Bytecode y máquina virtual de CLISP

Juan Sebastian Navarro

CLISP: Descripción general

- Implementación de Common Lisp.
- Primera versión: 1987.
- Última versión estable (2.49): 2010.
- Para el proyecto: 2.49.
- `http://www.clisp.org`

CL: Código intermedio

- Depende de la implementación:
  - SBCL: assembly,
  - ABCL: JVM,
  - CMUCL, CLISP: código intermedio propio,
  - etc.

CLISP: Máquina Virtual

- Máquina de stack.

CLISP: Máquina Virtual

- Máquina de stack.
- 2 stacks:
  - STACK: stack principal
  - SP

CLISP: Máquina Virtual

- Máquina de stack.
- 2 stacks:
  - STACK: stack principal
  - SP
- otras estructuras (registros?):
  - values: tope del stack
  - mv_count: siempre igual a 1
  - mv_space: no utilizado
  - PC
- 256 instrucciones.

Algunas instrucciones importantes

- (NIL) : value1 := NIL
- (PUSH-NIL n)
- (T) : value1 := T
- (CONST n) : value1 := consts[n]
- (PUSH) : *−−STACK := value1
- (POP) : value1 := *STACK++
- (LOAD n) : value1 := *(STACK+n)
- (STORE n) : *(STACK+n) := value1
- (CALL k n) : llama a la función consts[n] con argumentos *(STACK+k-1), ..., *(STACK+0)
- (SKIP&RET n) : vacía los n primeros elementos del stack, y retorna

Función compilada

- nombre : Sólo para debugging.
- codevec : Código
- constantes : Otros

Ejemplo

Codevec

- spdepth_1 (2 bytes): primera parte de la máxima profundidad en SP (para el proyecto: 0)
- spdepth_jmpbufsize (2 bytes): jmpbufsize. (para el proyecto: 0).
- numreq (2 bytes): cantidad de parámetros requeridos.
- numopt (2 bytes): cantidad de parámetros opcionales (0).
- flags (1 byte):
  - bit 0 (1 si tiene &rest): 0
  - bit 7 (1 si tiene &key): 0
  - bit 6 (1 si tiene &allow-other-keys): 0
  - bit 5 (siempre 0): 0
  - bit 4 (función genérica): 1
  - bit 3 (se retorna el código y no se ejecuta): 1
  - bit 2: X
  - bit 1: X
- signature (1 byte): en el proyecto: numreq+1
- instrucciones.

Instrucciones

- 1 byte hexadecimal.
- Archivo: bytecode.java
- Argumentos: se codifican según la instrucción en uno o dos bytes.
- Más detalles:
  `http://www.clisp.org/impnotes/instr-struct.html`

Constantes

- Vector de constantes.
- Puede contener cualquier tipo de elemento.
- Se acceden por posición con `CONST`.
- Debe contener todos los valores literales, y los símbolos de las funciones que se utilicen.

Constantes, formato

- enteros: se agrega un punto (ej: 57.)
- Strings: entre " " (ej: "un string")
- Chars: precedido de #\ (ej: #\A)
- Bool: "true": `|COMMON-LISP-USER|::|T|`
- Bool: "false": `|COMMON-LISP-USER|::|NIL|`
- Símbolos de función: `|PACKAGE|::|NAME|` (ej: `|COMMON-LISP-USER|::|IS-PRIME|`)

Variables “normales”

- Posiciones en el stack.
- Se acceden con `STORE` y `LOAD`.
- Las posiciones son relativas a STACK: Cambian al hacer push o pop.

Definición de funciones

- defun
- Codevec: `#20Y(00 00 00 00 00 00 00 00 20 01 DA 2F 01 DA DC 32 9C C5 19 01)`
- Vector de constantes:
  - Símbolo de la función, por ejemplo: `|COMMON-LISP-USER|::|IS-ZERO|`
  - `|SYSTEM|::|REMOVE-OLD-DEFINITIONS|`
  - Función compilada a crear (vector de 3 elementos).

Archivo compilado

- Versión.
- Charset.
- Definiciones de funciones.

Ejecución

Para cargar un archivo: (load “/path/al/archivo”)
Para ejecutar una función F con 2 parámetros: (F parametro1 parametro2)

Funciones predefinidas

- Archivo sather.fas

Ejemplo

Ver simple.fas