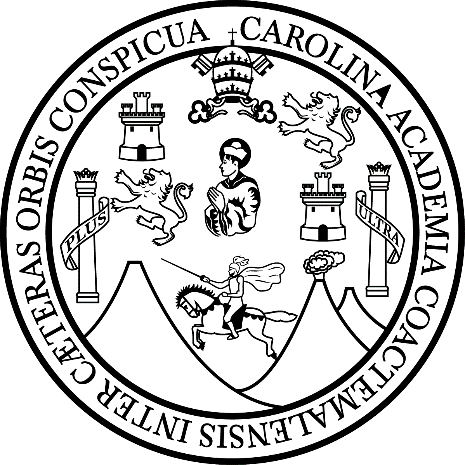
**UNIVERSIDAD DE SAN CARLOS DE GUATEMALA**

Facultad de Ingeniería

Escuela de Ciencias y Sistemas



PROYECTO 2

Manual Tecnico

**Rodrigo Eduardo Carcuz Ortega**

201700633

Guatemala de la Asunción 30 de abril de 2021

Contenido

[Objetivo General: 2](#_Toc69083725)

[Objetivos Específicos: 2](#_Toc69083726)

[RESUMEN : **¡Error! Marcador no definido.**](#_Toc69083727)

[Cargar archivo **¡Error! Marcador no definido.**](#_Toc69083728)

[Ordenar **¡Error! Marcador no definido.**](#_Toc69083729)

[Generar reporte **¡Error! Marcador no definido.**](#_Toc69083730)

# Objetivo General:

● Aplicar los conocimientos adquiridos en el curso sobre el lenguaje ensamblador.

# Objetivos Específicos:

● Aplicar el conocimiento de operaciones básicas a nivel ensamblador.

● Conocer el funcionamiento de las interrupciones.

● Comprender el uso de la memoria en los programas informáticos.

● Consolidar el uso del modo video en lenguaje a bajo nivel.

● Emplear y manipular la paleta de colores del modo video.

● Realizar cálculos para implementar los delays en bajo nivel.

# Gramatica:

|  |  |
| --- | --- |
| **Tokens** | |
| \* | '\*' |
| / | '/' |
| '/' | '-' |
| + | '+' |
| ^ | '^' |
| ( | '(' |
| ) | ')' |
| exec | 'exec |
| { | '{' |
| } | '}' |
| [ | '[' |
| ] | ']' |
| import | 'import' |
| ; | ';' |
| int | int |
| double | 'double' |
| boolean | 'boolean' |
| char | 'char' |
| String | 'String' |
| , | ',' |
| add | 'add' |
| list | 'list' |
| toLower | 'tolower' |
| toUpper | 'toupper' |
| lenght | 'lenght' |
| truncate | 'truncate' |
| round | 'round' |
| typeof | 'typeof' |
| toString | 'tostring' |
| toCharArray | 'tochararray' |
| new | 'new' |
| void | 'void' |
| main | 'main' |
| . | '.' |
| print | 'print' |
| println | 'println' |
| % | '%' |
| : | ':' |
| break | 'break' |
| default | 'default' |
| case | 'case' |
| while | 'while' |
| do | 'do' |
| == | '==' |
| != | '!=' |
| >= | '>=' |
| <= | '<=' |
| && | '&&' |
| || | '||' |
| ! | '!' |
| > | >' |
| < | '<' |
| = | '=' |
| true | 'true' |
| false | 'false' |
| for | 'for' |
| continue | 'continue' |
| return | 'return' |
| switch | 'switch' |
| if | 'if' |
| else | 'else' |
| [\n] | 'SALTO' |
| [0-9]+("."[0-9]+) | 'DECIMAL' |
| [0-9]+ | 'ENTERO' |
| [A-Za-z|"\_"]+[A-Za-z|0-9|"\_"]\*"+""+" | 'AUMETO' |
| [A-Za-z|"\_"]+[A-Za-z|0-9|"\_"]\*"-""-" | 'DECREMENTO' |

# GRAMATICA

Scl:S1

S1: Imprimir

|Func

|Metodo

|LFunc

|Var

|Avar

|DecVec

|ModifVec

|Listas

|Modilis

|Addlist

|ejecuciones

;

ejecuciones : exec Identificador '(' ')' ';'

| exec Identificador '(' Lista\_E ')' ';'

Var: Tipo Lista\_Id Var1

Tipo: int

|double

|boolean

|char

|String

Lista\_Id: Lista\_Id ',' Identificador

|Identificador

Var1:'=' e ';'

|';'

Avar: Identificador '(' ')' ';'

|Identificador '=' e ';'

Func: Tipo Identificador '(' Lista\_Parametro ')' '{' Sent123

|Tipo Identificador '(' ')' '{' Sent123

Sent123: Sent1 return e ';' '}'

|return e ';' '}'

| Sent1 '}'

|'}'

Lista\_Parametro: Var2

Var2: Var2 ',' Tipo Identificador

|Tipo Identificador

Metodo: void Identificador '(' Lista\_Parametro ')' '{' Sent113

|void main '(' ')' '{' Sent113

|void Identificador '(' ')' '{' Sent113

Sent113: Sent1 return ';' '}'

| return ';' '}'

|Sent1'}'

|'}'

LFunc: Identificador '(' Lista\_E ')' ';'

Lista\_E: Lista\_E ',' e{}

|e

Imprimir: print '(' e ')' ';'

IF: if '(' e ')' '{' Sent11

|if '(' e ')' '{' Sent11 else '{' Sent11

|if '(' e ')' '{' Sent11 ELS

|if '(' e ')' '{' Sent11 ELS else'{' Sent11

ELS: ELS else if '(' e ')' '{' Sent11

|else if '(' e ')' '{' Sent11

Sent11: Sent1 '}'

|Sent1 Senten ';' '}'

|Senten';' '}'

|'}'

Senten: break

|return

|return e

|continue

e: e '&&' e

| e '||' e

|'!' e

| '-' e %prec UMINUS

| '\*''-' e %prec UMINUS

| '(' e ')'

| e '==' e

| e '!=' e

| e '<=' e

| e '>=' e

| e '<' e

| e '>' e

| e '^' e

| e '/' e

| e '\*' e

| e '%' e

| e '+' e

| e '-' e

|CADENA

|Identificador '[' e ']'

|CARACTER

| ENTERO

| DECIMAL

|true

|false

|LFunc |Tipo

|tolower '(' e ')'

|toupper '(' e ')'

|lenght '(' e ')'

|truncate '(' e ')'

|round '(' e ')'

|typeof '(' e ')'

|tostring '(' e ')'

|tochararray '(' e ')'

|Identificador

Swit: switch '(' e ')' '{' Cas Def '}'

Cas: Cas case e ':' Sent1

|Cas case e ':'

|Cas case e ':' Sent1 break ';'

|Cas case e ':' break ';'

|case e ':'

|case e ':' Sent1

|case e ':' break ';'

|case e ':' Sent1 break ';'

Def: default ':' Sent1 break ';'

|default ':' break ';'

|default ':' Sent1

|default ':'

Whil: while '(' e ')' '{' Sent111

Sent111: Sent1 Senten ';' '}'

|Sent1 '}'{}

|Senten ';' '}'

|'}'

Do: do '{' Sent11 while '(' e ')' ';'

Fo: for '(' Fo1 ';' e ';' Aum ')' '{' Sent112

Sent112: Sent1 Senten ';' '}'

|Senten ';' '}'

|Sent1 '}'

|'}'

;

Fo1: Tipo Identificador '=' e

| Identificador '=' e

Aum:AUMETO

|DECREMENTO

DecVec: Tipo '[' ']' Identificador '=' new Tipo '[' e ']' ';'

| Tipo '[' ']' Identificador '=' '{' Lista\_E '}' ';'

ModifVec: Identificador '[' ENTERO ']' '=' e ';'

Listas: list '<' Tipo '>' Identificador '=' new list '<' Tipo '>' ';'

| list '<' Tipo '>' Identificador '=' tochararray '(' e ')' ';'

Modilist : Identificador '[' '[' ENTERO ']' ']' = e ';'

Addlist : Identificador '.' '(' e ')' ';'

Sent1: Sent1 IF

|Sent1 Swit

|Sent1 Whil

|Sent1 Do

|Sent1 Var

|Sent1 Imprimir

|Sent1 LFunc

|Sent1 Avar

|Sent1 Fo

|Sent1 DecVec

|Sent1 ModifVec

|Sent1 Listas

|Sent1 Modilist

|Sent1 Addlist

|Sent1 ejecuciones

|IF

|Swit

|Whil

|Do

|LFunc

|Imprimir

|Var

|Avar

|Fo

|DecVec

|ModifVec

|Listas

|Modilist

|Addlist

|ejecuciones

;