ANEXO 1: MSIL

Segundo Tomazelli (2004, p. 20-21), a plataforma Microsoft .NET é baseada em uma máquina virtual de pilha (*stack-based*), na qual são executados os programas compilados nas linguagens disponíveis para a plataforma. Todos os programas são compilados para *MicroSoft Intermediate Language* (MSIL). A máquina virtual possui um conjunto de instruções para:

a) colocar operandos na pilha de avaliação;

b) efetuar operações com os operandos que estão no topo da pilha, sem considerar o tipo dos mesmos, isto é, não existe, uma instrução `add` específica para o tipo `int64` e outra para o tipo `float64`;

c) retirar os operandos da pilha e armazená-los na memória.

Observa-se que "o MSIL não possui instruções para manipulação de registradores" (RICHTER, 2002, p. 19 apud TOMAZELLI, 2004, p. 31). Existem em torno de 220 instruções. Abaixo são descritas algumas delas[1].

TIPO	CÓDIGO	TIPO	CÓDIGO	TIPO	CÓDIGO	TIPO	CÓDIGO

aritméticas	`add`	lógica	`and`	constantes (`bool`, `int64`, `float64`, `string`)	`//bool`
					`ldc.i4.1`
					`ldc.i4.0`
						de desvio	`brfalse`

	`div`		`//not`
			`ldc.i4.1`
			`xor`
							`br`

	`mul`				`//int64` `ldc.i8`		`brtrue`
	`sub`		`or`			duplicar	`dup`
relacionais	`ceq`	variáveis (declarar, armazenar, usar)	`.locals`		`//float64` `ldc.r8`	entrada / saída	`Readline`
	`cgt`		`stloc`				`Write`
	`clt`		`ldloc`		`ldstr`		

CÓDIGO	PARÂMETRO	DESCRIÇÃO

`add`		executa operação aritmética **soma**
		`v1 := pilha.desempilha`
		`v2 := pilha.desempilha`
		`v3 := v2 + v1`
		`pilha.empilha (v3)`

`and`		executa operação lógica **E**
		`v1 := pilha.desempilha`
		`v2 := pilha.desempilha`
		`v3 := v2 AND v1`
		`pilha.empilha (v3)`

`br`	`label`	desvia para a instrução rotulada com `label`

`brfalse`	`label`	desvia quando o valor do topo da pilha for `false` (`0`) para a instrução rotulada com `label`
		`valor := pilha.desempilha`
		`se v = 0`
		`então desviar para instrução com rótulo label`

`brtrue`	`label`	desvia quando o valor do topo da pilha for `true` (`1`) para a instrução rotulada com `label`
		`valor := pilha.desempilha`
		`se valor = 1`
		`então desviar para instrução com rótulo label`

`ceq`		executa operação relacional **igual**
		`v1 := pilha.desempilha`
		`v2 := pilha.desempilha`
		`v3 := v2 = v1`
		`pilha.empilha (v3)`

[1] Nesse anexo encontram-se apenas algumas instruções de MSIL. Para obter a relação completar, consultar:

MICROSOFT . **MSDN library**: opCodes. [S.l.], 2012. Disponível em: <http://msdn.microsoft.com/en-us/library/system.reflection.emit.opcodes.aspx>. Acesso em: 22 maio 2012.

TOMAZELLI, Giancarlo. **Implementação de um compilador para uma linguagem de programação com geração de código Microsoft .NET Intermediate Language**. 2004. 83 f. Trabalho de Conclusão de curso (Bacharelado em Ciências da Computação) – Universidade Regional de Blumenau, Blumenau. Disponível em: <http://campeche.inf.furb.br/tccs/2004-I/2004-1giancarlotomazellivf.pdf>. Acesso em: 21 maio 2012.

CÓDIGO	PARÂMETRO	DESCRIÇÃO

cgt		executa operação relacional **maior que**
		`v1 := pilha.desempilha`
		`v2 := pilha.desempilha`
		`v3 := v2 > v1`
		`pilha.empilha (v3)`

clt		executa operação relacional **menor que**
		`v1 := pilha.desempilha`
		`v2 := pilha.desempilha`
		`v3 := v2 < v1`
		`pilha.empilha (v3)`

div		executa operação aritmética **divisão**
		`v1 := pilha.desempilha`
		`v2 := pilha.desempilha`
		`se v1 = 0`
		`então` *`ERRO de execução`*`: divisão por 0`
		`  PARAR`
		`senão v3 := v2 / v1`
		`  pilha.empilha (v3)`

dup		duplica o valor do topo da pilha
		`valor := pilha.topo`
		`pilha.empilha (valor)`

ldc.i4.0		empilha a `constante` lógica `false` `pilha.empilha (0)`
ldc.i4.1		empilha a `constante` lógica `true` `pilha.empilha (1)`

ldc.i4.1
xor
		executa operação lógica **NÃO**
		`v1 := pilha.desempilha`
		`v2 := NOT v1`
		`pilha.empilha (v2)`

ldc.i8	`constante`	empilha a `constante` inteira `pilha.empilha (constante)`
ldc.r8	`constante`	empilha a `constante` real `pilha.empilha (constante)`

ldloc	`identificador`	empilha o valor armazenado na variável (`identificador`)
		`valor := identificador`
		`pilha.empilha (valor)`

ldstr	`constante`	empilha a `constante` literal `pilha.empilha (constante)`
.locals	(<variáveis>) onde: – <variáveis> pode ser uma ou mais ocorrências de <tipo> identificador, separadas umas das outras por vírgula; – <tipo> pode ser `bool`, `int64`, `float64` ou `string`	aloca variáveis do tipo especificado em <tipo>

mul		executa operação aritmética **multiplicação**
		`v1 := pilha.desempilha`
		`v2 := pilha.desempilha`
		`v3 := v2 * v1`
		`pilha.empilha (v3)`

or		executa operação lógica **OU**
		`v1 := pilha.desempilha`
		`v2 := pilha.desempilha`
		`v3 := v2 OR v1`
		`pilha.empilha (v3)`

stloc	`identificador`	armazena o valor do topo da pilha na variável (`identificador`) `identificador:= pilha.desempilha`

Sub		executa operação aritmética **subtração**
		`v1 := pilha.desempilha`
		`v2 := pilha.desempilha`
		`v3 := v2 – v1`
		`pilha.empilha (v3)`

CÓDIGO | PARÂMETRO	DESCRIÇÃO

• para leitura de `bool`, `int64` ou `float64`:
`call string [mscorlib]System.Console::ReadLine()`
`call` **`<tipo>`** `[mscorlib]System.<classe>::Parse(string)`
onde:
– `<tipo>` pode ser `bool`, `int64` ou `float64`
– `<classe>` pode ser `Boolean`, `Int64` ou `Double`
• para leitura de `string`:
`call string [mscorlib]System.Console::ReadLine()`
	lê (da entrada padrão) um valor do `<tipo>` especificado e armazena-o no topo da pilha
	`LEIA (valor)`
	`se valor for do tipo` **`<tipo>`**
	`então pilha.empilha (valor)`
	`senão` *`ERRO de execução`*
	`PARAR`

`call void [mscorlib]System.Console::Write(<tipo>)`
onde:
– `<tipo>` pode ser `bool`, `int64`, `float64` ou `string`
	escreve (na saída padrão) o valor do topo da pilha
	`valor := pilha.desempilha`
	`ESCREVA (valor)`

Para converter `int64` para `float64` usar `conv.r8`. Para converter `float64` para `int64` usar `conv.i8`.