home ⇐ course01 course03 ⇒

Implementing a FORTH virtual machine - 2

course02.c (140 lines) - interpreter

This is only a small extension to course01. We are introducing strings. Other than in traditional FORTH we handle strings as regular literal, like numbers. In traditional FORTH strings are read from input which leads to some difficulties. Since our goal is to build a FORTH like system on top of a host language it seems more useful to coexist whith this language while still within the rubrics of FORTH.

Application

```
ok> "Hello World" <ENTER>
4298101456 ok>
```

What we see here (4298101456) is the memory address where the string "Hello World" has been stored. Since we have no type information on our data stack we only can display the number which represents an address.

The simplest way to identify strings is to have a single string area. To know if the value possibly can be a string we can check the value about within the string area.

Next we put the content on screen

```
ok> "Hello World" type cr
Hello World
ok>
```

Source code

Let us compare the old and new version of the function word(). If we found a " character as first non space character we are not scanning to the first space character. We are scanning for the " character.

```
static char *word(void) { // symbol might have maximal 256 bytes
        static char buffer[256], *end=buffer+sizeof(buffer)-1;
        char *p=buffer, ch;
        if(!(ch=skip_space())) return 0; // no more input
        *p++=ch;
        if(ch=='"') { // +course02: string handling
                while(p<end && (ch=next_char()) && ch!='"') *p++=ch;
        } else {
                while(p<end && (ch=next_char()) && !isspace(ch)) *p++=ch;
        }
        *p=0; // zero terminated string
        return buffer;
}
```

And here we see the changes to out interpret() function

```
static void interpret(char *w) {
        if(*w=='"') { // +course02: string handling
                sp_push((cell_t)to_pad(w+1)); // store the string
        } else if((current_xt=find(w))) {
                current_xt->prim();
        } else { // not found, may be a number
                char *end;
                int number=strtol(w, &end, 0);
                if(*end) terminate("word not found");
                else sp_push(number);
        }
}
```

```
static char *to_pad(char *str) { // add for course02: copy str into scratch pad
        static char scratch[1024];
        int len=strlen(str);
        if(len>sizeof(scratch)-1) len=sizeof(scratch)-1;
        memcpy(scratch, str, len);
        scratch[len]=0; // zero byte at string end
        return scratch;
}
```

And now lets see how to implement the compiler course03 ⇒