

Compilers are Surprising.

A heartwarming story about how **Roslyn** and the **JIT compiler** compile your code and how this can affect performance in unexpected ways.

iyuno • SDI
GROUP

About Me.

- **Research and Innovation Director**
- **Big Data + Machine Learning**



badamczewski01



leveluppp



leveluppp.ghost.io

Agenda.

00 Why this lecture exists?

01 What are compilers?

02 What is a JIT compiler?

03 Decompilation Tools.

04 Optimizations and Surprises in C# and JIT.

05 Questions?

Why this lecture exists?

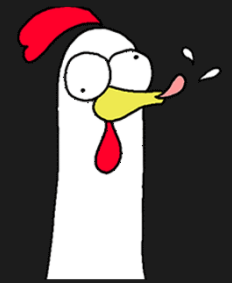
„In a real problem space the compiler can solve about 1 to 10% of your problem space”

- Mike Acton

Why this lecture exists?

Thesis

Compilers are Dumb



Why this lecture exists?

Problem map

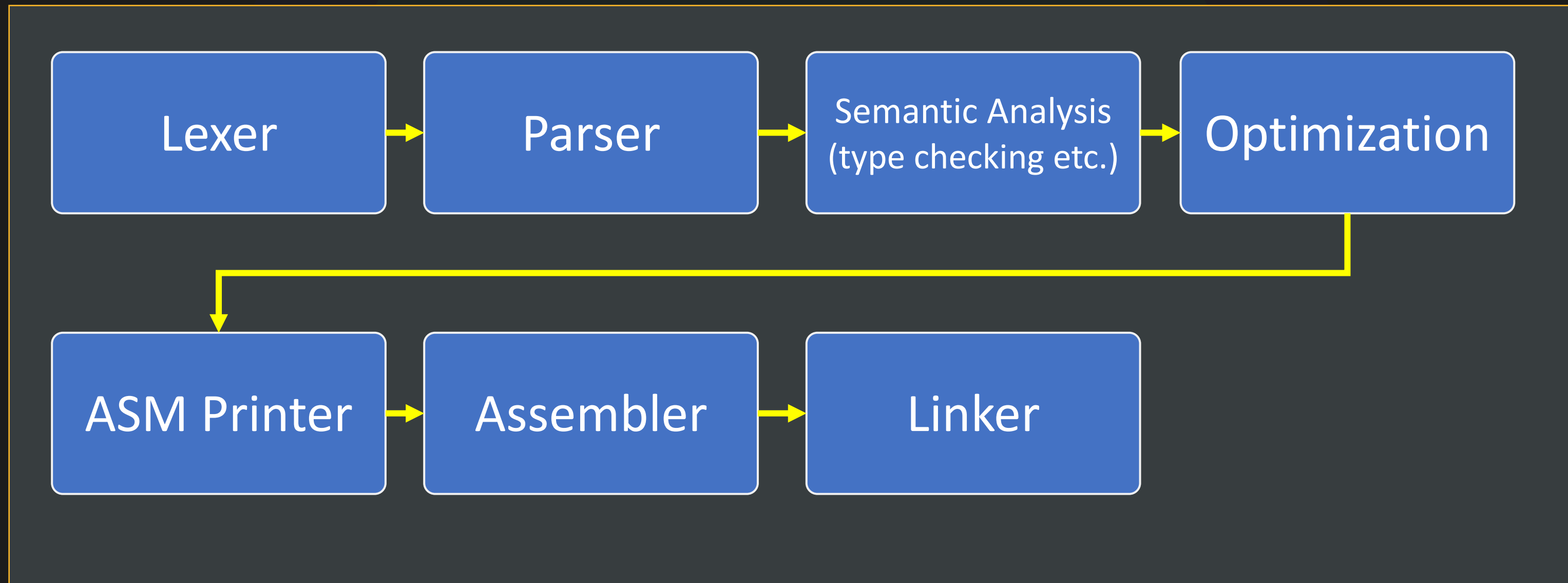
<p>Simple Algorithm implemented in a simple way</p>	<p>Complex Algorithm implemented in a simple way</p>
<p>Simple Algorithm implemented in a complex way</p>	<p>Complex Algorithm implemented in a complex way</p>

What are compilers?

Computer program that translates computer code written in one programming language into another language.

What are compilers?

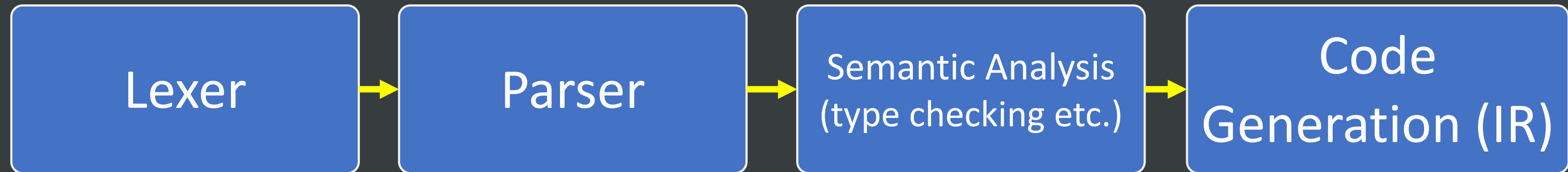
In the old days



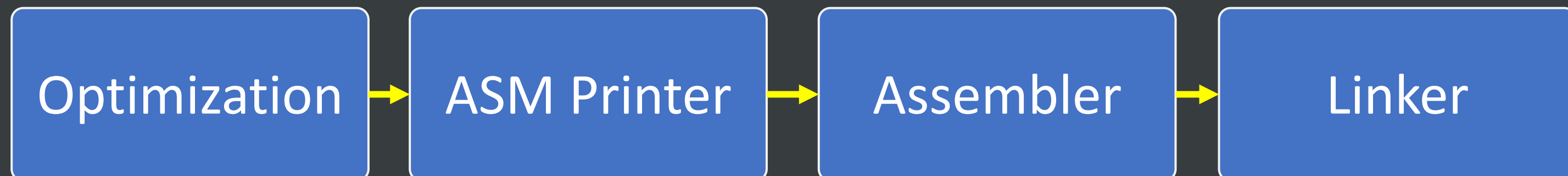
What are compilers?

Now

Front-End Compiler



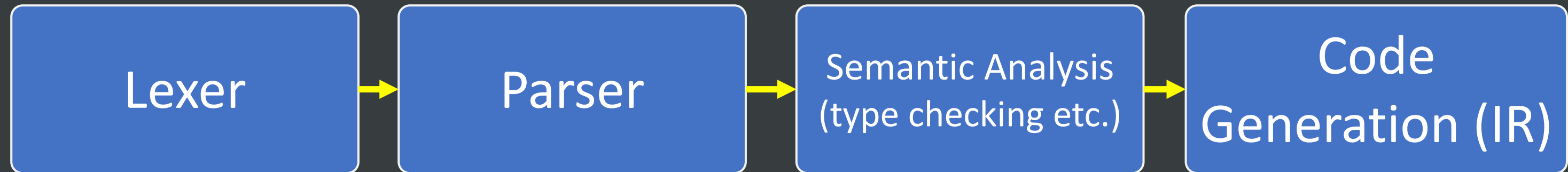
Back-End Compiler



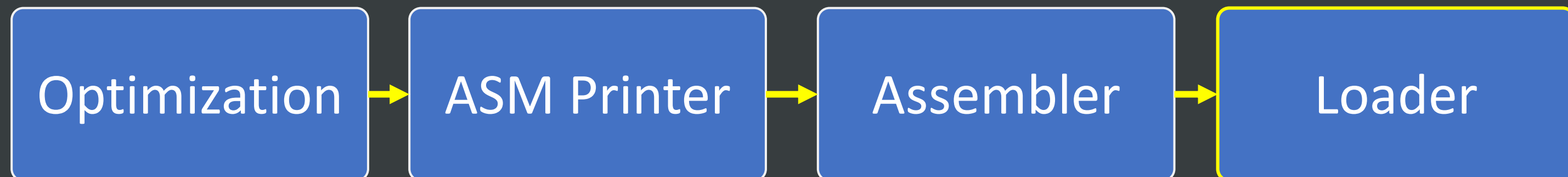
What are compilers?

JIT

Front-End Compiler

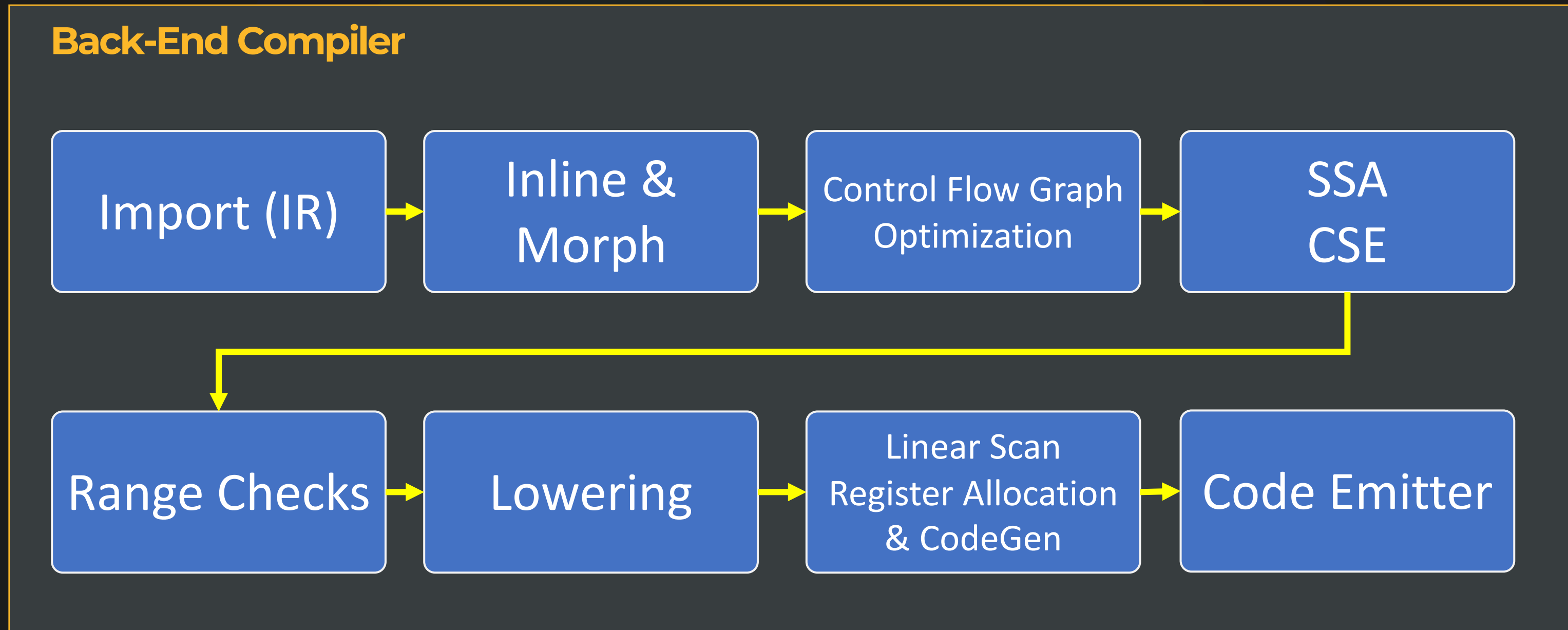


Back-End Compiler



What are compilers?

.NET JIT



Tools?

SharpLab

```
using System;
public class C {
    public static void M() {
        Console.WriteLine("We shall be using SharpLab");
    }
}
```

```
; Core CLR v5.0.421.11614 on amd64
```

```
C..ctor()
```

```
L0000: ret
```

```
C.M()
```

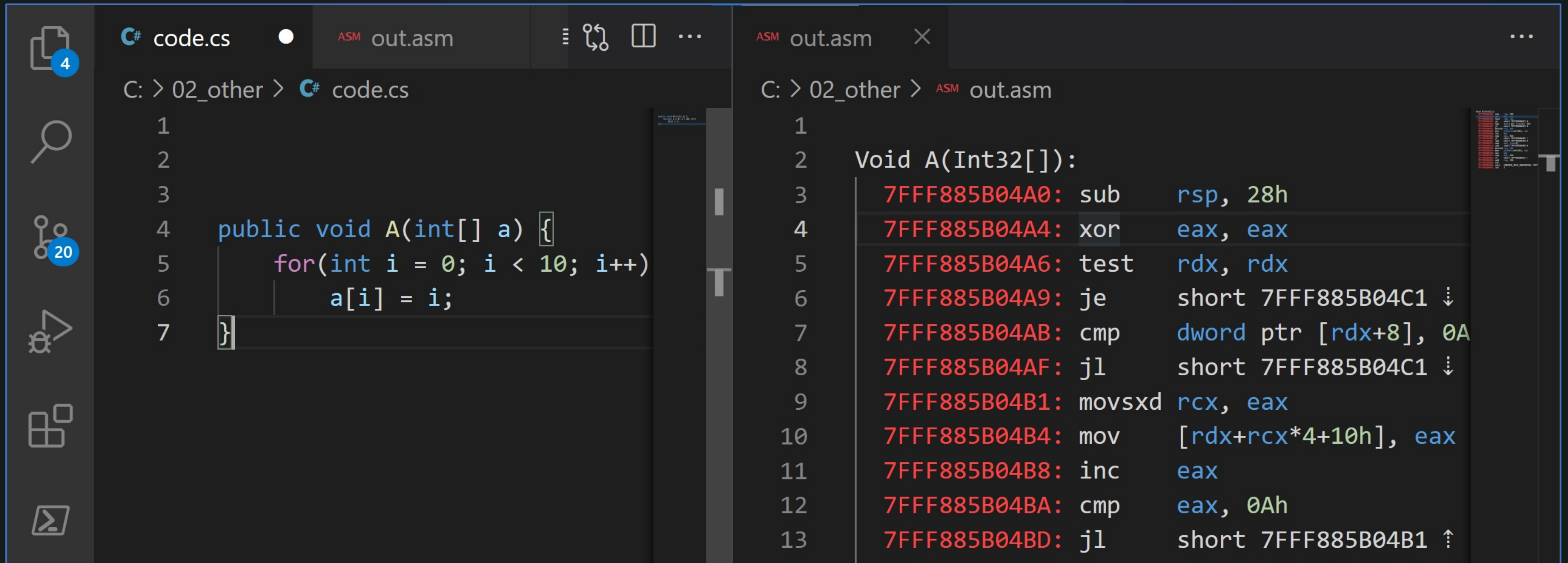
```
L0000: mov rcx, 0x2374005d460
```

```
L000a: mov rcx, [rcx]
```

```
L000d: jmp System.Console.WriteLine(System.String)
```

Tools?

PowerUP – Custom Decompiler



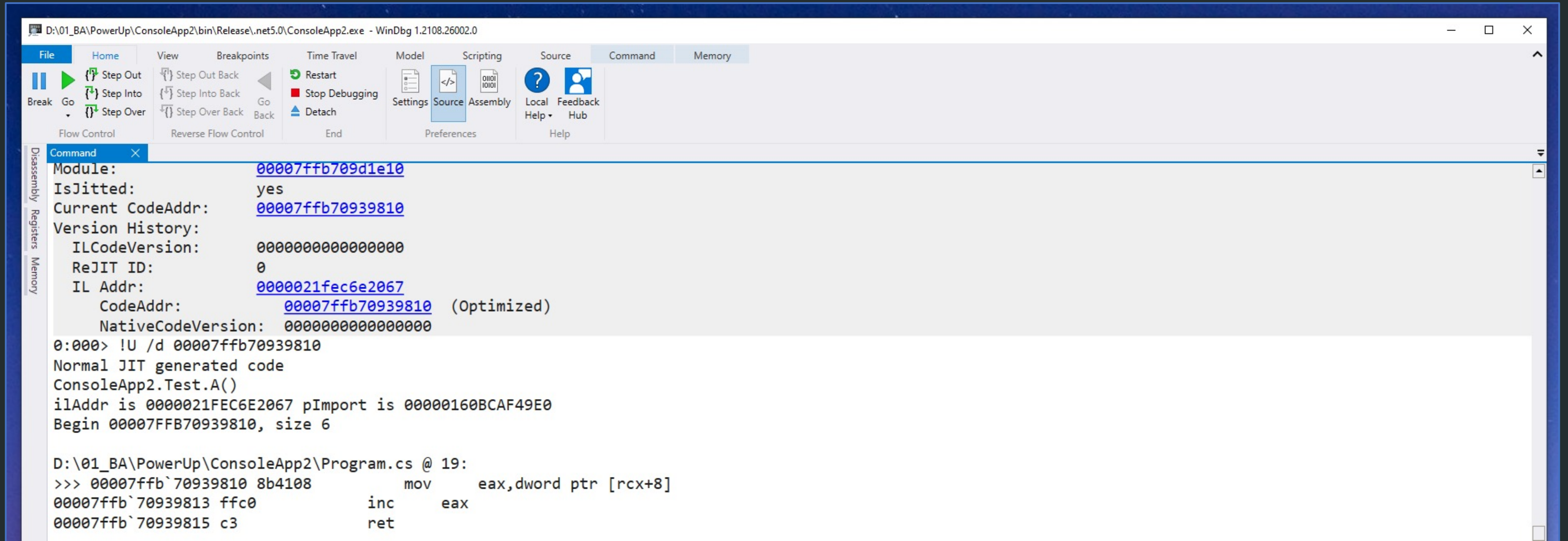
The image shows a side-by-side comparison of C# source code and its assembly output. On the left, the C# source file 'code.cs' contains a simple loop that iterates from 0 to 9 and assigns the value of 'i' to 'a[i]'. On the right, the assembly file 'out.asm' shows the corresponding machine code for this loop, including instructions for stack management, register manipulation, and conditional jumps.

```
C# code.cs
C: > 02_other > C# code.cs
1
2
3
4 public void A(int[] a) {
5     for(int i = 0; i < 10; i++)
6         a[i] = i;
7 }
```

```
ASM out.asm
C: > 02_other > ASM out.asm
1
2 Void A(Int32[]):
3     7FFF885B04A0: sub    rsp, 28h
4     7FFF885B04A4: xor    eax, eax
5     7FFF885B04A6: test   rdx, rdx
6     7FFF885B04A9: je     short 7FFF885B04C1 ↓
7     7FFF885B04AB: cmp    dword ptr [rdx+8], 0A
8     7FFF885B04AF: jl     short 7FFF885B04C1 ↓
9     7FFF885B04B1: movsxd rcx, eax
10    7FFF885B04B4: mov    [rdx+rcx*4+10h], eax
11    7FFF885B04B8: inc    eax
12    7FFF885B04BA: cmp    eax, 0Ah
13    7FFF885B04BD: jl     short 7FFF885B04B1 ↑
```


Tools?

WinDBG – Best Debugger Ever Created



The screenshot shows the WinDBG interface with the Command window open. The Command window displays the following output:

```
Module: 00007ffb709d1e10
IsJitted: yes
Current CodeAddr: 00007ffb70939810
Version History:
  ILCodeVersion: 0000000000000000
  ReJIT ID: 0
  IL Addr: 0000021fec6e2067
  CodeAddr: 00007ffb70939810 (Optimized)
  NativeCodeVersion: 0000000000000000
0:000> !U /d 00007ffb70939810
Normal JIT generated code
ConsoleApp2.Test.A()
ilAddr is 0000021FEC6E2067 pImport is 00000160BCAF49E0
Begin 00007FFB70939810, size 6

D:\01_BA\PowerUp\ConsoleApp2\Program.cs @ 19:
>>> 00007ffb`70939810 8b4108      mov     eax,dword ptr [rcx+8]
00007ffb`70939813 ffc0      inc     eax
00007ffb`70939815 c3      ret
```

The Mine Field.

Optimizations and Surprises in .NET 5 / 6

00 Folding

01 Conditions

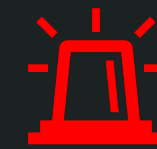
02 Throw Helper

03 Loops

04 Structs (AKA ValueTypes)

05 Try-Catch

06 Inlining



← Hazard Level

Introduction to X86 Assembly.

Part 1 – Basic Operations

mov A, B A = B

mov A, [B] A = Memory[B]

lea A, [B+C] A = B + C

add A, B A += B

cmp A, B A Compare B (A - B)

Introduction to X86 Assembly.

Part 2 – Conditions

`je` `A == B` `jne` `A != B`

`j1` `A < B` `jg` `A > B`

`jle` `A <= B` `jge` `A >= B`

Demo.

Let's start with a bomb



Bounds Check Demo.

Bounds Check (huh?)

Method	Job	Runtime	size	Mean	Error
LoopBounds	.NET 5.0	.NET 5.0	10000	6.442 us	0.1090 us
Loop	.NET 5.0	.NET 5.0	10000	7.731 us	0.1434 us

Compound Demo.

Compound

```
BenchmarkDotNet=v0.13.1, OS=Windows 10.0.19042.1237 (20H2/October2020Update)
Intel Core i7-6700HQ CPU 2.60GHz (Skylake), 1 CPU, 8 logical and 4 physical cores
.NET SDK=6.0.100-preview.7.21379.14
  [Host] : .NET 5.0.0 (5.0.20.51904), X64 RyuJIT
  .NET 5.0 : .NET 5.0.0 (5.0.20.51904), X64 RyuJIT
```

```
Job=.NET 5.0 Runtime=.NET 5.0
```

Method	size	x	Mean	Error	StdDev	Ratio	RatioSD
CompoundAdd	10000	0	8.945 us	0.1704 us	0.1673 us	1.00	0.00
NonCompoundAdd	10000	0	8.017 us	0.1595 us	0.2184 us	0.90	0.03

Compound Demo.

Compound + Loop Clone

```
BenchmarkDotNet=v0.13.1, OS=Windows 10.0.19042.1237 (20H2/October2020Update)
Intel Core i7-6700HQ CPU 2.60GHz (Skylake), 1 CPU, 8 logical and 4 physical cores
.NET SDK=6.0.100-preview.7.21379.14
  [Host] : .NET 5.0.0 (5.0.20.51904), X64 RyuJIT
  .NET 5.0 : .NET 5.0.0 (5.0.20.51904), X64 RyuJIT
```

```
Job=.NET 5.0 Runtime=.NET 5.0
```

Method	size	x	Mean	Error	StdDev	Ratio	RatioSD
CompoundAdd	10000	0	9.994 us	0.1500 us	0.1330 us	1.00	0.00
NonCompoundAdd	10000	0	7.962 us	0.1368 us	0.1464 us	0.80	0.02

Hoisting Demo.

Hoisting

BenchmarkDotNet=v0.13.1, OS=Windows 10.0.19042.1237 (20H2/October2020Update)
Intel Core i7-6700HQ CPU 2.60GHz (Skylake), 1 CPU, 8 logical and 4 physical cores

.NET SDK=6.0.100-preview.7.21379.14

[Host] : .NET 5.0.0 (5.0.20.51904), X64 RyuJIT

.NET 5.0 : .NET 5.0.0 (5.0.20.51904), X64 RyuJIT

Job=.NET 5.0 Runtime=.NET 5.0

Method	size	x	Mean	Error	StdDev	Ratio
FailedHoisting	10000	0	16.867 us	0.2611 us	0.2442 us	1.00
ManualHoisting	10000	0	6.258 us	0.0942 us	0.0835 us	0.37

Struct Demo.

Increment

```
BenchmarkDotNet=v0.13.1, OS=Windows 10.0.19042.1237 (20H2/October2020Update)
Intel Core i7-6700HQ CPU 2.60GHz (Skylake), 1 CPU, 8 logical and 4 physical cores
.NET SDK=6.0.100-preview.7.21379.14
  [Host] : .NET 5.0.0 (5.0.20.51904), X64 RyuJIT
.NET 5.0 : .NET 5.0.0 (5.0.20.51904), X64 RyuJIT
```

```
Job=.NET 5.0 Runtime=.NET 5.0
```

Method	size	Mean	Error	StdDev	Ratio	RatioSD
StructAdd	10000	18.575 us	0.3679 us	0.8672 us	1.00	0.00
StructAddFast	10000	4.182 us	0.0955 us	0.2755 us	0.23	0.02

Struct Demo.

DeVirt

```
BenchmarkDotNet=v0.13.1, OS=Windows 10.0.19042.1237 (20H2/October2020Update)
Intel Core i7-6700HQ CPU 2.60GHz (Skylake), 1 CPU, 8 logical and 4 physical cores
.NET SDK=6.0.100-preview.7.21379.14
  [Host] : .NET 5.0.0 (5.0.20.51904), X64 RyuJIT
  .NET 5.0 : .NET 5.0.0 (5.0.20.51904), X64 RyuJIT
```

```
Job=.NET 5.0 Runtime=.NET 5.0
```

Method	size	Mean	Error	StdDev	Ratio
DeVirt	10000	27.550 us	0.2539 us	0.2120 us	1.00
NoVirt	10000	3.457 us	0.0264 us	0.0220 us	0.13

„.NET 6 will fix all of the problems”

- Someone on [Reddit](#) | [FB](#) | [Twitter](#)

Compound Demo.

Compound

Method	Job	Runtime	size	x	Mean	Error	StdDev	Median	Ratio	RatioSD
CompoundAdd	.NET 5.0	.NET 5.0	10000	0	9.671 us	0.2524 us	0.7441 us	9.423 us	1.00	0.00
NonCompoundAdd	.NET 5.0	.NET 5.0	10000	0	7.739 us	0.1194 us	0.1116 us	7.722 us	0.75	0.05
CompoundAdd	.NET 6.0	.NET 6.0	10000	0	8.898 us	0.1602 us	0.1420 us	8.873 us	1.00	0.00
NonCompoundAdd	.NET 6.0	.NET 6.0	10000	0	7.841 us	0.1268 us	0.1059 us	7.859 us	0.88	0.02

Bounds Check Demo.

Bounds Check (huh?)

Method	Job	Runtime	size	Mean	Error	StdDev	Ratio	RatioSD
LoopBounds	.NET 5.0	.NET 5.0	10000	6.442 us	0.1090 us	0.1020 us	1.00	0.00
Loop	.NET 5.0	.NET 5.0	10000	7.731 us	0.1434 us	0.2190 us	1.21	0.04
LoopBounds	.NET 6.0	.NET 6.0	10000	6.633 us	0.1320 us	0.1571 us	1.00	0.00
Loop	.NET 6.0	.NET 6.0	10000	6.400 us	0.1162 us	0.1087 us	0.96	0.03

Struct Demo.

Increment

Method	Job	Runtime	size	Mean	Error	StdDev	Ratio
StructAdd	.NET 5.0	.NET 5.0	10000	19.051 us	0.2479 us	0.2318 us	1.00
StructAddFast	.NET 5.0	.NET 5.0	10000	3.899 us	0.0774 us	0.1227 us	0.21
StructAdd	.NET 6.0	.NET 6.0	10000	19.432 us	0.1135 us	0.1006 us	1.00
StructAddFast	.NET 6.0	.NET 6.0	10000	3.853 us	0.0731 us	0.0684 us	0.20

Struct Demo.

DeVirt

Method	Job	Runtime	size	Mean	Error	StdDev	Ratio
NoVirt	.NET 5.0	.NET 5.0	10000	3.449 us	0.0332 us	0.0277 us	0.13
DeVirt	.NET 6.0	.NET 6.0	10000	3.460 us	0.0224 us	0.0210 us	1.00
NoVirt	.NET 6.0	.NET 6.0	10000	3.463 us	0.0205 us	0.0171 us	1.00

Questions?



[badamczewski01](#)



[leveluppp.ghost.io](#)



[leveluppp](#)

iyuno • SDI
GROUP