Memoria en Forth/r3

r3 es un lenguaje de programación derivado del ColorForth.
https://github.com/phreda4/r3

Pablo H. Reda

2024

Una diferencia con otros lenguajes de programación es el manejo de memoria.
Se incentiva la construcción de un mapa de memoria real, o sea, que hay una descripción detallada de la forma de la memoria en la ejecución de un código.

Memoria estática o definición de variables

Cada definición de variable asigna celdas de memoria con los nombres indicados.

#var1 1234

la variable var1 es el nombre de la dirección de memoria donde se encuentra el número 1234 en 8 bytes de memoria (64bits)

puedo definir memoria en 32bits y en 8

#var32 [1234]

#var8 (123)

las variables definen el mapa de memoria, en el orden que se definen
Además puedo poner varias celdas de memoria indicando números en forma de lista.

#lista 1 2 3 4

con cualquiera de los modificadores de tamaño.

#listavar 1 2 [3 4] (5 6 7) 8

Cuando se está definiendo memoria las referencias a las palabras siempre son las direcciones, por lo que las dos definiciones siguientes son iguales

#vector1 ‘suma
#vector2 suma

Por último, se puede definir un fragmento de memoria con el operador * (por)

#buffer * $ffff | 64 kb
#pad * 1024 | 1 kb

las variables indican los comienzos de estos fragmentos de memoria

Memoria dinámica

La única palabra que provee el lenguaje para el acceso a la memoria libre es

MEM | – direccion_de_inicio_de_la_memoria_libre

por supuesto que se puede llamar al SO mediante librerías a las funciones de pedido o liberación de memoria, pero toda la distribución actual no usa este mecanismo.

El inicio de la memoria libre está a continuación de la definición de las variables.

A partir de esta palabra se genera un mecanismo que permite administrar la memoria libre. Este mecanismo son tres palabra que tiene un significado distinto que otros FORTH, y son HERE, MARK y EMPTY

HERE es un variable que contiene la dirección de memoria libre que se incrementa a medida que se usa

MARK guarda en una pila HERE, o sea, que marca donde se encuentra la memoria libre actual

EMPTY desapila HERE, por lo que funciona como una liberación de la memoria usada, ojo, es un pila, no necesita marcas para indicar si está usada o no.

Se deduce de este mecanismo que cuando definimos memoria podemos tener muchos fragmentos de tamaño fijo y uno solo (el último) de tamaño variable.
Esto obliga a tener organizada la forma en que utilizamos la memoria, tiene la ventaja que no es necesario un recolector de basura (garbage collector).

Este esquema de memoria obliga a ser más cuidadoso con la forma de asignación de la memoria y este cuidado mejora al sistema.

Los sistemas que necesitan memoria a demanda generalmente son programas de creación, donde la memoria que varía de tamaño puede ponerse al final de esta pila de direcciones y cuando se necesita reorganizar más profundo se rearma esta pila de fragmentos.

Puede darse el caso que una parte de esta memoria deje de usarse y aun así, tomar la decisión de mantener esta memoria, por ejemplo, cargar un texto, tokenizar el texto y usar esta tokenización. El texto se deja de usar directamente pero puede dejarse en la memoria.

String o cadena de caracteres

Los string son secuencia de bytes, se definen encerrandolos entre doble comillas.
para representar la comilla se indica mediante dos doble comillas.
El fin de línea se incluye dentro del string por lo que es posible definir cadenas con multilineas
Existen dos lugares donde se pueden definir strings:

cuando está definiendo código

:hola “hola mundo” print ;

Aca se graba en una memoria de constantes de string, separado de las variables y lo que hace es apilar esta dirección de memoria.

cuando se define datos:

#palabras “uno”

En este caso el string deja en memoria los bytes correspondientes y agrega un byte 0 al final de la cadena. La memoria donde se graba esto es en el mismo lugar donde están las variables.

En el caso que se quiera hacer una lista de las direcciones de string utilizando variables, es necesario definir la variable y luego utilizar esta dirección

#a “uno”
#b “dos”
#c “tres”

#lista ‘a ‘b ‘c

ya que esta definición genera otra estructura

#lista “uno” “dos” “tres”