Software II: Principles of Programming Languages

The FORTH Programming Language

What is FORTH

- FORTH is a threaded interpreted language developed by Charles Moore.
  - FORTH stores all symbols in a dictionary.
  - When running a program the interpreter looks up these in a dictionary, finds its address and executes the code stored at that address.
  - FORTH programs runs very quickly and experienced FORTH programmers claim that their programs can be developed as fast as in higher-level languages.

Expressions in FORTH

- FORTH uses a stack to manage data.
- Expressions in FORTH are al written in postfix notation:

```
3 4 +
3 4 * 5 +
3 4 * 5 6 * +
```

- A number is printed using the period:

```
3  4  +  .  7  ok
3  4 * 5 + . 17  ok
3  4 * 5 6 * + . 42  ok
```

Stack Management in FORTH

- FORTH provides several words for basic stack management:
  - swap – swithces the two top items on the stack
  - dup – places a copy of the top item on the stack
  - rot – rotates the top three items on the stack so that the bottommost of the three items in now on top.
  - drop – removes the top item from the stack
- Examples:

```
3 4 swap . . 3 4  ok
3 4 5 rot . . . 3 5 4  ok
3 4 dup . . . 4 4 3  ok
3 4 over . 3  ok..
```

Defining A New Word

- Definitions begin with a colon and then the name of the new word:

```
:  2times dup + . ;
```

- A defined word can be used immediately afterward:

```
4 2times 8 ok
```

Displaying Character Data in FORTH

- You can print a character string by writing ." to begin a string and " at the end inside a definition:

```
: hithere ." Hello, world " ;  ok
hithere Hello, world  ok
```

- You can print a character using the emit word if you first place the ASCII value on the stack:

```
: hithere ." Hello, world " ;  ok
42 emit * ok
```

Variables in FORTH

- You can declare a variable in FORTH by writing:

```
variable amount ok
```

- You can save integer (or single character) data here by writing:

```
42 amount !  ok
```

- You can retrieve it by writing:

```
amount @ . 42  ok
```

Constants in FORTH

- Constants can be declared by writing:

```
14 constant idno ok
```

- When you invoke the name, FORTH fetches it value:

```
idno . 14  ok
```

Loops in FORTH

- Counting loops can be created in FORTH:

```
: kilogreet 10 1 do ." hi, there " Cr loop ;  ok
```

- You can use i to get the loops's index:

```
: counttoten 10 1 do i . loop ;  ok
counttoten 1 2 3 4 5 6 7 8 9  ok
: countdown begin 2 / dup . dup  1 <
  until ;  ok
```

- Conditional loop can be created in FORTH:

```
: countdown begin 2 / dup . dup  1 <
  until ;  ok
4 countdown 2 1 0  ok.
8 countdown 4 2 1 0  ok..
```

Declaring an Array

- You can declare an array in FORTH by writing:

```
variable myarray 20 allot ok
```

- This sets aside 20 bytes of storage (using Win32Forth, every integer is 4 bytes in size.)

Using an Array Element

- You can assign myarray[3] the value 25 by writing

```
25 myarray 3 4 * + !
```

- You can retrieve it by writing:

```
myarray 3 4 * @ . 25  ok
```

- We can define a word that gives us the array element's address by writing:

```
: addrxi 4 * + ;
```

savethem and **fetchthem**

```
: savethem 4 0 do myarray i addrxi ! loop ;  ok
8 6 4 2 savethem  ok
: fetchthem 4 0 do myarray i addrxi @ . loop ;  ok
fetchthem 2 4 6 8  ok
```

Conditionals

- An IF-THEN construction:

```
: posmsg 0 > if ." positive " then ; ok
43 posmsg positive  ok
```

- An IF-THEN-ELSE construction:

```
: messg 0 > if ." positive " else ."
  negative " then ;  ok
45 messg positive  ok
-45 messg negative  ok
```

Calculating an average

```
: average dup numvalues ! 1 do + loop
numvalues @ / . ;
```

Floating Point Values

- The average before will:
  `5 3 / . 1 ok`
- To convert to float, we must first convert to double:
  `5 S>D D>F 3 S>D D>F F/ F.`
- Float values are placed on a separate stack so they do not interfere with integer values on the integer stack.

Float Operations

All the float operations begin with F:

- Arithmetic operations: **`F+ F- F* F/`**
- Variables and Constants
  `FVARIABLE FCONSTANT F! F@`
- Comparisons **`F> F< F=`**
- Basic functions: **`FSQRT FMIN FMAX FSIN FCOS FABS`**
- Stack Manipulation **`FSWAP FDUP FROT FDROP`**

Converting Back to Integer

- Converting back to integer is the reverse:
  `1.0E0 F>D D>S . 1 ok`