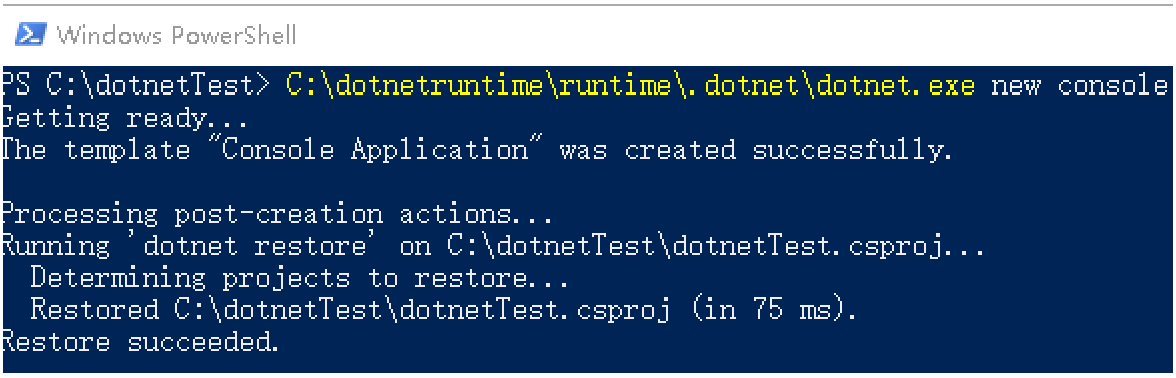
1.使用dotnet生成dll文件

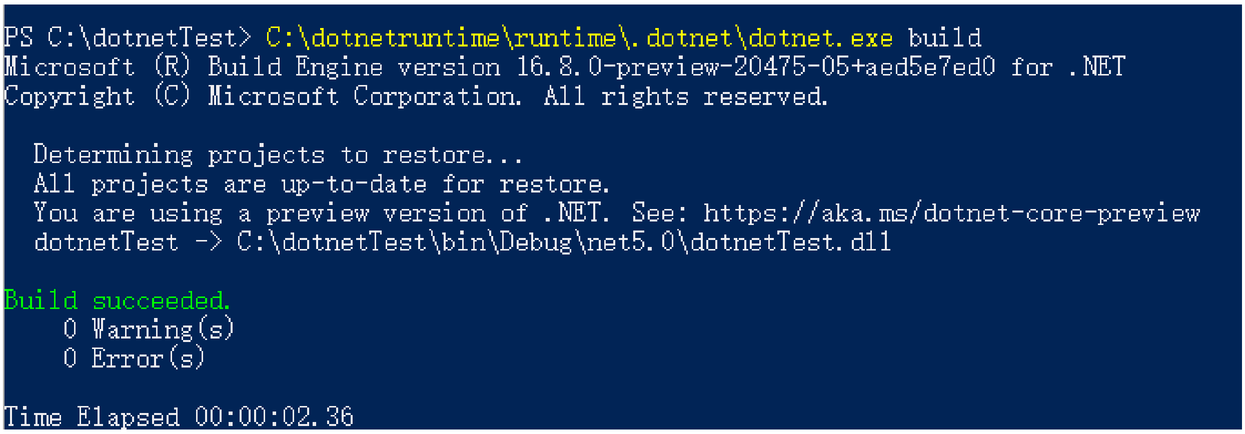
新建一个文件夹，在该文件夹目录下打开 powershell(shift + 鼠标右键)

C:\dotnetruntime\runtime.dotnet\dotnet.exe new console



```
PS C:\dotnetTest> C:\dotnetruntime\runtime\.dotnet\dotnet.exe new console
Getting ready...
The template "Console Application" was created successfully.

Processing post-creation actions...
Running 'dotnet restore' on C:\dotnetTest\dotnetTest.csproj...
  Determining projects to restore...
  Restored C:\dotnetTest\dotnetTest.csproj (in 75 ms).
Restore succeeded.
```

C:\dotnetruntime\runtime.dotnet\dotnet.exe build



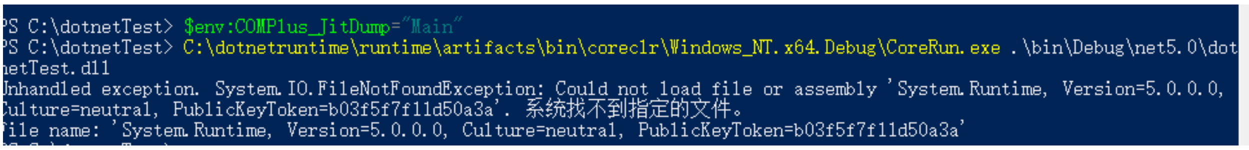
```
PS C:\dotnetTest> C:\dotnetruntime\runtime\.dotnet\dotnet.exe build
Microsoft (R) Build Engine version 16.8.0-preview-20475-05+aed5e7ed0 for .NET
Copyright (C) Microsoft Corporation. All rights reserved.

  Determining projects to restore...
  All projects are up-to-date for restore.
  You are using a preview version of .NET. See: https://aka.ms/dotnet-core-preview
  dotnetTest -> C:\dotnetTest\bin\Debug\net5.0\dotnetTest.dll

Build succeeded.
    0 Warning(s)
    0 Error(s)

Time Elapsed 00:00:02.36
```

ls

2对dll文件进行编译

$env : COMPlus_J itDump =" \ Main \ "$ （双引号内填函数名，*
就是全部函数）
env:COMPlus_JitDump="*"

C:\dotnetruntime\runtime\artifacts\bin\coreclr\Windows_NT.x64.Debug\CoreRun.exe .\bin\Debug\net5.0\dotnetTest.dll



```
PS C:\dotnetTest> $env:COMPlus_JitDump="Main"
PS C:\dotnetTest> C:\dotnetruntime\runtime\artifacts\bin\coreclr\Windows_NT.x64.Debug\CoreRun.exe .\bin\Debug\net5.0\dot
netTest.dll
Unhandled exception. System.IO.FileNotFoundException: Could not load file or assembly 'System.Runtime, Version=5.0.0.0,
Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a'. 系统找不到指定的文件。
File name: 'System.Runtime, Version=5.0.0.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a'
```

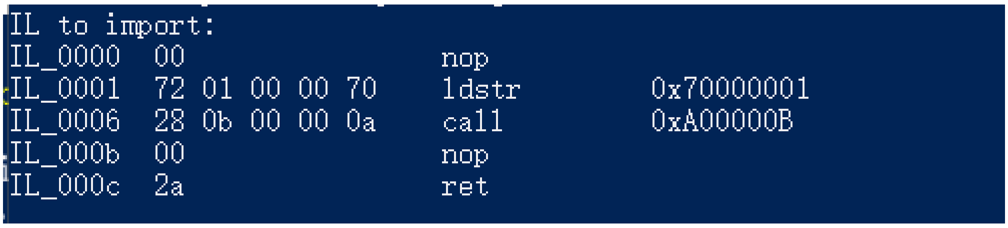
3设置环境变量

变量名 CORE_LIBRARIES

变量值 C:\dotnetruntime\runtime.dotnet\shared\Microsoft.NETCore.App\5.0.0-rc.2.20475.5

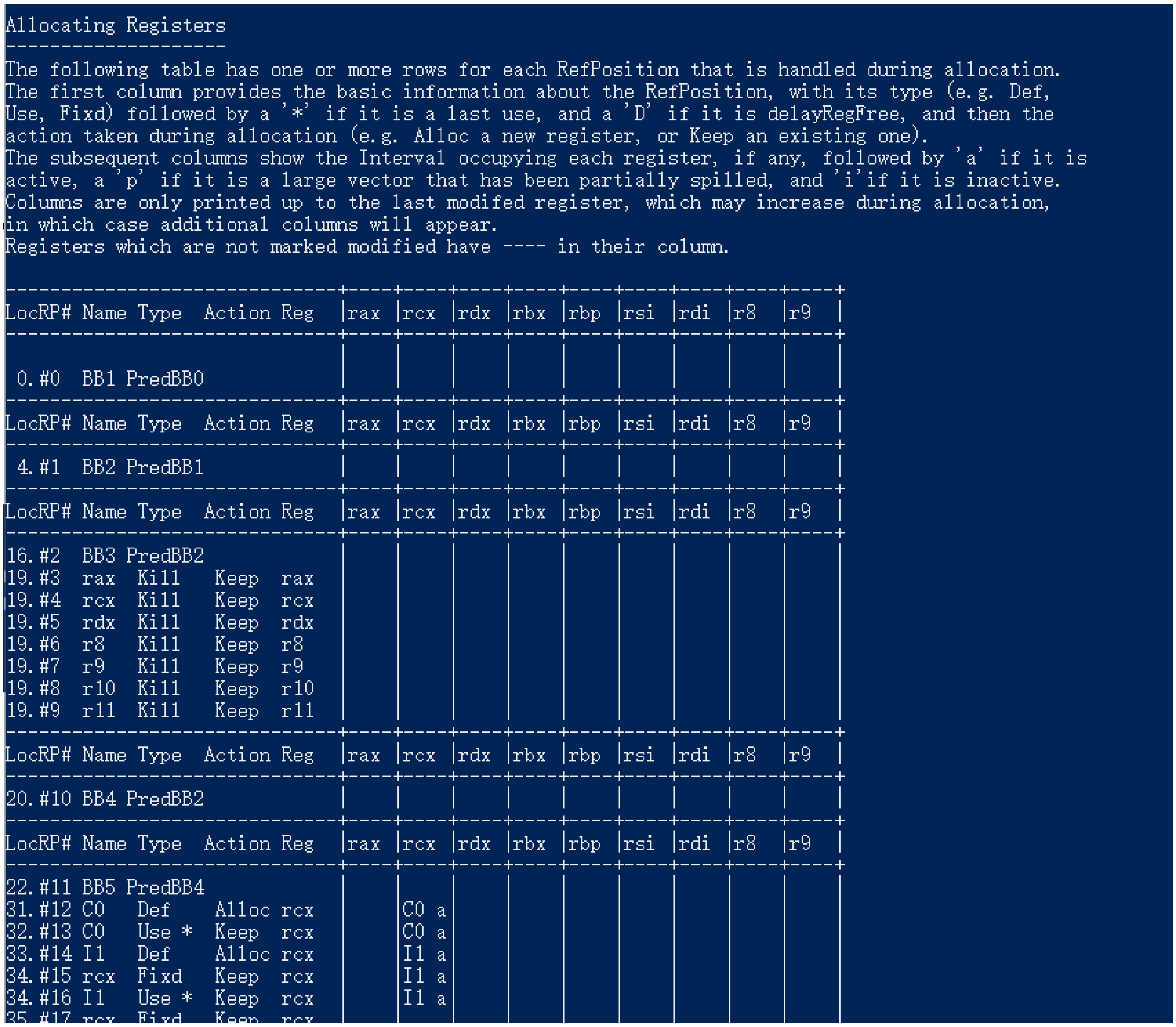
部分编译过程的output解读：

生成的dll代码：



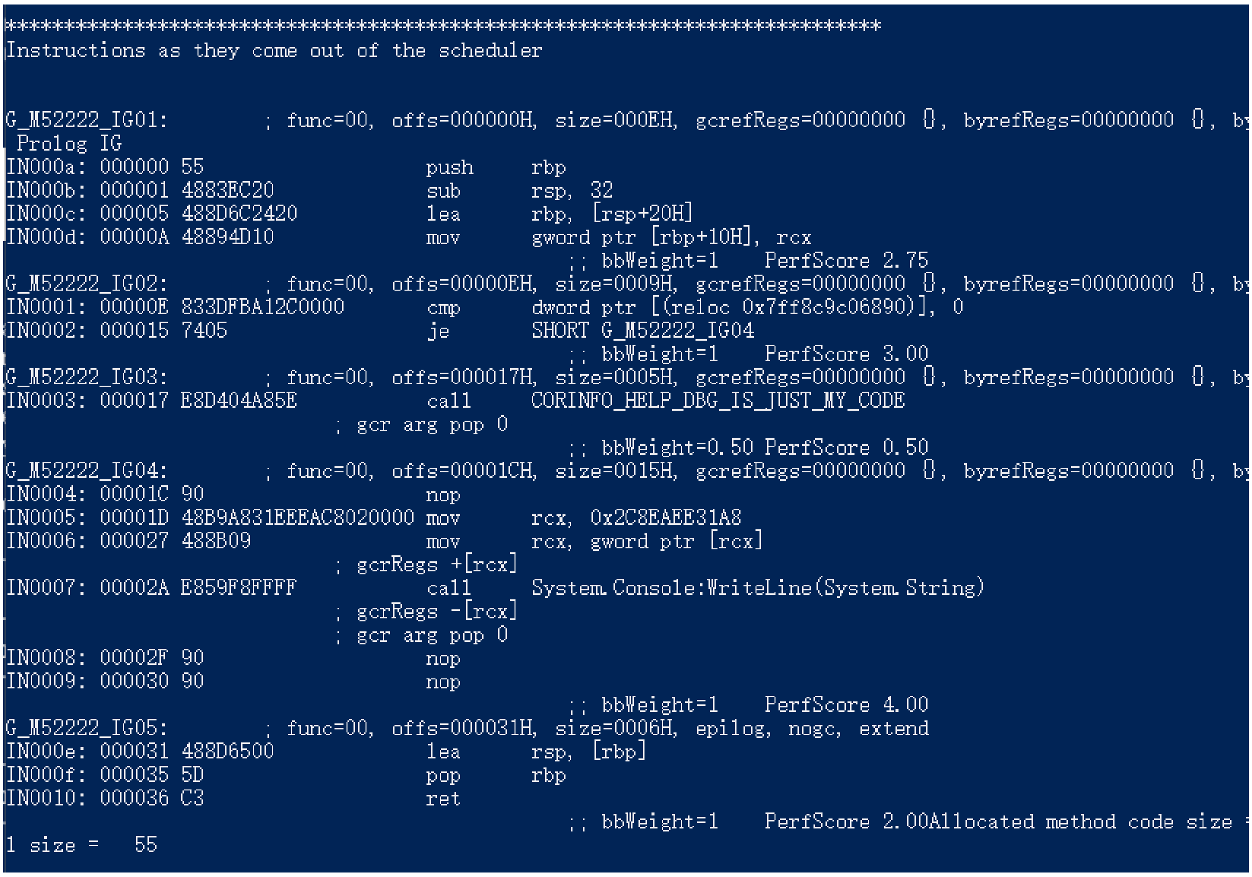
```
IL to import:
IL_0000  00                nop
IL_0001  72 01 00 00 70    ldstr        0x70000001
IL_0006  28 0b 00 00 0a    call         0xA000000B
IL_000b  00                nop
IL_000c  2a                ret
```

分配寄存器：



```
Allocating Registers
--------------------
The following table has one or more rows for each RefPosition that is handled during allocation.
The first column provides the basic information about the RefPosition, with its type (e.g. Def,
Use, Fixd) followed by a '*' if it is a last use, and a 'D' if it is delayRegFree, and then the
action taken during allocation (e.g. Alloc a new register, or Keep an existing one).
The subsequent columns show the Interval occupying each register, if any, followed by 'a' if it is
active, a 'p' if it is a large vector that has been partially spilled, and 'i'if it is inactive.
Columns are only printed up to the last modifed register, which may increase during allocation,
in which case additional columns will appear.
Registers which are not marked modified have ---- in their column.

------------------------------+----+----+----+----+----+----+----+----+----+
LocRP# Name Type  Action Reg  |rax |rcx |rdx |rbx |rbp |rsi |rdi |r8  |r9  |
------------------------------+----+----+----+----+----+----+----+----+----+
                              |    |    |    |    |    |    |    |    |    |
 0.#0  BB1 PredBB0            |    |    |    |    |    |    |    |    |    |
------------------------------+----+----+----+----+----+----+----+----+----+
LocRP# Name Type  Action Reg  |rax |rcx |rdx |rbx |rbp |rsi |rdi |r8  |r9  |
------------------------------+----+----+----+----+----+----+----+----+----+
 4.#1  BB2 PredBB1            |    |    |    |    |    |    |    |    |    |
------------------------------+----+----+----+----+----+----+----+----+----+
LocRP# Name Type  Action Reg  |rax |rcx |rdx |rbx |rbp |rsi |rdi |r8  |r9  |
------------------------------+----+----+----+----+----+----+----+----+----+
16.#2  BB3 PredBB2            |    |    |    |    |    |    |    |    |    |
19.#3  rax  Kill   Keep  rax  |    |    |    |    |    |    |    |    |    |
19.#4  rcx  Kill   Keep  rcx  |    |    |    |    |    |    |    |    |    |
19.#5  rdx  Kill   Keep  rdx  |    |    |    |    |    |    |    |    |    |
19.#6  r8   Kill   Keep  r8   |    |    |    |    |    |    |    |    |    |
19.#7  r9   Kill   Keep  r9   |    |    |    |    |    |    |    |    |    |
19.#8  r10  Kill   Keep  r10  |    |    |    |    |    |    |    |    |    |
19.#9  r11  Kill   Keep  r11  |    |    |    |    |    |    |    |    |    |
------------------------------+----+----+----+----+----+----+----+----+----+
LocRP# Name Type  Action Reg  |rax |rcx |rdx |rbx |rbp |rsi |rdi |r8  |r9  |
------------------------------+----+----+----+----+----+----+----+----+----+
20.#10 BB4 PredBB2            |    |    |    |    |    |    |    |    |    |
------------------------------+----+----+----+----+----+----+----+----+----+
LocRP# Name Type  Action Reg  |rax |rcx |rdx |rbx |rbp |rsi |rdi |r8  |r9  |
------------------------------+----+----+----+----+----+----+----+----+----+
22.#11 BB5 PredBB4            |    |    |    |    |    |    |    |    |    |
31.#12 C0   Def    Alloc rcx  |    |C0 a|    |    |    |    |    |    |    |
32.#13 C0   Use *  Keep  rcx  |    |C0 a|    |    |    |    |    |    |    |
33.#14 I1   Def    Alloc rcx  |    |I1 a|    |    |    |    |    |    |    |
34.#15 rcx  Fixd   Keep  rcx  |    |I1 a|    |    |    |    |    |    |    |
34.#16 I1   Use *  Keep  rcx  |    |I1 a|    |    |    |    |    |    |    |
35.#17 rcx  Fixd   Keep  rcx  |    |    |    |    |    |    |    |    |    |
```

编译生成成的汇编指令：



```
***************************************************************************
Instructions as they come out of the scheduler


G_M52222_IG01:        ; func=00, offs=000000H, size=000EH, gcrefRegs=00000000 {}, byrefRegs=00000000 {}, by
 Prolog IG
IN000a: 000000 55                   push     rbp
IN000b: 000001 4883EC20             sub      rsp, 32
IN000c: 000005 488D6C2420           lea      rbp, [rsp+20H]
IN000d: 00000A 48894D10             mov      gword ptr [rbp+10H], rcx
                                                 ;; bbWeight=1    PerfScore 2.75
G_M52222_IG02:        ; func=00, offs=00000EH, size=0009H, gcrefRegs=00000000 {}, byrefRegs=00000000 {}, by
IN0001: 00000E 833DFBA12C0000       cmp      dword ptr [(reloc 0x7ff8c9c06890)], 0
IN0002: 000015 7405                 je       SHORT G_M52222_IG04
                                                 ;; bbWeight=1    PerfScore 3.00
G_M52222_IG03:        ; func=00, offs=000017H, size=0005H, gcrefRegs=00000000 {}, byrefRegs=00000000 {}, by
IN0003: 000017 E8D404A85E           call     CORINFO_HELP_DBG_IS_JUST_MY_CODE
                            ; gcr arg pop 0
                                                 ;; bbWeight=0.50 PerfScore 0.50
G_M52222_IG04:        ; func=00, offs=00001CH, size=0015H, gcrefRegs=00000000 {}, byrefRegs=00000000 {}, by
IN0004: 00001C 90                   nop
IN0005: 00001D 48B9A831EEEAC8020000 mov      rcx, 0x2C8EAEE31A8
IN0006: 000027 488B09               mov      rcx, gword ptr [rcx]
                            ; gcrRegs +[rcx]
IN0007: 00002A E859F8FFFF           call     System.Console:WriteLine(System.String)
                            ; gcrRegs -[rcx]
                            ; gcr arg pop 0
IN0008: 00002F 90                   nop
IN0009: 000030 90                   nop
                                                 ;; bbWeight=1    PerfScore 4.00
G_M52222_IG05:        ; func=00, offs=000031H, size=0006H, epilog, nogc, extend
IN000e: 000031 488D6500             lea      rsp, [rbp]
IN000f: 000035 5D                   pop      rbp
IN0010: 000036 C3                   ret
                                                 ;; bbWeight=1    PerfScore 2.00Allocated method code size =
1 size =   55
```

编译dll文件的源程序

https://github.com/dotnet/runtime/blob/master/src/coreclr/src/jit/compiler.cpp

https://github.com/dotnet/runtime/blob/ee2355c801d892f2894b0f7b14a20e6cc50e0e54/docs/design/coreclr/jit/viewing-jit-dumps.md