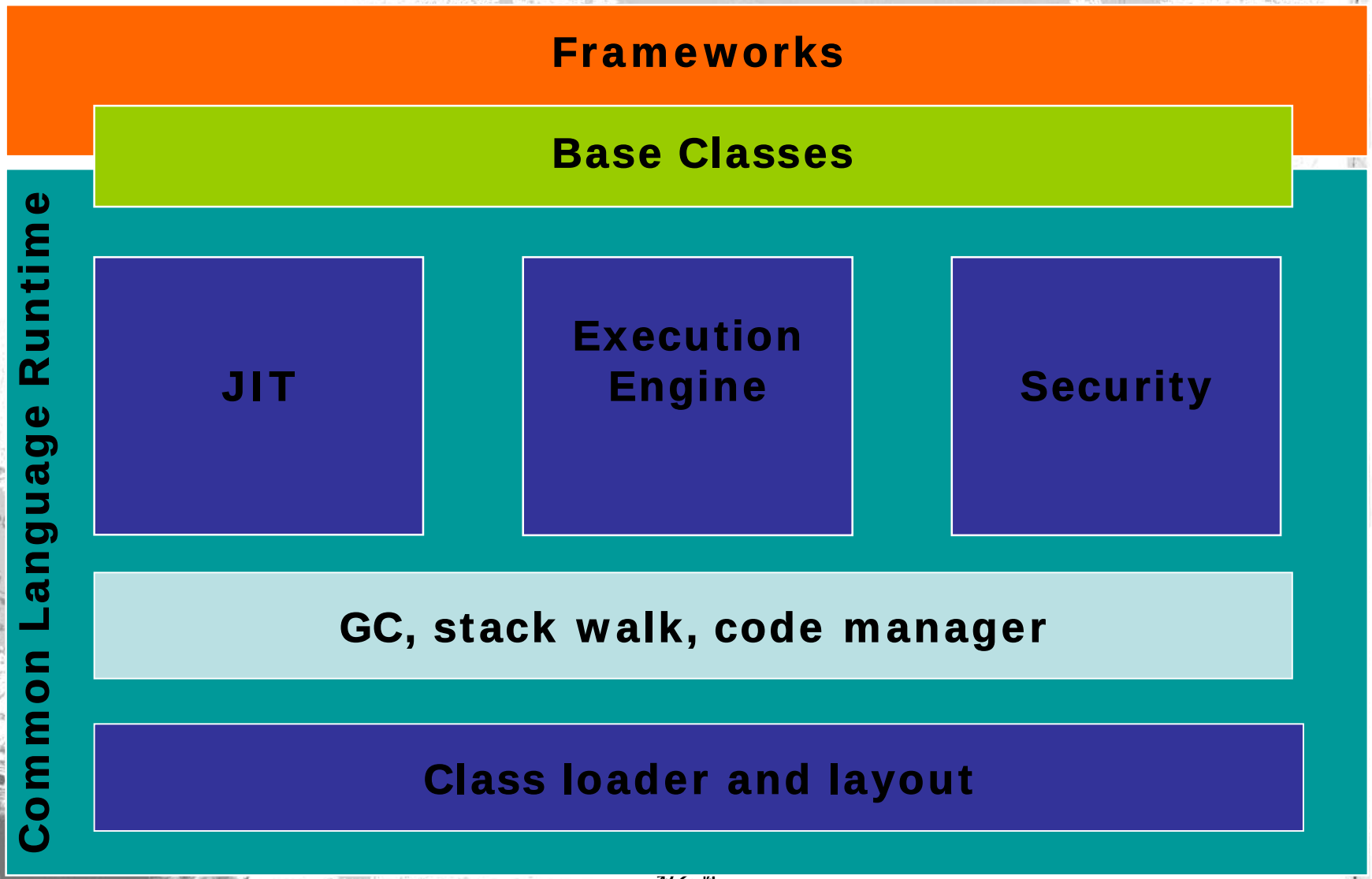
.NET安全分析
——基础架构与安全框架

卢小海

.NET 基础架构


Frameworks
Base Classes
Common Language Runtime
JIT
Execution Engine
Security
GC, stack walk, code manager
Class loader and layout

讨论议程

- 静态结构
  - 通过CTS提供语言无关类型支持；将类型定义存储在文件Metadata中；以IL指令描述行为。
- 动态执行
  - 将类型定义和代码载入并组织到内存；JIT编译并执行代码；通过堆栈帧提供遍历堆栈支持。
- 安全框架
  - 基于凭证的代码访问安全策略执行框架；
    基于角色的功能执行安全授权检查框架。

1.静态结构

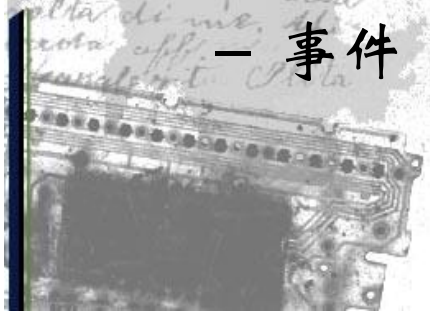
- 通用类型系统(CTS - Common Type System)
- 托管可执行文件结构 (Metadata)
- IL指令集(IL - Intermediate Language)

1.1通用类型系统

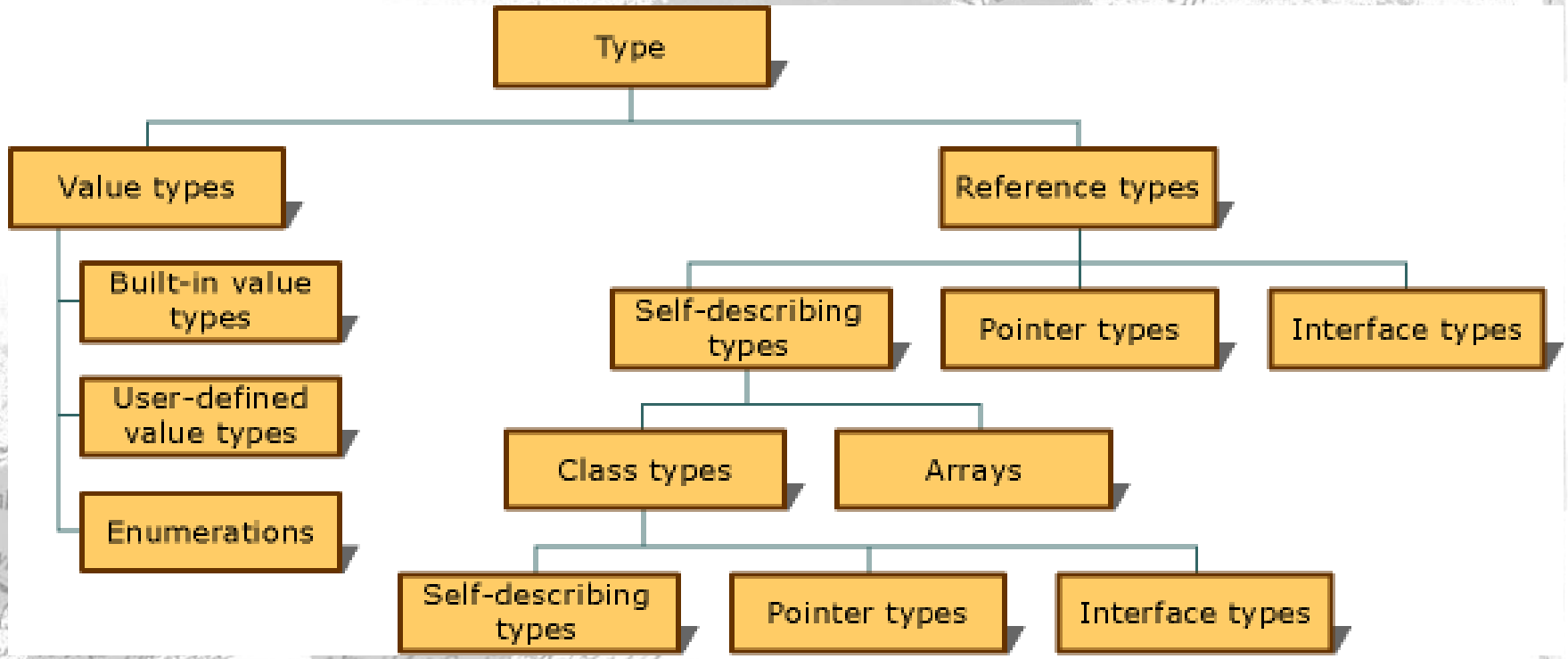
- 目标
  - 定义类型如何声明、使用和管理
  - CLR 跨语言集成的基础
- 特性
  - 可验证的强类型安全
  - 高性能的代码执行能力
  - 提供面向对象模型，以完整实现多种语言
  - 定义语言的元规则，以支持跨语言互操作

1.1.2.命名层次

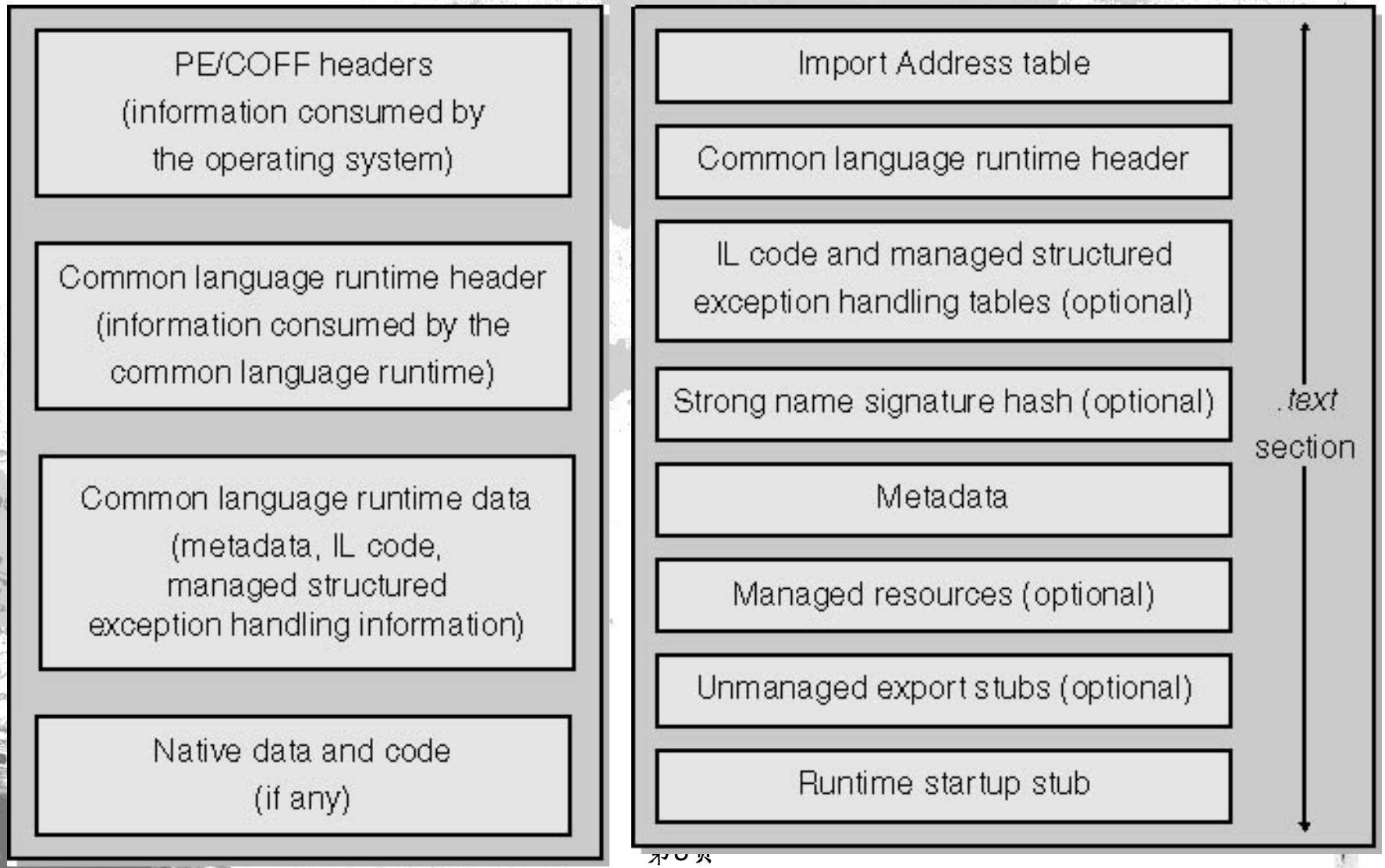
- 配件（Assembly）
- 模块（Module）
- 类型（Type）
- 成员（Member）
  - 方法（Method）
    - 参数和局部变量
  - 字段（Field）
  - 内嵌类型（Nested Type）
  - 属性（Property）
  - 事件（Event）



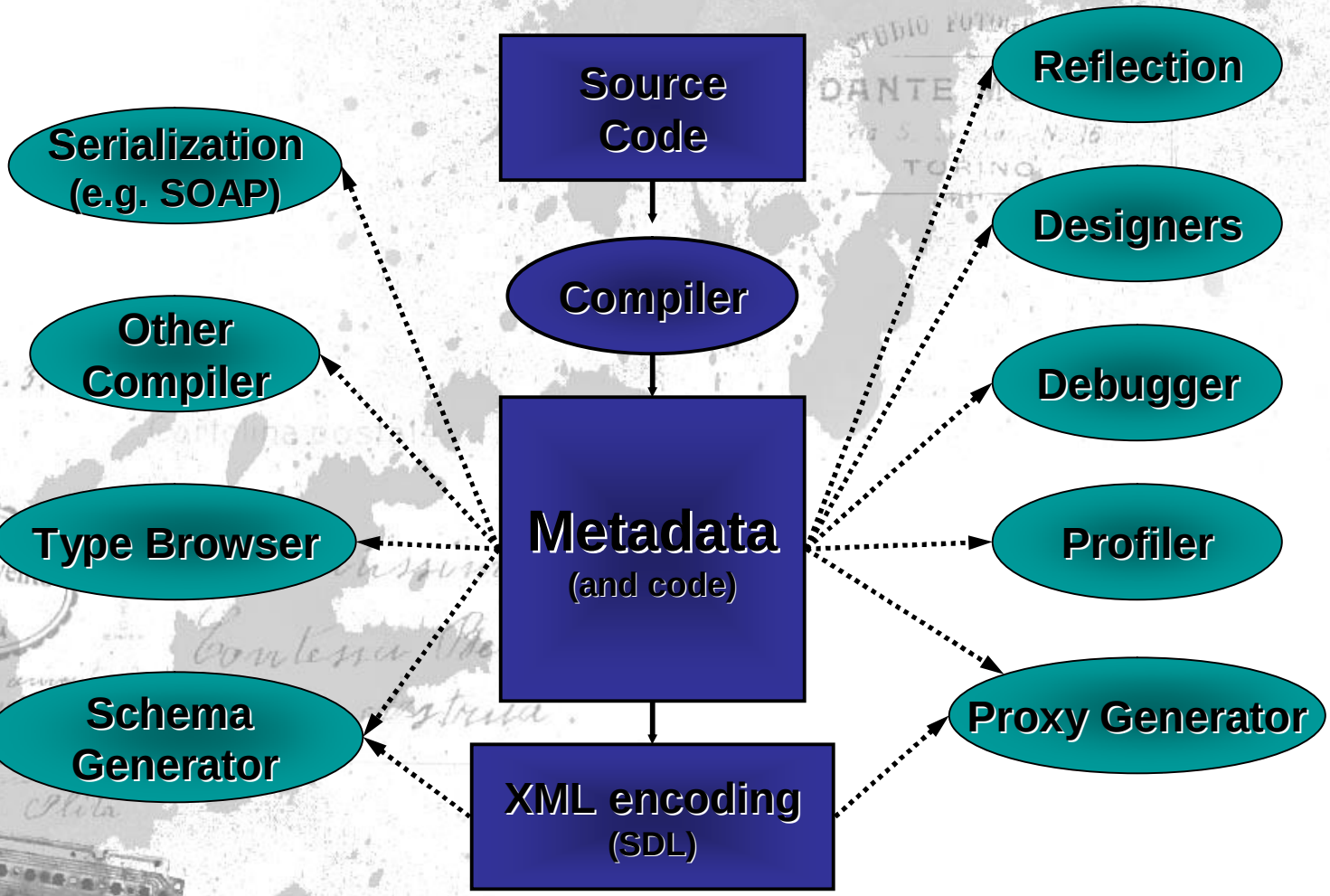
1.1.3.类型分类


Type
Value types
Reference types
Built-in value types
User-defined value types
Enumerations
Self-describing types
Pointer types
Interface types
Class types
Arrays
Self-describing types
Pointer types
Interface types

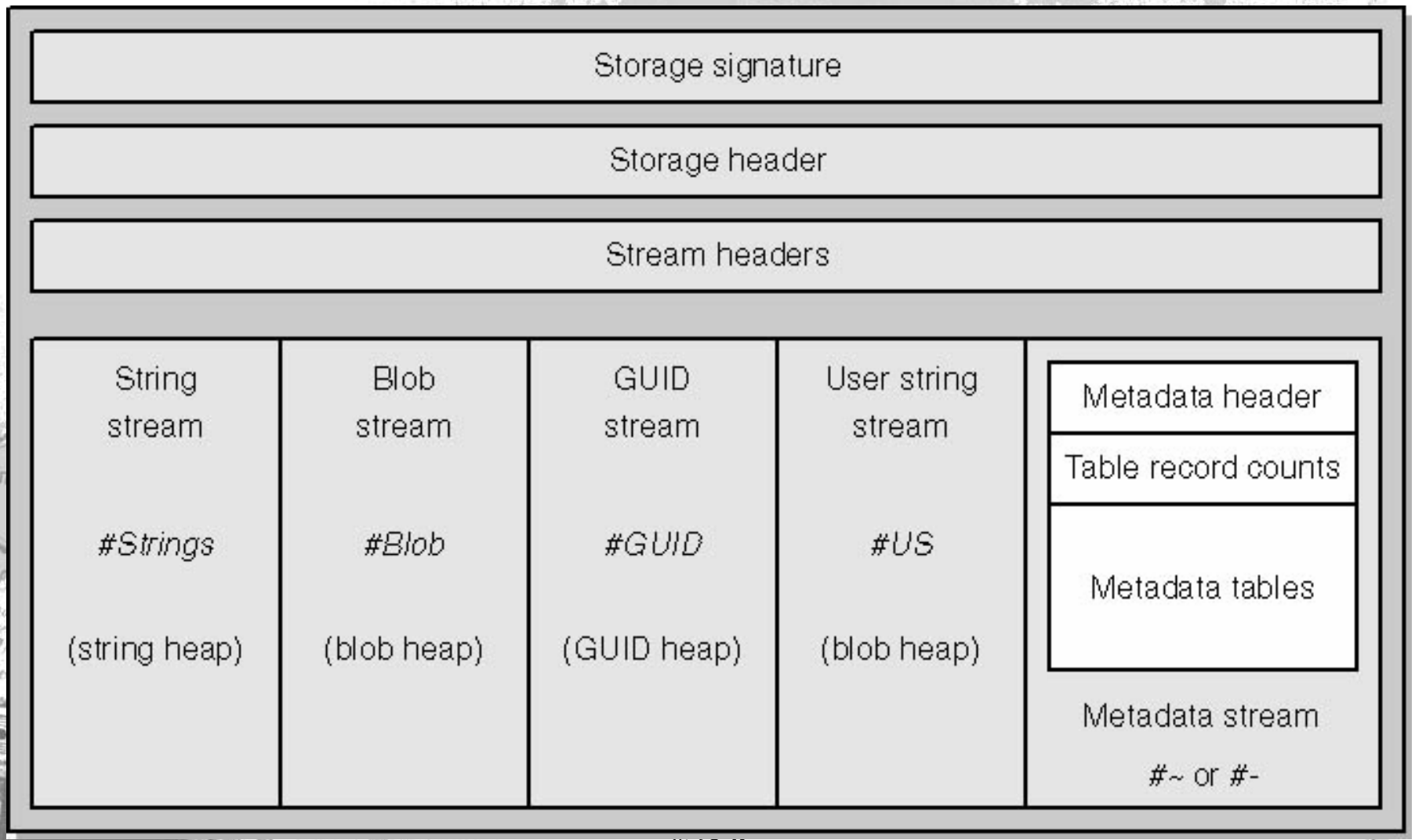
1.2.托管可执行文件结构



1.2.2Metadata的定位

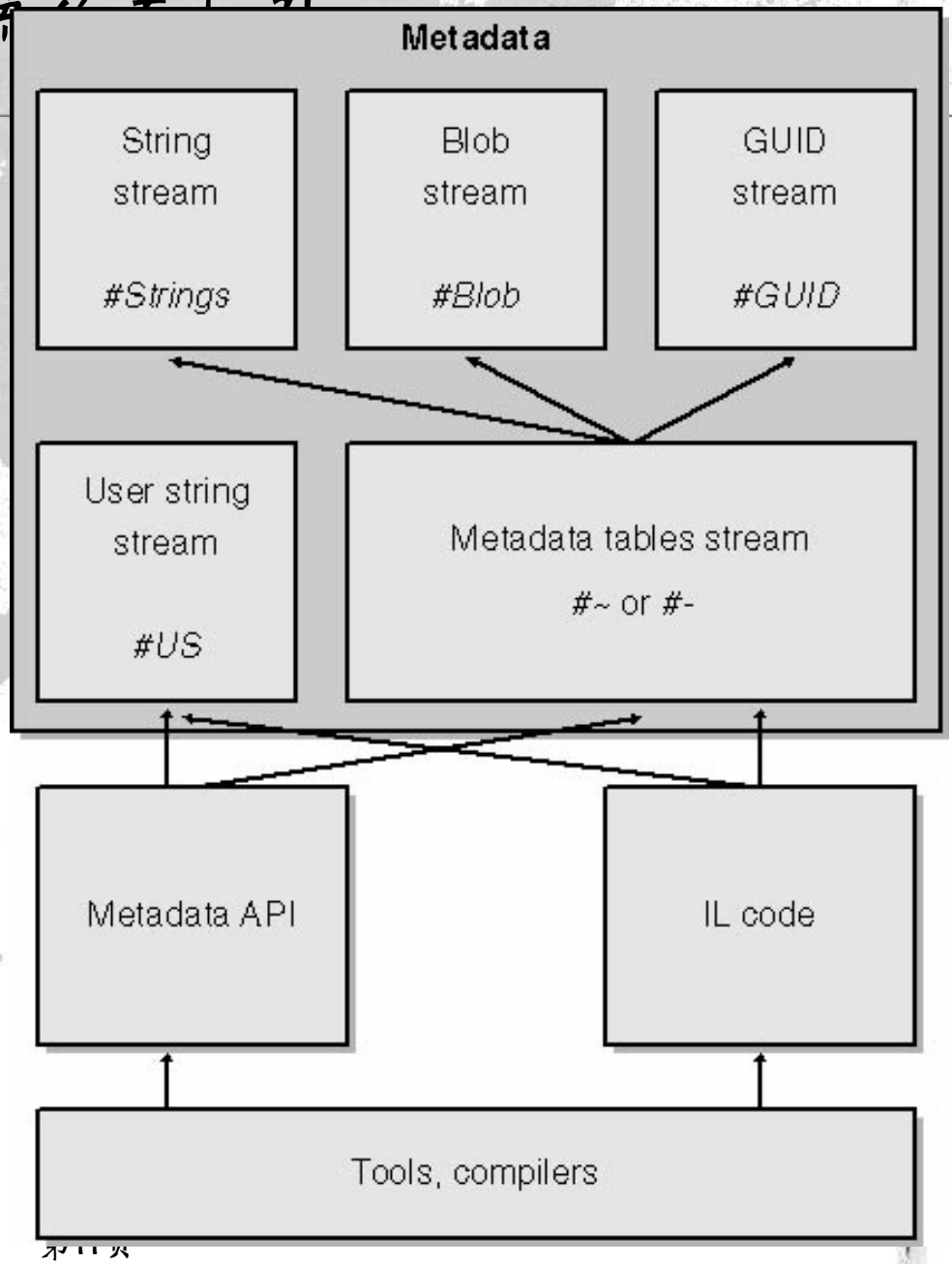


1.2.3Metadata的结构



1.2.4.Metadata流的开头部分

- #Strings
  - 元数据中数据项名称
- #Blob
  - 元数据中二进制对象
- #GUID
  - 各种 GUID
- #US
  - 用户自定义字符串
- #~ 或 #-
  - 元数据表系统

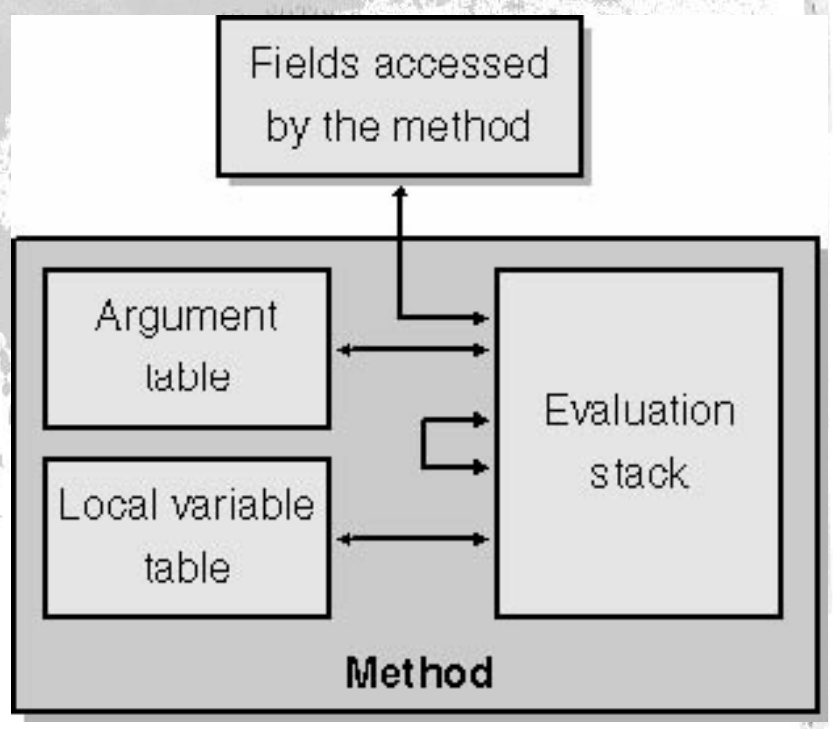


1.2.5.Metadata 数据表

Module	TypeRef	TypeDef
FieldDef	MethodDef	ParamDef
InterfaceImpl	MemberRef	CustomAttribute
Permission	Signature	Event
Property	ModuleRef	TypeSpec
Assembly	AssemblyRef	File
ExportedType	ManifestResource	GenericParam
GenericParamConstraint	MethodSpec	

1.3.IL指令集

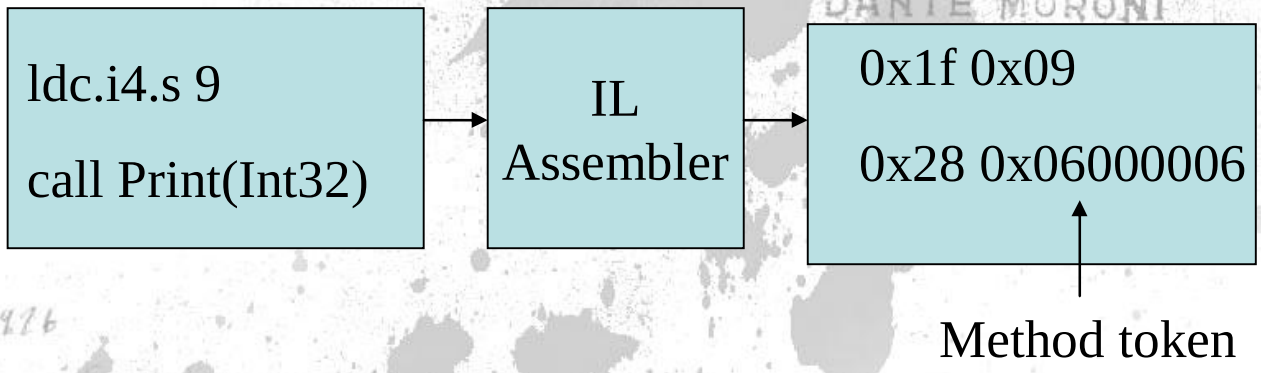
- 完全基于堆栈的模型
- 可静态验证有效性
- 使用1-2字节操作码
- 0xFE为2字节操作码的前导标记字节
- 内建面向对象指令
- 使用Token来替代对指针或偏移量的使用



1.3.2.IL 指令分类

- 流程控制, br, brtrue, beq, leave, endfinally, ret
- 运算指令, add, shl, ldc.i4, conv.i8, add.ovf
- 参数和局部变量, ldarg.0, starg, ldloc , stloc
- 字段访问, ldfld, ldsfld, stfld
- 方法调用, call, callvirt, ldftn, ldvirtftn, calli, jmp
- 引用和值类型, newobj, ldobj, ldstr, isinst, box
- 数组操作, newarr, ldlen, ldelem.r4, stelem.ref

1.3.3.使用Token定位



Token

1.3.4. 函数调用堆栈

```
ClassType a;
a.func(1, 2)
```

```
ldc.i4.1
ldc.i4.2
call ClassType::func(Int32, Int32)
```

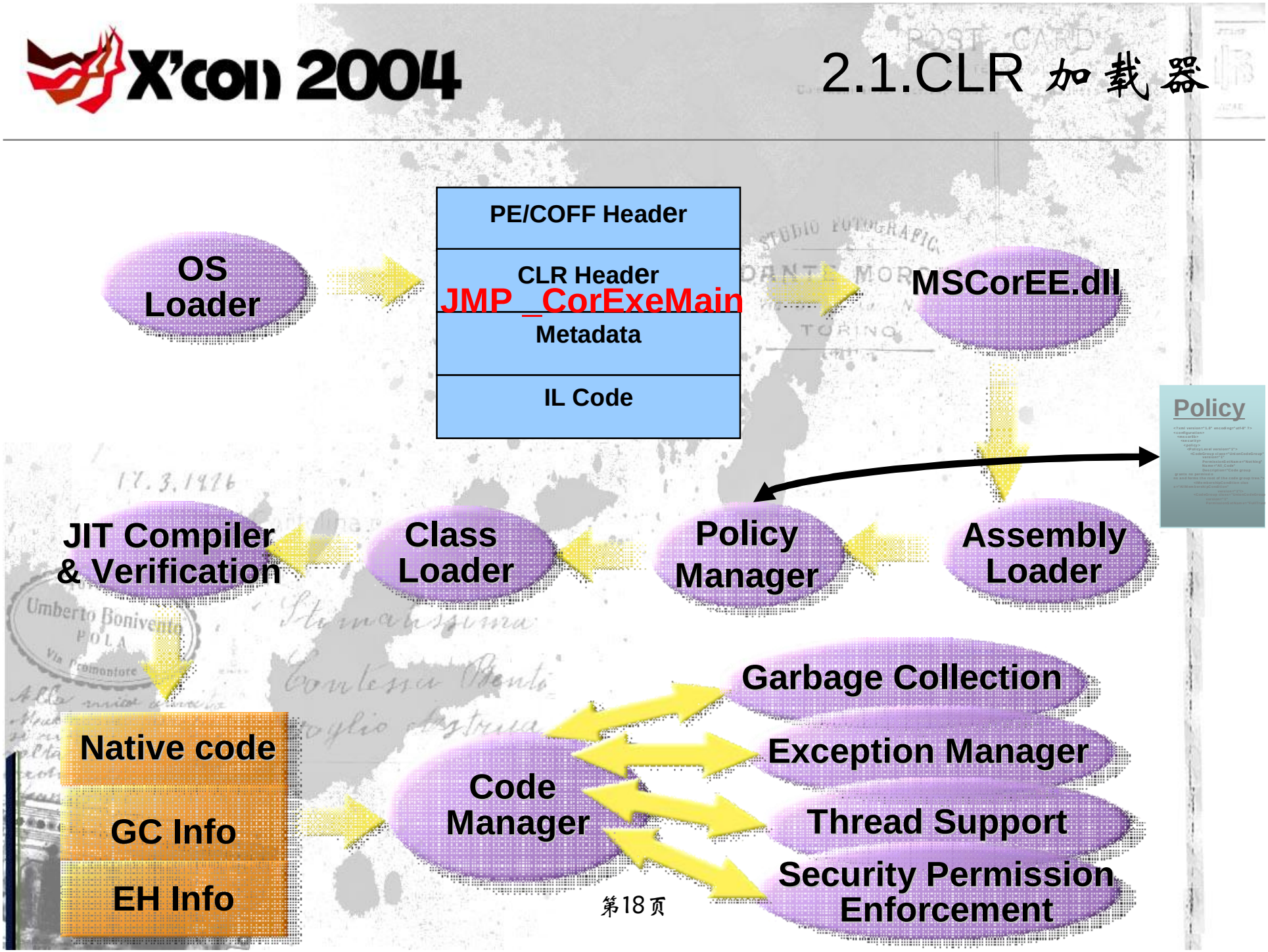
1
2
this pointer

Stack top

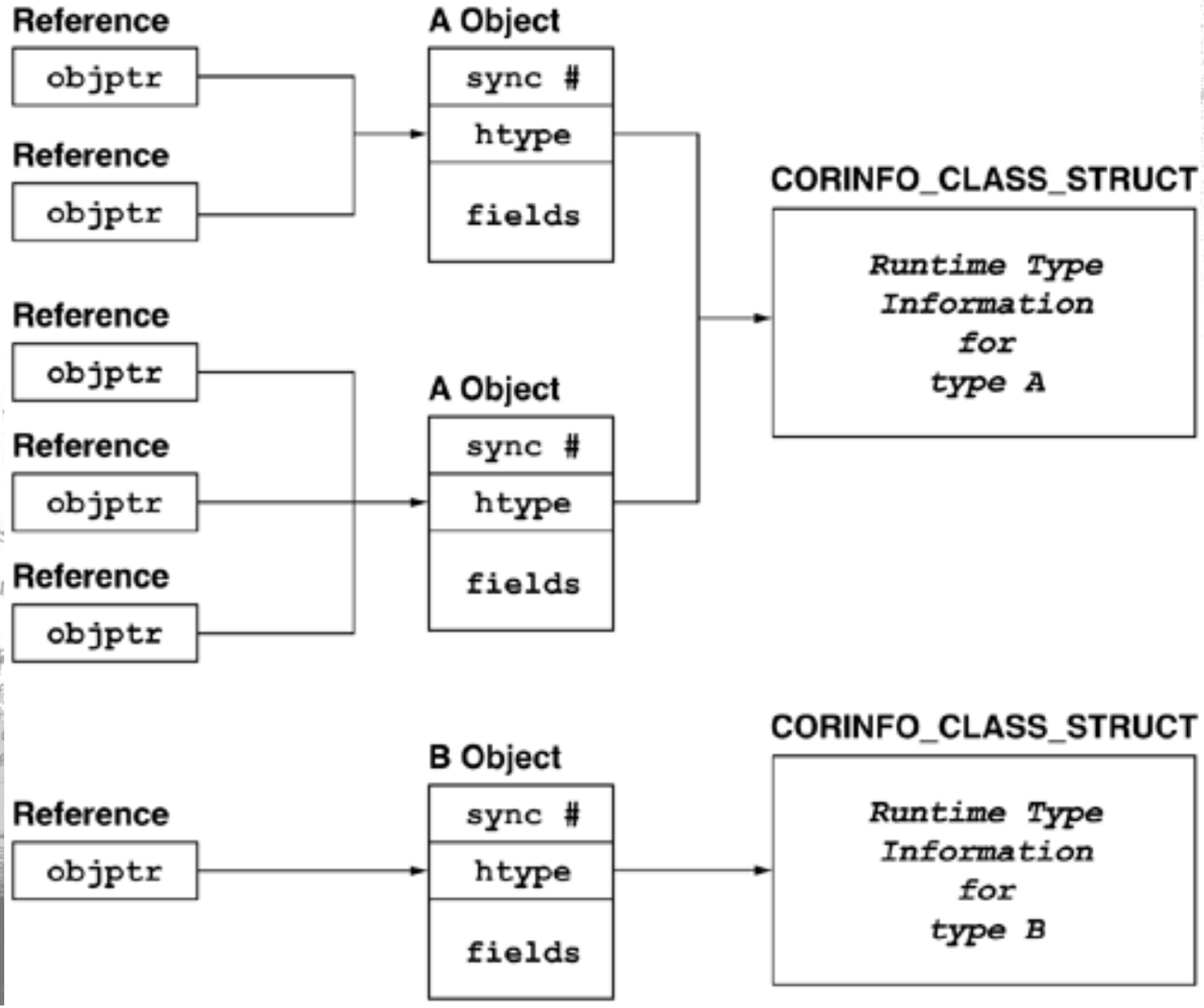
2.动态执行

- CLR 加载器 (fusion)
- 类型与对象的内存布局
- JIT 编译与方法调用
- 堆栈、帧与堆栈遍历
- 非托管代码互操作

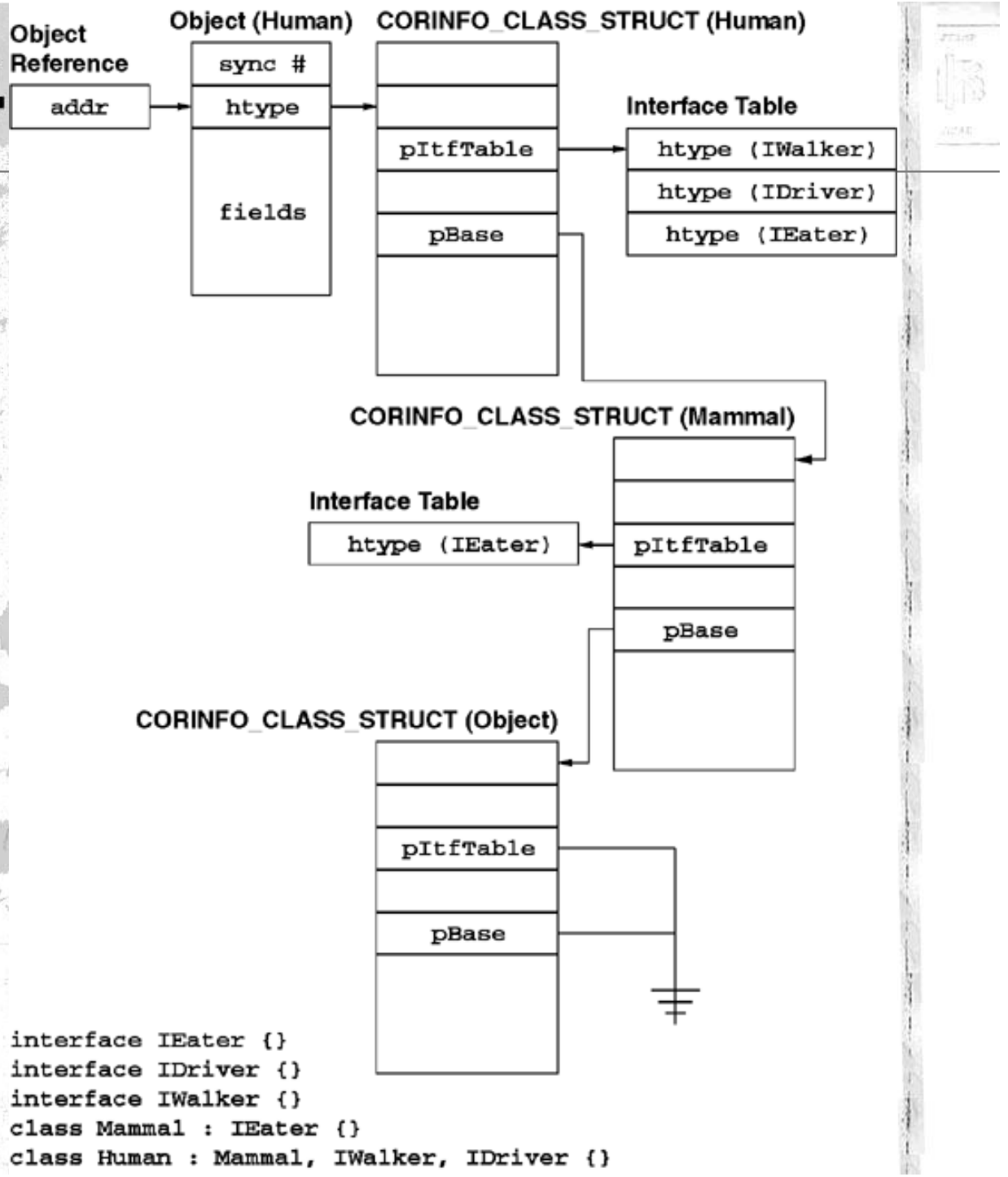
2.1.CLR 加载器


OS Loader
PE/COFF Header
CLR Header
JMP _CorExeMain
Metadata
IL Code
MSCorEE.dll
Policy
JIT Compiler & Verification
Class Loader
Policy Manager
Assembly Loader
Native code
GC Info
EH Info
Code Manager
Garbage Collection
Exception Manager
Thread Support
Security Permission Enforcement

2.2.类型与对象的内存布局



2.2.2. 类型运行时层次结构

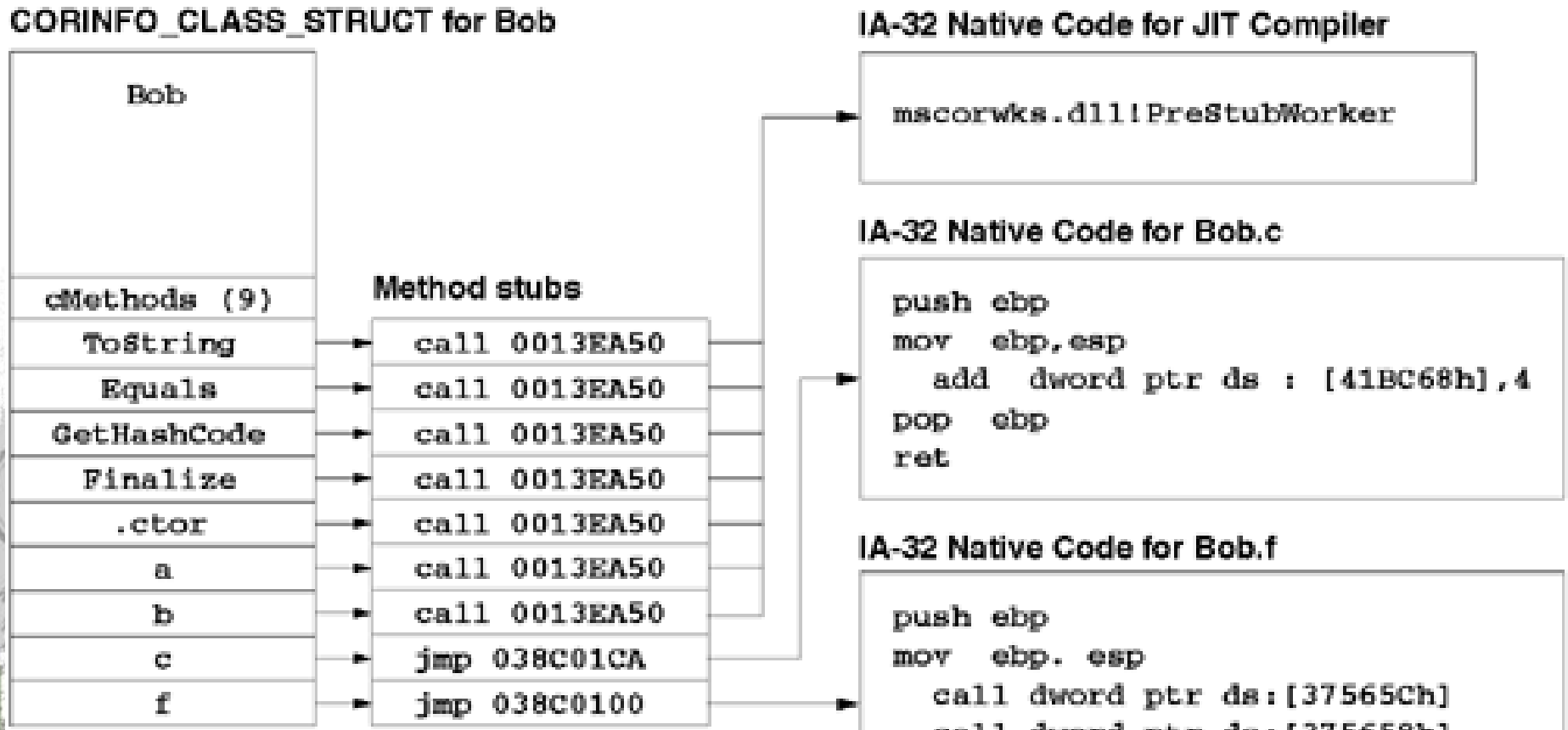



```
interface IEater {}
interface IDriver {}
interface IWalker {}
class Mammal : IEater {}
class Human : Mammal, IWalker, IDriver {}
```

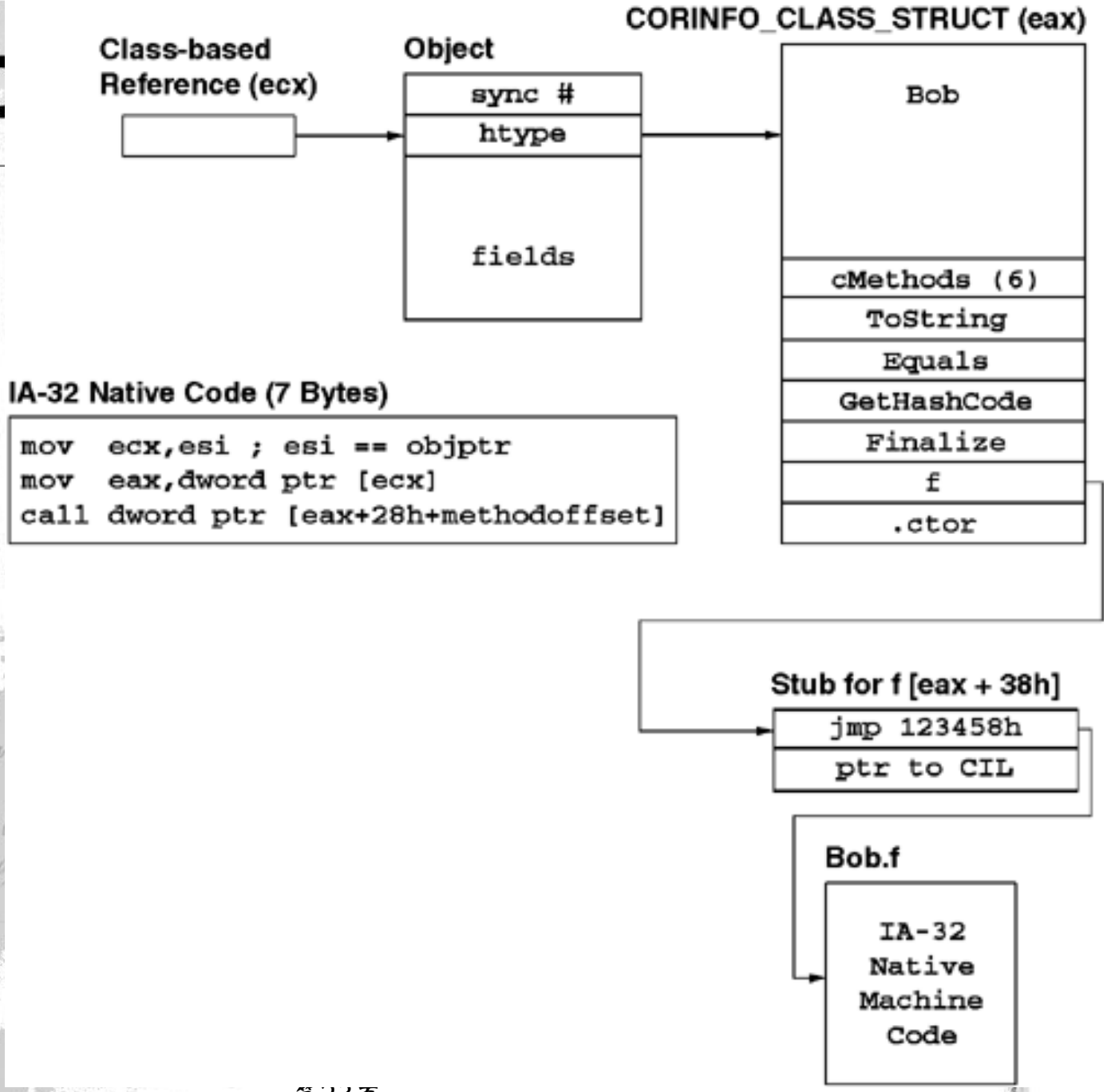
2.3.JIT编译与方法调用

- 2.3.1.JIT 编译与方法表
- 2.3.2.基于类引用的虚函数调用
- 2.3.3.基于接口引用的虚函数调用
- 2.3.4.异步方法调用
- 2.3.5.内部方法调用

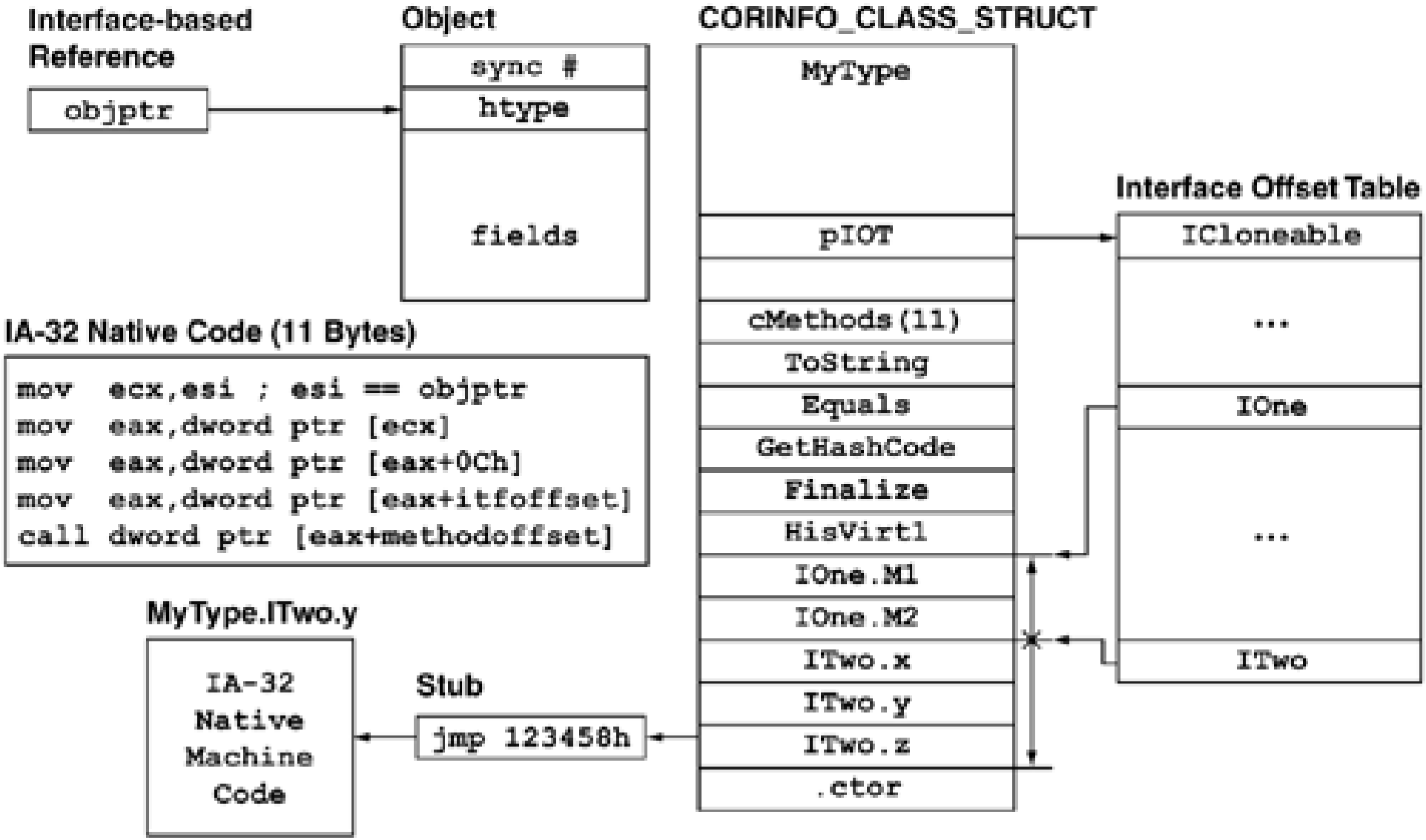
2.3.1.JIT 编译与方法表



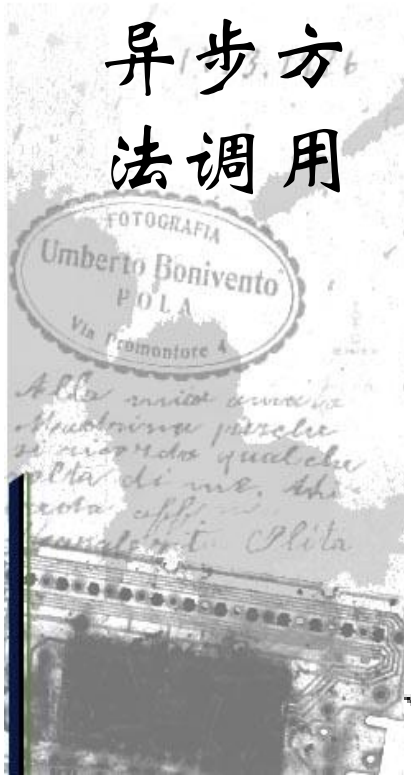
2.3.2. 基于类引用的虚函数调用

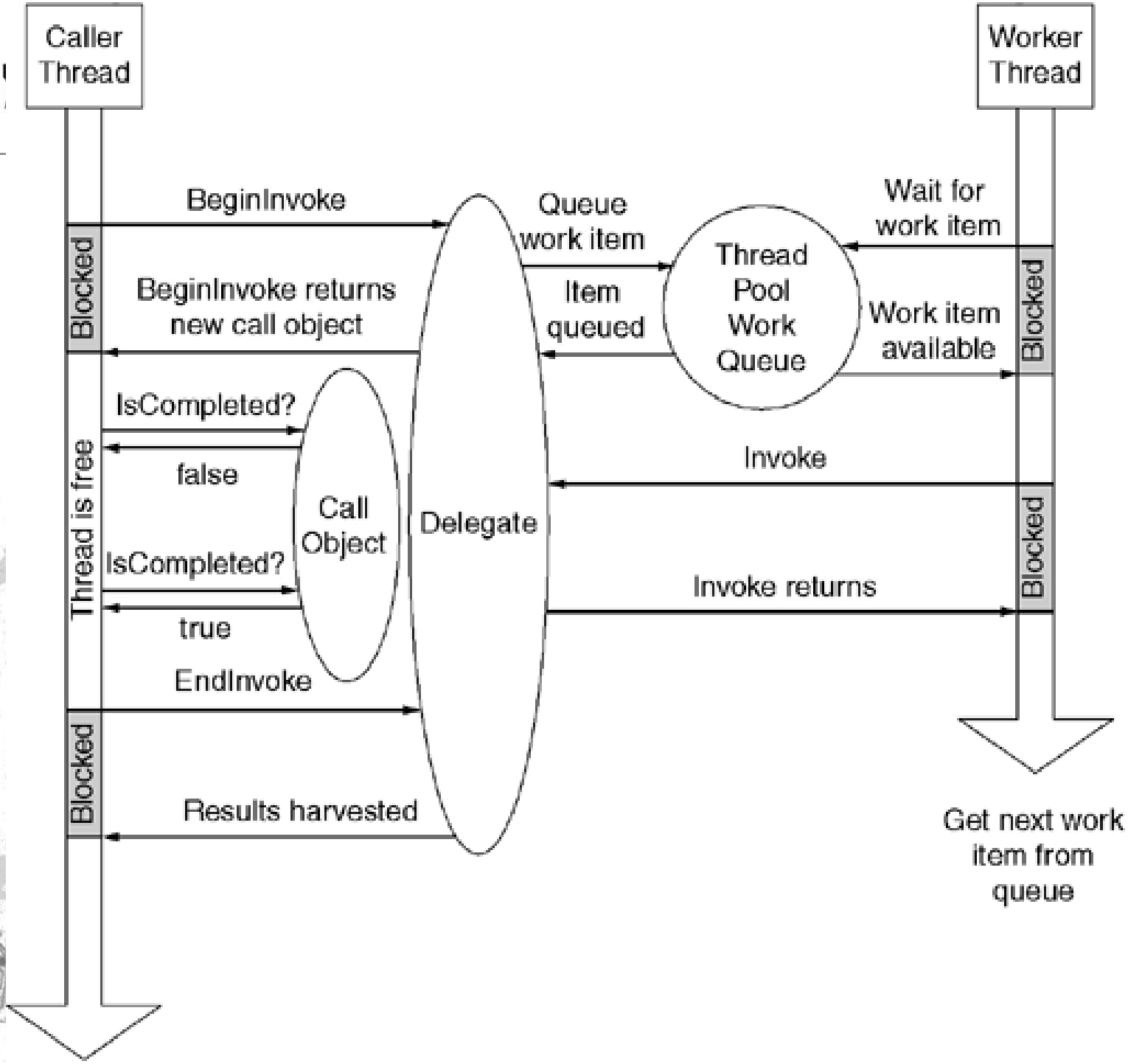


2.3.3. 基于接口引用的虚函数调用



2.3.4. 异步方法调用


FOTOGRAFIA
Umberto Bonivento
POLA


Caller Thread
Worker Thread
BeginInvoke
Queue work item
Wait for work item
Blocked
BeginInvoke returns new call object
Item queued
Thread Pool Work Queue
Work item available
Blocked
IsCompleted?
false
Thread is free
Call Object
Delegate
Invoke
Blocked
IsCompleted?
Invoke returns
true
EndInvoke
Blocked
Results harvested
Get next work item from queue

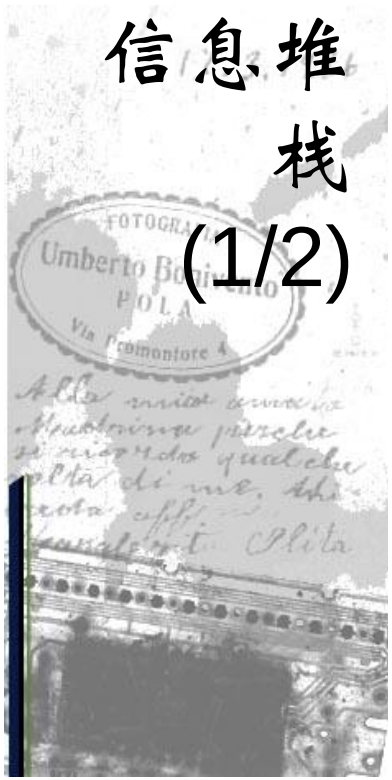
2.3.5.内部方法调用

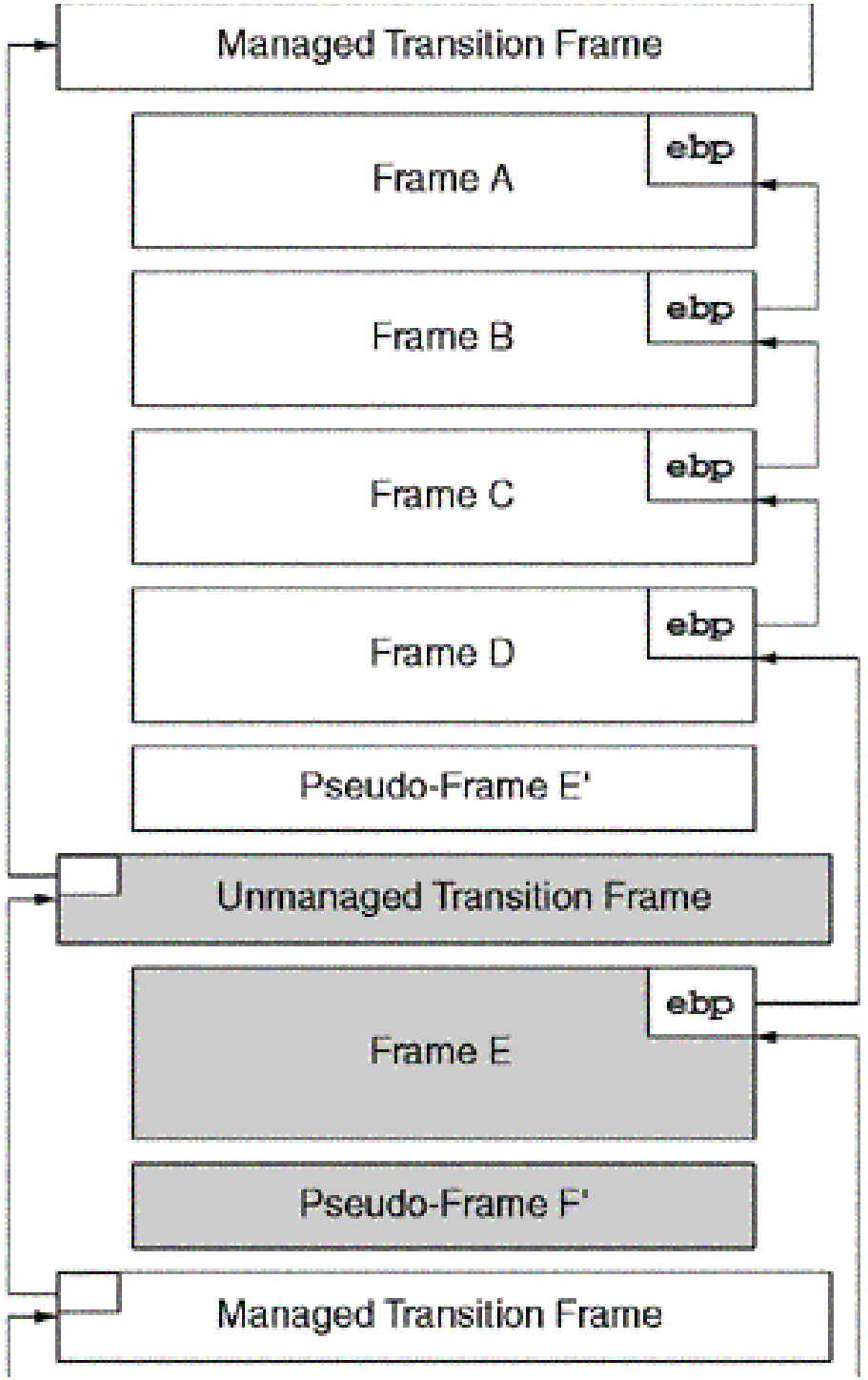
2.4.堆栈、帧与堆栈遍历

- 2.4.1.基于帧的信息堆栈
- 2.4.2.常用帧的继承结构
- 2.4.3.堆栈遍历原理与引用
- 2.4.4.堆栈遍历的实现流程

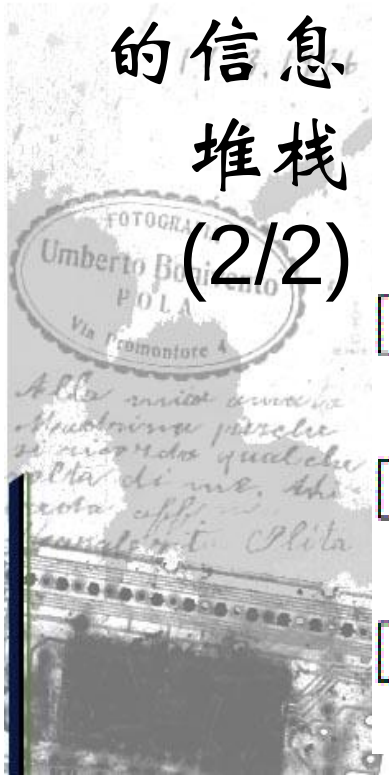
2.4.1. 基于帧的信息堆栈 (1/2)

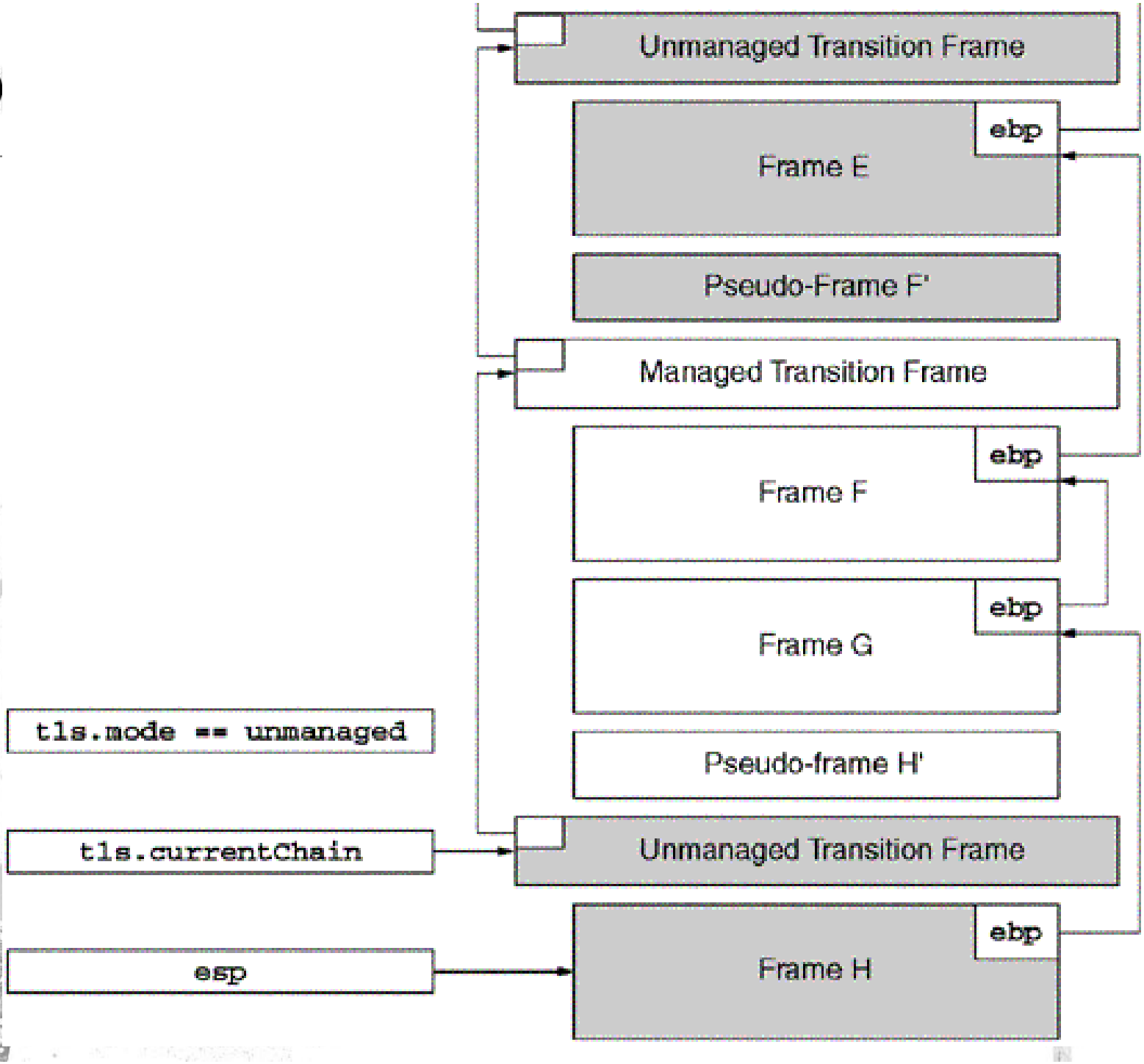
```
#pragma unmanaged
void H(){}
#pragma managed
void G() { H(); }
void F() { G(); }
#pragma unmanaged
void E() { F(); }
#pragma managed
void D() { E(); }
void C() { D(); }
void B() { C(); }
void A() { B(); }
```



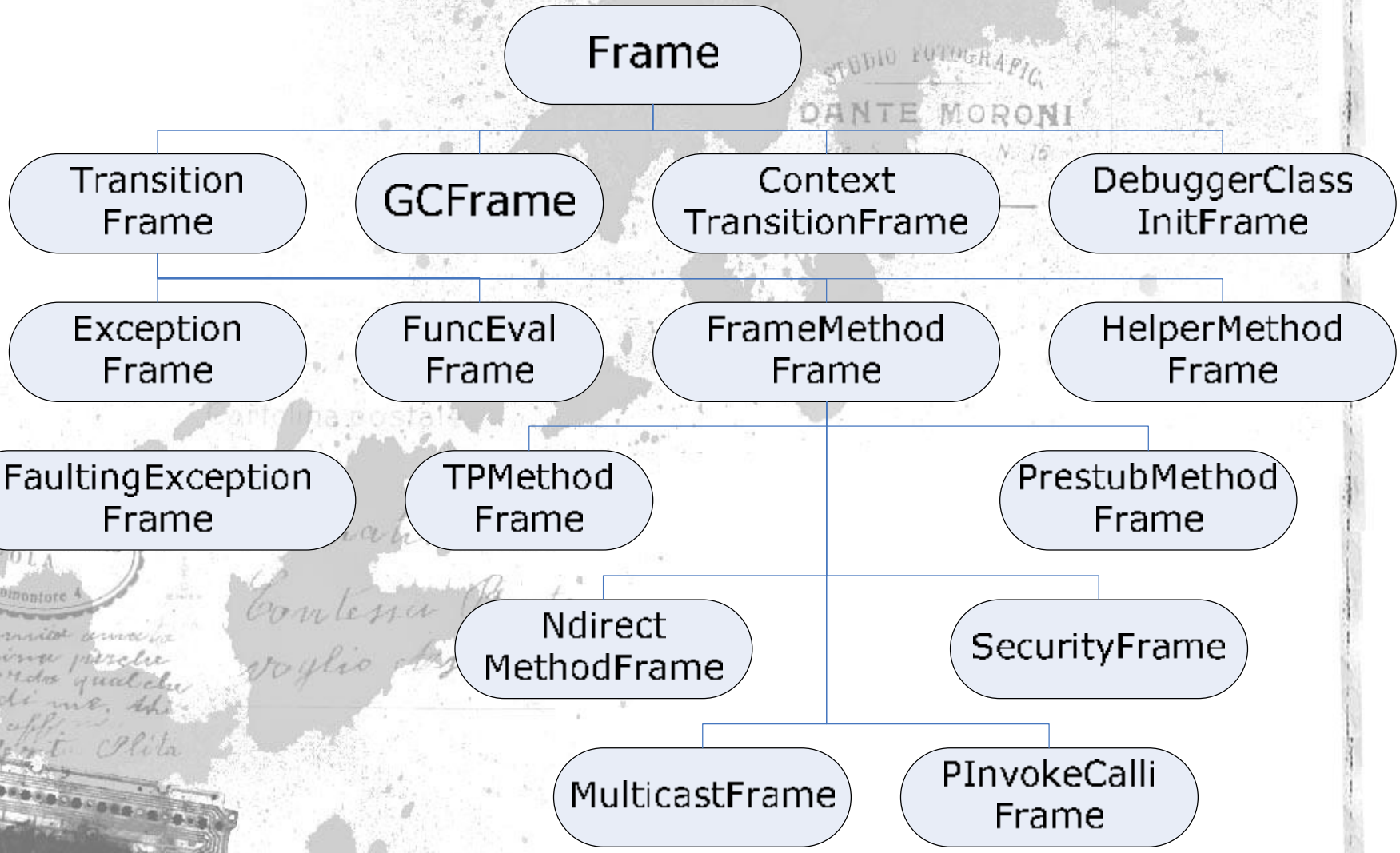


2.4.1. 基于帧的信息堆栈 (2/2)



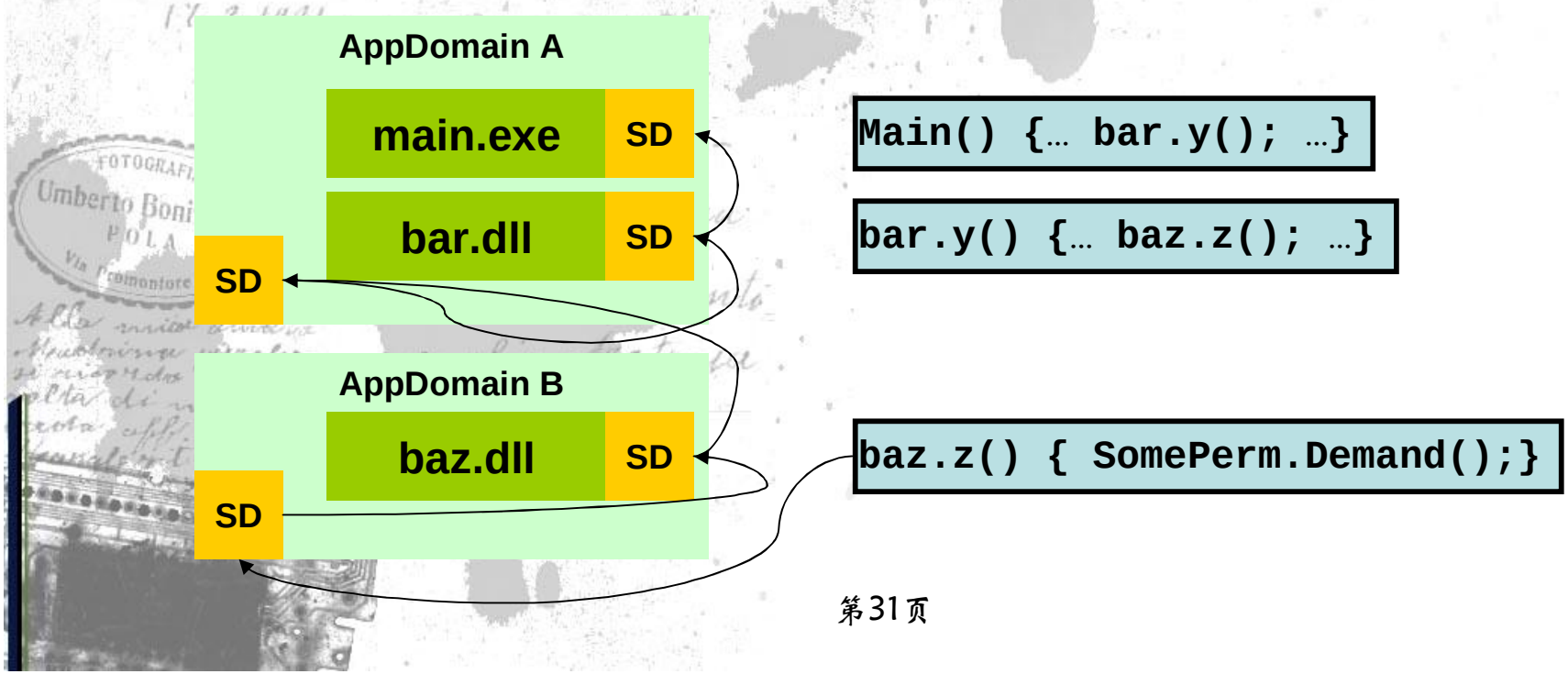


2.4.2. 常用帧的继承结构

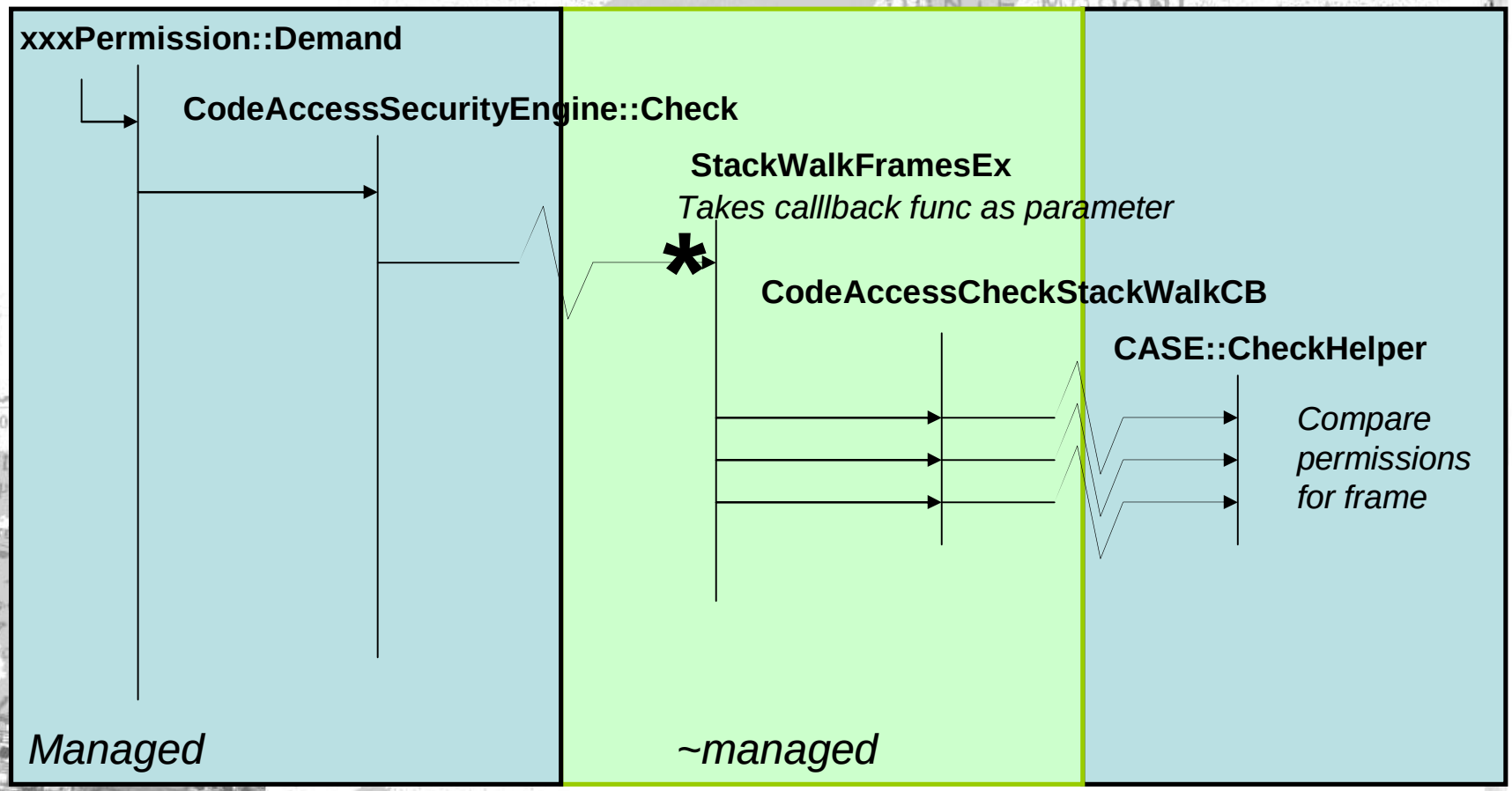


2.4.3.堆栈遍历原理与引用

- 基于帧的逆向堆栈遍历算法
- 主要用于安全框架实现和GC操作
- 通过Unmanaged代码实现堆栈遍历



2.4.4.堆栈遍历的实现流程



2.5.非托管代码互操作

3.安全框架

- 代码访问安全(Code-Access Security)
- 角色安全检查(Role-Based Security)
- 独立存储区(Isolated Storage)
- 密码学服务(Cryptographic Services)

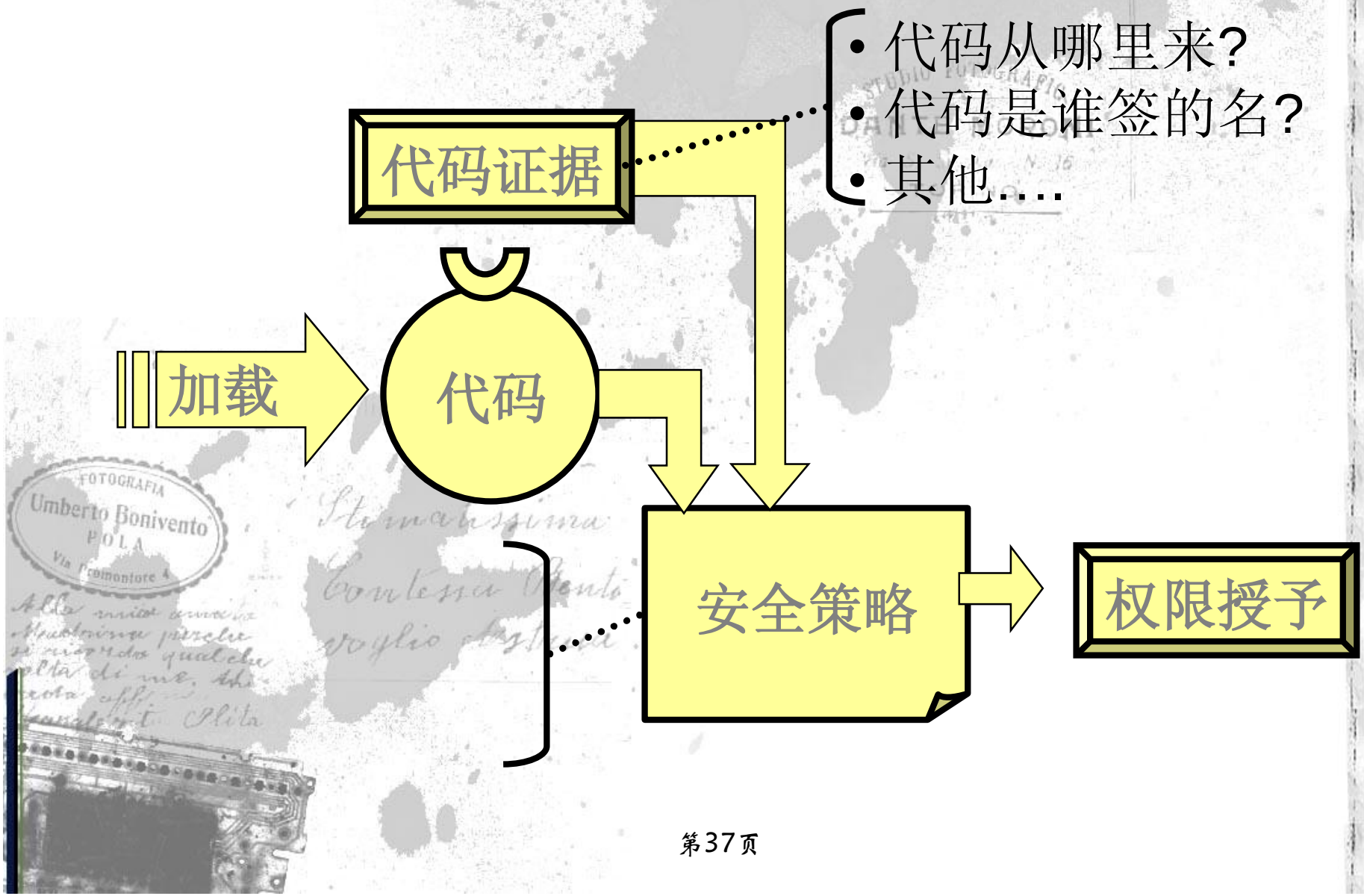
3.1.代码访问安全

- 安全策略(Security Policy)
- 代码凭据(Code Evidence)
- 权限(Permission)

3.1.2.控制代码访问资源的机制

- 定义权限
- 安全策略管理
- 允许代码要求拥有权限
- 授予代码权限
- 允许代码请求其调用者具备特定权限
- 动态限制代码安全 (Stackwalking)

3.1.3.验证过程



3.1.4.安全策略

- 安全策略级别
  - 企业范围
  - 本机范围
  - 用户范围
  - Application Domain 范围(不可配置)
- 通过 caspol.exe 或 mscorcfg.msc 配置
- 有效的安全策略是所有级别的交集

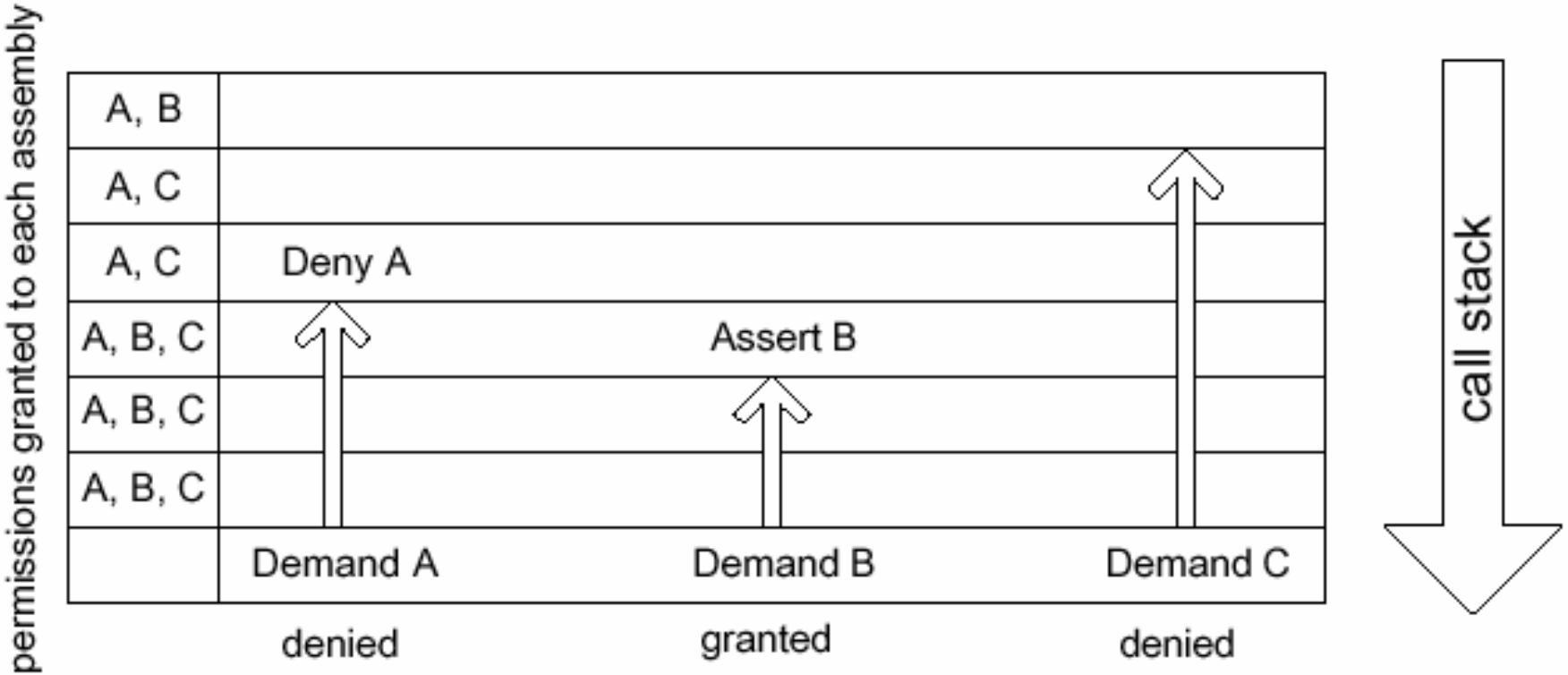
3.1.5.代码凭据

- 预定义代码凭据
  - Zone
  - Site
  - Url
  - Publisher
  - StrongName
  - …
- 代码凭据可扩展

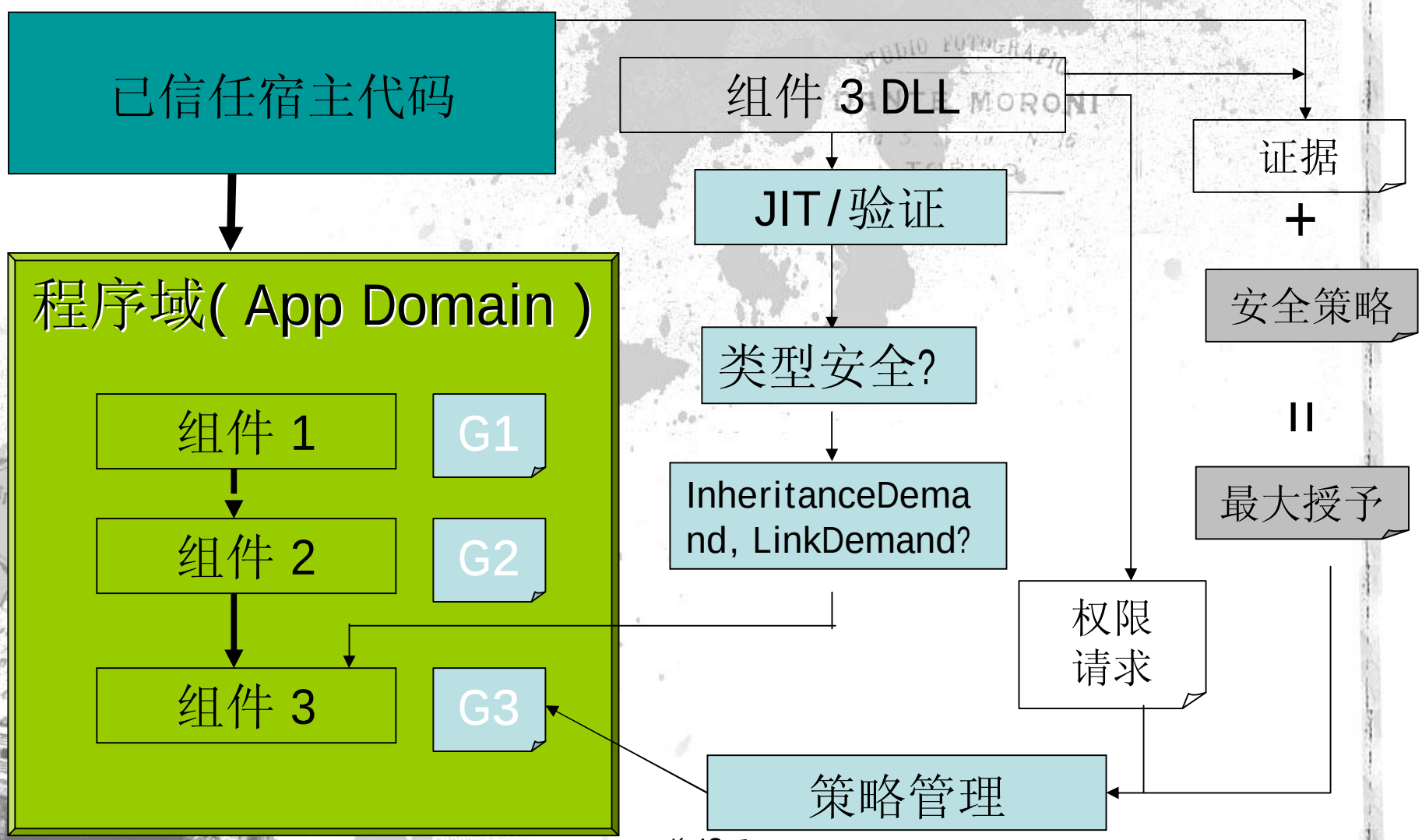
3.1.6.安全权限

- .NET 安全权限类定义
  - CodeAccessPermission
  - DBDataPermission
  - PrintingPermission
  - DnsPermission
  - WebPermission
  - EnvironmentPermission
  - FileIOPermission
  - RegistryPermission
  - UIPermission
  - …
- 自定义安全权限

3.1.7.安全权限验证过程



3.1.8.安全权限授予过程



3.2. 角色安全检查

- 基于角色的预定义类
  - GenericPrincipal
  - GenericIdentity
  - WindowsPrincipal
  - WindowsIdentity
  - 自定义类
  - PrincipalPermission

3.2.1.相关接口

- IPrincipal interface

```
namespace System.Security.Principal {
        public interface IPrincipal {
            IIdentity Identity { get; }
            bool      IsInRole( String role );
        }
}
```

- IIdentity interface

```
namespace System.Security.Principal {
        public interface IIdentity {
            String AuthenticationType { get; }
            bool   IsAuthenticated    { get; }
            String Name               { get; }
        }
}
```

3.2.2.静态安全检查

```
[ PrincipalPermissionAttribute( SecurityAction.Demand,
                                Name = “MyUser”,
                                Role = “User” ) ]
public static void PrivateInfo()
{
        // 打印保密数据
        Console.WriteLine( “\n\nYou have access to the private
data!” );
}
```

3.2.3.动态安全检查

```
String id1 = "Bob";
  String role1 = "Manager";
  PrincipalPermission PrincipalPerm1 =
                    new PrincipalPermission(id1, role1);
String id2 = "Louise";
  String role2 = "Supervisor";
  PrincipalPermission PrincipalPerm2 =
                  new PrincipalPermission(id2, role2);
 (PrincipalPerm1.Union(PrincipalPerm2)).Demand();
```

3.3.独立存储区

3.4.密码学服务

4.相关资源

Thanks !