Disassemble Memory File

Memory.DMP can be used to determine call graphs, outside the scope of analysis tools like kd.exe. With a full disassembly, the context around a particular symbol is conveniently explored.

```
uf nt!HalGetBusDataByOffset
├────────────────────────▷uf nt!HalpGetPCIData
│                                          uf nt!HalpReadPCIConfig
│                                          uf nt!memcpy
│                                          uf nt!_security_check_cookie
└────────────────────────▷uf nt!HalpGetSetCmosData
                                             uf nt!HalpAcquireCmosSpinLock
                                             uf nt!guard_dispatch_icall (N/A)
```

UfSymbol.ps1 renders the call graph based on a disassembly file. The file is generated once, reused at rendering stage. The disassembly is separated into individual function bodies. The root body contains the symbol requested by the user. A dependency graph contains the callers or the callees for each function. CLI switches determine the depth of the tree, target OS for rendering.

- Generic functions have many callers; ie. 1118 matches for `nt!KeBugCheckEx` at `-Depth 1`.

To keep the graph uncluttered, known functions are not disassembled: `KeYieldProcessorEx` calls other functions that are minute, `IofCompleteRequest`.

Sample output renders the call tree for `nt!KiSystemStartup`.

```
PS > (Measure-Command {
         $Image = 'D:\DataLake\2025-04-28\MEMORY.DMP'
         & '.\UfSymbol.ps1' -Symbol nt!KiSystemStartup -Image $Image -Depth 4 -Down | Out-Default
      }).TotalSeconds
File "D:\DataLake\2025-01-28\MEMORY.DMP" of 1194.36 Mb has been processed in 4570 seconds.
D:\Processing\53c6f2af-38db-4219-9f41-f794c7897f5a\53c6f2af-38db-4219-9f41-f794c7897f5a.disassembly
D:\Processing\53c6f2af-38db-4219-9f41-f794c7897f5a\53c6f2af-38db-4219-9f41-f794c7897f5a.meta
D:\Processing\53c6f2af-38db-4219-9f41-f794c7897f5a\53c6f2af-38db-4219-9f41-f794c7897f5a.retpoline
```

The 1st line shows an estimate for the minimum disassembly duration: a smaller file was processed in 1.26 hours on the same system. The decompilation is executed on all cores but 1. Besides the `.disassembly` file, `.meta` and `.retpoline` are created. The `.meta` file contains:

- *OS* and *computer* where the BSOD occurred
- *image* path and *hash*. The hash identifies duplicates, resulting in a decompilation bypass.
- *system* where disassembly took place, number of *cpus* alloted, cpu *model*, *duration* and image *size*.
- The default modules used to disassemble the image:
  - for a `.dmp` file *nt, pci, acpi and hal* functions are disassembled
  - *base name* for all others

The `.retpoline` file is an indirection table for bodies compiled with `/guard:cf`. Wherever `call nt!guard_dispatch_icall` is found, the source pointer is resolved in the memory file and displayed.

For `nt!KiSystemStartup` call tree:

- 1302 callees are identified with `-Depth 4`, 5318 at depth 6.
- Complete decompilation and identification takes **5215** seconds on an "Intel(R) Core(TM) i3-7100U CPU @ 2.40GHz" with 3 cpus.

```
uf nt!KdInitSystem
├──────────────────▷uf nt!KeQueryPerformanceCounter
│                                             uf nt!HalpTimerGetInternalData
│                                             uf nt!HalpTimerScaleCounter
│                  uf nt!ExAllocatePool2
│                  uf nt!_security_check_cookie
│                  uf nt!MmGetPagedPoolCommitPointer
├──────────────────▷uf nt!KdRegisterDebuggerDataBlock
 ...
uf nt!PpmUpdatePerformanceFeedback
uf nt!guard_dispatch_icall (nt!_security_cookie
                            nt!HalpOriginalPerformanceCounter
```

```
                        nt!HalPrivateDispatchTable+0x1b0=nt!HalpProcessorPrepareForIdle
                        nt!HalPrivateDispatchTable+0x1c0=nt!HalpProcessorResumeFromIdle
                        nt!HalpTimerReferencePage
                        nt!HalPrivateDispatchTable+0x418=nt!HalpLbrResumeRecording
                        nt!HalPrivateDispatchTable+0x2f8=nt!HalpTimerClockStop
                        nt!PopCsConsumption+0x140)
5215.506918
```

`-Setup` is a text based guide that configures the directory where disassemblies are stored. A symbol path can be specified, a lower limit can trigger a warning if other dissasemblies overlast it. Disassembly duration and system, cpu model, file size can be suppressed from future `.meta` files.

Notes

- Decompilation-ready processing is useful in support cases where the *Memory.DMP* file cannot be provided. Implementation differences between OS versions are also visible.
  - A `.dmp` file contains the dependencies from all modules, can trip the decompiler with inappropriate function bodies. This shortcoming does not apply to user mode.
  - An executable solves all functions, cannot solve dependencies.
- PowerShell *Core* is required. *Desktop 5.1* is slow.
- Hotpaths are moved to inflight *CSharp* assembly. Decompilation can be **8 times** faster.
- Decompilation through *kd.exe* can be superseded by *dbgeng.dll* COM interfaces. Direct access to `dbgeng.h` has the benefit of measuring the decompilation process through a progress bar. Trimming of function bodies occurs ad hoc. Parallel *kd.exe* execution binds trimming to disassembly completion.
  - `IDebugControl::WaitForEvent` fails when clients are created by multiple threads.
    error message:
    ```
    Can't set dump file contexts
    MachineInfo::SetContext failed - Thread: 000001A2CDA07900  Handle: 1  Id: 1 - Error == 0x8000FFFF
    ```
  - `IDebugControl::Execute` is serialized with a *critical section*.
    disassembly:
    ```
    0:017> k
    # Child-SP          RetAddr               Call Site
    00 00000035`6ed8d300 00007ffa`976b15e0     dbgeng!DebugClient::ExecuteWide+0x23
    01 00000035`6ed8d350 00007ffa`37eccdc5     dbgeng!DebugClient::Execute+0xf0

    0:000> uf dbgeng!DebugClient::ExecuteWide

    00000001`80101bca 488d0dafe97a00  lea     rcx,[dbgeng!g_EngineLock (00000001`808b0580)]
    00000001`80101bd1 48ff15b0e65600  call   qword ptr [dbgeng!_imp_EnterCriticalSection (00000001`80670288)

    00000001`80101c06 e8990ffdff      call    dbgeng!PushOutCtl (00000001`800d2ba4)
    00000001`80101c23 e8e8f2ffff      call    dbgeng!Execute (00000001`80100f10)
    00000001`80101c2f e81807fdff      call    dbgeng!PopOutCtl (00000001`800d234c)
    00000001`80101c45 e896c2fcff      call    dbgeng!FlushCallbacks (00000001`800cdee0)

    00000001`80101c50 488d0d29e97a00  lea     rcx,[dbgeng!g_EngineLock (00000001`808b0580)]
    00000001`80101c57 48ff1512e65600  call   qword ptr [dbgeng!_imp_LeaveCriticalSection (00000001`80670270)
    ```
- Inbox `dbgeng.dll` *10.0.19041.3636* identifies fewer functions compared with latest *10.0.26100.2454* version.
- *UfSymbol* is meant for USB migration. No internet connection is needed.
- Where `(N/A)` appears in rendering:
  - indirection table has no corresponding target symbol - ie. register is used.
    rax ← qword ptr [rcx+20h]:
    ```
    uf nt!IoCsqRemoveIrp
    fffff803`2c9d0980 48895c2410      mov     qword ptr [rsp+10h],rbx
    fffff803`2c9d0985 4889742418      mov     qword ptr [rsp+18h],rsi
    fffff803`2c9d098a 57              push    rdi
    fffff803`2c9d098b 4883ec20        sub     rsp,20h
    fffff803`2c9d098f 488b4120        mov     rax,qword ptr [rcx+20h]
    fffff803`2c9d0993 488bf2          mov     rsi,rdx
    fffff803`2c9d0996 4883613800      and     qword ptr [rcx+38h],0
    fffff803`2c9d099b 488d542430      lea     rdx,[rsp+30h]
    ```

```
fffff803`2c9d09a0 488bd9          mov     rbx,rcx
fffff803`2c9d09a3 c644243000      mov     byte ptr [rsp+30h],0
fffff803`2c9d09a8 e833f70400      call    nt!guard_dispatch_icall (fffff803`2ca200e0)
```

  - function is missing the body either due to absent module, or a large body has been decompiled and trimmed.
- `.retpoline` build is not parallelized. Only 2E+3 *poi* sources have to be decoded.
- Initially, the tool's objective was GUI rendering through SVG. With broad trees being prevalent, a point-and-click is deemed impractical.

```
PS > $prefix = "https://raw.githubusercontent.com/armaber/scripts/refs/heads/disasm/";
     "functions.ps1", "UfSymbol.ps1" | foreach {
         Invoke-WebRequest $prefix/DisassembleImage/$PSItem -OutFile $PSItem;
     }
     Get-Help .\UfSymbol.ps1 -Full;
```