

# Structure of Programming Languages – Lecture 10

CS 636 – 536

June 15, 2009

- 1 The Forth System
  - Forth Objects
  - Dump and Decompile
  - FORTH types.

# Looking under the hood.

The classical structure of a FORTH definition:

- name field: the object's name, variable length.  
a flag bit and a count (4 bytes), followed by the letters.
- link field: a pointer used by the system to organize the dictionary.
- code field: a type code – variable, constant, function, or user-defined.
- parameter field: the body of a function or value of a variable.

Modern variations:

- The name field might be in a different memory segment from the code and parameter fields.
- The link field might be absent or be used as a pointer from the name part to the code part.
- Most modern systems work with 4-byte addresses and integers. (Old ones were 2.)

# Seeing what is there.

FORTH provides tools to look at the compiled code.

- `.S` ( print the stack without destroying it.)
- `words` ( print all the words in the dictionary, newest first.)
- `see area` ( display the source code for the area function.)  
`see dup` ( display the assembly code for dup.)
- `debug` ( print the stack without destroying it.)
- `Dump`: print the hex byte codes from the compiled function.  
Note: the numbers below are all interpreted as hex, even though I did not tell FORTH to do that.
  - ' `area C - 100 dump` ( print the definition of area.)
  - ' `longnamefunction 18 - 100 dump`  
( go back more for longer names.)

## Interpreting a dump.

Every FORTH system uses a different organizational strategy.

- 1 Typing `' name` puts the code-field of an object on the stack.
- 2 Start with that address and back off a bit: `' name 8 -`
- 3 Dump enough bytes to see the name field of the next object in the dictionary.
- 4 Look for the name field. If you can't see the whole name and the count, back off more and dump again. Adjust the back-off amount so that your count shows in column 0, 4, 8, or 12 of the dump.
- 5 Look after the name field. You might see a marker like `20 20 20 20` or `FF FF FF FF` . Not including that marker, the bytes between the end of one name and the start of the next are EITHER the entire code and parameter fields OR a pointer to where that info is stored.

## Interpreting a dump.

After you find the parameter field of a function, you can begin to decode it.

- 1 Typing `' dup .` prints the code-field of `dup`. Do this with all of the basic FORTH words in the function and write down the addresses that are displayed.
- 2 Start with the first operation in your function code. Find its hex code in the dump. Remember that, on an Intel chip, the order of the bytes is reversed.
- 3 Triple-space your dump in a text file. Write the name of each operation that you can identify under its code. There will be some words you cannot find, and some codes that do not correspond to words. Highlight all of them.
- 4 Try to figure out what these unknown codes do. Hint: they were produced by the compiler to translate control structures in the code.

# FORTH data types, built-in and user-defined.

- Constant, Variable, and Function. (Look at the code field.)
- You can build your own type. It will have a unique code field code. You must supply both the compile-time and run-time behavior for the new type.

```
• : boxType ( d1 d2 >>> )  
  create ( Specify compile-time actions.)  
    2dup , , ( store 2 dims in the dictionary.)  
    * dup , ( store total size, also.)  
    4 * allot ( allocate space for elements.)  
  does> ( Specify run-time actions.)  
    6 + ( put adr of 1st value byte on stack.)  
;
```