



Compilador del lenguaje C-

Iván Bruno Muñoz Yhmoff
A01366756

Tabla de contenido

Introducción	2
Manual de Usuario	2
Apéndice A: Expresiones Regulares	8
Apéndice B: Forma EBNF de la GLC del lenguaje de programación C-	9
Apéndice C: Analizador Semántico.....	10
Apéndice D: Definición del lenguaje	12

Introducción

El código generado a través del compilador para el lenguaje C- es del lenguaje ensamblador para el procesador MIPS, sin embargo, se utilizó el simulador SPIM en su versión más reciente QtSpim, para la implementación del compilador.

Fue seleccionada este tipo de procesador porque usa una arquitectura RISC (Reduced instruction set computing) y este tipo de arquitectura es común en los procesadores actuales; Por otro lado, se escogió este tipo de procesador por la disponibilidad de éste debido al simulador antes mencionado.

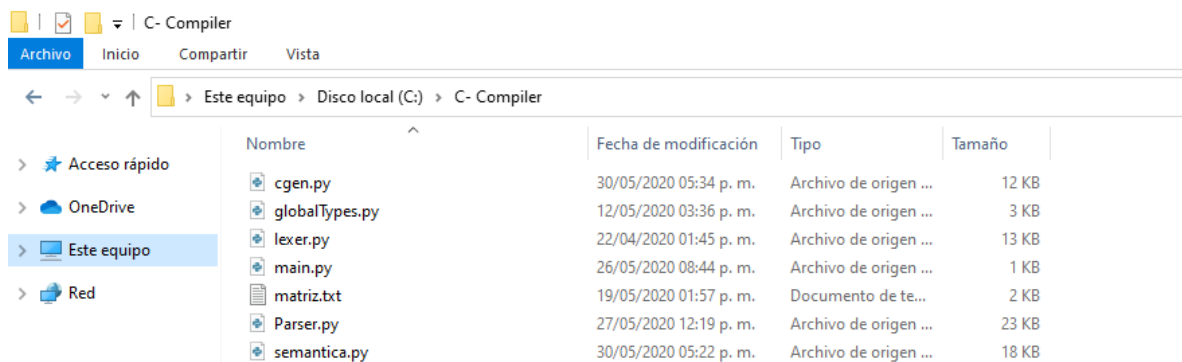
Manual de Usuario

Requisitos:

- Tener Python instalado en su computadora y listo para funcionar.
- Tener descargado y funcionando el simulador SPIM , de preferencia su última versión: QtSpim.

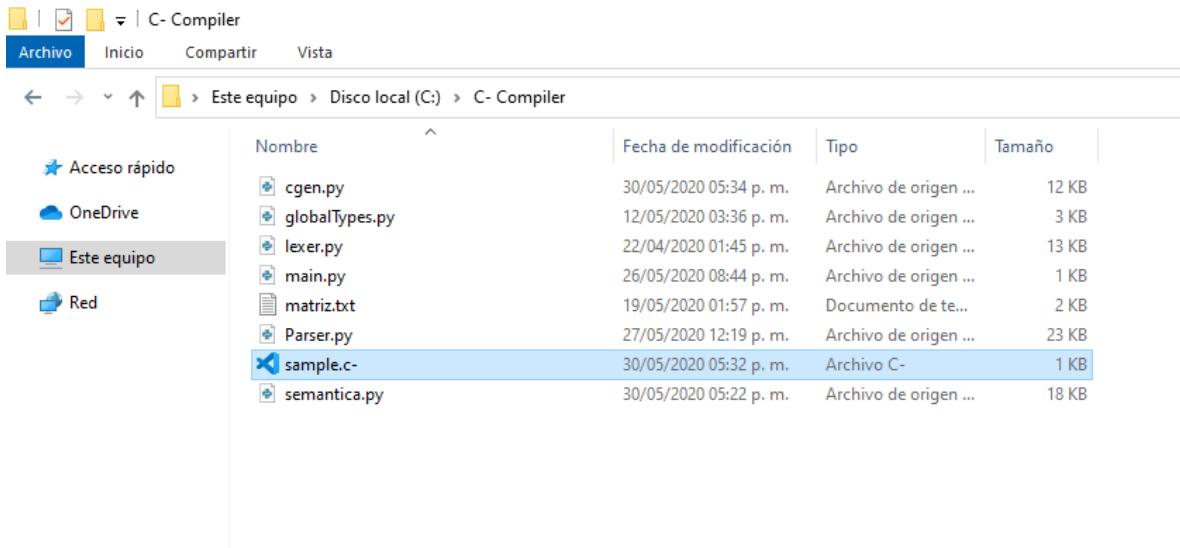
Este es el procedimiento para compilar un programa escrito en el lenguaje C-:

1. Obtener el paquete de archivos que componen el compilador:
 - a. matriz.txt
 - b. globalTypes.py
 - c. lexer.py
 - d. Parser.py
 - e. semántica.py
 - f. cgen.py
 - g. main.py

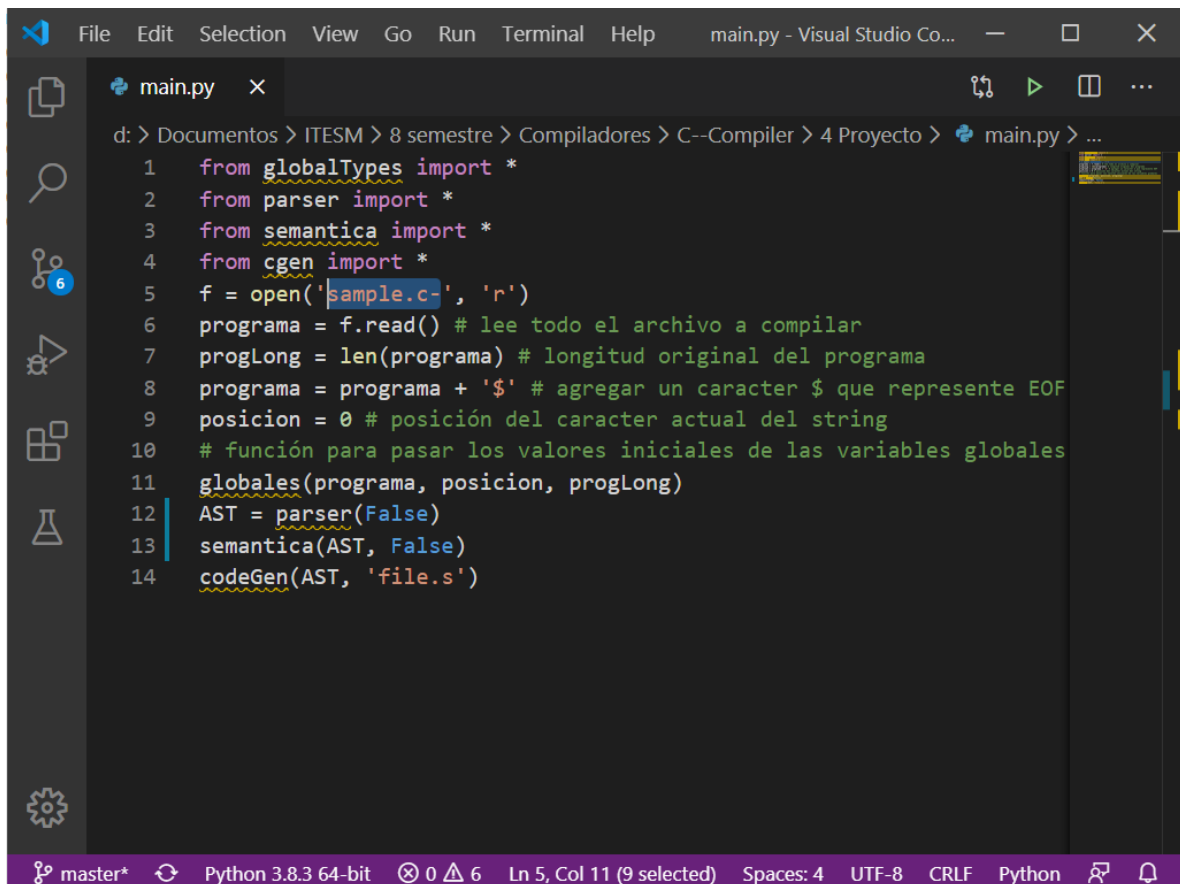


Nombre	Fecha de modificación	Tipo	Tamaño
cgen.py	30/05/2020 05:34 p. m.	Archivo de origen ...	12 KB
globalTypes.py	12/05/2020 03:36 p. m.	Archivo de origen ...	3 KB
lexer.py	22/04/2020 01:45 p. m.	Archivo de origen ...	13 KB
main.py	26/05/2020 08:44 p. m.	Archivo de origen ...	1 KB
matriz.txt	19/05/2020 01:57 p. m.	Documento de te...	2 KB
Parser.py	27/05/2020 12:19 p. m.	Archivo de origen ...	23 KB
semantica.py	30/05/2020 05:22 p. m.	Archivo de origen ...	18 KB

2. Debe tener en la misma carpeta el programa que quiere compilar (en este caso se llama "sample.c-"):



3. En caso de que su archivo tenga otro nombre debe cambiarlo por "sample.c-" o en su defecto entrar al archivo llamado "main.py" y editar la línea 5, poniendo el nombre de su archivo donde dice "sample.c-":



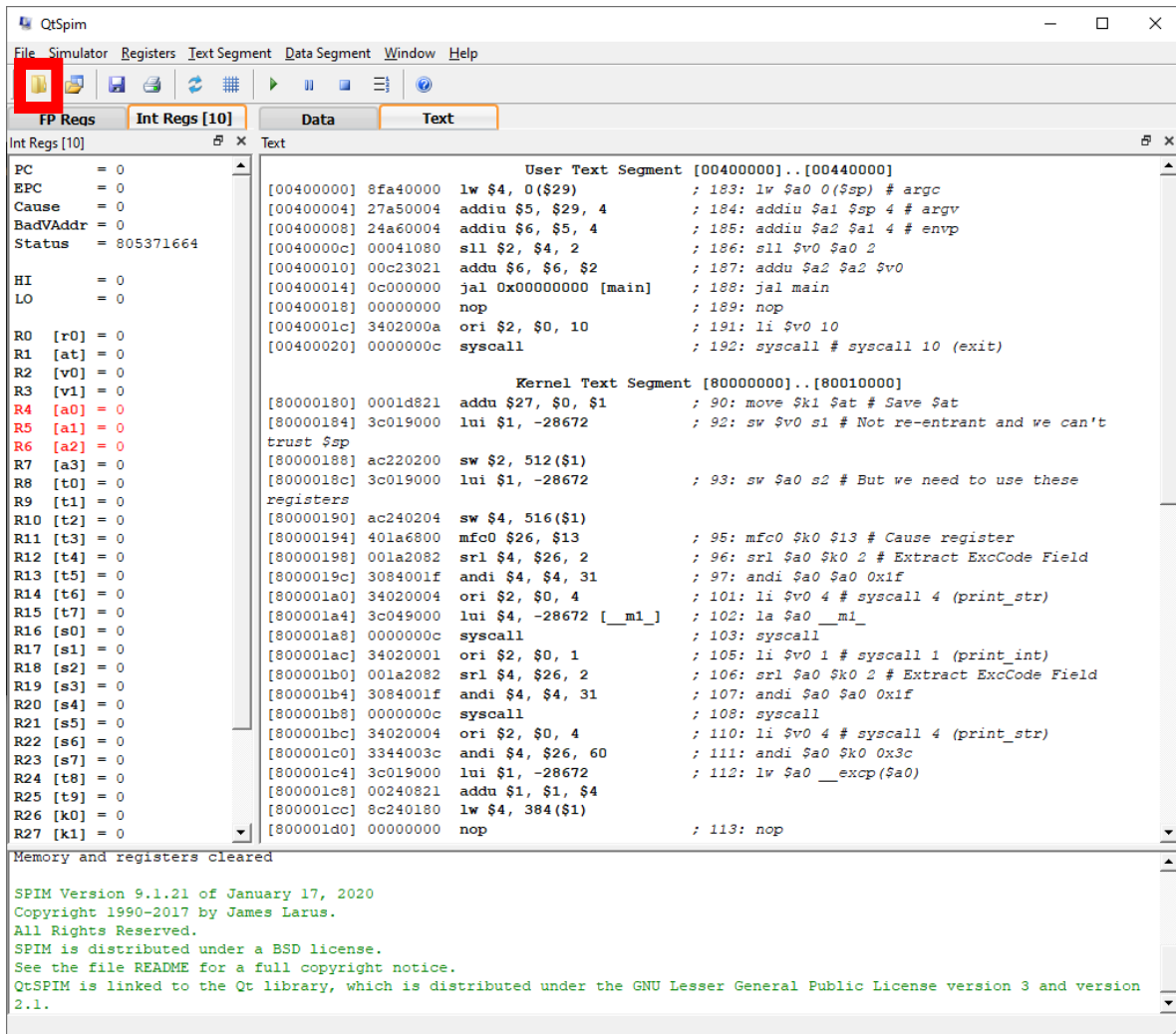
4. Ahora debe ejecutar el programa “main.py” entrando desde una terminal de comandos, abierta en la carpeta donde se ubican todos los archivos del compilador y escribir el siguiente comando:
 - a. `python main.py`



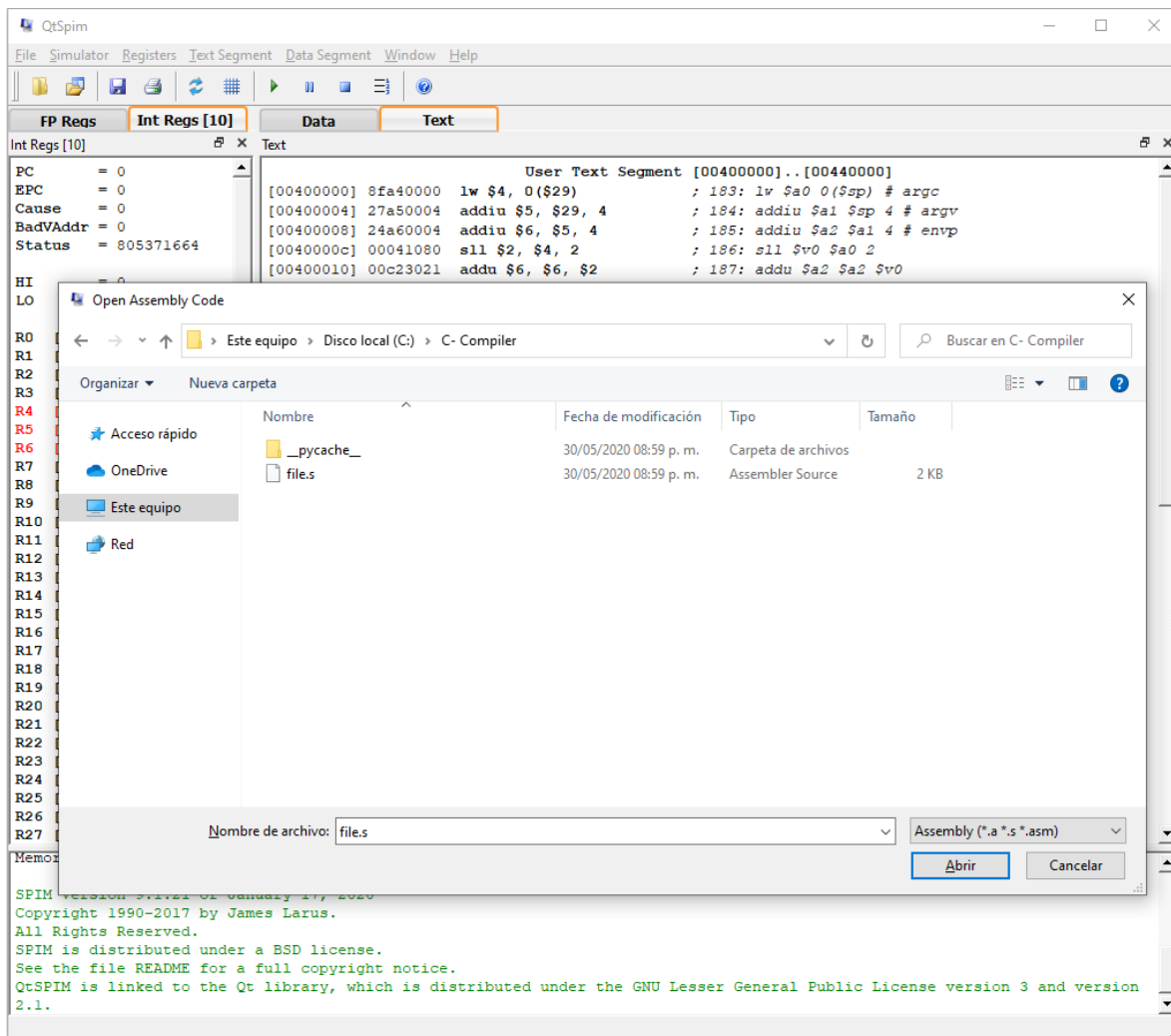
```
Símbolo del sistema
C:\C- Compiler>python main.py
C:\C- Compiler>
```

- b. En caso de que su programa tenga algún error se le notificará a través de la misma terminal.

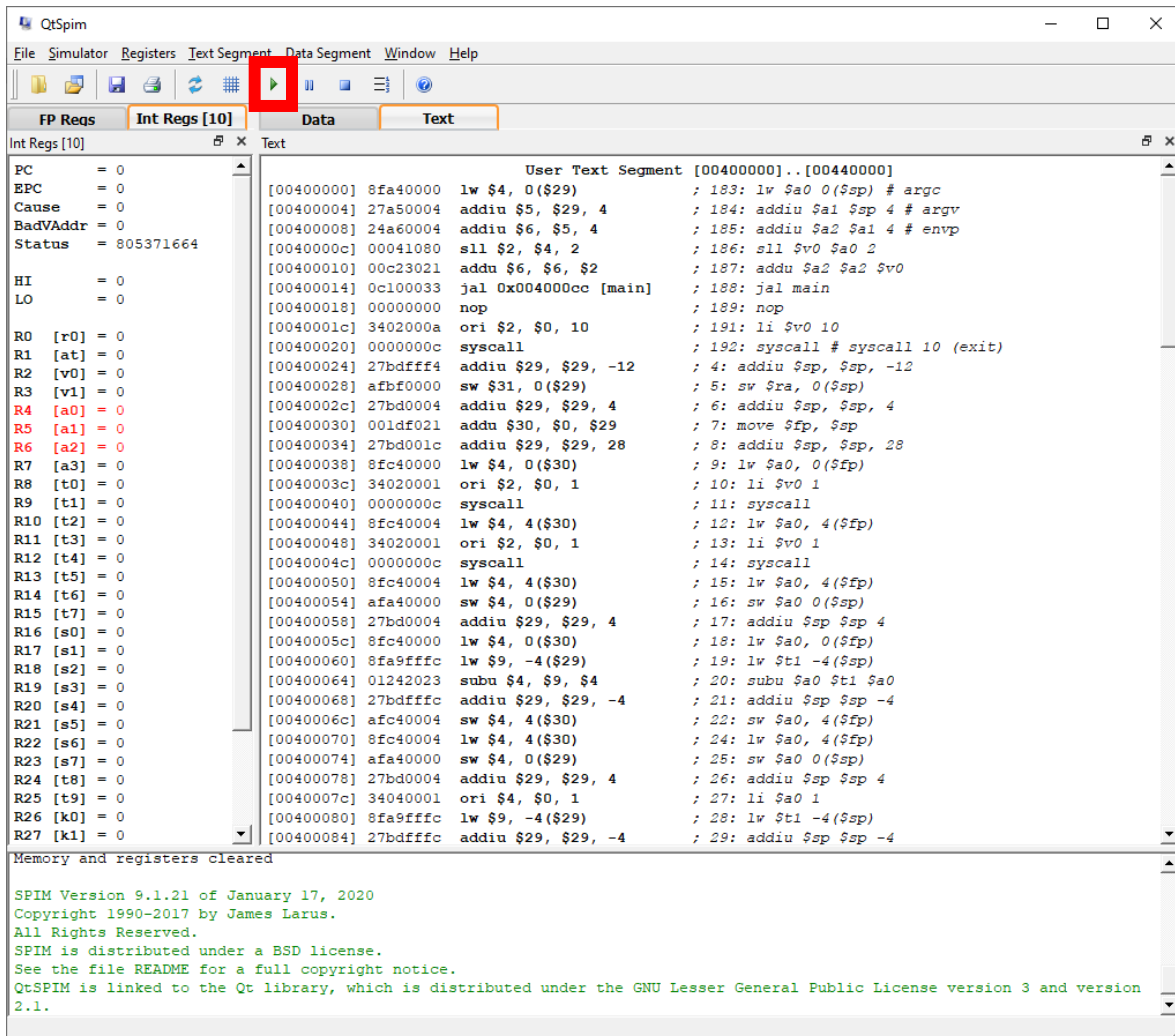
5. En la misma carpeta se habrá generado un archivo nuevo con el nombre de “file.s” y lo deberá abrir desde su simulador QtSpim:
 - a. Dando click en el botón “Load File”:



b. Seleccionar el archivo “file.s” que se generó dentro de su carpeta:



c. Dar click en el botón de “play”:



QtSPIM

File Simulator Registers Text Segment Data Segment Window Help

FP Regs Int Regs [10] Data Text

Int Regs [10]

```
PC = 0
EPC = 0
Cause = 0
BadVAddr = 0
Status = 805371664

HI = 0
LO = 0

R0 [r0] = 0
R1 [at] = 0
R2 [v0] = 0
R3 [v1] = 0
R4 [a0] = 0
R5 [a1] = 0
R6 [a2] = 0
R7 [a3] = 0
R8 [t0] = 0
R9 [t1] = 0
R10 [t2] = 0
R11 [t3] = 0
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
```

User Text Segment [00400000]..[00440000]

```
[00400000] 8fa40000 lw $4, 0($29) ; 183: lw $a0 0($sp) # argc
[00400004] 27a50004 addiu $5, $29, 4 ; 184: addiu $a1 $sp 4 # argv
[00400008] 24a60004 addiu $6, $5, 4 ; 185: addiu $a2 $a1 4 # envp
[0040000c] 00041080 sll $2, $4, 2 ; 186: sll $v0 $a0 2
[00400010] 00c23021 addu $6, $6, $2 ; 187: addu $a2 $a2 $v0
[00400014] 0c100033 jal 0x004000cc [main] ; 188: jal main
[00400018] 00000000 nop ; 189: nop
[0040001c] 3402000a ori $2, $0, 10 ; 191: li $v0 10
[00400020] 0000000c syscall ; 192: syscall # syscall 10 (exit)
[00400024] 27bdffff addiu $29, $29, -12 ; 4: addiu $sp, $sp, -12
[00400028] afbf0000 sw $31, 0($29) ; 5: sw $ra, 0($sp)
[0040002c] 27bd0004 addiu $29, $29, 4 ; 6: addiu $sp, $sp, 4
[00400030] 001df021 addu $30, $0, $29 ; 7: move $fp, $sp
[00400034] 27bd001c addiu $29, $29, 28 ; 8: addiu $sp, $sp, 28
[00400038] 8fc40000 lw $4, 0($30) ; 9: lw $a0, 0($fp)
[0040003c] 34020001 ori $2, $0, 1 ; 10: li $v0 1
[00400040] 0000000c syscall ; 11: syscall
[00400044] 8fc40004 lw $4, 4($30) ; 12: lw $a0, 4($fp)
[00400048] 34020001 ori $2, $0, 1 ; 13: li $v0 1
[0040004c] 0000000c syscall ; 14: syscall
[00400050] 8fc40004 lw $4, 4($30) ; 15: lw $a0, 4($fp)
[00400054] afa40000 sw $4, 0($29) ; 16: sw $a0 0($sp)
[00400058] 27bd0004 addiu $29, $29, 4 ; 17: addiu $sp $sp 4
[0040005c] 8fc40000 lw $4, 0($30) ; 18: lw $a0, 0($fp)
[00400060] 8fa9ffff lw $9, -4($29) ; 19: lw $t1 -4($sp)
[00400064] 01242023 subu $4, $9, $4 ; 20: subu $a0 $t1 $a0
[00400068] 27bdffff addiu $29, $29, -4 ; 21: addiu $sp $sp -4
[0040006c] afc40004 sw $4, 4($30) ; 22: sw $a0, 4($fp)
[00400070] 8fc40004 lw $4, 4($30) ; 24: lw $a0, 4($fp)
[00400074] afa40000 sw $4, 0($29) ; 25: sw $a0 0($sp)
[00400078] 27bd0004 addiu $29, $29, 4 ; 26: addiu $sp $sp 4
[0040007c] 34040001 ori $4, $0, 1 ; 27: li $a0 1
[00400080] 8fa9ffff lw $9, -4($29) ; 28: lw $t1 -4($sp)
[00400084] 27bdffff addiu $29, $29, -4 ; 29: addiu $sp $sp -4
```

Memory and registers cleared

SPIM Version 9.1.21 of January 17, 2020
Copyright 1990-2017 by James Larus.
All Rights Reserved.
SPIM is distributed under a BSD license.
See the file README for a full copyright notice.
QtSPIM is linked to the Qt library, which is distributed under the GNU Lesser General Public License version 3 and version 2.1.

Al completar estos pasos habrá compilado su programa escrito en el lenguaje C- y podrá ver sus resultados en la consola del simulador.

Apéndice A: Expresiones Regulares

LETRA = A|...|Z|a|...|z

DIGITO = 0|...|9

ID = LETRA LETRA*

NUM = DIGITO DIGITO*

COMENTARIO = /* (LETRA|DIGITO|^(*/))* */

IF = if

ELSE = else

INT = int

RETURN = return

VOID = void

WHILE = while

PLUS = +

LESS = -

MULT = *

DIV = /

LT = <

MT = >

LTEQUAL = <=

MTEQUAL = >=

EQUAL ==

DIFF = !=

ASSIGN = =

SEMICOL = ;

COMMA = ,

OPENB = (

CLOSEB =)

OPENSB = [

CLOSESB =]

OPENCB = {

CLOSECB = }

Apéndice B: Forma EBNF de la GLC del lenguaje de programación C-

1. program -> declaration {declaration}
2. declaration -> var-declaration | fun-declaration
3. var-declaration -> type-specifier ID ["["NUM "]" ";"
4. type-specifier -> int | void
5. fun-declaration -> type-specifier ID "("params")" compound-stmt
6. params -> param {"," param}
7. param -> type-specifier ID ["["NUM "]"
8. compound-stmt -> "{" local-declarations {statement}"}
9. local-declarations -> {var-declaration}
10. statement -> expression-stmt | compound-stmt | selection-stmt | iteration-stmt | return-stmt
11. expression-stmt -> [expression] ";"
12. selection-stmt -> if "("expression")"statement [else statement]
13. iteration-stmt -> while "("expression")" statement
14. return-stmt -> return [expression];"
15. expression -> var "=" expression | simple-expression
16. var -> ID ["["expression"]"]
17. simple-expression -> additive-expression [relop additive-expression]
18. relop -> <= | < | > | >= | == | !=
19. additive-expression -> term {addop term}
20. addop -> +|-
21. term -> factor {mulop factor}
22. mulop -> *|/
23. factor -> "("expression")" | var | call | NUM
24. call -> ID "("args")"
25. args - [expression {"," expression}]

Apéndice C: Analizador Semántico

Estructura del stack:

El stack tendrá la estructura de una lista, y sólo contendrá los números que representen cada scope de la tabla de símbolos, además en la primera posición siempre contendrá el número 0 que representará la tabla de símbolos global y éste no podrá ser sacado del stack.

Un ejemplo del stack sería este:

- Stack = [0,2,3]

En donde significa que el scope 3 está dentro del scope 2, y a su vez éste está dentro del scope 0.

Estructura de la tabla de símbolos:

La tabla de símbolos correspondiente a cada scope, se encontrará dentro de un diccionario de diccionarios, siendo identificada cada una mediante un valor entero, que además representa el orden en el que fueron creadas.

Cada una de estas tablas tendrá la siguiente estructura:

Nombre del atributo	Tipo de variable	Notas	Ejemplo
Identificador del atributo	Cadena de caracteres	Con este se va a encontrar cada variable o función dentro de la tabla.	"main"
Tipo de variable o función	Cadena de caracteres	Este va a representar el tipo de la variable o función, pudiendo ser "void" o "int".	"int"
Si es arreglo	Booleano	Este atributo será "True" en caso de que la variable que representa sea un arreglo y "False" en cualquier otro caso.	False
Tamaño del arreglo o número de parámetros de la función	Entero	Este atributo tendrá un valor de -1 cuando el identificador en cuestión no sea arreglo ni función. En caso de ser función tendrá el número de parámetros correspondientes y en caso de ser arreglo	0

		tendrá el tamaño del arreglo.	
Scope de la función	Entero	En caso de que el atributo en cuestión sea una función, aquí estará el scope correspondiente a esta. En caso de que sea cualquier otro tipo de atributo, tendrá el valor de -1.	5

Reglas lógicas de inferencia de tipos:

1. Identificadores:
 - a. $\text{TablaDeSimbolos}(\text{Id}) = \text{INTEGER} \rightarrow \vdash \text{Id}: \text{INTEGER}$
2. Arreglos:
 - a. $\text{TablaDeSimbolos}(\text{Id}) = \text{ARRAY} \rightarrow \vdash \text{Id}: \text{ARRAY}$
3. Identificadores:
 - a. Identificador es una literal entera $\rightarrow \vdash \text{Identificador}: \text{INTEGER}$
4. Constantes:
 - a. Constante $\rightarrow \vdash \text{Constante}: \text{INTEGER}$
5. Arreglos:
 - a. Identificador es un arreglo $\rightarrow \vdash \text{Identificador}: \text{ARRAY}$
6. Operadores lógicos:
 - a. String está dentro de los operadores lógicos $\rightarrow \vdash \text{String}: \text{BOOLEAN}$
7. Operadores enteros:
 - a. String está dentro de los operadores $\rightarrow \vdash \text{String}: \text{INTEGER}$
8. Asignación:
 - a. String es de tipo assign $\rightarrow \vdash \text{String}: \text{INTEGER}$
9. Constante dentro de los corchetes de un arreglo:
 - a. $\vdash \text{Identificador}: \text{ARRAY} \wedge \vdash \text{Constante}: \text{INTEGER} \rightarrow \vdash \text{Identificador}[\text{Constante}]: \text{INTEGER}$
10. Variable dentro de los corchetes de un arreglo:
 - a. $\vdash \text{Identificador1}: \text{ARRAY} \wedge \vdash \text{Identificador2}: \text{INTEGER} \rightarrow \vdash \text{Identificador1}[\text{Identificador2}]: \text{INTEGER}$
11. Suma:
 - a. $\vdash \text{Id1}: \text{INTEGER} \wedge \vdash \text{Id2}: \text{INTEGER} \rightarrow \vdash \text{Identificador1} + \text{Identificador2}: \text{INTEGER}$
12. Resta:
 - a. $\vdash \text{Id1}: \text{INTEGER} \wedge \vdash \text{Id2}: \text{INTEGER} \rightarrow \vdash \text{Identificador1} - \text{Identificador2}: \text{INTEGER}$
13. División:
 - a. $\vdash \text{Id1}: \text{INTEGER} \wedge \vdash \text{Id2}: \text{INTEGER} \rightarrow \vdash \text{Identificador1} / \text{Identificador2}: \text{INTEGER}$
14. Multiplicación:

- a. $\vdash \text{Id1: INTEGER} \wedge \vdash \text{Id2: INTEGER} \rightarrow \vdash \text{Identificador1} * \text{Identificador2: INTEGER}$
- 15. Mayor que:
 - a. $\vdash \text{Id1: INTEGER} \wedge \vdash \text{Id2: INTEGER} \rightarrow \vdash \text{Identificador1} > \text{Identificador2: BOOLEAN}$
- 16. Menor que:
 - a. $\vdash \text{Id1: INTEGER} \wedge \vdash \text{Id2: INTEGER} \rightarrow \vdash \text{Identificador1} < \text{Identificador2: BOOLEAN}$
- 17. Menor o igual que:
 - a. $\vdash \text{Id1: INTEGER} \wedge \vdash \text{Id2: INTEGER} \rightarrow \vdash \text{Identificador1} \leq \text{Identificador2: BOOLEAN}$
- 18. Mayor o igual que:
 - a. $\vdash \text{Id1: INTEGER} \wedge \vdash \text{Id2: INTEGER} \rightarrow \vdash \text{Identificador1} \geq \text{Identificador2: BOOLEAN}$
- 19. Igual que:
 - a. $\vdash \text{Id1: INTEGER} \wedge \vdash \text{Id2: INTEGER} \rightarrow \vdash \text{Identificador1} == \text{Identificador2: BOOLEAN}$
- 20. Diferente que:
 - a. $\vdash \text{Id1: INTEGER} \wedge \vdash \text{Id2: INTEGER} \rightarrow \vdash \text{Identificador1} \neq \text{Identificador2: BOOLEAN}$
- 21. Estatuto if:
 - a. $\vdash \text{OP: BOOLEAN} \rightarrow \vdash \text{if(OP):BOOLEAN}$
- 22. Estatuto while:
 - a. $\vdash \text{OP: BOOLEAN} \rightarrow \vdash \text{while(OP):BOOLEAN}$
- 23. Estatuto return:
 - a. $\vdash \text{EXP: INTEGER} \rightarrow \vdash \text{return EXP:INTEGER}$
- 24. Estatuto call:
 - a. $\vdash \text{TablaDeSimbolos(parametro)} = \text{INTEGER} \wedge \vdash \text{TablaDeSimbolos(id)} = \text{INTEGER} \rightarrow \vdash$
- 25. Estatuto call:
 - a. $\vdash \text{TablaDeSimbolos(parametro)} = \text{INTEGER} \wedge \vdash \text{TablaDeSimbolos(id)} = \text{INTEGER} \rightarrow$
 $\vdash \text{Es correcto}$
- 26. Estatuto call:
 - a. $\vdash \text{TablaDeSimbolos(parametro)} = \text{ARRAY} \wedge \vdash \text{TablaDeSimbolos(id)} = \text{ARRAY} \rightarrow \vdash \text{Es}$
 correcto

Apéndice D: Definición del lenguaje

[Lenguaje C-.pdf](#)