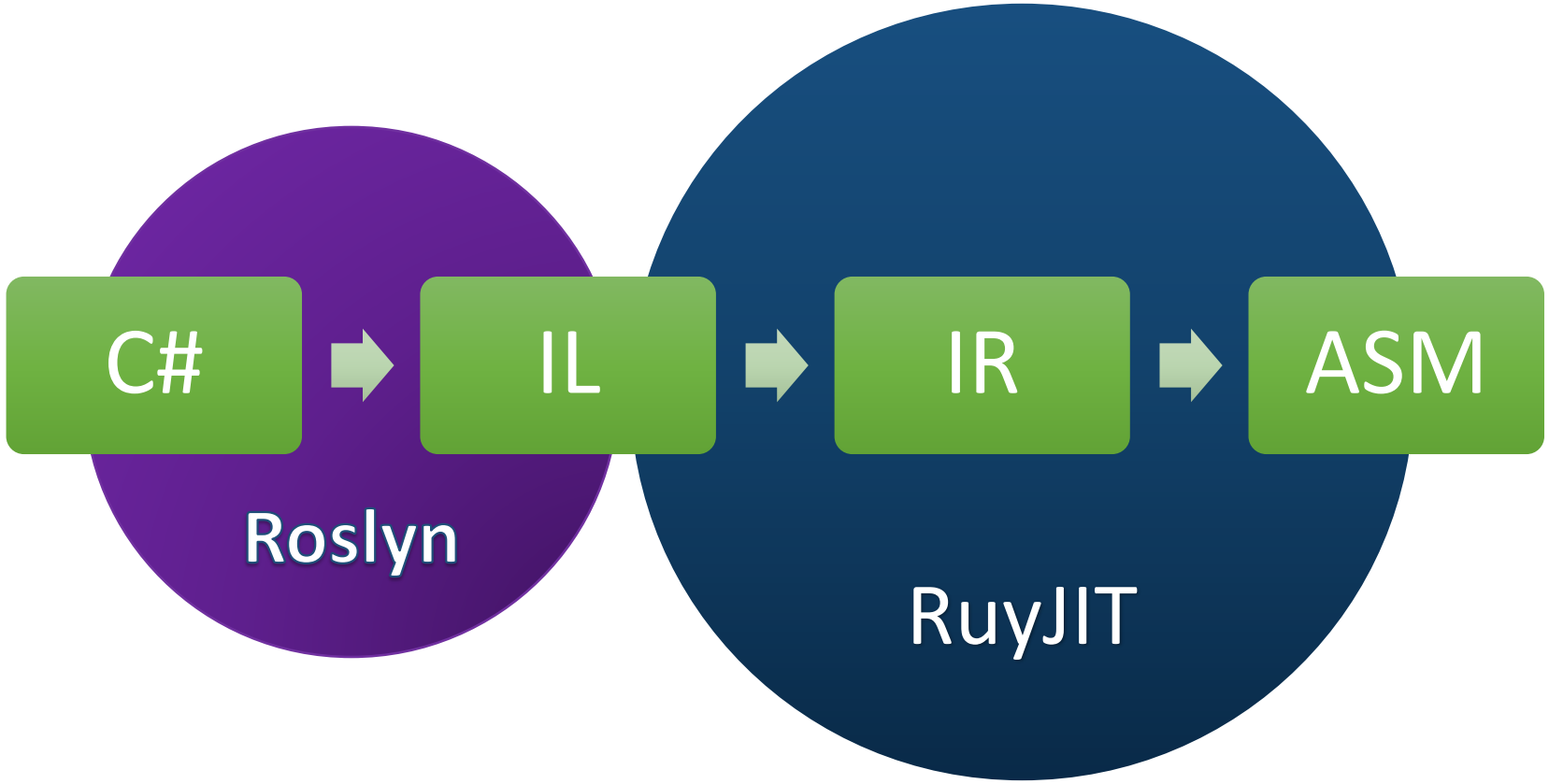
How to add an optimization for C# to JIT compiler

@EgorBo
Engineer at Microsoft

From C# to machine code



Let’s start with a simple optimization

```
float y = x / 2;        vdivss (Latency: 20, R.Throughput: 14)

float y = x * 0.5f;     vmulss (Latency: 5, R.Throughput: 0.5)
```

^ for my MacBook (Haswell)

X/C

```
double y = x / 2;      = x * 0.5
float  y = x / 2;      = x * 0.5f
double y = x / 10;     = x * 0.1
double y = x / 8;      = x * 0.125
double y = x / -0.5;   = x * -2
```

```
float x = 48.665f;
Console.WriteLine(x / 10f);  // 4.8665
Console.WriteLine(x * 0.1f); // 4.8665004
```

X / C – let me optimize it in Roslyn!

```
static float GetSpeed(float distance, float time)
{
    return distance / time;
}


...
float speed = GetSpeed(distance, 2);
```

Does Roslyn see “X/C” here?

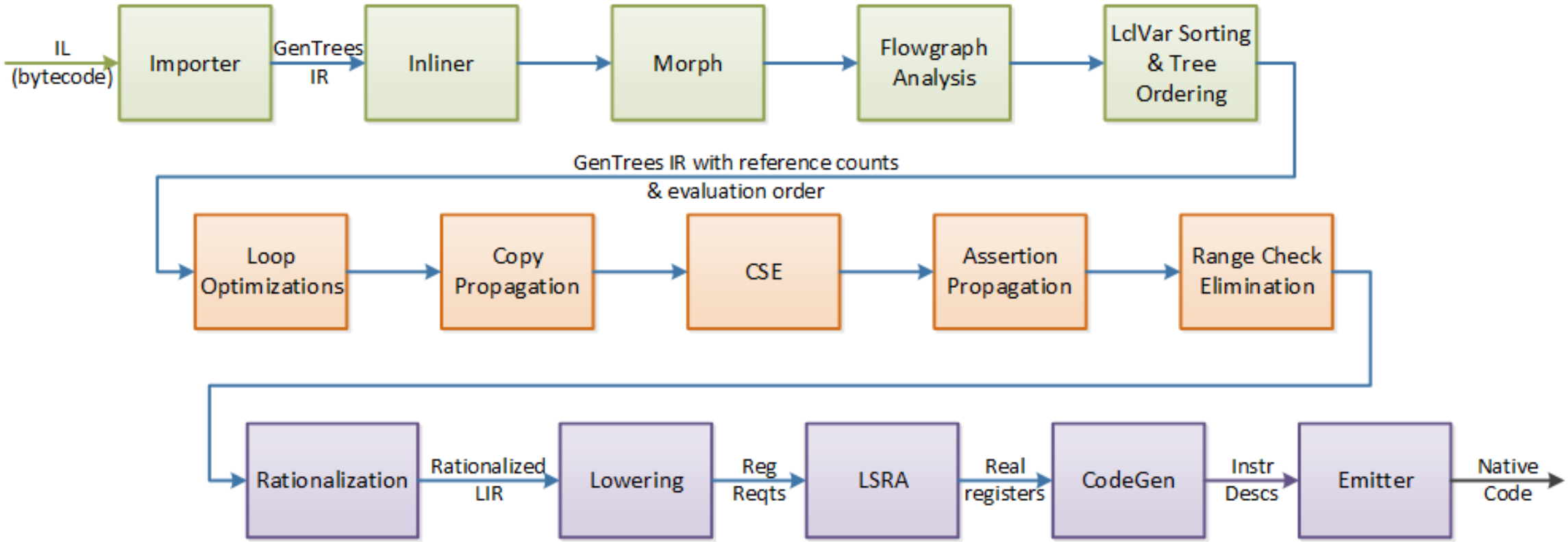
NO! It doesn’t inline methods

Where to implement my custom optimization?

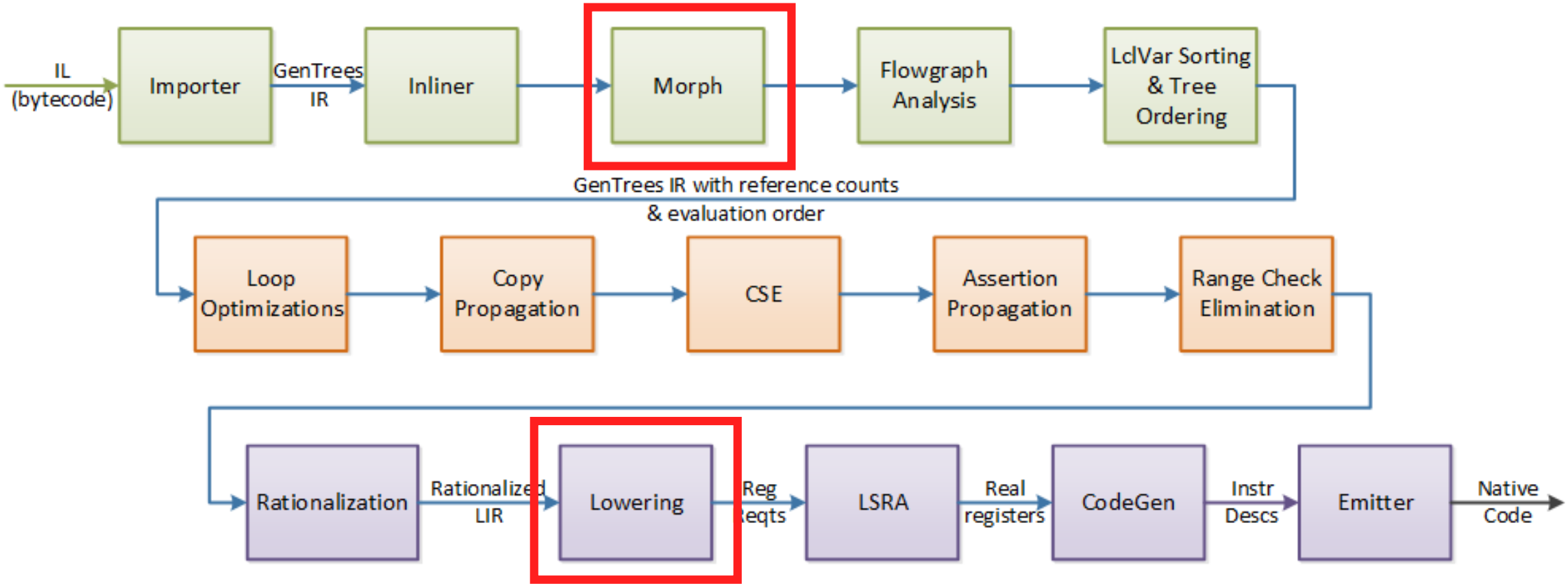
- **Roslyn**
  - + No time constraints
  - + It's written in C# - easy to add optimizations, easy to debug and experiment
  - - No cross-assembly optimizations
  - - No CPU-dependent optimizations (IL is cross-platform)
  - - Doesn't know how the code will look like after inlining, CSE, loop optimizations, etc.
  - - F# doesn't use Roslyn
- **JIT**

  - + Inlining, CSE, Loop opts, etc phases create more opportunities for optimizations
  - + Knows everything about target platform, CPU capabilities
  - - Written in C++, difficult to experiment
  - - Time constraints for optimizations (probably not that important with Tiering)
- **R2R (AOT)**
  - + No time constraints (some optimizations are really time consuming, e.g. full escape analysis)
  - - No CPU-dependent optimizations
  - - Will be most likely re-jitted anyway?
- **ILLink Custom Step**
  - + Cross-assembly IL optimizations
  - + Written in C#
  - + We can manually de-virtualize types/methods/calls (if we know what we are doing)
  - - Still no inlining, CSE, etc..

RuyJIT: phases

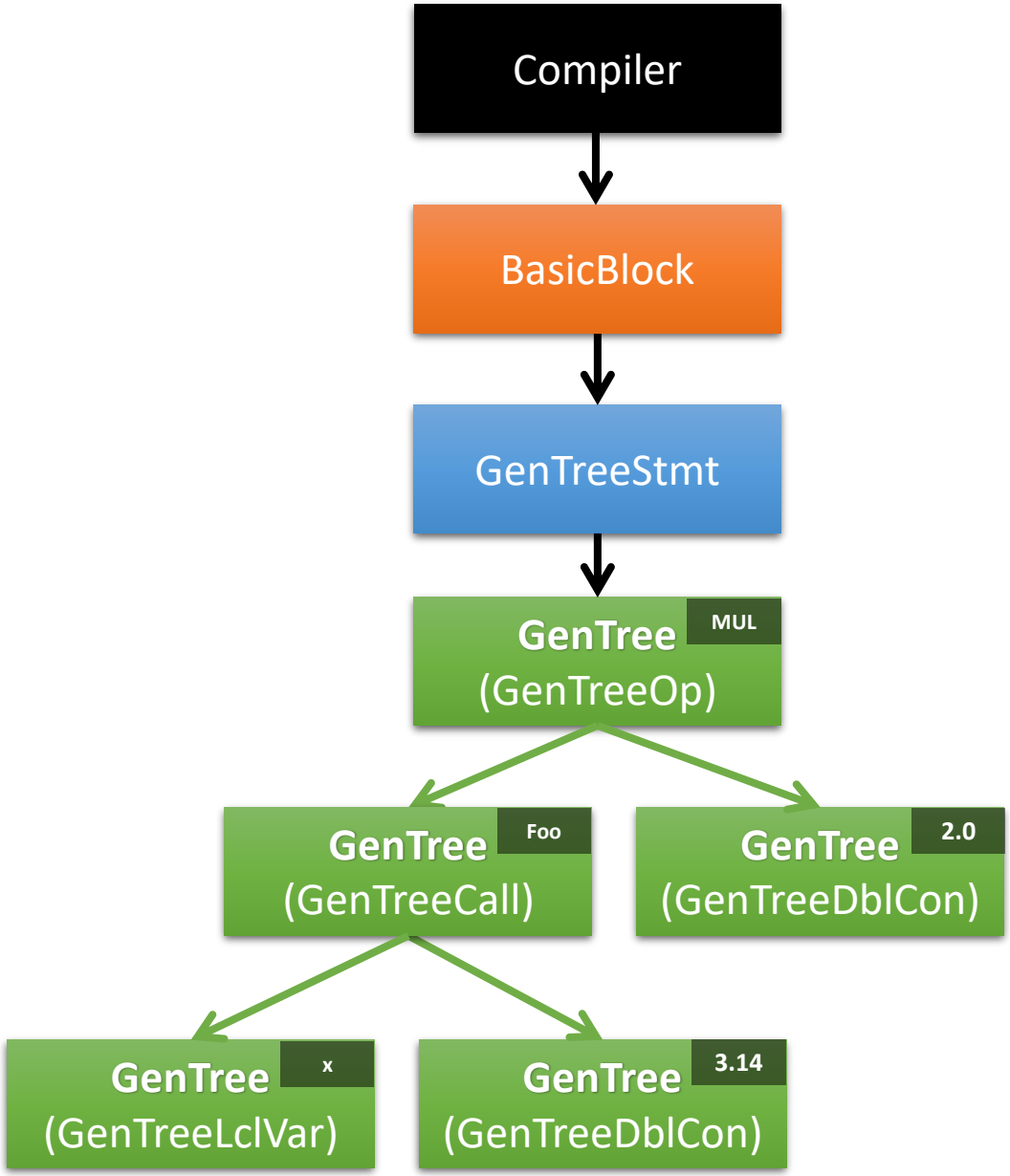


RuyJIT: where to morph my X/C?



GenTree

```
static float Test(float x)
{
    return Foo(x, 3.14) * 2;
}
```



Dump IR via COMPlus_JitDump

ConsoleApp182.MathMaxBer | _testInput

```
1 reference
public static double GetCircleArea(double d)
{
    return Math.Pow(d, 2) / (4 * Math.PI);
}
```

Disasmo

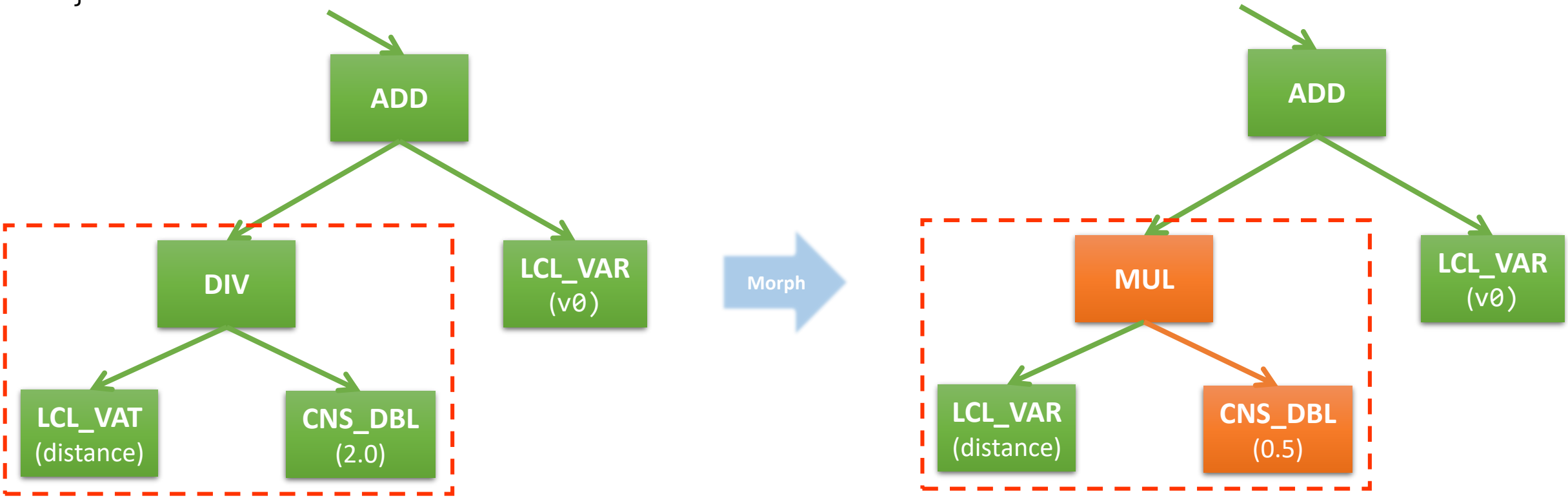
CoreCLR Path: C:\prj\coreclr-baseline ... Refresh Tiered JIT

Output | Previous output | Settings | S.R.Intrinsics | Tests

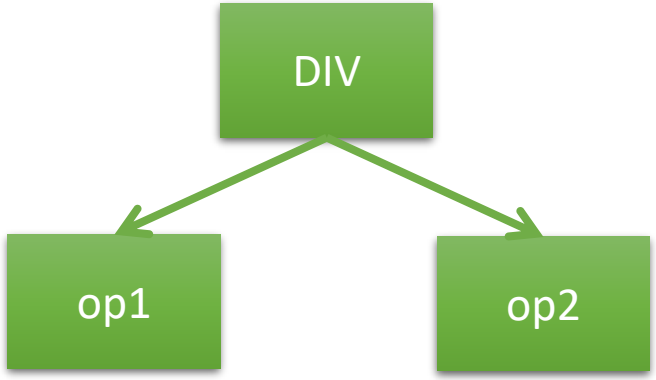
```
45
46 impImportBlockPending for BB01
47
48 Importing BB01 (PC=000) of 'Program:GetCircleArea(double):double'
49     [ 0]   0 (0x000) ldarg.0
50     [ 1]   1 (0x001) ldc.r8 2.0000000000000000
51     [ 2]  10 (0x00a) call 0A000018
52 In Compiler::impImportCall: opcode is call, kind=0, callRetType is double, structSize is 0
53
54     [ 1]  15 (0x00f) ldc.r8 12.566370614359172
55     [ 2]  24 (0x018) div
56     [ 1]  25 (0x019) ret
57
58                [000006] ------------              *  STMT      void  (IL 0x000...  ???)
59                [000005] --C---------              \--*  RETURN    double
60                [000004] --C---------                 \--*  DIV       double
61                [000002] --C---------                    +--*  INTRINSIC double pow
62                [000000] ------------                    |  +--*  LCL_VAR   double V00 arg0
63                [000001] ------------                    |  \--*  CNS_DBL   double 2.0000000000000000
64                [000003] ------------                    \--*  CNS_DBL   double 12.566370614359172
65
66 *************** in fgTransformIndirectCalls(root)
```

Back to X/C: Morph IR Tree

```
static float Calculate(float distance, float v0)
{
    return distance / 2 + v0;
}
```



Implementing the opt in morph.cpp




```
case GT_DIV:
    // Replace "val / dcon" with "val * (1.0 / dcon)" if dcon is a power of two.
    // Powers of two within range are always exactly represented,
    // so multiplication by the reciprocal is safe in this scenario
    if (op2->IsCnsFltOrDbl())
    {
        double divisor = op2->AsDblCon()->gtDconVal;
        if (((typ == TYP_DOUBLE) && FloatingPointUtils::hasPreciseReciprocal(divisor)) ||
            ((typ == TYP_FLOAT) && FloatingPointUtils::hasPreciseReciprocal(forceCastToFloat(divisor))))
        {
            oper = GT_MUL;
            tree->ChangeOper(oper);
            op2->AsDblCon()->gtDconVal = 1.0 / divisor;
        }
    }
```

Inspired by GT_ROL

```
/// <summary>
/// Rotates the specified value left by the specified number of bits.
/// </summary>
public static uint RotateLeft(uint value, int offset)
    => (value << offset) | (value >> (32 - offset));
```

```
          OR                      ROL
        /    \                   /   \
       /      \                 /     \
     LSH      RSZ      =>      x       y
     / \      / \
    x  AND   x  AND
       / \      / \
      y  31   ADD  31
              / \
            NEG  32
             |
             y
```

```
rol eax, cl
```

JIT: Optimize "constant_string".Length #26000

Open EgorBo wants to merge 8 commits into dotnet:master from EgorBo:str-len-cns

```
"Hello".Length -> 5
```

```
static void Append(string str)
{
    if (str.Length <= 2)
        QuickAppend(str);
    else
        SlowAppend(str);
}
```

```
...
builder.Append("/>");
```

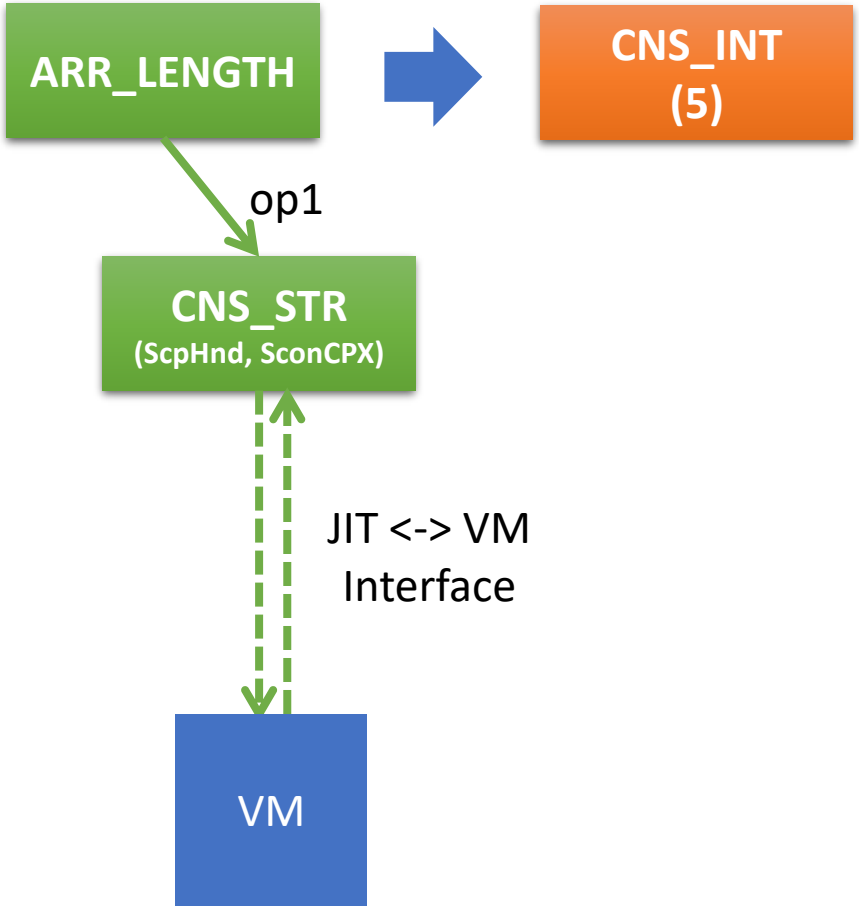
Inline, remove if (2 <= 2)

```
builder.QuickAppend("/>");
```

"Hello".Length => 5

```
case GT_ARR_LENGTH:
{
    if (op1->OperIs(GT_CNS_STR))
    {
        GenTreeStrCon* strCon = op1->AsStrCon();

        int len = info.compCompHnd->getStringLength(
                strCon->gtScpHnd, strCon->gtSconCPX);

        return gtNewIconNode(len);
    }
    break;
}
```

Access VM's data from JIT

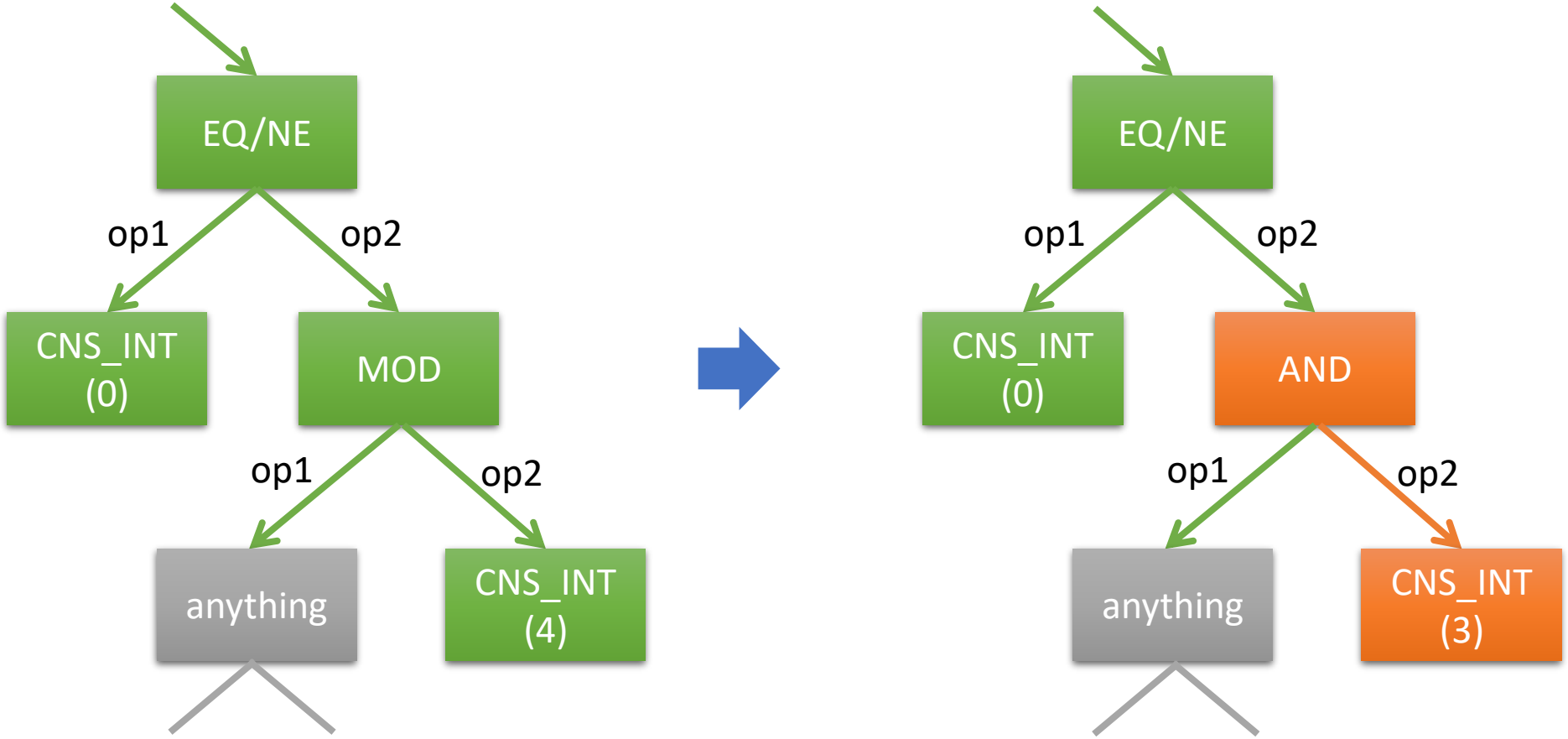


JIT: Transform X % C == 0 to X & (C-1) == 0 #25744

Open EgorBo wants to merge 9 commits into dotnet:master from EgorBo:jit-opt-mod

```
bool Test1(int x) => x % 4 == 0;
```

```
bool Test1(int x) => x & 3 == 0;
```



Roslyn and “!=“ operator
(unexpected optimization)

```
public static bool Test1(int x) => x != 42;
```

```
IL_0000: ldarg.0
IL_0001: ldc.i4.42
IL_0002: ceq
IL_0004: ldc.i4.0
IL_0005: ceq
IL_0007: ret
```

return (x == 42) == false

JIT: ok, it’s **GT_NE**

```
public static bool Test2(int x) => x != 0;
```

```
IL_0000: ldarg.0
IL_0001: ldc.i4.0
IL_0002: cgt.un
IL_0004: ret
```

return (uint)x > 0

JIT: what?.. ok, **GT_GT**

Optimize Math.Pow(x, c) in JIT #26552

Open EgorBo wants to merge 10 commits into `dotnet:master` from `EgorBo:math-pow`

```
Math.Pow(x, 2)    | x * x
Math.Pow(x, 1)    | x
Math.Pow(x, 4)    | x * x * x * x
Math.Pow(x, -1)   | 1 / x

// can be added:
Math.Pow(42, 3)   | 74088
Math.Pow(1, x)    | 1
Math.Pow(2, x)    | exp2(x)
Math.Pow(x, 0)    | 1
Math.Pow(x, 0.5)  | sqrt(x)
```

JIT: Improve constant folding for bitwise OR #27325

Open EgorBo wants to merge 5 commits into `dotnet:master` from `EgorBo:const-fold-or`

```
int Test1(int x) => x | 5 | 3;                  // x | 7

int Test2(int x, int y) => (x | 5) | (y | 3);  // x | y | 7
```

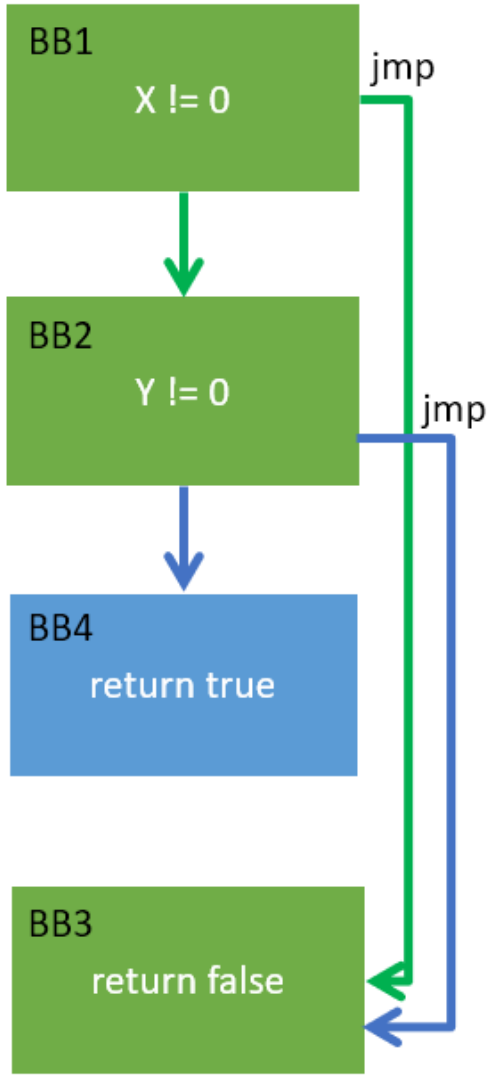
Expand BBJ_RETURN blocks with bool conditions #27167

Open EgorBo wants to merge 1 commit into `dotnet:master` from `EgorBo:fg-experiments`

```
bool AreZero (int x, int y)
{
    return x == 0 && y == 0;
}
```

Expand BBJ_RETURN blocks with bool conditions #27167

Open EgorBo wants to merge 1 commit into `dotnet:master` from `EgorBo:fg-experiments`



Expand BBJ_RETURN blocks with bool conditions #27167

Open EgorBo wants to merge 1 commit into `dotnet:master` from `EgorBo:fg-experiments`

```
bool AreZero (int x, int y)
{
    return x == 0 && y == 0;
}
```

```
G_M22205_IG01:
G_M22205_IG02:
        test      ecx, ecx
        jne       SHORT G_M22205_IG05
G_M22205_IG03:            ;; bbWeight=0.50
        test      edx, edx
        sete      al
        movzx     rax, al
G_M22205_IG04:            ;; bbWeight=0.50
        ret
G_M22205_IG05:            ;; bbWeight=0.50
        xor       eax, eax
G_M22205_IG06:            ;; bbWeight=0.50
        ret
; Total bytes of code: 16
```

```
G_M22205_IG01:
G_M22205_IG02:
        or        edx, ecx
        sete      al
        movzx     rax, al
G_M22205_IG03:
        ret
```

JIT: Optimize simple range checks with `uint` hack #27480

Open EgorBo wants to merge 12 commits into `dotnet:master` from `EgorBo:range-pattern`

Program.cs

ConsoleApp182 | C | Case2(int i)

```
1 reference
14  int Case1(int i, int[] array)
15  {
16      if (i < 0 || i >= array.Length) // if ((uint)i >= (uint)array.Length)
17          throw new ArgumentException();
18
19      return array[i];
20  }
21
```

100 % No issues found

tmpD432 - Copy.tmp vs. tmpD433.tmp

tmpD432 - Copy.tmp

```
1   ; Method C:Case1(int,ref):int:this
2
3   G_M63057_IG01:
4       push    rsi
5       sub     rsp, 32
6
7   G_M63057_IG02:
8       test    edx, edx
9       jl      SHORT G_M63057_IG05
10      mov     eax, dword ptr [r8+8]
11      cmp     eax, edx
12      jle     SHORT G_M63057_IG05
13
14  G_M63057_IG03:
15      cmp     edx, eax
16      jae     SHORT G_M63057_IG06
17      movsxd  rax, edx
18      mov     eax, dword ptr [r8+4*rax+16]
```

tmpD433.tmp

```
1   ; Method C:Case1(int,ref):int:this
2
3   G_M63057_IG01:
4       push    rsi
5       sub     rsp, 32
6
7   G_M63057_IG02:
8       mov     eax, dword ptr [r8+8]
9       cmp     eax, edx
10      jbe     SHORT G_M63057_IG05
11
12  G_M63057_IG03:
13      movsxd  rax, edx
14      mov     eax, dword ptr [r8+4*rax+16]
```

JIT: Optimize simple range checks with `uint` hack #27480

Open EgorBo wants to merge 12 commits into `dotnet:master` from `EgorBo:range-pattern`

Program.cs*

ConsoleApp182 | C | Main()

```
25
26
    1 reference
27  int Case2(int i)
28  {
29      if (i < 10 || i > 100)            // if ((uint)i - 10 > 90)
30          throw new ArgumentException();
31
32      return 42;
33  }
34
35
```

100 % No issues found

tmp92C2 - Copy.tmp vs. tmp92C3.tmp

tmp92C2 - Copy.tmp

```
1   ; Method C:Case2(int):int:this
2
3   G_M63058_IG01:
4           push    rsi
5           sub     rsp, 32
6
7   G_M63058_IG02:
8           cmp     edx, 10
9           jl      SHORT G_M63058_IG05
10          cmp     edx, 100
11          jg      SHORT G_M63058_IG05
12
13  G_M63058_IG03:
14          mov     eax, 42
15
16  G_M63058_IG04:
```

tmp92C3.tmp

```
1   ; Method C:Case2(int):int:this
2
3   G_M63058_IG01:
4           push    rsi
5           sub     rsp, 32
6
7   G_M63058_IG02:
8           add     edx, -10
9           cmp     edx, 90
10          ja      SHORT G_M63058_IG05
11
12  G_M63058_IG03:
13          mov     eax, 42
14
15  G_M63058_IG04:
```

Auto-vectorization

.cs

oleApp182 | Program | MyTest(float* array, int leng

```
0 references
unsafe class Program
{
    0 references
    static void Main()
    {
        float[] array = new float[1024];
        fixed (float* p = array)
            MyTest(p, 0);
    }


    1 reference
    static void MyTest(float* array, int length)
    {
        for (int i = 0; i < length; i++)
        {
            // partially unrolled loop:
            array[i + 0] = 0;
            array[i + 1] = 0;
            array[i + 2] = 0;
            array[i + 3] = 0;
            array[i + 4] = 0;
            array[i + 5] = 0;
            array[i + 6] = 0;
            array[i + 7] = 0;

            // JIT will replace it with
            // Avx.Store(a, Vector256<double>.Zero);
            //    or Vector256.Create(AnyValue);
        }
    }
}
```

Disasmo

CoreCLR Path: C:\prj\coreclr-baseline ... Refresh

Output | Previous output | Settings | S.R.Intrinsics | Tests

```
; Method Program:MyTest(long,int)

G_M57665_IG01:
        vzeroupper

G_M57665_IG02:
        xor     eax, eax
        test    edx, edx
        jle     SHORT G_M57665_IG04

G_M57665_IG03:      ;; bbWeight=4
        movsxd  r8, eax
        vxorps  xmm0, xmm0
        vmovss  dword ptr [rcx+4*r8], xmm0
        lea     r8d, [rax+1]
        movsxd  r9, r8d
        vmovss  dword ptr [rcx+4*r9], xmm0
        lea     r9d, [rax+2]
        movsxd  r9, r9d
        vmovss  dword ptr [rcx+4*r9], xmm0
        lea     r9d, [rax+3]
        movsxd  r9, r9d
        vmovss  dword ptr [rcx+4*r9], xmm0
        lea     r9d, [rax+4]
        movsxd  r9, r9d
        vmovss  dword ptr [rcx+4*r9], xmm0
        lea     r9d, [rax+5]
        movsxd  r9, r9d
        vmovss  dword ptr [rcx+4*r9], xmm0
        lea     r9d, [rax+6]
        movsxd  r9, r9d
        vmovss  dword ptr [rcx+4*r9], xmm0
        add     eax, 7
        movsxd  rax, eax
        vmovss  dword ptr [rcx+4*rax], xmm0
        mov     eax, r8d
        cmp     eax, edx
```

Disasmo

CoreCLR Path: C:\prj\coreclr-egor ... Refre

Output | Previous output | Settings | S.R.Intrinsics | Tests

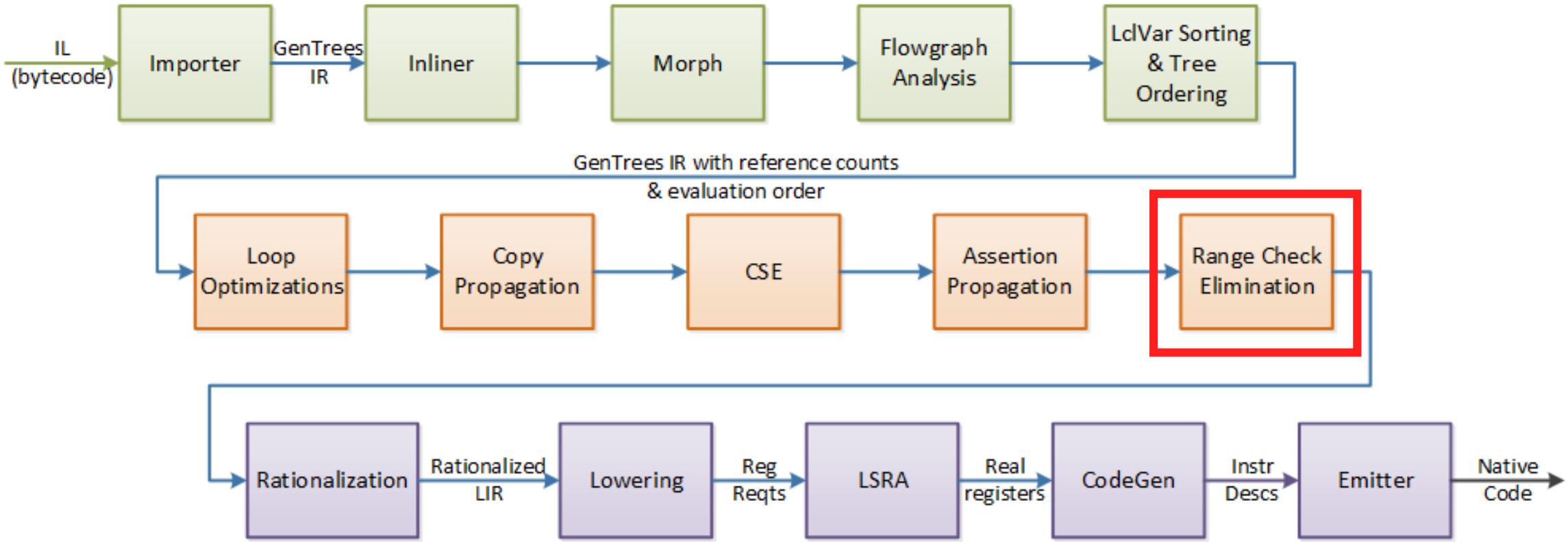
```
; Method Program:MyTest(long,int)

G_M57665_IG01:
        vzeroupper

G_M57665_IG02:
        xor     eax, eax
        test    edx, edx
        jle     SHORT G_M57665_IG04

G_M57665_IG03:      ;; bbWeight=4
AVX     vxorps  ymm0, ymm0, ymm0
        vmovups ymmword ptr[rcx], ymm0
        inc     eax
        cmp     eax, edx
        jl      SHORT G_M57665_IG03

G_M57665_IG04:
        vzeroupper
        ret
; Total bytes of code: 27
```

Auto-vectorization

```
***** BB01
STMT00000 (IL 0x000...0x007)
N009 (  8,  8) [000009] -A-XG-------             *  ASG       int
N007 (  6,  6) [000008] *--X---N----             +--*  IND       int    $44
N006 (  4,  5) [000006] -------N----             |  \--*  ADD       long   $142
N001 (  1,  1) [000000] ------------             |     +--*  LCL_VAR   long   V01 arg1
N005 (  3,  4) [000005] -------N----             |     \--*  LSH       long   $141
N003 (  2,  3) [000002] ------------             |        +--*  CAST      long <- int $140
N002 (  1,  1) [000001] ------------             |        |  \--*  LCL_VAR   int    V02 arg2
N004 (  1,  1) [000004] ------------             |        \--*  CNS_INT   long   2 $180
N008 (  1,  1) [000007] ------------             \--*  CNS_INT   int    0 $44

***** BB01
STMT00001 (IL 0x008...0x011)
N011 ( 10, 10) [000021] -A-XG-------             *  ASG       int
N009 (  8,  8) [000020] *--X---N----             +--*  IND       int    $44
N008 (  6,  7) [000018] -------N----             |  \--*  ADD       long   $145
N001 (  1,  1) [000010] ------------             |     +--*  LCL_VAR   long   V01 arg1
N007 (  5,  6) [000017] -------N----             |     \--*  LSH       long   $144
N005 (  4,  5) [000014] ------------             |        +--*  CAST      long <- int $143
N004 (  3,  3) [000013] ------------             |        |  \--*  ADD       int    $200
N002 (  1,  1) [000011] ------------             |        |     +--*  LCL_VAR   int    V02 arg2
N003 (  1,  1) [000012] ------------             |        |     \--*  CNS_INT   int    1 $40
N006 (  1,  1) [000016] ------------             |        \--*  CNS_INT   long   2 $180
N010 (  1,  1) [000019] ------------             \--*  CNS_INT   int    0 $44
```

…

Range check elimination



Range check elimination

```
public static void Test(int[] a)
{

    a[0] = 4;

    a[1] = 2;
    for (int i = 0; i < a.Length; i++)
    {

        a[i] = 0;
    }

    a[1] = 2;
}
```

Range check elimination

```
public static void Test(int[] a)
{
    if (a.Length <= 0) throw new IndexOutOfRangeException();
    a[0] = 4;
    if (a.Length <= 1) throw new IndexOutOfRangeException();
    a[1] = 2;
    for (int i = 0; i < a.Length; i++)
    {
        if (a.Length <= i) throw new IndexOutOfRangeException();
        a[i] = 0;
    }
    if (a.Length <= 2) throw new IndexOutOfRangeException();
    a[1] = 2;
}
```

Range check elimination

```
public static void Test(int[] a)
{
    if (a.Length <= 0) throw new IndexOutOfRangeException();
    a[0] = 4;
    if (a.Length <= 1) throw new IndexOutOfRangeException();
    a[1] = 2;
    for (int i = 0; i < a.Length; i++)
    {
        if (a.Length <= i) throw new IndexOutOfRangeException();
        a[i] = 0;
    }
    if (a.Length <= 2) throw new IndexOutOfRangeException();
    a[1] = 2;
}
```

Range check elimination

```
public static void Test(int[] a)
{
    if (a.Length <= 1) throw new IndexOutOfRangeException();
    a[1] = 2;
    if (a.Length <= 1) throw new IndexOutOfRangeException();
    a[0] = 4;
    for (int i = 0; i < a.Length; i++)
    {
        if (a.Length <= i) throw new IndexOutOfRangeException();
        a[i] = 0;
    }
    if (a.Length <= 2) throw new IndexOutOfRangeException();
    a[1] = 2;
}
```

Range check elimination

```
public static void Test(int[] a)
{
    if (a.Length <= 1) throw new IndexOutOfRangeException();
    a[1] = 2;
    a[0] = 4;
    for (int i = 0; i < a.Length; i++)
    {
        a[i] = 0;
    }
    a[1] = 2;
}
```

rangecheck.cpp (simplified)

```
void RangeCheck::OptimizeRangeCheck(GenTreeBoundsChk* bndsChk)
{
    // Get the range for this index.
    Range range = GetRange(...);

    // If upper or lower limit is unknown, then return.
    if (range.UpperLimit().IsUnknown() || range.LowerLimit().IsUnknown())
    {
        return;
    }

    // Is the range between the lower and upper bound values.
    if (BetweenBounds(range, 0, bndsChk->gtArrLen))
    {
        m_pCompiler->optRemoveRangeCheck(treeParent, stmt);
    }
    return;
}
```

rangecheck.cpp (simplified)

```
void RangeCheck::OptimizeRangeCheck(GenTreeBoundsChk* bndsChk)
{
    // Get the range for this index.
    Range range = GetRange(...);

    // If upper or lower limit is unknown, then return.
    if (range.UpperLimit().IsUnknown() || range.LowerLimit().IsUnknown())
    {
        return;
    }

    // Is the range between the lower and upper bound values.
    if (BetweenBounds(range, 0, bndsChk->gtArrLen))
    {
        m_pCompiler->optRemoveRangeCheck(treeParent, stmt);
    }
    return;
}
```

rangecheck.cpp (simplified)

```
void RangeCheck::OptimizeRangeCheck(GenTreeBoundsChk* bndsChk)
{
    // Get the range for this index.
    Range range = GetRange(...);

    // If upper or lower limit is unknown, then return.
    if (range.UpperLimit().IsUnknown() || range.LowerLimit().IsUnknown())
    {
        return;
    }

    // Is the range between the lower and upper bound values.
    if (BetweenBounds(range, 0, bndsChk->gtArrLen))
    {
        m_pCompiler->optRemoveRangeCheck(treeParent, stmt);
    }
    return;
}
```

Byte array

```
private static readonly byte[] _data =
        new byte[256] { 1, 2, 3, … };


public static byte GetByte(int i)
{
    return _data[i];
}
```

```
; Method P:GetByte(int):ubyte

G_M5240_IG01:
        push     rsi
        sub      rsp, 32
        mov      esi, ecx

G_M5240_IG02:
        mov      rcx, 0xD1FFAB1E
        mov      edx, 1
        call     CORINFO_HELP_GETSHARED_NONGCSTATIC_BASE
        mov      rax, 0xD1FFAB1E
        mov      rax, gword ptr [rax]
        cmp      esi, dword ptr [rax+8]
        jae      SHORT G_M5240_IG04
        movsxd   rdx, esi
        movzx    rax, byte  ptr [rax+rdx+16]

G_M5240_IG03:
        add      rsp, 32
        pop      rsi
        ret

G_M5240_IG04:
        call     CORINFO_HELP_RNGCHKFAIL
        int3
; Total bytes of code: 65
```

Byte array: Roslyn hack (new feature)

```
private static ReadOnlySpan<byte> _data =>
        new byte[256] { 1, 2, 3, … };


public static byte GetByte(int i)
{
    return _data[i];
}
```

```
; Method P:GetByte(int):ubyte

G_M5244_IG01:
        sub     rsp, 40

G_M5244_IG02:
        cmp     ecx, 256
        jae     SHORT G_M5244_IG04
        movsxd  rax, ecx
        mov     rdx, 0xD1FFAB1E
        movzx   rax, byte  ptr [rax+rdx]

G_M5244_IG03:
        add     rsp, 40
        ret

G_M5244_IG04:
        call    CORINFO_HELP_RNGCHKFAIL
        int3
; Total bytes of code: 37
```

Byte array: byte index (my PR)

```
private static ReadOnlySpan<byte> _data =>
        new byte[256] { 1, 2, 3, … };


public static byte GetByte(int i)
{
    return _data[(byte)i];
}
```

```
; Method Program:GetByte(int):ubyte

G_M30997_IG01:

G_M30997_IG02:
       movzx    rax, cl
       movsxd   rax, eax
       mov      rdx, 0xD1FFAB1E
       movzx    rax, byte  ptr [rax+rdx]

G_M30997_IG03:
       ret
; Total bytes of code: 21
```

Byte indexer will never go out of bounds!

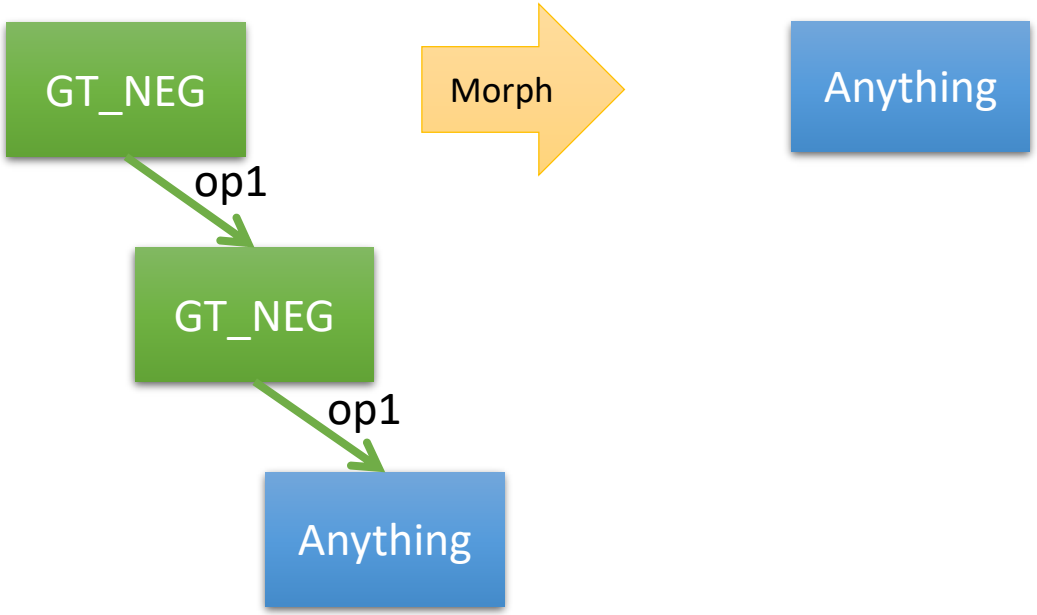
rangecheck.cpp (simplified)

```
void RangeCheck::OptimizeRangeCheck(GenTreeBoundsChk* bndsChk)
{
    // Get the range for this index.
    Range range = GetRange(...);    [Byte.MinValue ... Byte.MaxValue]

    // If upper or lower limit is unknown, then return.
    if (range.UpperLimit().IsUnknown() || range.LowerLimit().IsUnknown())
    {
        return;
    }

    // Is the range between the lower and upper bound values.
    if (BetweenBounds(range, 0, bndsChk->gtArrLen))  ArrLen = 256
    {
        m_pCompiler->optRemoveRangeCheck(treeParent, stmt);
    }
    return;
}
```

Homework! Fix JIT: Optimize -(-x) to x #27442

Open EgorBo opened this issue 4 days ago · 1 comment

```
int Foo(int a)
{
    return -(-a);
}
```



1) Clone CoreCLR repo
2) Build it: build.cmd –checked –skiptests
3) Open CoreCLR.sln
4) Optional: follow debugging-instructions.md
5) Open morph.cpp, line ~12755 (`case: GT_NEG`)
6) Optimize ☺

Loop-related optimizations

Loop Invariant Code Hoisting

```
public static bool Test(int[] a, int c)
{
    for (int i = 0; i < a.Length; i++)
    {
        if (a[i] == c + 44)
            return false;
    }
    return true;
}
```

Loop Invariant Code Hoisting

```
public static bool Test(int[] a, int c)
{
    int tmp = c + 44;
    for (int i = 0; i < a.Length; i++)
    {
        if (a[i] == tmp)
            return false;
    }
    return true;
}
```

NYI: Loop-unrolling

```
public static int Test(int[] a)
{
    int sum = 0;
    for (int i = 0; i < a.Length; i++)
    {
        sum += a[i];
    }
    return sum;
}
```

NYI: Loop-unrolling

```
public static int Test(int[] a)
{
    int sum = 0;
    for (int i = 0; i < a.Length - 3; i += 4)
    {
        sum += a[i];
        sum += a[i+1];
        sum += a[i+2];
        sum += a[i+3];
    }
    return sum;
}
```

NYI: Loop-unswitch

```
public static int Test(int[] a, bool condition)
{
    int agr = 0;
    for (int i = 0; i < a.Length; i++)
    {
        if (condition)
            agr += a[i];
        else
            agr *= a[i];
    }
    return agr;
}
```

NYI: Loop-unswitch

```
public static int Test (int[] a, bool condition)
{
    int agr = 0;
    if (condition)
        for (int i = 0; i < a.Length; i++)
            agr += a[i];
    else
        for (int i = 0; i < a.Length; i++)
            agr *= a[i];
    return agr;
}
```

NYI: Loop-deletion

```
public static void Test()
{
    for (int i = 0; i < 10; i++) { }
}
```

```
; Method Program:Test()
G_M44187_IG01:
G_M44187_IG02:
    xor     eax, eax
G_M44187_IG03:
    inc     eax
    cmp     eax, 10
    jl      SHORT G_M44187_IG03
G_M44187_IG04:
    ret
; Total bytes of code: 10
```

NYI: Loop-deletion

```
public static void Test()
{
}
```

```
; Method Program:DeadLoop()
    ret
; Total bytes of code: 1
```

NYI: InductiveRangeCheckElimination

```
public static void Zero1000Elements(int[] array)
{
    for (int i = 0; i < 1000; i++)
        array[i] = 0; // bound checks will be inserted here
}
```

NYI: InductiveRangeCheckElimination

```
public static void Zero1000Elements(int[] array)
{
    int limit = Math.Min(array.Length, 1000);

    for (int i = 0; i < limit; i++)
        array[i] = 0; // bound checks are not needed here!

    for (int i = limit; i < 1000; i++)
        array[i] = 0; // bound checks are needed here

    // so at least we could "zero" first `limit` elements without bound checks
}
```

NOTE: this LLVM optimization pass is not enabled by default in `opt –O2`.
Contributed by Azul developers (LLVM for JVM)

NYI: InductiveRangeCheckElimination

```
public static void Zero1000Elements(int[] array)
{
    int limit = Math.Min(array.Length, 1000);

    for (int i = 0; i < limit - 3; i += 4)
    {
        array[i] = 0;
        array[i+1] = 0;
        array[i+2] = 0;
        array[i+3] = 0;
    }

    for (int i = limit; i < 1000; i++)
        array[i] = 0; // bound checks are needed here

    // so at least we could "zero" first `limit` elements without bound checks
}
```

Now we can even unroll the first loop!

NYI: InductiveRangeCheckElimination

```
public static void Zero1000Elements(int[] array)
{
    int limit = Math.Min(array.Length, 1000);

    memset(array, 0, limit);
```

Or just replace with memset call

```
    for (int i = limit; i < 1000; i++)
        array[i] = 0; // bound checks are needed here

    // so at least we could "zero" first `limit` elements without bound checks
}
```

Q&A

Twitter: EgorBo