Оптимизации в JIT .NET 10

Дмитрий Егоров
.NET разработчик, Ozon

План

- Escape analysis
- Bound check
- Inline

ozon{ech

Escape analysis

Зачем ?

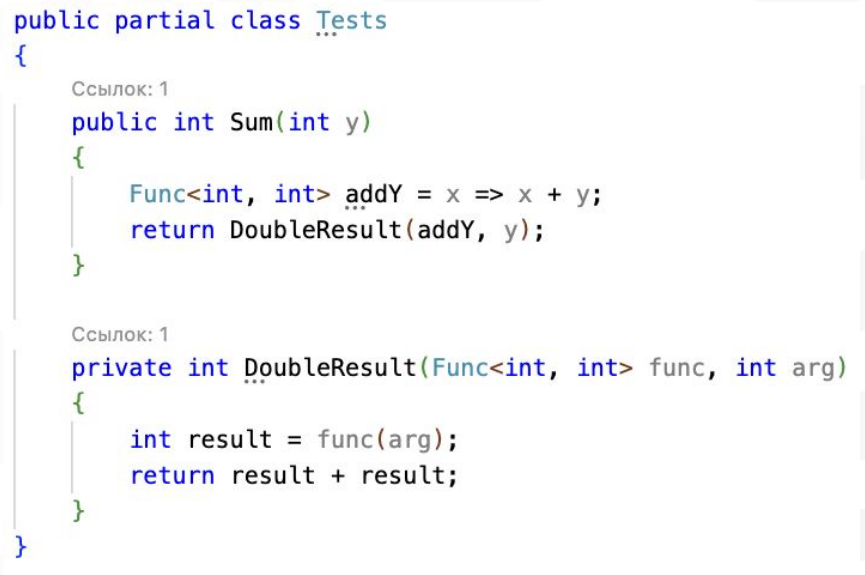
Позволяет не бояться создавать новые классы

Аллокация массивов на стеке

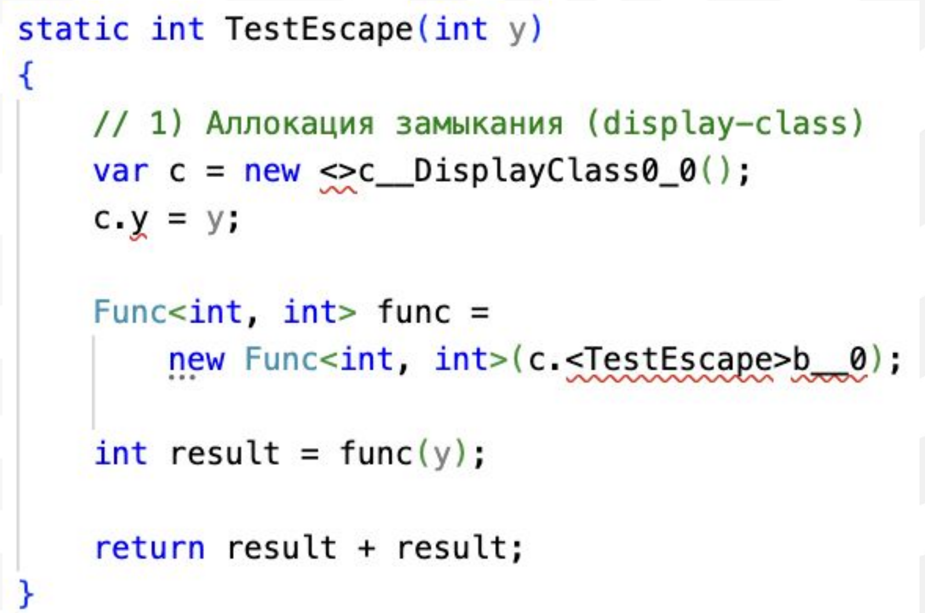
```csharp
public static void Sum(int a, int b, int c)
{
    var array = new[] {a, b, c };
    Process(array);
}
```

```
mov    rdi, 0x7FE79B338D28
mov    qword ptr [rbp-0x30], rdi     ; *MT (method table) для int[]
lea    rdi, [rbp-0x30]
mov    dword ptr [rdi+0x08], 3       ; Length = 3
lea    rbx, [rbp-0x30]


mov    rdi, 0x7FE815169B10
mov    rax, qword ptr [rdi]          ; первые 8 байт (обычно 1 и 2 упакованы)
mov    qword ptr [rbx+0x10], rax     ; elements[0..1]
mov    eax, dword ptr [rdi+0x08]     ; третий int (3)
mov    dword ptr [rbx+0x18], eax     ; elements[2]
```

Аллокация на стеке и замыкание: было

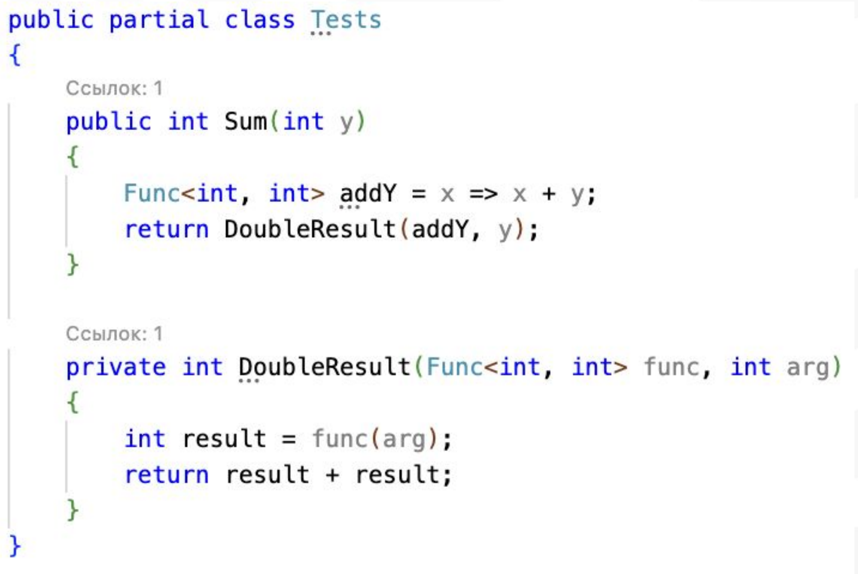


```
public partial class Tests
{
    Ссылок: 1
    public int Sum(int y)
    {
        Func<int, int> addY = x => x + y;
        return DoubleResult(addY, y);
    }

    Ссылок: 1
    private int DoubleResult(Func<int, int> func, int arg)
    {
        int result = func(arg);
        return result + result;
    }
}
```

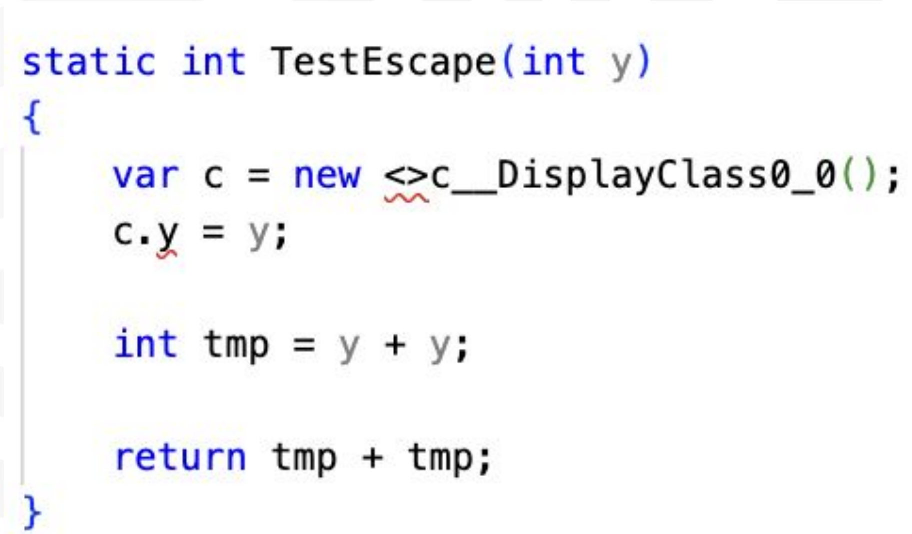


```
static int TestEscape(int y)
{
    // 1) Аллокация замыкания (display-class)
    var c = new <>c__DisplayClass0_0();
    c.y = y;

    Func<int, int> func =
        new Func<int, int>(c.<TestEscape>b__0);

    int result = func(y);

    return result + result;
}
```

Аллокация на стеке и замыкание: стало



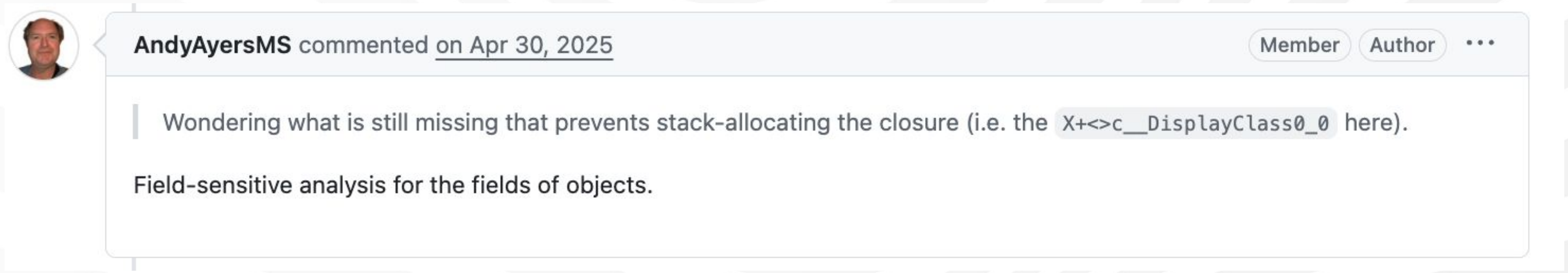
```
public partial class Tests
{
    Ссылок: 1
    public int Sum(int y)
    {
        Func<int, int> addY = x => x + y;
        return DoubleResult(addY, y);
    }

    Ссылок: 1
    private int DoubleResult(Func<int, int> func, int arg)
    {
        int result = func(arg);
        return result + result;
    }
}
```



```
static int TestEscape(int y)
{
    var c = new <>c__DisplayClass0_0();
    c.y = y;

    int tmp = y + y;

    return tmp + tmp;
}
```

Sensitive Escape analysis

```
push      rbx
mov       ebx,esi // y сохранили сюда
mov       rdi,offset MT_Tests+<>c__DisplayClass0_0
call      CORINFO_HELP_NEWSFAST
mov       [rax+8],ebx
mov       eax,[rax+8] // eax ссылка на класс, сюда y
mov       ecx,eax
add       eax,ecx
add       eax,eax
pop       rbx
ret
```

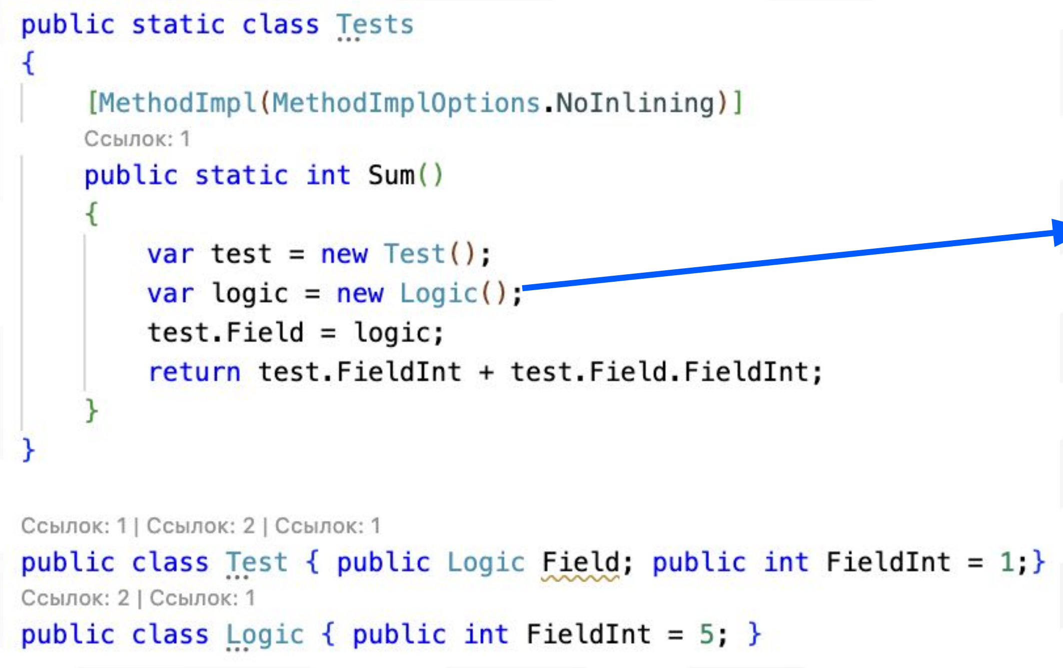


AndyAyersMS commented on Apr 30, 2025 · Member · Author

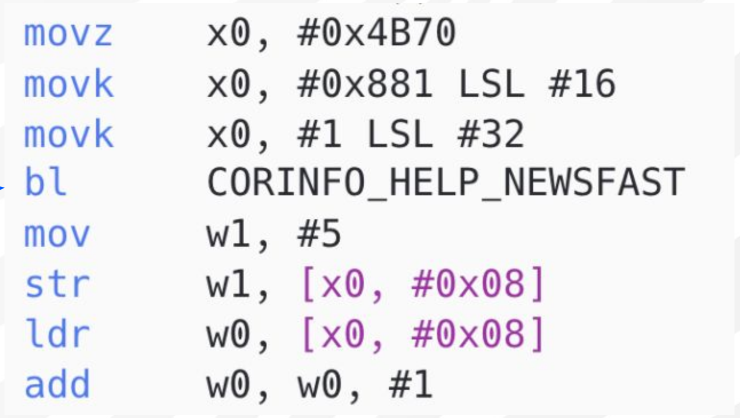
> Wondering what is still missing that prevents stack-allocating the closure (i.e. the `X+<>c__DisplayClass0_0` here).

Field-sensitive analysis for the fields of objects.

Sensitive Escape analysis

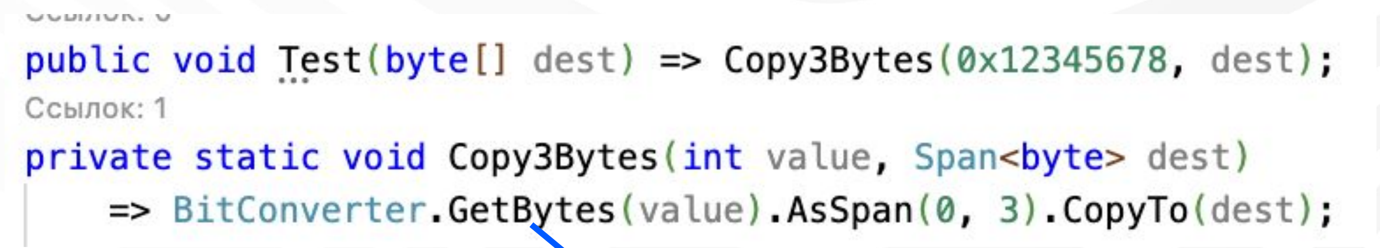


```
public static class Tests
{
    [MethodImpl(MethodImplOptions.NoInlining)]
    Ссылок: 1
    public static int Sum()
    {
        var test = new Test();
        var logic = new Logic();
        test.Field = logic;
        return test.FieldInt + test.Field.FieldInt;
    }
}

Ссылок: 1 | Ссылок: 2 | Ссылок: 1
public class Test { public Logic Field; public int FieldInt = 1;}
Ссылок: 2 | Ссылок: 1
public class Logic { public int FieldInt = 5; }
```

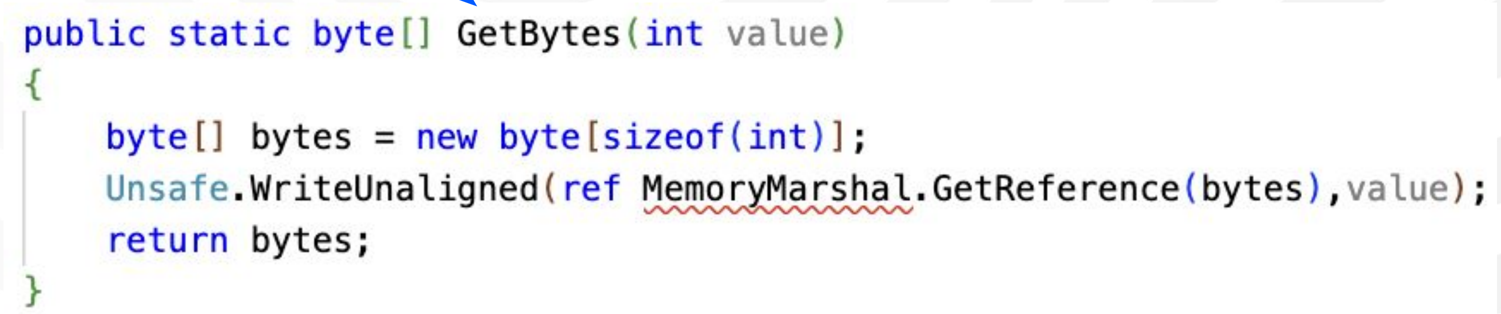


```
movz    x0, #0x4B70
movk    x0, #0x881 LSL #16
movk    x0, #1 LSL #32
bl      CORINFO_HELP_NEWSFAST
mov     w1, #5
str     w1, [x0, #0x08]
ldr     w0, [x0, #0x08]
add     w0, w0, #1
```

Sensitive Escape analysis

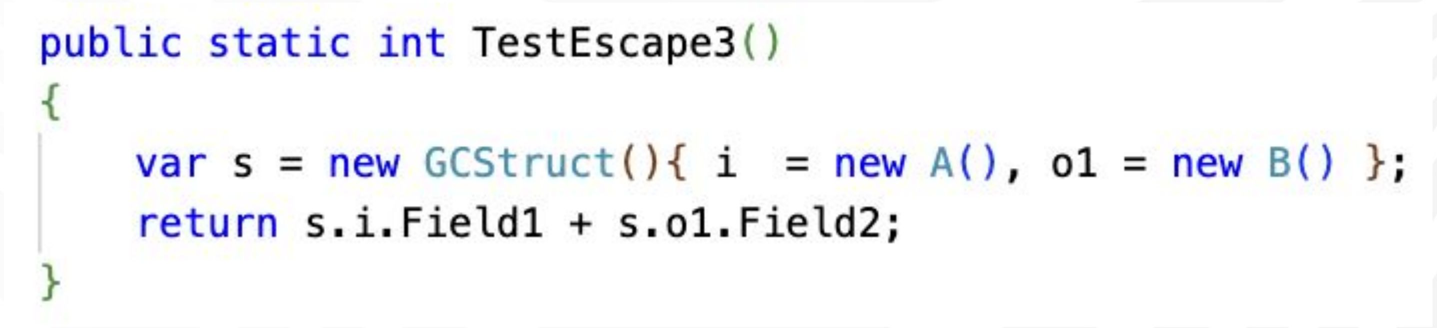


```
public void Test(byte[] dest) => Copy3Bytes(0x12345678, dest);
Ссылок: 1
private static void Copy3Bytes(int value, Span<byte> dest)
    => BitConverter.GetBytes(value).AsSpan(0, 3).CopyTo(dest);
```

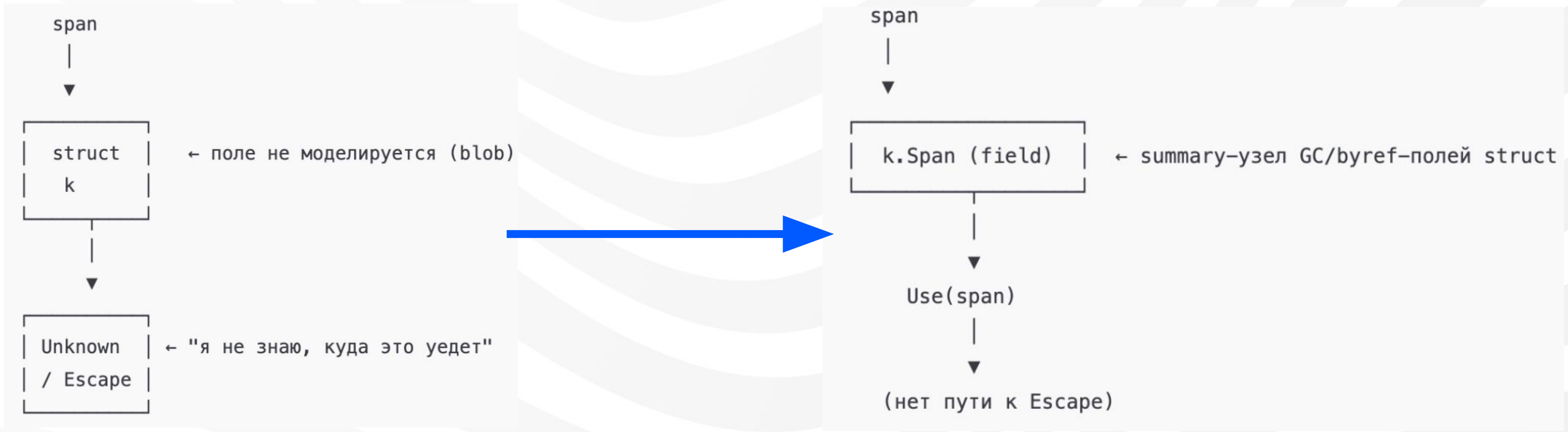


```
public static byte[] GetBytes(int value)
{
    byte[] bytes = new byte[sizeof(int)];
    Unsafe.WriteUnaligned(ref MemoryMarshal.GetReference(bytes),value);
    return bytes;
}
```

Sensitive Escape analysis

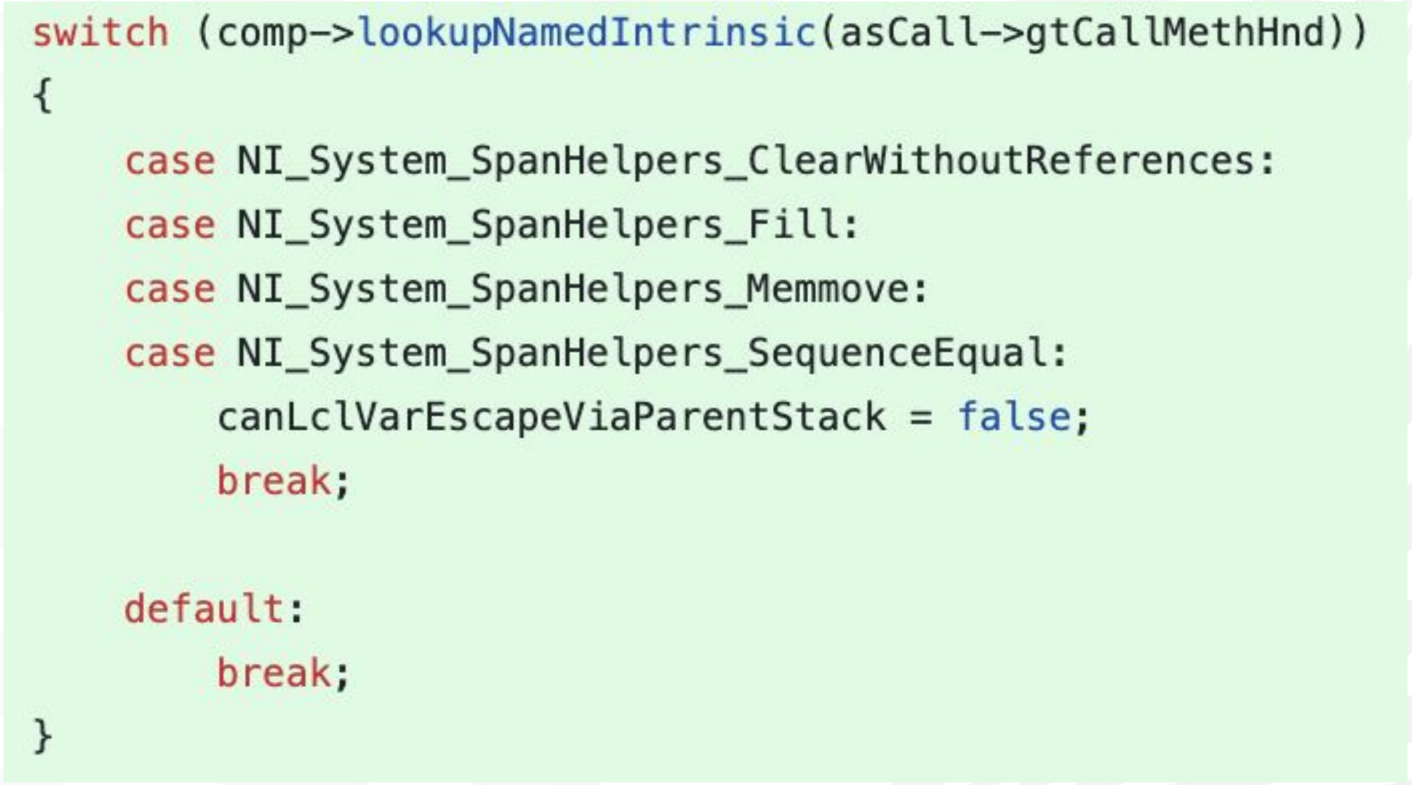


```
public static int TestEscape3()
{
    var s = new GCStruct(){ i  = new A(), o1 = new B() };
    return s.i.Field1 + s.o1.Field2;
}
```



span
struct k
← поле не моделируется (blob)
Unknown / Escape
← "я не знаю, куда это уедет"
span
k.Span (field)
← summary-узел GC/byref-полей struct
Use(span)
(нет пути к Escape)

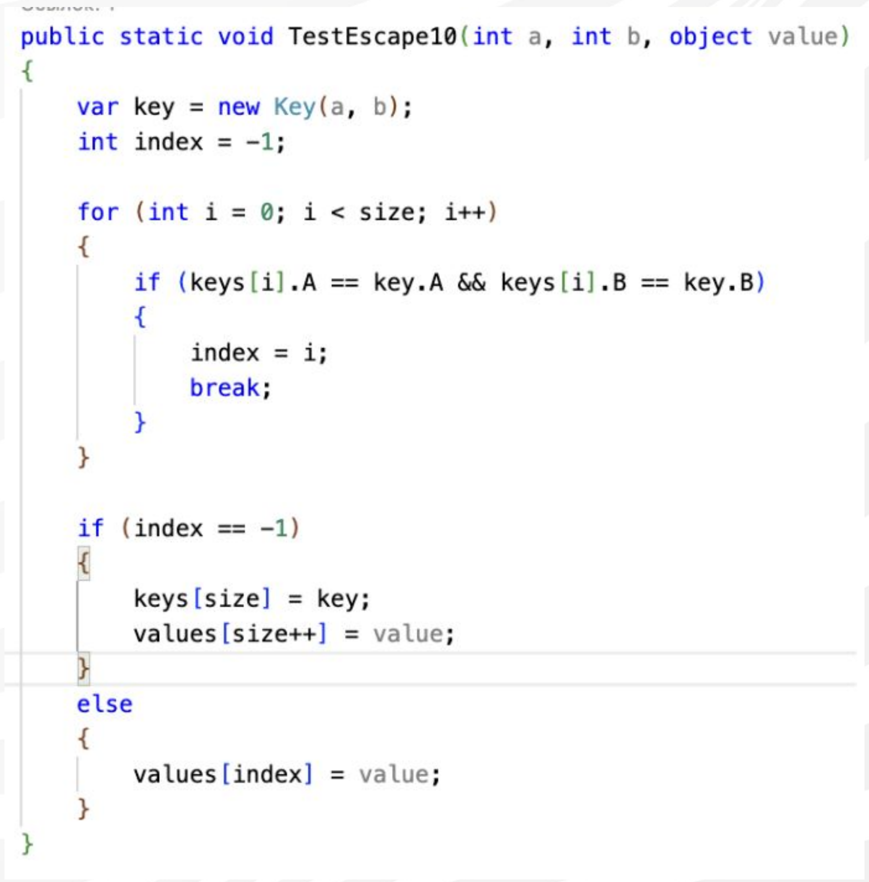
Memmove создавал утечку



```
switch (comp->lookupNamedIntrinsic(asCall->gtCallMethHnd))
{
    case NI_System_SpanHelpers_ClearWithoutReferences:
    case NI_System_SpanHelpers_Fill:
    case NI_System_SpanHelpers_Memmove:
    case NI_System_SpanHelpers_SequenceEqual:
        canLclVarEscapeViaParentStack = false;
        break;

    default:
        break;
}
```

Точка роста

1. Циклы
2. Partial escape
3. Circular references
4. String
5. boxed nullables
6. fields of unescaped ref classes



```
public static void TestEscape10(int a, int b, object value)
{
    var key = new Key(a, b);
    int index = -1;

    for (int i = 0; i < size; i++)
    {
        if (keys[i].A == key.A && keys[i].B == key.B)
        {
            index = i;
            break;
        }
    }

    if (index == -1)
    {
        keys[size] = key;
        values[size++] = value;
    }
    else
    {
        values[index] = value;
    }
}
```

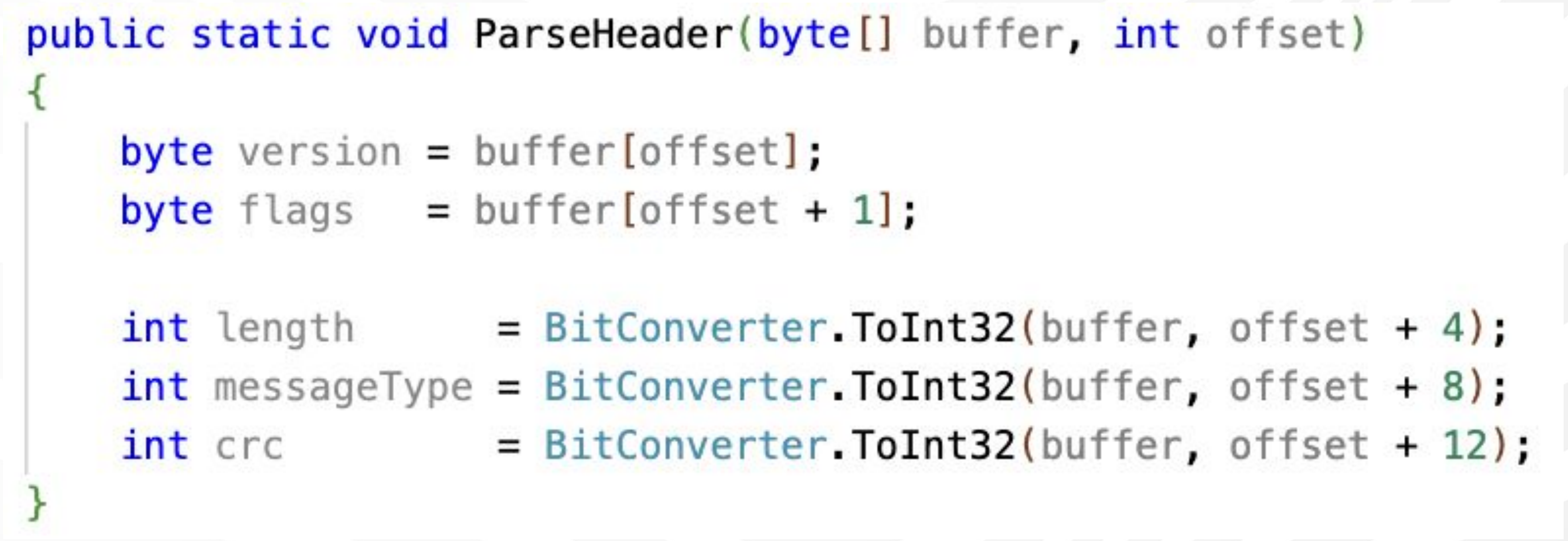
ozon{ech

Bound Check

Зачем?

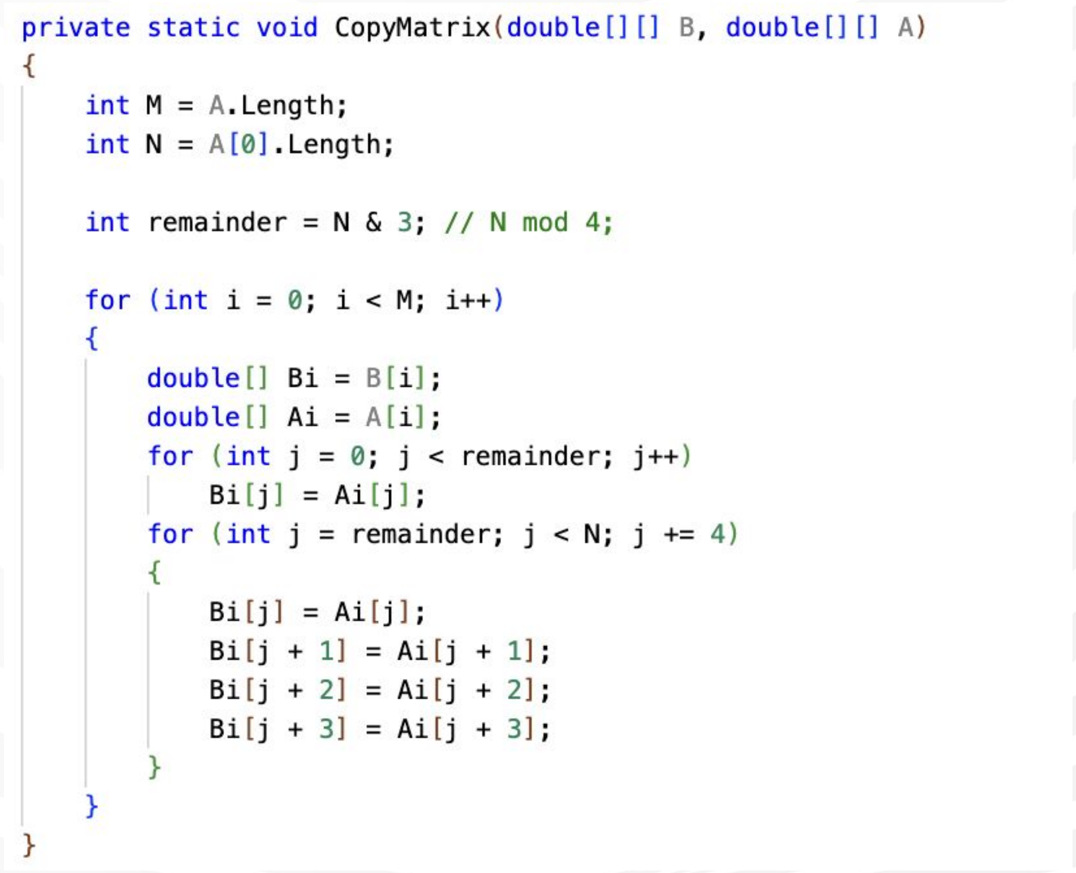
Memory model

Bound check: кейсы



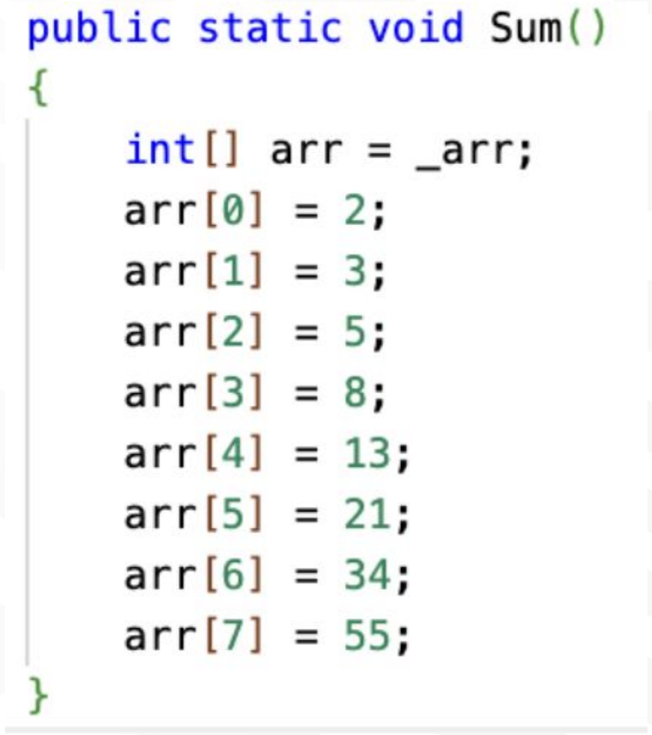
```
public static void ParseHeader(byte[] buffer, int offset)
{
    byte version = buffer[offset];
    byte flags   = buffer[offset + 1];

    int length      = BitConverter.ToInt32(buffer, offset + 4);
    int messageType = BitConverter.ToInt32(buffer, offset + 8);
    int crc         = BitConverter.ToInt32(buffer, offset + 12);
}
```

Матрицы

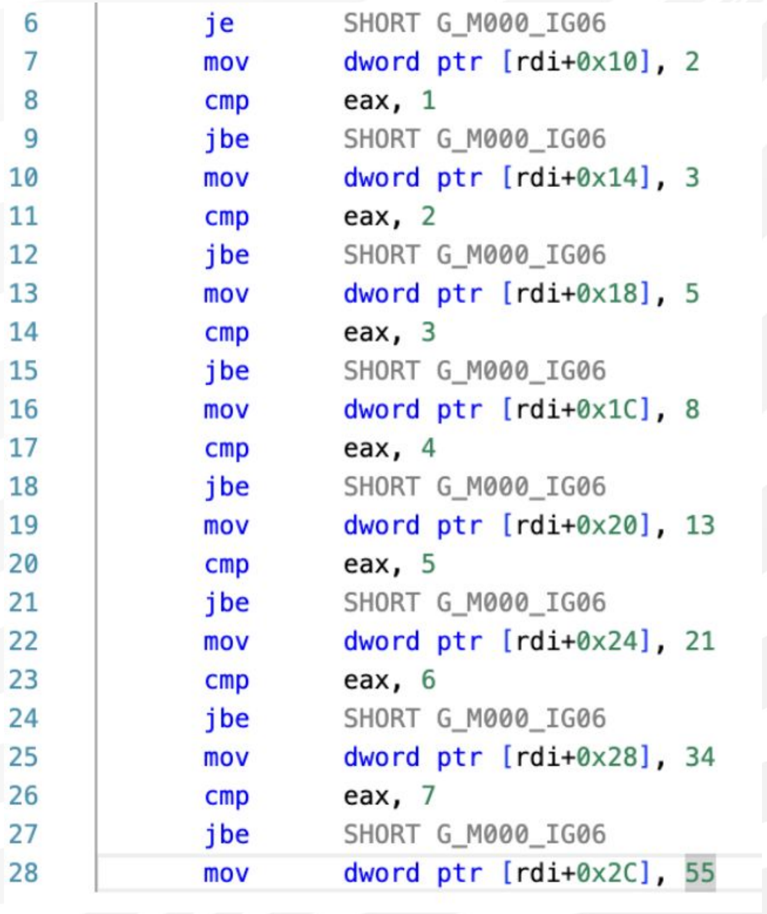


```
private static void CopyMatrix(double[][] B, double[][] A)
{
    int M = A.Length;
    int N = A[0].Length;

    int remainder = N & 3; // N mod 4;

    for (int i = 0; i < M; i++)
    {
        double[] Bi = B[i];
        double[] Ai = A[i];
        for (int j = 0; j < remainder; j++)
            Bi[j] = Ai[j];
        for (int j = remainder; j < N; j += 4)
        {
            Bi[j] = Ai[j];
            Bi[j + 1] = Ai[j + 1];
            Bi[j + 2] = Ai[j + 2];
            Bi[j + 3] = Ai[j + 3];
        }
    }
}
```

Было

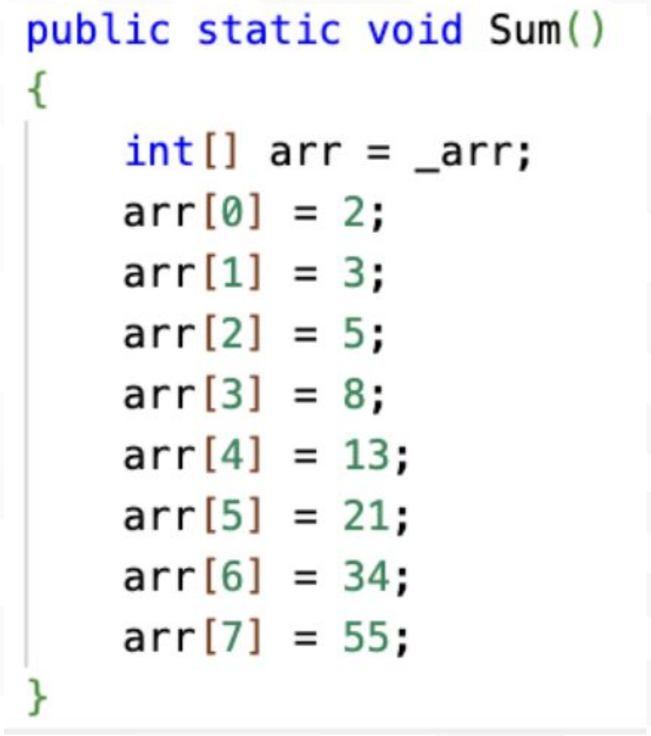


```
public static void Sum()
{
    int[] arr = _arr;
    arr[0] = 2;
    arr[1] = 3;
    arr[2] = 5;
    arr[3] = 8;
    arr[4] = 13;
    arr[5] = 21;
    arr[6] = 34;
    arr[7] = 55;
}
```

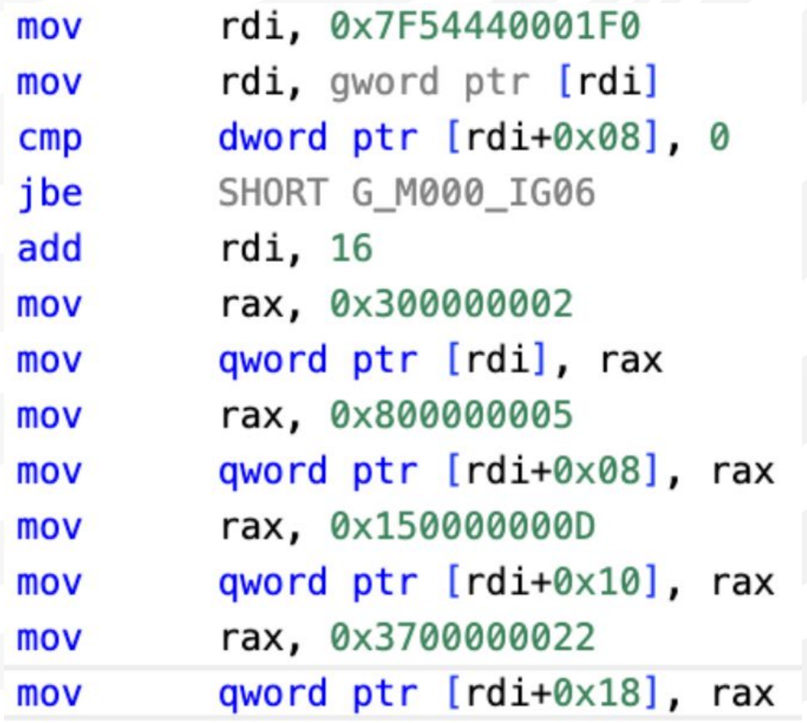


```
       je       SHORT G_M000_IG06
       mov      dword ptr [rdi+0x10], 2
       cmp      eax, 1
       jbe      SHORT G_M000_IG06
       mov      dword ptr [rdi+0x14], 3
       cmp      eax, 2
       jbe      SHORT G_M000_IG06
       mov      dword ptr [rdi+0x18], 5
       cmp      eax, 3
       jbe      SHORT G_M000_IG06
       mov      dword ptr [rdi+0x1C], 8
       cmp      eax, 4
       jbe      SHORT G_M000_IG06
       mov      dword ptr [rdi+0x20], 13
       cmp      eax, 5
       jbe      SHORT G_M000_IG06
       mov      dword ptr [rdi+0x24], 21
       cmp      eax, 6
       jbe      SHORT G_M000_IG06
       mov      dword ptr [rdi+0x28], 34
       cmp      eax, 7
       jbe      SHORT G_M000_IG06
       mov      dword ptr [rdi+0x2C], 55
```

Стало



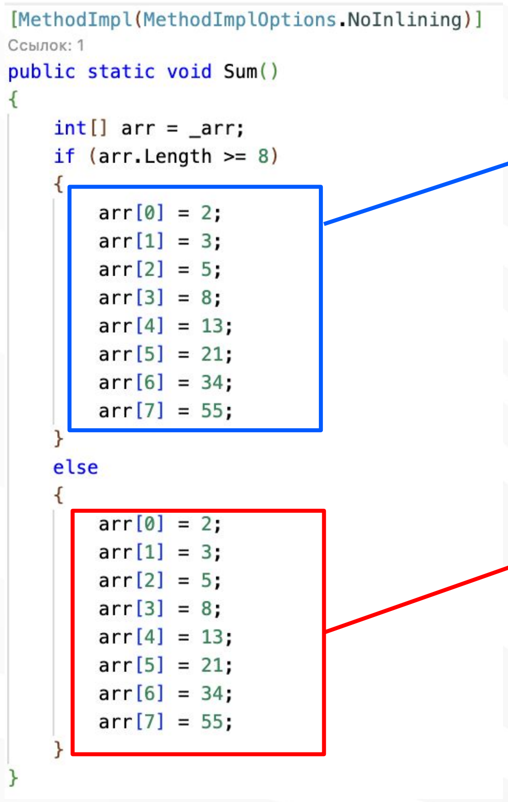
```csharp
public static void Sum()
{
    int[] arr = _arr;
    arr[0] = 2;
    arr[1] = 3;
    arr[2] = 5;
    arr[3] = 8;
    arr[4] = 13;
    arr[5] = 21;
    arr[6] = 34;
    arr[7] = 55;
}
```



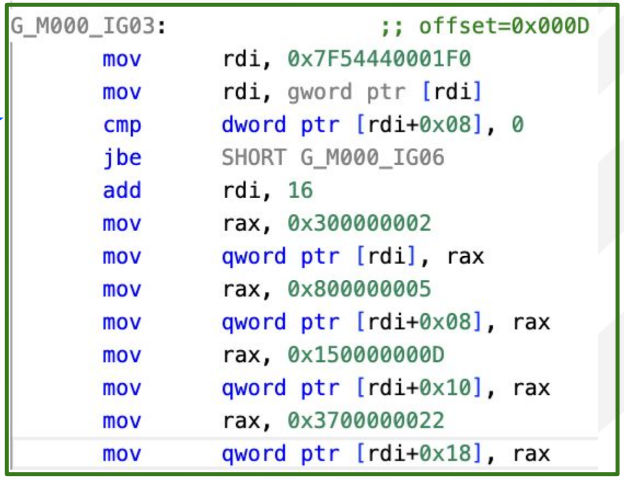
```asm
mov      rdi, 0x7F54440001F0
mov      rdi, gword ptr [rdi]
cmp      dword ptr [rdi+0x08], 0
jbe      SHORT G_M000_IG06
add      rdi, 16
mov      rax, 0x300000002
mov      qword ptr [rdi], rax
mov      rax, 0x800000005
mov      qword ptr [rdi+0x08], rax
mov      rax, 0x150000000D
mov      qword ptr [rdi+0x10], rax
mov      rax, 0x3700000022
mov      qword ptr [rdi+0x18], rax
```

ozon{ech

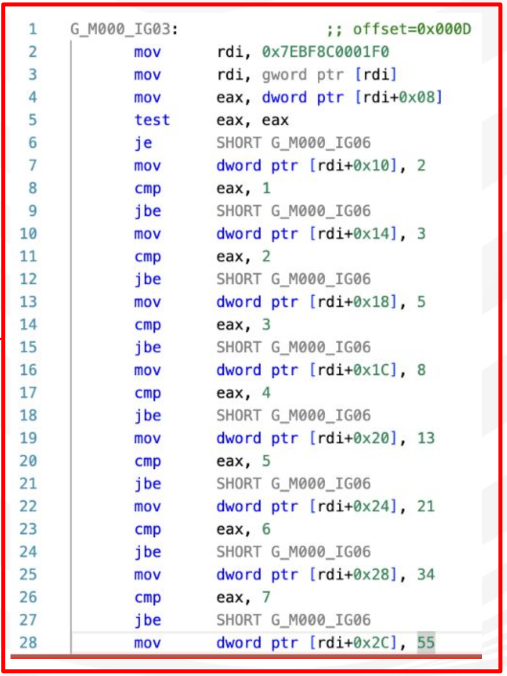
Code cloning



```
[MethodImpl(MethodImplOptions.NoInlining)]
Ссылок: 1
public static void Sum()
{
    int[] arr = _arr;
    if (arr.Length >= 8)
    {
        arr[0] = 2;
        arr[1] = 3;
        arr[2] = 5;
        arr[3] = 8;
        arr[4] = 13;
        arr[5] = 21;
        arr[6] = 34;
        arr[7] = 55;
    }
    else
    {
        arr[0] = 2;
        arr[1] = 3;
        arr[2] = 5;
        arr[3] = 8;
        arr[4] = 13;
        arr[5] = 21;
        arr[6] = 34;
        arr[7] = 55;
    }
}
```



```
G_M000_IG03:                ;; offset=0x000D
       mov      rdi, 0x7F54440001F0
       mov      rdi, gword ptr [rdi]
       cmp      dword ptr [rdi+0x08], 0
       jbe      SHORT G_M000_IG06
       add      rdi, 16
       mov      rax, 0x300000002
       mov      qword ptr [rdi], rax
       mov      rax, 0x800000005
       mov      qword ptr [rdi+0x08], rax
       mov      rax, 0x150000000D
       mov      qword ptr [rdi+0x10], rax
       mov      rax, 0x3700000022
       mov      qword ptr [rdi+0x18], rax
```



```
G_M000_IG03:                ;; offset=0x000D
       mov      rdi, 0x7EBF8C0001F0
       mov      rdi, gword ptr [rdi]
       mov      eax, dword ptr [rdi+0x08]
       test     eax, eax
       je       SHORT G_M000_IG06
       mov      dword ptr [rdi+0x10], 2
       cmp      eax, 1
       jbe      SHORT G_M000_IG06
       mov      dword ptr [rdi+0x14], 3
       cmp      eax, 2
       jbe      SHORT G_M000_IG06
       mov      dword ptr [rdi+0x18], 5
       cmp      eax, 3
       jbe      SHORT G_M000_IG06
       mov      dword ptr [rdi+0x1C], 8
       cmp      eax, 4
       jbe      SHORT G_M000_IG06
       mov      dword ptr [rdi+0x20], 13
       cmp      eax, 5
       jbe      SHORT G_M000_IG06
       mov      dword ptr [rdi+0x24], 21
       cmp      eax, 6
       jbe      SHORT G_M000_IG06
       mov      dword ptr [rdi+0x28], 34
       cmp      eax, 7
       jbe      SHORT G_M000_IG06
       mov      dword ptr [rdi+0x2C], 55
```

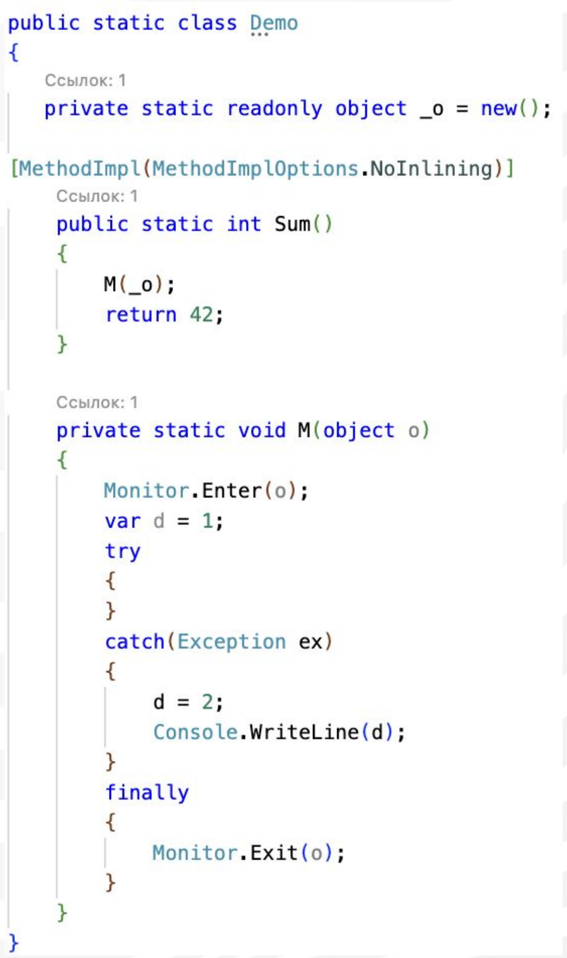
Inline

Зачем ?

Для всех других оптимизаций

Inline: try/finally

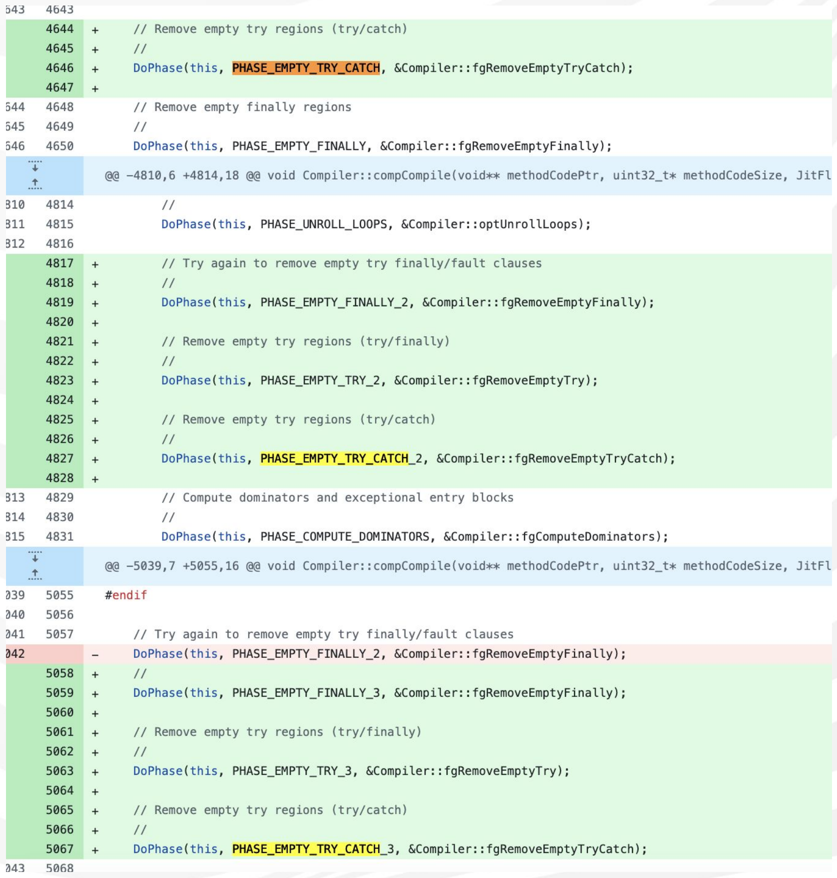
1. Теперь это не ограничение
2. Хитрость в удалении catch блока, с ним бы не встроилось



```
public static class Demo
{
    Ссылок: 1
    private static readonly object _o = new();

[MethodImpl(MethodImplOptions.NoInlining)]
    Ссылок: 1
    public static int Sum()
    {
        M(_o);
        return 42;
    }

    Ссылок: 1
    private static void M(object o)
    {
        Monitor.Enter(o);
        var d = 1;
        try
        {
        }
        catch(Exception ex)
        {
            d = 2;
            Console.WriteLine(d);
        }
        finally
        {
            Monitor.Exit(o);
        }
    }
}
```

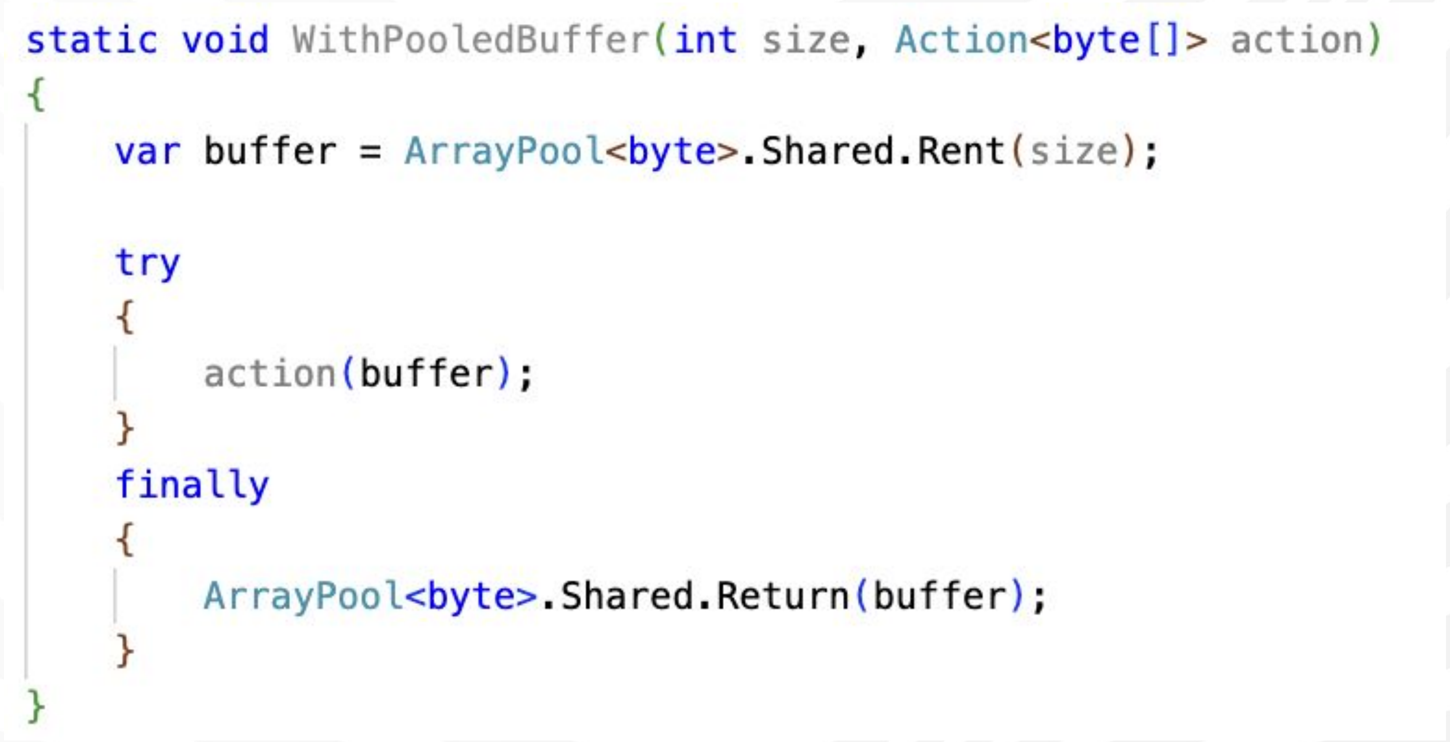
Удаление try/catch

1) Tiar
2) JitOptRepeat
3) Some try phase



```diff
+    // Remove empty try regions (try/catch)
+    //
+    DoPhase(this, PHASE_EMPTY_TRY_CATCH, &Compiler::fgRemoveEmptyTryCatch);
+
     // Remove empty finally regions
     //
     DoPhase(this, PHASE_EMPTY_FINALLY, &Compiler::fgRemoveEmptyFinally);
@@ -4810,6 +4814,18 @@ void Compiler::compCompile(void** methodCodePtr, uint32_t* methodCodeSize, JitFl
         //
         DoPhase(this, PHASE_UNROLL_LOOPS, &Compiler::optUnrollLoops);

+        // Try again to remove empty try finally/fault clauses
+        //
+        DoPhase(this, PHASE_EMPTY_FINALLY_2, &Compiler::fgRemoveEmptyFinally);
+
+        // Remove empty try regions (try/finally)
+        //
+        DoPhase(this, PHASE_EMPTY_TRY_2, &Compiler::fgRemoveEmptyTry);
+
+        // Remove empty try regions (try/catch)
+        //
+        DoPhase(this, PHASE_EMPTY_TRY_CATCH_2, &Compiler::fgRemoveEmptyTryCatch);
+
         // Compute dominators and exceptional entry blocks
         //
         DoPhase(this, PHASE_COMPUTE_DOMINATORS, &Compiler::fgComputeDominators);
@@ -5039,7 +5055,16 @@ void Compiler::compCompile(void** methodCodePtr, uint32_t* methodCodeSize, JitFl
 #endif

     // Try again to remove empty try finally/fault clauses
-    DoPhase(this, PHASE_EMPTY_FINALLY_2, &Compiler::fgRemoveEmptyFinally);
+    //
+    DoPhase(this, PHASE_EMPTY_FINALLY_3, &Compiler::fgRemoveEmptyFinally);
+
+    // Remove empty try regions (try/finally)
+    //
+    DoPhase(this, PHASE_EMPTY_TRY_3, &Compiler::fgRemoveEmptyTry);
+
+    // Remove empty try regions (try/catch)
+    //
+    DoPhase(this, PHASE_EMPTY_TRY_CATCH_3, &Compiler::fgRemoveEmptyTryCatch);

```

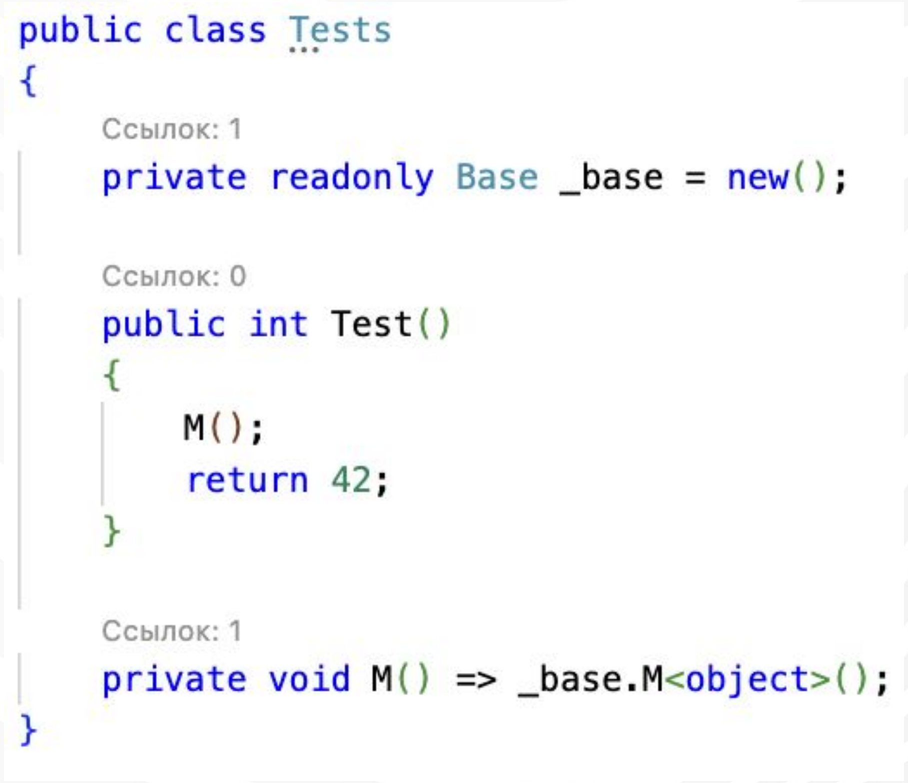
Inline: try/finally. Кейс



```
static void WithPooledBuffer(int size, Action<byte[]> action)
{
    var buffer = ArrayPool<byte>.Shared.Rent(size);

    try
    {
        action(buffer);
    }
    finally
    {
        ArrayPool<byte>.Shared.Return(buffer);
    }
}
```

ozon{ech

Inline: method with call GVM



```
public class Tests
{
    Ссылок: 1
    private readonly Base _base = new();

    Ссылок: 0
    public int Test()
    {
        M();
        return 42;
    }

    Ссылок: 1
    private void M() => _base.M<object>();
}
```

Хотелось рассказать, но не успел

Хотелось рассказать, но не успел

1. У Ref struct удаляется вставка GC Write Barriers, если свойство ссылочное.
2. IEnumerable → GetEnumerator → MoveNext → Current → Dispose стало еще ближе к for (i = 0; i < len; i++) (а где-то и неожиданно лучше). Большая работа.
3. Улучшения Code Layout.

https://devblogs.microsoft.com/dotnet/performance-improvements-in-net-10

Всем спасибо за внимание

@dimoner1