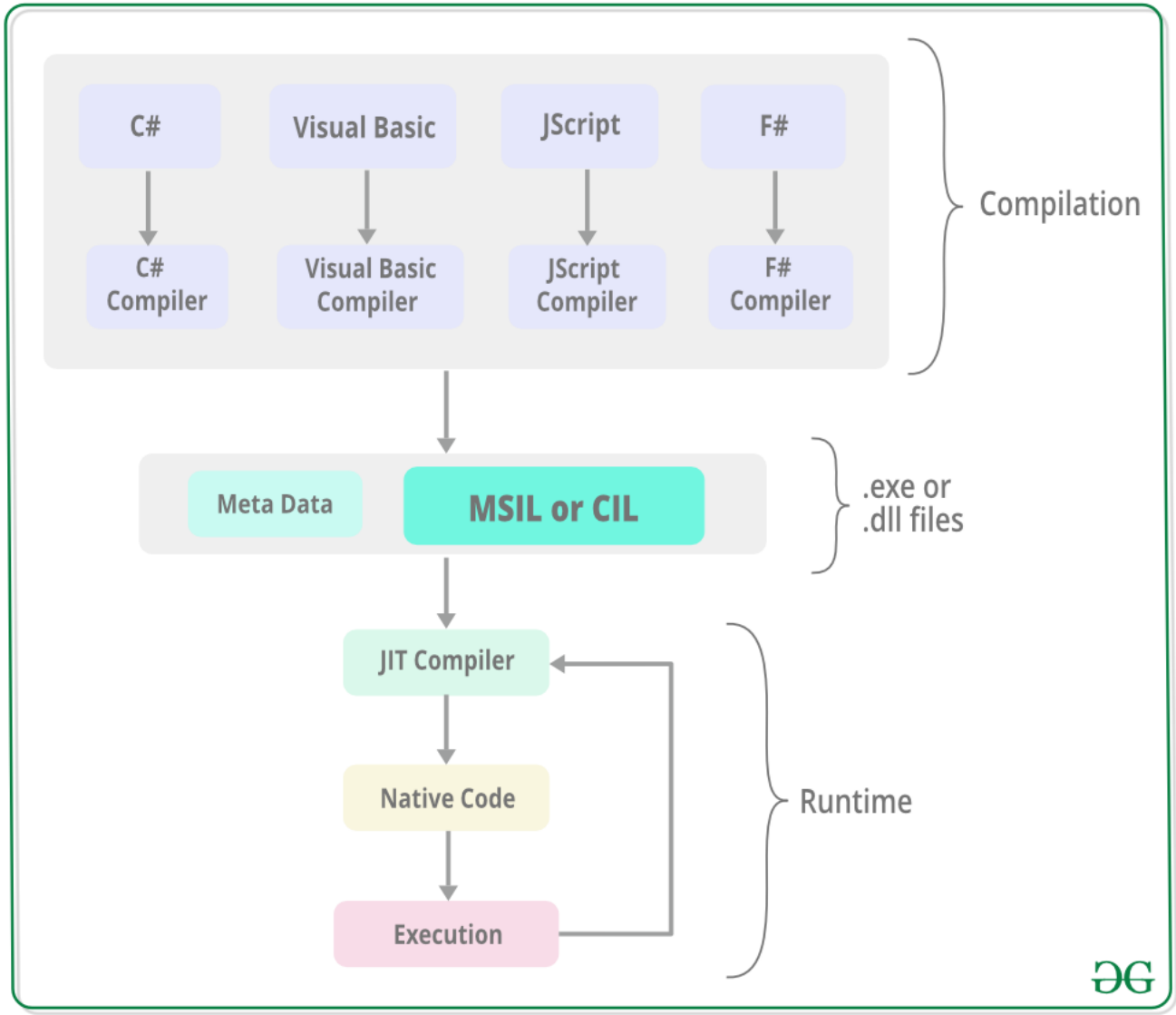
中国科学技术大学
University of Science and Technology of China

报告目录

中国科学技术大学
University of Science and Technology of China

Bytecode & CIL

.NET



Bytecode & CIL

- **Bytecode**
- Source code --(compile)--> Low-level code
- Designed for a software interprete

Bytecode & CIL

CIL example:

```
using System;

namespace HelloWorld
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World!");
        }
    }
}
```

Bytecode & CIL

中国科学技术大学
University of Science and Technology of China

CIL example:

```
//  Microsoft (R) .NET Framework IL Disassembler.  Version 4.5.30319.0



// Metadata version: v4.0.30319
.assembly extern System.Runtime
{
  .publickeytoken = (B0 3F 5F 7F 11 D5 0A 3A )                         // .?_....:
  .ver 4:2:1:0
}
.assembly extern System.Console
{
  .publickeytoken = (B0 3F 5F 7F 11 D5 0A 3A )                         // .?_....:
  .ver 4:1:1:0
}
.assembly HelloWorld
{
  .custom instance void [System.Runtime]System.Runtime.CompilerServices.CompilationRelaxationsAttribute::.ctor(int32) = ( 01 00 08 00 00 00 00 00 )
  .custom instance void [System.Runtime]System.Runtime.CompilerServices.RuntimeCompatibilityAttribute::.ctor() = ( 01 00 01 00 54 02 16 57 72 61 70 4E 6F 6E 45 78   //
>    ....T..WrapNonEx…
  // --- The following custom attribute is added automatically, do not uncomment -------
  //  .custom instance void [System.Runtime]System.Diagnostics.DebuggableAttribute::.ctor(valuetype [System.Runtime]System.Diagnostics.DebuggableAttribute/
  DebuggingModes) = ( 01 00 07 01 00 00 00 00 )

  .custom instance void [System.Runtime]System.Runtime.Versioning.TargetFrameworkAttribute::.ctor(string) = ( 01 00 18 2E 4E 45 54 43 6F 72 65 41 70 70 2C 56   //
>  ....NETCoreApp,V…
  .custom instance void [System.Runtime]System.Reflection.AssemblyCompanyAttribute::.ctor(string) = ( 01 00 0A 48 65 6C 6C 6F 57 6F 72 6C 64 00 00 )    //
  ...HelloWorld..
  .custom instance void [System.Runtime]System.Reflection.AssemblyConfigurationAttribute::.ctor(string) = ( 01 00 05 44 65 62 75 67 00 00 )                  //
  ...Debug..
  .custom instance void [System.Runtime]System.Reflection.AssemblyFileVersionAttribute::.ctor(string) = ( 01 00 07 31 2E 30 2E 30 2E 30 00 00 )             //
  ...1.0.0.0..
  .custom instance void [System.Runtime]System.Reflection.AssemblyInformationalVersionAttribute::.ctor(string) = ( 01 00 05 31 2E 30 2E 30 00 00 )                  //
   ...1.0.0..
  .custom instance void [System.Runtime]System.Reflection.AssemblyProductAttribute::.ctor(string) = ( 01 00 0A 48 65 6C 6C 6F 57 6F 72 6C 64 00 00 )    //
  ...HelloWorld..
  .custom instance void [System.Runtime]System.Reflection.AssemblyTitleAttribute::.ctor(string) = ( 01 00 0A 48 65 6C 6C 6F 57 6F 72 6C 64 00 00 )    //
  ...HelloWorld..
  .hash algorithm 0x00008004
  .ver 1:0:0:0
}
.module HelloWorld.dll
// MVID: {c0c19afb-0633-48ea-9bb5-b0790ef056c3}
.imagebase 0x00400000
.file alignment 0x00000200
.stackreserve 0x00100000
.subsystem 0x0003       // WINDOWS_CUI
.corflags 0x00000001    //  ILONLY
```

Bytecode & CIL

CIL example:

```
// =============== CLASS MEMBERS DECLARATION ==================

.class private auto ansi beforefieldinit HelloWorld.Program
       extends [System.Runtime]System.Object
{
  .method private hidebysig static void  Main(string[] args) cil managed
  {
    .entrypoint
    // Code size       13 (0xd)
    .maxstack  8
    IL_0000:  nop
    IL_0001:  ldstr      "Hello World!"
    IL_0006:  call       void [System.Console]System.Console::WriteLine(string)
    IL_000b:  nop
    IL_000c:  ret
  } // end of method Program::Main

  .method public hidebysig specialname rtspecialname
          instance void  .ctor() cil managed
  {
    // Code size       8 (0x8)
    .maxstack  8
    IL_0000:  ldarg.0
    IL_0001:  call       instance void [System.Runtime]System.Object::.ctor()
    IL_0006:  nop
    IL_0007:  ret
  } // end of method Program::.ctor

} // end of class HelloWorld.Program
```

Bytecode & CIL

Minimal vision:

```
.assembly Hello {}
.method public static void Main() cil
managed
{
  .entrypoint
  .maxstack 1
  ldstr "Hello world!"
  call void
[mscorlib]System.Console::WriteLine(string)
  ret
}
```

CIL Document:

Document

Bytecode & CIL

More details about CIL:

Opcode	Instruction	Opcode	Instruction
0x00	Nop	0x72	ldstr
0x01	Break	0x73	newobj
0x02	ldarg.0	0x7A	throw
0x03	ldarg.1	0x8C	box

Evaluation Stack

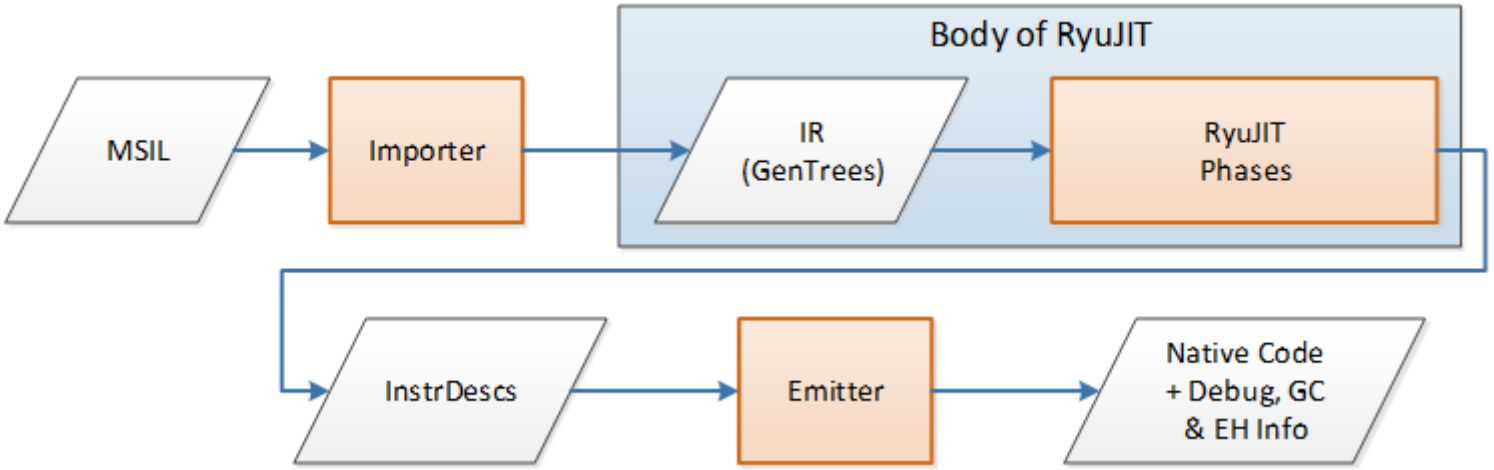
Bytecode & CIL

Why bytecode?

Imagine work without Bytecode:

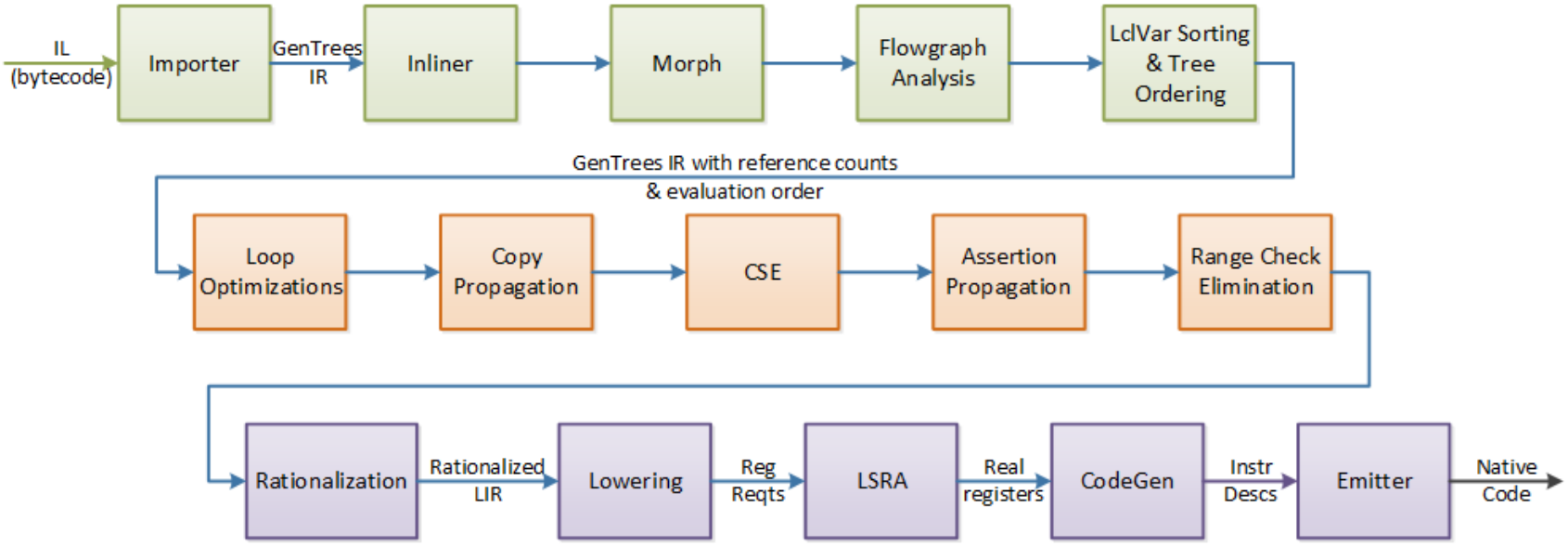
Bytecode ---(compile)---> Machine code

Something about RyuJIT Compiler (Just In Time Compiler):



Bytecode & CIL

A Deeper insight:



Bytecode & CIL

Why bytecode -> Performance

Common optimizing phases:

Phase 1 - inlining

Phase 2 - local optimizations

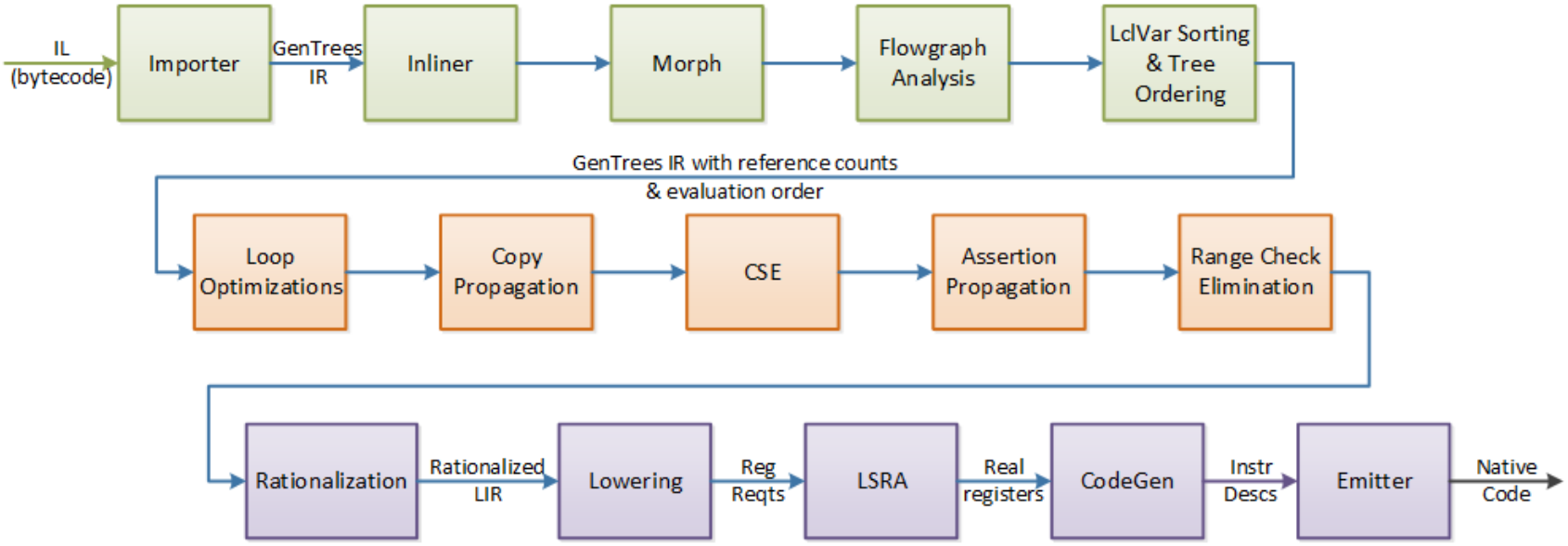
Phase 3 - control flow optimizations

Phase 4 - global optimizations

Phase 5 - native code generation

Bytecode & CIL

Review again:



Bytecode & CIL

Some Modern Idea:

More recently, the authors of V8 and Dart have challenged the notion that intermediate bytecode is needed for fast and efficient VM implementation. Both of these language implementations currently do direct JIT compiling from source code to machine code with no bytecode intermediary.

报告目录

中国科学技术大学
University of Science and Technology of China

1 CIL

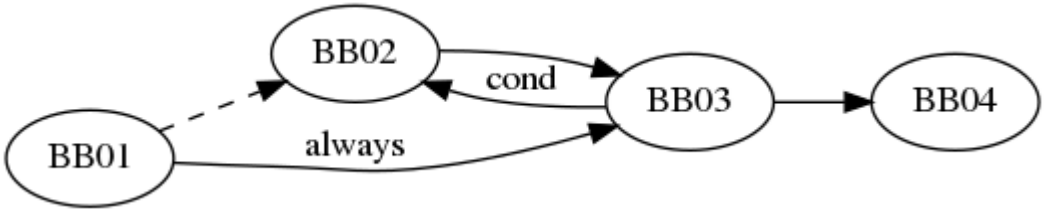
2 **Flowgraph**

3 Gentree

FlowGraph

FlowGraph

- Intermediate represention of Methods
- Nodes: BasicBlock
  - An instruction sequence without jump statement
- Edges: branch (jump)
  - condition
  - always
  - none
  - switch
  - ...

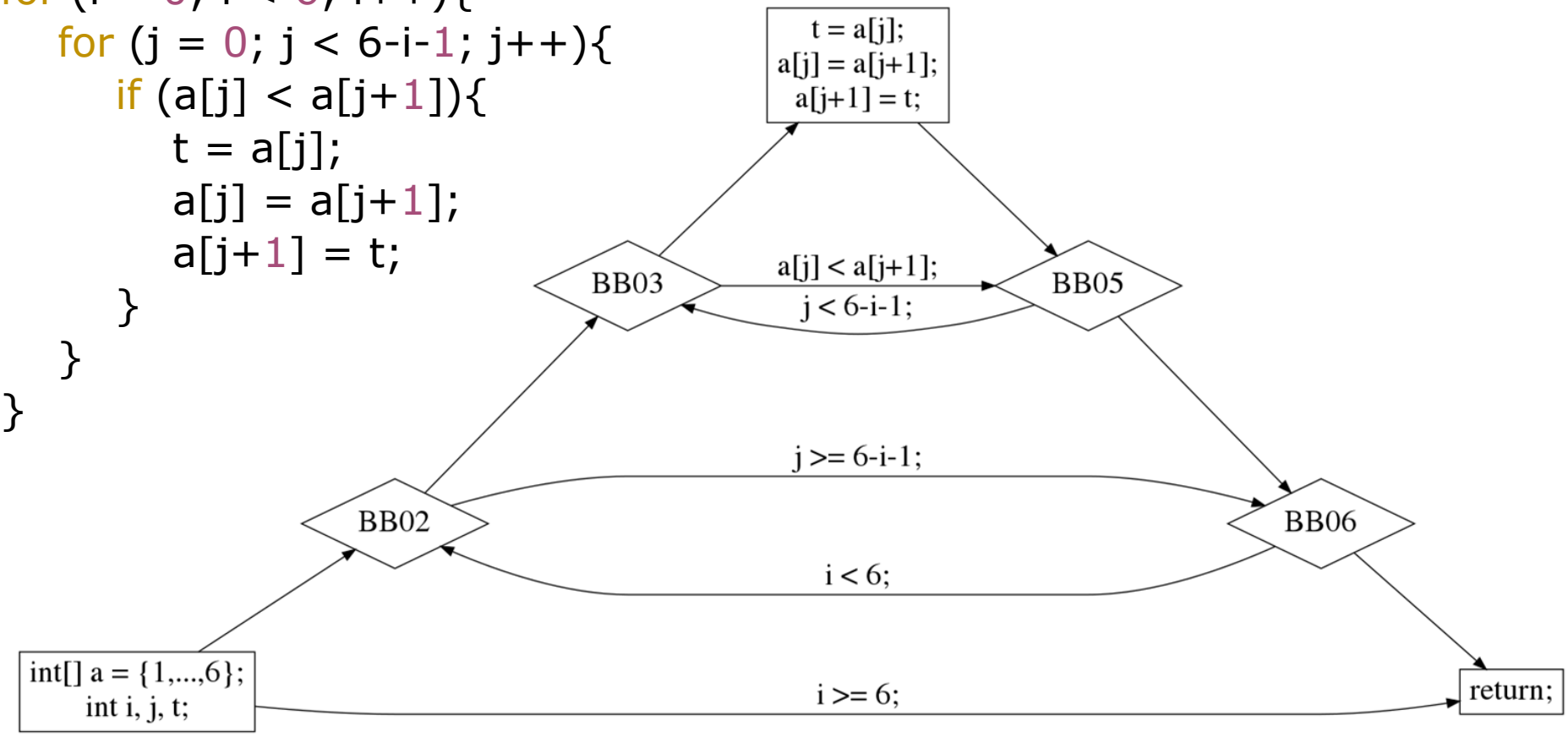


Flowgraph of while statement

FlowGraph

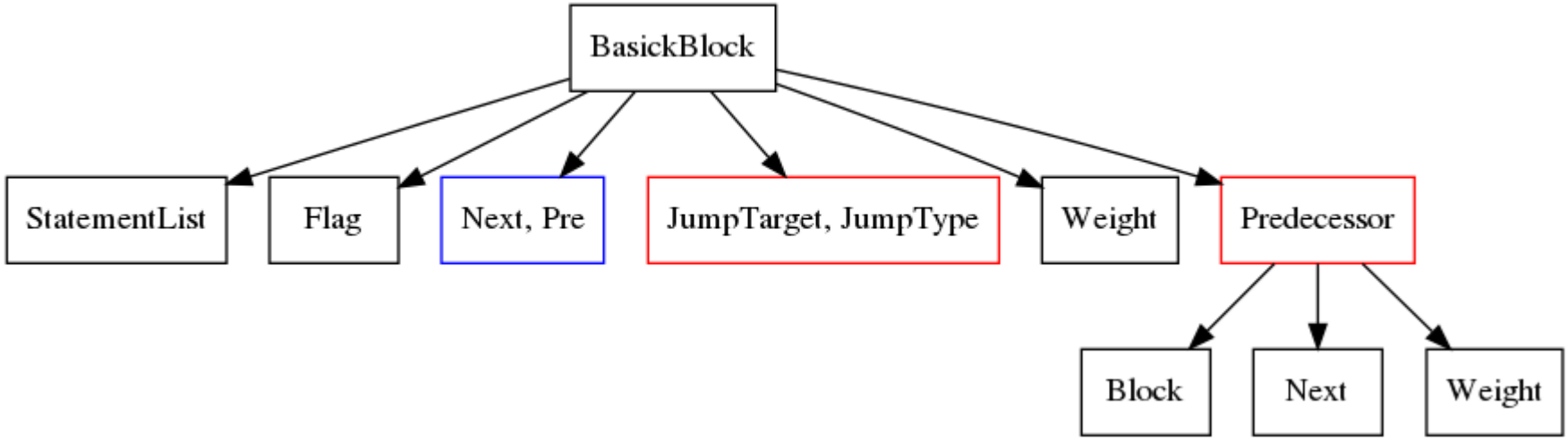
Example: bubble sort

```
for (i = 0; i < 6; i++){
    for (j = 0; j < 6-i-1; j++){
        if (a[j] < a[j+1]){
            t = a[j];
            a[j] = a[j+1];
            a[j+1] = t;
        }
    }
}
```



FlowGraph

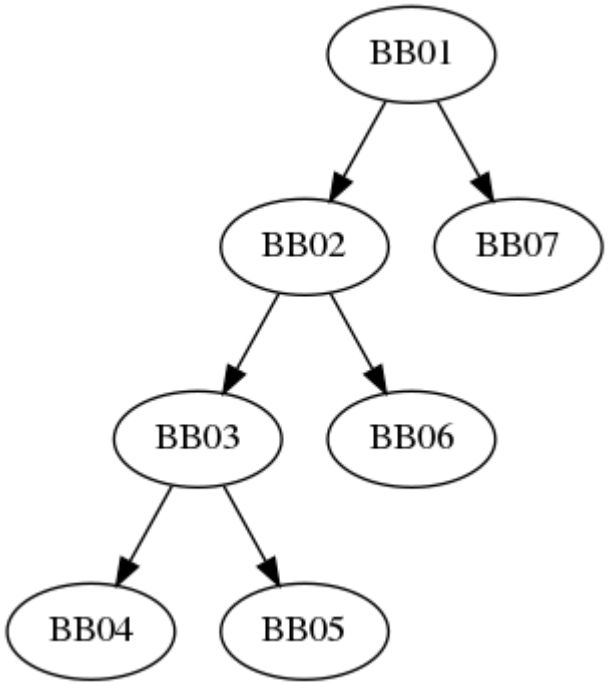
BasicBlock



FlowGraph

Dominator Tree

- A block M dominates a block N if every path from the entry that reaches block N has to pass through block M.
- For each block, there is only one immediate dominator.

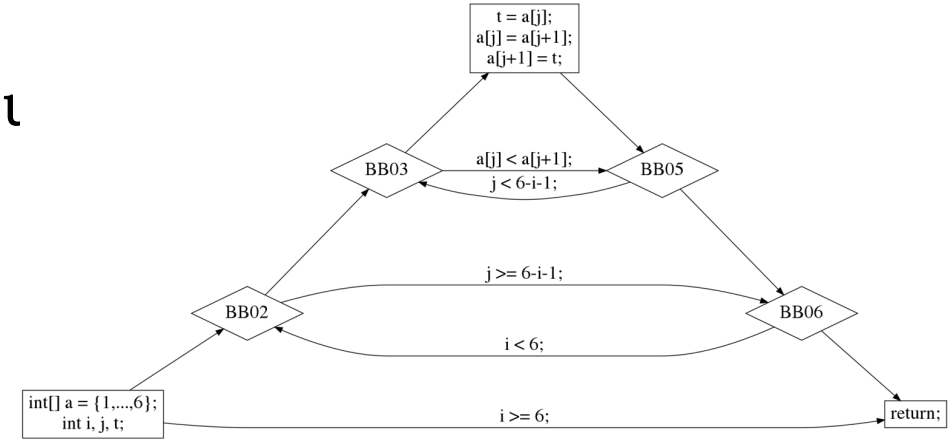


FlowGraph

- **BasicBlock**
  - Flags: Mark BB status
    - Imported?
    - Visited during Opt?
    - Starts a loop?
    - Rarely run?
    - Allocate Mem?
    - Call a function?
    - …
  - Weights
    - How often will this BB run?

FlowGraph

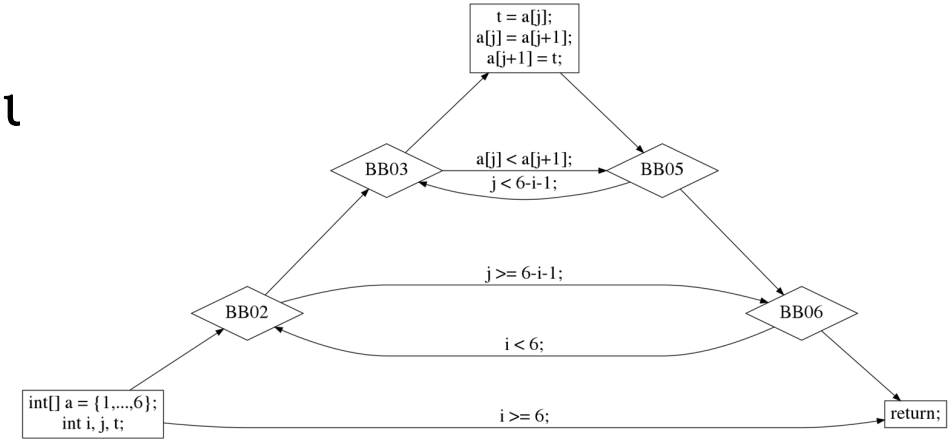
- **BasicBlock**
  - Flags: Mark BB statu
    - Imported?
    - Visited during Opt?
    - Starts a loop?
    - Rarely run?




```
------------------------------------------------------------------------------------------------------------------------
BBnum BBid ref try hnd preds        weight    lp [IL range]     [jump]      [EH region]        [flags]
------------------------------------------------------------------------------------------------------------------------
BB01 [0000]  1                        1         [000..016)-> BB07 ( cond )                   i label target new[]
BB02 [0001]  2      BB01,BB06         0.5     0 [016..01A)-> BB06 ( cond )                   i Loop label target bwd
BB03 [0002]  2      BB02,BB05         0.5     1 [01A..022)-> BB05 ( cond )                   i Loop label target idxlen
bwd
BB04 [0003]  1      BB03              0.5     1 [022..030)                                   i idxlen bwd
BB05 [0004]  2      BB03,BB04         0.5     1 [030..03A)-> BB03 ( cond )                    i label target bwd
BB06 [0006]  2      BB02,BB05         0.5     0 [03A..042)-> BB02 ( cond )                    i label target bwd
BB07 [0008]  2      BB01,BB06         1         [042..043)                (return)             i label target
------------------------------------------------------------------------------------------------------------------------
```

FlowGraph

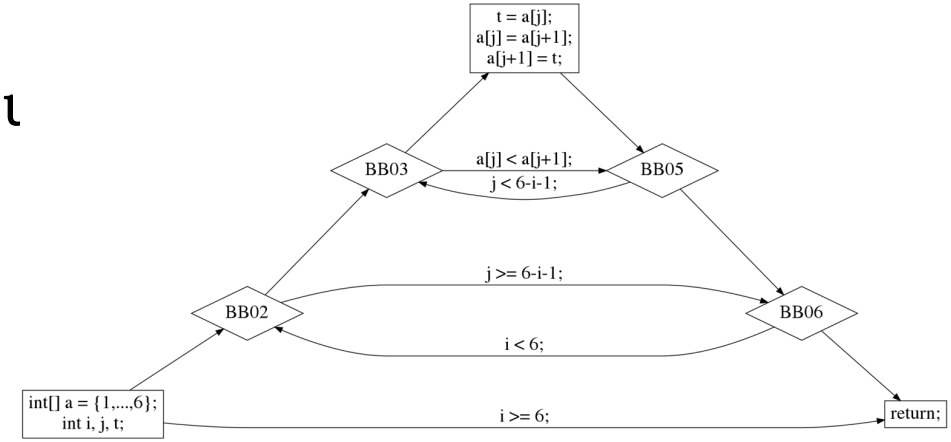
- **BasicBlock**
  - Flags: Mark BB statu
    - Imported?
    - Visited during Opt?
    - Starts a loop?
    - Rarely run?




```
------------------------------------------------------------------------------------------------------------------------
BBnum BBid ref try hnd preds        weight    lp [IL range]     [jump]     [EH region]        [flags]
------------------------------------------------------------------------------------------------------------------------
BB01 [0000]  1                         1       [000..016)-> BB07 ( cond )              i label target new[]
BB02 [0001]  2       BB01,BB06         4     0 [016..01A)-> BB06 ( cond )              i Loop label target bwd
BB03 [0002]  2       BB02,BB05         2     1 [01A..022)-> BB05 ( cond )              i Loop label target idxlen
bwd
BB04 [0003]  1       BB03              2     1 [022..030)                              i idxlen bwd
BB05 [0004]  2       BB03,BB04         2     1 [030..03A)-> BB03 ( cond )               i label target bwd
BB06 [0006]  2       BB02,BB05         4     0 [03A..042)-> BB02 ( cond )               i label target bwd
BB07 [0008]  2       BB01,BB06         1       [042..043)              (return)         i label target
------------------------------------------------------------------------------------------------------------------------
```

FlowGraph

- **BasicBlock**
  - Flags: Mark BB statu
    - Imported?
    - Visited during Opt?
    - Starts a loop?
    - Rarely run?




```
------------------------------------------------------------------------------------------------------------------------
BBnum BBid ref try hnd preds        weight    lp [IL range]     [jump]      [EH region]        [flags]
------------------------------------------------------------------------------------------------------------------------
BB01 [0000]  1                         1       [000..016)-> BB07 ( cond )                    i label target new[]
BB02 [0001]  2      BB01,BB06          4     0 [016..01A)-> BB06 ( cond )                    i Loop label target bwd
BB03 [0002]  2      BB02,BB05         16     1 [01A..022)-> BB05 ( cond )                    i Loop label target idxlen
bwd
BB04 [0003]  1      BB03               8     1 [022..030)                                    i idxlen bwd
BB05 [0004]  2      BB03,BB04         16     1 [030..03A)-> BB03 ( cond )                     i label target bwd
BB06 [0006]  2      BB02,BB05          4     0 [03A..042)-> BB02 ( cond )                     i label target bwd
BB07 [0008]  2      BB01,BB06          1       [042..043)                (return)             i label target
------------------------------------------------------------------------------------------------------------------------
```

报告目录

中国科学技术大学
University of Science and Technology of China

1 CIL

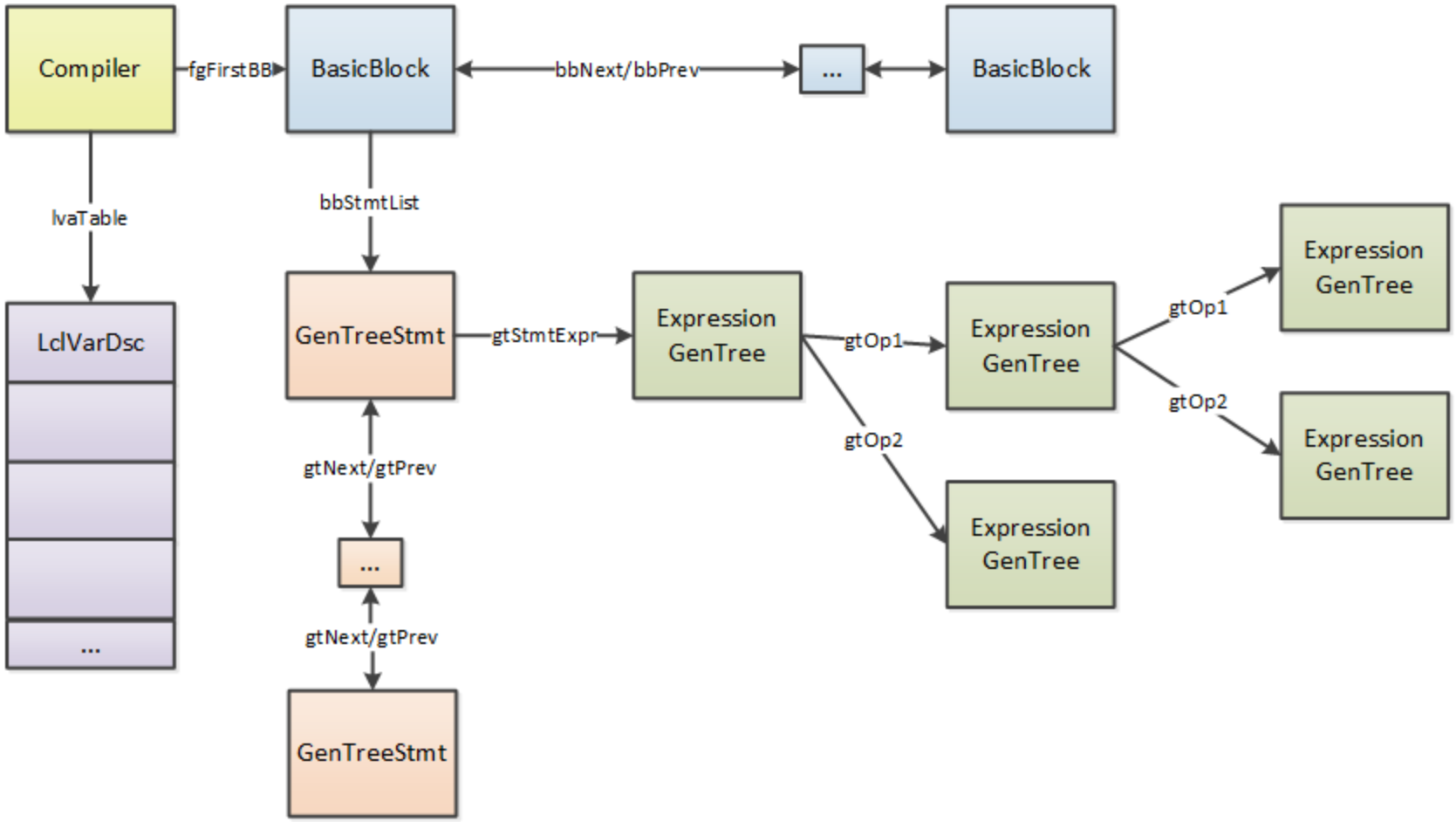
2 Flowgraph

3 **Gentree**

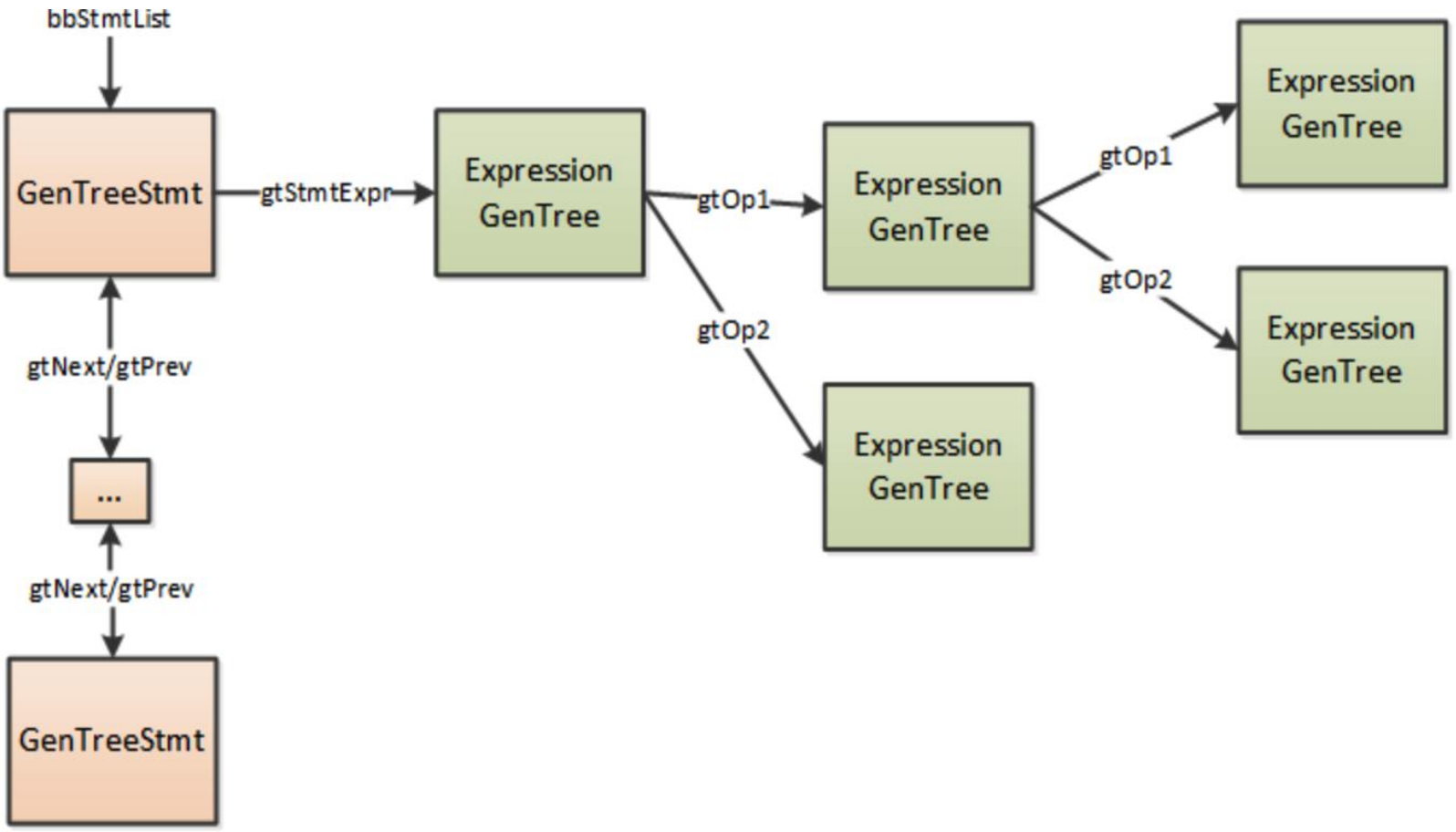
GenTree

- **Code-gen Tree**
- **Used nearly all-through the JIT phases**
  - Represent operation corresponding to the node
  - Other information related to code-gen

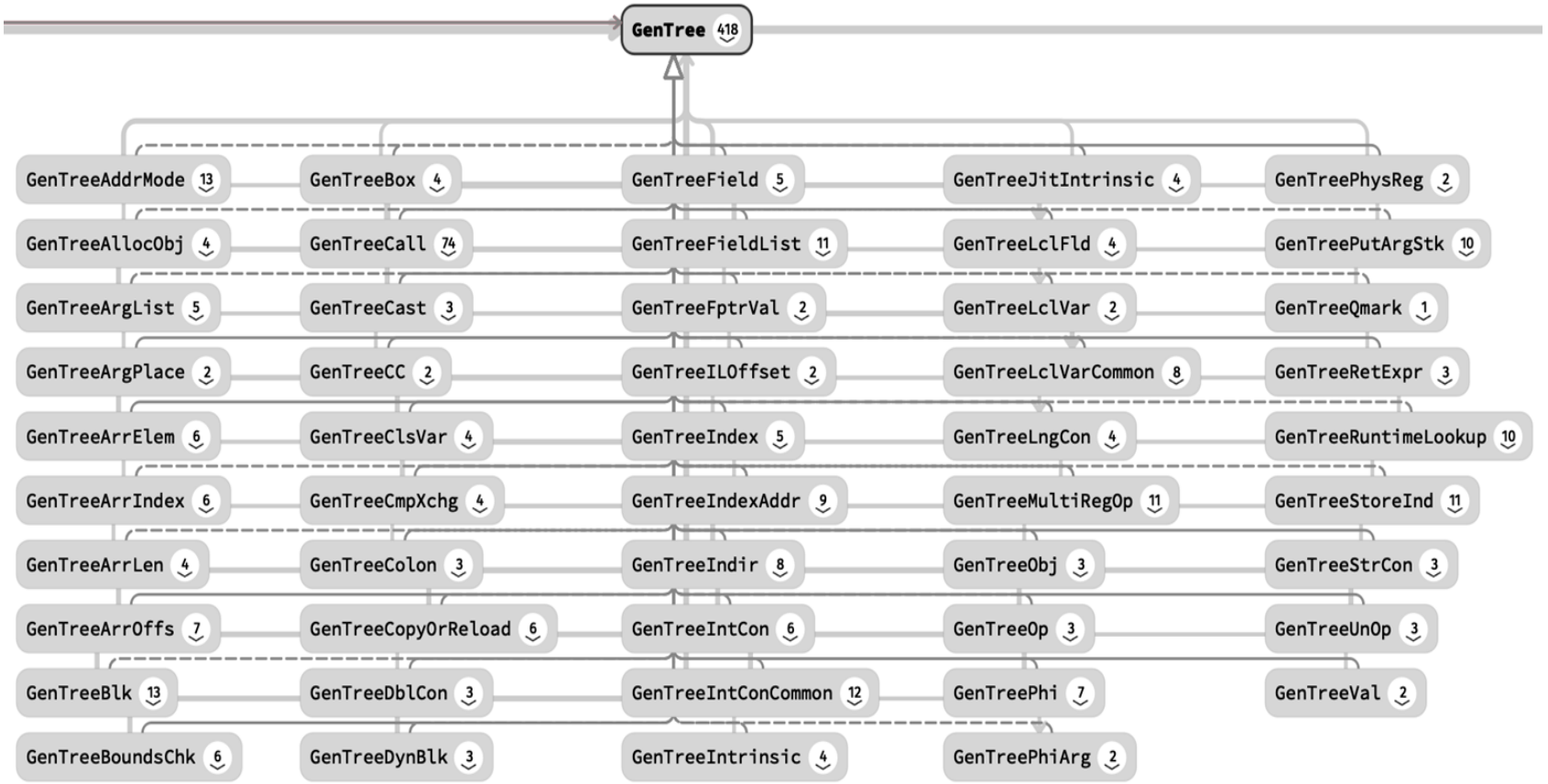
GenTree > Overview



GenTree > HIR


bbStmtList
GenTreeStmt
gtStmtExpr
Expression GenTree
gtOp1
Expression GenTree
gtOp1
Expression GenTree
gtOp2
Expression GenTree
gtOp2
Expression GenTree
gtNext/gtPrev
...
gtNext/gtPrev
GenTreeStmt

GenTree > GenTreeNode



GenTree > HIR > Example

```
r1 = (-b + Math.Sqrt(b*b - 4*a*c))/(2*a);
```

```
STMT (IL 0x000...0x026)
     ▌ ASG  double  // assignment nodes
     ├─▌ IND  double
     │  └─▌ LCL_VAR  byref V03 arg3  // r1
     └─▌ DIV  double
          ├─▌ ADD  double
          │  ├─▌ NEG  double
          │  │  └─▌ LCL_VAR  double V01 arg1
          │  └─▌ INTRINSIC double sqrt
          │        └─▌ SUB  double
          │           ├─▌ MUL  double
          │           │  ├─▌ LCL_VAR  double V01 arg1
          │           │  └─▌ LCL_VAR  double V01 arg1
          │           └─▌ MUL  double
          │              ├─▌ MUL  double
          │              │  ├─▌ LCL_VAR  double V00 arg0
          │              │  └─▌ CNS_DBL  double
     4.0000000000000000
          │              └─▌ LCL_VAR  double V02 arg2
          └─▌ MUL  double
             ├─▌ LCL_VAR  double V00 arg0  // a
             └─▌ CNS_DBL  double 2.0000000000000000  //
const double
```

GenTree > HIR > Eval Order

- **Post Order**

  Depth first, Left to right

- **Exception**
  - flag GTF_REVERSE_OPS is set
  - Dynamically-sized block

GenTree > LIR

- **Doubly linked list of `GenTree` Nodes**
  - `gtNext/gtPrev`

- **Execution order: order given by the list**

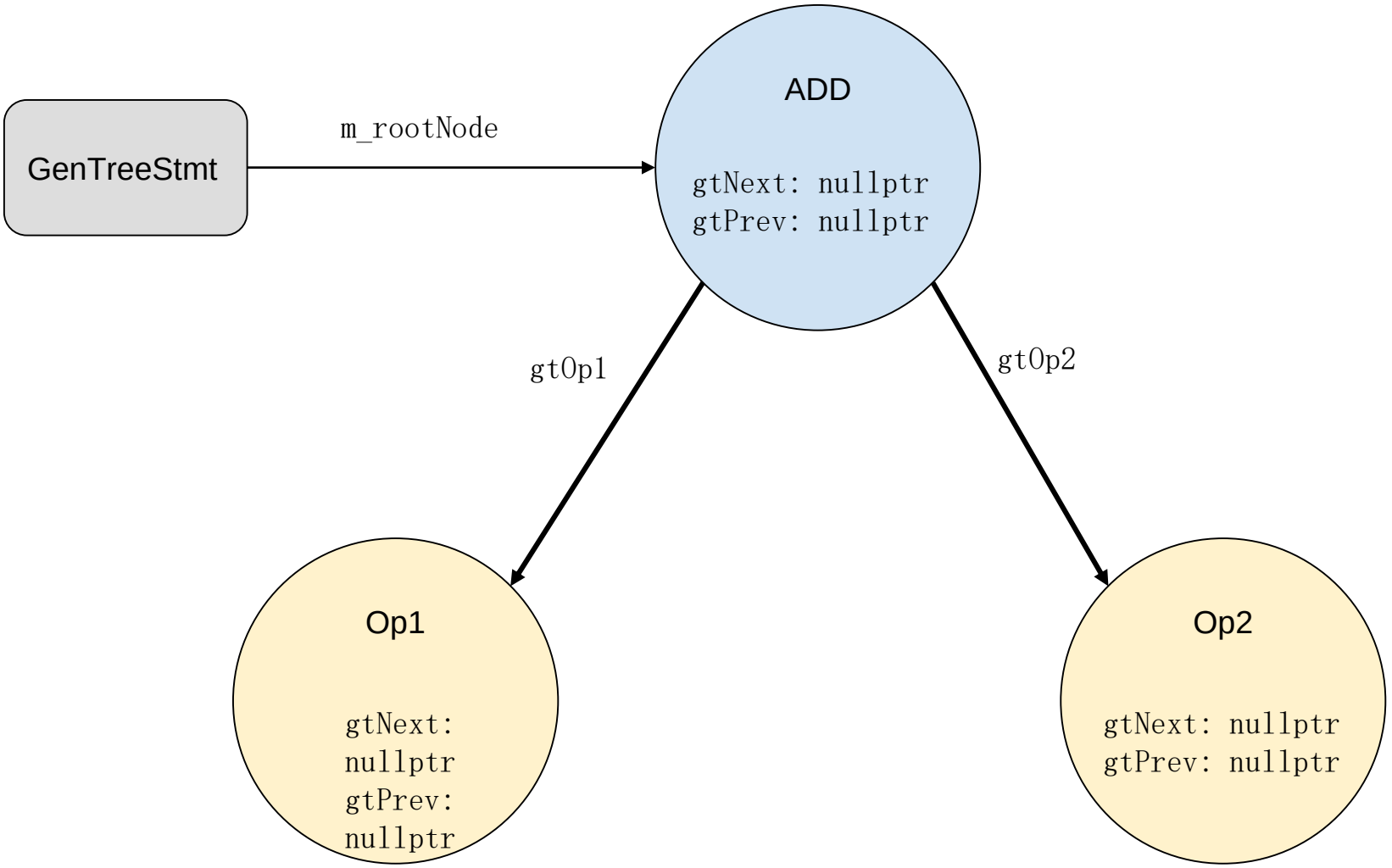
GenTree > HIR to LIR

- **Phase: rationalization**
- **In-place transformation**

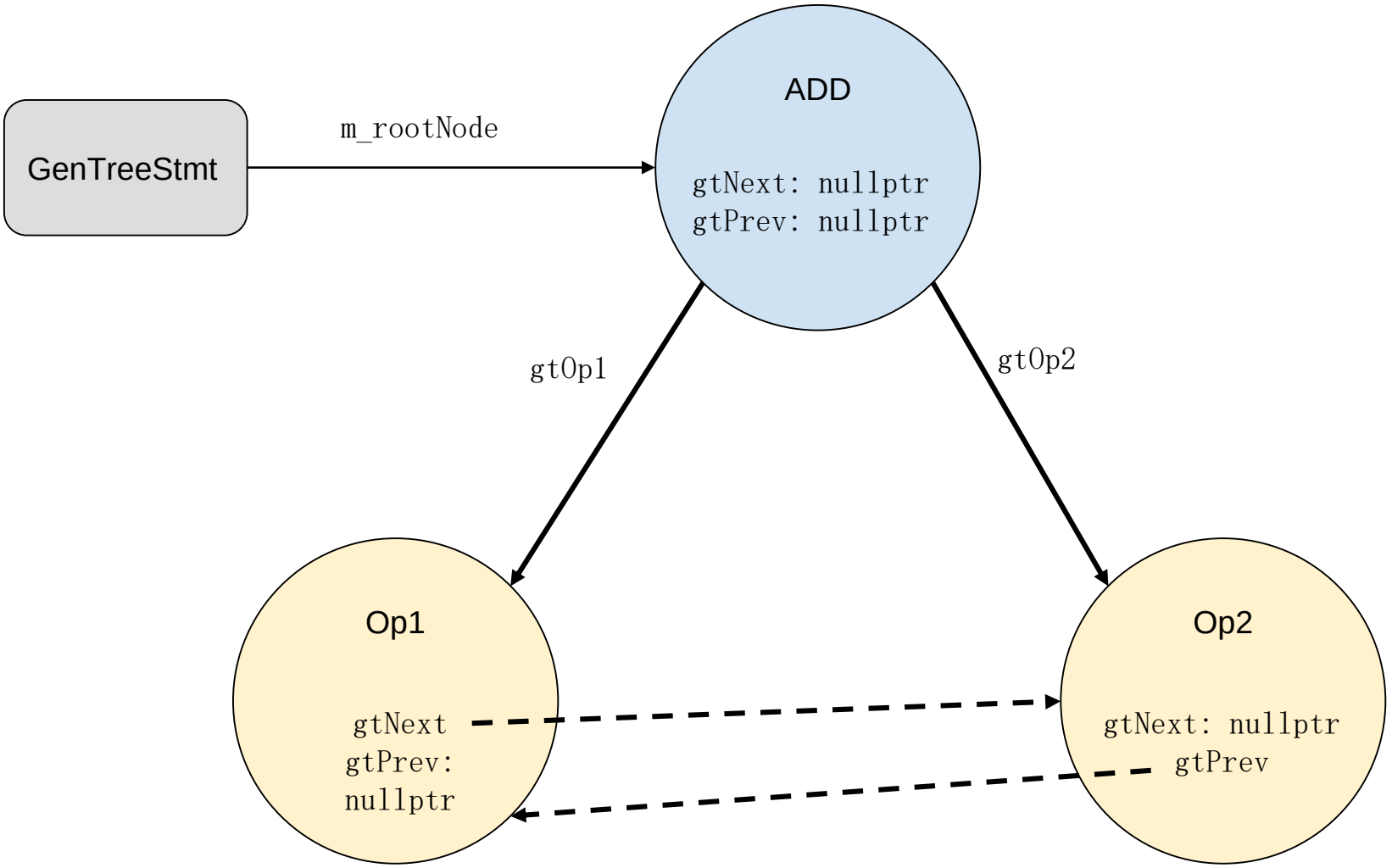
  What does in-place mean?

  - threaded tree

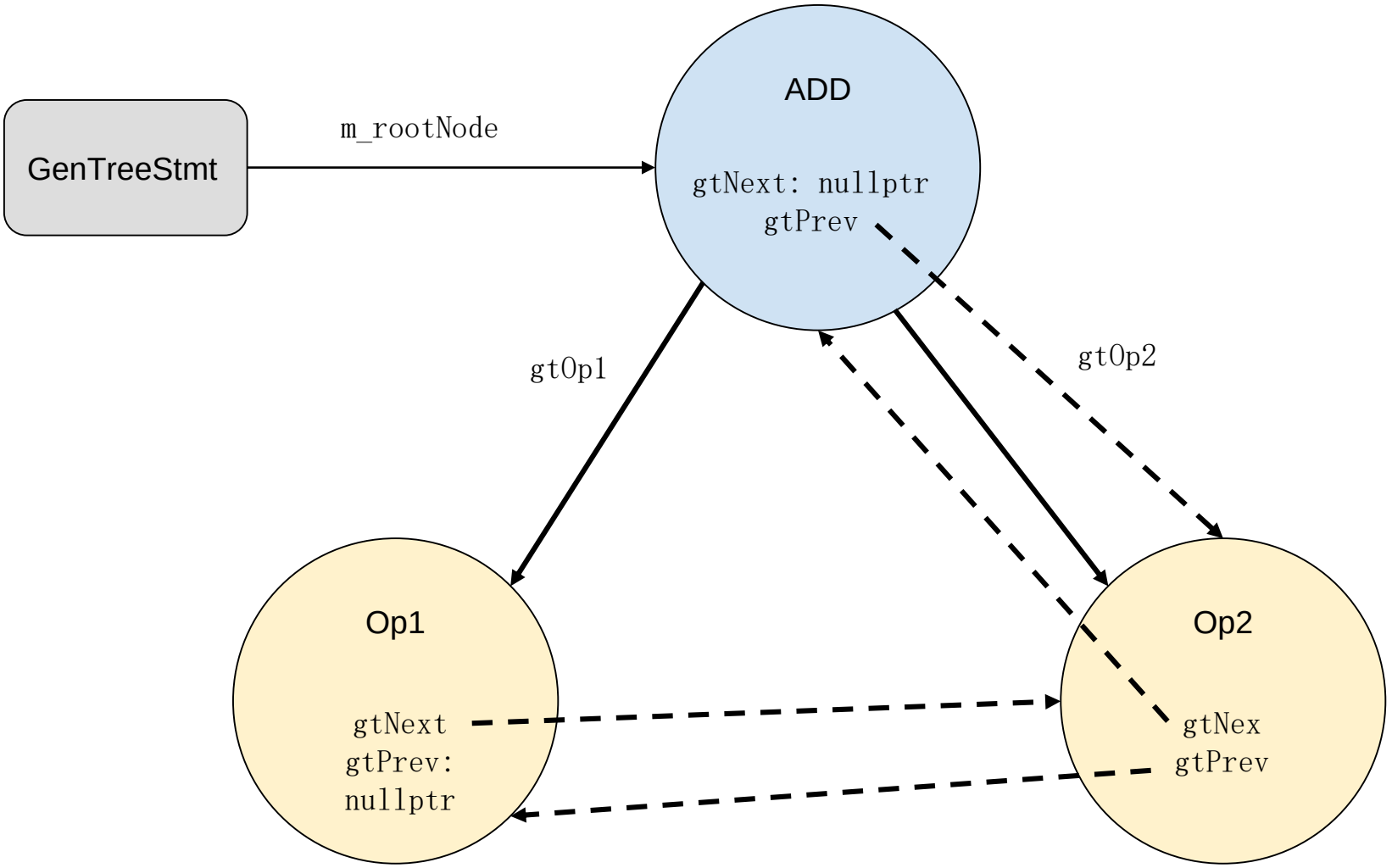
GenTree > HIR to LIR



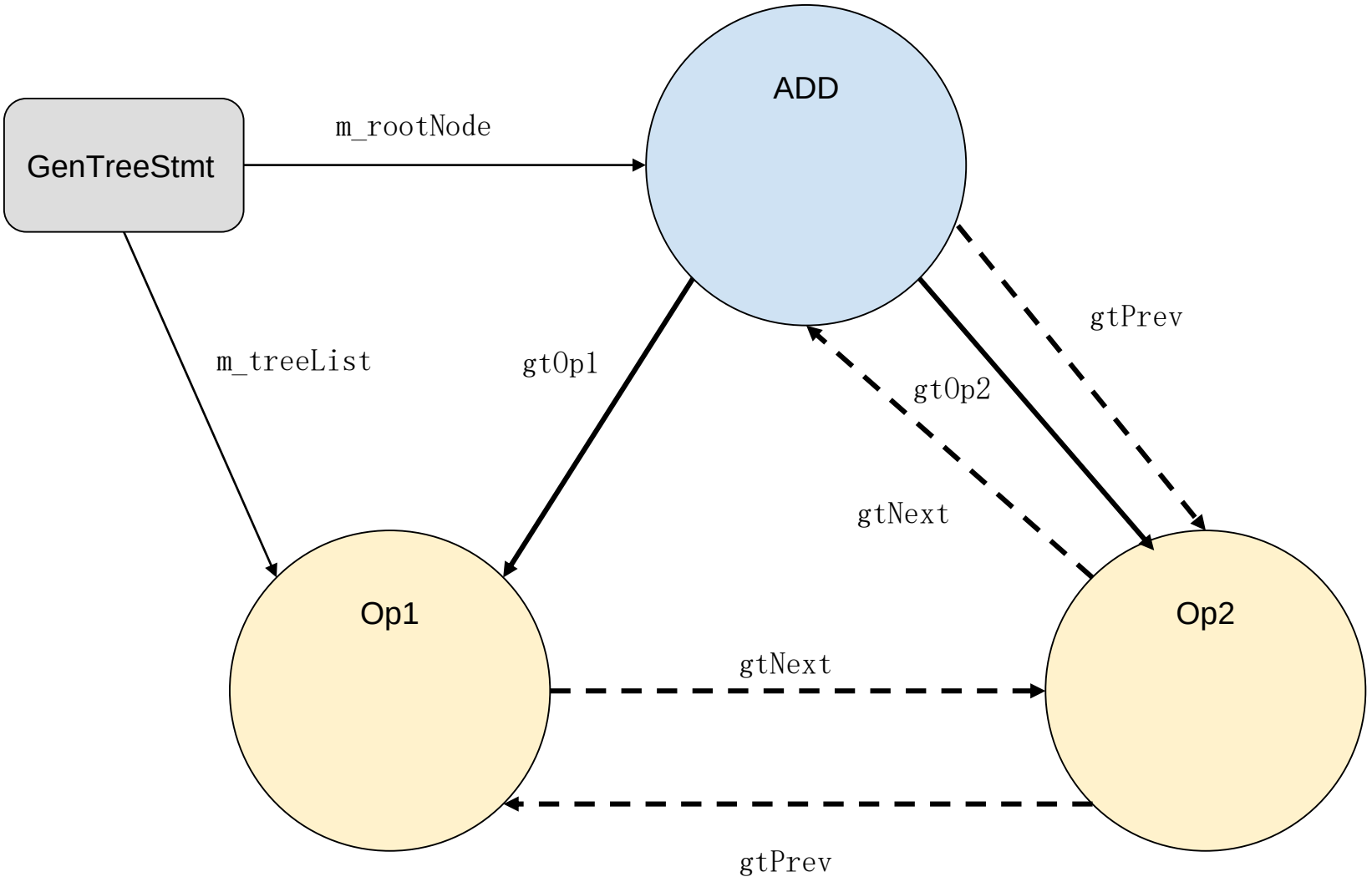
GenTree > HIR to LIR



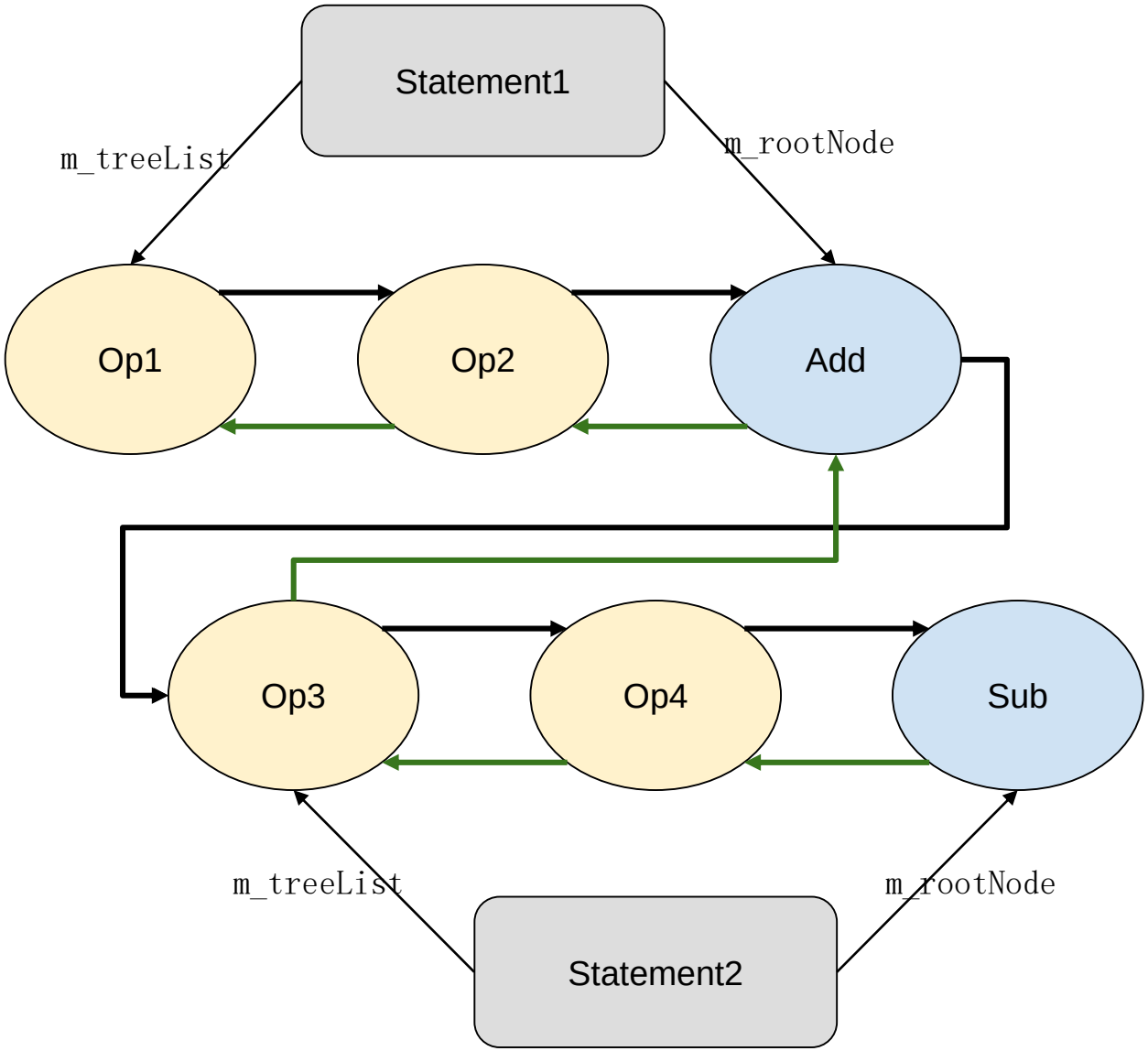
GenTree > HIR to LIR



GenTree > HIR to LIR



GenTree > HIR to LIR



GenTree > HIR to LIR

- **Pointer fields set. Enough?**
  - Elimination of Assignment Node
  - SSA
  - Elimination of statements
  - ...

Example

```
STMT (IL 0x000...0x026)
     ▌ ASG  double  // assignment nodes
     ├─▌ IND  double
     │  └─▌ LCL_VAR  byref V03 arg3  // r1
     └─▌ DIV  double
        ├─▌ ADD  double
        │  ├─ ...
        └─▌ MUL  double
```

```
          t16 = ▌ MUL double $145
          ┌─▌ t16 double
                    ▌ STORE_LCL_VAR double V07 cse1
t51 = LCL_VAR double V07 cse1 $145
          ┌─▌ t13 double
          ├─▌ t51 double
t17 = ▌ DIV double $146
t0 = LCL_VAR byref V03 arg3 u:1 (last use) $c0
          ┌─▌ t0 byref
          ├─▌ t17 double
                    ▌ STOREIND double
```

中国科学技术大学
University of Science and Technology of China

谢谢！