	C /2 C
ejecutar [[Programa: Programa -> Definicion*]]() =	for(Definicion d: Definicion*)
	if(d instanceof DefVariable)
	ejecutar[[d]]();
	for(Definicion d: Definicion*)
	if(d instanceof DefFuncion)
	ejecutar[[d]]();
ejecutar[[DefFuncion: sentencia -> definición* sentencia*]]() =	for(Definicion d : definición*)
	ejecutar[[d]];
	for(Sentencia s : sentecia*)
	ejecutar[[s]];
ejecutar[[Asignacion: sentencia -> expresion1 expresion2]]() =	direccion[[expresion1]]();
ayers a grant and a property and	valor[[expresion2]]();
	GC.convertirA(expresion2.tipo, expresion1.tipo);
	<store> expresion1.tipo.sufijo();</store>
ejecutar [[Escritura: sentencia -> Expresion*]]() =	for(Expresion exp : Expresion*){
ejecutai [[Escritura: seritericia -> Expresion]]() -	valor [[exp]];
	,
	<out> exp.tipo.sufijo;</out>
valauffluoraasiano amusele e e elektrone e e e e e e e e e e e e e e e e e e	for (Companies and Companies #1)
valor[[Invocacion: expresion -> variable Expresion*]]() =	for(Expresion exp : Expresion*){
	contador = 0;
	valor[[expresion]]();
	GC.convertir(expresion.tipo, (TipoFuncion)
	variable.tipo.parametro(contador).tipo);
	contador++;
	}
ejecutar[[Invocacion: sentencia -> variable Expresion*]]() =	valor[[(Expresion) sentencia]]();
	if(!(Expresion)sentencia.tipo instanceof TipoVoid)
	<pop>((Expresion)sentencia).tipo.sufijo;</pop>
ejecutar [[Lectura: sentencia -> Expresion*]]() =	for(Expresion exp : Expresion*){
	Direccion[[exp]]();
	<in> exp.tipo.sufijo;</in>
	<store> exp.tipo.sufijo;</store>
	}
ejecutar[[return: sentencia -> expresion]](df: DefFuncion) =	valor[[expresion]];
ejecutar[[return: sentencia > expresion]](ur. Den uncion) =	GC.convertir(expresión.tipo, df.tipo.tipoRetorno);
	<ret> df.tipo.tipoRetorno.numBytes <,> df.numBytesLocal</ret>
aiocutavIIContonoialfi contonoia > augustica contonoia 4*	<pre><,> df.tipo.paramereos.numBytesParam; int count = CC got! abole(2);</pre>
ejecutar[[Sentencialf: sentencia -> expresion sentencias1*	int count = GC.getLabels(2);
sentencias2*]] =	valor[[expression]]();
	GC.convertirA(expresion.tipo, TipoEntero);
	<pre><jz> <label> count;</label></jz></pre>
	for(Sentencia s: sentencias1*)
	ejecutar[[s]];
	<jmp> <label> count + 1;</label></jmp>
	<label> count <:>;</label>
	for(Sentencia s: sentencias1*)
	ejecutar[[s]];
	<label> count + 1 <:>;</label>
ejecutar[[SentenciaWhile: sentencia -> expresion	GC.getLabels(2);
Sentencia*]]() =	int count = 0;
	<label> count <:>;</label>
	valor[[sentencia]]();
	GC.convertirA(expresion.tipo, TipoEntero);
	<pre></pre>
	for(Sentencia s : Sentencia*)
	ejecutar[[s]];

	<jmp> <label> count;</label></jmp>
	< abel> count + 1 <:>
valor[[Access Array v aversaion1 > aversaion2 aversaion2]]/	
valor[[AccesoArray: expresion1 -> expresion2 expresion3]]() =	direction[[expresion1]];
diagram ((A constant of A comparing)	<pre><load> expresion1.tipo.sufijo;</load></pre>
direccion [[AccesoArray: expresion1 -> expresion2	direction[[expresion2]];
expresion3]]() =	<push> expresion1.tipo.nBytes();</push>
	valor[[expresion3]];
	GC.convertirA(expresion3.tipo, TipoEntero);
	<mul></mul>
	<add></add>
valor[[AccesoCampo: expresion1 -> expresion2 ID]]() =	direccion[[expresion1]]();
	<load> expresion1.tipo.sufijo;</load>
direccion[[AccesoCampo: expresion1 -> expresion2 ID]]() =	direccion[[expresion2]];
	<push> expresion1.tipo.Campo(ID).offset;</push>
	<add></add>
Valor[[Aritmetica: expresion1 -> expresion2 expresion3]]() =	valor[[expresion2]];
	GC.convertirA(expresion2.tipo, expresion1.tipo);
	valor[[expresion3]];
	GC.convertirA(expresion3.tipo, expresion1.tipo);
	GC.Aritmetrica(expresion1.operador, expresion1.tipo);
Valor[[Cast: expresion1 -> tipo expresion2]]() =	valor[[expresion2]];
	GC.convertirA(expresion2.tipo, expresion1.tipo);
Valor[[Compracacion: expresion1 -> expresion2 expresion3]]()	Tipo mayor = expresion2.tipo.mayor(expresion3.tipo);
=	valor[[expresion2]];
	GC.convertirA(expresion2.tipo, mayor);
	valor[[expresion3]];
	GC.convertirA(expresion3.tipo, mayor);
	GC.Aritmetrica(expresion1.operador, expresion1.tipo);
valor[[LiteralCaracter: expresion -> Cte_Caracter]]() =	<push> expresión.valor.sufijo;</push>
valor[[LiteralEntero: expresion -> Cte_Entera]]() =	<push> expresión.valor.sufijo;</push>
valor[[LiteralReal: expresion -> Cte_Real]]() =	<push> expresión.valor.sufijo;</push>
valor[[Logica: expresion1 -> expresion2 expresion3]]() =	valor[[expresion2]];
	valor[[expresion3]];
	GC.Logica(expresion1.operador);
valor[[MenosUnario: expresion1 -> expresion2]]() =	valor[[expresion2]];
2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2	GC.Logica(expresion1.operador);
valor[[Negacion: expresion1 -> expresion2]]() =	valor[[expresion2]];
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	<pre></pre>
	GC.convertirA(TipoEntero, expresion1.tipo);
	<mul></mul>
valor[[Variable: expresion -> ID]]() =	direccion[expresión];
Tally	<pre><!-- Color</td--></pre>
direccion[[Variable: expresión -> ID]]() =	if(expresion.def.ambito == 0)
an ecolonitivariable. expresion > 10 11() -	<pre>PUSHA> expresión.def.offset;</pre>
	else {
	<pre>PUSH BP></pre>
	<push expresión.def.offset;<="" td=""></push>
	\rushin expresion.der.onset,
	\