



Standardizing Information and Communication Systems

Common Language Infrastructure (CLI)
Part 5: Annexes

Draft 01 – October 2000

This contribution is being provided “AS IS”, and the SPONSORS EXPRESSLY DISCLAIM ANY AND ALL WARRANTIES REGARDING THIS CONTRIBUTION, INCLUDING ANY WARRANTY THAT THIS CONTRIBUTION DOES NOT VIOLATE THE RIGHTS OF OTHERS OR IS FIT FOR A PARTICULAR PURPOSE.

Brief History

This ECMA Standard has been adopted by the ECMA General Assembly of

Table of contents

Annex A.	Informative: Sample Programs	3
A.1.	Mutually Recursive Program (with tail calls)	3
A.2.	Using Value Types	4
A.3.	User Interface Sample	8
A.3.1.	CountDown.il	10
A.3.2.	CountDownComponents.il	18
A.3.3.	Counter.il	31
A.3.4.	CountDownSecondsLabel.cs	44
A.3.5.	CountDownErrorLabel.cs	44
A.3.6.	CountDownErrorLabel.cpp	45
A.3.7.	compile.bat	46
Annex B.	Informative: List of ILASM Keywords	47
Annex C.	Informative: ILASM Complete Grammar	1
Annex D.	Informative: ILASM Instruction Syntax	15
D.1.	Top-level Instruction Syntax	15
D.2.	Instructions with no operand	16
D.3.	Instructions that Refer to Parameters or Local Variables	17
D.4.	Instructions that Take a Single 32-bit Integer Argument	17
D.5.	Instructions that Take a Single 64-bit Integer Argument	17
D.6.	Instructions that Take a Single Floating Point Argument	18
D.7.	Branch instructions	18
D.8.	Instructions that Take a Method as an Argument	18
D.9.	Instructions that Take a Field of a Class as an Argument	19
D.10.	Instructions that Take a Type as an Argument	19
D.11.	Instructions that Take a String as an Argument	19
D.12.	Instructions that Take a Signature as an Argument	19
D.13.	Instructions that Take a Metadata Token as an Argument	20
D.14.	The SSA Φ -Node Instruction	20
D.15.	Switch instruction	20
Annex E.	Normative: Opcode Definitions	22
Annex F.	Normative: Values for .subsystem	30
Annex G.	Normative: Values for .corflags	31
Annex H.	Normative: Values for Hash Algorithm ID	32
Annex I.	Normative: Values for .os	36
Annex J.	Normative: Values for .processor	37

Annex K.	Normative: Default Marshalling - Unmanaged to Managed	38
Annex L.	Normative: Default Marshalling - Managed to Unmanaged	41
Annex M.	Informative: Metadata Validation Rules	45
Annex N.	Normative: Explicit Class Layout Rules	45
Annex O.	Informative: Class Library Design Guidelines	45

Annex A. Informative: Sample Programs

This Annex shows several complete examples. Unlike the integrated examples, these examples have many features of the CLR merged into a program.

The examples can be saved to a file, assembled and are ready to run.

The first two examples show small programs. The last example shows a rather large program.

A.1. Mutually Recursive Program (with tail calls)

The following is an example of a mutually recursive program that uses tail calls. The methods below determine whether a number is even or odd.

```
.assembly test.exe { }
.class EvenOdd
{ .method private static bool IsEven(int32 N) il managed
  { .maxstack 2
    ldarg.0          // N
    ldc.i4.0
    bne.un          NonZero
    ldc.i4.1
    ret
NonZero:
    ldarg.0
    ldc.i4.1
    sub
    tail.
    call bool EvenOdd::IsOdd(int32)
    ret
  } // end of method 'EvenOdd::IsEven'

.method private static bool IsOdd(int32 N) il managed
{ .maxstack 2
  // Demonstrates use of argument names and labels
  // Notice that the assembler does not covert these
  // automatically to their short versions
  ldarg          N
  ldc.i4.0
  bne.un          NonZero
  ldc.i4.0
  ret
NonZero:
  ldarg          N
  ldc.i4.1
  sub
  tail.
```

```

                                - 4 -
    call bool EvenOdd::IsEven(int32)
    ret
} // end of method 'EvenOdd::IsOdd'

.method public static void Test(int32 N) il managed
{ .maxstack    1
  ldarg        N
  call         void System.Console::Write(int32)
  ldstr        " is "
  call         void System.Console::Write(class System.String)
  ldarg        N
  call         bool EvenOdd::IsEven(int32)
  brfalse      LoadOdd
  ldstr        "even"
Print:
  call         void System.Console::WriteLine(class System.String)
  ret
LoadOdd:
  ldstr        "odd"
  br           Print
} // end of method 'EvenOdd::Test'
} // end of class 'EvenOdd'

//Global method

.method public static void main() il managed
{ .entrypoint
  .maxstack    1
  ldc.i4.5
  call         void EvenOdd::Test(int32)
  ldc.i4.2
  call         void EvenOdd::Test(int32)
  ldc.i4       100
  call         void EvenOdd::Test(int32)
  ldc.i4       1000001
  call         void EvenOdd::Test(int32)
  ret
} // end of global method 'main'

```

A.2.Using Value Types

The following program shows how rational numbers can be implemented using value types.


```

.assembly rational.exe { }
.class private value sealed Rational extends System.ValueType
    implements System.IComparable
{
    .field public int32 Numerator
    .field public int32 Denominator

    .method virtual public int32 CompareTo(class System.Object o)
    // Implements IComparable::CompareTo(Object)
    { ldarg.0      // this as managed pointer
      ldfld int32 value class Rational::Numerator
      ldarg.1      // o as object
      unbox value class Rational
      ldfld int32 value class Rational::Numerator
      beq.s TryDenom
      ldc.i4.0
      ret
    TryDenom:
      ldarg.0      // this as managed pointer
      ldfld int32 value class Rational::Denominator
      ldarg.1      // o as object
      unbox value class Rational
      ldfld int32 class Rational::Denominator
      ceq
      ret
    }

    .method virtual public class System.String ToString()
    // Implements Object::ToString
    { .locals init (class System.Text.StringBuilder SB,
                    class System.String S,
                    class System.Object N,
                    class System.Object D)
      newobj void System.Text.StringBuilder::.ctor()
      stloc.s SB
      ldstr "The value is: {0}/{1}"
      stloc.s S
      ldarg.0      // Managed pointer to self
      dup
      ldflda int32 value class Rational::Numerator
      box System.Int32
      stloc.s N
      ldflda int32 value class Rational::Denominator
      box System.Int32
      stloc.s D
      ldloc.s SB

```

- 6 -

```
ldloc.s S
ldloc.s N
ldloc.s D
call instance class System.Text.StringBuilder
    System.Text.StringBuilder::AppendFormat(class System.String,
                                             class System.Object,
                                             class System.Object)
callvirt instance class System.String System.Object::ToString()
ret
}
.method public value class Rational Mul(value class Rational)
{ .locals init (value class Rational Result)

    ldloca.s Result
    dup
    ldarg.0      // this
    ldflld int32 value class Rational::Numerator
    ldarga.s     1      // arg
    ldflld int32 value class Rational::Numerator
    mul
    stflld int32 value class Rational::Numerator
    ldarg.0      // this
    ldflld int32 value class Rational::Denominator
    ldarga.s     1      // arg
    ldflld int32 value class Rational::Denominator
    mul
    stflld int32 value class Rational::Denominator
    ldloc.s Result
    ret
}
}
.method void main()
{ .entrypoint
    .locals init (value class Rational Half,
                  value class Rational Third,
                  value class Rational Temporary,
                  class System.Object H,
                  class System.Object T)
    // Initialize Half, Third, H, and T
    ldloca.s Half
    dup
    ldc.i4.1
    stflld int32 value class Rational::Numerator
```

```
ldc.i4.2
stfld int32 value class Rational::Denominator
ldloc.s Third
dup
ldc.i4.1
stfld int32 value class Rational::Numerator
ldc.i4.3
stfld int32 value class Rational::Denominator
ldloc.s Half
box value class Rational
stloc.s H
ldloc.s Third
box value class Rational
stloc.s T
// WriteLine(H.IComparable::CompareTo(H))
// Call CompareTo via interface using boxed instance
ldloc H
dup
callvirt int32 System.IComparable::CompareTo(class System.Object)
call void System.Console::WriteLine(bool)
// WriteLine(Half.CompareTo(T))
// Call CompareTo via value type directly
ldloc.s Half
ldloc T
call instance int32
value class Rational::CompareTo(class System.Object)
call void System.Console::WriteLine(bool)
// WriteLine(Half.ToString())
// Call virtual method via value type directly
ldloc.s Half
call instance class System.String class Rational::ToString()
call void System.Console::WriteLine(class System.String)
// WriteLine(T.ToString())
// Call virtual method inherited from Object, via boxed instance
ldloc T
callvirt class System.String System.Object::ToString()
call void System.Console::WriteLine(class System.String)
// WriteLine((Half.Mul(T)).ToString())
// Mul is called on two value types, returning a value type
// ToString is then called directly on that value type
// Note that we are required to introduce a temporary variable
// since the call to ToString requires a managed pointer (address)
ldloc.s Half
ldloc.s Third
call instance value class Rational
    Rational::Mul(value class Rational)
```

```
stloc.s Temporary
ldloca.s Temporary
call instance class System.String Rational::ToString()
call void System.Console::WriteLine(class System.String)
ret
}
```

A.3. User Interface Sample

This section shows a larger application written in the IL assembly language. It shows many features of the CLR discussed in this document and interoperation with C# and managed C++.

Parts of this sample were used in integrated samples in the document. Some of the integrated samples were modified to fit the context better.

Motivated by waiting for the release date of the CLR, it seemed appropriate to write an application that counts down the seconds left. It is called Microsoft .NET Countdown. The following application has the following features:

- Displays the seconds left in a window (not the console) with appropriate labeling and updates the display every second until zero is reached.
- Has a start/stop button that allows the count down to halt or resume.
- Well, let's be honest ship dates do move. So, we need functionality to enter a new value for the seconds left using the GUI.
- Rejects values entered for the number of seconds that are not positive and valid integers.
- Has resizing behavior for all elements.
- Produces a beep every second and a final sound when the count reaches zero.

The following is a screen shot of the application:



Figure A.1: Screenshot of Microsoft.NET Countdown. The user just entered the value shown and clicked Start.

Even though the program could have been written as one monolithic piece, it is split into components and modules. This way the various pieces become usable, independent of each other.

The window itself, the text box, the button, and the two labels form the UI components. In addition, there are components in the backend that represent the application, the counter, the count value, the exceptions, the error values, and the various events happening in the application. The application forms one assembly that consists of three modules written in the IL assembly language:

- *CountDown.exe*
- *CountDownComponents.dll*
- *Counter.dll*

The component displaying the text "Seconds left" is written in C# and comes from the module *CountDownSecondsLabel.dll*. The component displaying the error messages at the top of the window, one of which is shown in the screenshot above, is available both in managed C++ and C#. The module is called *CountDownErrorLabel.dll*. The beeps are produced using native code from the Windows library *user32.dll* (if Windows9x is used, replace the reference to the DLL by *user.dll*). (Since C++ does not produce verifiable code, you may need to alter your security settings to run the program.)

The following graph gives an overview of the design of the application:

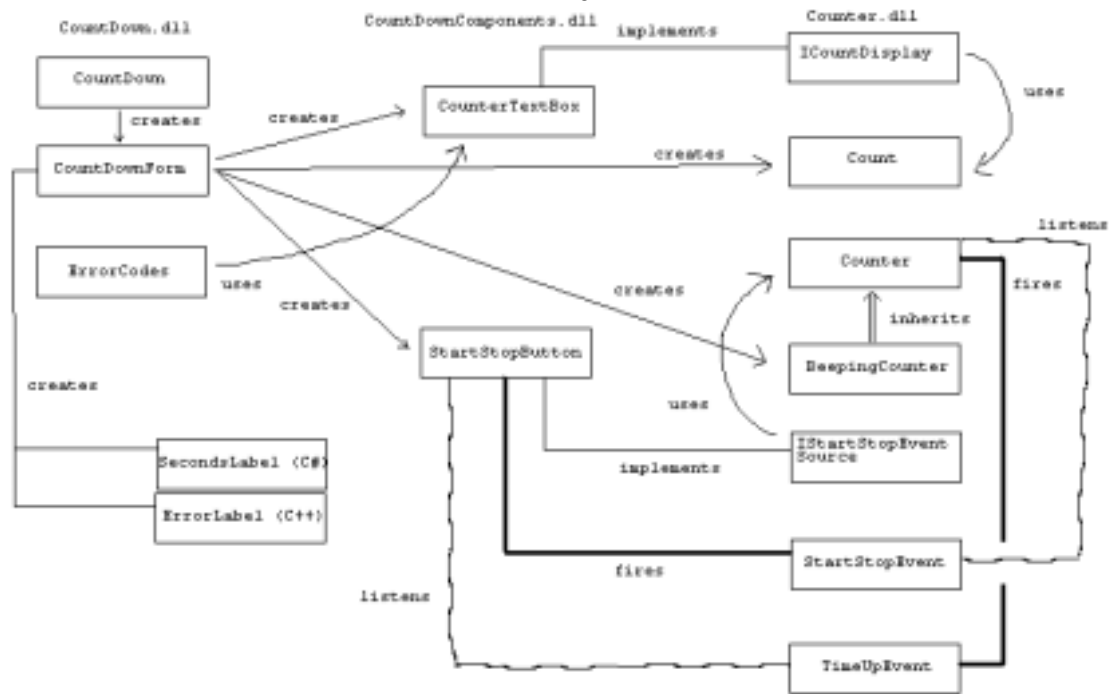


Figure A.2: Overview of the design of Microsoft .NET Countdown.

The following subsections show each module of the sample. To compile and execute the program, each section needs to be Copied into a file that has the same name as the section itself. Section A.3.1 to A.3.3 show IL assembly language source code. Sections A.3.4 and A.3.5 show C# source code. And section A.3.6 shows managed C++ source code.

Two of the sections contain the source code for batch files. The file *compile.bat* can be used to compile the files. The syntax is:

```
compile [C++]
```

If the option C++ is specified, the error label will be compiled with the C++ compiler. If no option is specified, C# version of the error label will be compiled.

The program is called *CountDown.exe*.

A.3.1. Countdown.il

```

/*
 * Microsoft .Net Countdown
 *
 * This is the main file of the assembly.
 * It defines the assembly and contains the entrypoint.
 * This file declares the classes
 * - Countdown
 * - DountDownForm
  
```

Standard master

Beta 1 Release

```
* This file declares the enumeration
* - ErrorCodes
*
*/

/*****
/*      Information about the assembly      */
*****/

/* The assembly declaration. */
.assembly Countdown {
    .hash algorithm 32772      // selected algorithm is SHA1
    .ver 1:0:0:0              // version 1.0
}

/* The assembly declaration. */
.assembly Countdown {
    .hash algorithm 32772      // selected algorithm is SHA1
    .ver 1:0:0:0              // version 1.0
    // may also add originator key
}

/* The assembly references. */
.assembly extern mscorlib {
    .originator = (03 68 91 16 D3 A4 AE 33)
    // may also add hash value and version
}

.assembly extern System.Windows.Forms {
    .originator = (03 68 91 16 D3 A4 AE 33)
}

.assembly extern System.Drawing {
    .originator = (03 68 91 16 D3 A4 AE 33)
}

.assembly extern System.Timers {
    .originator = (03 68 91 16 D3 A4 AE 33)
}

/* Files referenced by the assembly, all DLLs need to appear here.
   All references by modules need to be listed here, too. */
.file CountdownComponents.dll
.file Counter.dll
.file CountdownSecondsLabel.dll
.file CountdownErrorLabel.dll
```

```

/*****
/*      Information about this module      */
*****/

/* The module declaration. */
.module Countdown.exe

/* The module references.
   All these modules become part of this assembly. */
.module extern CountdownComponents.dll
.module extern Counter.dll
.module extern CountdownSecondsLabel.dll
.module extern CountdownErrorLabel.dll

/*****
/*      Class: Countdown      */
*****/

/* This is the main class of the application. It contains
   * the entrypoint. */
.class public auto autochar Countdown extends [mscorlib]System.Object {

    /* constructor */
    .method public rtspecialname specialname hidebysig instance void
    .ctor() il managed {
        .maxstack 1
        ldarg.0          // load this pointer
        // call super constructor
        call instance void [mscorlib]System.Object::.ctor()
        ret
    }

    /* entrypoint to the application
       * (can have any name)
       */
    .method public static hidebysig int32 Main() il managed {
        .entrypoint
        .maxstack 1
        .zeroinit          // initialize all locals
        // declare locals
    }
}

```


Standard master**Beta 1 Release**

```

.locals (class CountdownForm countdownForm)
// GC will run finalizers
call void [mscorlib]System.GC::RequestFinalizeOnShutdown()
// instantiate CountdownForm
newobj instance void CountdownForm::.ctor()
stloc countdownForm           // store instance
ldloc countdownForm           // load form onto stack
// start the Form on a new thread
call void [System.Windows.Forms]System.Windows.Forms.Application::Run(class
[System.Windows.Forms]System.Windows.Forms.Form)
ldc.i4.0                      // successful return
ret
}
} // end of class Countdown

```

```

/*****
/*          Class: CountdownForm          */
*****/

/* Represents the form displayed on the screen. Contains and manages all
the controls
* of the application */
.class private auto autochar CountdownForm extends
[System.Windows.Forms]System.Windows.Forms.Form {

    /* the instance fields */
    .field private class [.module CountdownSecondsLabel.dll]SecondsLabel
secondsLabel
    .field private class [.module CountdownComponents.dll]CounterTextBox
counterBox
    .field private class [.module CountdownComponents.dll]StartStopButton
button
    .field private class [.module CountdownErrorLabel.dll]ErrorLabel
errorLabel
    .field private class [.module Counter.dll]Counter counter
    .field private class System.String[] errorStrings

    /* constructor */
    .method public rtspecialname specialname hidebysig instance void
.ctor() il managed {
        .maxstack 3
        .locals init (class [.module Counter.dll]Count count)

        // call super constructor
        ldarg.0
        call instance void [System.Windows.Forms]System.Windows.Forms.Form::.ctor()

        // initialize error strings

```

- 14 -

```
ldarg.0
callvirt instance void CountdownForm::InitializeErrorStrings()

// setup form
ldarg.0
callvirt instance void CountdownForm::Initialize()

// create the button and store in an instance variable
ldarg.0
dup
newobj instance void [.module
CountDownComponents.dll]StartStopButton::.ctor(class
[System.WinForms]System.WinForms.Form)
stfld class [.module CountdownComponents.dll]StartStopButton
CountDownForm::button

// create the text label and store in an instance variable, the
label initializes itself
ldarg.0
dup
newobj instance void [.module
CountDownSecondsLabel.dll]SecondsLabel::.ctor(class
[System.WinForms]System.WinForms.Form)
stfld class [.module CountdownSecondsLabel.dll]SecondsLabel
CountDownForm::secondsLabel

// create the time box and store in an instance variable
ldarg.0
dup
newobj instance void [.module
CountDownComponents.dll]CounterTextBox::.ctor(class CountdownForm)
stfld class [.module CountdownComponents.dll]CounterTextBox
CountDownForm::counterBox

// create the count object
ldarg.0
ldfld class [.module CountdownComponents.dll]CounterTextBox
CountDownForm::counterBox
newobj instance void [.module Counter.dll]Count::.ctor(class
[.module Counter.dll]ICountDisplay)
stloc count

// create the counter
ldarg.0
dup
ldfld class [.module CountdownComponents.dll]StartStopButton
CountDownForm::button
ldloc count
```

```

newobj instance void [.module
Counter.dll]BeepingCounter::.ctor(class [.module
Counter.dll]IStartStopEventSource, class [.module Counter.dll]Count)

stfld class [.module Counter.dll]Counter CountdownForm::counter

// add button to the time up event
ldarg.0
ldfld class [.module CountdownComponents.dll]StartStopButton
CountdownForm::button
ldarg.0
ldfld class [.module Counter.dll]Counter CountdownForm::counter
call instance void [.module
CountdownComponents.dll]StartStopButton::AddToTimeUp(class [.module
Counter.dll]Counter)

// create the error label and store in an instance variable
ldarg.0
dup
newobj instance void [.module
CountdownErrorLabel.dll>ErrorLabel::.ctor(class
[System.Windows.Forms]System.Windows.Forms.Form)
stfld class [.module CountdownErrorLabel.dll>ErrorLabel
CountdownForm::errorLabel

ret
}

/* Initializes the form by setting the title, size,
and background color */
.method virtual newslot family hidebysig instance void Initialize()
il managed {
    .maxstack 3
    .locals init (value class [System.Drawing]System.Drawing.Size
size)

    // set the title
    ldarg.0 // load this pointer
    ldstr "CountDown" // load window title onto stack
    // call the setter of the property
    callvirt instance void CountdownForm::set_Text(class
System.String)

    // set the size
    ldloca size
    initobj value class [System.Drawing]System.Drawing.Size
    ldloca size
    ldc.i4 425 // width
    ldc.i4 300 // height
    call instance void
[System.Drawing]System.Drawing.Size::.ctor(int32, int32)
    ldarg.0

```

```
ldloc size
    callvirt instance void CountdownForm::set_Size(value class
[System.Drawing]System.Drawing.Size)

    // set the background
    ldarg.0
    call value class [System.Drawing]System.Drawing.Color
[System.Drawing]System.Drawing.Color::get_CadetBlue()
    callvirt instance void CountdownForm::set_BackColor(value class
[System.Drawing]System.Drawing.Color)

    ret
}

/* Initializes the error strings displayed in case of errors. */
.method virtual newslot family hidebysig instance void
InitializeErrorStrings() il managed {
    // create the array
    ldarg.0
    ldc.i4.4    // number of error strings + 1
    newarr      class System.String
    stfld class System.String[] CountdownForm::errorStrings

    ldarg.0
    ldfld class System.String[] CountdownForm::errorStrings
    ldc.i4.0
    ldstr "" // the empty string, used to display no error
    stelem.ref

    // add the first error string to the array
    ldarg.0
    ldfld class System.String[] CountdownForm::errorStrings
    ldc.i4.1
    ldstr "Please enter an integer!"
    stelem.ref

    // add the next error string to the array
    ldarg.0
    ldfld class System.String[] CountdownForm::errorStrings
    ldc.i4.2
    ldstr "Number is out of range!"
    stelem.ref

    // add the next error string to the array
```

```

        ldarg.0
        ldfld class System.String[] CountdownForm::errorStrings
        ldc.i4.3
        ldstr "Number must be positive!"
        stelem.ref

        ret
    }

    /* Displays the error string according to the passed in error code.
       Error code must be one of the values declared by the enumeration.
    */
    .method virtual newslot famorassem hidebysig instance void
    ShowErrorText(value class ErrorCodes errorCode) synchronized il
    managed {
        // load the appropriate error string
        ldarg.0
        ldfld class [.module CountdownErrorLabel.dll]ErrorLabel
        CountdownForm::errorLabel
        ldarg.0
        ldfld class System.String[] CountdownForm::errorStrings
        ldarg errorCode // load the value of the enumeration
        ldelem.ref
        // error string on stack, display
        callvirt instance void class [.module
        CountdownErrorLabel.dll]ErrorLabel::set_Text(class System.String)
        ret
    }
} // end of class CountdownForm

/*****
/*      Enumeration: ErrorCodes      */
*****/

/* This enumeration defines the valid error codes
 * Must be sealed, since value type. */
.class value sealed serializable auto autochar public ErrorCodes extends
[mscorlib]System.Enum {
    // the field of the enumeration, underlying type is unsigned int8
    .field public specialname rtspecialname unsigned int8 value__

    // Metadata fields documenting the legal values of the enumeration
    and
    // giving them names.
    // This fields are accessible only with reflection.
    .field public static literal value class ErrorCodes no_error =
int8(0)
    .field public static literal value class ErrorCodes format_error =
int8(1)

```

- 18 -

```
.field public static literal value class ErrorCodes overflow_error =
int8(2)
.field public static literal value class ErrorCodes nonpositive_error
= int8(3)
} // end of enumeration ErrorCodes
```

A.3.2. CountdownComponents.il

```
/*
 * Microsoft .Net Countdown
 *
 * This is file contains UI components for Microsoft .NET Countdown.
 * This file declares the classes
 * - CounterTextBox
 * - StartStopButton
 * This file declares the exception
 * - NonPositiveNumberException
 * This file declares the event
 * - StartStopEvent (in StartStopButton)
 *
 */

/* This module does not declare an assembly.
 * As a consequence, it must be referenced by another assembly to
 * be usable. This module does not have an entrypoint
 */
```

```
/******
/*      Information about this module      */
/******

/* The module declaration. compiles into a dll*/
.module CountdownComponents.dll

/* Assembly references */
.assembly extern mscorlib {
    .originator = (03 68 91 16 D3 A4 AE 33)
}

.assembly extern System.WinForms {
    .originator = (03 68 91 16 D3 A4 AE 33)
}

.assembly extern System.Drawing {
    .originator = (03 68 91 16 D3 A4 AE 33)
```

Standard master
}

Beta 1 Release

```
/* Files referenced by the module, all DLLs need to appear here. */
.file Countdown.exe
.file Counter.dll

/* Module references. */
.module extern Countdown.exe
.module extern Counter.dll

/*****
/*          Class: CounterTextBox          */
*****/

/* Implements the text box that shows the seconds. Accepts values
 * from the user. Validates entered values.
 * The CounterTextBox is an ICountDisplay. */
.class private auto autochar CounterTextBox extends
[System.WinForms]System.WinForms.TextBox implements [.module
Counter.dll]ICountDisplay {

    /* Nested class. Declares the NegativeNumberException */
    .class nested assembly NonPositiveNumberException extends
[mscorlib]System.Exception {

        /* constructor */
        .method public rtspecialname specialname void .ctor() il managed {
            .maxstack 1

            ldarg.0                // load this pointer
            // call super constructor
            call instance void [mscorlib]System.Exception::.ctor()
            ret
        }
    } // end of exception NegativeNumberException

/* The instance fields */
.field private class [.module Countdown.exe]CountDownForm parent

/* constructor */
.method famorassem rtspecialname specialname hidebysig instance void
.ctor(class [.module Countdown.exe]CountDownForm parent) il managed {
    .maxstack 2
    ldarg.0                // load this pointer
    // call super constructor
    call instance void
[System.WinForms]System.WinForms.TextBox::.ctor()
```

```
// set parent field
ldarg.0
ldarg parent
stfld class [.module Countdown.exe]CountDownForm
CounterTextBox::parent

// initialize
ldarg.0
callvirt instance void CounterTextBox::Initialize()

// add the text box to the form
ldarg parent
call instance class
[System.WinForms]System.WinForms.Control$ControlCollection
[System.WinForms]System.WinForms.Form::get_Controls()
ldarg.0
callvirt instance void
[System.WinForms]System.WinForms.Control$ControlCollection::Add(class
[System.WinForms]System.WinForms.Control)

ret
}

/* Initializes this text box */
.method virtual newslot private hidebysig instance void Initialize()
il managed {
    .maxstack 3
    .locals init (value class [System.Drawing]System.Drawing.Size
size,
                value class [System.Drawing]System.Drawing.Point
point)

    // set position
    ldloca      point
    initobj     value class [System.Drawing]System.Drawing.Point
    ldloca      point
    ldc.i4      75      // x
    ldc.i4      100     // y
    call instance void
[System.Drawing]System.Drawing.Point::.ctor(int32, int32)
    ldarg.0
    ldloc point
    call instance void
[System.WinForms]System.WinForms.TextBox::set_Location(value class
[System.Drawing]System.Drawing.Point)
```



```

    // set anchor
    ldarg.0
    ldc.i4.3    // TopBottom
    callvirt instance void
[System.WinForms]System.WinForms.TextBox::set_Anchor(value class
[System.WinForms]System.WinForms.AnchorStyles)

    // set the size of the text box
    ldloc     size
    initobj   value class [System.Drawing]System.Drawing.Size
    ldloc     size
    ldc.i4     115    // width
    ldc.i4     20     // height
    call instance void
[System.Drawing]System.Drawing.Size::.ctor(int32, int32)
    ldarg.0
    ldloc size
    callvirt instance void
[System.WinForms]System.WinForms.TextBox::set_Size(value class
[System.Drawing]System.Drawing.Size)

    // set the text of the text box, should be set to a different
value by counter
    ldarg.0
    ldstr "0"
    callvirt instance void
[System.WinForms]System.WinForms.TextBox::set_Text(class
System.String)

    ret
}

/* Sets the value of the count to be displayed */
.method virtual newslot public hidebysig instance void SetCount(int32
count) il managed {
    .maxstack 2
    ldarg.0
    ldarg count
    // convert count to string
    call class System.String [mscorlib]System.Int32::ToString(int32)
    // set teh display
    callvirt instance void
[System.WinForms]System.WinForms.TextBox::set_Text(class
System.String)
    ret
}

/* Retrieves the value currently displayed.
* This includes entries maid by the user.
* Validates the number displayed. If the number is invalid request
CountDownForm to display

```

- 22 -

```
* an appropriate error message
*/
.method virtual newslot public hidebysig instance int32 GetCount() il
managed {
    .maxstack 4
    // local rvalue contains the value to return
    // rvalue is initialized to zero
    .locals      init (int32 rvalue,
                      value class [.module Countdown.exe]ErrorCodes
errorCode)

    // initialize error code
    ldc.i4.0      // no_error
    stloc errorCode // store into enumeration

    ldarg.0
    callvirt instance class System.String CounterTextBox::get_Text()
    // string representing displayed value on stack

    // two catch clauses, need two try declarations
    .try {
    .try {
        // try to convert into integer
        callvirt instance int32 [mscorlib]System.String::ToInt32()
        // integer on stack
        stloc rvalue
        // leave the try and continue with processing
        leave is_integer
    } catch [mscorlib]System.FormatException {
        // entered value is not an integer
        // pop the exception object
        pop

        ldc.i4.1    // format_error
        stloc errorCode

        leave done // leave the catch
    } } catch [mscorlib]System.OverflowException {
        // entered value is number but does not fit into int32
        // pop the exception object
        pop

        ldc.i4.2    // overflow_error
        stloc errorCode
    }
```

```

        leave done // leave the catch
    }
is_integer:
    .try {
        // check whether value is positive
        ldloc rvalue
        ldc.i4.0
        bgt in_range
        // not positive, throw an exception
        newobj instance void
CounterTextBox/NonPositiveNumberException::.ctor()
        throw
in_range:
        // value is positive, leave the try
        leave done
    } catch CounterTextBox/NonPositiveNumberException {
        // catch exception thrown in try block
        // number is not positive
        // pop the exception object
        pop

        ldc.i4.3 // nonpositive_error
        stloc errorCode

        leave done // leave the catch
    }
done:
    // set the error text
    ldarg.0
    ldfld class [.module Countdown.exe]CountDownForm
CounterTextBox::parent
    ldloc errorCode
    callvirt instance void [.module
CountDown.exe]CountDownForm::ShowErrorText(value class [.module
CountDown.exe]ErrorCodes)

    // return with rvalue
    ldloc rvalue
    ret
}
} // end of class CounterTextBox

/*****
Class: StartStopButton
*****/

```

```
/* Represents the start stop button. The button starts or stops the
   counter.
* The button displays the appropriate label and changes state if the
  counter stops.
* Declares the StartStopEvent.
* The StartStopButton is an ISartStopEventSource. */
.class private auto autochar StartStopButton extends
[System.WinForms]System.WinForms.Button implements [.module
Counter.dll]ISartStopEventSource {

    /* The instance fields */
    .field private class [mscorlib]System.EventHandler
onClickEventHandler
    .field private class [.module Counter.dll]TimeUpEventHandler
timeUpEventHandler
    .field private class [.module Counter.dll]StartStopEventHandler
startStopEventHandler
    .field private bool state // 0 = counter stopped, 1 = counter
started

    /* constructor */
    .method public rtspecialname specialname hidebysig instance void
ctor(class [System.WinForms]System.WinForms.Form parent) il managed
{
    .maxstack 2

    ldarg.0 // load this pointer
    // call super constructor
    call instance void
[System.WinForms]System.WinForms.Button::.ctor()

    // intialize state
    ldarg.0
    ldc.i4.0 // counter stopped
    stfld bool StartStopButton::state

    // initialize the button
    ldarg.0
    callvirt instance void StartStopButton::Initialize()

    // add the button to the form
    ldarg parent
    call instance class
[System.WinForms]System.WinForms.Control$ControlCollection
[System.WinForms]System.WinForms.Form::get_Controls()
    ldarg.0
```

```

        callvirt instance void
[System.WinForms]System.WinForms.Control$ControlCollection::Add(class
[System.WinForms]System.WinForms.Control)

        // setup Click event
        ldarg.0
        call instance void StartStopButton::AddToClick()

        ret
    }

    /* finalizer */
    .method virtual family hidebysig instance void Finalize() il managed
    {
        .maxstack 3

        // remove the OnClick event
        ldarg.0
        dup
        ldfld class [mscorlib]System.EventHandler
StartStopButton::onClickEventHandler
        call instance void
[System.WinForms]System.WinForms.Button::remove_Click(class
[mscorlib]System.EventHandler)

        ret
    }

    /* Initializes this button */
    .method virtual newslot private hidebysig instance void Initialize()
il managed {
        .maxstack 3
        .locals init (value class [System.Drawing]System.Drawing.Size
size,
                        value class [System.Drawing]System.Drawing.Point
point)

        // set position
        ldloca        point
        initobj        value class [System.Drawing]System.Drawing.Point
        ldloca        point
        ldc.i4        100    // x
        ldc.i4        200    // y
        call instance void
[System.Drawing]System.Drawing.Point::.ctor(int32, int32)
        ldarg.0
        ldloc point
        call instance void
[System.WinForms]System.WinForms.Button::set_Location(value class
[System.Drawing]System.Drawing.Point)

```

```
// set anchor
ldarg.0
ldc.i4      15      // TopBottom
callvirt instance void
[System.WinForms]System.WinForms.Button::set_Anchor(value class
[System.WinForms]System.WinForms.AnchorStyles)

// set the size of the button
ldloca      size
initobj     value class [System.Drawing]System.Drawing.Size
ldloca      size
ldc.i4      200     // width
ldc.i4      50      // height
call instance void
[System.Drawing]System.Drawing.Size::.ctor(int32, int32)
ldarg.0
ldloc size
callvirt instance void
[System.WinForms]System.WinForms.Button::set_Size(value class
[System.Drawing]System.Drawing.Size)

// set the color
ldarg.0
call value class [System.Drawing]System.Drawing.Color
[System.Drawing]System.Drawing.Color::get_Gold()
callvirt instance void
[System.WinForms]System.WinForms.Button::set_BackColor(value class
[System.Drawing]System.Drawing.Color)

// set the text color
ldarg.0
call value class [System.Drawing]System.Drawing.Color
[System.Drawing]System.Drawing.Color::get_Navy()
callvirt instance void
[System.WinForms]System.WinForms.Button::set_ForeColor(value class
[System.Drawing]System.Drawing.Color)

// set the font height
ldarg.0
ldstr "Arial"
ldc.r4      20
newobj     instance void
[System.Drawing]System.Drawing.Font::.ctor(class System.String,
float32)
callvirt instance void
[System.WinForms]System.WinForms.Button::set_Font(class
[System.Drawing]System.Drawing.Font)

// set the text of the button
```

```

    ldarg.0
    ldstr "Start"
    callvirt instance void
[System.WinForms]System.Windows.Button::set_Text(class
System.String)

    ret
}

/* Sets the state of the button and updates the label
* 0 = counter stopped
* 1 = counter started */
.method virtual newslot famorassem hidebysig instance void
SetState(int32 newState) il managed {
    // set the state
    ldarg.0
    ldarg newState
    stfld bool StartStopButton::state

    // set the label
    ldarg newState
    ldc.i4.0
    beq      stop_state
    // state is counter started, set label to "Stop"
    ldarg.0
    ldstr "Stop"
    callvirt instance void StartStopButton::set_Text(class
System.String)
    br      done
stop_state:
    // state is counter stopped, set label to "Start"
    ldarg.0
    ldstr "Start"
    callvirt instance void StartStopButton::set_Text(class
System.String)
done:
    ret
}

/* adds this button to the Click event */
.method public hidebysig instance void AddToClick() il managed {
    .maxstack 3

    // load method pointer to handler and create a delegate for it
    ldarg.0
    dup    // duplicate top of stack
    dup    // need three version of this

```

- 28 -

```
ldvirtftn instance void StartStopButton::OnClick(class
System.Object, class [mscorlib]System.EventArgs)
    newobj instance void
[mscorlib]System.EventHandler::ctor(class System.Object, int32)
    stfld class [mscorlib]System.EventHandler
StartStopButton::onClickEventHandler

    // load delegate and pass to event source
    ldarg.0
    dup
    ldfld class [mscorlib]System.EventHandler
StartStopButton::onClickEventHandler
    call instance void
[System.Windows.Forms.Button::add_Click(class
[mscorlib]System.EventHandler)
    ret
}

/* Event handler for Click events.
 * Click events are fired for this component when the user presses
the button
 * Declaration follows EventHandler delegate */
.method virtual newslot public hidebysig instance void OnClick(class
System.Object, class [mscorlib]System.EventArgs) il managed {
    .maxstack 3

    // fire the start stop event, counter must be started or stopped
    ldarg.0
    callvirt instance void StartStopButton::fire_StartStopEvent()
    ret
}

/* Adds this button to the TimeUp event.
 * called by CountdownForm. */
.method public hidebysig instance void AddToTimeUp(class [.module
Counter.dll]Counter counter) il managed {
    .maxstack 3

    // load pointer to handler and create a delegate for it
    ldarg.0
    dup // duplicate top of stack
    ldftn instance void StartStopButton::OnTimeUp()
    newobj instance void [.module
Counter.dll]TimeUpEventHandler::ctor(class System.Object, int32)
    stfld class [.module Counter.dll]TimeUpEventHandler
StartStopButton::timeUpEventHandler
    ldarg counter
```



```

    // load delegate and pass to event source
    ldarg.0
    ldfl class [.module Counter.dll]TimeUpEventHandler
    StartStopButton::timeUpEventHandler
    call instance void [.module Counter.dll]Counter::add_TimeUp(class
[.module Counter.dll]TimeUpEventHandler)
    ret
}

/* Event handler for time up events.
 * The TimeUp event signals that the counter has stopped.
 * The state of the button needs to be updated. */
.method virtual newslot famorassem hidebysig instance void OnTimeUp()
il managed {
    // timer stopped, so set state to timer stopped
    ldarg.0
    ldc.i4.0
    callvirt instance void StartStopButton::SetState(int32)
    ret
}

/** Definition of the StartStopEvent */

// Event is implemented using delegate StartStopEventHandler from
Counter.dll

/* Declaration of the StartStopEvent */
.event [.module Counter.dll]StartStopEventHandler StartStopEvent {
    .addon instance void add_StartStopEvent(class [.module
Counter.dll]StartStopEventHandler 'handler')
    .removeon instance void remove_StartStopEvent(class [.module
Counter.dll]StartStopEventHandler 'handler')
    .fire instance void fire_StartStopEvent()
}

/* Add a listener to the StartStopEvent */
.method virtual newslot public specialname hidebysig instance void
add_StartStopEvent(class [.module Counter.dll]StartStopEventHandler
'handler') il managed {
    .maxstack 4
    // combine with the multicast delegate
    ldarg.0
    dup
    ldfl class [.module Counter.dll]StartStopEventHandler
    StartStopButton::startStopEventHandler
    ldarg 'handler'
    call class[mscorlib]System.Delegate
[mscorlib]System.Delegate::Combine(class [mscorlib]System.Delegate,
class [mscorlib]System.Delegate)

```

- 30 -

```
    castclass [.module Counter.dll]StartStopEventHandler
    stfld class [.module Counter.dll]StartStopEventHandler
StartStopButton::startStopEventHandler
    ret
}

/* Remove a listener from the StartStopEvent */
.method virtual newslot public specialname hidebysig instance void
remove_StartStopEvent(class [.module
Counter.dll]StartStopEventHandler 'handler') il managed {
    .maxstack 4
    // remove from the multicast delegate
    ldarg.0
    dup
    ldfld class [.module Counter.dll]StartStopEventHandler
StartStopButton::startStopEventHandler
    ldarg 'handler'
    call class[mscorlib]System.Delegate
[mscorlib]System.Delegate::Remove(class [mscorlib]System.Delegate,
class [mscorlib]System.Delegate)
    castclass [.module Counter.dll]StartStopEventHandler
    stfld class [.module Counter.dll]StartStopEventHandler
StartStopButton::startStopEventHandler
    ret
}

/* Fire the StartStopEvent */
.method virtual newslot family specialname hidebysig instance void
fire_StartStopEvent() il managed {
    .maxstack 3
    // if state is counter stopped, fire start argument
    // if state is counter started, fire stop argument
    ldarg.0
    ldfld bool StartStopButton::state
    brtrue      stop_it
    ldarg.0
    ldc.i4.1    // start counter
    callvirt instance void StartStopButton::SetState(int32)
    br      continue
stop_it:
    ldarg.0
    ldc.i4.0    // stop counter
    callvirt instance void StartStopButton::SetState(int32)
continue:
    // invoke the mutlicast delegate
    ldarg.0
```

```

        ldflld class [.module Counter.dll]StartStopEventHandler
        StartStopButton::startStopEventHandler
        ldarg.0
        ldflld bool StartStopButton::state
        callvirt instance void [.module
Counter.dll]StartStopEventHandler::Invoke(int32)
        ret
    }

    /** End of the definition of the StartStopEvent */

} // end of class StartStopButton

A.3.3. Counter.il
/*
 * Microsoft .Net CountDown
 *
 * This is file defines the counter functionality for Microsoft .NET
CountDown.
 * This file declares the classes
 * - Count
 * - Counter
 * - BeepingCounter
 * - User32
 * This file declares the interfaces
 * - ICountDisplay
 * - IStartStopEventSource
 * This file declares the delegates
 * - StartStopEventHandler
 * - TimeUpEventHandler
 * This file declares the event
 * - TimeUpEvent (in Counter)
 *
 */

/* This module does not declare an assembly.
 * As a consequence, it must be referenced by another assembly to
 * be usable. This module does not have an entrypoint
 */

/*****
 * Information about this module */
*****/

/* Module declaration, compiles into a dll. */
.module CountDownComponents.dll

/* The assembly references. */

```

- 32 -

```
.assembly extern mscorlib {
    .originator = (03 68 91 16 D3 A4 AE 33)
}

.assembly extern System.Timers {
    .originator = (03 68 91 16 D3 A4 AE 33)
}

/*****
/*      Global Data embedded in PEFile      */
*****/

// the default value of the counter
.data COUNTER_DEFAULT = int32(10)    // 10 seconds

/*****
/*      Interface: ICountDisplay      */
*****/

/* Implementer is able to display a count value */
.class interface abstract public auto autochar ICountDisplay {
    .method virtual abstract public hidebysig instance void
        SetCount(int32 count) il managed {}
    .method virtual abstract public hidebysig instance int32 GetCount()
        il managed {}
} // end of interface ICountDisplay

/*****
/*      Interface: IStartStopEventSource      */
*****/

/* Implementer defines a StartSopEvent */
.class interface abstract auto autochar public IStartStopEventSource {
    .method virtual abstract public hidebysig instance void
        add_StartStopEvent(class StartStopEventHandler) il managed {}
    .method virtual abstract public hidebysig instance void
        remove_StartStopEvent(class StartStopEventHandler) il managed {}
} // end of interface IStartStopEventSource

/*****
/*      Class: Count      */
*****/
```

```

/*****
/* Represents a count value. Defines a Count property.
* Interacts with an ICountDisplay to display the value.
* It is not appropriate for Count to be a value type because an
* instances of it is shared between classes.
*/
.class public auto autochar Count extends [mscorlib]System.Object {

    /* Instance fields */
    .field private int32 count
    .field static family int32 counterDefault at COUNTER_DEFAULT //
    defined at global data
    .field family class ICountDisplay display

    /* constructor */
    .method public rtspecialname specialname hidebysig instance void
    .ctor(class ICountDisplay display) il managed {
        // make super call
        ldarg.0
        call instance void [mscorlib]System.Object::.ctor()

        // set display
        ldarg.0
        ldarg display
        stfld class ICountDisplay Count::display

        // initialize count with counterDefault from global data
        // must come after display is set
        ldarg.0
        ldsfld      int32 Count::counterDefault
        callvirt instance void Count::set_Count(int32)

        ret
    }

    /*** Property Count ***/

    /* Declaration of property */
    .property int32 Count() {
        .backing int32 count
        .get instance int32 get_Count()
        .set instance void set_Count(int32)
        .other instance void refresh_Count()
    }

    /* Getter of Count property */

```

- 34 -

```
.method virtual newslot public hidebysig instance int32 get_Count()
il managed {
    ldarg.0
    ldfld int32 Count::count
    ret
}

/* Setter of Count property */
.method virtual newslot public hidebysig instance void
set_Count(int32 newCount) synchronized il managed {
    ldarg.0
    ldarg newCount
    stfld int32 Count::count
    ldarg.0
    ldfld class ICountDisplay Count::display
    ldarg newCount
    callvirt instance void ICountDisplay::SetCount(int32)
    ret
}

/* Other method of Count property.
 * Updates the count value reading the value at the display.
 * This way user entered values are considered. */
.method virtual newslot public hidebysig instance void
refresh_Count() synchronized il managed {
    ldarg.0
    dup
    ldfld class ICountDisplay Count::display
    callvirt instance int32 ICountDisplay::GetCount()
    stfld int32 Count::count
    ret
}

/** End of property Count */

} // end of class Count

/*****
/*
Class: Counter
*/
*****/

/* Represents a counter. Uses a Count object and decrements its value
until zero is reached.
```

Standard master**Beta 1 Release**

* Stops when the count has reached zero. Fires a TimeUpEvent when count has reached zero.

* Defines the TimeUpEvent */

```
.class public auto autochar Counter extends [mscorlib]System.Object {

    /* Instance fields */
    .field private class [System.Timers]System.Timers.Timer timer
    .field private class [mscorlib]System.EventHandler timerEventHandler
    .field family class Count count
    .field private class IStartStopEventSource startStopEventSource
    .field private class StartStopEventHandler startStopEventHandler
    .field private class TimeUpEventHandler timeUpEventHandler

    /* constructor */
    .method public rtspecialname specialname hidebysig instance void
    .ctor(class IStartStopEventSource startStopEventSource, class Count
    count) il managed {
        // call super constructor
        ldarg.0
        call instance void [mscorlib]System.Object::.ctor()

        // set startStopEventSource
        ldarg.0
        ldarg startStopEventSource
        stfld class IStartStopEventSource Counter::startStopEventSource

        // set count
        ldarg.0
        ldarg count
        stfld class Count Counter::count

        // setup the time
        ldarg.0
        callvirt instance void Counter::SetupTimer()

        // listen to the StartStopEvent
        ldarg.0
        callvirt instance void Counter::SetupStartStopEvent()
        ret
    }

    /* finalizer */
    .method virtual family hidebysig instance void Finalize() il managed
    {
        .maxstack 3

        // remove the StartStop event
```

```
ldarg.0
ldfld class IStartStopEventSource Counter::startStopEventSource
ldarg.0
ldfld class StartStopEventHandler Counter::startStopEventHandler
callvirt instance void
IStartStopEventSource::remove_StartStopEvent(class
StartStopEventHandler)

// remove the timer event
ldarg.0
ldfld class [System.Timers]System.Timers.Timer Counter::timer
ldarg.0
ldfld class [mscorlib]System.EventHandler
Counter::timerEventHandler
call instance void
[System.Timers]System.Timers.Timer::remove_Tick(class
[mscorlib]System.EventHandler)

ret
}

/* setup the timer and start listening to the OnTick event */
.method virtual newslot famorassem hidebysig instance void
SetupTimer() il managed {
    .maxstack 3

    // setup the timer
    ldarg.0
    ldc.r8      1000
    newobj      instance void
[System.Timers]System.Timers.Timer::.ctor(float64)
    stfld class [System.Timers]System.Timers.Timer Counter::timer

    // set auto reset property of timer
    ldarg.0
    ldfld class [System.Timers]System.Timers.Timer Counter::timer
    ldc.i4.1
    call instance void
[System.Timers]System.Timers.Timer::set_AutoReset(bool)

    // load method pointer to event handler and create delegate
    ldarg.0
    dup        // duplicate top of stack
    dup        // duplicate top of stack again
    ldvirtftn instance void Counter::OnTick(class System.Object, class
[mscorlib]System.EventArgs)
```



```

    newobj          instance void
[mscorlib]System.EventHandler::.ctor(class System.Object, int32)
    stfld class [mscorlib]System.EventHandler
Counter::timerEventHandler

    // start listening OnTick event
    ldarg.0
    ldfld class [System.Timers]System.Timers.Timer Counter::timer
    ldarg.0
    ldfld class [mscorlib]System.EventHandler
Counter::timerEventHandler
    call instance void
[System.Timers]System.Timers.Timer::add_Tick(class
[mscorlib]System.EventHandler)

    ret
}

/* start listening to the StartStopEvent
 * fired when user want to start or stop the counter */
.method virtual newslot famorassem hidebysig instance void
SetupStartStopEvent() il managed {
    .maxstack 3

    // load method pointer to event handler and create delegate
    ldarg.0
    dup    // duplicate top of stack
    ldftn instance void Counter::OnStartStop(int32)
    newobj          instance void StartStopEventHandler::.ctor(class
System.Object, int32)
    stfld class StartStopEventHandler Counter::startStopEventHandler

    // start listening to StartStopEvent
    ldarg.0
    ldfld class IStartStopEventSource Counter::startStopEventSource
    ldarg.0
    ldfld class StartStopEventHandler Counter::startStopEventHandler
    callvirt instance void
IStartStopEventSource::add_StartStopEvent(class
StartStopEventHandler)

    ret
}

/* Event handler for the StartStopEvent
 * action = 0, stop counter
 * action = 1, start counter */
.method public hidebysig instance void OnStartStop(int32 action) il
managed {
    ldarg action

```

```
    brtrue      start
    // stop the counter
    ldarg.0
    call  instance void Counter::Stop()
    br     done
start:
    // start the counter
    ldarg.0
    call  instance void Counter::Start()
done:
    ret
}

/* starts the counter */
.method private hidebysig instance void Start() il managed {
    // refresh
    ldarg.0
    ldfld class Count Counter::count
    callvirt instance void Count::refresh_Count()

    // check if > 0
    // 0 indicates some error
    ldarg.0
    ldfld class Count Counter::count
    callvirt instance int32 Count::get_Count()
    ldc.i4.0
    ble      do_not_start

    // start the timer
    ldarg.0
    ldfld class [System.Timers]System.Timers.Timer Counter::timer
    call  instance void [System.Timers]System.Timers.Timer::Start()
    br     done

do_not_start:
    // fire the TimeUpEvent to make sure all listeners are in stop
    state
    ldarg.0
    callvirt instance void Counter::fire_TimeUpEvent()
done:
    ret
}
```

```

/* stops the counter */
.method private hidebysig instance void Stop() il managed {
    // stop the timer
    ldarg.0
    ldfld class [System.Timers]System.Timers.Timer Counter::timer
    call instance void [System.Timers]System.Timers.Timer::Stop()
    ret
}

/* Event handler for the Tick event.
 * Subclasses may override event handler. */
.method virtual newslot family hidebysig instance void OnTick(class
System.Object, class [mscorlib]System.EventArgs) il managed {
    .maxstack 3
    // subtract one from counter
    ldarg.0
    ldfld class Count Counter::count
    dup
    callvirt instance int32 Count::get_Count()
    ldc.i4.1
    sub
    callvirt instance void Count::set_Count(int32)

    // check whether to stop
    ldarg.0
    ldfld class Count Counter::count
    callvirt instance int32 Count::get_Count()
    ldc.i4.0
    ble     time_up

    br     done
time_up:
    // stop timer
    ldarg.0
    call instance void Counter::Stop()
    ldarg.0
    callvirt instance void Counter::fire_TimeUpEvent()
done:
    ret
}

/**** Definition of the TimeUp event ****/

// Event is implemented using delegate TimeUpEventHandler

/* Declaration of the event */

```

- 40 -

```
.event TimeUpEventHandler TimeUpEvent {
    .addon instance void add_TimeUp(class TimeUpEventHandler
    'handler')
    .removeon instance void remove_TimeUp(class TimeUpEventHandler
    'handler')
    .fire instance void fire_TimeUpEvent()
}

/* add a listener */
.method virtual newslot public specialname hidebysig instance void
add_TimeUp(class TimeUpEventHandler 'handler') il managed {
    .maxstack 4
    ldarg.0
    dup
    ldfld class TimeUpEventHandler Counter::timeUpEventHandler
    ldarg 'handler'
    call class[mscorlib]System.Delegate
    [mscorlib]System.Delegate::Combine(class [mscorlib]System.Delegate,
    class [mscorlib]System.Delegate)
    castclass TimeUpEventHandler
    stfld class TimeUpEventHandler Counter::timeUpEventHandler
    ret
}

/* remove a listener */
.method virtual newslot public specialname hidebysig instance void
remove_TimeUp(class TimeUpEventHandler 'handler') il managed {
    .maxstack 4
    ldarg.0
    dup
    ldfld class TimeUpEventHandler Counter::timeUpEventHandler
    ldarg 'handler'
    call class[mscorlib]System.Delegate
    [mscorlib]System.Delegate::Remove(class [mscorlib]System.Delegate,
    class [mscorlib]System.Delegate)
    castclass TimeUpEventHandler
    stfld class TimeUpEventHandler Counter::timeUpEventHandler
    ret
}

/* fire the event */
.method virtual newslot family specialname hidebysig instance void
fire_TimeUpEvent() il managed {
    .maxstack 3
    ldarg.0
    ldfld class TimeUpEventHandler Counter::timeUpEventHandler
    callvirt instance void TimeUpEventHandler::Invoke()
}
```

```

        ret
    }

    /** End of the definition of the TimeUp event */

} // end of class Counter

/*****
Class: BeepingCounter
*****/

/* Modifies the behavior of the counter to beep every seconds and play
final time up sound
* Uses Windows DLL user32 to create the beeps. */
.class public auto autochar BeepingCounter extends Counter {

    /* constructor */
    .method public rtspecialname specialname hidebysig instance void
    .ctor(class IStartStopEventSource startStopEventSource, class Count
    count) il managed {
        // call super constructor
        ldarg.0
        ldarg startStopEventSource
        ldarg count
        call instance void Counter::.ctor(class IStartStopEventSource,
        class Count)
        ret
    }

    /* Overrides Counter::OnTick.
    * Creates beeps when the tick is handled */
    .method virtual family hidebysig instance void OnTick(class
    System.Object object, class [mscorlib]System.EventArgs eventArgs) il
    managed {
        // super call
        ldarg.0
        ldarg object
        ldarg eventArgs
        call instance void Counter::OnTick(class System.Object, class
        [mscorlib]System.EventArgs)

        // check whether time is up and create appropriate beep
        ldarg.0
        dup // for beep call
        ldfld class Count Counter::count
        callvirt instance int32 Count::get_Count()
        ldc.i4.0
        ble final_beep
    }
}

```

```
// normal beep
ldc.i4.0
br        beep_it
final_beep:
// final beep
ldc.i4.1
beep_it:
// call the beep method
callvirt instance void BeepingCounter::Beep(bool)
ret
}

/* Uses User32 to create a beep. May be overridden by subclasses to
 * have better beeps.
 * finalBeep = 0, counter is still counting
 * finalBeep = 1, counter is done, final beep */
.method virtual newslot private hidebysig instance void Beep(bool
finalBeep) il managed {
    ldarg finalBeep
    brtrue    'final'
    ldc.i4.0    // load value for normal beep
    br        continue
'final':
    ldc.i4      48    // load value for final beep
continue:
    call  int8 User32::MessageBeep(unsigned int32)
    pop
    ret
}
} // end of class BeepingCounter

/*****
/*          Class: User32          */
*****/

/* Uses PInvoke to call functionality in user32.dll.
 * (change to "user.dll" if Windows9x is used).
 * Class is abstract such that no instances can be created. All methods
   are static.
 * Class is sealed to prevent subtyping */
.class abstract sealed public auto autochar User32 extends
[mscorlib]System.Object {
```

```

/* declaration of the method MessageBeep in user32.dll */
.method public static pinvokeimpl("user32.dll" cdecl) int8
MessageBeep(unsigned int32) native unmanaged {}
}

/*****
/*      Delegate: StartStopEventHandler      */
*****/

/* Defines delegate for StartStopEvent */
.class private sealed auto autochar StartStopEventHandler extends
[mscorlib]System.MulticastDelegate {

    .method public specialname rtspecialname hidebysig instance void
    .ctor(class System.Object object, int32 'method') runtime managed {}

    .method virtual newslot public hidebysig instance void Invoke(int32
action) runtime managed {}

    .method virtual newslot public hidebysig instance class
['mscorlib']System.IAsyncResult BeginInvoke(int32 action,
        class ['mscorlib']System.AsyncCallback callback, class
System.Object object) runtime managed {}

    .method virtual newslot public hidebysig instance void
EndInvoke(class ['mscorlib']System.IAsyncResult result) runtime
managed {}
} // end of delegate StartStopEventHandler

/*****
/*      Delegate: TimeUpHandler      */
*****/

/* Defines delegate for TimeUpEvent */
.class private sealed auto autochar TimeUpEventHandler extends
[mscorlib]System.MulticastDelegate {

    .method public specialname rtspecialname hidebysig instance void
    .ctor(class System.Object object, int32 'method') runtime managed {}

    .method virtual newslot public hidebysig instance void Invoke()
runtime managed {}

    .method virtual newslot public newslot hidebysig instance class
['mscorlib']System.IAsyncResult BeginInvoke(
        class ['mscorlib']System.AsyncCallback callback, class
System.Object object) runtime managed {}
}

```

- 44 -

```
.method virtual newslot public hidebysig instance void  
EndInvoke(class ['mscorlib']System.IAsyncResult result) runtime  
managed {}  
} // end of delegate TimeUpEventHandler
```

A.3.4. CountdownSecondsLabel.cs

```
using System.Windows.Forms;  
using System.Drawing;  
  
/* Represents a label that displays the text "Seconds left" */  
class SecondsLabel : System.Windows.Forms.Label {  
  
    /* constructor */  
    internal SecondsLabel(System.Windows.Forms.Form parent) {  
        Initialize();  
        // add to form  
        parent.Controls.Add(this);  
    }  
  
    /* initializes this label */  
    private void Initialize() {  
        // set the position  
        Location = new Point(200,100);  
  
        // set anchor  
        Anchor = System.Windows.Forms.AnchorStyles.TopBottom;  
  
        // set the size of the label  
        Size = new Size(200,30);  
  
        // set the color  
        ForeColor = System.Drawing.Color.DarkRed;  
  
        // set the font height  
        Font = new System.Drawing.Font("Verdana",15);  
  
        // set the text of the label  
        Text = "Seconds left";  
    }  
}
```

A.3.5. CountdownErrorLabel.cs

```
using System.Windows.Forms;  
using System.Drawing;
```


Standard master

Beta 1 Release

```
/* Label that shows errpr message at top of window in big, red
   characters */
```

```
class ErrorLabel : System.Windows.Label {

    /* constructor */
    internal ErrorLabel(System.Windows.Form parent) {
        Initialize();
        // add to form
        parent.Controls.Add(this);
    }

    /* initializes this label */
    private void Initialize() {
        // set the position
        Location = new Point(10,25);

        // set anchor
        Anchor = System.Windows.AnchorStyles.TopBottom;

        // set the size of the label
        Size = new Size(400,50);

        // set the color
        ForeColor = System.Drawing.Color.Red;

        // set the font height
        Font = new System.Drawing.Font("Arial Black",20);

        // set the text of the label
        Text = "";
    }
}
```

A.3.6. CountdownErrorLabel.cpp

```
#using <mscorlib.dll>
#using <System.dll>
#using <System.Windows.dll>
#using <System.Drawing.dll>
#using <Microsoft.Win32.Interop.dll>

/* Label that shows errpr message at top of window in big, red
   characters
   __gc declares that instances of the class will be garbage collected
   */
__gc public class ErrorLabel : public System::WinForms::Label {

public:
    /* constructor */
```

- 46 -

```
ErrorLabel(System::WinForms::Form* parent) :
System::WinForms::Label() {
    Initialize();
    // add to form
    parent->get_Controls()->Add(this);
}

private:
    /* initializes this label */
    virtual void Initialize() {
        // set position
        set_Location(System::Drawing::Point(10,25));

        // set anchor
        set_Anchor(System::WinForms::AnchorStyles::TopBottom);

        // set the size of the label
        set_Size(System::Drawing::Size(400,50));

        // set the color
        set_ForeColor(System::Drawing::Color::get_Red());

        // set the font height
        set_Font(new System::Drawing::Font("Arial Black",20));

        // set the text of the label
        set_Text("");
    }
};
```

A.3.7. compile.bat

```
@echo off
csc /t:module CountdownSecondsLabel.cs /r:System.dll
/r:System.WinForms.dll /r:System.Drawing.dll
/r:Microsoft.Win32.Interop.dll
rem if C++ is specified, create C++ DLL, otherwise create C# DLL
if "C++"=="%1" (cl /CLR /LD CountdownErrorLabel.cpp /link
/OUT:CountDownErrorLabel.dll) else (csc /t:module
CountDownErrorLabel.cs /r:System.dll /r:System.WinForms.dll
/r:System.Drawing.dll /r:Microsoft.Win32.Interop.dll)
ilasm Counter.il /dll
ilasm CountdownComponents.il /dll
ilasm Countdown.il
```

Annex B. Informative: List of ILASM Keywords

This annex presents all keywords of the hypothetical CIL assembler *ilasm*. Note that ILASM is case-sensitive.

.addon	.override	ann.live	bne.un
.algorithm	.pack	ann.phi	bne.un.s
.assembly	.param	ann.ref	bool
.backing	.permission	ann.ref.s	box
.blob	.processor	ansi	br
.capability	.property	any	br.s
.cctor	.removeon	arglist	break
.class	.set	array	brfalse
.comtype	.size	as	brfalse.s
.corflags	.subsystem	assembly	brtrue
.ctor	.title	assert	brtrue.s
.custom	.try	at	bstr
.data	.ver	auto	bytearray
.data	.vtable	autochar	byvalstr
.emitbyte	.vtbody	beq	call
.entrypoint	.vtbody	beq.s	calli
.event	.zeroinit	bge	callvirt
.field	#line	bge.s	castclass
.fire	abstract	bge.un	catch
.get	add	bge.un.s	cdecl
.hash	add.ovf	bgt	cdecl
.implicitcom	add.ovf.un	bgt.s	ceq
.line	and	bgt.un	cf
.locale	ann.arg	bgt.un.s	cgt
.locals	ann.call	ble	cgt.un
.manifestres	ann.catch	ble.s	char
.maxstack	ann.data	ble.un	ckfinite
.method	ann.data.s	ble.un.s	class
.module	ann.dead	blob	clsid
.namespace	ann.def	blt	clt
.originator	ann.hoisted	blt.s	clt.un
.os	ann.hoisted_call	blt.un	const
.other	ann.lab	blt.un.s	conv.i

conv.i1	cpobj	hidebysig	ldarg.s
conv.i2	currency	hresult	ldarga
conv.i4	decimal	idispatch	ldarga.s
conv.i8	default	il	ldc.i4
conv.ovf.i	demand	implements	ldc.i4.0
conv.ovf.i.un	deny	implicitres	ldc.i4.1
conv.ovf.i1	div	import	ldc.i4.2
conv.ovf.i1.un	div.un	import	ldc.i4.3
conv.ovf.i2	dup	in	ldc.i4.4
conv.ovf.i2.un	endfilter	inheritcheck	ldc.i4.5
conv.ovf.i4	endfinally	init	ldc.i4.6
conv.ovf.i4.un	enum	initblk	ldc.i4.7
conv.ovf.i8	error	initobj	ldc.i4.8
conv.ovf.i8.un	explicit	initonly	ldc.i4.m1
conv.ovf.u	extends	instance	ldc.i4.s
conv.ovf.u.un	famandassem	int	ldc.i8
conv.ovf.u1	family	int16	ldc.r4
conv.ovf.u1.un	famorassem	int32	ldc.r8
conv.ovf.u2	fastcall	int64	ldelem.i
conv.ovf.u2.un	fastcall	int8	ldelem.i1
conv.ovf.u4	fault	interface	ldelem.i2
conv.ovf.u4.un	file	internalcall	ldelem.i4
conv.ovf.u8	filetime	isinst	ldelem.i8
conv.ovf.u8.un	filter	iunkown	ldelem.r4
conv.r.un	finally	jmp	ldelem.r8
conv.r4	finaly	jmpj	ldelem.ref
conv.r8	float	lateinit	ldelem.u1
conv.u	float32	lcid	ldelem.u2
conv.u1	float64	ldarg	ldelema
conv.u2	forwardref	ldarg.0	ldfld
conv.u4	fromunmanaged	ldarg.1	ldflda
conv.u8	fullorigin	ldarg.2	ldftn
cpblk	handler	ldarg.3	ldind.i

*Standard master**Beta 1 Release*

ldind.i1	managed	pop	stelem.i1
ldind.i2	marshal	prejitdeny	stelem.i2
ldind.i4	method	prejitgrant	stelem.i4
ldind.i8	mkrefany	private	stelem.i8
ldind.r4	modopt	privatescope	stelem.r4
ldind.r8	modreq	public	stelem.r8
ldind.ref	mul	record	stelem.ref
ldind.u1	mul.ovf	refanytype	stfld
ldind.u2	mul.ovf.un	refanyval	stind.i
ldlen	native	rem	stind.i1
ldloc	neg	rem.un	stind.i2
ldloc.0	nested	reqmin	stind.i4
ldloc.1	newarr	reqopt	stind.i8
ldloc.2	newobj	reqrefuse	stind.r4
ldloc.3	newslot	request	stind.r8
ldloc.s	noappdomain	ret	stind.ref
ldloca	noinlining	rethrow	stloc
ldloca.s	nomachine	retval	stloc.0
ldnull	nomangle	rtspecialname	stloc.1
ldobj	nometadata	runtime	stloc.2
ldsfld	nop	safearray	stloc.3
ldsflda	noprocess	sealed	stloc.s
ldstr	not	sequential	stobj
ldtoken	not_in_gc_heap	serializable	storage
ldvirtftn	notserialized	shl	stream
leave	null	shr	struct
leave.s	objectref	shr.un	stsfld
linkcheck	ole	sizeof	sub
literal	opt	specialname	sub.ovf
localloc	optil	starg	sub.ovf.un
lpstr	or	starg.s	switch
lpstruct	out	static	synchronized
lptstr	permitonly	stdcall	syschar
lpvoid	pinbokeimpl	stdcall	sysstring
lpwstr	pinned	stelem.i	tail.

tbstr
thiscall
thiscall
throw
tls
to
typedref
unaligned.
unbox
unicode
unmanaged
unmanagedexp
unsigned
value
vararg
variant
vector
virtual
void
volatile.
wchar
winapi
with
xor

Annex C. Informative: ILASM Complete Grammar

Lexical tokens

ID - C style alphaNumeric identifier (e.g. Hello_There2)

QSTRING - C style quoted string (e.g. "hi\n")

SQSTRING - C style singlely quoted string(e.g. 'hi')

INT32 - C style 32 bit integer (e.g. 235, 03423, 0x34FFF)

INT64 - C style 64 bit integer (e.g. -2353453636235234, 0x34FFFFFFFFFFFF)

FLOAT64 - C style floating point number (e.g. -0.2323, 354.3423, 3435.34E-5)

INSTR_* - IL instructions of a particular class (see opcode.def).

```
START          : decls
                ;

decls           : /* EMPTY */
                | decls decl
                ;

decl            : classHead '{' classDecls '}'
                | namespaceHead '{' decls '}'
                | methodHead methodDecls '}'
                | fieldDecl
                | dataDecl
                | vtableDecl
                | vtfixupDecl
                | extSourceSpec
                | fileDecl
                | exelocDecl
                | assemblyHead '{' assemblyDecls '}'
                | assemblyRefHead '{' assemblyRefDecls '}'
                | comtypeHead '{' comtypeDecls '}'
                | manifestResHead '{' manifestResDecls '}'
                | moduleHead
                | secDecl
                | customAttrDecl
                | '.subsystem' int32
                | '.corflags' int32
                ;

compQstring     : QSTRING
                | compQstring '+' QSTRING
                ;

customAttrDecl  : '.custom' customType
                | '.custom' customType '=' compQstring
                | customHead bytes ')'
                ;

moduleHead      : '.module'
                | '.module' name1
                | '.module' 'extern' name1
                ;

vtfixupDecl     : '.vtfixup' '[' int32 ']' vtfixupAttr 'at' id
                ;
```

```
vtfixupAttr      : /* EMPTY */
                  | vtfixupAttr 'int32'
                  | vtfixupAttr 'int64'
                  | vtfixupAttr 'fromunmanaged'
                  | vtfixupAttr 'callmostderived'
                  ;

vtableDecl       : vtableHead bytes ')'
                  ;

vtableHead       : '.vtable' '=' '('
                  ;

namespaceHead    : '.namespace' name1
                  ;

classHead        : '.class' classAttr id extendsClause implClause
                  ;

classAttr        : /* EMPTY */
                  | classAttr 'public'
                  | classAttr 'private'
                  | classAttr 'value'
                  | classAttr 'unmanaged'
                  | classAttr 'not_in_gc_heap'
                  | classAttr 'interface'
                  | classAttr 'sealed'
                  | classAttr 'abstract'
                  | classAttr 'auto'
                  | classAttr 'sequential'
                  | classAttr 'explicit'
                  | classAttr 'ansi'
                  | classAttr 'unicode'
                  | classAttr 'autochar'
                  | classAttr 'import'
                  | classAttr 'serializable'
                  | classAttr 'nested' 'public'
                  | classAttr 'nested' 'private'
                  | classAttr 'nested' 'family'
                  | classAttr 'nested' 'assembly'
                  | classAttr 'nested' 'famandassem'
                  | classAttr 'nested' 'famorassem'
                  | classAttr 'lateinit'
                  | classAttr 'specialname'
                  | classAttr 'rtspecialname'
                  ;

extendsClause    : /* EMPTY */
                  | 'extends' className
                  ;

implClause       : /* EMPTY */
                  | 'implements' classNames
                  ;

classNames      : classNames ',' className
                  | className
                  ;

classDecls      : /* EMPTY */
                  | classDecls classDecl
                  ;
```



```
classDecl      : methodHead methodDecls '}'
                | classHead '{' classDecls '}'
                | eventHead '{' eventDecls '}'
                | propHead '{' propDecls '}'
                | fieldDecl
                | dataDecl
                | secDecl
                | extSourceSpec
                | customAttrDecl
                | '.size' int32
                | '.pack' int32
                | comtypeHead '{' comtypeDecls '}'
                | exportHead '{' comtypeDecls '}'
                | '.override' typeSpec '::' methodName 'with' callConv type
typeSpec '::' methodName '(' sigArgs0 ')'
;

fieldDecl      : '.field' repeatOpt fieldAttr type id atOpt initOpt
;

atOpt          : /* EMPTY */
                | 'at' id
;

initOpt        : /* EMPTY */
                | '=' fieldInit
;

repeatOpt      : /* EMPTY */
                | '[' int32 ']'
;

customHead     : '.custom' customType '=' '('
;

customType     : typeSpec
                | callConv type typeSpec '::' methodName '(' sigArgs0 ')'
                | callConv type methodName '(' sigArgs0 ')'
;

eventHead      : '.event' eventAttr typeSpec id
                | '.event' eventAttr id
;

eventAttr      : /* EMPTY */
                | eventAttr 'rtspecialname' /**/
                | eventAttr 'specialname'
;

eventDecls     : /* EMPTY */
                | eventDecls eventDecl
;

eventDecl      : '.addon' callConv type typeSpec '::' methodName '('
sigArgs0 ')'
                | '.addon' callConv type methodName '(' sigArgs0 ')'
                | '.removeon' callConv type typeSpec '::' methodName '('
sigArgs0 ')'
                | '.removeon' callConv type methodName '(' sigArgs0 ')'
                | '.fire' callConv type typeSpec '::' methodName '(' sigArgs0
')'
                | '.fire' callConv type methodName '(' sigArgs0 ')'
```

```
sigArgs0 ')'
| '.other' callConv type typeSpec '::' methodName '('
| '.other' callConv type methodName '(' sigArgs0 ')'
| extSourceSpec
| customAttrDecl
;

propHead
: '.property' propAttr callConv type id '(' sigArgs0 ')'
;

propAttr
: /* EMPTY */
| propAttr 'rtspecialname' /**/
| propAttr 'specialname'
;

propDecls
: /* EMPTY */
| propDecls propDecl
;

propDecl
: '.set' callConv type typeSpec '::' methodName '(' sigArgs0
'| '.set' callConv type methodName '(' sigArgs0 ')'
'| '.get' callConv type typeSpec '::' methodName '(' sigArgs0
'| '.get' callConv type methodName '(' sigArgs0 ')'
'| '.other' callConv type typeSpec '::' methodName '('
sigArgs0 ')'
'| '.other' callConv type methodName '(' sigArgs0 ')'
'| '.backing' type id
| customAttrDecl
| extSourceSpec
;

methodHeadPart1
: '.method'
;

methodHead
: methodHeadPart1 methAttr callConv paramAttr type methodName
 '(' sigArgs0 ')' implAttr '{'
| methodHeadPart1 methAttr callConv paramAttr type 'marshal'
 '(' nativeType ')' methodName '(' sigArgs0 ')' implAttr '{'
;

methAttr
: /* EMPTY */
| methAttr 'static'
| methAttr 'public'
| methAttr 'private'
| methAttr 'family'
| methAttr 'final'
| methAttr 'specialname'
| methAttr 'virtual'
| methAttr 'abstract'
| methAttr 'assembly'
| methAttr 'famandassem'
| methAttr 'famorassem'
| methAttr 'privatescope'
| methAttr 'hidebysig'
| methAttr 'newslot'
| methAttr 'rtspecialname' /**/
| methAttr 'unmanagedexp'
| methAttr 'pinvokeimpl' '(' compQstring 'as' compQstring
pinvAttr ')'
```

```
| methAttr 'pinvokeimpl' '(' compQstring pinvAttr ')'
| methAttr 'pinvokeimpl' '(' pinvAttr ')'
;

pinvAttr      : /* EMPTY */
| pinvAttr 'nomangle'
| pinvAttr 'ansi'
| pinvAttr 'unicode'
| pinvAttr 'autochar'
| pinvAttr 'ole'
| pinvAttr 'lasterr'
| pinvAttr 'winapi'
| pinvAttr 'cdecl'
| pinvAttr 'stdcall'
| pinvAttr 'thiscall'
| pinvAttr 'fastcall'
;

methodName   : '.ctor'
| '.cctor'
| name1
;

paramAttr     : /* EMPTY */
| paramAttr '[' 'in' ']'
| paramAttr '[' 'out' ']'
| paramAttr '[' 'opt' ']'
| paramAttr '[' 'lcid' ']'
| paramAttr '[' 'retval' ']'
| paramAttr '[' 'int32 ']'
;

fieldAttr     : /* EMPTY */
| fieldAttr 'static'
| fieldAttr 'public'
| fieldAttr 'private'
| fieldAttr 'family'
| fieldAttr 'initonly'
| fieldAttr 'rtspecialname' /**/
| fieldAttr 'specialname'
/* commented out because PInvoke for fields is
not supported by EE
pinvAttr ')'
| fieldAttr 'pinvokeimpl' '(' compQstring 'as' compQstring
| fieldAttr 'pinvokeimpl' '(' compQstring pinvAttr ')'
| fieldAttr 'pinvokeimpl' '(' pinvAttr ')'
*/
| fieldAttr 'marshal' '(' nativeType ')'
| fieldAttr 'assembly'
| fieldAttr 'famandassem'
| fieldAttr 'famorassem'
| fieldAttr 'privatescope'
| fieldAttr 'literal'
| fieldAttr 'notserialized'
;

implAttr      : /* EMPTY */
| implAttr 'native'
| implAttr 'il'
| implAttr 'optil'
| implAttr 'managed'
| implAttr 'unmanaged'
| implAttr 'forwardref'
| implAttr 'ole'
```

```
        | implAttr 'runtime'
        | implAttr 'internalcall'
        | implAttr 'synchronized'
        | implAttr 'noinlining'
        ;

localsHead      : '.locals'
                ;

methodDecl      : '.emitbyte' int32
                | sehBlock
                | '.maxstack' int32
                | localsHead '(' sigArgs0 ')'
                | localsHead 'init' '(' sigArgs0 ')'
                | '.entrypoint'
                | '.zeroinit'
                | dataDecl
                | instr
                | id ':'
                | secDecl
                | extSourceSpec
                | customAttrDecl
                | '.vtable' int32 ':' int32
                | '.override' typeSpec '::' methodName
                | scopeBlock
                | '.param' '[' int32 ']'
                | '.param' '[' int32 ']' '=' fieldInit
                ;

scopeBlock      : scopeOpen methodDecls '}'
                ;

scopeOpen       : '{'
                ;

sehBlock        : tryBlock sehClauses
                ;

sehClauses      : sehClause sehClauses
                | sehClause
                ;

tryBlock        : tryHead scopeBlock
                | tryHead id 'to' id
                | tryHead int32 'to' int32
                ;

tryHead         : '.try'
                ;

sehClause       : catchClause handlerBlock
                | filterClause handlerBlock
                | finallyClause handlerBlock
                | faultClause handlerBlock
                ;

filterClause    : filterHead scopeBlock
                | filterHead id
                | filterHead int32
                ;
```

```
filterHead      : 'filter'
                 ;

catchClause     : 'catch' className
                 ;

finallyClause   : 'finally'
                 ;

faultClause     : 'fault'
                 ;

handlerBlock    : scopeBlock
                 | 'handler' id 'to' id
                 | 'handler' int32 'to' int32
                 ;

methodDecls     : /* EMPTY */
                 | methodDecls methodDecl
                 ;

dataDecl       : ddHead ddBody
                 ;

ddHead         : '.data' tls id '='
                 | '.data' tls
                 ;

tls            : /* EMPTY */
                 | 'tls'
                 ;

ddBody         : '{' ddItemList '}'
                 | ddItem
                 ;

ddItemList     : ddItem ',' ddItemList
                 | ddItem
                 ;

ddItemCount    : /* EMPTY */
                 | '[' int32 ']'
                 ;

ddItem         : 'char' '*' '(' compQstring ')'
                 | 'wchar' '*' '(' compQstring ')'
                 | '&' '(' id ')'
                 | bytearrayhead bytes
                 | 'float32' '(' float64 ')' ddItemCount
                 | 'float64' '(' float64 ')' ddItemCount
                 | 'int64' '(' int64 ')' ddItemCount
                 | 'int32' '(' int32 ')' ddItemCount
                 | 'int16' '(' int32 ')' ddItemCount
                 | 'int8' '(' int32 ')' ddItemCount
                 | 'float32' ddItemCount
                 | 'float64' ddItemCount
                 | 'int64' ddItemCount
                 | 'int32' ddItemCount
                 | 'int16' ddItemCount
                 | 'int8' ddItemCount
                 ;

fieldInit      : 'float32' '(' float64 ')'
```

```
| 'float64' '(' float64 ')'
| 'float32' '(' int64 ')'
| 'float64' '(' int64 ')'
| 'int64' '(' int64 ')'
| 'int32' '(' int64 ')'
| 'int16' '(' int64 ')'
| 'int8' '(' int64 ')'
| compQstring
| 'wchar' '*' '(' compQstring ')'
| bytearrayhead bytes ')'
;

bytearrayhead      : 'bytearray' '('

bytes              : HEXBYTE
                  | bytes HEXBYTE
                  ;

instr_r_head       : INSTR_R '('
                  ;

methodSpec         : 'method'
                  ;

instr              : INSTR_NONE
                  | INSTR_VAR int32
                  | INSTR_VAR id
                  | INSTR_I int32
                  | INSTR_I8 int64
                  | INSTR_R float64
                  | INSTR_R int64
                  | instr_r_head bytes ')'
                  | INSTR_BRTARGET int32
                  | INSTR_BRTARGET id
                  | INSTR_METHOD callConv type typeSpec '::' methodName '('
sigArgs0 ')'
                  | INSTR_METHOD callConv type methodName '(' sigArgs0 ')'
                  | INSTR_FIELD type typeSpec '::' id
                  | INSTR_FIELD type id
                  | INSTR_TYPE typeSpec
                  | INSTR_STRING compQstring
                  | INSTR_STRING bytearrayhead bytes ')'
                  | INSTR_SIG callConv type '(' sigArgs0 ')'
                  | INSTR_RVA id
                  | INSTR_RVA int32
                  | INSTR_TOK methodSpec callConv type typeSpec '::' methodName
 '(' sigArgs0 ')'
                  | INSTR_TOK methodSpec callConv type methodName '(' sigArgs0
 ')'
                  | INSTR_TOK 'field' type typeSpec '::' id
                  | INSTR_TOK 'field' type id
                  | INSTR_TOK typeSpec
                  | INSTR_SWITCH '(' labels ')'
                  | INSTR_PHI int16s
                  ;

sigArgs0           : /* EMPTY */
                  | sigArgs1
                  ;

sigArgs1           : sigArg
                  | sigArgs1 ',' sigArg
                  ;
```

```
sigArg      : '...'
              | paramAttr type
              | paramAttr type id
              | paramAttr type 'marshal' '(' nativeType ')'
              | paramAttr type 'marshal' '(' nativeType ')' id
              ;

name1       : id
              | DOTTEDNAME
              | name1 '.' name1
              ;

className   : '[' name1 ']' slashedName
              | '[' '.module' name1 ']' slashedName
              | slashedName
              ;

slashedName : name1
              | slashedName '/' name1
              ;

typeSpec    : className
              | '[' name1 ']'
              | '[' '.module' name1 ']'
              | type
              ;

callConv    : 'instance' callConv
              | 'explicit' callConv
              | callKind
              ;

callKind    : /* EMPTY */
              | 'default'
              | 'vararg'
              | 'unmanaged' 'cdecl'
              | 'unmanaged' 'stdcall'
              | 'unmanaged' 'thiscall'
              | 'unmanaged' 'fastcall'
              ;

nativeType  : /* EMPTY */
              | 'custom' '(' compQstring ',' compQstring ',' compQstring
',' compQstring ')'
              | 'fixed' 'sysstring' '[' int32 ']'
              | 'fixed' 'array' '[' int32 ']'
              | 'variant'
              | 'currency'
              | 'syschar'
              | 'void'
              | 'bool'
              | 'int8'
              | 'int16'
              | 'int32'
              | 'int64'
              | 'float32'
              | 'float64'
              | 'error'
              | 'unsigned' 'int8'
              | 'unsigned' 'int16'
              | 'unsigned' 'int32'
              | 'unsigned' 'int64'
              | nativeType '*'
```

```
nativeType '[' ' ']  
nativeType '[' int32 ']  
nativeType '[' '.size' '.param' '=' int32 ']  
nativeType '[' '.size' '.param' '=' int32 '*' int32 ']  
'decimal'  
'date'  
'bstr'  
'lpstr'  
'lpwstr'  
'lptstr'  
'objectref'  
'iunknown'  
'idispatch'  
'struct'  
'interface'  
'safearray' variantType  
'int'  
'unsigned' 'int'  
'nested' 'struct'  
'byvalstr'  
'ansi' 'bstr'  
'tbstr'  
'variant' 'bool'  
methodSpec  
'lpvoid'  
'as' 'any'  
'float'  
'lpstruct'  
;
```

```
variantType : /* EMPTY */  
            'null'  
            'variant'  
            'currency'  
            'void'  
            'bool'  
            'int8'  
            'int16'  
            'int32'  
            'int64'  
            'float32'  
            'float64'  
            'unsigned' 'int8'  
            'unsigned' 'int16'  
            'unsigned' 'int32'  
            'unsigned' 'int64'  
            '*'  
            variantType '[' ' ']  
            variantType 'vector'  
            variantType '&  
            'decimal'  
            'date'  
            'bstr'  
            'lpstr'  
            'lpwstr'  
            'iunknown'  
            'idispatch'  
            'safearray'  
            'int'  
            'unsigned' 'int'  
            'error'  
            'hresult'  
            'carray'  
            'userdefined'
```



```

| 'record'
| 'filetime'
| 'blob'
| 'stream'
| 'storage'
| 'streamed_object'
| 'stored_object'
| 'blob_object'
| 'cf'
| 'clsid'
;

type : 'class' className
      | 'value' 'class' className
      | type '[' '?' ']'
      | type '[' ']'
      | type '[' bounds1 ']'
      | type '&'
      | type '*'
      | type '%'
      | type 'pinned'
      | type 'modreq' '(' className ')'
      | type 'modopt' '(' className ')'
      | '!' int32
      | methodSpec callConv type '*' '(' sigArgs0 ')'
      | 'typedref'
      | 'wchar'
      | 'char'
      | 'void'
      | 'bool'
      | 'int8'
      | 'int16'
      | 'int32'
      | 'int64'
      | 'float32'
      | 'float64'
      | 'unsigned' 'int8'
      | 'unsigned' 'int16'
      | 'unsigned' 'int32'
      | 'unsigned' 'int64'
      | 'native' 'int'
      | 'native' 'unsigned' 'int'
      | 'native' 'float'
;

bounds1 : bound
        | bounds1 ',' bound
;

bound : /* EMPTY */
      | '...'
      | int32
      | int32 '...' int32
      | int32 '...'
;

labels : /* empty */
        | id ',' labels
        | int32 ',' labels
        | id
        | int32
;
```

```
id          : ID
            | SQSTRING
            ;

int16s      : /* EMPTY */
            | int16s int32
            ;

int32       : INT64
            ;

int64       : INT64
            ;

float64     : FLOAT64
            | 'float32' '(' int32 ')'
            | 'float64' '(' int64 ')'
            ;

secDecl     : '.permission' secAction className '(' nameValPairs ')'
            | '.capability' secAction SQSTRING
            | capHead bytes ')'
            ;

capHead     : '.capability' secAction '=' '('

nameValPairs : nameValPair
            | nameValPair ',' nameValPairs
            ;

nameValPair : SQSTRING '=' SQSTRING
            ;

secAction   : 'request'
            | 'demand'
            | 'assert'
            | 'deny'
            | 'permitonly'
            | 'linkcheck'
            | 'inheritcheck'
            | 'reqmin'
            | 'reqopt'
            | 'reqrefuse'
            | 'prejitgrant'
            | 'prejitdeny'
            | 'noncasdemand'
            | 'noncaslinkdemand'
            | 'noncasinheritance'
            ;

extSourceSpec : '.line' int32 SQSTRING
            | '.line' int32
            | P_LINE int32 QSTRING
            ;

fileDecl    : '.file' fileAttr name1 hashHead bytes ')'
            | '.file' fileAttr name1
            ;

fileAttr    : /* EMPTY */
            | fileAttr 'readonly'
            | fileAttr 'nometadata'
            ;
```

```
hashHead      : '.hash' '=' '('
               ;

exelocDecl    : '.exeloc' name1 '(' compQstring ')' 'at' compQstring
               | '.exeloc' name1 'at' compQstring
               ;

assemblyHead  : '.assembly' asmAttr name1 'as' compQstring
               | '.assembly' asmAttr name1
               ;

asmAttr       : /* EMPTY */
               | asmAttr 'implicitcom'
               | asmAttr 'implicitres'
               | asmAttr 'noappdomain'
               | asmAttr 'noprocess'
               | asmAttr 'nomachine'
               ;

assemblyDecls : /* EMPTY */
               | assemblyDecls assemblyDecl
               ;

assemblyDecl  : '.title' compQstring '(' compQstring ')'
               | '.title' compQstring
               | '.hash' 'algorithm' int32
               | secDecl
               | asmOrRefDecl
               ;

asmOrRefDecl  : originatorHead bytes ')'
               | '.ver' int32 ':' int32 ':' int32 ':' int32
               | '.locale' compQstring
               | localeHead bytes ')'
               | '.processor' int32
               | '.os' int32 '.ver' int32 ':' int32
               | '.config' compQstring
               | configHead bytes ')'
               | customAttrDecl
               ;

originatorHead : '.originator' '=' '('
                ;

localeHead     : '.locale' '=' '('
                ;

configHead     : '.config' '=' '('
                ;

assemblyRefHead : '.assembly' 'extern' asmRefAttr name1 'as' compQstring
                  | '.assembly' 'extern' asmRefAttr name1
                  ;

asmRefAttr     : /* EMPTY */
                  | asmRefAttr 'fullorigin'
                  ;

assemblyRefDecls : /* EMPTY */
                  | assemblyRefDecls assemblyRefDecl
                  ;

assemblyRefDecl : hashHead bytes ')'
```

```
| '.exeloc' name1
| asmOrRefDecl
;

comtypeHead      : '.comtype' comtAttr name1 '(' compQstring ')'
| '.comtype' comtAttr name1
;

exportHead       : '.export' comtAttr name1 '(' compQstring ')'
| '.export' comtAttr name1
;

comtAttr         : /* EMPTY */
| comtAttr 'private'
| comtAttr 'public'
| comtAttr 'nested' 'public'
| comtAttr 'nested' 'private'
| comtAttr 'nested' 'family'
| comtAttr 'nested' 'assembly'
| comtAttr 'nested' 'famandassem'
| comtAttr 'nested' 'famorassem'
;

comtypeDecls     : /* EMPTY */
| comtypeDecls comtypeDecl
;

comtypeDecl      : '.file' name1
| '.assembly' 'extern' name1
| '.comtype' name1
| '.class' int32
| '.exeloc' name1
| customAttrDecl
;

manifestResHead  : '.manifestres' manresAttr name1 '(' compQstring ')'
| '.manifestres' manresAttr name1
;

manresAttr       : /* EMPTY */
| manresAttr 'public'
| manresAttr 'private'
;

manifestResDecls : /* EMPTY */
| manifestResDecls manifestResDecl
;

manifestResDecl  : '.mime' compQstring
| mimeHead bytes ')'
| '.file' name1 'at' int32
| '.assembly' 'extern' name1
| '.locale' compQstring
| localeHead bytes ')'
| customAttrDecl
;

mimeHead         : '.mime' '=' '('
```

Annex D. Informative: ILASM Instruction Syntax

While each section specifies the exact list of instructions that are included in a grammar class, this information is subject to change over time. The precise format of an instruction can be determined can be found by combining the information in the file **opcode.def** with the information in the following table:

Table 1: Instruction Syntax classes

Grammar Class	Format(s) Specified in opcode.def
<instr_brtarget>	InlineBrTarget, ShortInlineBrTarget
<instr_field>	InlineField
<instr_i>	InlineI, ShortInlineI
<instr_i8>	InlineI8
<instr_method>	InlineMethod
<instr_none>	InlineNone
<instr_phi>	InlinePhi
<instr_r>	InlineR, ShortInlineR
<instr_rva>	InlineRVA
<instr_sig>	InlineSig
<instr_string>	InlineString
<instr_switch>	InlineSwitch
<instr_tok>	InlineTok
<instr_type>	InlineType
<instr_var>	InlineVar, ShortInlineVar

D.1. Top-level Instruction Syntax

```

<instr> ::=
    <instr_brtarget> <int32>
  | <instr_brtarget> <label>
  | <instr_field> <type> [ <typeSpec> :: ] <id>
  | <instr_i> <int32>
  | <instr_i8> <int64>
  | <instr_method>
    <callConv> <type> [ <typeSpec> :: ] <methodName> ( <parameters> )
  | <instr_none>
  | <instr_phi> <int16>*
  | <instr_r> ( <bytes> ) // <bytes> represent the binary image of
                          // float or double (4 or 8 bytes, respectively)
  | <instr_r> <float64>
  | <instr_r> <int64> // integer is converted to float with possible
                     // loss of precision
  | <instr_sig> <callConv> <type> ( <parameters> )
  | <instr_string> bytearray ( <bytes> )
  | <instr_string> <QSTRING>
  | <instr_switch> ( <labels> )

```

```
| <instr_tok> field <type> [ <typeSpec> :: ] <id>
| <instr_tok> method
|   <callConv> <type> [ <typeSpec> :: ] <methodName> ( <parameters> )
| <instr_tok> <typeSpec>
| <instr_type> <typeSpec>
| <instr_var> <int32>
| <instr_var> <localname>
```

D.2. Instructions with no operand

These instructions require no operands, so they simply appear by themselves.

```
<instr> ::= <instr_none>
```

```
<instr_none> ::= // Derived from opcode.def
```

add	add.ovf	add.ovf.un	and
ann.catch	ann.def	ann.hoisted	ann.lab
arglist	break	ceq	cgt
cgt.un	ckfinite	clt	clt.un
conv.i	conv.i1	conv.i2	conv.i4
conv.i8	conv.ovf.i	conv.ovf.i.un	conv.ovf.i1
conv.ovf.i1.un	conv.ovf.i2	conv.ovf.i2.un	conv.ovf.i4
conv.ovf.i4.un	conv.ovf.i8	conv.ovf.i8.un	conv.ovf.u
conv.ovf.u.un	conv.ovf.u1	conv.ovf.u1.un	conv.ovf.u2
conv.ovf.u2.un	conv.ovf.u4	conv.ovf.u4.un	conv.ovf.u8
conv.ovf.u8.un	conv.r.un	conv.r4	conv.r8
conv.u	conv.u1	conv.u2	conv.u4
conv.u8	cpblk	div	div.un
dup	endfault	endfilter	endfinally
initblk		ldarg.0	ldarg.1
ldarg.2	ldarg.3	ldc.i4.0	ldc.i4.1
ldc.i4.2	ldc.i4.3	ldc.i4.4	ldc.i4.5
ldc.i4.6	ldc.i4.7	ldc.i4.8	ldc.i4.M1
ldelem.i	ldelem.i1	ldelem.i2	ldelem.i4
ldelem.i8	ldelem.r4	ldelem.r8	ldelem.ref
ldelem.u1	ldelem.u2	ldelem.u4	ldind.i
ldind.i1	ldind.i2	ldind.i4	ldind.i8
ldind.r4	ldind.r8	ldind.ref	ldind.u1
ldind.u2	ldind.u4	ldlen	ldloc.0
ldloc.1	ldloc.2	ldloc.3	ldnull
localloc	mul	mul.ovf	mul.ovf.un
neg	nop	not	or
pop	refanytype	rem	rem.un
ret	rethrow	shl	shr
shr.un	stelem.i	stelem.i1	stelem.i2
stelem.i4	stelem.i8	stelem.r4	stelem.r8
stelem.ref	stind.i	stind.i1	stind.i2
stind.i4	stind.i8	stind.r4	stind.r8

<code>stind.ref</code>		<code>stloc.0</code>		<code>stloc.1</code>		<code>stloc.2</code>	
<code>stloc.3</code>		<code>sub</code>		<code>sub.ovf</code>		<code>sub.ovf.un</code>	
<code>tail.</code>		<code>throw</code>		<code>volatile.</code>		<code>xor</code>	

Examples:

```
ldlen
not
```

D.3. Instructions that Refer to Parameters or Local Variables

These instructions take one operand, which references a parameter or local variable of the current method. The variable can be referenced by its number (starting with variable 0) or by name (if the names are supplied as part of a signature using the form that supplies both a type and a name).

```
<instr> ::= <instr_var> <int32> |
          <instr_var> <localname>
<instr_var> ::= // Derived from opcode.def
  ann.dead | ann.live | ann.ref
  ann.ref.s | ldarg    | ldarg.s | ldarga
  ldarga.s  | ldloc    | ldloc.s | ldloca
  ldloca.s  | starg     | starg.s | stloc
  stloc.s
```

Examples:

```
stloc 0      // store into 0th local
ldarg X3     // load from argument named X3
```

D.4. Instructions that Take a Single 32-bit Integer Argument

These instructions take one operand, which must be a 32-bit integer.

```
<instr> ::= <instr_i> <int32>
<instr_i> ::= // Derived from opcode.def
  ldc.i4 | ldc.i4.s | unaligned.
```

Examples:

```
ldc.i4 123456 // Load the number 123456
ldc.i4.s 10   // Load the number 10
```

D.5. Instructions that Take a Single 64-bit Integer Argument

These instructions take one operand, which must be a 64-bit integer.

```
<instr> ::= <instr_i8> <int64>
<instr_i8> ::= // Derived from opcode.def
  ldc.i8
```

Examples:

```
ldc.i8 0x123456789AB
ldc.i8 12
```

D.6. Instructions that Take a Single Floating Point Argument

These instructions take one operand, which must be a floating point number.

```
<instr> ::= <instr_r> <float64> |  
           <instr_r> <int64>   |  
           <instr_r> ( <bytes> ) // <bytes> is binary image  
<instr_r> ::= // Derived from opcode.def  
ldc.r4 | ldc.r8
```

Examples:

```
ldc.r4 10.2  
ldc.r4 10  
ldc.r4 0x123456789ABCDEF  
ldc.r8 (00 00 00 00 00 00 F8 FF)
```

D.7. Branch instructions

The assembler does not optimize branches. The branch must be specified explicitly as using either the short or long form of the instruction. If the displacement is too large for the short form, then the assembler will display an error.

```
<instr> ::=  
  <instr_brtarget> <int32> |  
  <instr_brtarget> <label>  
<instr_brtarget> ::= // Derived from opcode.def  
ann.data | ann.data.s | beq      | beq.s    | bge      | bge.s    |  
bge.un   | bge.un.s   | bgt      | bgt.s    | bgt.un   | bgt.un.s |  
ble       | ble.s           | ble.un   | ble.un.s | blt       | blt.s    |  
blt.un    | blt.un.s        | bne.un   | bne.un.s | br        | br.s     |  
brfalse   | brfalse.s       | brtrue   | brtrue.s | leave     | leave.s
```

Example:

```
br.s 22  
br foo
```

D.8. Instructions that Take a Method as an Argument

These instructions reference a method, either in another class (first instruction format) or in the current class (second instruction format).

```
<instr> ::=  
  <instr_method>  
  <callConv> <type> [ <typeSpec> :: ] <methodName> ( <parameters> )  
<instr_method> ::= // Derived from opcode.def  
ann.call | ann.hoisted_call | call  | callvirt | jmp |  
ldftn    | ldvirtftn           | newobj
```

Examples:


```
call instance int32 C.D.E::X(class W, native int)
ldftn vararg char F(...)    // Global Function F
```

D.9. Instructions that Take a Field of a Class as an Argument

These instructions reference a field of a class.

```
<instr> ::=
  <instr_field> <type> <typeSpec> :: <id>
<instr_field> ::= // Derived from opcode.def
  ldfld | ldflda | ldsfld | ldsflda | stfld | stsfld
```

Examples:

```
ldfld native int X::IntField
stsfld int32 Y::AnotherField
```

D.10. Instructions that Take a Type as an Argument

These instructions reference a type.

```
<instr> ::= <instr_type> <typeSpec>
<instr_type> ::= // Derived from opcode.def
  box      | castclass | cpobj      | initobj  | isinst   |
  ldelema  | ldoobj    | mkrefany  | newarr   | refanyval |
  sizeof   | stobj     | unbox    |
```

Examples:

```
initobj System.Console
sizeof class X
```

D.11. Instructions that Take a String as an Argument

These instructions take a string as an argument.

```
<instr> ::= <instr_string> <QSTRING>
<instr_string> ::= // Derived from opcode.def
  ldstr
```

Examples:

```
ldstr "This is a string"
ldstr "This has a\nnewline in it"
```

D.12. Instructions that Take a Signature as an Argument

These instructions take a stand-alone signature as an argument.

```
<instr> ::= <instr_sig> <callConv> <type> ( <parameters> )
<instr_sig> ::= // Derived from opcode.def
  calli
```

Examples:

```
calli class A.B(wchar *)
calli vararg bool(int32[,] X, ...)
    // Returns a boolean, takes at least one argument. The first
    // argument, named X, must be a two-dimensional array of
    // 32-bit ints
```

D.13. Instructions that Take a Metadata Token as an Argument

This instruction takes a metadata token as an argument. The token can reference a type, a method, or a field of a class.

```
<instr> ::= <instr_tok> <typeSpec> |
    <instr_tok> method
        <callConv> <type> <typeSpec> :: <methodName>
            ( <parameters> ) |
    <instr_tok> method
        <callConv> <type> <methodName>
            ( <parameters> ) |
    <instr_tok> field <type> <typeSpec> :: <id>
<instr_tok> ::= // Derived from opcode.def
    ldtoken
```

Examples:

```
ldtoken class System.Console
ldtoken method int32 X::Fn()
ldtoken method bool GlobalFn(int32 &)
ldtoken field class X.Y Class::Field
```

D.14. The SSA Φ -Node Instruction

This instruction embeds a static single assignment (SSA) Φ -Node into the instruction stream as an annotation.

```
<instr> ::= <instr_phi> <int16>*
<instr_phi> ::= // Derived from opcode.def
    ann.phi
```

Examples:

```
ann.phi 10 3 15
ann.phi 3 -2 0x3
```

D.15. Switch instruction

The switch instruction takes a set of labels or decimal relative values.

```
<instr> ::= <instr_switch> ( <labels> )
<instr_switch> ::= // Derived from opcode.def
    switch
```

Examples:

```
switch (0x3, -14, Label1)
```

switch (5, Label2)

Annex E. Normative: Opcode Definitions

```
#ifndef __OPCODE_DEF_
#define __OPCODE_DEF_

#define MOOT 0x00 // Marks unused second byte when encoding single
#define STP1 0xFE // Prefix code 1 for Standard Map
#define REFPRE 0xFF // Prefix for Reference Code Encoding
#define RESERVED_PREFIX_START 0xF7

#endif

// If the first byte of the standard encoding is 0xFF, then
// the second byte can be used as 1 byte encoding. Otherwise
// the encoding is two bytes.
//
// l b b
// e y y
// n t t
// g e e
// t
// Canonical Name h String Name Stack Behaviour Operand Params
Opcode Kind 1 2 Control Flow
// -----
OPDEF(CEE_NOP, "nop", Pop0, Push0, InlineNone,
IPrimitive, 1, 0xFF, 0x00, NEXT)
OPDEF(CEE_BREAK, "break", Pop0, Push0, InlineNone,
IPrimitive, 1, 0xFF, 0x01, BREAK)
OPDEF(CEE_LDARG_0, "ldarg.0", Pop0, Push1, InlineNone,
IMacro, 1, 0xFF, 0x02, NEXT)
OPDEF(CEE_LDARG_1, "ldarg.1", Pop0, Push1, InlineNone,
IMacro, 1, 0xFF, 0x03, NEXT)
OPDEF(CEE_LDARG_2, "ldarg.2", Pop0, Push1, InlineNone,
IMacro, 1, 0xFF, 0x04, NEXT)
OPDEF(CEE_LDARG_3, "ldarg.3", Pop0, Push1, InlineNone,
IMacro, 1, 0xFF, 0x05, NEXT)
OPDEF(CEE_LDLOC_0, "ldloc.0", Pop0, Push1, InlineNone,
IMacro, 1, 0xFF, 0x06, NEXT)
OPDEF(CEE_LDLOC_1, "ldloc.1", Pop0, Push1, InlineNone,
IMacro, 1, 0xFF, 0x07, NEXT)
OPDEF(CEE_LDLOC_2, "ldloc.2", Pop0, Push1, InlineNone,
IMacro, 1, 0xFF, 0x08, NEXT)
OPDEF(CEE_LDLOC_3, "ldloc.3", Pop0, Push1, InlineNone,
IMacro, 1, 0xFF, 0x09, NEXT)
OPDEF(CEE_STLOC_0, "stloc.0", Pop1, Push0, InlineNone,
IMacro, 1, 0xFF, 0x0A, NEXT)
OPDEF(CEE_STLOC_1, "stloc.1", Pop1, Push0, InlineNone,
IMacro, 1, 0xFF, 0x0B, NEXT)
OPDEF(CEE_STLOC_2, "stloc.2", Pop1, Push0, InlineNone,
IMacro, 1, 0xFF, 0x0C, NEXT)
OPDEF(CEE_STLOC_3, "stloc.3", Pop1, Push0, InlineNone,
IMacro, 1, 0xFF, 0x0D, NEXT)
OPDEF(CEE_LDARG_S, "ldarg.s", Pop0, Push1, ShortInlineVar,
IMacro, 1, 0xFF, 0x0E, NEXT)
OPDEF(CEE_LDARGA_S, "ldarga.s", Pop0, PushI, ShortInlineVar,
IMacro, 1, 0xFF, 0x0F, NEXT)
OPDEF(CEE_STARG_S, "starg.s", Pop1, Push0, ShortInlineVar,
IMacro, 1, 0xFF, 0x10, NEXT)
OPDEF(CEE_LDLOC_S, "ldloc.s", Pop0, Push1, ShortInlineVar,
IMacro, 1, 0xFF, 0x11, NEXT)
OPDEF(CEE_LDLOCA_S, "ldloca.s", Pop0, PushI, ShortInlineVar,
IMacro, 1, 0xFF, 0x12, NEXT)
OPDEF(CEE_STLOC_S, "stloc.s", Pop1, Push0, ShortInlineVar,
IMacro, 1, 0xFF, 0x13, NEXT)
OPDEF(CEE_LDNUL, "ldnull", Pop0, PushRef, InlineNone,
IPrimitive, 1, 0xFF, 0x14, NEXT)
OPDEF(CEE_LDC_I4_M1, "ldc.i4.m1", Pop0, PushI, InlineNone,
IMacro, 1, 0xFF, 0x15, NEXT)
OPDEF(CEE_LDC_I4_0, "ldc.i4.0", Pop0, PushI, InlineNone,
IMacro, 1, 0xFF, 0x16, NEXT)
OPDEF(CEE_LDC_I4_1, "ldc.i4.1", Pop0, PushI, InlineNone,
IMacro, 1, 0xFF, 0x17, NEXT)
OPDEF(CEE_LDC_I4_2, "ldc.i4.2", Pop0, PushI, InlineNone,
IMacro, 1, 0xFF, 0x18, NEXT)
OPDEF(CEE_LDC_I4_3, "ldc.i4.3", Pop0, PushI, InlineNone,
IMacro, 1, 0xFF, 0x19, NEXT)
OPDEF(CEE_LDC_I4_4, "ldc.i4.4", Pop0, PushI, InlineNone,
IMacro, 1, 0xFF, 0x1A, NEXT)
OPDEF(CEE_LDC_I4_5, "ldc.i4.5", Pop0, PushI, InlineNone,
IMacro, 1, 0xFF, 0x1B, NEXT)
```

OPDEF(CEE_LDC_I4_6,			"ldc.i4.6",	Pop0,	PushI,	InlineNone,
IMacro,	1,	0xFF,	NEXT)			
OPDEF(CEE_LDC_I4_7,			"ldc.i4.7",	Pop0,	PushI,	InlineNone,
IMacro,	1,	0xFF,	NEXT)			
OPDEF(CEE_LDC_I4_8,			"ldc.i4.8",	Pop0,	PushI,	InlineNone,
IMacro,	1,	0xFF,	NEXT)			
OPDEF(CEE_LDC_I4_S,			"ldc.i4.s",	Pop0,	PushI,	ShortInlineI,
IMacro,	1,	0xFF,	NEXT)			
OPDEF(CEE_LDC_I4,			"ldc.i4",	Pop0,	PushI,	InlineI,
IPrimitive,	1,	0xFF,	NEXT)			
OPDEF(CEE_LDC_I8,			"ldc.i8",	Pop0,	PushI8,	InlineI8,
IPrimitive,	1,	0xFF,	NEXT)			
OPDEF(CEE_LDC_R4,			"ldc.r4",	Pop0,	PushR4,	ShortInlineR,
IPrimitive,	1,	0xFF,	NEXT)			
OPDEF(CEE_LDC_R8,			"ldc.r8",	Pop0,	PushR8,	InlineR,
IPrimitive,	1,	0xFF,	NEXT)			
OPDEF(CEE_LDPTR,			"ldptr",	Pop0,	PushI,	
InlineRVA,	IPrimitive,	1,	0xFF,	0x24,	NEXT)	
OPDEF(CEE_DUP,			"dup",	Pop1,	Push1+Push1,	InlineNone,
IPrimitive,	1,	0xFF,	NEXT)			
OPDEF(CEE_POP,			"pop",	Pop1,	Push0,	InlineNone,
IPrimitive,	1,	0xFF,	NEXT)			
OPDEF(CEE_JMP,			"jmp",	Pop0,	Push0,	InlineMethod,
IPrimitive,	1,	0xFF,	CALL)			
OPDEF(CEE_CALL,			"call",	VarPop,	VarPush,	InlineMethod,
IPrimitive,	1,	0xFF,	CALL)			
OPDEF(CEE_CALLI,			"calli",	VarPop,	VarPush,	InlineSig,
IPrimitive,	1,	0xFF,	CALL)			
OPDEF(CEE_RET,			"ret",	VarPop,	Push0,	InlineNone,
IPrimitive,	1,	0xFF,	RETURN)			
OPDEF(CEE_BR_S,			"br.s",	Pop0,	Push0,	
ShortInlineBrTarget,IMacro,	1,	0xFF,	0x2B,	BRANCH)		
OPDEF(CEE_BRFALSE_S,			"brfalse.s",	PopI,	Push0,	
ShortInlineBrTarget,IMacro,	1,	0xFF,	0x2C,	COND_BRANCH)		
OPDEF(CEE_BRTRUE_S,			"brtrue.s",	PopI,	Push0,	
ShortInlineBrTarget,IMacro,	1,	0xFF,	0x2D,	COND_BRANCH)		
OPDEF(CEE_BEQ_S,			"beq.s",	Pop1+Pop1,	Push0,	
ShortInlineBrTarget,IMacro,	1,	0xFF,	0x2E,	COND_BRANCH)		
OPDEF(CEE_BGE_S,			"bge.s",	Pop1+Pop1,	Push0,	
ShortInlineBrTarget,IMacro,	1,	0xFF,	0x2F,	COND_BRANCH)		
OPDEF(CEE_BGT_S,			"bgt.s",	Pop1+Pop1,	Push0,	
ShortInlineBrTarget,IMacro,	1,	0xFF,	0x30,	COND_BRANCH)		
OPDEF(CEE_BLE_S,			"ble.s",	Pop1+Pop1,	Push0,	
ShortInlineBrTarget,IMacro,	1,	0xFF,	0x31,	COND_BRANCH)		
OPDEF(CEE_BLT_S,			"blt.s",	Pop1+Pop1,	Push0,	
ShortInlineBrTarget,IMacro,	1,	0xFF,	0x32,	COND_BRANCH)		
OPDEF(CEE_BNE_UN_S,			"bne.un.s",	Pop1+Pop1,	Push0,	
ShortInlineBrTarget,IMacro,	1,	0xFF,	0x33,	COND_BRANCH)		
OPDEF(CEE_BGE_UN_S,			"bge.un.s",	Pop1+Pop1,	Push0,	
ShortInlineBrTarget,IMacro,	1,	0xFF,	0x34,	COND_BRANCH)		
OPDEF(CEE_BGT_UN_S,			"bgt.un.s",	Pop1+Pop1,	Push0,	
ShortInlineBrTarget,IMacro,	1,	0xFF,	0x35,	COND_BRANCH)		
OPDEF(CEE_BLE_UN_S,			"ble.un.s",	Pop1+Pop1,	Push0,	
ShortInlineBrTarget,IMacro,	1,	0xFF,	0x36,	COND_BRANCH)		
OPDEF(CEE_BLT_UN_S,			"blt.un.s",	Pop1+Pop1,	Push0,	
ShortInlineBrTarget,IMacro,	1,	0xFF,	0x37,	COND_BRANCH)		
OPDEF(CEE_BR,			"br",	Pop0,	Push0,	InlineBrTarget,
IPrimitive,	1,	0xFF,	BRANCH)			
OPDEF(CEE_BRFALSE,			"brfalse",	PopI,	Push0,	InlineBrTarget,
IPrimitive,	1,	0xFF,	COND_BRANCH)			
OPDEF(CEE_BRTRUE,			"brtrue",	PopI,	Push0,	InlineBrTarget,
IPrimitive,	1,	0xFF,	COND_BRANCH)			
OPDEF(CEE_BEQ,			"beq",	Pop1+Pop1,	Push0,	InlineBrTarget,
IMacro,	1,	0xFF,	COND_BRANCH)			
OPDEF(CEE_BGE,			"bge",	Pop1+Pop1,	Push0,	InlineBrTarget,
IMacro,	1,	0xFF,	COND_BRANCH)			
OPDEF(CEE_BGT,			"bgt",	Pop1+Pop1,	Push0,	InlineBrTarget,
IMacro,	1,	0xFF,	COND_BRANCH)			
OPDEF(CEE_BLE,			"ble",	Pop1+Pop1,	Push0,	InlineBrTarget,
IMacro,	1,	0xFF,	COND_BRANCH)			
OPDEF(CEE_BLT,			"blt",	Pop1+Pop1,	Push0,	InlineBrTarget,
IMacro,	1,	0xFF,	COND_BRANCH)			
OPDEF(CEE_BNE_UN,			"bne.un",	Pop1+Pop1,	Push0,	InlineBrTarget,
IMacro,	1,	0xFF,	COND_BRANCH)			
OPDEF(CEE_BGE_UN,			"bge.un",	Pop1+Pop1,	Push0,	InlineBrTarget,
IMacro,	1,	0xFF,	COND_BRANCH)			
OPDEF(CEE_BGT_UN,			"bgt.un",	Pop1+Pop1,	Push0,	InlineBrTarget,
IMacro,	1,	0xFF,	COND_BRANCH)			

OPDEF(CEE_BLE_UN,		"ble.un",	PopI+PopI,	Push0,	InlineBrTarget,
IMacro, 1, 0xFF,	0x43,	COND_BRANCH)			
OPDEF(CEE_BLT_UN,		"blt.un",	PopI+PopI,	Push0,	InlineBrTarget,
IMacro, 1, 0xFF,	0x44,	COND_BRANCH)			
OPDEF(CEE_SWITCH,		"switch",	PopI,	Push0,	InlineSwitch,
IPrimitive, 1, 0xFF,	0x45,	COND_BRANCH)			
OPDEF(CEE_LDIND_I1,		"ldind.i1",	PopI,	PushI,	InlineNone,
IPrimitive, 1, 0xFF,	0x46,	NEXT)			
OPDEF(CEE_LDIND_U1,		"ldind.u1",	PopI,	PushI,	InlineNone,
IPrimitive, 1, 0xFF,	0x47,	NEXT)			
OPDEF(CEE_LDIND_I2,		"ldind.i2",	PopI,	PushI,	InlineNone,
IPrimitive, 1, 0xFF,	0x48,	NEXT)			
OPDEF(CEE_LDIND_U2,		"ldind.u2",	PopI,	PushI,	InlineNone,
IPrimitive, 1, 0xFF,	0x49,	NEXT)			
OPDEF(CEE_LDIND_I4,		"ldind.i4",	PopI,	PushI,	InlineNone,
IPrimitive, 1, 0xFF,	0x4A,	NEXT)			
OPDEF(CEE_LDIND_U4,		"ldind.u4",	PopI,	PushI,	InlineNone,
IPrimitive, 1, 0xFF,	0x4B,	NEXT)			
OPDEF(CEE_LDIND_I8,		"ldind.i8",	PopI,	PushI8,	InlineNone,
IPrimitive, 1, 0xFF,	0x4C,	NEXT)			
OPDEF(CEE_LDIND_I,		"ldind.i",	PopI,	PushI,	InlineNone,
IPrimitive, 1, 0xFF,	0x4D,	NEXT)			
OPDEF(CEE_LDIND_R4,		"ldind.r4",	PopI,	PushR4,	InlineNone,
IPrimitive, 1, 0xFF,	0x4E,	NEXT)			
OPDEF(CEE_LDIND_R8,		"ldind.r8",	PopI,	PushR8,	InlineNone,
IPrimitive, 1, 0xFF,	0x4F,	NEXT)			
OPDEF(CEE_LDIND_REF,		"ldind.ref",	PopI,	PushRef,	InlineNone,
IPrimitive, 1, 0xFF,	0x50,	NEXT)			
OPDEF(CEE_STIND_REF,		"stind.ref",	PopI+PopI,	Push0,	InlineNone,
IPrimitive, 1, 0xFF,	0x51,	NEXT)			
OPDEF(CEE_STIND_I1,		"stind.i1",	PopI+PopI,	Push0,	InlineNone,
IPrimitive, 1, 0xFF,	0x52,	NEXT)			
OPDEF(CEE_STIND_I2,		"stind.i2",	PopI+PopI,	Push0,	InlineNone,
IPrimitive, 1, 0xFF,	0x53,	NEXT)			
OPDEF(CEE_STIND_I4,		"stind.i4",	PopI+PopI,	Push0,	InlineNone,
IPrimitive, 1, 0xFF,	0x54,	NEXT)			
OPDEF(CEE_STIND_I8,		"stind.i8",	PopI+PopI8,	Push0,	InlineNone,
IPrimitive, 1, 0xFF,	0x55,	NEXT)			
OPDEF(CEE_STIND_R4,		"stind.r4",	PopI+PopR4,	Push0,	InlineNone,
IPrimitive, 1, 0xFF,	0x56,	NEXT)			
OPDEF(CEE_STIND_R8,		"stind.r8",	PopI+PopR8,	Push0,	InlineNone,
IPrimitive, 1, 0xFF,	0x57,	NEXT)			
OPDEF(CEE_ADD,		"add",	PopI+PopI,	Push1,	InlineNone,
IPrimitive, 1, 0xFF,	0x58,	NEXT)			
OPDEF(CEE_SUB,		"sub",	PopI+PopI,	Push1,	InlineNone,
IPrimitive, 1, 0xFF,	0x59,	NEXT)			
OPDEF(CEE_MUL,		"mul",	PopI+PopI,	Push1,	InlineNone,
IPrimitive, 1, 0xFF,	0x5A,	NEXT)			
OPDEF(CEE_DIV,		"div",	PopI+PopI,	Push1,	InlineNone,
IPrimitive, 1, 0xFF,	0x5B,	NEXT)			
OPDEF(CEE_DIV_UN,		"div.un",	PopI+PopI,	Push1,	InlineNone,
IPrimitive, 1, 0xFF,	0x5C,	NEXT)			
OPDEF(CEE_REM,		"rem",	PopI+PopI,	Push1,	InlineNone,
IPrimitive, 1, 0xFF,	0x5D,	NEXT)			
OPDEF(CEE_REM_UN,		"rem.un",	PopI+PopI,	Push1,	InlineNone,
IPrimitive, 1, 0xFF,	0x5E,	NEXT)			
OPDEF(CEE_AND,		"and",	PopI+PopI,	Push1,	InlineNone,
IPrimitive, 1, 0xFF,	0x5F,	NEXT)			
OPDEF(CEE_OR,		"or",	PopI+PopI,	Push1,	InlineNone,
IPrimitive, 1, 0xFF,	0x60,	NEXT)			
OPDEF(CEE_XOR,		"xor",	PopI+PopI,	Push1,	InlineNone,
IPrimitive, 1, 0xFF,	0x61,	NEXT)			
OPDEF(CEE_SHL,		"shl",	PopI+PopI,	Push1,	InlineNone,
IPrimitive, 1, 0xFF,	0x62,	NEXT)			
OPDEF(CEE_SHR,		"shr",	PopI+PopI,	Push1,	InlineNone,
IPrimitive, 1, 0xFF,	0x63,	NEXT)			
OPDEF(CEE_SHR_UN,		"shr.un",	PopI+PopI,	Push1,	InlineNone,
IPrimitive, 1, 0xFF,	0x64,	NEXT)			
OPDEF(CEE_NEG,		"neg",	PopI,	Push1,	InlineNone,
IPrimitive, 1, 0xFF,	0x65,	NEXT)			
OPDEF(CEE_NOT,		"not",	PopI,	Push1,	InlineNone,
IPrimitive, 1, 0xFF,	0x66,	NEXT)			
OPDEF(CEE_CONV_I1,		"conv.i1",	PopI,	PushI,	InlineNone,
IPrimitive, 1, 0xFF,	0x67,	NEXT)			
OPDEF(CEE_CONV_I2,		"conv.i2",	PopI,	PushI,	InlineNone,
IPrimitive, 1, 0xFF,	0x68,	NEXT)			
OPDEF(CEE_CONV_I4,		"conv.i4",	PopI,	PushI,	InlineNone,
IPrimitive, 1, 0xFF,	0x69,	NEXT)			

OPDEF(CEE_CONV_I8,			"conv.i8",	Pop1,	PushI8,	InlineNone,
IPrimitive, 1, 0xFF,	0x6A,	NEXT)				
OPDEF(CEE_CONV_R4,			"conv.r4",	Pop1,	PushR4,	InlineNone,
IPrimitive, 1, 0xFF,	0x6B,	NEXT)				
OPDEF(CEE_CONV_R8,			"conv.r8",	Pop1,	PushR8,	InlineNone,
IPrimitive, 1, 0xFF,	0x6C,	NEXT)				
OPDEF(CEE_CONV_U4,			"conv.u4",	Pop1,	PushI,	InlineNone,
IPrimitive, 1, 0xFF,	0x6D,	NEXT)				
OPDEF(CEE_CONV_U8,			"conv.u8",	Pop1,	PushI8,	InlineNone,
IPrimitive, 1, 0xFF,	0x6E,	NEXT)				
OPDEF(CEE_CALLVIRT,			"callvirt",	VarPop,	VarPush,	InlineMethod,
IObjModel, 1, 0xFF,	0x6F,	CALL)				
OPDEF(CEE_CPOBJ,			"cpobj",	PopI+PopI,	Push0,	InlineType,
IObjModel, 1, 0xFF,	0x70,	NEXT)				
OPDEF(CEE_LDOBJ,			"ldobj",	PopI,	Push1,	InlineType,
IObjModel, 1, 0xFF,	0x71,	NEXT)				
OPDEF(CEE_LDSTR,			"ldstr",	Pop0,	PushRef,	InlineString,
IObjModel, 1, 0xFF,	0x72,	NEXT)				
OPDEF(CEE_NEWOBJ,			"newobj",	VarPop,	PushRef,	InlineMethod,
IObjModel, 1, 0xFF,	0x73,	CALL)				
OPDEF(CEE_CASTCLASS,			"castclass",	PopRef,	PushRef,	InlineType,
IObjModel, 1, 0xFF,	0x74,	NEXT)				
OPDEF(CEE_ISINST,			"isinst",	PopRef,	PushI,	InlineType,
IObjModel, 1, 0xFF,	0x75,	NEXT)				
OPDEF(CEE_CONV_R_UN,			"conv.r.un",	Pop1,	PushR8,	InlineNone,
IPrimitive, 1, 0xFF,	0x76,	NEXT)				
OPDEF(CEE_ANN_DATA_S,			"ann.data.s",	Pop0,	Push0,	
ShortInlineBrTarget, IAnnotation, 1,	0xFF,	0x77,	BRANCH)			
OPDEF(CEE_BOX,			"box",	PopI,	PushRef,	InlineType,
IPrimitive, 1, 0xFF,	0x78,	NEXT)				
OPDEF(CEE_UNBOX,			"unbox",	PopRef,	PushI,	InlineType,
IPrimitive, 1, 0xFF,	0x79,	NEXT)				
OPDEF(CEE_THROW,			"throw",	PopRef,	Push0,	InlineNone,
IObjModel, 1, 0xFF,	0x7A,	THROW)				
OPDEF(CEE_LDFLD,			"ldfld",	PopRef,	Push1,	InlineField,
IObjModel, 1, 0xFF,	0x7B,	NEXT)				
OPDEF(CEE_LDFLDA,			"ldflda",	PopRef,	PushI,	InlineField,
IObjModel, 1, 0xFF,	0x7C,	NEXT)				
OPDEF(CEE_STFLD,			"stfld",	PopRef+Pop1,	Push0,	InlineField,
IObjModel, 1, 0xFF,	0x7D,	NEXT)				
OPDEF(CEE_LDSFLD,			"ldsfld",	Pop0,	Push1,	InlineField,
IObjModel, 1, 0xFF,	0x7E,	NEXT)				
OPDEF(CEE_LDSFLDA,			"ldsfllda",	Pop0,	PushI,	InlineField,
IObjModel, 1, 0xFF,	0x7F,	NEXT)				
OPDEF(CEE_STSFDA,			"stsflda",	Pop1,	Push0,	InlineField,
IObjModel, 1, 0xFF,	0x80,	NEXT)				
OPDEF(CEE_STOBJ,			"stobj",	PopI+Pop1,	Push0,	InlineType,
IPrimitive, 1, 0xFF,	0x81,	NEXT)				
OPDEF(CEE_CONV_OVF_I1_UN,			"conv.ovf.i1.un",	Pop1,	PushI,	InlineNone,
IPrimitive, 1, 0xFF,	0x82,	NEXT)				
OPDEF(CEE_CONV_OVF_I2_UN,			"conv.ovf.i2.un",	Pop1,	PushI,	InlineNone,
IPrimitive, 1, 0xFF,	0x83,	NEXT)				
OPDEF(CEE_CONV_OVF_I4_UN,			"conv.ovf.i4.un",	Pop1,	PushI,	InlineNone,
IPrimitive, 1, 0xFF,	0x84,	NEXT)				
OPDEF(CEE_CONV_OVF_I8_UN,			"conv.ovf.i8.un",	Pop1,	PushI8,	InlineNone,
IPrimitive, 1, 0xFF,	0x85,	NEXT)				
OPDEF(CEE_CONV_OVF_U1_UN,			"conv.ovf.u1.un",	Pop1,	PushI,	InlineNone,
IPrimitive, 1, 0xFF,	0x86,	NEXT)				
OPDEF(CEE_CONV_OVF_U2_UN,			"conv.ovf.u2.un",	Pop1,	PushI,	InlineNone,
IPrimitive, 1, 0xFF,	0x87,	NEXT)				
OPDEF(CEE_CONV_OVF_U4_UN,			"conv.ovf.u4.un",	Pop1,	PushI,	InlineNone,
IPrimitive, 1, 0xFF,	0x88,	NEXT)				
OPDEF(CEE_CONV_OVF_U8_UN,			"conv.ovf.u8.un",	Pop1,	PushI8,	InlineNone,
IPrimitive, 1, 0xFF,	0x89,	NEXT)				
OPDEF(CEE_CONV_OVF_I_UN,			"conv.ovf.i.un",	Pop1,	PushI,	InlineNone,
IPrimitive, 1, 0xFF,	0x8A,	NEXT)				
OPDEF(CEE_CONV_OVF_U_UN,			"conv.ovf.u.un",	Pop1,	PushI,	InlineNone,
IPrimitive, 1, 0xFF,	0x8B,	NEXT)				
OPDEF(CEE_UNUSED49,			"unused",	Pop0,	Push0,	InlineNone,
IPrimitive, 1, 0xFF,	0x8C,	NEXT)				
OPDEF(CEE_NEWARR,			"newarr",	PopI,	PushRef,	InlineType,
IObjModel, 1, 0xFF,	0x8D,	NEXT)				
OPDEF(CEE_LDLEN,			"ldlen",	PopRef,	PushI,	InlineNone,
IObjModel, 1, 0xFF,	0x8E,	NEXT)				
OPDEF(CEE_LDELEMA,			"ldelema",	PopRef+PopI,	PushI,	InlineType,
IObjModel, 1, 0xFF,	0x8F,	NEXT)				
OPDEF(CEE_LDELEM_I1,			"ldelem.i1",	PopRef+PopI,	PushI,	InlineNone,
IObjModel, 1, 0xFF,	0x90,	NEXT)				

OPDEF(CEE_LDELEM_U1,		"ldelem.u1",	PopRef+PopI,	PushI,	InlineNone,
IObjModel, 1, 0xFF,	0x91,	NEXT)			
OPDEF(CEE_LDELEM_I2,		"ldelem.i2",	PopRef+PopI,	PushI,	InlineNone,
IObjModel, 1, 0xFF,	0x92,	NEXT)			
OPDEF(CEE_LDELEM_U2,		"ldelem.u2",	PopRef+PopI,	PushI,	InlineNone,
IObjModel, 1, 0xFF,	0x93,	NEXT)			
OPDEF(CEE_LDELEM_I4,		"ldelem.i4",	PopRef+PopI,	PushI,	InlineNone,
IObjModel, 1, 0xFF,	0x94,	NEXT)			
OPDEF(CEE_LDELEM_U4,		"ldelem.u4",	PopRef+PopI,	PushI,	InlineNone,
IObjModel, 1, 0xFF,	0x95,	NEXT)			
OPDEF(CEE_LDELEM_I8,		"ldelem.i8",	PopRef+PopI,	PushI8,	InlineNone,
IObjModel, 1, 0xFF,	0x96,	NEXT)			
OPDEF(CEE_LDELEM_I,		"ldelem.i",	PopRef+PopI,	PushI,	InlineNone,
IObjModel, 1, 0xFF,	0x97,	NEXT)			
OPDEF(CEE_LDELEM_R4,		"ldelem.r4",	PopRef+PopI,	PushR4,	InlineNone,
IObjModel, 1, 0xFF,	0x98,	NEXT)			
OPDEF(CEE_LDELEM_R8,		"ldelem.r8",	PopRef+PopI,	PushR8,	InlineNone,
IObjModel, 1, 0xFF,	0x99,	NEXT)			
OPDEF(CEE_LDELEM_REF,		"ldelem.ref",	PopRef+PopI,	PushRef,	InlineNone,
IObjModel, 1, 0xFF,	0x9A,	NEXT)			
OPDEF(CEE_STELEM_I,		"stelem.i",	PopRef+PopI+PopI,	Push0,	InlineNone,
IObjModel, 1, 0xFF,	0x9B,	NEXT)			
OPDEF(CEE_STELEM_I1,		"stelem.i1",	PopRef+PopI+PopI,	Push0,	InlineNone,
IObjModel, 1, 0xFF,	0x9C,	NEXT)			
OPDEF(CEE_STELEM_I2,		"stelem.i2",	PopRef+PopI+PopI,	Push0,	InlineNone,
IObjModel, 1, 0xFF,	0x9D,	NEXT)			
OPDEF(CEE_STELEM_I4,		"stelem.i4",	PopRef+PopI+PopI,	Push0,	InlineNone,
IObjModel, 1, 0xFF,	0x9E,	NEXT)			
OPDEF(CEE_STELEM_I8,		"stelem.i8",	PopRef+PopI+PopI8,	Push0,	InlineNone,
IObjModel, 1, 0xFF,	0x9F,	NEXT)			
OPDEF(CEE_STELEM_R4,		"stelem.r4",	PopRef+PopI+PopR4,	Push0,	InlineNone,
IObjModel, 1, 0xFF,	0xA0,	NEXT)			
OPDEF(CEE_STELEM_R8,		"stelem.r8",	PopRef+PopI+PopR8,	Push0,	InlineNone,
IObjModel, 1, 0xFF,	0xA1,	NEXT)			
OPDEF(CEE_STELEM_REF,		"stelem.ref",	PopRef+PopI+PopRef,	Push0,	InlineNone,
IObjModel, 1, 0xFF,	0xA2,	NEXT)			
OPDEF(CEE_UNUSED2,		"unused",	Pop0,	Push0,	InlineNone,
IPrimitive, 1, 0xFF,	0xA3,	NEXT)			
OPDEF(CEE_UNUSED3,		"unused",	Pop0,	Push0,	InlineNone,
IPrimitive, 1, 0xFF,	0xA4,	NEXT)			
OPDEF(CEE_UNUSED4,		"unused",	Pop0,	Push0,	InlineNone,
IPrimitive, 1, 0xFF,	0xA5,	NEXT)			
OPDEF(CEE_UNUSED5,		"unused",	Pop0,	Push0,	InlineNone,
IPrimitive, 1, 0xFF,	0xA6,	NEXT)			
OPDEF(CEE_UNUSED6,		"unused",	Pop0,	Push0,	InlineNone,
IPrimitive, 1, 0xFF,	0xA7,	NEXT)			
OPDEF(CEE_UNUSED7,		"unused",	Pop0,	Push0,	InlineNone,
IPrimitive, 1, 0xFF,	0xA8,	NEXT)			
OPDEF(CEE_UNUSED8,		"unused",	Pop0,	Push0,	InlineNone,
IPrimitive, 1, 0xFF,	0xA9,	NEXT)			
OPDEF(CEE_UNUSED9,		"unused",	Pop0,	Push0,	InlineNone,
IPrimitive, 1, 0xFF,	0xAA,	NEXT)			
OPDEF(CEE_UNUSED10,		"unused",	Pop0,	Push0,	InlineNone,
IPrimitive, 1, 0xFF,	0xAB,	NEXT)			
OPDEF(CEE_UNUSED11,		"unused",	Pop0,	Push0,	InlineNone,
IPrimitive, 1, 0xFF,	0xAC,	NEXT)			
OPDEF(CEE_UNUSED12,		"unused",	Pop0,	Push0,	InlineNone,
IPrimitive, 1, 0xFF,	0xAD,	NEXT)			
OPDEF(CEE_UNUSED13,		"unused",	Pop0,	Push0,	InlineNone,
IPrimitive, 1, 0xFF,	0xAE,	NEXT)			
OPDEF(CEE_UNUSED14,		"unused",	Pop0,	Push0,	InlineNone,
IPrimitive, 1, 0xFF,	0xAF,	NEXT)			
OPDEF(CEE_UNUSED15,		"unused",	Pop0,	Push0,	InlineNone,
IPrimitive, 1, 0xFF,	0xB0,	NEXT)			
OPDEF(CEE_UNUSED16,		"unused",	Pop0,	Push0,	InlineNone,
IPrimitive, 1, 0xFF,	0xB1,	NEXT)			
OPDEF(CEE_UNUSED17,		"unused",	Pop0,	Push0,	InlineNone,
IPrimitive, 1, 0xFF,	0xB2,	NEXT)			
OPDEF(CEE_CONV_OVF_I1,		"conv.ovf.i1",	Pop1,	PushI,	InlineNone,
IPrimitive, 1, 0xFF,	0xB3,	NEXT)			
OPDEF(CEE_CONV_OVF_U1,		"conv.ovf.u1",	Pop1,	PushI,	InlineNone,
IPrimitive, 1, 0xFF,	0xB4,	NEXT)			
OPDEF(CEE_CONV_OVF_I2,		"conv.ovf.i2",	Pop1,	PushI,	InlineNone,
IPrimitive, 1, 0xFF,	0xB5,	NEXT)			
OPDEF(CEE_CONV_OVF_U2,		"conv.ovf.u2",	Pop1,	PushI,	InlineNone,
IPrimitive, 1, 0xFF,	0xB6,	NEXT)			
OPDEF(CEE_CONV_OVF_I4,		"conv.ovf.i4",	Pop1,	PushI,	InlineNone,
IPrimitive, 1, 0xFF,	0xB7,	NEXT)			

OPDEF(CEE_CONV_OVF_U4,		"conv.ovf.u4",	Pop1,	PushI,	InlineNone,
IPrimitive, 1, 0xFF,	0xB8,	NEXT)			
OPDEF(CEE_CONV_OVF_I8,		"conv.ovf.i8",	Pop1,	PushI8,	InlineNone,
IPrimitive, 1, 0xFF,	0xB9,	NEXT)			
OPDEF(CEE_CONV_OVF_U8,		"conv.ovf.u8",	Pop1,	PushI8,	InlineNone,
IPrimitive, 1, 0xFF,	0xBA,	NEXT)			
OPDEF(CEE_UNUSED50,		"unused",	Pop0,	Push0,	InlineNone,
IPrimitive, 1, 0xFF,	0xBB,	NEXT)			
OPDEF(CEE_UNUSED18,		"unused",	Pop0,	Push0,	InlineNone,
IPrimitive, 1, 0xFF,	0xBC,	NEXT)			
OPDEF(CEE_UNUSED19,		"unused",	Pop0,	Push0,	InlineNone,
IPrimitive, 1, 0xFF,	0xBD,	NEXT)			
OPDEF(CEE_UNUSED20,		"unused",	Pop0,	Push0,	InlineNone,
IPrimitive, 1, 0xFF,	0xBE,	NEXT)			
OPDEF(CEE_UNUSED21,		"unused",	Pop0,	Push0,	InlineNone,
IPrimitive, 1, 0xFF,	0xBF,	NEXT)			
OPDEF(CEE_UNUSED22,		"unused",	Pop0,	Push0,	InlineNone,
IPrimitive, 1, 0xFF,	0xC0,	NEXT)			
OPDEF(CEE_UNUSED23,		"unused",	Pop0,	Push0,	InlineNone,
IPrimitive, 1, 0xFF,	0xC1,	NEXT)			
OPDEF(CEE_REFANYVAL,		"refanyval",	Pop1,	PushI,	InlineType,
IPrimitive, 1, 0xFF,	0xC2,	NEXT)			
OPDEF(CEE_CKFINITE,		"ckfinite",	Pop1,	PushR8,	InlineNone,
IPrimitive, 1, 0xFF,	0xC3,	NEXT)			
OPDEF(CEE_UNUSED24,		"unused",	Pop0,	Push0,	InlineNone,
IPrimitive, 1, 0xFF,	0xC4,	NEXT)			
OPDEF(CEE_UNUSED25,		"unused",	Pop0,	Push0,	InlineNone,
IPrimitive, 1, 0xFF,	0xC5,	NEXT)			
OPDEF(CEE_MKREFANY,		"mkrefany",	PopI,	PushI,	InlineType,
IPrimitive, 1, 0xFF,	0xC6,	NEXT)			
OPDEF(CEE_ANN_CALL,		"ann.call",	Pop0,	Push0,	InlineMethod,
IAnnotation, 1, 0xFF,	0xC7,	NEXT)			
OPDEF(CEE_ANN_CATCH,		"ann.catch",	Pop0,	Push0,	InlineNone,
IAnnotation, 1, 0xFF,	0xC8,	NEXT)			
OPDEF(CEE_ANN_DEAD,		"ann.dead",	Pop1,	PushI,	InlineVar,
IAnnotation, 1, 0xFF,	0xC9,	NEXT)			
OPDEF(CEE_ANN_HOISTED,		"ann.hoisted",	Pop0,	Push0,	InlineNone,
IAnnotation, 1, 0xFF,	0xCA,	NEXT)			
OPDEF(CEE_ANN_HOISTED_CALL,		"ann.hoisted_call",	Pop0,	Push0,	InlineMethod,
IAnnotation, 1, 0xFF,	0xCB,	NEXT)			
OPDEF(CEE_ANN_LAB,		"ann.lab",	Pop0,	Push0,	InlineNone,
IAnnotation, 1, 0xFF,	0xCC,	NEXT)			
OPDEF(CEE_ANN_DEF,		"ann.def",	Pop0,	Push0,	InlineNone,
IAnnotation, 1, 0xFF,	0xCD,	NEXT)			
OPDEF(CEE_ANN_REF_S,		"ann.ref.s",	Pop0,	Push0,	ShortInlineVar,
IAnnotation, 1, 0xFF,	0xCE,	NEXT)			
OPDEF(CEE_ANN_PHI,		"ann.phi",	Pop0,	Push0,	InlinePhi,
IAnnotation, 1, 0xFF,	0xCF,	PHI)			
OPDEF(CEE_LD_TOKEN,		"ldtoken",	Pop0,	PushI,	InlineTok,
IPrimitive, 1, 0xFF,	0xD0,	NEXT)			
OPDEF(CEE_CONV_U2,		"conv.u2",	Pop1,	PushI,	InlineNone,
IPrimitive, 1, 0xFF,	0xD1,	NEXT)			
OPDEF(CEE_CONV_U1,		"conv.u1",	Pop1,	PushI,	InlineNone,
IPrimitive, 1, 0xFF,	0xD2,	NEXT)			
OPDEF(CEE_CONV_I,		"conv.i",	Pop1,	PushI,	InlineNone,
IPrimitive, 1, 0xFF,	0xD3,	NEXT)			
OPDEF(CEE_CONV_OVF_I,		"conv.ovf.i",	Pop1,	PushI,	InlineNone,
IPrimitive, 1, 0xFF,	0xD4,	NEXT)			
OPDEF(CEE_CONV_OVF_U,		"conv.ovf.u",	Pop1,	PushI,	InlineNone,
IPrimitive, 1, 0xFF,	0xD5,	NEXT)			
OPDEF(CEE_ADD_OVF,		"add.ovf",	Pop1+Pop1,	PushI,	InlineNone,
IPrimitive, 1, 0xFF,	0xD6,	NEXT)			
OPDEF(CEE_ADD_OVF_UN,		"add.ovf.un",	Pop1+Pop1,	PushI,	InlineNone,
IPrimitive, 1, 0xFF,	0xD7,	NEXT)			
OPDEF(CEE_MUL_OVF,		"mul.ovf",	Pop1+Pop1,	PushI,	InlineNone,
IPrimitive, 1, 0xFF,	0xD8,	NEXT)			
OPDEF(CEE_MUL_OVF_UN,		"mul.ovf.un",	Pop1+Pop1,	PushI,	InlineNone,
IPrimitive, 1, 0xFF,	0xD9,	NEXT)			
OPDEF(CEE_SUB_OVF,		"sub.ovf",	Pop1+Pop1,	PushI,	InlineNone,
IPrimitive, 1, 0xFF,	0xDA,	NEXT)			
OPDEF(CEE_SUB_OVF_UN,		"sub.ovf.un",	Pop1+Pop1,	PushI,	InlineNone,
IPrimitive, 1, 0xFF,	0xDB,	NEXT)			
OPDEF(CEE_ENDFINALLY,		"endfinally",	Pop0,	Push0,	InlineNone,
IPrimitive, 1, 0xFF,	0xDC,	RETURN)			
OPDEF(CEE_LEAVE,		"leave",	Pop0,	Push0,	InlineBrTarget,
IPrimitive, 1, 0xFF,	0xDD,	BRANCH)			
OPDEF(CEE_LEAVE_S,		"leave.s",	Pop0,	Push0,	
ShortInlineBrTarget, IPrimitive,	1, 0xFF,	0xDE,	BRANCH)		

OPDEF(CEE_STIND_I,		"stind.i",	PopI+PopI,	Push0,	InlineNone,
IPrimitive, 1, 0xFF,	0xDF,	NEXT)			
OPDEF(CEE_CONV_U,		"conv.u",	Pop1,	PushI,	InlineNone,
IPrimitive, 1, 0xFF,	0xE0,	NEXT)			
OPDEF(CEE_UNUSED26,		"unused",	Pop0,	Push0,	InlineNone,
IPrimitive, 1, 0xFF,	0xE1,	NEXT)			
OPDEF(CEE_UNUSED27,		"unused",	Pop0,	Push0,	InlineNone,
IPrimitive, 1, 0xFF,	0xE2,	NEXT)			
OPDEF(CEE_UNUSED28,		"unused",	Pop0,	Push0,	InlineNone,
IPrimitive, 1, 0xFF,	0xE3,	NEXT)			
OPDEF(CEE_UNUSED29,		"unused",	Pop0,	Push0,	InlineNone,
IPrimitive, 1, 0xFF,	0xE4,	NEXT)			
OPDEF(CEE_UNUSED30,		"unused",	Pop0,	Push0,	InlineNone,
IPrimitive, 1, 0xFF,	0xE5,	NEXT)			
OPDEF(CEE_UNUSED31,		"unused",	Pop0,	Push0,	InlineNone,
IPrimitive, 1, 0xFF,	0xE6,	NEXT)			
OPDEF(CEE_UNUSED32,		"unused",	Pop0,	Push0,	InlineNone,
IPrimitive, 1, 0xFF,	0xE7,	NEXT)			
OPDEF(CEE_UNUSED33,		"unused",	Pop0,	Push0,	InlineNone,
IPrimitive, 1, 0xFF,	0xE8,	NEXT)			
OPDEF(CEE_UNUSED34,		"unused",	Pop0,	Push0,	InlineNone,
IPrimitive, 1, 0xFF,	0xE9,	NEXT)			
OPDEF(CEE_UNUSED35,		"unused",	Pop0,	Push0,	InlineNone,
IPrimitive, 1, 0xFF,	0xEA,	NEXT)			
OPDEF(CEE_UNUSED36,		"unused",	Pop0,	Push0,	InlineNone,
IPrimitive, 1, 0xFF,	0xEB,	NEXT)			
OPDEF(CEE_UNUSED37,		"unused",	Pop0,	Push0,	InlineNone,
IPrimitive, 1, 0xFF,	0xEC,	NEXT)			
OPDEF(CEE_UNUSED38,		"unused",	Pop0,	Push0,	InlineNone,
IPrimitive, 1, 0xFF,	0xED,	NEXT)			
OPDEF(CEE_UNUSED39,		"unused",	Pop0,	Push0,	InlineNone,
IPrimitive, 1, 0xFF,	0xEE,	NEXT)			
OPDEF(CEE_UNUSED40,		"unused",	Pop0,	Push0,	InlineNone,
IPrimitive, 1, 0xFF,	0xEF,	NEXT)			
OPDEF(CEE_UNUSED41,		"unused",	Pop0,	Push0,	InlineNone,
IPrimitive, 1, 0xFF,	0xF0,	NEXT)			
OPDEF(CEE_UNUSED42,		"unused",	Pop0,	Push0,	InlineNone,
IPrimitive, 1, 0xFF,	0xF1,	NEXT)			
OPDEF(CEE_UNUSED43,		"unused",	Pop0,	Push0,	InlineNone,
IPrimitive, 1, 0xFF,	0xF2,	NEXT)			
OPDEF(CEE_UNUSED44,		"unused",	Pop0,	Push0,	InlineNone,
IPrimitive, 1, 0xFF,	0xF3,	NEXT)			
OPDEF(CEE_UNUSED45,		"unused",	Pop0,	Push0,	InlineNone,
IPrimitive, 1, 0xFF,	0xF4,	NEXT)			
OPDEF(CEE_UNUSED46,		"unused",	Pop0,	Push0,	InlineNone,
IPrimitive, 1, 0xFF,	0xF5,	NEXT)			
OPDEF(CEE_UNUSED47,		"unused",	Pop0,	Push0,	InlineNone,
IPrimitive, 1, 0xFF,	0xF6,	NEXT)			
OPDEF(CEE_UNUSED48,		"unused",	Pop0,	Push0,	InlineNone,
IPrimitive, 1, 0xFF,	0xF7,	NEXT)			
OPDEF(CEE_PREFIX7,		"prefix7",	Pop0,	Push0,	InlineNone,
IInternal, 1, 0xFF,	0xF8,	META)			
OPDEF(CEE_PREFIX6,		"prefix6",	Pop0,	Push0,	InlineNone,
IInternal, 1, 0xFF,	0xF9,	META)			
OPDEF(CEE_PREFIX5,		"prefix5",	Pop0,	Push0,	InlineNone,
IInternal, 1, 0xFF,	0xFA,	META)			
OPDEF(CEE_PREFIX4,		"prefix4",	Pop0,	Push0,	InlineNone,
IInternal, 1, 0xFF,	0xFB,	META)			
OPDEF(CEE_PREFIX3,		"prefix3",	Pop0,	Push0,	InlineNone,
IInternal, 1, 0xFF,	0xFC,	META)			
OPDEF(CEE_PREFIX2,		"prefix2",	Pop0,	Push0,	InlineNone,
IInternal, 1, 0xFF,	0xFD,	META)			
OPDEF(CEE_PREFIX1,		"prefix1",	Pop0,	Push0,	InlineNone,
IInternal, 1, 0xFF,	0xFE,	META)			
OPDEF(CEE_PREFIXREF,		"prefixref",	Pop0,	Push0,	InlineNone,
IInternal, 1, 0xFF,	0xFF,	META)			
OPDEF(CEE_ARGLIST,		"arglist",	Pop0,	PushI,	InlineNone,
IPrimitive, 2, 0xFE,	0x00,	NEXT)			
OPDEF(CEE_CEQ,		"ceq",	Pop1+Pop1,	PushI,	InlineNone,
IPrimitive, 2, 0xFE,	0x01,	NEXT)			
OPDEF(CEE_CGT,		"cgt",	Pop1+Pop1,	PushI,	InlineNone,
IPrimitive, 2, 0xFE,	0x02,	NEXT)			
OPDEF(CEE_CGT_UN,		"cgt.un",	Pop1+Pop1,	PushI,	InlineNone,
IPrimitive, 2, 0xFE,	0x03,	NEXT)			
OPDEF(CEE_CLT,		"clt",	Pop1+Pop1,	PushI,	InlineNone,
IPrimitive, 2, 0xFE,	0x04,	NEXT)			
OPDEF(CEE_CLT_UN,		"clt.un",	Pop1+Pop1,	PushI,	InlineNone,
IPrimitive, 2, 0xFE,	0x05,	NEXT)			

```
OPDEF(CEE_LDFTN, "ldftn", Pop0, PushI, InlineMethod,
IPrimitive, 2, 0xFE, 0x06, NEXT)
OPDEF(CEE_LDVRTFTN, "ldvirtftn", PopRef, PushI, InlineMethod,
IPrimitive, 2, 0xFE, 0x07, NEXT)
OPDEF(CEE_JMPI, "jmp", PopI, Push0, InlineNone,
IPrimitive, 2, 0xFE, 0x08, CALL)
OPDEF(CEE_LDARG, "ldarg", Pop0, Push1, InlineVar,
IPrimitive, 2, 0xFE, 0x09, NEXT)
OPDEF(CEE_LDARGA, "ldarga", Pop0, PushI, InlineVar,
IPrimitive, 2, 0xFE, 0x0A, NEXT)
OPDEF(CEE_STARG, "starg", Pop1, Push0, InlineVar,
IPrimitive, 2, 0xFE, 0x0B, NEXT)
OPDEF(CEE_LDLOC, "ldloc", Pop0, Push1, InlineVar,
IPrimitive, 2, 0xFE, 0x0C, NEXT)
OPDEF(CEE_LDLOCA, "ldloca", Pop0, PushI, InlineVar,
IPrimitive, 2, 0xFE, 0x0D, NEXT)
OPDEF(CEE_STLOC, "stloc", Pop1, Push0, InlineVar,
IPrimitive, 2, 0xFE, 0x0E, NEXT)
OPDEF(CEE_LOCALLOC, "localloc", PopI, PushI, InlineNone,
IPrimitive, 2, 0xFE, 0x0F, NEXT)
OPDEF(CEE_UNUSED57, "unused", Pop0, Push0, InlineNone,
IPrimitive, 2, 0xFE, 0x10, NEXT)
OPDEF(CEE_ENDFILTER, "endfilter", PopI, Push0, InlineNone,
IPrimitive, 2, 0xFE, 0x11, RETURN)
OPDEF(CEE_UNALIGNED, "unaligned.", Pop0, Push0, ShortInlineI,
IPrefix, 2, 0xFE, 0x12, META)
OPDEF(CEE_VOLATILE, "volatile.", Pop0, Push0, InlineNone,
IPrefix, 2, 0xFE, 0x13, META)
OPDEF(CEE_TAILCALL, "tail.", Pop0, Push0, InlineNone,
IPrefix, 2, 0xFE, 0x14, META)
OPDEF(CEE_INITOBJ, "initobj", PopI, Push0, InlineType,
IObjModel, 2, 0xFE, 0x15, NEXT)
OPDEF(CEE_ANN_LIVE, "ann.live", Pop1, Push1, InlineVar,
IAnnotation, 2, 0xFE, 0x16, NEXT)
OPDEF(CEE_CPBLK, "cpblk", PopI+PopI+PopI, Push0, InlineNone,
IPrimitive, 2, 0xFE, 0x17, NEXT)
OPDEF(CEE_INITBLK, "initblk", PopI+PopI+PopI, Push0, InlineNone,
IPrimitive, 2, 0xFE, 0x18, NEXT)
OPDEF(CEE_ANN_REF, "ann.ref", Pop0, Push0, InlineVar,
IAnnotation, 2, 0xFE, 0x19, NEXT)
OPDEF(CEE_RETHROW, "rethrow", Pop0, Push0, InlineNone,
IObjModel, 2, 0xFE, 0x1A, THROW)
OPDEF(CEE_UNUSED51, "unused", Pop0, Push0, InlineNone,
IPrimitive, 2, 0xFE, 0x1B, NEXT)
OPDEF(CEE_SIZEOF, "sizeof", Pop0, PushI, InlineType,
IPrimitive, 2, 0xFE, 0x1C, NEXT)
OPDEF(CEE_REFANYTYPE, "refanytype", Pop1, PushI, InlineNone,
IPrimitive, 2, 0xFE, 0x1D, NEXT)
OPDEF(CEE_UNUSED52, "unused", Pop0, Push0, InlineNone,
IPrimitive, 2, 0xFE, 0x1E, NEXT)
OPDEF(CEE_UNUSED53, "unused", Pop0, Push0, InlineNone,
IPrimitive, 2, 0xFE, 0x1F, NEXT)
OPDEF(CEE_UNUSED54, "unused", Pop0, Push0, InlineNone,
IPrimitive, 2, 0xFE, 0x20, NEXT)
OPDEF(CEE_UNUSED55, "unused", Pop0, Push0, InlineNone,
IPrimitive, 2, 0xFE, 0x21, NEXT)
OPDEF(CEE_ANN_DATA, "ann.data", Pop0, Push0, InlineBrTarget,
IAnnotation, 2, 0xFE, 0x22, BRANCH)

#ifdef OPALIAS
#define _OPALIAS_DEFINED_
#define OPALIAS(canonicalName, stringName, realOpcode)
#endif

OPALIAS(CEE_BRNULL, "brnull", CEE_BRFALSE)
OPALIAS(CEE_BRNULL_S, "brnull.s", CEE_BRFALSE_S)
OPALIAS(CEE_BRZERO, "brzero", CEE_BRFALSE)
OPALIAS(CEE_BRZERO_S, "brzero.s", CEE_BRFALSE_S)
OPALIAS(CEE_BRINST, "brinst", CEE_BRTRUE)
OPALIAS(CEE_BRINST_S, "brinst.s", CEE_BRTRUE_S)
OPALIAS(CEE_LDIND_U8, "ldind.u8", CEE_LDIND_I8)
OPALIAS(CEE_LDELEM_U8, "ldelem.u8", CEE_LDELEM_I8)
OPALIAS(CEE_LDC_I4_M1x, "ldc.i4.M1", CEE_LDC_I4_M1)
OPALIAS(CEE_ENDFAULT, "endfault", CEE_ENDFINALLY)

#ifdef _OPALIAS_DEFINED_
#undef OPALIAS
#undef _OPALIAS_DEFINED_
#endif
```

Annex F. Normative: Values for .subsystem

Below is the list of possible values accepted by the ILASM **.subsystem** keyword. These are defined in the Windows WinNT.h header file. Note that .subsystem does not accept the symbolic name – you must supply the numeric value:

```
#define IMAGE_SUBSYSTEM_UNKNOWN      0 // Unknown subsystem.
#define IMAGE_SUBSYSTEM_NATIVE      1 // Image doesn't require a subsystem.
#define IMAGE_SUBSYSTEM_WINDOWS_GUI  2 // Image runs in the Windows GUI subsystem.
#define IMAGE_SUBSYSTEM_WINDOWS_CUI  3 // Image runs in the Windows character
subsystem.
#define IMAGE_SUBSYSTEM_OS2_CUI      5 // image runs in the OS/2 character subsystem.
#define IMAGE_SUBSYSTEM_POSIX_CUI    7 // image runs in the Posix character
subsystem.
#define IMAGE_SUBSYSTEM_NATIVE_WINDOWS 8 // image is a native Win9x driver.
#define IMAGE_SUBSYSTEM_WINDOWS_CE_GUI 9 // Image runs in the Windows CE subsystem.
```

Annex G. Normative: Values for .corflags

Below is the list of possible values accepted by the ILASM **.subsystem** keyword. These are defined in the CLR CorHdr.h header file. Note that **.subsystem** does not accept the symbolic name – you must supply the numeric value:

```
// CLR Header entry point flags.  
    COMIMAGE_FLAGS_ILONLY           = 0x00000001,  
    COMIMAGE_FLAGS_32BITREQUIRED    = 0x00000002,  
    COMIMAGE_FLAGS_IL_LIBRARY       = 0x00000004,  
    COMIMAGE_FLAGS_TRACKDEBUGDATA   = 0x00010000,
```

Annex H. Normative: Values for Hash Algorithm ID

Below is the list of possible values accepted by ILASM for Hash Algorithm ID. These are defined in the Windows WinCrypt.h header file. Note that ILASM does not accept the symbolic name – you must supply the numeric value:

```
// algorithm identifier definitions
#define CALG_MD2 (ALG_CLASS_HASH | ALG_TYPE_ANY | ALG_SID_MD2)
#define CALG_MD4 (ALG_CLASS_HASH | ALG_TYPE_ANY | ALG_SID_MD4)
#define CALG_MD5 (ALG_CLASS_HASH | ALG_TYPE_ANY | ALG_SID_MD5)
#define CALG_SHA (ALG_CLASS_HASH | ALG_TYPE_ANY | ALG_SID_SHA)
#define CALG_SHA1 (ALG_CLASS_HASH | ALG_TYPE_ANY | ALG_SID_SHA1)
#define CALG_MAC (ALG_CLASS_HASH | ALG_TYPE_ANY | ALG_SID_MAC)
#define CALG_RSA_SIGN (ALG_CLASS_SIGNATURE | ALG_TYPE_RSA | ALG_SID_RSA_ANY)
#define CALG_DSS_SIGN (ALG_CLASS_SIGNATURE | ALG_TYPE_DSS | ALG_SID_DSS_ANY)
#define CALG_RSA_KEYX (ALG_CLASS_KEY_EXCHANGE | ALG_TYPE_RSA | ALG_SID_RSA_ANY)
#define CALG_DES (ALG_CLASS_DATA_ENCRYPT | ALG_TYPE_BLOCK | ALG_SID_DES)
#define CALG_3DES_112 (ALG_CLASS_DATA_ENCRYPT | ALG_TYPE_BLOCK | ALG_SID_3DES_112)
#define CALG_3DES (ALG_CLASS_DATA_ENCRYPT | ALG_TYPE_BLOCK | ALG_SID_3DES)
#define CALG_RC2 (ALG_CLASS_DATA_ENCRYPT | ALG_TYPE_BLOCK | ALG_SID_RC2)
#define CALG_RC4 (ALG_CLASS_DATA_ENCRYPT | ALG_TYPE_STREAM | ALG_SID_RC4)
#define CALG_SEAL (ALG_CLASS_DATA_ENCRYPT | ALG_TYPE_STREAM | ALG_SID_SEAL)
#define CALG_DH_SF (ALG_CLASS_KEY_EXCHANGE | ALG_TYPE_DH | ALG_SID_DH_SANDF)
#define CALG_DH_EPHEM (ALG_CLASS_KEY_EXCHANGE | ALG_TYPE_DH | ALG_SID_DH_EPHEM)
#define CALG_AGREEDKEY_ANY (ALG_CLASS_KEY_EXCHANGE | ALG_TYPE_DH | ALG_SID_AGREED_KEY_ANY)
#define CALG_KEYX (ALG_CLASS_KEY_EXCHANGE | ALG_TYPE_DH | ALG_SID_KEYX)
#define CALG_HUGHES_MD5 (ALG_CLASS_KEY_EXCHANGE | ALG_TYPE_ANY | ALG_SID_MD5)
#define CALG_SKIPJACK (ALG_CLASS_DATA_ENCRYPT | ALG_TYPE_BLOCK | ALG_SID_SKIPJACK)
#define CALG_TEK (ALG_CLASS_DATA_ENCRYPT | ALG_TYPE_BLOCK | ALG_SID_TEK)
#define CALG_CYLINK_MEK (ALG_CLASS_DATA_ENCRYPT | ALG_TYPE_BLOCK | ALG_SID_CYLINK_MEK)
#define CALG_SSL3_SHAMD5 (ALG_CLASS_HASH | ALG_TYPE_ANY | ALG_SID_SSL3SHAMD5)
#define CALG_SSL3_MASTER (ALG_CLASS_MSG_ENCRYPT | ALG_TYPE_SECURECHANNEL | ALG_SID_SSL3_MASTER)
#define CALG_SCHANNEL_MASTER_HASH (ALG_CLASS_MSG_ENCRYPT | ALG_TYPE_SECURECHANNEL | ALG_SID_SCHANNEL_MASTER_HASH)
#define CALG_SCHANNEL_MAC_KEY (ALG_CLASS_MSG_ENCRYPT | ALG_TYPE_SECURECHANNEL | ALG_SID_SCHANNEL_MAC_KEY)
#define CALG_SCHANNEL_ENC_KEY (ALG_CLASS_MSG_ENCRYPT | ALG_TYPE_SECURECHANNEL | ALG_SID_SCHANNEL_ENC_KEY)
#define CALG_PCT1_MASTER (ALG_CLASS_MSG_ENCRYPT | ALG_TYPE_SECURECHANNEL | ALG_SID_PCT1_MASTER)
#define CALG_SSL2_MASTER (ALG_CLASS_MSG_ENCRYPT | ALG_TYPE_SECURECHANNEL | ALG_SID_SSL2_MASTER)
```

```
#define CALG_TLS1_MASTER  
(ALG_CLASS_MSG_ENCRYPT|ALG_TYPE_SECURECHANNEL|ALG_SID_TLS1_MASTER)  
#define CALG_RC5          (ALG_CLASS_DATA_ENCRYPT|ALG_TYPE_BLOCK|ALG_SID_RC5)  
#define CALG_HMAC          (ALG_CLASS_HASH | ALG_TYPE_ANY | ALG_SID_HMAC)
```

These definitions, in-turn, depend upon the following, also drawn from the same header file --

```
//  
// Algorithm IDs and Flags  
//  
  
// ALG_ID crackers  
#define GET_ALG_CLASS(x)          (x & (7 << 13))  
#define GET_ALG_TYPE(x)          (x & (15 << 9))  
#define GET_ALG_SID(x)           (x & (511))  
  
// Algorithm classes  
#define ALG_CLASS_ANY             (0)  
#define ALG_CLASS_SIGNATURE      (1 << 13)  
#define ALG_CLASS_MSG_ENCRYPT     (2 << 13)  
#define ALG_CLASS_DATA_ENCRYPT    (3 << 13)  
#define ALG_CLASS_HASH           (4 << 13)  
#define ALG_CLASS_KEY_EXCHANGE   (5 << 13)  
  
// Algorithm types  
#define ALG_TYPE_ANY              (0)  
#define ALG_TYPE_DSS              (1 << 9)  
#define ALG_TYPE_RSA              (2 << 9)  
#define ALG_TYPE_BLOCK            (3 << 9)  
#define ALG_TYPE_STREAM           (4 << 9)  
#define ALG_TYPE_DH               (5 << 9)  
#define ALG_TYPE_SECURECHANNEL    (6 << 9)  
  
// Generic sub-ids  
#define ALG_SID_ANY               (0)  
  
// Some RSA sub-ids  
#define ALG_SID_RSA_ANY           0  
#define ALG_SID_RSA_PKCS          1  
#define ALG_SID_RSA_MSATWORK      2  
#define ALG_SID_RSA_ENTRUST        3  
#define ALG_SID_RSA_PGP           4  
  
// Some DSS sub-ids  
//
```

```
#define ALG_SID_DSS_ANY 0
#define ALG_SID_DSS_PKCS 1
#define ALG_SID_DSS_DMS 2

// Block cipher sub ids
// DES sub_ids
#define ALG_SID_DES 1
#define ALG_SID_3DES 3
#define ALG_SID_DESX 4
#define ALG_SID_IDEA 5
#define ALG_SID_CAST 6
#define ALG_SID_SAFERSK64 7
#define ALG_SID_SAFERSK128 8
#define ALG_SID_3DES_112 9
#define ALG_SID_CYLINK_MEK 12
#define ALG_SID_RC5 13

// Fortezza sub-ids
#define ALG_SID_SKIPJACK 10
#define ALG_SID_TEK 11

// KP_MODE
#define CRYPT_MODE_CBCI 6 // ANSI CBC Interleaved
#define CRYPT_MODE_CFBP 7 // ANSI CFB Pipelined
#define CRYPT_MODE_OFBP 8 // ANSI OFB Pipelined
#define CRYPT_MODE_CBCOFM 9 // ANSI CBC + OF Masking
#define CRYPT_MODE_CBCOFMI 10 // ANSI CBC + OFM Interleaved

// RC2 sub-ids
#define ALG_SID_RC2 2

// Stream cipher sub-ids
#define ALG_SID_RC4 1
#define ALG_SID_SEAL 2

// Diffie-Hellman sub-ids
#define ALG_SID_DH_SANDF 1
#define ALG_SID_DH_EPHEM 2
#define ALG_SID_AGREED_KEY_ANY 3
#define ALG_SID_KEYA 4

// Hash sub ids
#define ALG_SID_MD2 1
#define ALG_SID_MD4 2
#define ALG_SID_MD5 3
```



```
#define ALG_SID_SHA 4
#define ALG_SID_SHA1 4
#define ALG_SID_MAC 5
#define ALG_SID_RIPEMD 6
#define ALG_SID_RIPEMD160 7
#define ALG_SID_SSL3SHAMD5 8
#define ALG_SID_HMAC 9

// secure channel sub ids
#define ALG_SID_SSL3_MASTER 1
#define ALG_SID_SCHANNEL_MASTER_HASH 2
#define ALG_SID_SCHANNEL_MAC_KEY 3
#define ALG_SID_PCT1_MASTER 4
#define ALG_SID_SSL2_MASTER 5
#define ALG_SID_TLS1_MASTER 6
#define ALG_SID_SCHANNEL_ENC_KEY 7
```

Annex I. Normative: Values for .os

Below is the list of possible values accepted by ILASM for the **.os** keyword. These are defined in the Windows WinBase.h header file. Note that ILASM does not accept the symbolic name – you must supply the numeric value:

```
#define VER_PLATFORM_WIN32s          0
#define VER_PLATFORM_WIN32_WINDOWS  1
#define VER_PLATFORM_WIN32_NT       2
```

Annex J.Normative: Values for .processor

Below is the list of possible values accepted by ILASM for the **.processor** keyword. These are defined in the Windows WinNT.h header file. Note that ILASM does not accept the symbolic name – you must supply the numeric value:

```
#define PROCESSOR_INTEL_386      386
#define PROCESSOR_INTEL_486      486
#define PROCESSOR_INTEL_PENTIUM 586
#define PROCESSOR_MIPS_R4000     4000    // incl R4101 & R3910 for Windows CE
#define PROCESSOR_ALPHA_21064    21064
#define PROCESSOR_PPC_601        601
#define PROCESSOR_PPC_603        603
#define PROCESSOR_PPC_604        604
#define PROCESSOR_PPC_620        620
#define PROCESSOR_HITACHI_SH3    10003   // Windows CE
#define PROCESSOR_HITACHI_SH3E   10004   // Windows CE
#define PROCESSOR_HITACHI_SH4    10005   // Windows CE
#define PROCESSOR_MOTOROLA_821    821     // Windows CE
#define PROCESSOR_SHx_SH3        103     // Windows CE
#define PROCESSOR_SHx_SH4        104     // Windows CE
#define PROCESSOR_STRONGARM       2577    // Windows CE - 0xA11
#define PROCESSOR_ARM720          1824    // Windows CE - 0x720
#define PROCESSOR_ARM820          2080    // Windows CE - 0x820
#define PROCESSOR_ARM920          2336    // Windows CE - 0x920
#define PROCESSOR_ARM_7TDMI       70001   // Windows CE
```

Annex K. Normative: Default Marshalling - Unmanaged to Managed

The following type library type	Is imported as the following Type	Element	With the following ¹ MarshalAs Attribute	System Type	Built-in Language	
					VB	Mgd C
char, boolean, small	I1			System.SByte	n/a	byte
wchar_t, short	I2			System.Int16	short	short
long, int	I4			System.Int32	integer	long
hyper	I8			System.Int64	long	__int6
unsigned char, byte	UI1			System.Byte	byte	byte
unsigned short	UI2			System.UInt16	N/A	unsign short
unsigned long, unsigned int	UI4			System.UInt32	N/A	unsign long
unsigned hyper	UI8			System.UInt64	N/A	unsign __int6
single	R4			System.Single	single	float
double	R8			System.Double	double	double
VARIANT_BOOL	Boolean		VariantBool		boolean	bool
void *	I4				n/a	n/a
	change to UI4 post Beta 1		Add MarshalAs attrib post Beta 1			
HRESULT	UI4		Error		n/a	unsign long
SCODE	UI4		Error		n/a	unsign long
BSTR	STRING		BStr	System.String	string	n/a
LPSTR	or STRING		LPStr	System.String	string	n/a
[string, ...] char *						
LPWSTR	or STRING		LPWStr	System.String	string	n/a
[string, ...] wchar_t *						
VARIANT	OBJECT			System.Object		
DECIMAL	VALUETYPE <System.Decimal>			System.Decimal		
DATE	VALUETYPE <System.DateTime>			System.DateTime		

¹ All entries in this column are from the System.Runtime.InteropServices.UnmanagedType enum. The unmanaged type is supplied with the MarshalAs attribute.

² Blanks in this column indicate that the language has no built-in type that maps to that type. In such cases, the types from the System namespace can be used directly.

The following type library type	Is imported as the following Type	as the Element	With the following ¹ MarshalAs Attribute	System Type	Built-in Lang	
					VB	M
GUID	VALUETYPE			System.Guid		
	<System.Guid>					
CURRENCY	VALUETYPE			System.Currency		
	<System.Currency>					
IUnknown *	OBJECT		IUnknown	System.Object		
IDispatch *	OBJECT		IDispatch	System.Object		
SAFEARRAY(<i>type</i>)	SZARRAY(<i>type</i>)		SAFEARRAY	<i>type</i> []		
typedef <i>BaseType MyType</i>	<i>BaseType</i>					
<i>MyStruct</i>	VALUETYPE					
	< <i>MyStruct</i> >					
<i>MyEnum</i>	VALUETYPE					
	< <i>MyEnum</i> >					
<i>MyInterface</i> *	CLASS		Interface			
	< <i>MyInterface</i> >					
<i>MyCoClass</i>	CLASS		Interface			
	<_Class> ³					
Pointer Types						
char *, boolean *, small *	ByRef I1			System.SByte	n/a	b
wchar_t *, short *	ByRef I2			System.Int16	short	s
long *, int *	ByRef I4			System.Int32	integer	I
hyper *	ByRef I8			System.Int64	long	—
unsigned char *, byte *	ByRef UI1			System.Byte	byte	b
unsigned short *	ByRef UI2			System.UInt16	N/A	u s
unsigned long *, unsigned int *	ByRef UI4			System.UInt32	N/A	u l
unsigned hyper *	ByRef UI8			System.UInt64	N/A	u —
single *	ByRef R4			System.Single	single	f
double *	ByRef R8			System.Double	double	d
VARIANT_BOOL *	ByRef Boolean		VariantBool		boolean	b
void **	ByRef I4				n/a	n
	change to UI4 post Beta 1		Add MarshalAs attrib post Beta 1			

³ If a coclass is used as a parameter type in a type library, the coclass's default interface is used in its place. In this table, the interface _Class is assumed to be the default interface for the coclass MyClass.

The following type library type	Is imported as the following Type	Element	With the following ¹ MarshalAs Attribute	System Type	Built-in Language	
					VB	Mgd C
HRESULT *	ByRef I4		Error		n/a	unsign short
	change to UI4 post Beta 1					
SCODE *	ByRef I4		Error		n/a	unsign short
	change to UI4 post Beta 1					
BSTR *	ByRef STRING		BStr	System.String	string	n/a
LPSTR *	ByRef STRING		LPStr	System.String	string	n/a
LPWSTR *	ByRef STRING		LPWStr	System.String	string	n/a
VARIANT *	ByRef OBJECT			System.Object		
DECIMAL *	ByRef VALUETYPE <System.Decimal>			System.Decimal		
DATE *	ByRef VALUETYPE <System.DateTime>			System.DateTime		
GUID *	ByRef VALUETYPE <System.Guid>			System.Guid		
CURRENCY *	ByRef VALUETYPE <System.Currency>			System.Currency		
IUnknown **	ByRef OBJECT		IUnknown	System.Object		
IDispatch **	ByRef OBJECT		IDispatch	System.Object		
SAFEARRAY(<i>type</i>) *	ByRef SZARRAY(<i>type</i>)		SAFEARRAY	<i>type</i> []		
typedef <i>BaseType MyType</i>	ByRef <i>BaseType</i>					
<i>MyStruct</i> *	ByRef VALUETYPE < <i>MyStruct</i> >					
<i>MyEnum</i> *	ByRef VALUETYPE < <i>MyEnum</i> >					
<i>MyInterface</i> **	ByRef CLASS < <i>MyInterface</i> >	Interface				
<i>MyCoClass</i> *	ByRef CLASS <_Class> ⁴	Interface				

⁴ If a coclass is used as a parameter type in a type library, the coclass's default interface is used in its place. In this table, the interface _Class is assumed to be the default interface for the coclass MyClass.

Annex L. Normative: Default Marshalling - Managed to Unmanaged

The following Element Type	With the following MarshalAs Attribute ⁵	Is converted to the following Type library type	System Type	Built-in Language Type	
				VB	Mgd C++
Boolean		VARIANT_BOOL	System.Boolean	boolean	bool
Boolean (Add post beta1)	Bool	boolean	System.Boolean	boolean	bool
I1		char	System.SByte	n/a	byte
I2		short	System.Int16	short	short
I4		long	System.Int32	integer	long
I8		hyper	System.Int64	long	__int64
UI1		unsigned char	System.Byte	byte	byte
UI2, CHAR		unsigned short	System.UInt16 System.Char	n/a	unsigned short,
CHAR	Char	char	System.Char	n/a	char
UI2	Error	SCODE	System.UInt16	n/a	unsigned short
UI4		unsigned long, unsigned int	System.UInt32	N/A	unsigned long
UI4	Error	SCODE		n/a	n/a
UI4 (Add post beta 1)	Void	void *		n/a	n/a
UI8		unsigned hyper	System.UInt64	N/A	unsigned __int64
R4		single	System.Single	single	float
R8		double	System.Double	double	double
STRING		BSTR	System.String	string	n/a
STRING	BStr	BSTR	System.String	string	n/a
STRING	LPStr	LPSTR	System.String	string	n/a
STRING	LPWStr	LPWSTR	System.String	string	n/a
OBJECT		VARIANT	System.Object		
OBJECT	Struct	VARIANT	System.Object		
OBJECT	LPStruct	VARIANT *	System.Object		

⁵ All entries in this column are from the System.Runtime.InteropServices.UnmanagedType enum. The unmanaged type is supplied with the MarshalAs attribute.

⁶ Blanks in this column indicate that the language has no built-in type that maps to that type. In such cases, the types from the BCL can be used directly.

The following Element Type	With the following MarshalAs Attribute ⁵	Is converted to the following Type library type	System Type	Built-in Language Type ⁶		
				VB	Mgd C++	C
OBJECT	IUnknown	IUnknown *	System.Object			
OBJECT	IDispatch	IDispatch *	System.Object			
VALUETYPE <System.Decimal>		DECIMAL	System.Decimal			
VALUETYPE <System.DateTime>		DATE	System.DateTime			
VALUETYPE <System.Guid>		GUID	System.Guid			
VALUETYPE <System.Currency>		CURRENCY	System.Currency			
SZARRAY(<i>type</i>) <i>BaseType</i>	SAFEARRAY	SAFEARRAY(<i>type</i>) typedef <i>BaseType</i> <i>MyType</i>	<i>type</i> []			
VALUETYPE < <i>MyStruct</i> >		<i>MyStruct</i>				
VALUETYPE < <i>MyEnum</i> >		<i>MyEnum</i>				
CLASS < <i>MyInterface</i> >	Interface	<i>MyInterface</i>				
CLASS < <i>MyClass</i> >	Interface	_MyClass				
Reference Types						
ByRef I4 (Add post beta 1)	Void	void *		N/A	Void	?
ByRef Boolean		VARIANT_BOOL *	System.Boolean	boolean	bool	b
ByRef Boolean (Add post beta 1)	Bool	boolean	System.Boolean	boolean	bool	b
ByRef I1		char *	System.SByte	boolean	byte	s
ByRef I2		short *	System.Int16	short	short	s
ByRef I4		long *	System.Int32	integer	long	in
ByRef I8		hyper *	System.Int64	long	__int64	lo
ByRef UI1		unsigned char *	System.Byte	byte	byte	b
ByRef UI2, ByRef CHAR		unsigned short *	System.UInt16	N/A	unsigned short	u
ByRef CHAR	Char	char *	System.Char	N/A	char	c
ByRef I2 (not supported until post beta1)	Error	SCODE *	System.Int16	N/A	short	u

The following Element Type	With the following MarshalAs Attribute ⁵	Is converted to the following Type library type	System Type	Built-in Language Type	
				VB	Mgd C++
ByRef UI4		unsigned long *	System.UInt32	N/A	unsigned long
ByRef I4	Error	HRESULT *	System.UInt32	N/A	unsigned long
ByRef UI8 (not supported until post beta1)		unsigned hyper *	System.UInt64	N/A	unsigned __int64
ByRef R4		single *	System.Single	single	float
ByRef R8		double *	System.Double	double	double
ByRef STRING		BSTR *	System.String	string	n/a
ByRef STRING	BStr	BSTR *	System.String	string	n/a
ByRef STRING	LPStr	LPSTR *	System.String	string	n/a
ByRef STRING	LPWStr	LPWSTR *	System.String	string	n/a
ByRef OBJECT		VARIANT *	System.Object		
ByRef OBJECT	Struct	VARIANT *	System.Object		
ByRef OBJECT	IUnknown	IUnknown **	System.Object		
ByRef OBJECT	IDispatch	IDispatch **	System.Object		
ByRef VALUETYPE <System.Decimal>		DECIMAL *	System.Decimal		
ByRef VALUETYPE <System.DateTime>		DATE *	System.DateTime		
ByRef VALUETYPE <System.Guid>		GUID *	System.Guid		
ByRef VALUETYPE <System.Currency>		CURRENCY *	System.Currency		
ByRef SZARRAY(<i>type</i>)	SAFEARRAY	SAFEARRAY(<i>type</i>) *	<i>type</i> []		
ByRef <i>BaseType</i>		typedef <i>BaseType</i> <i>MyType</i>			
ByRef VALUETYPE <MyStruct>		<i>MyStruct</i> *			
ByRef VALUETYPE <MyEnum>		<i>MyEnum</i> *			
ByRef CLASS Interface <MyInterface>		<i>MyInterface</i> *			
ByRef CLASS Interface <MyClass> ⁷		_MyClass *			

⁷ If a coclass is used as a parameter type in a type library, the coclass's default interface is used in its place. In this table, the interface _Class is assumed to be the default interface for the coclass MyCoClass.

Below is the list of possible values accepted by ILASM for the **.processor** keyword. These are defined in the Windows WinNT.h header file. Note that ILASM does not

Annex M. Informative: Metadata Validation Rules

To be supplied.

Annex N. Normative: Explicit Class Layout Rules

To be supplied.

Annex O. Informative: Class Library Design Guidelines

To be supplied.

Free printed copies can be ordered from:

ECMA

114 Rue du Rhône
CH-1204 Geneva
Switzerland

Fax: +41 22 849.60.01

Email: documents@ecma.ch

Files of this Standard can be freely downloaded from the ECMA web site (www.ecma.ch). This site gives full information on ECMA, ECMA activities, ECMA Standards and Technical Reports.

ECMA
114 Rue du Rhône
CH-1204 Geneva
Switzerland

See inside cover page for obtaining further soft or hard copies.