Función	Plantillas de código
execute : programa → Instruccion*	execute[[**programa** → definiciones:definicion*]] = CALL MAIN HALT execute[[definiciones i]]
execute: definicion → Instruccion*	execute[[**defFuncion**:*definicion* → id:*variable* tipo:*tipoFuncion* definiciones:*defVariable** cuerpo:*sentencia** totalLocalVariableSize:*int* parametersSize:*int*]] = {id}: ENTER { totalLocalVariableSize } execute[[cuerpo i]] si tipo.tipoRetorno == Void RET 0,{ totalLocalVariableSize }, { parametersSize }
	execute[[**defVariable**:*definicion* → id:*String* tipo:*Tipo*]] =
execute : sentencia → Instruccion*	execute[[**asignacion:***sentencia* → exp1:*Expresion* exp2:*Expresion*]] = adress[[exp1]] value[[exp2]] cg.convertTo(exp2.tipo,exp1.tipo) <STORE> {exp1.tipo.sufijo}
	execute[[**escritura:***sentencia* → exp:*Expresion**]] = value[[exp i]] <OUT> {exp i.sufijo}
	execute[[**if:***sentencia* → condicion:*Expresion* cuerpo_i*f:Sentencia** cuerpo_else:*Sentencia**]] = int label = cg.getLabels(2) value[[condicion]] cg.convertTo(condicion.tipo, Entero.getInstancia()) <JZ> { "label_" + label } execute[[cuerpo_if i]] <JMP> { "label_" + (label+1) } <{ "label_"+ label + ":" }> execute[[cuerpo_else i]] <{ "label_"+ (label+1) + ":" }>
	execute[[**lectura:***sentencia* → exp:*Expresion**]] = address[[exp i]] cg.in(exp i.tipo.sufijo) cg.store(exp i.tipo.sufijo)
	execute[[**llamadaFuncion:***sentencia* → nombre:*Variable* params:*Expresion**]] = value[[params i]] cg.convertTo(params i.tipo , ((TipoFuncion) llamadaFuncion.definicion.tipo).param(i).tipo) <CALL> { nombre.nombre } si ((TipoFuncion) llamadaFuncion.definicion.tipo).tipoRetorno is Void: <POP> { ((TipoFuncion) llamadaFuncion.definicion.tipo).tipoRetorno.sufijo }
	execute[[**return:***sentencia* → exp:*Expresion*]] = value[[expr]] cg.convertTo(expr.tipo , ((TipoFuncion) definicion.tipo).tipoRetorno) <RET> { definicion.tipo.tipoRetorno.numBytes } , { definicion.localVariablesSize } , { definicion.parametersSize }

	execute[[**while**:*sentencia* → condicion:*Expresion* cuerpo:*Sentencia**]] = int label = cg.getLabels(2) <{ "label_"+ label + ":" }> value[[condicion]] cg.convertTo(condicion.tipo, Entero.getInstancia()) <JZ> { "label_" + (label+1) } execute[[cuerpo i]] <JMP> { "label_" + label } <{ "label_"+ (label+1) + ":" }>
value: expresion → Instruccion*	value[[**variable**:*expresion* → nombre:*String* definicion:*Definicion*]] = address[[variable]] <LOAD> { variable.tipo.sufijo }
	value[[**notUnario**:*expresion* → exp:*Expresion*]] = value[[expr]] <NOT>
	value[[**menosUnario**:*expresion* → exp:*Expresion*]] = cg.push(expr.sufijo , 0) value[[expr]] cg.sub(expr.sufijo)
	value[[**logica**:*expresion* → exp1:*Expresion* exp2:*Expresion* operador:*String*]] = value[[exp1]] cg.convertTo(exp1.tipo , logica.tipo) value[[exp2]] cg.convertTo(exp2.tipo , logica.tipo) cg.logica(operador , logica.tipo)
	value[[**comparacion**:*expresion* → exp1:*Expresion* exp2:*Expresion* operador:*String*]] = Tipo tipoMayor = exp1.tipo.esMayor(exp2.tipo) value[[exp1]] cg.convertTo(exp1.tipo , tipoMayor) value[[exp2]] cg.convertTo(exp2.tipo , tipoMayor) cg.comparacion(operador , tipoMayor)
	value[[**aritmetica**:*expresion* → exp1:*Expresion* exp2:*Expresion* operador:*String*]] = value[[exp1]] cg.convertTo(exp1.tipo , aritmetica.tipo) value[[exp2]] cg.convertTo(exp2.tipo , aritmetica.tipo) cg.logica(operador , aritmetica.tipo)
	value[[**literalReal**:*expresion* → valor:*float*]] = <PUSHF> { valor }
	value[[**literalEntero**:*expresion* → valor:int]] = <PUSHI> { valor }
	value[[**literalChar**:*expresion* → valor:char]] = <PUSHB> { valor }
	value[[**invocacionFuncion**:*expresion* → nombre:*Variable* params:*Expresion**]] = value[[params i]] cg.convertTo(params i.tipo , ((TipoFuncion) llamadaFuncion.definicion.tipo).param(i).tipo) <CALL> { nombre.nombre }
	value[[**cast**:*expresion* → exp:*Expresion* tipoDinamico:*Tipo*]] = value[[exp]] cg.convertTo(exp.tipo , cast.tipo)

	value[[**accesoCampoStruct**:*expresion* → nombre:*Expresion* campo:*String*]] = address[[accesoCampoStruct]] <LOAD> { accesoCampoStruct.tipo.sufijo }
	value[[**accesoArray**:*expresion* → nombre:*Expresion* index:*Expresion*]] = address[[accesoArray]] <LOAD> { accesoArrat.tipo.sufijo }
address : expresion → Instruccion*	address[[**variable**:*expresion* → nombre:*String* definicion:*Definicion*]] = si variable.definicion is DefVariable : cg.pusha((DefVariable) variable.definicion)
	address[[**accesoCampoStruct:***expresion* → nombre:*Expresion* campo:*String*]] = address[[nombre]] cg.push("i" , ((Campo)((TipoStruct)accesoCampoStruct.nombre.tipo).campo(accesoCampoStruct.campo)).offset
	address[[**accesoArray:***expresion* → nombre:*Expresion* index:*Expresion*]] = address[[nombre]] value[[index]] cg.convertTo(index.tipo , Entero.getInstancia()) cg.push("i" , accesoArray.tipo.numeroBytes) cg.mul("i") cg.add("i")