Code Specification

Función	Plantillas de Código
Run[[**programa**]]	run[[**programa** → *definiciones*:Definicion*]] = CALL main halt ejecuta[[definiciones]]
metadatos[[**Definicion**]]	metadatos [[**DefVariable** → *nombre*:String *tipo*:Tipo]] =
	metadatos [[**DefFuncion** → *nombre*:String *tipo*:Tipo *defVariableLocal*:Definicion* *sentencias*:Sentencia*]] = {nombre}: ENTER { Σ *defVariableLocal*$_i$.tipo.tam} ejecuta[[sentencias$_i$]] Si tipo == Tipovoid RET 0,{ Σ *defVariableLocal*$_i$.tipo.tam, Σ definicion. *defParametros*.tipo
	metadatos [[**DefStruct** → *nombre*:String *campos*:Definicion* *tipo*:Tipo
	metadatos [[**DefCampoStruct** → *nombre*:String *tipo*:Tipo]] =
ejecuta[[**Sentencia**]]	ejecuta [[**Print** → *expresion*:Expresion]] = valor[[expresion]] OUT<expresión.tipo>
	ejecuta [[**PrintLn** → *expresion*:Expresion]] = valor[[expresion]] OUT<expresión.tipo> PUSHB 10 OUTB
	ejecuta [[**PrintSp** → *expresion*:Expresion]] = valor[[expresion]] OUT<expresión.tipo> PUSH 32 OUTB
	ejecuta [[**Asignacion** → *expIzq*:Expresion *expDrcha*:Expresion]] = direccion[[expIzq]] valor[[expDrcha]] STORE<expIzq.tipo>

	ejecuta [[**If** → *condicion*:Expresion *sentenciasIf*:Sentencia*]] = valor[[condicion]] Jz fin_if ejecuta[[*sentenciasIf*]] fin_if
	ejecuta [[**IfElse** → *condicion*:Expresion *sentenciasIf*:Sentencia* *sentenciasElse*:S valor[[condicion]] Jz else ejecuta[[*sentenciasIf*]] Jmp fin_if else ejecuta[[*sentenciasElse*]] fin_if
	ejecuta [[**While** → *condicion*:Expresion *sentencias*:Sentencia*]] = inicio_while valor[[*condicion*]] jz fin_while ejecuta[[sentencias]] jmp inicio_while fin_while
	ejecuta [[**InvocacionProcedimiento** → *nombre*:String *argumentos*:Exp valor[[argumentos$_{i.}$]] CALL {nombre} Si definicion.tipo.tipoRetorno != Tipovoid POP<definicion.tipo.tipoRetorno>
	ejecuta [[**Return** → *expresion*:Expresion]] = SI(*expresión*!= null) RET { *expresion*.tipo.tam}, {Σ funcion.defvariable$_{i.}$tipo.tam}, {Σ definicion. *defParametros* $_{i.}$tipo.tam} SINO RET 0, {Σ funcion.defvariable$_{i.}$tipo.tam}, {Σ definicion. *defParametros* $_{i.}$tip
	ejecuta [[**Read** → *expresion*:Expresion]] = direccion[[expresion]] IN<expresion.tipo> STORE<expresion.tipo>
valor [[**Expresion**]]	valor [[**ExprAritmetica** → *expIzq*:Expresion *operator*:String *expDrcha*:Expre valor[[expIzq]] valor[[expDrcha]] Si operator == +

	ADD< *expIzq*.tipo> Si operator == - SUB< *expIzq*.tipo> Si operator == * MUL< *expIzq*.tipo> Si operator == / DIV< *expIzq*.tipo>
	valor [[**ExprComparacion** → *expIzq*:Expresion *operator*:String *expDrcha*:Ex valor[[expIzq]] valor[[expDrcha]] Si operator == '<' LT<expIzq.tipo> Si operator == '>' GT<expIzq.tipo> Si operator == '<=' LE<expIzq.tipo> Si operator == '>=' GE<expIzq.tipo> Si operator == '!=' NE<expIzq.tipo> Si operator == '==' EQ<expIzq.tipo>
	valor [[**ExprLogica** → *expIzq*:Expresion *operator*:String *expDrcha*:Expr valor[[expIzq]] valor[[expDrcha]] Si operator == && AND Si operator == || OR
	valor [[**ExprMenosUnario** → *expr*:Expresion]] = Si expr.tipo == TipoInt PUSHI 0 Sino PUSHF 0.0 valor[[expr]] SUB <expr.tipo>
	valor [[**ExprNot** → *expr*:Expresion]] = valor[[expr]] NOT
	valor [[**AccesoArray** → *identificador*:Expresion *posicion*:Expresion]] = dirección[[AccesoArray]] LOAD<AccesoArray.tipo>
	valor [[**ExpresionCast** → *tipoCast*:Tipo *expresion*:Expresion]] = valor[[expresión]] si tipoCast ==TipoChar && *expresion*.tipo == TipoFloat f2i i2b SINO SI tipoCast == TipoFloat && *expresion*.tipo == TipoChar

	b2i i2f SINO <expresión.tipo>2<tipoCast>
	valor [[**AccesoStruct** → *expresion*:Expresion *identificador*:String]] = direccion[[AccesoStruct]] LOAD<AccesoStruct.tipo>
	valor [[**ExpresionInvocacion** → *nombre*:String *argumentos*:Expresion'] valor[[argumentos]] CALL{nombre}
	valor [[**CTE_Real** → *valor*:String]] = PUSHF{valor}
	valor [[**CTE_Entera** → *valor*:String]] = PUSHI{valor}
	valor [[**CTE_Char** → *lexema*:String]] = PUSHB{valor}
	valor [[**Variable** → *nombre*:String]] = dirección[[Variable]] LOAD<Variable.definicion.tipo>
Direccion[[Expresion]]	direccion [[**Variable** → *nombre*:String]] = SI ámbito == 0 PUSHA {Variable.definicion.direccion} SI ámbito != 0 PUSH BP PUSH {Variable.definicion.direccion} ADD
	direccion [[**AccesoStruct** → *expresion*:Expresion *identificador*:String]] Dirección[[expresión]] PUSH {Σ DefStruct.campos$_i$.tipo.tam} ADD
	direccion [[**AccesoArray** → *identificador*:Expresion *posicion*:Expresion dirección[[identificador]] valor[[posición]] PUSH{AccesoArray.tipo.tam} MUL ADD
f_5[[**Tipo**]]	f_5[[**TipoInt** → λ]] =
	f_5[[**TipoStruct** → *identificador*:String]] =
	f_5[[**TipoFloat** → λ]] =

	f_5[[**TipoChar** → λ]] =
	f_5[[**TipoArray** → *size*:Integer *tipoDe*:Tipo]] =
	f_5[[**TipoVoid** → λ]] =
	f_5[[**TipoFuncion** → *tipoRetorno*:Tipo *defParametros*:Definicion*]] =