

A surreal image featuring two people in a green canoe on a body of water. The person on the left is wearing a light blue hoodie and blue pants, holding a paddle. The person on the right is wearing a green and white striped hoodie and dark pants. The water is a deep teal color, and the sky is a pale, hazy blue. In the foreground, several dark, gnarled, dead tree trunks emerge from the water. The reflection of the canoe and the people is visible in the water, but it is distorted and appears to be a different scene, possibly a person standing on a log or a different part of the canoe. The overall mood is mysterious and dreamlike.

Imagine, you  
develop some  
image processor  
tool like:  
[tiny.cc/m3ta5y](https://tiny.cc/m3ta5y)

# Example

---

# You have to handle tons of pixels

```
public override Color Calculate(IEnumerable<Color> colors)
{
    var a = colors.ElementAt(0);
    var b = colors.ElementAt(1);
    var aY = Math.Abs(128 - GetY(a));
    var bY = Math.Abs(128 - GetY(b));
    return aY > bY ? a : b;
}
```

```
static double GetY(Color c) => 0.299 * c.R + 0.587 * c.G + 0.114 * c.B;
```

# You have to handle tons of pixels

```
public override Color Calculate(IEnumerable<Color> colors)
{
    var a = colors.ElementAt(0);
    var b = colors.ElementAt(1);
    var aY = Math.Abs(128 - GetY(a));
    var bY = Math.Abs(128 - GetY(b));
    Calc...
    Calc...
    Calc...
    return aY > bY ? a : b;
}

static double GetY(Color c) => 0.299 * c.R + 0.587 * c.G + 0.114 * c.B;
```

# Demo

---

# **CORINFO\_FLG\_DONT\_INLINE**

The method should not be inlined

# Just In Time Hooking

---

By George Plotnikov

[accembler@gmail.com](mailto:accembler@gmail.com)

# Have you ever

try to...?

- Write inlined code
- Write tail-called code
- Predict compiler optimizations
- Anticipate how the new compiler version will make up the code
- Develop code based on low level optimizations
- Write unit tests for code not for the functionality



```

; Total bytes of code 23, prolog size 4 for method
jitdumpptest.Program:Main(ref)
; =====
; Assembly listing for method jitdumpptest.Program:Calc():ref
; Emitting BLENDED_CODE for X64 CPU with AVX
; optimized code
; rsp based frame
; partially interruptible
; Final local variable assignments
;
; V00 tmp0      [V00,T00] ( 4, 8 )    ref -> rsi
class-hnd exact
; V01 tmp1      [V01,T01] ( 4, 8 )    ref -> rdi
class-hnd exact
; V02 OutArgs   [V02  ] ( 1, 1 )    lclBlk (32)
[rsp+0x00]
;
; Lcl frame size = 40

```

# Long live the Open Source

As far as the source code became open, It means I can see all  
**external functions**  
and **interfaces**,  
which I can use how I want.

CoreCLR is the runtime for .NET Core. It includes the garbage collector, JIT compiler, primitive data types and low-level classes.

<https://docs.microsoft.com/dotnet/core/>

coreclr

corefx

c-sharp

dotnet

dotnet-core

dotnetcore

runtime

jit

📄 20,426 commits

🌿 18 branches

📦 77 releases

👤 551 contributors

📄 View license

Branch: master ▾

New pull request

Create new file

Upload files

Find File

Clone or download ▾



sandreenko Revert "Add call to setup-stress-dependencies.sh to build-test.sh. (#...

Latest commit 8343724 a day ago

Documentation

Reintroduce PR #22617 (Update added types and methoddefs on ApplyMeta...

5 days ago

cross

Fix tizen rootfs building (#22715)

a month ago

eng

Update Ubuntu 18.04 ARM64 queues (#23374)

3 days ago

scripts

SuperPMI Collect/Replay/AsmDiff tool (#21252)

2 months ago

src

Replace slow implementations in ASCIIUtility with fast implementations (

2 days ago

tests

Revert "Add call to setup-stress-dependencies.sh to build-test.sh. (#...

a day ago

.editorconfig

Invalid setting csharp\_new\_line\_within\_query\_expression\_clauses in .e...

4 months ago

.gitattributes

Add Word2Vec Benchmark Harness (#17350)

a year ago

.gitignore

Adding \*.nuget.dgspec.json to the gitignore file. (#23033)

18 days ago

.gitmirrorselective

Rename gitmirrorfile

2 years ago

BuildToolsVersion.txt

Update BuildTools, CoreClr, IbcData, PgoData to preview1-03723-01, pr...

25 days ago

CMakeLists.txt

Renaming RuntimeEventSource to NativeRuntimeEventSource (#22533)

a month ago

CODE\_OWNERS.TXT

Update CODE\_OWNERS.TXT (#17114)

a year ago

CONTRIBUTING.md

Create CONTRIBUTING.md (#6386)

3 years ago

Directory.Build.props

Set copyright and package license

a month ago

Directory.Build.targets

Add Azure DevOps signing support (#21545)

3 months ago

If get the  
information  
directly from the  
JIT compiler

Get the ILCode, and  
information used for  
optimization. Will my  
method be optimized or  
not, why?

---

**If I know the  
entrypoint and the  
interface...**

Does it mean I can  
hook the compile at  
the runtime?

# How to hook the external code?

Just the brief reminder

The method, suitable for hooking should:

- be marked as external, otherwise, I won't be able to see it from the assembly
- be virtual, so I could interact with its VTable in late binding
- arguments mostly should give me the information I'm looking for

- Find the method
- Wrap native functions with the C#
- Hello win32
- ???
- compileMethod parameters
- Dump Info

- **Find the method**
- Wrap native functions with the C#
- Hello win32
- ???
- compileMethod parameters
- Dump Info



# ICorJitCompiler

```
extern "C" ICorJitCompiler* __stdcall getJit();
```

```
// ICorJitCompiler is the interface that the EE uses to get IL bytecode converted to native code. Note that  
// to accomplish this the JIT has to call back to the EE to get symbolic information. The code:ICorJitInfo  
// type passed as 'comp' to compileMethod is the mechanism to get this information. This is often the more  
// interesting interface.
```

# compileMethod

```
virtual CorJitResult __stdcall compileMethod (  
    ICorJitInfo *comp,  
    struct CORINFO_METHOD_INFO *info,  
    unsigned flags,  
    BYTE **nativeEntry,  
    ULONG *nativeSizeOfCode) = 0;
```

- **Find the method**
- Wrap native functions with the C#
- Hello win32
- ???
- compileMethod parameters
- Dump Info

- Find the method
- **Wrap native functions with the C#**
- Hello win32
- ???
- compileMethod parameters
- Dump Info

```
extern "C" ICorJitCompiler* __stdcall getJit(); // function for
// import
[DllImport(
    #if _TARGET_X64_
        "Clrjit.dll"
    #else
        "Mscorjit.dll"
    #endif,
    CallingConvention = CallingConvention.StdCall, SetLastError =
true, EntryPoint = "getJit", BestFitMapping = true)]
public static extern IntPtr GetJit();
```

```
virtual CorJitResult __stdcall compileMethod (ICorJitInfo *comp,  
struct CORINFO_METHOD_INFO *info, unsigned flags, BYTE  
**nativeEntry, ULONG *nativeSizeOfCode) = 0;
```

```
#define CompileFunctionSig CorJitResult(*compileMethod)(  
    void*,  
    struct CORINFO_METHOD_INFO*,  
    unsigned flags,  
    BYTE**,  
    ULONG*)
```

```
[UnmanagedFunctionPointer(CallingConvention.StdCall, SetLastError  
= true)]
```

```
public unsafe delegate CorJitResult CompileMethodDel(  
    IntPtr thisPtr,  
    [In] IntPtr corJitInfo,  
    [In] CorInfo* methodInfo,  
    CorJitFlag flags,  
    [Out] IntPtr nativeEntry,  
    [Out] IntPtr nativeSizeOfCode);
```

- Find the method
- **Wrap native functions with the C#**
- Hello win32
- ???
- compileMethod parameters
- Dump Info



- Find the method
- Wrap native functions with the C#
- **Hello win32**
- ???
- compileMethod parameters
- Dump Info

```
[DllImport("Kernel32.dll", BestFitMapping = true,  
CallingConvention = CallingConvention.Winapi, SetLastError = true,  
ExactSpelling = true)]
```

```
public static extern bool VirtualProtect(  
    IntPtr lpAddress,  
    UInt32 dwSize,  
    MemoryProtectionConstants flNewProtect,  
    out UInt32 lpflOldProtect);
```

```
public enum MemoryProtectionConstants
{
    PAGE_EXECUTE = 0x10,
    PAGE_EXECUTE_READ = 0x20,
    PAGE_EXECUTE_READWRITE = 0x40,
    PAGE_EXECUTE_WRITECOPY = 0x80,
    PAGE_NOACCESS = 0x01,
    PAGE_READONLY = 0x02,
    PAGE_READWRITE = 0x04,
    PAGE_WRITECOPY = 0x08,
    PAGE_TARGETS_INVALID = 0x40000000,
    PAGE_TARGETS_NO_UPDATE = 0x40000000,
    PAGE_GUARD = 0x100,
    PAGE_NOCACHE = 0x200,
    PAGE_WRITECOMBINE = 0x400
}
```

- Find the method
- Wrap native functions with the C#
- **Hello win32**
- ???
- compileMethod parameters
- Dump Info

# Is everything ok?

flashback

## What have we done with the `compileMethod`

- Get the original function pointer
- Unlock memory
- Set the new function pointer
- Lock memory

- Find the method
- Wrap native functions with the C#
- Hello win32
- **SOF**
- compileMethod parameters
- Dump Info

# RuntimeHelpers

- [SecurityCriticalAttribute]  
`public static void PrepareDelegate(Delegate d)`
- [SecurityCriticalAttribute]  
`public static void PrepareMethod(RuntimeMethodHandle method,  
RuntimeTypeHandle[] instantiation)`

- Find the method
- Wrap native functions with the C#
- Hello win32
- **SOF**
- compileMethod parameters
- Dump Info



- Find the method
- Wrap native functions with the C#
- Hello win32
- SOF
- **compileMethod parameters**
- Dump Info

# ICorJitInfo

```
class ICorJitInfo : public ICorDynamicInfo
{
public:
    virtual IEEMemoryManager* getMemoryManager() = 0;

    virtual void getModuleNativeEntryPointRange(void ** pStart, void ** pEnd) = 0;

    virtual HRESULT getBBProfileData(CORINFO_METHOD_HANDLE ftnHnd, ULONG * count, ProfileBuffer
** profileBuffer, ULONG *
                                numRuns) = 0;

    virtual DWORD getExpectedTargetArchitecture() = 0;

    ...
}
```

```
[StructLayout(layoutKind: LayoutKind.Sequential, Pack = 1, Size = 0x88)]  
public unsafe struct CorInfo  
{  
    public IntPtr methodHandle;  
    public IntPtr moduleHandle;  
    public IntPtr ILCode;  
    public UInt32 ILCodeSize;  
    public UInt16 maxStack;  
    public UInt16 EHcount;  
    public CorInfoOptions options;  
    public CorInfoRegionKind regionKind;  
    public CorInfoSigInfo args;  
    public CorInfoSigInfo locals;  
}
```

- Find the method
- Wrap native functions with the C#
- Hello win32
- SOF
- **compileMethod parameters**
- Dump Info

- Find the method
- Wrap native functions with the C#
- Hello win32
- SOF
- compileMethod parameters
- **Dump Info**

# Method attribs

```
enum CorInfoFlag
{
    // CORINFO_FLG_UNUSED           = 0x00000001,
    // CORINFO_FLG_UNUSED           = 0x00000002,
    CORINFO_FLG_PROTECTED = 0x00000004,
    CORINFO_FLG_STATIC = 0x00000008,
    CORINFO_FLG_FINAL = 0x00000010,
    CORINFO_FLG_SYNCH = 0x00000020,
    CORINFO_FLG_VIRTUAL = 0x00000040,
    CORINFO_FLG_NATIVE = 0x00000100,
    CORINFO_FLG_INTRINSIC_TYPE = 0x00000200, // This type is marked by [Intrinsic]
    CORINFO_FLG_ABSTRACT = 0x00000400,
    ...
}
```

# OpCodes

IL_OPCODE(0x00,	"nop	", 0, ILDUMP_VOID, "", 0)
IL_OPCODE(0x01,	"break	", 0, ILDUMP_VOID, "", 0)
IL_OPCODE(0x02,	"ldarg.0	", 0, ILDUMP_VOID, "", 0)
IL_OPCODE(0x03,	"ldarg.1	", 0, ILDUMP_VOID, "", 0)
IL_OPCODE(0x04,	"ldarg.2	", 0, ILDUMP_VOID, "", 0)
IL_OPCODE(0x05,	"ldarg.3	", 0, ILDUMP_VOID, "", 0)
IL_OPCODE(0x06,	"ldloc.0	", 0, ILDUMP_VOID, "", 0)
IL_OPCODE(0x07,	"ldloc.1	", 0, ILDUMP_VOID, "", 0)
IL_OPCODE(0x08,	"ldloc.2	", 0, ILDUMP_VOID, "", 0)
IL_OPCODE(0x09,	"ldloc.3	", 0, ILDUMP_VOID, "", 0)

...

Proof!

# The result

```
[MethodImpl(MethodImplOptions.NoInlining)]  
public sealed override int  
Calc(int x, int y)  
{  
    var r = Math.Asin((double)x);  
    return (int)r * y * 1;  
}
```

- native size of code: 12
- IL code: 000001C923D7A5F0
- method attribs: 140000050



# IL code: 000001C923D7A5F0

```
IL_0000: nop
IL_0001: ldarg.1
IL_0002: conv.r8
IL_0003: call <0x0a00000e>
IL_0008: stloc.0
IL_0009: ldloc.0
IL_000a: conv.i4
IL_000b: ldarg.2
IL_000c: mul
IL_000d: stloc.1
IL_000e: br.s IL_0010
IL_0010: ldloc.1
IL_0011: ret
```

method attribs: 14000050

CORINFO\_FLG\_FINAL

CORINFO\_FLG\_VIRTUAL

CORINFO\_FLG\_NOSECURITYWRAP: The method requires no security checks

CORINFO\_FLG\_DONT\_INLINE: The method should not be inlined

If the compiler's  
source code is open  
- we can use it to  
improve our one

# Instead of conclusion

It's not the super easy code, but it works and the result is useful. Approach is viable and can help to reach your runtime analysis goals. Just recall your C++ knowledge 😁.

# Links

- <https://github.com/dotnet/coreclr>
- <https://github.com/dotnet/BenchmarkDotNet>
- <https://github.com/dotnet/coreclr/blob/master/Documentation/building/viewing-jit-dumps.md>
- <https://en.wikipedia.org/wiki/Hooking>
- <https://msdn.microsoft.com/ru-ru/library/windows/desktop/aa366898%28v=vs.85%29.aspx>
- <https://github.com/GeorgePlotnikov/ClrAnalyzer>

# Cheers!

Mail: [accembler@gmail.com](mailto:accembler@gmail.com)

GitHub/Twitter: @georgeplotnikov

Contribute:

[github.com/GeorgePlotnikov/ClrAnalyzer](https://github.com/GeorgePlotnikov/ClrAnalyzer)

Site: [georgeplotnikov.github.io](https://georgeplotnikov.github.io)

