Ambientes Virtuais de Execução

(2ºS 2009/2010)

1ª Aula Prática

1. Considere o ficheiro presente na directoria **Questao1** do *zip* anexo a este enunciado. Compile o ficheiro `Hello.cs`. Analise, recorrendo à ferramenta *ildasm*, o IL produzido.

2. Considere os ficheiros presentes na directoria **Questao2** do *zip* anexo a este enunciado. Compile o ficheiro `RUT.cs` para um módulo .Net, através da opção `/t:module`. De seguida, produza FUT.dll como módulo primário de um assembly multi-módulo, utilizando as opções /`addmodule (/addmodule:RUT.netmodule)` e `/t:library`.

3. Considere os ficheiros presentes na pasta **Questao3** do *zip* anexo a este enunciado.

a) Quantos *assemblies* são definidos por estes ficheiros? Descreva a estrutura de módulos de cada *assembly*.

b) Quais os tipos definidos e quais os tipos referenciados pelos *assemblies* definidos? Descreva em diagrama de classes UML os tipos existentes nos *assemblies* definidos.

c) Descreva as acções realizadas por cada um dos métodos presentes nos tipos definidos por estes ficheiros. Valide a sua análise executando o componente com método `Main`.

d) Recorrendo às ferramentas *ildasm* e *ilasm*, altere os nomes dos métodos dos tipos `C1` e `C3` de modo a reflectirem a sua funcionalidade. Teste as alterações.

4. Considere o assembly presente na pasta **Questao2** do *zip* anexo a este enunciado.

e) Admitindo que o tipo **RarelyUsed** tem uma utilização infrequente, tirando partido das ferramentas `ildasm` e `ilasm` modifique o *assembly* de modo a que o tipo **RarelyUsed** seja colocado num segundo módulo constituinte do *assembly*.

f) Altere agora os ficheiros em linguagem CIL de forma a que o tipo **RarelyUsed** seja disponibilizado em *assembly* distinto.

5. Seja o seguinte programa em CIL:

```
.assembly extern mscorlib {
  .publickeytoken = (B7 7A 5C 56 19 34 E0 89 )
  .ver 2:0:0:0
}

.assembly what {
  .ver 1:0:0:0
}

.namespace Utils {
  .class public auto Globals {

       .method public  static void recursiveOperation(int32 n) cil managed {
             .maxstack  2

             ldarg.0
             ldc.i4.s   10
             bge.s      label1
             ldarg.0
             call       void [mscorlib]System.Console::Write(int32)
             br.s       label2

          label1:
             ldarg.0
             ldc.i4.s   10
             div
             call       void  Utils.Globals::recursiveOperation (int32)
             ldarg.0
             ldc.i4.s   10
             rem
             ldc.i4.s      10
             call       void [mscorlib]System.Console::Write(char)
             call       void [mscorlib]System.Console::Write(int32)
          label2:
             ret
       } // end of method


  }
}
```

a) Indique o que faz o método `recursiveOperation`.

b) Faça um programa em C# que teste o método `recursiveOperation` da classe `Utils.Globals`. A classe `Globals` e a classe realizada em C# deverão ficar em *assemblies* diferentes. Utilize apenas ferramentas de linha de comandos para compilar o código fonte.

c) Acrescente ao tipo `Utils.Globals` o método `int countDigits(int v)` que retorna o número de dígitos de `v`.