Universidade Federal de Uberlândia
Ciência da Computaçõa
Construção de Compiladores

Construção do mini compilador Python para CLR

Aluno: Arthur Xavier Giffoni
E-mail: arthurxaviergiffoni@gmail.com
Professor:Alexsandro Santos Soares

Abril, 2017

Sumário

1 Introdução

Common Language Runtime - CLR, é um ambiente de tempo de execução do .NET Framework, o qual que executa o código e provê serviços que tornam o processo de desenvolvimento mais fácil.Compiladores e ferramentas expõem as funcionalidades do Common Language Runtime e habilitam você escrever código que se beneficia desse ambiente de execução gerenciado. Código que você desenvolve com um compilador de linguagem que tem como alvo o runtime é chamado de código gerenciado; ele se beneficia de recursos como integração entre linguagens, tratamento de exceção entre linguagens, segurança aprimorada, suporte a versionamento e implantação, um modelo simplificado para interação entre componentes, e serviços de depuração e de perfil.

O CLR descreve o código executável e o ambiente de execução que formam o núcleo do Framework .NET da Microsoft, do MONO e do Portable.NET. A Common Language Runtime é uma máquina virtual que segue um padrão internacional e a base para a criação e execução de ambientes de desenvolvimento em que as linguagens e as bibliotecas trabalham juntos.

Programas em linguagem Visual Basic, Visual C ++ ou C # são compilados em um formulário intermediário de codificar chamado Common Intermediate Language (CIL) em um arquivo de execução portátil (PE) que pode, em seguida, ser gerenciado e executado pelo Common Language Runtime.

2 Arquitetura

O common language runtime (CLR) API de depuração foi projetado para ser usado como se fosse parte do kernel do sistema operacional. Em código não gerenciado, quando um programa gera uma exceção, o kernel suspende a execução do processo e passa as informações de exceção para o depurador usando o Win32 API de depuração. API de depuração CLR fornece a mesma funcionalidade para código gerenciado. Quando o código gerenciado gera uma exceção, a API de depuração CLR suspende a execução do processo e passa as informações de exceção para o depurador.

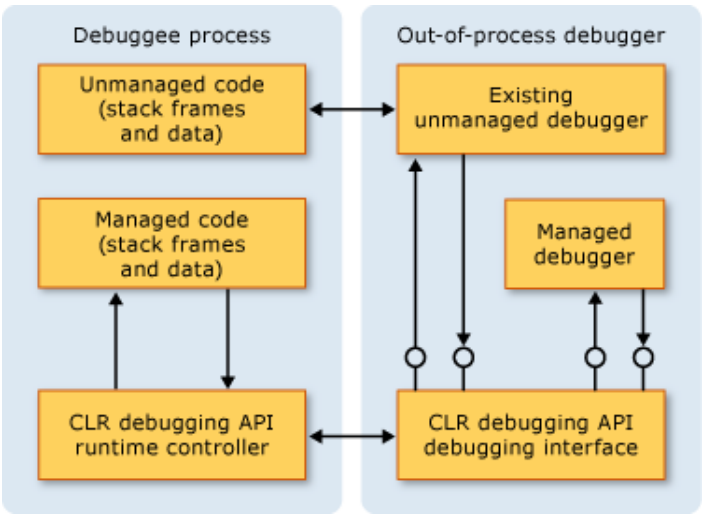
API de depuração CLR inclui dois componentes principais:

- A DLL de depuração, que é sempre carregada no mesmo processo como o programa que está sendo depurado. O controlador de tempo de execução é responsável pela comunicação com o CLR e executar o controle de execução e inspeção de threads que estejam executando o código gerenciado.

- A interface do depurador, que é carregada em um processo diferente do programa que está sendo depurado. A interface do depurador é responsável pela comunicação com o controlador de tempo de execução em nome do depurador. Também é responsável pela manipulação de eventos de depuração do Win32 que vêm do processo que está sendo depurado e ambos lidando com esses eventos ou passá-los para um depurador de código não

gerenciado. A interface do depurador é a única parte da API que tem uma API exposta de depuração CLR.

O CLR API de depuração não oferece suporte ao uso remoto entre computadores ou entre processos; ou seja, um depurador que usa a API deverá fazê-lo de dentro de seu próprio processo, conforme a figura no seguinte diagrama de arquitetura da API. Esta ilustração mostra onde os diferentes componentes do CLR API de depuração estão localizados e como eles interagem com o CLR e o depurador.



3 Instruções

O assembly da plataforma, MSIL, é um conjunto de instruções baseado em pilha e orientado a objetos. As funcionalidade de algumas intruções em relação a manipulação da pilha listadas a seguir:

- add: retira os dois elementos do topo da pilha e coloca o resultado no topo.
- add.ovf: mesma operação que o add porém com uma verificação de over- flow(gerando uma exceção de overflow).
- and: retira os dois elementos do topo da pilha, faz a operação and bit a bit e coloca o resultado no topo da pilha.
- arglist: pega argumento da lista (usado em para pegar argumentos de função)
- beq.[length]: branch para label se o dois valores da pilha são iguais
- bge.[length]: branch para label se o primeiro valor é menor que o se- gundo da pilha

- bge.un.[length]: branch para label se maior ou igual, comparação sem sinal
- bgt.[length]: branch para label quando o segundo valor é maior que o primeiro
- bgt.un.[length]: branch para label quando o topo é maior que o segundo valor, sem sinal
- ble.[length]: branch para label quando o topo é menor ou igual ao segundo valor
- ble.un.[length]: branch para label se topo é menor ou igual ao segundo valor, sem sinal
- blt.[length]: branch para label quando quando o topo é menor que o segundo valor
- blt.un.[length]: branch para label se topo é menor que segundo valor, sem sinal
- bne.un[length]: branch para label quando topo não for igual ou não ordenada
- br.[length]: branch incondicional
- break: instrução breakpoint
- brfalse.[length]: branch para label se falso, nulo, ou zero
- brtrue.[length]: branch para label quando não falseo ou não nulo
- call: chama um método
- calli: chama um método indireto
- ceq: compara se igual
- cgt: compara se maior que
- cgt.un: compara maior que,sem sinal e não ordenado
- ckfinite: Checa se é um número real e finito
- clt: compara se menor que
- clt.un: compara se menor que, sem sinal e não ordenado
- conv.[to type]: conversão de dados
- conv.ovf.[to type]: conversão de dados com detecção de overflow
- conv.ovf.[to type].un: conversão de dados sem sinal com detecção de overflow

- cpblk: copia dados da memória para a memória
- div: divide valores
- div.un: divide valores inteiros, sem sinal
- dup: duplica o valor do topo da pilha
- endfilter: fim do filtro da cláusula de SEH
- endfinally: finaliza a cláusula do bloco de exceção
- initblk: inciliaza um bloco de memória para um valores
- jmp: pula para um método
- jmpi: pula para um ponteiro de método
- ldarg.[length]: carrega um argumento na pilha
- ldarga.[length]: carrega um argumento a partir de um endereço
- ldc.[type]: carrega uma constante numérica
- ldftn : carrega um ponteiro de método
- ldind.[type]: carrega um valor indireto na pilha
- ldloc: load local variable onto the stack
- ldloc: carrega na pilha o valor da variavel local.
- ldloca index: carrega na pilha o valor da variavel local com index.
- ldnull: carrega na pilha um ponteiro pra null
- leave target: sai de uma região protegida do códdigo
- localloc: aloca um espaço na memory pool do tamanho do primeiro elemento dap ilha e retorna o endereço da área.
- mul: multiplica os dois valores do topo da pilha(retirando-os) e coloca o resultado.
- mul.ovf.[type]: multiplica valores inteiros levando em conta o overflow.
- neg: retira o valor do topo da pilha, nega ele e põe o resultado no topo, retornando o mesmo tipo de operando.
- nop: faz nada :D
- not: retira um inteiro e coloca seu complemento(inverção de bits) na pilha.
- or: faz o OU bit a bit dos dois valores inteiros no topo da pilha(retirando-os) e coloca o resultado.

- pop: remove o elemento do topo da pilha.
- rem: computa o resto da divisão do valor abaixo do topo da pilha pelo valor que está no topo(retirando-os) e coloca o resultado no topo.
- rem.un: mesmo que o rem só que para inteiros unsigned.
- ret: retorna para o método corrente, o tipo de retorno do método em questão será o utilizado.
- shl: Deslocamento de inteiro para a esquerda..., valor, qntd d e d eslocamentoretiraestesdoisdapilhaecolocaoresultado.
- shr: mesmo que o shl porém com deslocamneto para a direita.
- shr.un: memso que shr só que para inteiros unsigned.
- starg.[length]: retira o elemento do topo da pilha e o coloca em um argumento(starg num).
- stind.[type]: coloca o valor(topo da pilha) no endereço (logo abaixo).
- stloc: retira o valor do topo da pilha e o poe na variavel(stloc x)
- sub: subtrai do segundo valor o primeiro e poe o resultado no topo.
- sub.ovf.[type]: subtração de inteiros com overflow
- tail.: deve preceder imediatamente instruções de call, calli ou callvirt. Ela indica que o método corrente na pilha deve ser removido antes da chamada da função ser executada.
- unaligned. (prefix code): expecifica que o endereço na pilha não está alinhado ao tamanho natural.
- volatile. (prefix code): especifica que o endereço no topo da pilha é volátil.
- xor: executa a operação XOR(bit a bit) entre os dois primeiros elementos da pilha colocando seu resultado na mesma.
- ldelem.[type]: carrega um elemento do vetor(segunda posição) no in- dex(primeira posição, topo da pilha) na pilha
- ldelema: poe o endereço do vetor na posição index na pilha.
- ldlen: poe na pilha o tamanho do vetor(topo da pilha)
- ldstr: poe a string(ldstr string) no topo da pilha
- newarr: cria um array com o tipo definido(newarr int32) onde seu tamanho está no topo da pilha.
- sizeof: carrega o tamanho em bits do tipo definido(sizeof int32) na pilha.
- stelem.[type]: coloca no array(terceiro da pilha) no index(segunda da pilha) o valor(topo da pilha).

4 Instalações

Nessa seção sera listados os comandos para instalação das ferramentas necessárias para manipulação e execução dos códigos. Antes de iniciar qualquer instalação no Linux, devemos atualizar nosso pacote utilizando o codigo:

sudo apt-get update

Apos autualizado, podemos instalar as ferramentas especificas.

4.1 Instalação do Mono

Patrocinado pela Microsoft, Mono é uma implementação de código aberto do Microsoft .NET Framework baseado nos padrões ECMA para C # e Common Language Runtime. Para instalar o Mono com apt-get utiliza-se o pacote:

sudo apt-get install mono-complete

4.2 Instalação do OCaml

Para instalar o OCaml basta digitar o seguinte comando no terminal:

sudo apt-get install ocaml

4.3 Instalação do Python

Nesse trabalho iremos utilizar o Python 2.7. Para instalar basta digitar o seguinte comando no terminal:

sudo apt-get install python2.7

5 Gerando a Tradução dos Programas

Os programas em Python devem terminar com a Extensão ".py"

Para a transformação do codigo o mesmo deve ser convergido para C # para depois ser lido o assembly.
Os programas em C # devem terminar com a Extensão ".cs"

Executar o cógido Python

python nanoC.py

Compilar o cógido C # e gerar o .exe

dmcs nanoC.cs

Gerar assembly em cima do oexecutavel C # .exe

monodis nanoC.exe > nanoC.txt

6 Códigos: MiniPython - C# - Assembly

6.1 nano01

MiniPython nano01

```
def nano01():
pass
```

MiniC# nano01

```
using System;

namespace nano01
{
    class Program
    {
        static void Main(String[] args){}
    }
}
```

Assembly nano01

```
.assembly extern mscorlib{}

.assembly 'nano01'
{
  .ver  0:0:0:0
}
.module nano01.exe

.namespace nano01
{
  .class private auto ansi beforefieldinit Program
```

```
    extends [mscorlib]System.Object
  {

    .method public hidebysig specialname rtspecialname
           instance default void '.ctor' ()  cil managed
    {
        .maxstack 8
        ldarg.0
        call instance void object::'.ctor'()
        ret
    }

    // method line 2
    .method private static hidebysig
           default void Main ()  cil managed
    {
        .entrypoint
        .maxstack 0
        ret
    }

  }
}
```

6.2 nano02

MiniPython nano02

```
def nano02():
    n = int(n)
```

MiniC# nano02

```
using System;

namespace nano02
{
    class Program
    {
        static void Main(String[] args)
        {
            int n;
        }
    }
}
```

Assembly nano02

```
.assembly extern mscorlib{}

.assembly 'nano02'
```

```
{
  .ver  0:0:0:0
}
.module nano02.exe

.namespace nano02
{
  .class private auto ansi beforefieldinit Program
    extends [mscorlib]System.Object
  {

    .method public hidebysig specialname rtspecialname
           instance default void '.ctor' ()  cil managed
    {
        .maxstack 8
        ldarg.0
        call instance void object::'.ctor'()
        ret
    }

    // method line 2
    .method private static hidebysig
            default void Main ()  cil managed
    {
        .entrypoint
        .maxstack 0
        .locals init (int32 V_0)
        ret
    }

  }
}
```

6.3 nano03

MiniPython nano03

```
def nano03():
    n = int(n)
    n=1
```

MiniC# nano03

```
using System;

namespace nano03
{
    class Program
    {
        static void Main(String[] args)
```

```
        {
            int n;
            n = 1;
        }
    }

}
```

Assembly nano03

```
.assembly extern mscorlib{}

.assembly 'nano03'
{
  .ver  0:0:0:0
}
.module nano03.exe

.namespace nano03
{
  .class private auto ansi beforefieldinit Program
    extends [mscorlib]System.Object
  {

    .method public hidebysig specialname rtspecialname
           instance default void '.ctor' ()  cil managed
    {
        .maxstack 8
        ldarg.0
        call instance void object::'.ctor'()
        ret
    }

    // method line 2
    .method private static hidebysig
           default void Main ()  cil managed
    {
        .entrypoint
        .maxstack 0
        .locals init (int32 V_0)
        ldc.i4.1
        stloc.0
        ret
    }

  }
}
```

6.4 nano04

MiniPython nano04

```
def nano04():
    n = int(n)
    n = 1 + 2
```

MiniC# nano04

```
using System;

namespace nano04
{
    class Program
    {
        static void Main(String[] args)
        {
            int n;
            n = 1 + 2;
        }
    }

}
```

Assembly nano04

```
.assembly extern mscorlib{}

.assembly 'nano04'
{
  .ver  0:0:0:0
}
.module nano04.exe

.namespace nano04
{
  .class private auto ansi beforefieldinit Program
    extends [mscorlib]System.Object
  {

    .method public hidebysig specialname rtspecialname
           instance default void '.ctor' ()  cil managed
    {
        .maxstack 8
        ldarg.0
        call instance void object::'.ctor'()
        ret
    }

    // method line 2
    .method private static hidebysig
```

```
            default void Main ()  cil managed
    {
        .entrypoint
        .maxstack 0
        .locals init (int32 V_0)
        ldc.i4.3
        stloc.0
        ret
    }

  }
}
```

6.5 nano05

MiniPython nano05

```
def nano05():
    n = 2
    print(n,end="")

nano05()
```

MiniC# nano05

```
using System;

namespace nano05
{
    class Program
    {
        static void Main(String[] args)
        {
            int n;
            n = 2;
            Console.WriteLine(n);
        }
    }

}
```

Assembly nano05

```
.assembly extern mscorlib{}

.assembly 'nano05'
{
  .ver  0:0:0:0
}
.module nano05.exe

```

```
.namespace nano05
{
  .class private auto ansi beforefieldinit Program
    extends [mscorlib]System.Object
  {

    .method public hidebysig specialname rtspecialname
           instance default void '.ctor' ()  cil managed
    {
         .maxstack 8
         ldarg.0
         call instance void object::'.ctor'()
         ret
    }

    // method line 2
    .method private static hidebysig
            default void Main ()  cil managed
    {
         .entrypoint
         .maxstack 0
         .locals init (int32 V_0)
         ldc.i4.2
         stloc.0
         ldloc.0
         call void class [mscorlib]System.Console::WriteLine(
            int32)
         ret
    }

  }
}
```

6.6 nano06

MiniPython nano06

```
def nano06():
    n = 1 - 2
    print(n,end="")

nano06()
```

MiniC# nano06

```
using System;

namespace nano06
{
    class Program
```

```
    {
        static void Main(String[] args)
        {
            int n;
            n = 1 - 2;
            Console.WriteLine(n);
        }
    }

}
```

Assembly nano06

```
.assembly extern mscorlib{}

.assembly 'nano06'
{
  .ver  0:0:0:0
}
.module nano06.exe

.namespace nano06
{
  .class private auto ansi beforefieldinit Program
    extends [mscorlib]System.Object
  {

    .method public hidebysig specialname rtspecialname
            instance default void '.ctor' ()  cil managed
    {
        .maxstack 8
        ldarg.0
        call instance void object::'.ctor'()
        ret
    }

    // method line 2
    .method private static hidebysig
            default void Main ()  cil managed
    {
        .entrypoint
        .maxstack 0
        .locals init (int32 V_0)
        ldc.i4.m1
        stloc.0
        ldloc.0
        call void class [mscorlib]System.Console::WriteLine(
            int32)
        ret
    }

```

```
  }
}
```

6.7 nano07

MiniPython nano07

```
def nano07():
    n=1
    if n ==1:
        print(n,end="")

nano07()
```

MiniC# nano07

```
using System;

namespace nano07
{
    class Program
    {
        static void Main(String[] args)
        {
            int n;
            n = 1;
            if(n == 1)
            {
                Console.WriteLine(n);
            }
        }
    }

}
```

Assembly nano07

```
.assembly extern mscorlib{}

.assembly 'nano07'
{
  .ver  0:0:0:0
}
.module nano07.exe

.namespace nano07
{
  .class private auto ansi beforefieldinit Program
    extends [mscorlib]System.Object
  {

```

```
    .method public hidebysig specialname rtspecialname
           instance default void '.ctor' ()  cil managed
    {
        .maxstack 8
        ldarg.0
        call instance void object::'.ctor'()
        ret
    }

    // method line 2
    .method private static hidebysig
           default void Main ()  cil managed
    {
           .entrypoint
           .maxstack 0
           .locals init (int32 V_0)

           ldc.i4.1
           stloc.0
           ldloc.0
           ldc.i4.1
           bne.un IL_000f

           ldloc.0
           call void class [mscorlib]System.Console::
               WriteLine(int32)
  IL_000f:  ret
    }

  }
}
```

6.8 nano08

MiniPython nano08

```
def nano08():
    n=1
    if n ==1:
        print(n,end="")
    else:
        print(0,end="")

nano08()
```

MiniC# nano08

```
using System;

namespace nano08
```

```
{
    class Program
    {
        static void Main(String[] args)
        {
            int n;
            n = 1;
            if(n == 1)
            {
                Console.WriteLine(n);
            }
            else{
                Console.WriteLine(0);
            }
        }
    }

}
```

Assembly nano08

```
.assembly extern mscorlib{}

.assembly 'nano08'
{
  .ver  0:0:0:0
}
.module nano08.exe

.namespace nano08
{
  .class private auto ansi beforefieldinit Program
    extends [mscorlib]System.Object
  {

    .method public hidebysig specialname rtspecialname
           instance default void '.ctor' ()  cil managed
    {
        .maxstack 8
        ldarg.0
        call instance void object::'.ctor'()
        ret
    }

    // method line 2
    .method private static hidebysig
           default void Main ()  cil managed
    {
           .entrypoint
           .maxstack 0
           .locals init (int32 V_0)
```

```
31
32              ldc.i4.1
33              stloc.0
34              ldloc.0
35              ldc.i4.1
36              bne.un IL_0014
37
38              ldloc.0
39              call void class [mscorlib]System.Console::
                    WriteLine(int32)
40              br IL_001a
41
42  IL_0014:    ldc.i4.0
43              call void class [mscorlib]System.Console::
                    WriteLine(int32)
44  IL_001a:    ret
45      }
46
47    }
48  }
```

6.9 nano09

MiniPython nano09

```
1  def nano09():
2      n=1
3      if n ==1:
4          n = n + 1
5          print(n,end="")
6      else:
7          print(0,end="")
8
9  nano09()
```

MiniC# nano09

```
1  using System;
2
3  namespace nano09
4  {
5      class Program
6      {
7          static void Main(String[] args)
8          {
9              int n;
10             n = 1;
11             if(n == 1)
12             {
13                 n = n + 1;
```

```
                Console.WriteLine(n);
            }
            else{
                Console.WriteLine(0);
            }
        }
    }

}
```

Assembly nano09

```
.assembly extern mscorlib{}

.assembly 'nano09'
{
  .ver  0:0:0:0
}
.module nano09.exe

.namespace nano09
{
  .class private auto ansi beforefieldinit Program
    extends [mscorlib]System.Object
  {

    .method public hidebysig specialname rtspecialname
           instance default void '.ctor' ()  cil managed
    {
        .maxstack 8
        ldarg.0
        call instance void object::'.ctor'()
        ret
    }

    // method line 2
    .method private static hidebysig
           default void Main ()  cil managed
    {
            .entrypoint
            .maxstack 0
            .locals init (int32 V_0)

            ldc.i4.1
            stloc.0
            ldloc.0
            ldc.i4.1
            bne.un IL_0018

            ldloc.0
            ldc.i4.1
```

```
            add
            stloc.0
            ldloc.0
            call void class [mscorlib]System.Console::
                WriteLine(int32)
            br IL_001e

IL_0018:    ldc.i4.0
            call void class [mscorlib]System.Console::
                WriteLine(int32)
IL_001e:    ret
    }

  }
}
```

6.10 nano10

MiniPython nano10

```
def nano10():
    n=1
    m=2
    if n ==m:
        print(n,end="")
    else:
        print(0,end="")

nano10()
```

MiniC# nano10

```
using System;

namespace nano10
{
    class Program
    {
        static void Main(String[] args)
        {
            int n;
            int m;
            n = 1;
            m = 2;
            if(n == m)
            {
                Console.WriteLine(n);
            }
            else{
                Console.WriteLine(0);
```

```
            }
        }
    }

}
```

Assembly nano10

```
.assembly extern mscorlib{}

.assembly 'nano10'
{
  .ver  0:0:0:0
}
.module nano10.exe

.namespace nano10
{
  .class private auto ansi beforefieldinit Program
    extends [mscorlib]System.Object
  {

    .method public hidebysig specialname rtspecialname
           instance default void '.ctor' ()  cil managed
    {
        .maxstack 8
        ldarg.0
        call instance void object::'.ctor'()
        ret
    }

    // method line 2
    .method private static hidebysig
           default void Main ()  cil managed
    {
           .entrypoint
           .maxstack 0
           .locals init (
               int32   V_0,
               int32   V_1)

           ldc.i4.1
           stloc.0
           ldc.i4.2
           stloc.1
           ldloc.0
           ldloc.1
           bne.un IL_0016

           ldloc.0
           call void class [mscorlib]System.Console::
```

```
                WriteLine(int32)
            br IL_001c


IL_0016     ldc.i4.0
            call void class [mscorlib]System.Console::
                WriteLine(int32)
IL_001c:    ret
    }

  }
}
```

6.11 nano11

MiniPython nano11

```
def nano11():
    n=1
    m=2
    x=5
    while x >n:
        n = n + m
        print(n,end="")

nano11()
```

MiniC# nano11

```
using System;

namespace nano11
{
    class Program
    {
        static void Main(String[] args)
        {
            int n;
            int m;
            int x;
            n = 1;
            m = 2;
            x = 5;
            while(x > n)
            {
                n = n + m;
                Console.WriteLine(n);
            }
        }
    }
```

```
22
23 }
```

Assembly nano11

```
1  .assembly extern mscorlib{}
2
3  .assembly 'nano11'
4  {
5    .ver  0:0:0:0
6  }
7  .module nano11.exe
8
9  .namespace nano11
10 {
11   .class private auto ansi beforefieldinit Program
12     extends [mscorlib]System.Object
13   {
14
15     .method public hidebysig specialname rtspecialname
16            instance default void '.ctor' ()  cil managed
17     {
18         .maxstack 8
19         ldarg.0
20         call instance void object::'.ctor'()
21         ret
22     }
23
24     // method line 2
25     .method private static hidebysig
26            default void Main ()  cil managed
27     {
28            .entrypoint
29            .maxstack 0
30            .locals init (
31                int32   V_0,
32                int32   V_1,
33                int32   V_2)
34
35            ldc.i4.1
36            stloc.0
37            ldc.i4.2
38            stloc.1
39            ldc.i4.5
40            ldloc.2
41            bne.un IL_0015
42
43  IL_000b:  ldloc.0
44            ldloc.1
45            add
46            stloc.0
```

```
            ldloc.0
            call void class [mscorlib]System.Console::
                WriteLine(int32)
 IL_0015:   ldloc.2
            ldloc.0
            br IL_000b


  IL_001c:  ret
    }

  }
}
```

6.12 nano12

MiniPython nano12

```
def nano12():
    n=1
    m=2
    x=5
    while x >n:
        if n ==m:
            print(n,end="")
        else:
            print(0,end="")
        x = x -1

nano12()
```

MiniC# nano12

```
using System;

namespace nano12
{
    class Program
    {
        static void Main(String[] args)
        {
            int n = 1;
            int m = 2;
            int x = 5;
            while(x > n)
            {
                if(n == m){
                    Console.WriteLine(n);
                }
                else{
```

```
                    Console.WriteLine(0);
                }
                x = x - 1;
            }
        }
    }

}
```

Assembly nano12

```
.assembly extern mscorlib{}

.assembly 'nano12'
{
  .ver  0:0:0:0
}
.module nano12.exe

.namespace nano12
{
  .class private auto ansi beforefieldinit Program
    extends [mscorlib]System.Object
  {

    .method public hidebysig specialname rtspecialname
           instance default void '.ctor' ()  cil managed
    {
        .maxstack 8
        ldarg.0
        call instance void object::'.ctor'()
        ret
    }

    // method line 2
    .method private static hidebysig
           default void Main ()  cil managed
    {
            .entrypoint
            .maxstack 0
            .locals init (
                int32   V_0,
                int32   V_1,
                int32   V_2)

            ldc.i4.1
            stloc.0
            ldc.i4.2
            stloc.1
            ldc.i4.5
            ldloc.2
```

```
            br IL_0027

 IL_000b:   ldloc.0
            ldloc.1
            bne.un IL_001d

            ldloc.0
            call void class [mscorlib]System.Console::
                WriteLine(int32)
            br IL_0023

 IL_001d:   ldc.i4.0
            call void class [mscorlib]System.Console::
                WriteLine(int32)
            ldloc.2
 IL_0023    ldc.i4.1
            sub
            stloc.2
 IL_0027:   ldloc.2
            ldloc.0
            bgt IL_000b

 IL_002e:   ret
    }

  }
}
```

6.13 micro01

MiniPython micro01

```
def micro01():
   cel , far = 0.0 , 0.0
    print("    Tabela de conversao: Celsius -> Fahrenheit")
    print("Digite a temperatura em Celsius: ",end="")
    cel = input()
    far = (9*cel+160)/5
    print("A nova temperatura   :"+str(far)+"F")

micro01()
```

MiniC# micro01

```
using System;

namespace micro01
{
    class Program
    {
```

```
7           static void Main(String[] args)
8           {
9               /*
10              Fun   o: Ler uma temperatura em graus Celsius e
                    apresent -la
11              convertida em graus Fahrenheit. A f rmula de
                    convers o   :
12              F=(9*C+160) / 5,
13              sendo F a temperatura em Fahrenheit e C a
                    temperatura em
14              Celsius.
15              */
16
17              float cel;
18              float far;
19
20              Console.WriteLine("    Tabela de converes o:
                    Celsius -> Fahrenheit");
21              Console.Write("Digite a temperatura em Celsius:
                    ");
22              cel = Console.ReadLine();
23              far = (9*cel+160)/5;
24
25              Console.WriteLine("A nova temperatura   : ",far,
                    " F");
26          }
27      }
28
29  }
```