Icono

Descripción generada automáticamenteUn dibujo de una persona

Descripción generada automáticamente con confianza bajaINSTITUTO POLITÉCNICO NACIONALESCUELA SUPERIOR DE CÓMPUTO

**PRACTICA 6**

“CICLO FOR”

ALUMNO:

LOPEZ PEREZ ALBERTO ANDREI

PROFESOR:

TECLA PARRA ROBERTO

GRUPO:

5CM4

MATERIA:

COMPILADORES

## Introducción

En esta práctica se implementaron los conceptos base de compiladores y la implementación de YACC, donde se da continuidad a la práctica número 5, la cual fue una una "Calculadora de Vectores con Decisiones y Ciclos". Lo que nos atañe realizar en esta práctica es:

Agregar un ciclo ( for ) similar al del lenguaje C

Gracias a que se realizo con anterioridad la implementación de un ciclo while, lo más lógico sería intenta llegar a una implementación similar para crear un ciclo **for**. Sin embargo, es necesario primero entender cómo es que funciona el ciclo **for.**

El ciclo **for** en C cuenta con lo siguiente:

for ( sentencia1 ; condición ; sentencia2 ) {

//Cuerpo del ciclo for

}

Donde es importante remarcar qué:

Se ponen entre paréntesis las 3 sentencias de la cabeza del for, separadas por ; .

La **sentencia1** es cualquier fragmento de código (normalmente una inicialización de una variable) que se ejecuta una sola vez, siendo esta lanzada antes de iniciarse el ciclo.

La **condición** es cualquier operación lógica o valor de verdadero/falso, la cual se va a evaluar cada vez que se requiera iniciar un ciclo (en otras palabras, antes de entrar al **for** y cada vez que se finalice el cuerpo del ciclo).

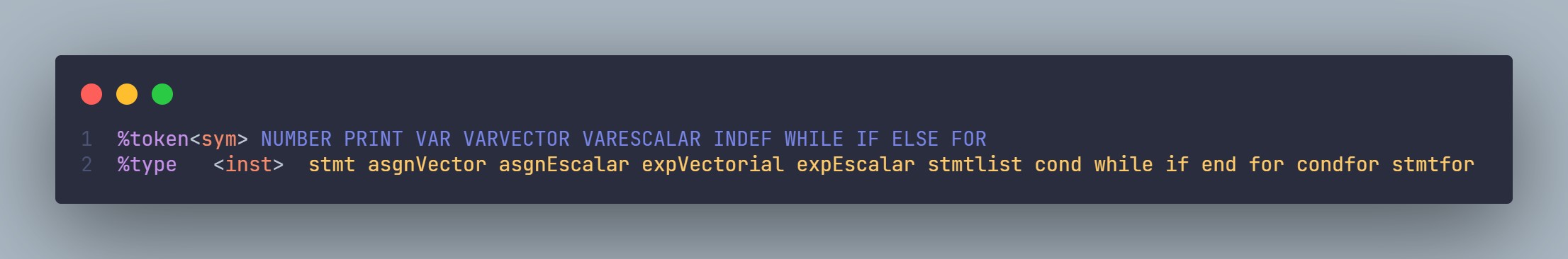
La **sentencia2** es cualquier fragmento de código (normalmente una redefinición de una variable) que se ejecuta cada vez que se finaliza de ejecutar el cuerpo de un ciclo.

El cuerpo del ciclo **for** puede ser tan extenso como se requiera.

Con lo anterior nos podemos dar una idea de cómo podemos llevar este ciclo a nuestra "Calculadora de Vectores" y para poder realizarlo con éxito debemos de plantear muy bien sus producciones y su mapa de memoria para poder usarlo correctamente con nuestra *máquina virtual de pila*.

## Desarrollo

Para empezar, lo primero que definimos fueron nuestros nuevos terminales y no terminales, que en este caso solo son el terminal **FOR** y los no terminales **for**, **condfor** y **stmtfor**.



Nuevos terminales y no terminales en las Definiciones de YACC

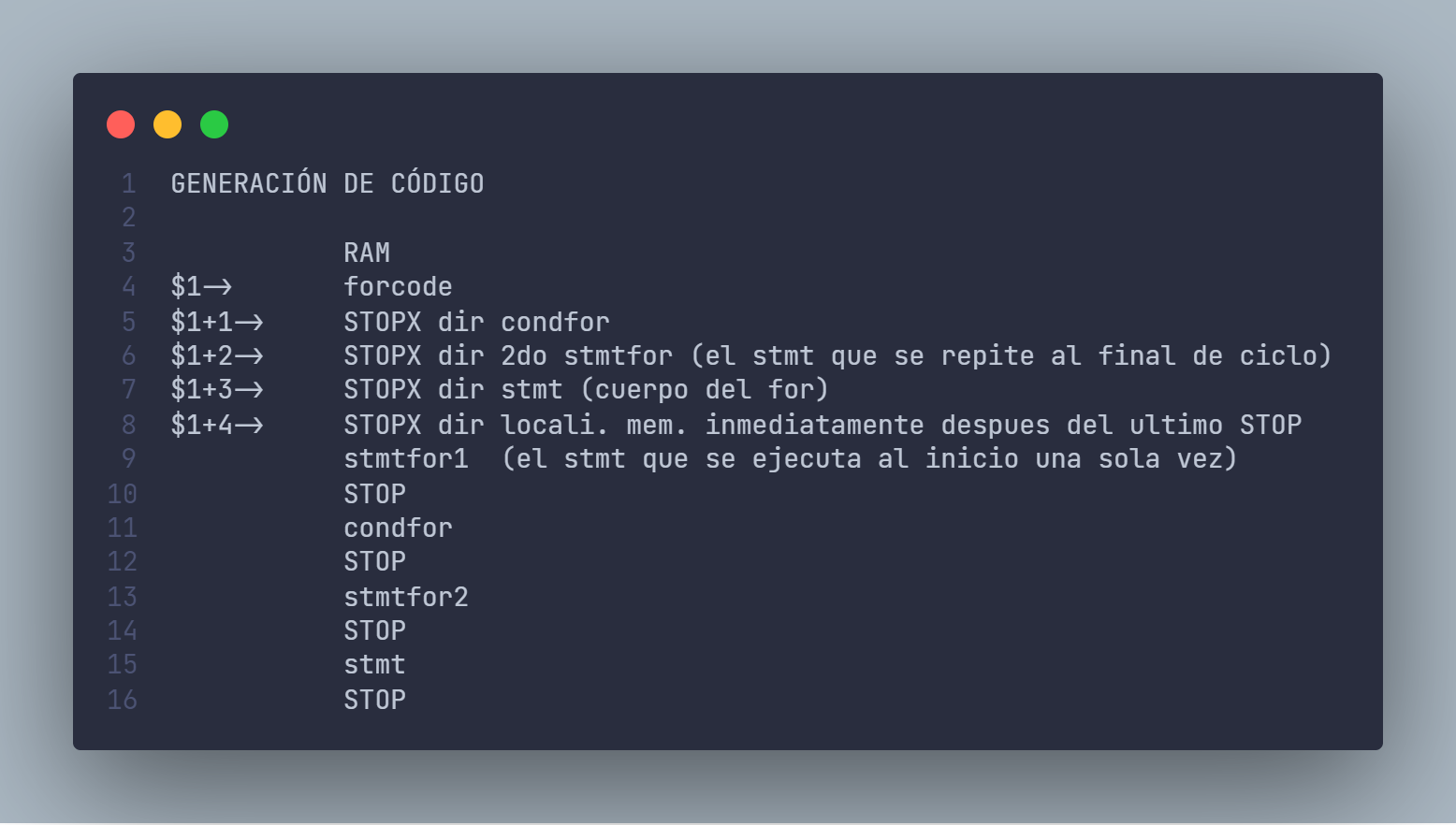
Luego, en las reglas de YACC, agregamos una nueva producción en **stmt** para generar la sintaxis del ciclo **for** y agregamos también las producciones para **condfor** y **stmtfor.**



Nuevas reglas de YACC para generar ciclo for

Como podemos observar, las sentencias **stmtfor** por cuestiones de funcionamiento de la calculadora de vectores las estamos acotando a que sean asignaciones e impresiones de cualquier tipo y la **condfor** esperamos que sea cualquier valor escalar (de preferencia el valor de cualquier operación lógica y de comparación).

Con las anteriores producciones vemos que respetamos la sintaxis del **for** en lenguaje C, ahora solo falta respetar su funcionalidad, la cual fue añadida en la instrucción **forcode()** implementada en **code.c**.Sin embargo, para antes de llegar a esa parte, lo primero que se requirió fue trazar el mapa de memoria que nos ayudaría a saber como vamos a apartar los espacios para generar la funcionalidad completa del programa, el cual se presenta a continuación:

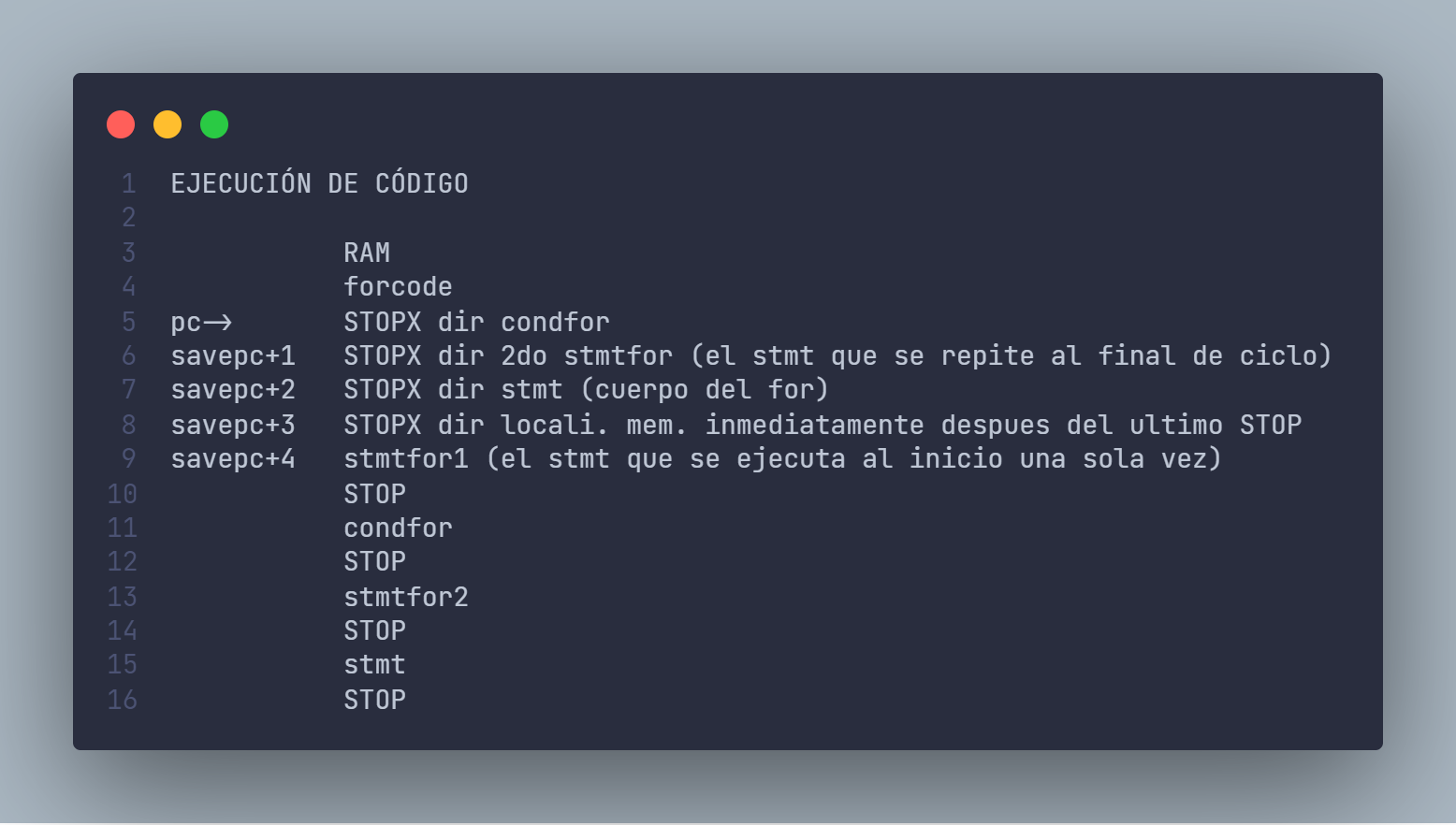


Mapa de Memoria de la Generación de Código

Donde podemos ver que usamos 4 **STOP** para apartar los lugares de la dirección de la *condicional* del for, del *statement* que se repite, del inicio del *cuerpo* del for y de la localidad de memoria inmediatamente después del último **STOP** (ósea al terminarse su ejecución). Lo que sigue en la RAM después de los stops es la inserción en orden de lo que sería el primer **stmtfor**, la **condfor**, el segundo **stmtfor** y el cuerpo **stmt**.

Con lo anterior vemos que ahora tienen más sentido las acciones gramaticales ingresadas en las producciones que creamos anteriormente.

Ahora sí, a partir del mapa de memoria observamos como sería la ejecución del código en la RAM



Mapa de memoria de la ejecución de código.

Donde con base en lo anterior, implementamos la funcionalidad del ciclo **for** en **forcode()** tomando en cuenta el cómo funciona el ciclo **for** en C.



Función forcode() para generar un ciclo for en **code.c**

Y con lo anterior, generamos perfectamente el ciclo for.

ℹ

Es importante agregar en las

**keywords**

del archivo

**init.c**

la palabra

"for"

ligada con el

token

**FOR**

.

### Compilación y Ejecución

Para poder compilar y ejecutar:

Compilación de YACC: "yacc -d calcVectores.y"

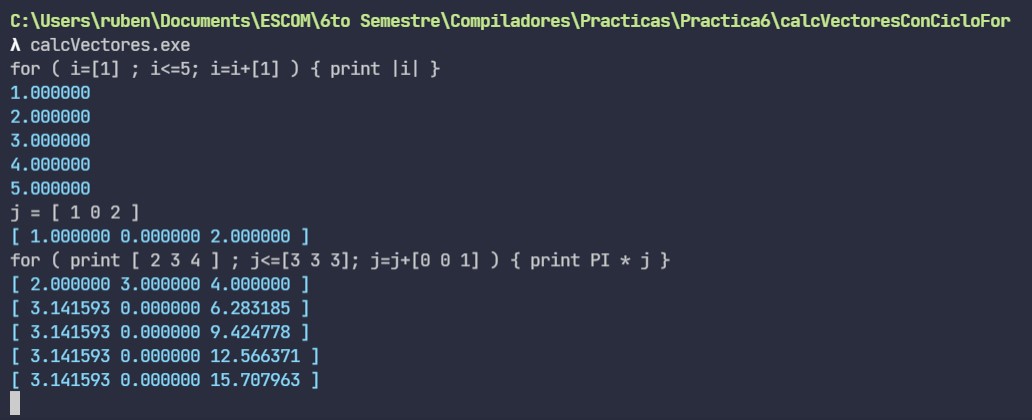
Compilación en programa C (WINDOWS): "gcc y.tab.c vector.c code.c symbol.c init.c -o calcVectores.exe -lm"

Compilación en programa C (LINUX): "gcc y.tab.c vector.c code.c symbol.c init.c -o calcVectores.out -lm"

Ejecución (WINDOWS): "calcVectores.exe"

Ejecución (LINUX): "./calcVectores.out"

### Programa en Ejecución



Funcionamiento de ciclo for probando una cuenta del 1 al 5 y probando sentencias en cabeza de for para generar una funcionalidad similar a la del for en C.

## Conclusión

Esta práctica tal vez se sienta corta al agregar solo una nueva sentencia a nuestros *statements*, sin embargo, considero que puede volverse una práctica de mayor complejidad si aún no se comprende del todo los conceptos de la *máquina virtual de pila* y su funcionamiento en cuanto a las dos etapas que integra (la generación de código y la ejecución de código) ya que, como en esta ocasión esta sentencia no la propone el profesor, requiere que se sepa bien como utilizar la memoria RAM a favor de la sentencia nueva que vamos a agregar, sobre todo en el *"¿Cuántos STOP nos serán de ayuda para apartar lugar?"*, y el cómo mandaremos a ejecutar cada una de esas partes ya en la función **forcode()**. Si no se entienden bien los mapas de memoria, puede que haya problemas, pero sin duda, con un tanto de estudio en los anteriores statements y con conocimiento sobre otros lenguajes es posible librar bien esta práctica.