Jasmin[a]

1. Jasmin é um assembler para Java que recebe um descrição textual (ASCII) de classes Java e as converte para arquivos (binários) no formato class file.
2. Foi desenhado para ser um assembler simples com uma relação quase 1-para-1 com os componentes de um class file. Não faz um série de verificações como: (i) verificação da existência de classes sendo referenciadas, (ii) verificação da correta formação dos descritores, (iii) *inlining* de expressões matemáticas, (iv) substituição de variáveis, e (v) suporte a macros.
3. Foi desenhado para que se possa testar rapidamente quase todas as *features* da máquina virtual.
4. Pode ser executado com o comando: `java -jar jasmin.jar file.j`.

[a] Baseado no Jasmin User Guide. (`http://jasmin.sourceforge.net/`)

5. Os comandos de Jasmin podem ser: (i) diretivas, (ii) instruções, ou (iii) rótulos.

6. Diretivas: Comandos de meta-nível que começam com ‘.’. Podem ser: `.catch .class .end .field .implements .interface .limit .line .method .source .super .throws .var.`

7. Exemplos de diretivas:

```
.limit stack 10

.method public myMethod()V

.class Foo
```

8. Instruções: Comandos que manipulam a pilha de operandos. Exemplos:

```
ldc    "Hello World"
iinc   1 -1
bipush 10
```

9. Rótulos: Marcam as instruções para execução de desvios. Não podem começar com números nem conter os caracteres `= , : , . , " , - .` Só podem ser utilizados dentro de métodos e são locais a um método. Exemplo:

```
Label: bipush 10
```

10. Comentários: Começam com ‘;’, seguida de espaço ou tabulação e termina com quebra de linha. Exemplo:

```
; Isso é um comentário.

abc;def              ; ‘def’ não é um comentário.

Ljava/lang/String; ; ‘Ljava/lang/String;’ é um
                     ; token só.

foo ; baz ding       ; Token ‘foo’ seguido de um
                     ; comentário.
```

11. Números e strings: somente decimais e inteiros são reconhecidos.

Ponto flutanete em notação científica não são. Permitido: `1, 123, .25, 0.03, 0xA`. Não permitido: `1e-10, 'a', '\u123'`. Somente os caracteres de escape `"\n" and "\t"` são reconhecidos.

12. Nomes de classe: nome completo de uma classe. Exemplo `java/lang/String`.

13. Descritores de tipo: seguem a notação de class files. (Ver slides sobre JVM.)

14. Nomes de método são dados por uma única string resultante da concatenação de três partes: nome da classe, nome do método e descritor do método segundo descritores de mátodo em class files.

```
foo/baz/Myclass/myMethod(Ljava/lang/String;)V
---------------          --------------------
     |          --------          |
     |             |              |
   class         method       descriptor
```

Uma chamada ao método

```
class mypackage.MyClass {
      int foo(Object a, int b[]) { ... }
}
```

pode ser feita da seguinte maneira:

```
invokevirtual mypackage/MyClass/foo(Ljava/lang/Object;[I)I
```

15. Campos são especificados em Jasmin usando dois tokens, um informando o nome e classe do campo e o outro dando o descritor. Por exemplo:

    ```
    getstatic mypackage/MyClass/my_font Ljava/lang/Font;
    ```

    retorna o valor do campo chamado `"my_font"` na classe `mypackage.MyClass`. O tipo do campo é `"Ljava/lang/Font;"` (i.e. um objeto `Font`).

16. Um arquivo Jasmin começa com diretivas. Exemplo:

```
.source foo.j            ; Arquivo fonte: info para debug
.class public foo        ; Nome completo da classe c/ modif.
.super java/lang/Object ; Nome completo da superclasse
.implements Edible       ; Nome da interface que implementa
.implements java/lang/Throwable ;
```

17. Deifinições de campo.

    `.field <access-spec> <field-name> <descriptor> [ =`

    `<value> ]`

    A declaração `public int foo;` vira `.field public foo I`.

    E a declaração `public static final float PI = 3.14;` vira

    `.field public static final PI F = 3.14`.

18. Declarações de método seguem a forma:

```
.method <access-spec> <method-spec>
        <statements>
.end method
```

A string `<method-spec>` tem o nome e o descritor de tipo do método. Exemplo de mátodo mais básico:

```
.method foo() V
  return
.end method
```

19. Diretivas de métodos: A diretiva `.limit locals <integer>` define o número de variáveis locais que um método necessita.

20. A diretiva `.var <var-number> is <name> <descriptor> from <label1> to <label2>` é usada para definir o nome, descritor de tipo e escopo de uma variável local.

Exemplo de uso da diretiva `.var`:

```
.method foo()V
.limit locals 1
; declara a variável 0 como "int Count;"
; cujo escopo é o código entre Label1 e Label2
;
.var 0 is Count I from Label1 to Label2
Label1:
  bipush 10
  istore_0
Label2:
  return
.end method
```

21. Exemplo: Hello World.

```
public class HelloWorld {
  static public void main(String args[]) {
    System.out.println(‘‘Hello World!’’) ;
  }
}
```

```
.class public HelloWorld
.super java/lang/Object
;
; standard initializer
.method public <init>()V
  aload_0
  invokenonvirtual java/lang/Object/<init>()V
  return
.end method

.method public static main([Ljava/lang/String;)V
  .limit stack 2
  ; push System.out onto the stack
  getstatic java/lang/System/out Ljava/io/PrintStream;
  ; push a string onto the stack
  ldc "Hello World!"
  ; call the PrintStream.println() method.
  invokevirtual java/io/PrintStream/println(Ljava/lang/String;)V
  ; done
  return
.end method
```

Instruções Jasmin[a]

1. Já vimos os tipos de comandos da linguagem Jasmin assim como exemplos de diretivas (i.e. permitem a definição de métodos), instruções (i.e. para manipulação da pilha de operandos) e rótulos (i.e. em instruções "jump"). Vamos ver agora os tipos de instruções em Jasmin através de exemplos.

2. Instruções que fazem uso de variáveis locais: São parametrizadas pelo número que representa a variável local. São elas: `ret`, `aload`, `astore`, `dload`, `dstore`, `fload`, `fstore`, `iload`, `istore`, `lload`, `lstore`.
   Exemplos:

```
aload 1    ; empilha o valor na variável local 1
ret 2      ; retorna o fluxo de controle para o
           ; endereço armazenado na variável local 2
```

[a]Baseado no manual "Jasmin Instructions", por Jonathan Meyer. (`http://jasmin.sourceforge.net/instructions.html`)

3. Instruções para manipulação de inteiros:

```
bipush 100    ; empilha 100
iinc 3 -10    ; subtrai 10 da variável local 3
```

4. Instruções de desvio: Diversos tipos de desvio condicional e incondicional.

```
Label1:
    goto Label1    ; ‘‘jump’’ para o código no Label1
                   ; (loop infinito!)
```

5. Operações sobre classes e objetos: São as seguintes: `anewarray`, `checkcast`, `instanceof`, `new`.

   Exemplo:

```
new java/lang/String   ; cria uma instância de uma string
```

6. Instruções para invocar métodos: `invokenonvirtual`, `invokestatic`, `invokevirtual` e `invokeinterface`.

   Exemplo:

```
; chama java.io.PrintStream.println(String);
invokevirtual
      java/io/PrintStream/println(Ljava/lang/String;)V

; O segundo parâmetro de invokeinterface diz quantos
; parâmetros o método recebe.
invokeinterface foo/Baz/myMethod(I)V 1
```

7. Instruções para manipulação de atributos: `getfield`, `getstatic`, `putfield`, `putstatic`.

   Examplo:

```
; o atributo java.lang.System.out é um PrintStream
getstatic java/lang/System/out Ljava/io/PrintStream;
```

8. Instruções para manipulação de vetores: `newarray`, `multianewarray`.

   Examplo:

```
newarray float
```

```
; aloca uma matriz bidimensional de inteiros
multianewarray [[[I 2
```

9. Instruções para manipular constantes: `ldc`, `ldc_w`.

   Examplo:

```
ldc "Hello World"    ; empilha uma String
ldc_w 3.141592654    ; empilha um double
```

10. Instruções para `switch`: `lookupswitch` e `tableswitch`.

    Examplo:

```
 ; Se o int na pilha é 3, ‘‘jump’’ para Label1.
 ; Se é 5, ‘‘jump’’ para Label2.
 ; Caso contrário, desvia para DefaultLabel.
 lookupswitch
         3 : Label1
         5 : Label2
   default : DefaultLabel
   Label1: ; tinha 3 na pilha
   Label2: ; tinha 5 na pilha
   DefaultLabel: ; tinha outra coisa
```

11. Instruções que não requerem parâmetros: (Diversas.)

Examplo:

```
pop          ; remove o item no topo da pilha
iconst_1     ; empilha 1
swap         ; troca os dois primeiros elementos da
             ; pilha de posição
```