### 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:

1. colocar operandos na pilha de avaliação;
2. 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;
3. 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]](#footnote-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 |  | executar operação aritmética **soma**: (1) desempilhar dois valores; (2) somar o segundo valor com o primeiro valor; (3) empilhar o resultado |
| and |  | executar operação lógica **E**: (1) desempilhar dois valores; (2) efetuar **E-lógico** do segundo valor com o primeiro valor; (3) empilhar o resultado |
| br | label | executar desvio incondicional: (1) desviar para a instrução rotulada com label |
| brfalse | label | executar desvio condicional: (1) desempilhar um valor; (2) desviar para a instrução rotulada com label se o valor for false (0), caso contrário, desviar para a próxima instrução |
| brtrue | label | executar desvio condicional: (1) desempilhar um valor; (2) desviar para a instrução rotulada com label se o valor for true (1), caso contrário, desviar para a próxima instrução |
| ceq |  | executar operação relacional **igual**: (1) desempilhar dois valores; (2) efetuar a operação **igual** do segundo valor com o primeiro valor, resultando em um valor lógico; (3) empilhar o resultado |
| cgt |  | executar operação relacional **maior que**: (1) desempilhar dois valores; (2) efetuar a operação **maior que** do segundo valor com o primeiro valor, resultando em um valor lógico; (3) empilhar o resultado |
| clt |  | executar operação relacional **menor que**: (1) desempilhar dois valores; (2) efetuar a operação **menor que** do segundo valor com o primeiro valor, resultando em um valor lógico; (3) empilhar o resultado |
| div |  | executar operação aritmética **divisão**: (1) desempilhar dois valores; (2) se o primeiro valor for 0, *ERRO de execução*: divisão por 0, **PARAR**, caso contrário, dividir o segundo valor pelo primeiro valor; (3) empilhar o resultado |
| dup |  | duplicar o valor do topo da pilha |
| ldc.i4.0 |  | empilhar a constante lógica false (0) |
| ldc.i4.1 |  | empilhar a constante lógica true (1) |
| ldc.i4.1  xor |  | executar operação lógica **NÃO**: (1) desempilhar um valor; (2) efetuar **NÃO-lógico**; (3) empilhar o resultado |
| ldc.i8 | constante | empilhar a constante inteira |
| ldc.r8 | constante | empilhar a constante real |
| ldloc | identificador | empilhar o valor armazenado na variável (identificador) |
| ldstr | constante | empilhar a constante literal |
| .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 | alocar posições de memória para variável (identificador) do tipo especificado em <tipo> |
| mul |  | executar operação aritmética **multiplicação**: (1) desempilhar dois valores; (2) multiplicar o segundo valor pelo primeiro valor; (3) empilhar o resultado |
| or |  | executar operação lógica **OU**: (1) desempilhar dois valores; (2) efetuar **OU-lógico** do segundo valor com o primeiro valor (3) empilhar o resultado |
| stloc | identificador | armazenar o valor do topo da pilha na variável (identificador): (1) desempilhar um valor; (2) armazenar o valor na posição de memória reservada para a variável (identificador) |
| sub |  | executar operação aritmética **subtração**: (1) desempilhar dois valores; (2) subtrair do segundo valor o primeiro valor; (3) empilhar o resultado |

| **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() | | ler um valor do <tipo> especificado e armazenar no topo da pilha: (1) ler da entrada padrão um valor; (2) se o valor não for do <tipo> especificado, *ERRO de execução*, **PARAR**, caso contrário, empilhar o valor lido, conforme o <tipo> especificado |
| call void [mscorlib]System.Console::Write(<tipo>)  onde:   * <tipo> pode ser bool, int64, float64 ou string | | escrever o valor do topo da pilha: (1) desempilhar um valor; (2) escrever na saída padrão o valor, conforme o <tipo> especificado |

Observações:

* para converter int64 para float64 usar conv.r8;
* para converter float64 para int64 usar conv.i8;
* valores do tipo float64 devem ser informados como 10,5 na entrada padrão e 10.5 nas instruções do programa.

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

   [MICROSOFT . **MSDN library**: opCodes. [S.l.], 2012.](http://msdn.microsoft.com/en-us/library/system.reflection.emit.opcodes.aspx) 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. [↑](#footnote-ref-1)