What’s Inside a .NET Assembly?
Robert Pickering

@robertpi@functional.cafe

DATADOG

Why Talk Intermediate Language (IL)?

- Details of IL can be surprising
- .NET’s AOT compiles bring some interesting new opportunities …

Intermediate Language (IL)

- Sometime called MSIL or CIL

- Two forms:
  - Text - ildasm / ildasm
  - Binary

- A set of instructions like assembler, but easier

- All instruction operate on a stack

Hello World C#

```
namespace HelloWorld;
public static class Program
{
    static int Main(string[] args)
    {
        var message = "Hello,
World!";
        var result = 0;

        Console.WriteLine(message);

        return result;
    }

}
```

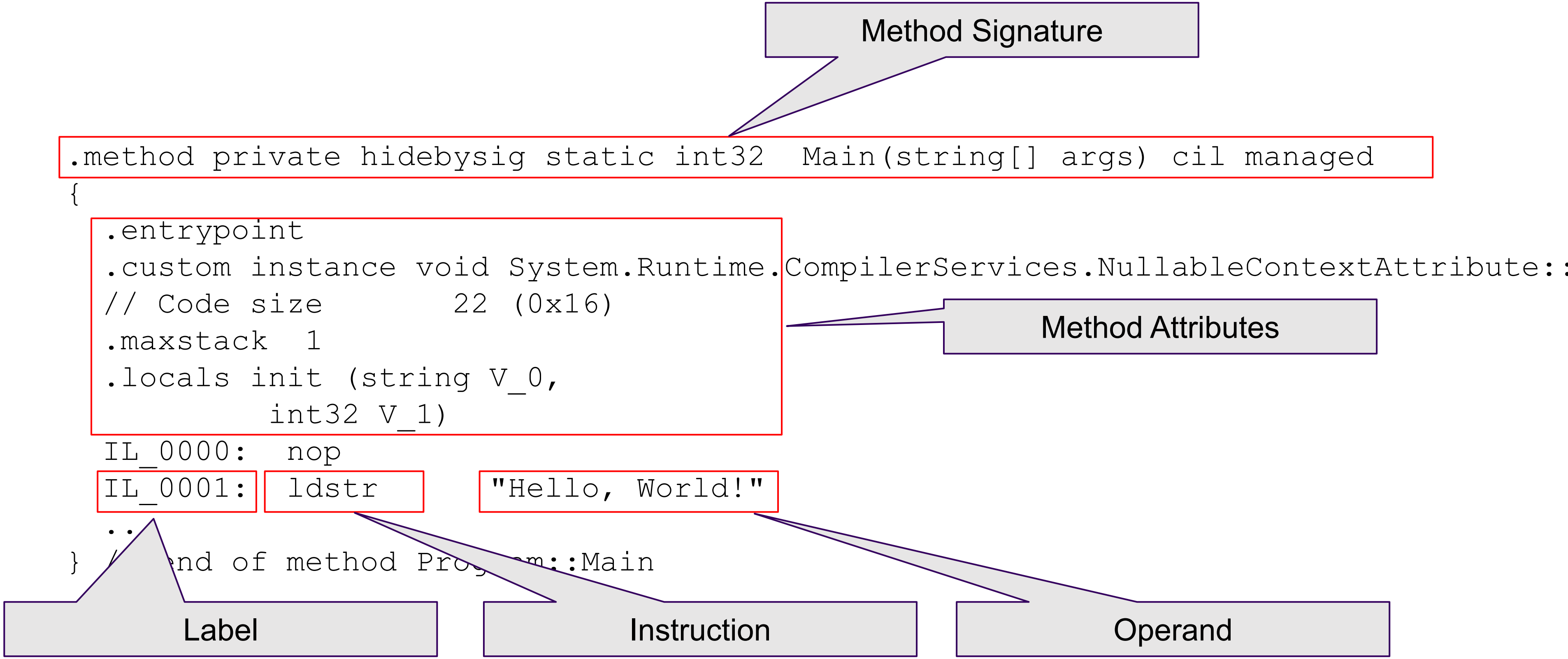
https://sharplab.io/

Hello World IL

```
.method private hidebysig static int32  M
{
  .entrypoint
  .custom instance void System.Runtime.Co
  // Code size       22 (0x16)
  .maxstack  1
  .locals init (string V_0,
           int32 V_1)
  IL_0000:  nop
  IL_0001:  ldstr      "Hello, World!"
  IL_0006:  stloc.0
  IL_0007:  ldc.i4.0
  IL_0008:  stloc.1
  IL_0009:  ldloc.0
  IL_000a:  call       void [System.Conso
  IL_000f:  nop
  IL_0010:  ldloc.1
  IL_0011:  ret
} // end of method Program::Main
```

DATADOG

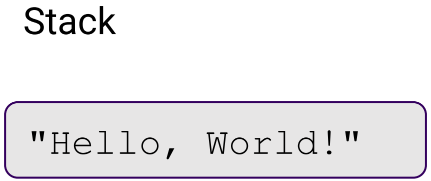
Hello World IL



```
.method private hidebysig static int32  Main(string[] args) cil managed
{
  .entrypoint
  .custom instance void System.Runtime.CompilerServices.NullableContextAttribute::
  // Code size       22 (0x16)
  .maxstack  1
  .locals init (string V_0,
           int32 V_1)
  IL_0000:  nop
  IL_0001:  ldstr      "Hello, World!"
  ...
} // end of method Program::Main
```

Hello World IL

```
.locals init (string V_0,
         int32 V_1)
IL_0000:  nop
IL_0001:  ldstr      "Hello, World!"
IL_0006:  stloc.0
IL_0007:  ldc.i4.0
IL_0008:  stloc.1
IL_0009:  ldloc.0
IL_000a:  call       void [System.Console]System.Console::WriteLine(string)
IL_000f:  nop
IL_0010:  ldloc.1
IL_0011:  ret
```



Stack

"Hello, World!"

Hello World IL

Stack

```
.locals init (string V_0,
         int32 V_1)
IL_0000:  nop
IL_0001:  ldstr      "Hello, World!"
IL_0006:  stloc.0
IL_0007:  ldc.i4.0
IL_0008:  stloc.1
IL_0009:  ldloc.0
IL_000a:  call       void [System.Console]System.Console::WriteLine(string)
IL_000f:  nop
IL_0010:  ldloc.1
IL_0011:  ret
```

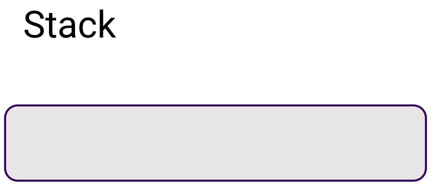
Hello World IL

```
.locals init (string V_0,
         int32 V_1)
IL_0000:  nop
IL_0001:  ldstr      "Hello, World!"
IL_0006:  stloc.0
IL_0007:  ldc.i4.0
IL_0008:  stloc.1
IL_0009:  ldloc.0
IL_000a:  call       void [System.Console]System.Console::WriteLine(string)
IL_000f:  nop
IL_0010:  ldloc.1
IL_0011:  ret
```

Stack

0

Hello World IL



```
.locals init (string V_0,
         int32 V_1)
IL_0000:  nop
IL_0001:  ldstr      "Hello, World!"
IL_0006:  stloc.0
IL_0007:  ldc.i4.0
IL_0008:  stloc.1
IL_0009:  ldloc.0
IL_000a:  call       void [System.Console]System.Console::WriteLine(string)
IL_000f:  nop
IL_0010:  ldloc.1
IL_0011:  ret
```

Hello World IL

```
.locals init (string V_0,
         int32 V_1)
IL_0000:  nop
IL_0001:  ldstr      "Hello, World!"
IL_0006:  stloc.0
IL_0007:  ldc.i4.0
IL_0008:  stloc.1
IL_0009:  ldloc.0
IL_000a:  call       void [System.Console]System.Console::WriteLine(string)
IL_000f:  nop
IL_0010:  ldloc.1
IL_0011:  ret
```

Stack

"Hello, World!"

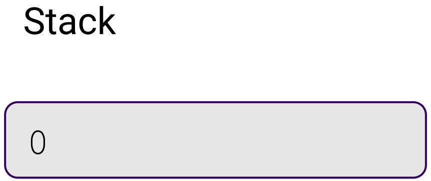
Hello World IL

Stack

```
.locals init (string V_0,
         int32 V_1)
IL_0000:  nop
IL_0001:  ldstr      "Hello, World!"
IL_0006:  stloc.0
IL_0007:  ldc.i4.0
IL_0008:  stloc.1
IL_0009:  ldloc.0
IL_000a:  call       void [System.Console]System.Console::WriteLine(string)
IL_000f:  nop
IL_0010:  ldloc.1
IL_0011:  ret
```

Hello World IL

```
.locals init (string V_0,
         int32 V_1)
IL_0000:  nop
IL_0001:  ldstr      "Hello, World!"
IL_0006:  stloc.0
IL_0007:  ldc.i4.0
IL_0008:  stloc.1
IL_0009:  ldloc.0
IL_000a:  call       void [System.Console]System.Console::WriteLine(string)
IL_000f:  nop
IL_0010:  ldloc.1
IL_0011:  ret
```



Hello World IL

Stack

```
.locals init (string V_0,
         int32 V_1)
IL_0000:  nop
IL_0001:  ldstr      "Hello, World!"
IL_0006:  stloc.0
IL_0007:  ldc.i4.0
IL_0008:  stloc.1
IL_0009:  ldloc.0
IL_000a:  call       void [System.Console]System.Console::WriteLine(string)
IL_000f:  nop
IL_0010:  ldloc.1
IL_0011:  ret
```

IL Labels

```
.locals init (string V_0,
         int32 V_1)
IL_0000:  nop
IL_0001:  ldstr      "Hello, World!"
IL_0006:  stloc.0
IL_0007:  ldc.i4.0
IL_0008:  stloc.1
IL_0009:  ldloc.0
IL_000a:  call       void [System.Console]System.Console::WriteLine(string)
IL_000f:  nop
IL_0010:  ldloc.1
IL_0011:  ret
```

Labels are free text

The number is the byte offset

We know that the `ldstr` instruction is 5 bytes long

Simple Branching C#

```
static int counter = 0;

static void Branching()
{
    if (counter > 0)
    {
        Console.WriteLine("Pos");
    }
}
```

Simple Branching IL

```
.method private hidebysig static void  Br
{
  // Code size       27 (0x1b)
  .maxstack  2
  .locals init (bool V_0)
  IL_0000:  nop
  IL_0001:  ldsfld     int32 HelloWorld.P
  IL_0006:  ldc.i4.0
  IL_0007:  cgt
  IL_0009:  stloc.0
  IL_000a:  ldloc.0
  IL_000b:  brfalse.s  IL_001a
  IL_000d:  nop
  IL_000e:  ldstr      "Pos"
  IL_0013:  call       void [System.Conso
  IL_0018:  nop
  IL_0019:  nop
  IL_001a:  ret
} // end of method Program::Branching
```

Simple Branching IL

```
.method private hidebysig static void  Branching() cil managed
{
  // Code size       27 (0x1b)
  .maxstack  2
  .locals init (bool V_0)
  rod:  nop
  jane:  ldsfld     int32 HelloWorld.Program::counter
  freddy:  ldc.i4.0
  geoffrey:  cgt
  george:  stloc.0
  bungle:  ldloc.0
  rodge:  brfalse.s  zippy
  matt:  nop
  julian:  ldstr      "Pos"
  karl:  call       void [System.Console]System.Console::WriteLine(string)
  dawn:  nop
  john:  nop
  zippy:  ret
} // end of method Program::Branching
```

Why Binary Formats are Hard

```
e2 80 99 54 77 61 73 20 62 72 69 6c 6c 69 67 2c 20
61 6e 64 20 74 68 65 20 73 6c 69 74 68 79 20 74 6f
76 65 73 0a 20 20 20 20 20 20 44 69 64 20 67 79 72
65 20 61 6e 64 20 67 69 6d 62 6c 65 20 69 6e 20 74
68 65 20 77 61 62 65 3a 0a 41 6c 6c 20 6d 69 6d 73
79 20 77 65 72 65 20 74 68 65 20 62 6f 72 6f 67 6f
76 65 73 2c 0a 20 20 20 20 20 20 41 6e 64 20 74 68
65 20 6d 6f 6d 65 20 72 61 74 68 73 20 6f 75 74 67
72 61 62 65 2e
```

Why Binary Formats are Hard

’Twas brillig, and the slithy toves
    Did gyre and gimble in the wabe:
All mimsy were the borogoves,
    And the mome raths outgrabe.

DATADOG

Hello World IL - with Bytes

Instruction Opcodes

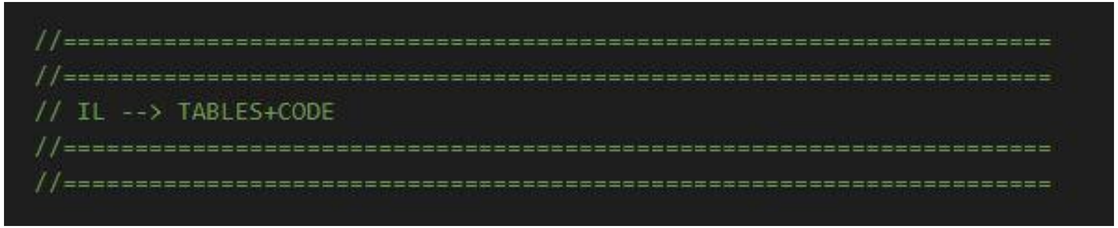
Operands

```
.method private hidebysig static int32  Main(string[] args) cil managed
// SIG: 00 01 08 1D 0E
{
  .custom ins    ce void Sys    Runtime.CompilerServices.NullableContextAttribute::.c
  // Method be   ns at RVA 0    94
  // Code size        22 (0x  )
  .maxstack  1
  .locals init (string V_0, int32 V_1)
  IL_0000:  /* 00   |                    */ nop
  IL_0001:  /* 72   | (70)000001         */ ldstr      "Hello, World!"
  IL_0006:  /* 0A   |                    */ stloc.0
  IL_0007:  /* 16   |                    */ ldc.i4.0
  IL_0008:  /* 0B   |                    */ stloc.1
  IL_0009:  /* 06   |                    */ ldloc.0
  IL_000a:  /* 28   | (0A)00000E         */ call       void [System.Console]System.Con
  IL_000f:  /* 00   |                    */ nop
  IL_0010:  /* 07   |                    */ ldloc.1
  IL_0015:  /* 2A   |                    */ ret
} // end of method Program::Main
```

Hello World IL - with Bytes

```
.method private hidebysig static int32  Main(string[] args) cil managed
// SIG: 00 01 08 1D 0E
{
  .entrypoint
  .custom instance void System.Runtime.CompilerServices.NullableContextAttribute::.c
  // Method begins at RVA 0x2094
  // Code size       22 (0x16)
  .maxstack  1
  .locals init (string V_0, int32 V_1)
  IL_0000:  /* 00   |                  */ nop
  IL_0001:  /* 72   | (70)000001       */ ldstr
  IL_0006:  /* 0A   |                  */ stloc.0
  IL_0007:  /* 16   |                  */ ldc.i4.0
  IL_0008:  /* 0B   |                  */ stloc.1
  IL_0009:  /* 06   |                  */ ldloc.0
  IL_000a:  /* 28   | (0A)00000E       */ call                        sole]System.Con
  IL_000f:  /* 00   |                  */ nop
  IL_0010:  /* 07   |                  */ ldloc.1
  IL_0015:  /* 2A   |                  */ ret
} // end of method Program::Main
```

From The F# Codebase

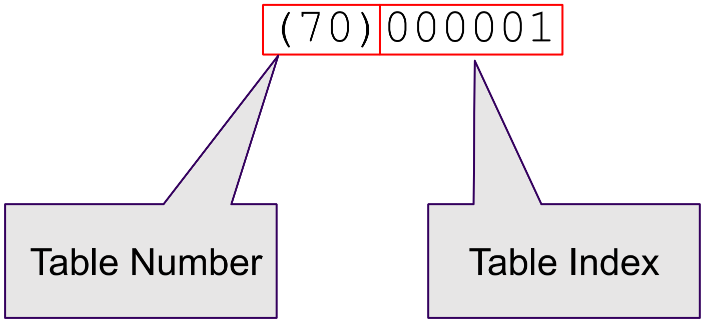


```
//==============================================================================
//==============================================================================
// IL --> TABLES+CODE
//==============================================================================
//==============================================================================
```

Table Tokens

- 4 Bytes
- UInt32 (little endian)
- Last byte is table number
- First 3 bytes are table entry index

Example:

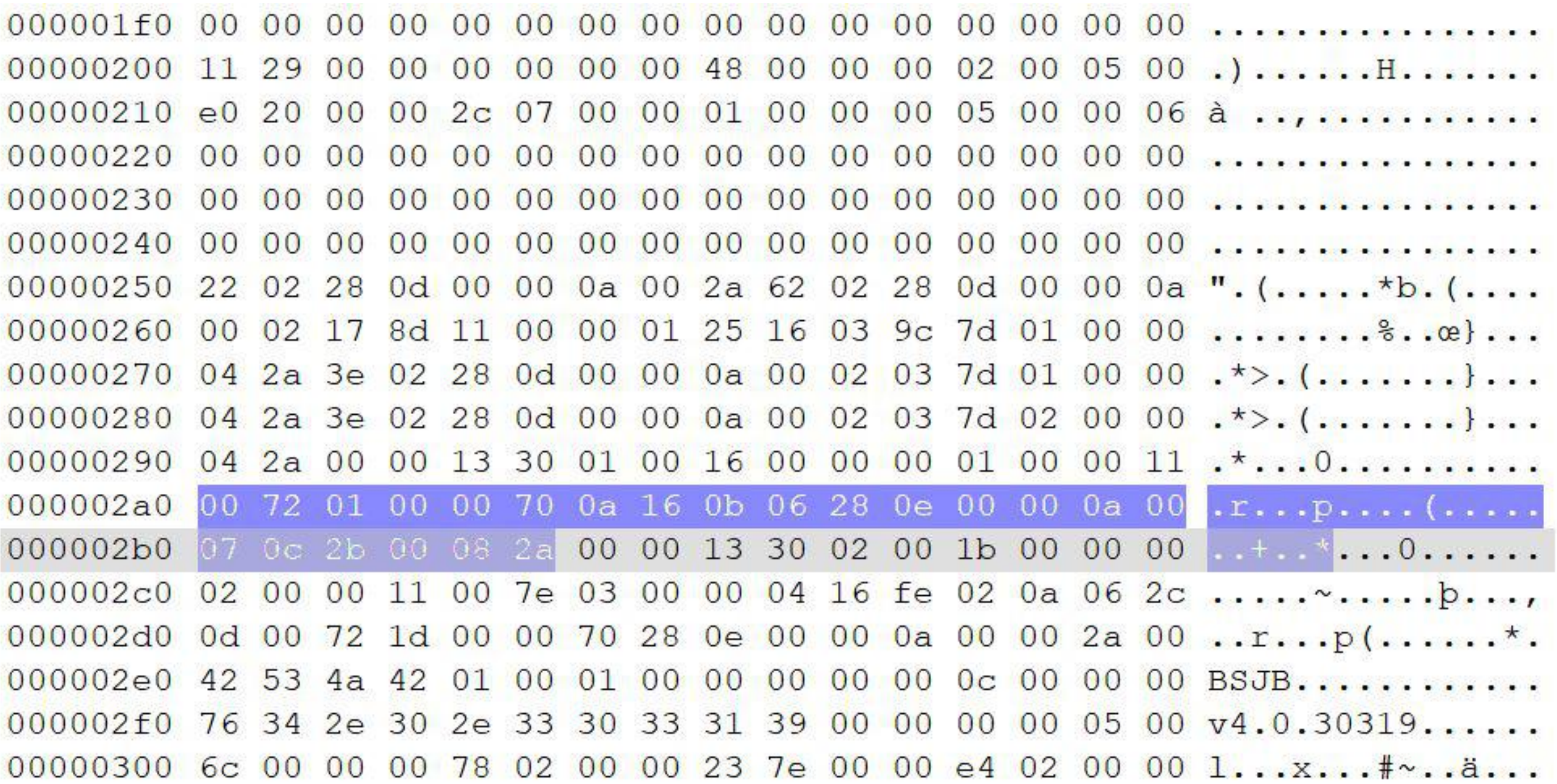


(70)000001

Table Number

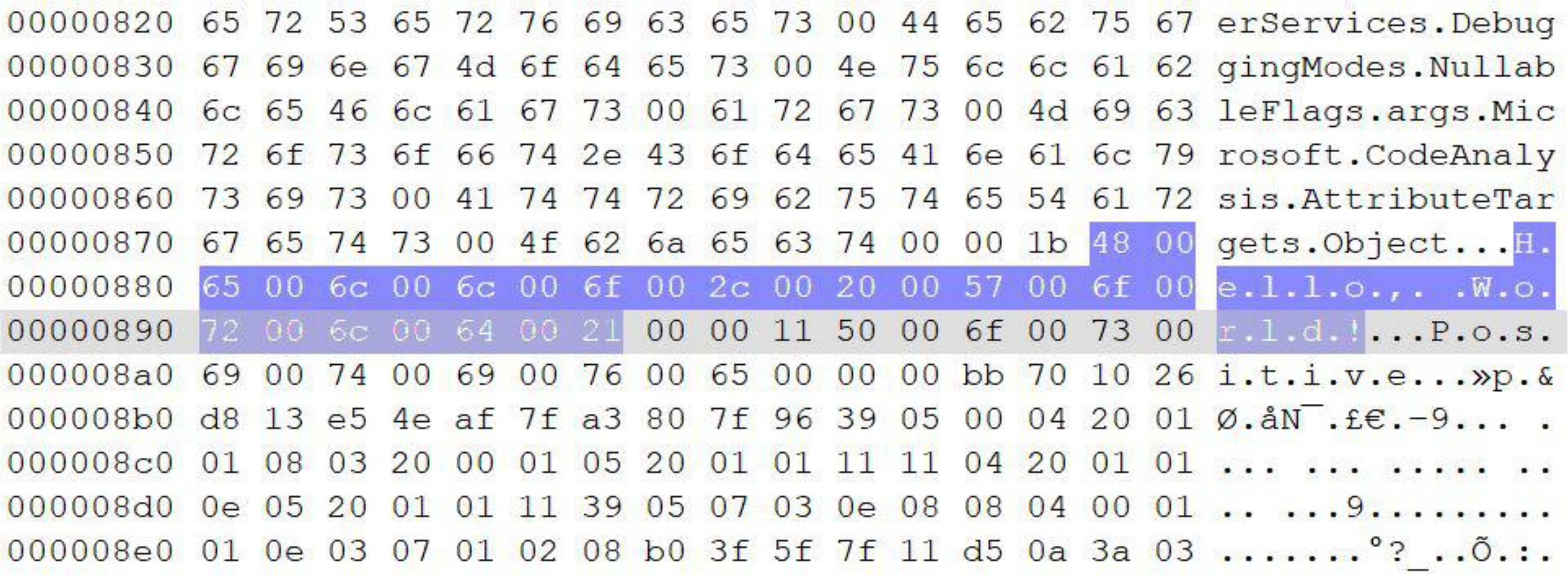
Table Index

Hello World Method



```
000001f0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000200 11 29 00 00 00 00 00 00 48 00 00 00 02 00 05 00 .)......H.......
00000210 e0 20 00 00 2c 07 00 00 01 00 00 00 05 00 00 06 à ..,...........
00000220 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000230 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000240 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000250 22 02 28 0d 00 00 0a 00 2a 62 02 28 0d 00 00 0a ".(.....*b.(....
00000260 00 02 17 8d 11 00 00 01 25 16 03 9c 7d 01 00 00 ........%..œ}...
00000270 04 2a 3e 02 28 0d 00 00 0a 00 02 03 7d 01 00 00 .*>.(.......}...
00000280 04 2a 3e 02 28 0d 00 00 0a 00 02 03 7d 02 00 00 .*>.(.......}...
00000290 04 2a 00 00 13 30 01 00 16 00 00 00 01 00 00 11 .*...0..........
000002a0 00 72 01 00 00 70 0a 16 0b 06 28 0e 00 00 0a 00 .r...p....(.....
000002b0 07 0c 2b 00 08 2a 00 00 13 30 02 00 1b 00 00 00 ..+..*...0......
000002c0 02 00 00 11 00 7e 03 00 00 04 16 fe 02 0a 06 2c .....~.....þ...,
000002d0 0d 00 72 1d 00 00 70 28 0e 00 00 0a 00 00 2a 00 ..r...p(......*.
000002e0 42 53 4a 42 01 00 01 00 00 00 00 00 0c 00 00 00 BSJB............
000002f0 76 34 2e 30 2e 33 30 33 31 39 00 00 00 00 05 00 v4.0.30319......
00000300 6c 00 00 00 78 02 00 00 23 7e 00 00 e4 02 00 00 l...x...#~..ä...
```

String Table



```
00000820 65 72 53 65 72 76 69 63 65 73 00 44 65 62 75 67 erServices.Debug
00000830 67 69 6e 67 4d 6f 64 65 73 00 4e 75 6c 6c 61 62 gingModes.Nullab
00000840 6c 65 46 6c 61 67 73 00 61 72 67 73 00 4d 69 63 leFlags.args.Mic
00000850 72 6f 73 6f 66 74 2e 43 6f 64 65 41 6e 61 6c 79 rosoft.CodeAnaly
00000860 73 69 73 00 41 74 74 72 69 62 75 74 65 54 61 72 sis.AttributeTar
00000870 67 65 74 73 00 4f 62 6a 65 63 74 00 00 1b 48 00 gets.Object...H.
00000880 65 00 6c 00 6c 00 6f 00 2c 00 20 00 57 00 6f 00 e.l.l.o.,. .W.o.
00000890 72 00 6c 00 64 00 21 00 00 11 50 00 6f 00 73 00 r.l.d.!...P.o.s.
000008a0 69 00 74 00 69 00 76 00 65 00 00 00 bb 70 10 26 i.t.i.v.e...»p.&
000008b0 d8 13 e5 4e af 7f a3 80 7f 96 39 05 00 04 20 01 Ø.åN¯.£€.–9... .
000008c0 01 08 03 20 00 01 05 20 01 01 11 11 04 20 01 01 ... ... ..... ..
000008d0 0e 05 20 01 01 11 39 05 07 03 0e 08 08 04 00 01 .. ...9.........
000008e0 01 0e 03 07 01 02 08 b0 3f 5f 7f 11 d5 0a 3a 03 .......°?_..Õ.:.
```

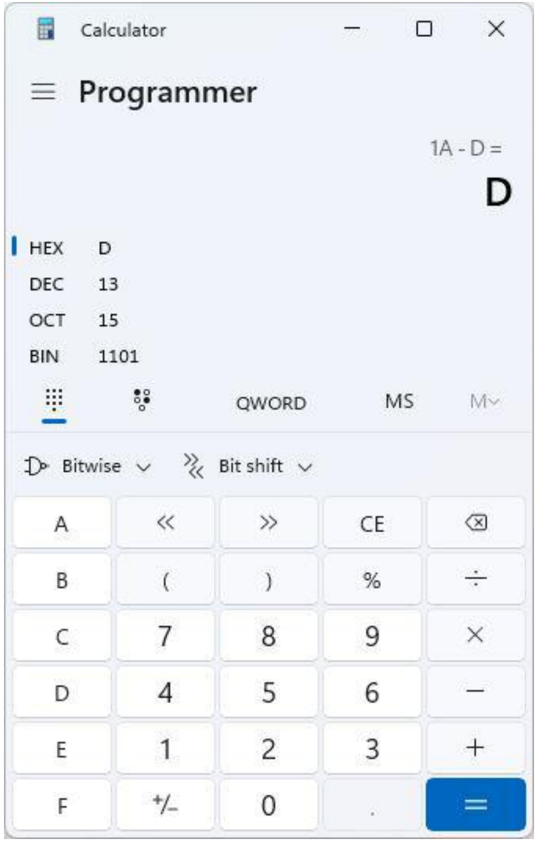
Simple Branching IL - with Bytes

```
.maxstack  2
.locals init (bool V_0)
IL_0000:  /* 00   |                  */ nop
IL_0001:  /* 7E   | (04)000003       */ ldsfld     int32 HelloWorld.Program::count
IL_0006:  /* 16   |                  */ ldc.i4.0
IL_0007:  /* FE02 |                  */ cgt
IL_0009:  /* 0A   |                  */ stloc.0
IL_000a:  /* 06   |                  */ ldloc.0
IL_000b:  /* 2C   | 0D               */ brfalse.s  IL_001a
IL_000d:  /* 00   |                  */ nop
IL_000e:  /* 72   | (70)00001D       */ ldstr      "Positive"
IL_0013:  /* 28   | (0A)00000E       */ call       void [System.Console]System.Con
IL_0018:  /* 00   |                  */ nop
IL_0019:  /* 00   |                  */ nop
IL_001a:  /* 2A   |                  */ ret
```

Offsets

The target instruction is represented as a 1-byte signed offset from the beginning of the instruction following the current instruction.

Next instruction start:	0D
Target instruction:	1A
Offset:	0D



Calculator

Programmer

1A - D =

D

HEX D

DEC 13

OCT 15

BIN 1101

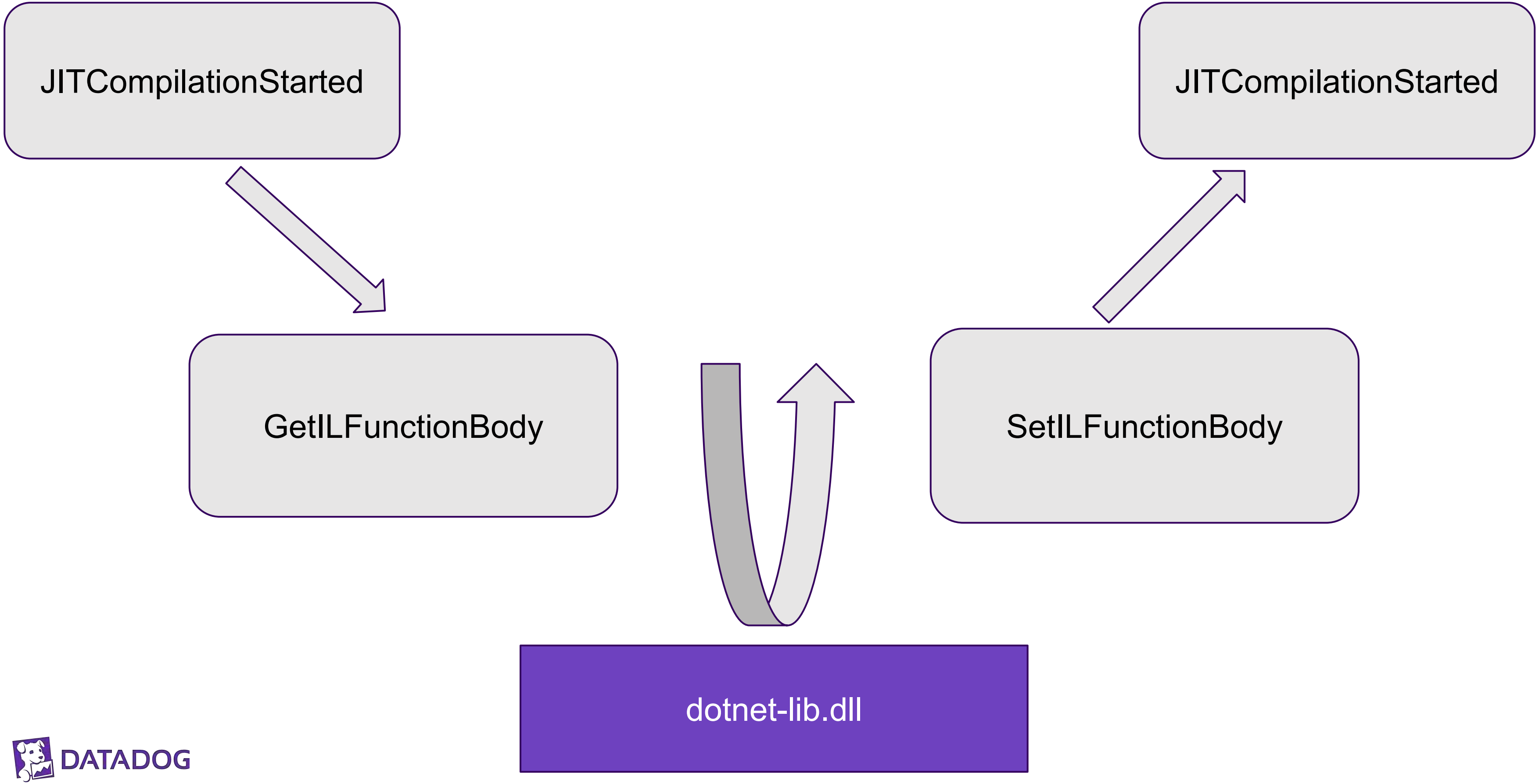
QWORD

IL Rewriting with the .NET Profiling API

Using C# / F# and AOT

DATADOG

IL Rewriting Overview



JITCompilationStarted

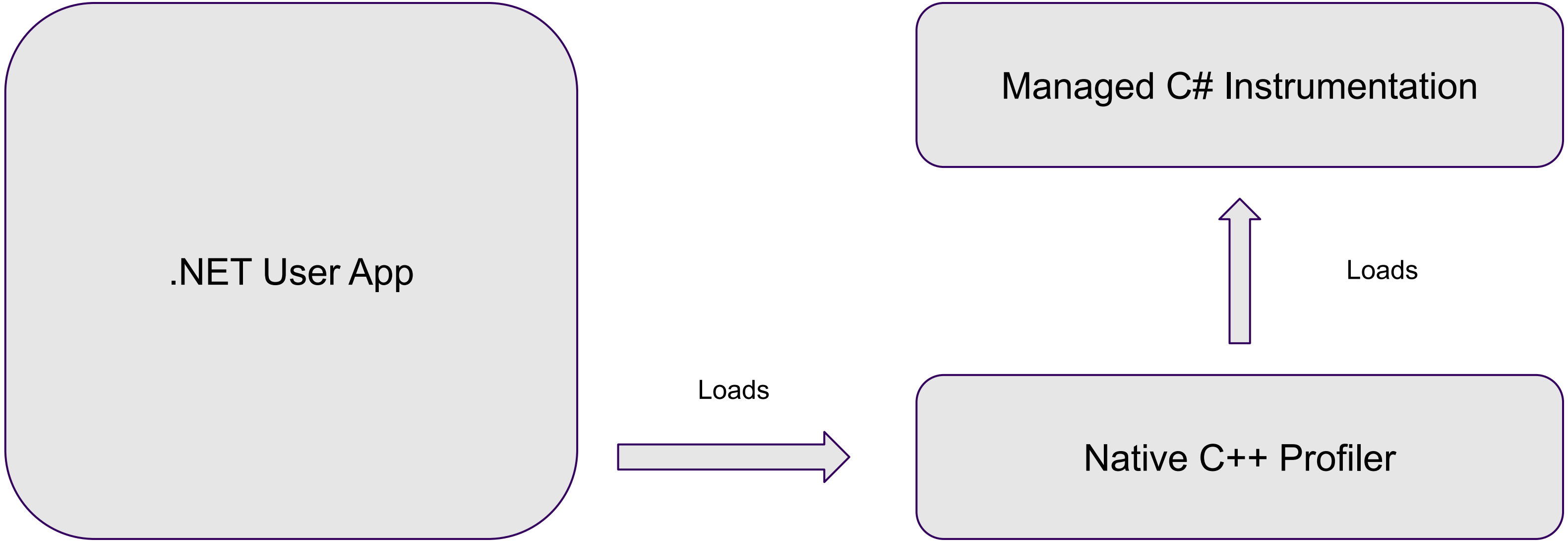
GetILFunctionBody

dotnet-lib.dll

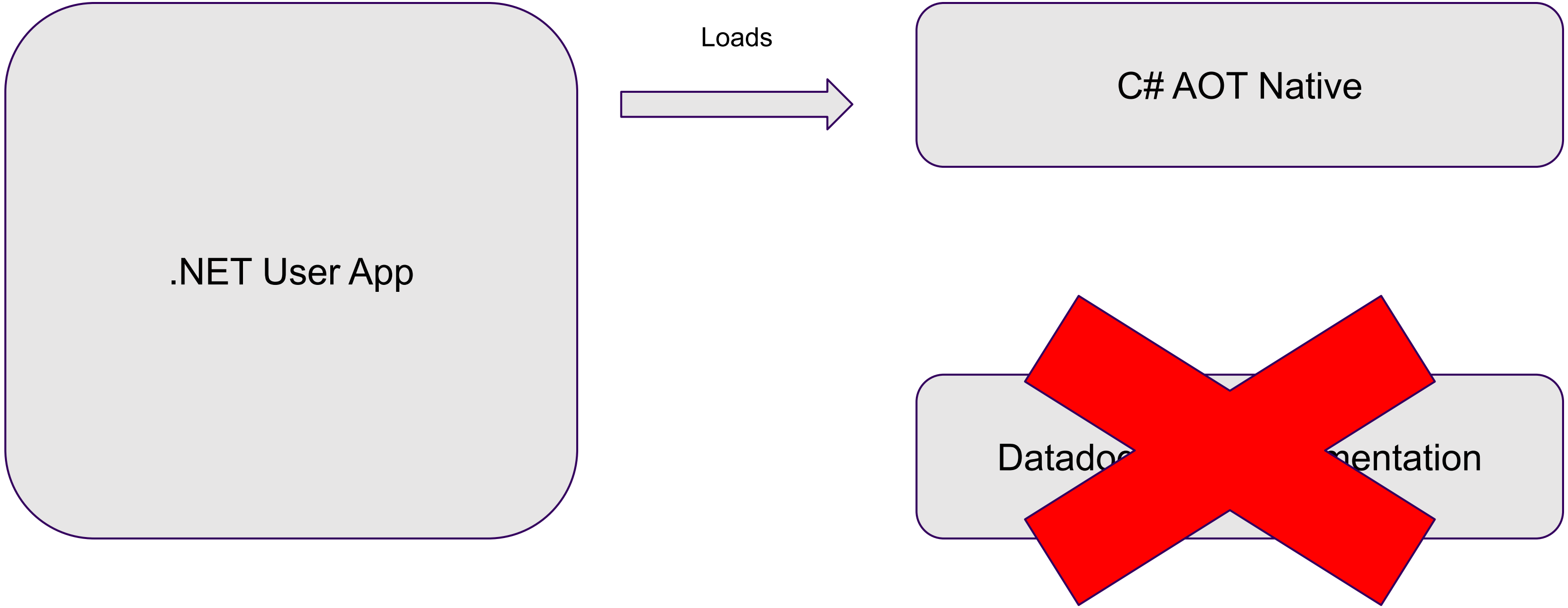
SetILFunctionBody

JITCompilationStarted

The .NET Profiling API



The .NET Profiling API



Kevin Gosse - Pushing C# to new places with NativeAOT (Dotnetos Conference 2022)

Diamond sponsors: allegro pay. Demant

Pushing C# to new places with NativeAOT

Kevin Gosse – Dotnetos Conference 2022
Twitter: @kookiz

DATADOG

https://www.youtube.com/watch?v=eE0AbO5_XSw

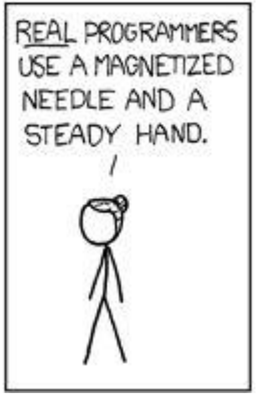
Instrumenting .NET - IL Rewriting

```
public HResult JITCompilationStarted(FunctionId fId,                 ck)
{
    ...

    GetILFunctionBody(ModuleMetadata.ModuleId,
        MethodDef,
        out byte* body,
        out uint methodSize);

    // body = new [] { 72, 01, 00, 00, 70, 28, 0A, 00, 00, 0C, 00, 2A, };
    ...
}
```

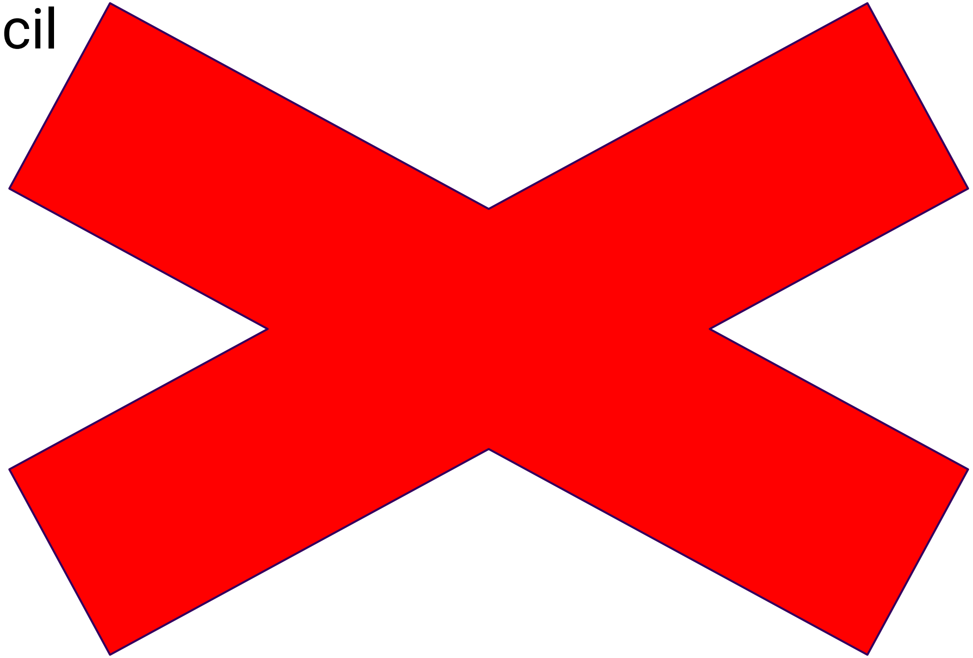
Real Programmers



https://xkcd.com/378/

IL Parsing Libraries

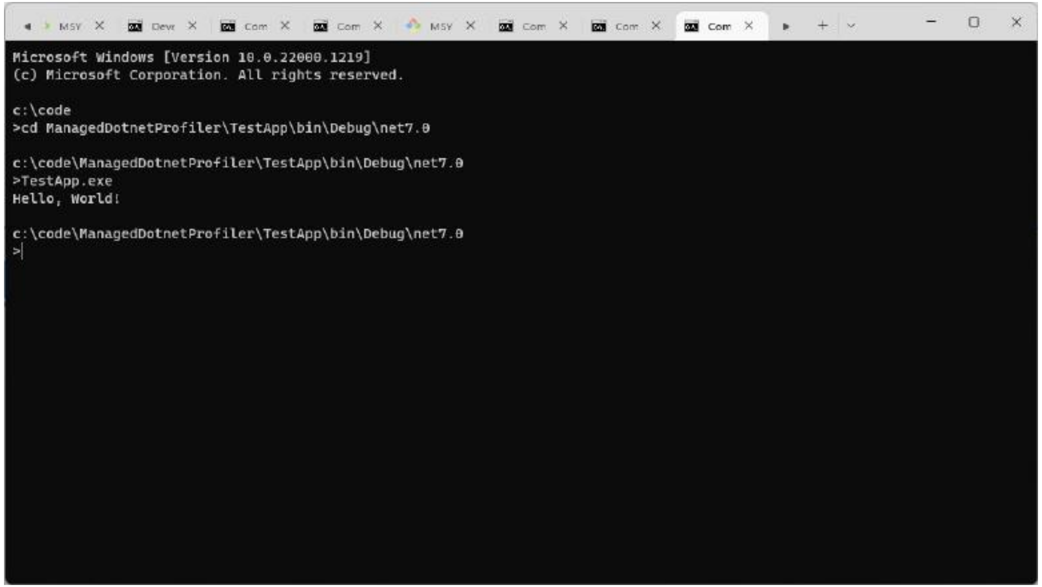
- Mono.Cecil
- dnlib
- ILSpy



F# To The Rescue!

```
type ILInstr =
    | I_call of ILTailcall * ILMethodSpec * ILVarArgs
    | I_callvirt of ILTailcall * ILMethodSpec * ILVarArgs
    | I_calli of ILTailcall * ILCallingSignature * ILVarArgs
    | I_ldftn of ILMethodSpec
    | I_newobj of ILMethodSpec * ILVarArgs
```

Demo! https://github.com/robertpi/ManagedDotnetProfiler



```
Microsoft Windows [Version 10.0.22000.1219]
(c) Microsoft Corporation. All rights reserved.

c:\code
>cd ManagedDotnetProfiler\TestApp\bin\Debug\net7.0

c:\code\ManagedDotnetProfiler\TestApp\bin\Debug\net7.0
>TestApp.exe
Hello, World!

c:\code\ManagedDotnetProfiler\TestApp\bin\Debug\net7.0
>
```

Questions?