Code Specification

Función	Plantillas de Código
run[[**Programa**]]	run[[**Programa** → *declaraciones*:Declaracion*]] =
define[[**Declaracion**]]	define[[**DefVariable** → *nombre*:String *tipo*:Tipo]] =
	define[[**DefStruct** → *nombre*:String *listaCampos*:DefCampo*]] =
	define[[**DefFuncion** → *nombre*:String *listaParametros*:DefVariable* *tipo*:Tipo *listaDeclaraciones*:DefVariable* *listaSentencias*:Sentencia*]] = {nombre}: si |listaDeclaraciones| > 0 ENTER {Σ listaDeclaraciones_i.tipo.bytes} si ∉ listaSentencias_n is SenReturn RET {tipo.bytes}, {Σ listaDeclaraciones_i.tipo.bytes}, {Σ listaParametros_i.tipo.bytes}
ejecuta[[**Sentencia**]]	ejecuta[[**SenInvocacion** → *nombre*:String *listaArgumentos*:Expresion*]] = PUSH<listaArgumentos_i.tipo> valor[[listaArgumentos_i]] CALL {nombre} POP<invocacion.tipo>
	ejecuta[[**Asignacion** → *left*:Expresion *right*:Expresion]] = direccion[[left]] valor[[right]] STORE<left.tipo>
	ejecuta[[**SenPrint** → *expresion*:Expresion]] = valor[[expresion]] OUT<expresion.tipo>
	ejecuta[[**SenPrintSp** → *expresion*:Expresion]] = valor[[expresion]]

	OUT<expresion.tipo> PUSHB 32 OUT

	ejecuta[[**SenPrintLn** → *expresion*:Expresion]] = valor[[expresion]]
	si ∈ expresion
	  valor[[expresion]]
	  OUT
	pushb 10
	OUT

	ejecuta[[**SenRead** → *expresion*:Expresion]] = direccion[[expresion]] IN<expresion.tipo> STORE<expresion.tipo>
	ejecuta[[**SenIf** → *condicion*:Expresion *sentenciasIf*:Sentencia* *sentenciasElse*:Sentencia*]] = valor[[condicion]] JZ else{n} ejecuta[[sentenciasIf]] JMP finElse{n} else{n}: ejecuta[[sentenciasElse]] finElse{n}
	ejecuta[[**SenWhile** → *condicion*:Expresion *sentencias*:Sentencia*]] = while{n}: valor[[condicion]] JZ finWhile{n} ejecuta[[sentencias]] JMP wile{n} finWhile{n}:
	ejecuta[[**SenReturn** → *retorno*:Expresion]] = valor[[retorno]] RET bytes[[retorno]], { Σ hisFunction.listaDeclaraciones_i.tipo.bytes}, { Σ hisFunction.listaParametros_i.tipo.bytes}
valor[[**Expresion**]]	valor[[**ExInvocacion** → *nombre*:String *listaArgumentos*:Expresion*]] = PUSH<listaArgumentos_i.tipo> valor[[listaArgumentos_i]] CALL {nombre}

	valor[[**ExAritmetica** → *left*:Expresion *operador*:String *right*:Expresion]] = valor[[left]] valor[[right]] si operador == “+” ADD<tipo> si operador == ‘-‘ SUB<tipo> si operador == “*” MUL<tipo> si operador == ‘/‘ DIV<tipo>
	valor[[**ExLogica** → *left*:Expresion *operador*:String *right*:Expresion]] = valor[[left]] valor[[right]] si operador == “&&” AND si operador == “||” OR
	valor[[**ExRelacional** → *left*:Expresion *operador*:String *right*:Expresion]] = valor[[left]] valor[[right]] si operador == “>” GT<left.tipo> si operador == ‘<‘ LT< left.tipo> si operador == “>=” GE< left.tipo> si operador == “<=” LE< left.tipo> si operador == “==” EQ< left.tipo> si operador == “!=” NE< left.tipo>

	valor[[**ExNot** → *expresion*:Expresion]] = valor[[expresion]] NOT
	valor[[**ExIndice** → *left*:Expresion *indice*:Expresion]] = direccion[[left]] PUSHI {left.tipo.tipobase.bytes} valor[[indice]] MULI ADDI LOAD<tipo>
	direccion[[**ExIndice** → *left*:Expresion *indice*:Expresion]] = direccion[[left]] PUSHI {left.tipo.tipobase.bytes} valor[[indice]] MULI ADDI
	valor[[**ExCampo** → *left*:Expresion *campo*:Expresion]] = direccion[[left]] direccion[[campo]] ADDI LOAD<left.tipo>
	direccion[[**ExCampo** → *left*:Expresion *campo*:Expresion]] = direccion[[left]] direccion[[campo]] ADDI
	valor[[**ExCast** → *to*:Tipo *from*:Expresion]] = valor[[from]] <from.tipo>2<to.tipo>
	valor[[**LitEntero** → *valor*:String]] = PUSHI {valor}
	valor[[**LitReal** → *valor*:String]] = PUSHF {valor}

	valor[[**LitCaracter** → *valor*:String]] = PUSHB {valor}
	valor[[**Variable** → *nombre*:String]] = direccion[[Variable]] LOAD<tipo>
	direccion[[**Variable** → *nombre*:String]] = si declaracion.ambito == GLOBAL PUSHa {declaracion.address} si declaracion.ambito ∈ {LOCAL, CAMPO} PUSHa BP PUSHi {declaracion.address} ADDi
f_6[[**Tipo**]]	f_6[[**TipoEntero** → λ]] =
	f_6[[**TipoReal** → λ]] =
	f_6[[**TipoCaracter** → λ]] =
	f_6[[**TipoStruct** → *nombre*:String]] =
	f_6[[**TipoArray** → *tipo*:Tipo *tamaño*:String]] =
	f_6[[**TipoVoid** → λ]] =