Ambientes Virtuais de Execução

Máquinas Virtuais

Máquina Virtual

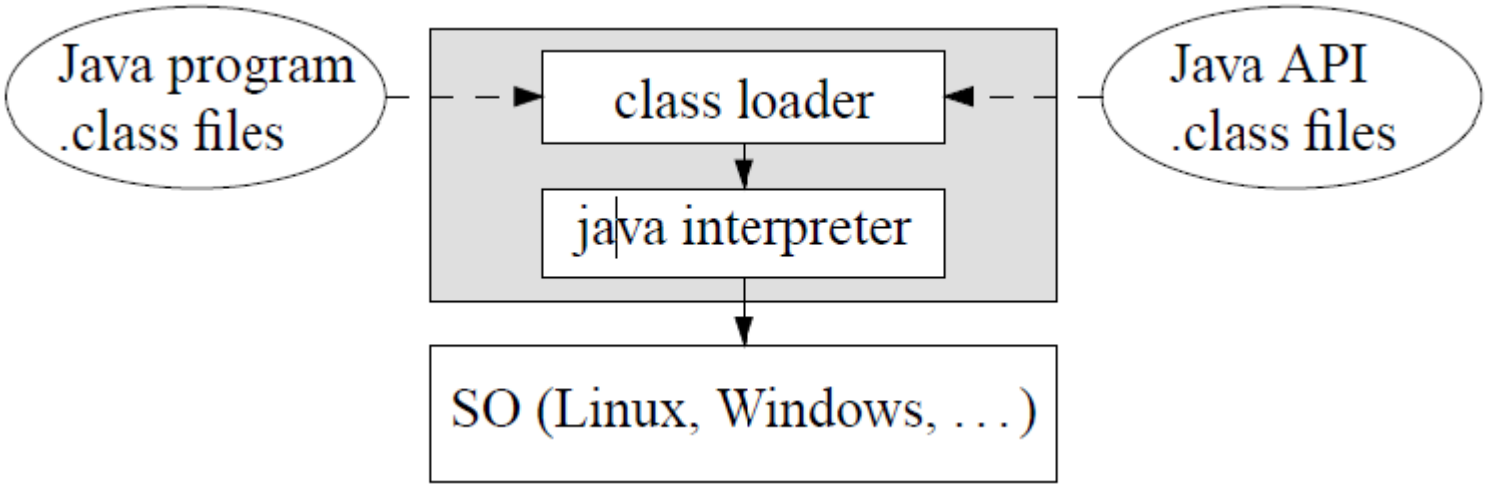
- máquina virtual é o nome dado a uma máquina, implementada através de *software*, que executa programas como um computador real.
- Existem dois tipos de máquinas virtuais:
  - de sistema, que fornece uma plataforma completa de um sistema, que suporta a execução de um sistema operativo completo;
  - de aplicação, que corre como uma aplicação normal dentro de um sistema operativo e que suporta um único processo.
- o software que corre numa máquina virtual está limitado aos recursos e abstracções fornecidas pela máquina virtual.

Máquinas Virtuais de Aplicação

- Exemplos de máquinas virtuais de aplicação
  - JVM (Java Virtual Machine)
  - CLR (Common Language Runtime): implementação do CLI, máquina virtual da plataforma .NET
    - Também classificada de *Execution Engine*
  - Mono: implementação do CLI, *open source*
  - Dalvik (Virtual Machine do OS Android)

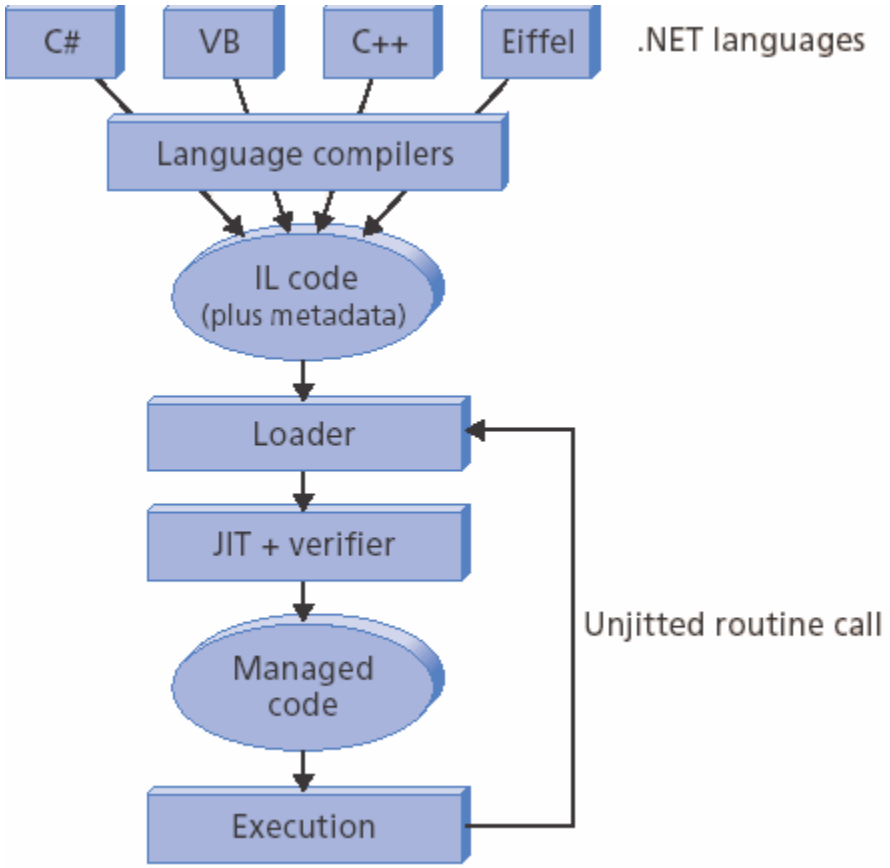
Exemplo: Java Virtual Machine

- Aplicações fazem uso da Java API, ignorando o hardware.
- Compilador de Java (ou outra linguagem) gera java bytecode
- Verificador de bytecode assegura certos invariantes:
  - permite protecção no acesso à memória sem ajuda do hardware
  - útil para hardware restrito
- Runtime executa bytecode com interpretador ou compilador JIT (just-in-time)



Exemplo: Common Language Runtime (CLR)

- CLR é a implementação da Microsoft do standard Common Language Infrastructure (CLI), que define o ambiente de execução
- Em CLR, o código é expresso numa forma de bytecode, designada por Common Intermediate Language (CIL).



Comparação da JVM e CLR

	CLR	JVM
Modelo da máquina	stack	stack
Gestão de memória	automático ou manual	automático
Segurança do código	sim	sim
Interpretador	não	sim
JIT	sim	sim
Pré-compilação	sim	não
Bibliotecas partilhadas	sim	sim
Common Language Object Model	sim	sim
Dynamic Typing	sim	não

Ambientes Virtuais de Execução

Programação orientada ao Componente (POC)

Programação Orientada ao Componente

- Um componente é uma unidade independente de *software* que encapsula um conjunto de funções relacionadas (ou dados).
- A Programação orientada ao Componente (COP) acenta no princípio de que o desenvolvimento do software se deverá realizar recorrendo à composição de componentes independentes, já desenvolvidas.
- COP começou a surgir para responder a algumas limitações do programação orientada por objectos (OOP):
  - herda (alguns) princípios da programação orientada por objectos;
  - as componentes são unidades de extensão;
  - reutilização a uma escala distribuída.

Diferenças entre COP e OOP

- Uma aplicação
  - orientada ao componente reúne um conjunto de modulos de aplicação binários, que interagem entre si;
  - orientada ao objecto resulta em geral numa unidade de código binário monolítico.
- A reutilização em aplicações
  - orientadas ao componente é do tipo *black box*, isto é, pode ser utilizada uma componente já existente, tendo apenas que cumprir com um conjutno de operações pré-definidas ou interfaces
  - orientadas ao objecto é do tipo *white box*, isto é, é necessário estar familiar com alguns detalhes de implementação.

Princípios da Programação Orientada ao Componente

- Separação entre a interface e a implementação
- Compatibilidade binária
- Independência da linguagem
- Transparência da localização
- Gestão da concorrência
- Controlo de versões
- Segurança Baseada em Componentes

Componentes

- Exemplos:
  - JavaBeans e enterprise Java Beans (EJB)
  - COM (Component Object Model),
  - DCOM (Distributed Component Object Model)
  - .NET components