Procesadores de Lenguajes

Memoria de proyecto - Hito 4: Compilador

Grupo 10
Sergio Colet García
Laura Martínez Tomás
Rodrigo Souto Santos
Li Jie Chen Chen

Grado en Ingeniería informática
Facultad de Informática
Universidad Complutense de Madrid

2023-2024

Índice general

1 Especificación del procesamiento de vinculación

```
var ts //Tabla de simbolos

vincula(prog(Blo)):
    ts = creaTS()
    vincula(Blo)

vincula(bloq(Decs,Insts)):
    abreAmbito(ts)
    vincula(Decs)
    vincula(Insts)
    cierraAmbito(ts)

vincula(si_decs(LDecs)):
    vincula1(LDecs)
    vincula2(LDecs)

vincula1(no_decs()): noop

vincula1(muchas_decs(LDecs,Dec)):
    vincula1(LDecs)
    vincula1(Dec)

vincula2(muchas_decs(LDecs,Dec)):
    vincula2(LDecs)
    vincula2(Dec)

vincula1(una_dec(Dec)):
    vincula1(Dec)

vincula2(una_dec(Dec)):
    vincula2(Dec)

vincula1(muchas_var(LVar,Var)):
    vincula1(LVar)
    vincula1(Var)

vincula2(muchas_var(LVar,Var)):
    vincula2(LVar)
    vincula2(Var)

vincula1(una_var(Var)):
    vincula1(Var)

vincula2(una_var(Var)):
    vincula2(Var)

vincula1(var(Tipo,id)):
    vincula1(Tipo)
    if contiene(ts,id) then
        error
    else
        inserta(ts,id,$)
    end if
```

```
vincula2(var(Tipo,id)):
    vincula2(Tipo)

vincula1(dec_simple(Var)):
    vincula1(Var)

vincula2(dec_simple(Var)):
    vincula2(Var)

vincula1(dec_type(Var)):
    vincula1(Var)

vincula2(dec_type(Var)):
    vincula2(Var)

vincula1(dec_proc(id,PFmls,Blo)):
    if contiene(ts,id) then
        error
    else
        inserta(ts,id,$)
    end if

vincula2(dec_proc(id,PFmls,Blo)):
    abreAmbito(ts)
    vincula2(PFmls)
    vincula1(PFmls)
    vincula(Blo)
    cierraAmbito(ts)

vincula1(tipo_array(Tipo,litEnt)):
    if Tipo != tipo_ident(_) then
        vincula1(Tipo)
    end if

vincula2(tipo_array(Tipo,litEnt)):
    if Tipo == tipo_ident(id) then
        Tipo.vinculo = vinculoDe(ts,id)
        if Tipo.vinculo != dec_type(_) then
            error
        end if
    else
        vincula2(Tipo)
    end if

vincula1(tipo_punt(Tipo)):
    if Tipo != tipo_ident(_) then
        vincula1(Tipo)
    end if

vincula2(tipo_punt(Tipo)):
    if Tipo == tipo_ident(id) then
        Tipo.vinculo = vinculoDe(ts,id)
        if Tipo.vinculo != dec_type(_) then
            error
        end if
    else
        vincula2(Tipo)
    end if
```

```
vincula1(tipo_bool()): noop

vincula2(tipo_bool()): noop

vincula1(tipo_int()): noop

vincula2(tipo_int()): noop

vincula1(tipo_real()): noop

vincula2(tipo_real()): noop

vincula1(tipo_string()): noop

vincula2(tipo_string()): noop

vincula1(tipo_ident(id)):
    $.vinculo = vinculoDe(ts,id)
    if $.vinculo != dec_type(_) then
        error
    end if

vincula2(tipo_ident(id)): noop

vincula(tipo_struct(LVar)):
    vincula1(LVar)

vincula(si_inst(LInst)):
    vincula(LInst):

vincula(no_inst()): noop

vincula(muchas_inst(LInst,Inst)):
    vincula(LInst)
    vincula(Inst)

vincula(una_inst(Inst)):
    vincula(Inst)

vincula1(si_pformal(LPFml)):
    vincula1(LPFml)

vincula2(si_pformal(LPFml)):
    vincula2(LPFml)

vincula1(no_pformal()): noop

vincula2(no_pformal()): noop

vincula1(muchos_pformal(LPFml,PFml)):
    vincula1(LPFml)
    vincula1(PFml)

vincula2(muchos_pformal(LPFml,PFml)):
    vincula2(LPFml)
    vincula2(PFml)

vincula1(un_pformal(PFml))
    vincula1(PFml)
```

```
vincula2(un_pformal(PFml))
    vincula2(PFml)

vincula1(pformal_ref(Tipo,id)):
    vincula1(Tipo)
    if contiene(ts,id) then
        error
    else
        inserta(ts,id,$)
    end if

vincula2(pformal_ref(Tipo,id)):
    vincula2(Tipo)

vincula1(pformal_noref(Tipo,id)):
    vincula1(Tipo)
    if contiene(ts,id) then
        error
    else
        inserta(ts,id,$)
    end if

vincula2(pformal_noref(Tipo,id)):
    vincula2(Tipo)

vincula(si_preales(LPReal)):
    vincula(LPReal)

vincula(no_preales()): noop

vincula(muchas_exp(LPReal,Exp)):
    vincula(LPReal)
    vincula(Exp)

vincula(una_exp(Exp)):
    vincula(Exp)

vincula(inst_eval(Exp)):
    vincula(Exp)

vincula(inst_if(Exp,Blo)):
    vincula(Exp)
    vincula(Blo)

vincula(inst_else(Exp,Blo1,Blo2)):
    vincula(Exp)
    vincula(Blo1)
    vincula(Blo2)

vincula(inst_while(Exp,Blo)):
    vincula(Exp)
    vincula(Blo)

vincula(inst_new(Exp)):
    vincula(Exp)

vincula(inst_delete(Exp)):
    vincula(Exp)

vincula(inst_read(Exp)):
```

```
    vincula(Exp)

vincula(inst_write(Exp)):
    vincula(Exp)

vincula(inst_call(id,PReales)):
    $.vinculo = vinculoDe(ts,Id)
    if $.vinculo == false then
        error
    end if
    vincula(PReales)

vincula(inst_nl()): noop

vincula(inst_blo(Blo)):
    vincula(Blo)

vincula(exp_asig(Opnd0,Opnd1)):
    vincula(Opnd0)
    vincula(Opnd1)

vincula(exp_menor(Opnd0,Opnd1)):
    vincula(Opnd0)
    vincula(Opnd1)

vincula(exp_menIgual(Opnd0,Opnd1)):
    vincula(Opnd0)
    vincula(Opnd1)

vincula(exp_mayor(Opnd0,Opnd1)):
    vincula(Opnd0)
    vincula(Opnd1)

vincula(exp_mayIgual(Opnd0,Opnd1)):
    vincula(Opnd0)
    vincula(Opnd1)

vincula(exp_igual(Opnd0,Opnd1)):
    vincula(Opnd0)
    vincula(Opnd1)

vincula(exp_dist(Opnd0,Opnd1)):
    vincula(Opnd0)
    vincula(Opnd1)

vincula(exp_sum(Opnd0,Opnd1)):
    vincula(Opnd0)
    vincula(Opnd1)

vincula(exp_resta(Opnd0,Opnd1)):
    vincula(Opnd0)
    vincula(Opnd1)

vincula(exp_mult(Opnd0,Opnd1)):
    vincula(Opnd0)
    vincula(Opnd1)

vincula(exp_div(Opnd0,Opnd1)):
    vincula(Opnd0)
    vincula(Opnd1)
```

```
vincula(exp_mod(Opnd0,Opnd1)):
    vincula(Opnd0)
    vincula(Opnd1)

vincula(exp_and(Opnd0,Opnd1)):
    vincula(Opnd0)
    vincula(Opnd1)

vincula(exp_or(Opnd0,Opnd1)):
    vincula(Opnd0)
    vincula(Opnd1)

vincula(exp_menos(Exp)):
    vincula(Exp)

vincula(exp_not(Exp)):
    vincula(Exp)

vincula(inst_index(Opnd0,Opnd1)):
    vincula(Opnd0)
    vincula(Opnd1)

vincula(exp_reg(Exp,id)):
    vincula(Exp)

vincula(exp_ind(Exp)):
    vincula(Exp)

vincula(exp_true()): noop

vincula(exp_false()): noop

vincula(exp_litEnt(litEnt)): noop

vincula(exp_litReal(litReal)): noop

vincula(exp_litCad(litCad)): noop

vincula(exp_iden(id)):
    $.vinculo = vinculoDe(ts,Id)
    if $.vinculo == false then
        error
    end if

vincula(exp_null()): noop
```

2 | Especificación del procesamiento de comprobación de tipos

```
tipado(prog(Blo)):
    tipado(Blo)
    $.tipo = Blo.tipo

tipado(bloq(Decs,Insts)):
    tipado(Decs)
    tipado(Insts)
    $.tipo = ambos-ok(Decs.tipo,Insts.tipo)

tipado(no_decs()): $.tipo = ok

tipado(si_decs(LDecs)):
    tipado(LDecs)
    $.tipo = LDecs.tipo

tipado(muchas_decs(LDecs,Dec)):
    tipado(LDecs)
    tipado(Dec)
    $.tipo = ambos-ok(LDecs.tipo,Dec.tipo)

tipado(una_dec(Dec)):
    tipado(Dec)
    $.tipo = Dec.tipo

tipado(muchas_var(LVar,Var)):
    tipado(LVar)
    tipado(Var)
    $.tipo = ambos-ok(LVar.tipo,Var.tipo)

tipado(una_var(Var)):
    tipado(Var)
    $.tipo = Var.tipo

tipado(var(Tipo,id)):
    tipado(Tipo)
    $.tipo = Tipo.tipo

tipo(dec_simple(Var)):
    tipado(Var)
    $.tipo = Var.tipo

tipo(dec_type(Var)):
    tipado(Var)
    $.tipo = Var.tipo

tipo(dec_proc(id,PFmls,Blo)):
    tipado(PFmls)
    tipado(Bloq)
    $.tipo = Blo.tipo

tipado(tipo_array(Tipo,litEnt)):
    if litEnt < 0 then
        $.tipo = error
    else
```

```
            tipado(Tipo)
            $.tipo = Tipo.tipo
        end if

    tipado(tipo_punt(Tipo)):
        tipado(Tipo)
        $.tipo = Tipo.tipo

    tipado(tipo_bool()): $.tipo = bool

    tipado(tipo_int()): $.tipo = int

    tipado(tipo_real()): $.tipo = real

    tipado(tipo_string()): $.tipo = string

    tipado(tipo_ident(id)):
        let $.vinculo = var(T,I) in
            $.tipo = T
        end let

    tipado(tipo_struct(LVar)):
        comprobarRepetidosStruct()
        tipado(LVar)
        $.tipo = LVar.tipo

    tipado(si_inst(LInst)):
        tipado(LInst)
        $.tipo = LInst.tipo

    tipado(no_inst()): $.tipo = ok

    tipado(muchas_inst(LInst,Inst)):
        tipado(LInst)
        tipado(Inst)
        $.tipo = ambos-ok(LInst.tipo,Inst.tipo)

    tipado(una_inst(Inst)):
        tipado(Inst)
        $.tipo = Inst.tipo

    tipado(si_pformal(LPFml)):
        tipado(LPFml)
        $.tipo = LPFml.tipo

    tipado(no_pformal()): $.tipo = ok

    tipado(muchos_pformal(LPFml,PFml)):
        tipado(LPFml)
        tipado(PFml)
        $.tipo = ambos-ok(LPFml.tipo,PFml.tipo)

    tipado(un_pformal(PFml))
        tipado(PFml)
        $.tipo = PFml.tipo

    tipado(si_preales(LPReal)):
        tipado(LPReal)
        $.tipo = LPReal.tipo
```

```
tipado(no_preales()): $.tipo = ok

tipado(muchas_exp(LPReal,Exp)):
    tipado(LPReal)
    tipado(Exp)
    $.tipo = ambos-ok(LPReal.tipo,Exp.tipo)

tipado(una_exp(Exp)):
    tipado(Exp)
    $.tipo = Exp.tipo

tipado(pformal_ref(Tipo,id)):
    tipado(Tipo)
    $.tipo = Tipo.tipo

tipado(pformal_noref(Tipo,id)):
    tipado(Tipo)
    $.tipo = Tipo.tipo

tipado(inst_eval(Exp)):
    tipado(Exp)
    $.tipo = Exp.tipo

tipado(inst_if(Exp,Blo)):
    tipado(Exp)
    tipado(Blo)
    if Exp.tipo == bool ^ Bloq.tipo == ok then
        $.tipo = ok
    else
        $.tipo = error

tipado(inst_else(Exp,Blo1,Blo2)):
    tipado(Exp)
    tipado(Blo1)
    tipado(Blo2)
    if Exp.tipo == bool ^ Blo1 == ok ^ Blo2 == ok then
        $.tipo = ok
    else
        $.tipo = error
    end if

tipado(inst_while(Exp,Blo)):
    tipado(Exp)
    tipado(Blo)
    if Exp.tipo == bool ^ Bloq.tipo == ok then
        $.tipo = ok
    else
        $.tipo = error

tipado(inst_new(Exp)):
    tipado(Exp)
    if ref!(Exp.tipo) == tipo_punt(T) then
        $.tipo = ok
    else
        aviso-error(T)
        $.tipo = error
    end if

tipado(inst_delete(Exp)):
    tipado(Exp)
```

```
    if ref!(Exp.tipo) == tipo_punt(T) then
        $.tipo = ok
    else
        aviso-error(T)
        $.tipo = error
    end if

tipado(inst_read(Exp)):
    tipado(Exp)
    if es-designador(Exp) then
        if ref!(Exp.tipo) == literalEntero v ref!(Exp.tipo) == literalReal v ref!(Exp.
            $.tipo = ok
        else
            $.tipo = error
        end if
    else
        $.tipo = error
    end if

tipado(inst_write(Exp)):
    tipado(Exp)
    if ref!(Exp.tipo) == literalEntero v ref!(Exp.tipo) == literalReal v ref!(Exp.tipo
        $.tipo = ok
    else
        $.tipo = error
    end if

tipado(inst_call(id,PReales)):
    if $.vinculo == dec_proc(id, PFmls, Blo) then
        if num_elems(PReales) != num_elems(PFmls) then
            $.tipo = error
        else
            tipado(PReales)
            $.tipo = tipado_parametros(PReales, PFmls)
        end if
    else
        $.tipo = error
    end if

tipado_parametros(no_preales(), no_pformal()):
    return ok

tipado_parametros(si_preales(LPReal), si_pformal(LPFml)):
    return tipado_paramentros(LPReal, LPFml)

tipado_parametros(muchas_exp(LPReal,Exp), muchas_pformal(LPFml, PFml)):
    return ambos_ok(tipado_parametros(LPReal,LPFml), tipado_parametros(Exp,PFml))

tipado_parametros(una_exp(Exp), pformal_noref(Tipo,id)):
    tipado(Exp)
    if compatibles(Exp.tipo,Tipo.tipo) then
        return ok
    else
        aviso-error(Tipo.tipo)
        return error

tipado_parametros(una_exp(Exp), pformal_ref(Tipo,id)):
    tipado(Exp)
    if es-designador(Exp) ^ compatibles(Exp.tipo,Tipo.tipo) then
        return ok
```

```
        else
            aviso-error(Tipo.tipo)
            return error

    tipado(inst_nl()): $.tipo = ok

    tipado(inst_blo(Blo)):
        tipado(Blo)
        $.tipo = Blo.tipo

    tipado(exp_asig(Opnd0,Opnd1)):
        tipado(Opnd0)
        tipado(Opnd1)
        if es-designador(Opnd0) then
            if compatibles(Opnd0.tipo, Opnd1.tipo) then
                $.tipo = Opnd0.tipo
            else
                aviso-error(Opnd0.tipo,Opnd1.tipo)
                $.tipo = error
            end if
        else
            error
            $.tipo = error
        end if

    tipado(exp_menor(Opnd0,Opnd1)):
        tipado-bin2(Opnd0,Opnd1,$)

    tipado(exp_menIgual(Opnd0,Opnd1)):
        tipado-bin2(Opnd0,Opnd1,$)

    tipado(exp_mayor(Opnd0,Opnd1)):
        tipado-bin2(Opnd0,Opnd1,$)

    tipado(exp_mayIgual(Opnd0,Opnd1)):
        tipado-bin2(Opnd0,Opnd1,$)

    tipado(exp_igual(Opnd0,Opnd1)):
        tipado-bin3(Opnd0,Opnd1,$)

    tipado(exp_dist(Opnd0,Opnd1)):
        tipado-bin3(Opnd0,Opnd1,$)

    tipado(exp_sum(Opnd0,Opnd1)):
        tipado-bin1(Opnd0,Opnd1,$)

    tipado(exp_resta(Opnd0,Opnd1)):
        tipado-bin1(Opnd0,Opnd1,$)

    tipado(exp_mult(Opnd0,Opnd1)):
        tipado-bin1(Opnd0,Opnd1,$)

    tipado(exp_div(Opnd0,Opnd1)):
        tipado-bin1(Opnd0,Opnd1,$)

    tipado(exp_mod(Opnd0,Opnd1)):
        tipado(Opnd0)
        tipado(Opnd1)
        if ref!(Opnd0.tipo) == literalEntero ^ ref!(Opnd1.tipo) == literalEntero then
            return literalEntero
```

```
    else
        return error
    end if

tipado(exp_and(Opnd0,Opnd1)):
    tipado(Opnd0)
    tipado(Opnd1)
    if ref!(Opnd0.tipo) == bool ^ ref!(Opnd1.tipo) == bool then
        return bool
    else
        return error
    end if

tipado(exp_or(Opnd0,Opnd1)):
    tipado(Opnd0)
    tipado(Opnd1)
    if ref!(Opnd0.tipo) == bool ^ ref!(Opnd1.tipo) == bool then
        return bool
    else
        return error
    end if

tipado(exp_menos(Exp)):
    tipado(Exp)
    if ref!(Exp.tipo) == literalEntero then
        return literalEntero
    if ref!(Exp.tipo) == literalReal then
        return literalReal
    else
        return error

tipado(exp_not(Exp)):
    tipado(Exp)
    if ref!(Exp.tipo) == bool then
        return bool
    else
        return error

tipado(inst_index(Opnd0,Opnd1)):
    tipado(Opnd0)
    tipado(Opnd1)
    if ref!(Opnd0.tipo) == tipo_array(T) ^ ref!(Opnd1.tipo) == literalEntero then
        return tipo_array(T)
    else
        return error
    end if

tipado(exp_reg(Exp,id)):
    tipado(Exp)
    let $.vinculo = var(T,I) in
        if ref!(Exp.tipo) == tipo_struct(T) ^ structContieneId(I) then
            return T.tipo
        else
            return error
    end let

tipado(exp_ind(Exp)):
    tipado(Exp)
    if ref!(Exp.tipo) == tipo_punt(T) then
        $.tipo = ok
```

```
    else
        aviso-error(T)
        $.tipo = error
    end if

tipado(exp_true()): $.tipo = true

tipado(exp_false()): $.tipo = false

tipado(exp_litEnt(litEnt)): $.tipo = literalEntero

tipado(exp_litReal(litReal)): $.tipo = literalReal

tipado(exp_litCad(litCad)): $.tipo = literalCadena

tipado(exp_iden(id)):
    let $.vinculo = Dec in
        if Dec = var(Tipo,id) v Dec = pformal_ref(Tipo,id) v Dec = pformal_noref(Tipo,
            $.tipo = Tipo
        else
            aviso-error(id)
            $.tipo = error()
    end let

tipado(exp_null()): $.tipo = null

ambos-ok(T0,T1):
    if T0 == ok ^ T1 == ok then
        return ok
    else
        return error
    end if

aviso-error(T0,T1):
    if T0 != error ^ T1 != error then
        error
    end if

aviso-error(T):
    if T != error then
        error
    end if

ref!(T):
    if T == Ref(I) then
        let T.vinculo = Dec_type(Var(T',I)) in
            return ref!(T')
        end let
    else
        return T
    end if

tipado-bin1(E0,E1,E):
    tipado(E0)
    tipado(E1)
    E.tipo = tipo-bin1(E0.tipo,E1.tipo)

tipo-bin1(T0,T1):
    if compatibles(T0,T1)  then
        return T0
```

```
    else if ref!(T1) == literalReal then
        if ref!(T0) == literalReal v ref!(T0) == literalEntero then
            return literalReal
        else
            return error
        end if
    else if ref!(T0) == literalReal then
        if ref!(T1) == literalReal v ref!(T1) == literalEntero then
            return literalReal
        else
            return error
        end if
    else
        return error
    end if

tipado-bin2(E0,E1,E):
    tipado(E0)
    tipado(E1)
    E.tipo = tipo-bin2(E0.tipo,E1.tipo)

tipo-bin2(T0,T1):
    if (ref!(T0) == literalReal v ref!(T0) == literalEntero) ^ (ref!(T1) == literalRea
        return bool
    else if ref!(T0) == bool ^ ref!(T1) == bool then
        return bool
    else if ref!(T0) == literalCadena ^ ref!(T1) == literalCadena then
        return bool
    else
        return error
    end if

tipado-bin3(E0,E1,E):
    tipado(E0)
    tipado(E1)
    E.tipo = tipo-bin3(E0.tipo,E1.tipo)

tipo-bin3(T0,T1):
    if (ref!(T0) == literalReal v ref!(T0) == literalEntero) ^ (ref!(T1) == literalRea
        return bool
    else if ref!(T0) == bool ^ ref!(T1) == bool then
        return bool
    else if ref!(T0) == literalCadena ^ ref!(T1) == literalCadena then
        return bool
    else if (ref!(T0) == tipo_punt(T) v ref!(T0) == null) ^ (ref!(T1) == tipo_punt(T)
        return bool
    else
        return error
    end if

compatibles(T1,T2):
    let T1' = ref!(T1) ^ T2' = ref!(T2) in
        if T1' == T2' then
            return true
        else if T1 == literalReal ^ (T2== literalEntero v T2==literalReal)) then
            return true
        else if T1 == tipo_array(T,literalEntero) ^ T2 == tipo_array(T,literalEntero)
            if tam(T1) != tam(T2) return false;
            else return true
        else if T1 == tipo_struct(LVar) ^ T2 == tipo_struct(LVar) then
```

```
            R1 = registroStruct(T1)
            R2 = registroStruct(T2)
            if R1 == null v R2 == null v R1.size != R2.size v !tiposStructMismoIdComp
                return false
            else return true
        else if T1== tipo_punt(T) ^ T2== null then
            return true
        else if  T1== tipo_punt(T) ^ T2 == tipo_punt(T') then
            if T != T' then return false
            else return true
        else
            return false
        end if
    end let

es-designador(E):
    return E = id(v) v E = elem1(E') v E = elem2(E')
```

3 | Especificación del procesamiento de asignación de espacio

```
global dir = 0 //contador de direcciones
global nivel = 0 //nivel de anidamiento
global max_dir = 0

asig-espacio(prog(Blo)):
    asig-espacio(Blo)

asig-espacio(bloq(Decs,Insts)):
    dir_ant = dir
    asig-espacio(Decs)
    asig-espacio(Insts)
    dir = dir_ant

asig-espacio(si_decs(LDecs)):
    asig-espacio1(Ldecs)
    asig-espacio2(Ldecs)

asig-espacio(no_decs()): skip

asig-espacio1(muchas_decs(LDecs,Dec)):
    asig-espacio1(LDecs)
    asig-espacio1(Dec)

asig-espacio2(muchas_decs(LDecs,Dec)):
    asig-espacio2(LDecs)
    asig-espacio2(Dec)

asig-espacio1(una_dec(Dec)):
    asig-espacio1(Dec)

asig-espacio2(una_dec(Dec)):
    asig-espacio2(Dec)

asig-espacio1(muchas_var(LVar,Var)):
    asig-espacio1(LVar)
    asig-espacio1(Var)

asig-espacio2(muchas_var(LVar,Var)):
    asig-espacio2(LVar)
    asig-espacio2(Var)

asig-espacio1(una_var(Var)):
    asig-espacio1(Var)

asig-espacio2(una_var(Var)):
    asig-espacio2(Var)

asig-espacio1(var(Tipo,id)):
    $.dir = dir
    $.nivel = nivel
    asig-espacio-tipo(Tipo)
    dir += Tipo.tam
```

```
asig-espacio2(var(Tipo,id)): skip

asig-espacio1(dec_simple(Var)):
    asig-espacio1(Var)

asig-espacio2(dec_simple(Var)):
    asig-espacio2(Var)

asig-espacio1(dec_type(Var)):
    asig-espacio1(Var)

asig-espacio2(dec_type(Var)):
    asig-espacio2(Var)

asig-espacio1(dec_proc(id,PFmls,Blo)):
    ant_dir = dir
    ant_max_dir = max_dir
    nivel = nivel + 1
    $.nivel = nivel
    dir = 0
    max_dir = 0
    asigna_espacio1(PFmls)
    asigna_espacio1(Blo)
    $.tam_datos = dir
    dir = ant_dir
    max_dir = max_dir_aux
    nivel = nivel - 1
    if dir > max_dir then
        max_dir = dir

asig-espacio2(dec_proc(id,PFmls,Blo)):
    asigna_espacio2(PFmls)
    asigna_espacio2(Blo)


asig-espacio-tipo(Tipo):
        si indefinido(Tipo.tam)
            asig-espacio-tipo1(Tipo)
            asig-espacio-tipo2(Tipo)
        fin s


asig-espacio-tipo1(tipo_array(Tipo,litEnt)):
    $.tam = litEnt.tam
    si Tipo != tipo_ident(_)
        asig-espacio-tipo1(Tipo)
    fin si

asig-espacio-tipo2(tipo_array(Tipo,litEnt)):
    si Tipo = tipo_ident(_)
        sea Tipo.vinculo = var(Tipo,_)
            asig-espacio-tipo(Tipo)
        $.tam= Tipo.tam*litEnt.tam
    fin si

asig-espacio-tipo1(tipo_punt(Tipo)):
    $.tam = 1
    si Tipo != tipo_ident(_)
        asig-espacio-tipo1(Tipo)
    fin si
```

```
asig-espacio-tipo2(tipo_punt(Tipo)):
    si Tipo = tipo_ident(_)
        sea Tipo.vinculo = var(Tipo,_)
            asig-espacio-tipo(Tipo)
        $.tam= Tipo.tam
    fin si


asig-espacio-tipo1(tipo_bool()):
    $.tam = 1

asig-espacio-tipo2(tipo_bool()): skip

asig-espacio-tipo1(tipo_int()):
    $.tam = 1

asig-espacio-tipo2(tipo_int()): skip

asig-espacio-tipo1(tipo_real()):
    $.tam = 1

asig-espacio-tipo2(tipo_real()): skip

asig-espacio-tipo1(tipo_string()):
    $.tam = 1

asig-espacio-tipo2(tipo_string()): skip

asig-espacio-tipo1(tipo_ident(id)):
    asig-espacio-tipo1($.vinculo)
    sea $.vinculo = var(Tipo,_) en
        $.tam = Tipo.tam

asig-espacio-tipo2(tipo_ident(id)): skip

asig-espacio1(tipo_struct(LVar)):
    dir_ant = dir
    dir = 0
    asig-espacio1(LVar)
    $.tam = dir
    dir  = dir_ant

asig-espacio2(tipo_struct(LVar)):
    asig-espacio2(LVar)

asig-espacio(si_inst(LInst)):
    asig-espacio(LInst):

asig-espacio(no_inst()): skip

asig-espacio(muchas_inst(LInst,Inst)):
    asig-espacio(LInst)
    asig-espacio(Inst)

asig-espacio(una_inst(Inst)):
    asig-espacio(Inst)

asig-espacio1(si_pformal(LPFml)):
    asig-espacio1(LPFml)
```

```
asig-espacio2(si_pformal(LPFml)):
    asig-espacio2(LPFml)

asig-espacio1(no_pformal()): skip

asig-espacio2(no_pformal()): skip

asig-espacio1(muchos_pformal(LPFml,PFml)):
    asig-espacio1(LPFml)
    asig-espacio1(PFml)

asig-espacio2(muchos_pformal(LPFml,PFml)):
    asig-espacio2(LPFml)
    asig-espacio2(PFml)

asig-espacio1(un_pformal(PFml))
    asig-espacio1(PFml)

asig-espacio2(un_pformal(PFml))
    asig-espacio2(PFml)

asig-espacio1(pformal_ref(Tipo,id)):
    $.dir = dir
    $.nivel = nivel
    asig-espacio-tipo(Tipo)
    dir += Tipo.tam

asig-espacio2(pformal_ref(Tipo,id)):
    asig-espacio-tipo(Tipo)

asig-espacio1(pformal_noref(Tipo,id)):
    $.dir = dir
    $.nivel = nivel
    asig-espacio-tipo(Tipo)
    dir += Tipo.tam

asig-espacio2(pformal_noref(Tipo,id)):
    asig-espacio-tipo(Tipo)


asig-espacio(si_preales(LPReal)):
    asig-espacio(LPReal)

asig-espacio(no_preales()): skip

asig-espacio(muchas_exp(LPReal,Exp)):
    asig-espacio(LPReal)
    asig-espacio(Exp)

asig-espacio(una_exp(Exp)):
    asig-espacio(Exp)

asig-espacio(inst_eval(Exp)): skip

asig-espacio(inst_if(Exp,Blo)):
    asig-espacio(Blo)

asig-espacio(inst_else(Exp,Blo1,Blo2)):
    asig-espacio(Blo1)
```

```
        asig-espacio(Blo2)

    asig-espacio(inst_while(Exp,Blo)):
        asig-espacio(Blo)

    asig-espacio(inst_new(Exp)): skip

    asig-espacio(inst_delete(Exp)): skip

    asig-espacio(inst_read(Exp)): skip

    asig-espacio(inst_write(Exp)): skip

    asig-espacio(inst_call(id,PReales)): skip

    asig-espacio(inst_nl()): skip

    asig-espacio(inst_blo(Blo)):
        asig-espacio(Blo)

    asig-espacio(exp_asig(Opnd0,Opnd1)):
        asig-espacio(Opnd0)
        asig-espacio(Opnd1)

    asig-espacio(exp_menor(Opnd0,Opnd1)):
        asig-espacio(Opnd0)
        asig-espacio(Opnd1)

    asig-espacio(exp_menIgual(Opnd0,Opnd1)):
        asig-espacio(Opnd0)
        asig-espacio(Opnd1)

    asig-espacio(exp_mayor(Opnd0,Opnd1)):
        asig-espacio(Opnd0)
        asig-espacio(Opnd1)

    asig-espacio(exp_mayIgual(Opnd0,Opnd1)):
        asig-espacio(Opnd0)
        asig-espacio(Opnd1)

    asig-espacio(exp_igual(Opnd0,Opnd1)):
        asig-espacio(Opnd0)
        asig-espacio(Opnd1)

    asig-espacio(exp_dist(Opnd0,Opnd1)):
        asig-espacio(Opnd0)
        asig-espacio(Opnd1)

    asig-espacio(exp_sum(Opnd0,Opnd1)):
        asig-espacio(Opnd0)
        asig-espacio(Opnd1)

    asig-espacio(exp_resta(Opnd0,Opnd1)):
        asig-espacio(Opnd0)
        asig-espacio(Opnd1)

    asig-espacio(exp_mult(Opnd0,Opnd1)):
        asig-espacio(Opnd0)
        asig-espacio(Opnd1)
```

```
asig-espacio(exp_div(Opnd0,Opnd1)):
    asig-espacio(Opnd0)
    asig-espacio(Opnd1)

asig-espacio(exp_mod(Opnd0,Opnd1)):
    asig-espacio(Opnd0)
    asig-espacio(Opnd1)

asig-espacio(exp_and(Opnd0,Opnd1)):
    asig-espacio(Opnd0)
    asig-espacio(Opnd1)

asig-espacio(exp_or(Opnd0,Opnd1)):
    asig-espacio(Opnd0)
    asig-espacio(Opnd1)

asig-espacio(exp_menos(Exp)):
    asig-espacio(Exp)

asig-espacio(exp_not(Exp)):
    asig-espacio(Exp)

asig-espacio(inst_index(Opnd0,Opnd1)):
    asig-espacio(Opnd0)
    asig-espacio(Opnd1)

asig-espacio(exp_reg(Exp,id)):
    asig-espacio($.vinculo)
    sea $.vinculo = var(Tipo,_) en
        $.tam = Tipo.tam


asig-espacio(exp_ind(Exp)):
    asig-espacio(Exp)

asig-espacio(exp_true()): skip

asig-espacio(exp_false()): skip

asig-espacio(exp_litEnt(litEnt)): skip

asig-espacio(exp_litReal(litReal)): skip

asig-espacio(exp_litCad(litCad)): skip

asig-espacio(exp_iden(id)):
    asig-espacio($.vinculo)
    sea $.vinculo = var(Tipo,_) en
        $.tam = Tipo.tam


asig-espacio(exp_null()): skip
```

4 | Instrucciones de la máquina-p necesarias para soportar la traducción de Tiny a código-p

4.0.1. Instrucciones Aritmético-Lógicas

Estas instrucciones:

- *Desapilan los argumentos de la pila de evaluación (los argumentos aparecen en la pila en orden inverso; por ejemplo, si la operación necesita dos argumentos, en la cima estará el 2ž argumento, en la sub-cima el 1er argumento).*
- *Realizan la operación.*
- *Apilan el resultado en la pila de evaluación.*

Cuadro 4.0.1: Instrucciones Aritmético-Lógicas

suma	Desapila los 2 primeros argumentos de la pila y apila su suma
resta	Desapila los 2 primeros argumentos de la pila y apila su resta
menos	Desapila el primer argumento de la pila y apila su negativo
mult	Desapila los 2 primeros argumentos de la pila y apila su multiplicación
div	Desapila los 2 primeros argumentos de la pila y apila su división
mod	Desapila los 2 primeros argumentos de la pila y apila su módulo
and	Desapila los 2 primeros argumentos de la pila y apila el resultado de la operación and
or	Desapila los 2 primeros argumentos de la pila y apila el resultado de la operación or
not	Desapila el primer argumento de la pila y apila su contrario
menor	Desapila los 2 primeros argumentos de la pila y apila true si el primer argumento es menor
menIgual	Desapila los 2 primeros argumentos de la pila y apila true si el primer argumento es menor o igual
mayor	Desapila los 2 primeros argumentos de la pila y apila true si el primer argumento es mayor
mayIgual	Desapila los 2 primeros argumentos de la pila y apila true si el primer argumento es mayor o igual
igual	Desapila los 2 primeros argumentos de la pila y apila true si ambos argumentos son iguales

Continúa en la siguiente página

Cuadro 4.0.1: Instrucciones Aritmético-Lógicas (Continuación)

distinto	Desapila los 2 primeros argumentos de la pila y apila true si los argumentos son distintos

4.0.2. Instrucciones de movimiento de datos

Cuadro 4.0.2: Instrucciones de movimiento de datos

apilaInt(v)	Apila el valor entero v en la pila de evaluación
apilaBool(v)	Apila el valor booleano v en la pila de evaluación
apilaReal(v)	Apila el valor real v en la pila de evaluación
apilaString(v)	Apila la cadena de caracteres v en la pila de evaluación
desapila(v)	desapila el valor de la cima de la pila de evaluación
apilaInd	Desapila una dirección (dir) de la pila de evaluación, y apila (en dicha pila) el contenido de la celda (dir) en la memoria de datos
desapilaind	Desapila el valor v y una dirección (dir) de la pila de evaluación (primero v, después d), y actualiza el contenido de la celda dir en la memoria de datos con el valor de v
mueve(n)	Desapila dos direcciones dir1 y dir0 de la pila de evaluación en ese orden y copia el contenido de las n celdas consecutivas desde la dirección dir1 a las correspondientes n celdas que comienzan en la dirección dir0.
int2real()	desapila un 'entero' y apila el valor 'real' equivalente.

4.0.3. Instrucciones de salto

Cuadro 4.0.3: Instrucciones de salto

irA(d)	Salto incondicional a la dirección d.
irF(d)	Desapila el valor v. Si es falso salta a la dirección d.
irV(d)	Desqapila el valor v. Si es verdadero salta a la dirección d.
irInd	Desapila una dirección d de la pila de evaluación, y realiza un salto incondicional a dicha dirección

4.0.4. Instrucciones de gestión de memoria dinámica

Cuadro 4.0.4: Instrucciones de gestión de memoria dinámica

alloc(t)	e Reserva un bloque de n celdas consecutivas en el heap y apila la dirección de comienzo en la pila de evaluación.
dealloc(t)	Desapila una dirección d de la pila de evaluación y libera en el heap el bloque de n celdas consecutivas que comienza en d
copia(t)	Copia el valor del tipo t que se encuentra almacenado a partir de la dirección d en el bloque que comienza a partir de la dirección d

4.0.5. Instrucciones de Entrada y Salida

Cuadro 4.0.5: Instrucciones de Entrada y salida

read	Desapila la dirección d de la pila de evaluacion, lee el valor v correspondiente de la entrada y lo escribe en la celda de direccion d.
write	Desapila un valor v de la pila de evaluacion y lo escribe por la salida

4.0.6. Instrucciones de Soporte de la ejecución de procedimientos

Cuadro 4.0.6: Instrucciones de Soporte de la ejecución de procedimientos

activa(n,t,dir)	Reserva espacio en el segmento de pila de registros de activación para ejecutar un procedimiento que tiene nivel de anidamiento **n** y tamaño de datos locales **t**. Así mismo, almacena en la zona de control de dicho registro **dir** como dirección de retorno. También almacena en dicha zona de control el valor del display de nivel **n**. Por último, apila en la pila de evaluación la dirección de comienzo de los datos en el registro creado
apilad(n)	Apila en la pila de evaluación el valor del display de nivel **n**
desapilad(n)	Desapila una dirección **dir** de la pila de evaluación en el display de nivel **n**

Continúa en la siguiente página

Cuadro 4.0.6: Instrucciones de Soporte de la ejecución de procedimientos (Continuación)

desactiva(n,t)	Libera el espacio ocupado por el registro de activación actual, restaurando adecuadamente el estado de la máquina. **n** indica el nivel de anidamiento del procedimiento asociado; **t** el tamaño de los datos locales. De esta forma, la instrucción: (i) apila en la pila de evaluación la dirección de retorno; (ii) restaura el valor del display de nivel **n** al antiguo valor guardado en el registro; (iii) decrementa el puntero de pila de registros de activación en el tamaño ocupado por el registro
dup	Consulta el valor **v** de la cima de la pila y apila de nuevo dicho valor.
stop	Detiene la máquina-p

5 | Especificación del procesamiento de etiquetado

```
var etq = 0
var sub_pendientes = pila_vacia()

etiquetado(prog(Blo)):
    etiquetado(Blo)

etiquetado(bloq(Decs,Insts)):
    $.prim = etq
    recolecta_procs(Decs)
    etiquetado(Insts)
    etq++
    while != es-vacia(sub_pendientes)
        sub = cima(sub_pendientes)
        desapila(sub_pendientes)
        let sub = dec_proc(id,Param,Decs,Is) in
            sub.prim = etq
            etq++
            recolecta_procs(Decs)
            etiquetado(Is)
            etq+=2
            sub.sig = etq
        end let
    end while
    $.sig = etq

etiquetado(si_decs(LDecs)):
    $.prim = etq
    etiquetado(Ldecs)
    $.sig = etq

etiquetado(no_decs()):noop

etiquetado(muchas_decs(LDecs,Dec)):
    $.prim = etq
    etiquetado(LDecs)
    etiquetado(Dec)
    $.sig = etq

etiquetado(una_dec(Dec)):
    $.prim = etq
    etiquetado(Dec)
    $.sig = etq

etiquetado(muchas_var(LVar,Var)):
    $.prim = etq
    etiquetado(LVar)
    etiquetado(Var)
    $.sig = etq

etiquetado(una_var(Var)):
    $.prim = etq
    etiquetado(Var)
    $.sig = etq
```

```
etiquetado(var(Tipo,id)):
    $.prim = etq
    etiquetado(Tipo)
    $.sig = etq

etiquetado(dec_simple(Var)):
    $.prim = etq
    etiquetado(Var)
    $.sig = etq

etiquetado(dec_type(Var)):
    $.prim = etq
    etiquetado(Var)
    $.sig = etq

etiquetado(dec_proc(id,PFmls,Blo)):
    $.prim = etq
    etiquetado(PFmls)
    etiquetado(Blo)
    $.sig = etq

etiquetado(tipo_array(Tipo,litEnt)):
    $.prim = etq
    etiquetado(Tipo)
    $.sig = etq

etiquetado(tipo_punt(Tipo)):
    $.prim = etq
    etiquetado(Tipo)
    $.sig = etq

etiquetado(tipo_bool()): noop

etiquetado(tipo_int()): noop

etiquetado(tipo_real()): noop

etiquetado(tipo_string()): noop

etiquetado(tipo_ident(id)):
    $.prim = etq
    etq++
    $.sig = etq

etiquetado(tipo_struct(LVar)):
    $.prim = etq
    etiquetado(LVar)
    $.sig = etq

etiquetado(si_inst(LInst)):
    $.prim = etq
    etiquetado(LInst):
    $.sig = etq

etiquetado(no_inst()): noop

etiquetado(muchas_inst(LInst,Inst)):
    $.prim = etq
    etiquetado(LInst)
```

```
        etiquetado(Inst)
        $.sig = etq

    etiquetado(una_inst(Inst)):
        $.prim = etq
        etiquetado(Inst)
        $.sig = etq

    etiquetado(si_pformal(LPFml)):
        $.prim = etq
        etiquetado(LPFml)
        $.sig = etq

    etiquetado(no_pformal()): noop

    etiquetado(muchos_pformal(LPFml,PFml)):
        $.prim = etq
        etiquetado(LPFml)
        etiquetado(PFml)
        $.sig = etq

    etiquetado(un_pformal(PFml))
        $.prim = etq
        etiquetado(PFml)
        $.sig = etq

    etiquetado(pformal_ref(Tipo,id)):
        $.prim = etq
        etiquetado(Tipo)
        $.sig = etq

    etiquetado(pformal_noref(Tipo,id)):
        $.prim = etq
        etiquetado(Tipo)
        $.sig = etq


    etiquetado(si_preales(LPReal)):
        $.prim = etq
        etiquetado(LPReal)
        $.sig = etq

    etiquetado(no_preales()): noop

    etiquetado(muchas_exp(LPReal,Exp)):
        $.prim = etq
        etiquetado(LPReal)
        etq++
        etiquetado(Exp)
        etiquetado-acc-val(Exp)
        etq++
        $.sig = etq

    etiquetado(una_exp(Exp)):
        $.prim = etq
        etiquetado(Exp)
        etq +=2
        $.sig = etq

    etiquetado(inst_eval(Exp)):
```

```
        $.prim = etq
        etiquetado(Exp)
        etq +=2
        $.sig = etq

    etiquetado(inst_if(Exp,Blo)):
        $.prim = etq
        etiquetado(Exp)
        etiquetado-acc-val(Exp)
        etq++
        etiquetado(Blo)
        etq++
        $.sig = etq

    etiquetado(inst_else(Exp,Blo1,Blo2)):
        $.prim = etq
        etiquetado(Exp)
        etiquetado-acc-val(Exp)
        etq++
        etiquetado(Blo1)
        etq++
        etiquetado(Blo2)
        etq++
        $.sig = etq

    etiquetado(inst_while(Exp,Blo)):
        $.prim = etq
        etiquetado(Exp)
        etiquetado-acc-val(Exp)
        etq++
        etiquetado(Blo)
        etq++
        $.sig = etq

    etiquetado(inst_new(Exp)):
        $.prim = etq
        etiquetado(Exp)
        etq +=2
        $.sig = etq

    etiquetado(inst_delete(Exp)):
        $.prim = etq
        etiquetado(Exp)
        etq +=2
        $.sig = etq

    etiquetado(inst_read(Exp)):
        $.prim = etq
        etiquetado(Exp)
        etq +=2
        $.sig = etq

    etiquetado(inst_write(Exp)):
        $.prim = etq
        etiquetado(Exp)
        etq +=2
        $.sig = etq

    etiquetado(inst_call(id,PReales)):
        $.prim = etq
```

```
        etq++
        etiquetado-paso-param($.vinculo, PReales)
        etq++
        $.sig = etq

    etiquetado(inst_nl()): noop

    etiquetado(inst_blo(Blo)):
        $.prim = etq
        etiquetado(Blo)
        $.sig = etq

    etiquetado(exp_asig(Opnd0,Opnd1)):
        $.prim = etq
        etiquetado-opnds(Opnd0,Opnd1)
        etq++
        $.sig = etq

    etiquetado(exp_menor(Opnd0,Opnd1)):
        $.prim = etq
        etiquetado-opnds(Opnd0,Opnd1)
        etq++
        $.sig = etq

    etiquetado(exp_menIgual(Opnd0,Opnd1)):
        $.prim = etq
        etiquetado-opnds(Opnd0,Opnd1)
        etq++
        $.sig = etq

    etiquetado(exp_mayor(Opnd0,Opnd1)):
        $.prim = etq
        etiquetado-opnds(Opnd0,Opnd1)
        etq++
        $.sig = etq

    etiquetado(exp_mayIgual(Opnd0,Opnd1)):
        $.prim = etq
        etiquetado-opnds(Opnd0,Opnd1)
        etq++
        $.sig = etq

    etiquetado(exp_igual(Opnd0,Opnd1)):
        $.prim = etq
        etiquetado-opnds(Opnd0,Opnd1)
        etq++
        $.sig = etq

    etiquetado(exp_dist(Opnd0,Opnd1)):
        $.prim = etq
        etiquetado-opnds(Opnd0,Opnd1)
        etq++
        $.sig = etq

    etiquetado(exp_sum(Opnd0,Opnd1)):
        $.prim = etq
        etiquetado-opnds(Opnd0,Opnd1)
        etq++
        $.sig = etq
```

```
etiquetado(exp_resta(Opnd0,Opnd1)):
    $.prim = etq
    etiquetado-opnds(Opnd0,Opnd1)
    etq++
    $.sig = etq

etiquetado(exp_mult(Opnd0,Opnd1)):
    $.prim = etq
    etiquetado-opnds(Opnd0,Opnd1)
    etq++
    $.sig = etq

etiquetado(exp_div(Opnd0,Opnd1)):
    $.prim = etq
    etiquetado-opnds(Opnd0,Opnd1)
    etq++
    $.sig = etq

etiquetado(exp_mod(Opnd0,Opnd1)):
    $.prim = etq
    etiquetado-opnds(Opnd0,Opnd1)
    etq++
    $.sig = etq

etiquetado(exp_and(Opnd0,Opnd1)):
    $.prim = etq
    etiquetado-opnds(Opnd0,Opnd1)
    etq++
    $.sig = etq

etiquetado(exp_or(Opnd0,Opnd1)):
    $.prim = etq
    etiquetado-opnds(Opnd0,Opnd1)
    etq++
    $.sig = etq

etiquetado(exp_menos(Exp)):
    $.prim = etq
    etiquetado(Exp)
    etiquetado-acc-val(Exp)
    $.sig = etq

etiquetado(exp_not(Exp)):
    $.prim = etq
    etiquetado(Exp)
    etiquetado-acc-val(Exp)
    etq +=2
    $.sig = etq

etiquetado(inst_index(Opnd0,Opnd1)):
    $.prim = etq
    etiquetado-opnds(Opnd0,Opnd1)
    etq++
    $.sig = etq

etiquetado(exp_reg(Exp,id)):
    $.prim = etq
    etq++
    etiquetado(Exp)
    etq++
```

```
        $.sig = etq


    etiquetado(exp_ind(Exp)):
        $.prim = etq
        etiquetado(Exp)
        etq++
        $.sig = etq

    etiquetado(exp_true()): noop

    etiquetado(exp_false()): noop

    etiquetado(exp_litEnt(litEnt)):
        $.prim = etq
        etq++
        $.sig = etq

    etiquetado(exp_litReal(litReal)):
        $.prim = etq
        etq++
        $.sig = etq

    etiquetado(exp_litCad(litCad)):
        $.prim = etq
        etq++
        $.sig = etq

    etiquetado(exp_iden(id)):
        $.prim = etq
        if $.nivel = 0 then
            etq++
        else
            etq += 3
        end if
        $.sig = etq

    etiquetado(exp_null()): noop


    etiquetado-cod-opns(Opnd0,Opnd1):
        etiquetado(Opnd0)
        etiquetado-acc-val(Opnd0)
        etiquetado(Opnd1)
        etiquetado-acc-val(Opnd1)

    etiquetado-acc-val(Exp):
        if es-designador(Exp) then
        etq++
        end if

    etiquetado-paso-param(dec_proc(id,Param,Decs,Is),PReales):
        etq += 2
        etiquetado(PReales)
        etq++

        recolecta_procs(si_decs(Decs)):
        recolecta_procs(Decs)

    recolecta_procs(muchas_decs(Decs, Dec)):
```

```
        recolecta_procs(muchas_decs(Decs))
        recolecta_procs(Dec)

    recolecta_procs(una_dec(Dec)):
        recolecta_procs(Dec)

    recolecta_procs(dec_simple(Tipo,id)): noop

    recolecta_procs(dec_type(Tipo,id)): noop

    recolecta_procs(dec_proc(id,PFml,Decs,Is)):
        emit apila(sub_pendientes,$)
```

6 | Especificación del procesamiento de generación de código

```
var sub_pendientes = pila_vacia()

gen_cod(prog(Blo)):
    gen_cod(Blo)

gen_cod(bloq(Decs,Insts)):
    recolecta_procs(Decs)
    gen_cod(Insts)
    emit stop()
    while not es_vacia(sub_pendientes)
        sub = cima(sub_pendientes)
        desapila(sub_pendientes)
        let sub = dec_proc(id,PFml,Decs,Is) in
            emit desapilad(sub.nivel)
            recolecta_procs(Decs)
            gen_cod(Is)
            emit desactiva(sub.nivel, sub.tam)
            emid ir_ind()
        end let
    end while

gen_cod(si_inst(LInst)):
    gen_cod(LInst):

gen_cod(no_inst()): noop

gen_cod(muchas_inst(LInst,Inst)):
    gen_cod(LInst)
    gen_cod(Inst)

gen_cod(una_inst(Inst)):
    gen_cod(Inst)

gen_cod(si_preales(LPReal)):
    gen_cod(LPReal)

gen_cod(no_preales()): noop

gen_cod(muchas_exp(LPReal,Exp)):
    gen_cod(LPReal)
    gen_cod(Exp)

gen_cod(una_exp(Exp)):
    gen_cod(Exp)

gen_cod(inst_eval(Exp)):
    gen_cod(Exp)
    gen_acc_val(Exp)

gen_cod(inst_if(Exp,Blo)):
    gen_cod(Exp)
    gen_acc_val(Exp)
    emit ir_f($.sig)
```

```
    gen_cod(Blo)

gen_cod(inst_else(Exp,Blo1,Blo2)):
    gen_cod(Exp)
    gen_acc_val(Exp)
    emit ir_v($.prim)
    gen_cod(Blo1)
    emit ir_f($.sig)
    gen_cod(Blo2)

gen_cod(inst_while(Exp,Blo)):
    gen_cod(Exp)
    gen_acc_val(Exp)
    emit ir_f($.sig)
    gen_cod(Blo)
    emit ir_a($.prim)

gen_cod(inst_new(Exp)):
    gen_cod(Exp)
    let ref!(Exp.tipo) = tipo_punt(T) in
        emit alloc(T.tam)
    end
    emit desapila_ind()

gen_cod(inst_delete(Exp)):
    gen_cod(Exp)
    emit apila_ind()
    let ref!(Exp.tipo) = tipo_punt(T) in
        emit dealloc(T.tam)
    end

gen_cod(inst_read(Exp)):
    gen_cod(Exp)
    emit desapila_ind()

gen_cod(inst_write(Exp)):
    gen_cod(Exp)
    gen_acc_val(Exp)
    emit write()

gen_cod(inst_call(id,PReales)):
    emit activa($.vinculo.nivel,$.vinculo.tam,$.sig)
    allocPFmls($.vinculo.pformal,$.prim)
    emit ir_a($.vinculo.prim)
    deallocPFmls($.vinculo.pformal)

gen_cod(inst_nl()):
    emit nl
    emit write

gen_cod(inst_blo(Blo)):
    gen_cod(Blo)

gen_cod(exp_asig(Opnd0,Opnd1)):
    gen_cod(Opnd0)
    gen_cod(Opnd1)
    if es_designador(E) then
        emit copia(E.tipo.tam)
    else
        emit desapila_ind()
```

```
    end

gen_cod(exp_menor(Opnd0,Opnd1)):
    gen_cod(Opnd0)
    gen_acc_val(Opnd0)
    gen_cod(Opnd1)
    gen_acc_val(Opnd1)
    emit menor

gen_cod(exp_menIgual(Opnd0,Opnd1)):
    gen_cod(Opnd0)
    gen_acc_val(Opnd0)
    gen_cod(Opnd1)
    gen_acc_val(Opnd1)
    emit menIgual

gen_cod(exp_mayor(Opnd0,Opnd1)):
    gen_cod(Opnd0)
    gen_acc_val(Opnd0)
    gen_cod(Opnd1)
    gen_acc_val(Opnd1)
    emit mayor

gen_cod(exp_mayIgual(Opnd0,Opnd1)):
    gen_cod(Opnd0)
    gen_acc_val(Opnd0)
    gen_cod(Opnd1)
    gen_acc_val(Opnd1)
    emit mayIgual

gen_cod(exp_igual(Opnd0,Opnd1)):
    gen_cod(Opnd0)
    gen_acc_val(Opnd0)
    gen_cod(Opnd1)
    gen_acc_val(Opnd1)
    emit igual

gen_cod(exp_dist(Opnd0,Opnd1)):
    gen_cod(Opnd0)
    gen_acc_val(Opnd0)
    gen_cod(Opnd1)
    gen_acc_val(Opnd1)
    emit distinto

gen_cod(exp_sum(Opnd0,Opnd1)):
    gen_cod(Opnd0)
    gen_acc_val(Opnd0)
    if Opnd0.tipo == tipoReal then
        casteo(Opnd0)
    gen_cod(Opnd1)
    gen_acc_val(Opnd1)
    if Opnd1.tipo == tipoReal then
        casteo(Opnd1)
    emit suma

gen_cod(exp_resta(Opnd0,Opnd1)):
    gen_cod(Opnd0)
    gen_acc_val(Opnd0)
    if Opnd0.tipo == tipoReal then
        casteo(Opnd0)
```

```
        gen_cod(Opnd1)
        gen_acc_val(Opnd1)
        if Opnd1.tipo == tipoReal then
            casteo(Opnd1)
        emit resta

    gen_cod(exp_mult(Opnd0,Opnd1)):
        gen_cod(Opnd0)
        gen_acc_val(Opnd0)
        if Opnd0.tipo == tipoReal then
            casteo(Opnd0)
        gen_cod(Opnd1)
        gen_acc_val(Opnd1)
        if Opnd1.tipo == tipoReal then
            casteo(Opnd1)
        emit mult

    gen_cod(exp_div(Opnd0,Opnd1)):
        gen_cod(Opnd0)
        gen_acc_val(Opnd0)
        if Opnd0.tipo == tipoReal then
            casteo(Opnd0)
        gen_cod(Opnd1)
        gen_acc_val(Opnd1)
        if Opnd1.tipo == tipoReal then
            casteo(Opnd1)
        emit div

    gen_cod(exp_mod(Opnd0,Opnd1)):
        gen_cod(Opnd0)
        gen_acc_val(Opnd0)
        gen_cod(Opnd1)
        gen_acc_val(Opnd1)
        emit mod

    gen_cod(exp_and(Opnd0,Opnd1)):
        gen_cod(Opnd0)
        gen_acc_val(Opnd0)
        gen_cod(Opnd1)
        gen_acc_val(Opnd1)
        emit or

    gen_cod(exp_or(Opnd0,Opnd1)):
        gen_cod(Opnd0)
        gen_acc_val(Opnd0)
        gen_cod(Opnd1)
        gen_acc_val(Opnd1)
        emit or

    gen_cod(exp_menos(Exp)):
        gen_cod(Exp)
        gen_acc_val(Exp)
        emit menos

    gen_cod(exp_not(Exp)):
        gen_cod(Exp)
        gen_acc_val(Exp)
        emit not

    gen_cod(inst_index(Opnd0,Opnd1)):
```

```
        gen_cod(Opnd0)
        gen_cod(Opnd1)
        gen_acc_val(Opnd1)
        let ref!(Opnd1.tipo) = array(T,d) in
            emit apilaInt(T.tam)
        end let
        emit mult
        emit suma

    gen_cod(exp_reg(Exp,id)):
        gen_cod(Exp)
        let ref!(E.tipo) = struct(Cs) in
            emit apilaInt(desplazamiento(Cs,c))
        end let
        emit suma

    gen_cod(exp_ind(Exp)):
        gen_cod(Exp)
        emit apilaInd()

    gen_cod(exp_true()):
        emit apilaBool("true")

    gen_cod(exp_false()):
        emit apilaBool("false")

    gen_cod(exp_litEnt(litEnt)):
        emit apilaInt(litEnt)

    gen_cod(exp_litReal(litReal)):
        emit apilaReal(litReal)

    gen_cod(exp_litCad(litCad)):
        emit apilaString(litCad)

    gen_cod(exp_iden(id)):
        if $.vinculo == Pformal_ref then
            emit apilad($.vinculo.nivel)
            emit apila_int($.vinculo.dir)
            emit suma()
            emit apila_ind()
        else if $.vinculo() == Pformal_noref then
            emit apilad($.vinculo.nivel)
            emit apila_int($.vinculo.dir)
            emit suma()
        else if $.vinculo() == Dec_simple then
            if $.vinculo().nivel() == 0 then
                emit apila_int($.vinculo.dir)
            else
                emit apilad($.vinculo.nivel)
                emit apila_int($.vinculo.dir)
                emit suma()
        emit apilaInt($.viculo.dir)

    gen_cod(exp_null()): noop

    gen_acc_val(Exp):
        if es_designador(ref!(E)) then
            emit apilaInd()
        end if
```

```
allocPFmls(PFmls, PReales):
    if PFmls == Si_pformal then
        allocLPFml(PFmls.lpfml, PReales.lpr)

alloLPFml(LPFml, LPReal):
    if LPFml == Muchos_pformal then
        allocLPFml(LPFml.lpfml, LPReal.lpr)
    allocPFml(LPFml.pfml, LPReal.exp)

allocPFml(PFml, Exp):
    if PFml.tipo == Pformal_ref then
        emit alloc(TipoEnt.tam)
        gen_cod(Exp)
        emit desapila_ind()
    else
        emit alloc(PFml.tipo.tam)
        gen_cod(Exp)
        if PFml.tipo == Tipo_real ^ e.tipo == Tipo_int then
            gen_acc_val(Exp)
            casteo(Exp)
            emit desapila_ind()
        else
            if esDesignador(Exp) then
                emit copia(PFml.tipo.tam)
            else
                emit desapila_ind()

deallocPFmls (PFmls):
            if PFmls.tipo == Si_pformal then
                    deallocLPFml(PFmls.lpfml)

    deallocLPFml(LPFml):
            if LPFml.tipo == Muchos_pformal then
                    deallocLPFml(LPFml.lpfml)
            deallocPFml(LPFml.pfml)

    deallocPFml(PFml):
            if PFml == Pformal_ref then
                    emit dealloc(TipoEnt.tam)
            else
            emit dealloc(PFml.tipo.tam)

recolecta_procs(si_decs(Decs)):
    recolecta_procs(Decs)

recolecta_procs(muchas_decs(Decs, Dec)):
    recolecta_procs(muchas_decs(Decs))
    recolecta_procs(Dec)

recolecta_procs(una_dec(Dec)):
    recolecta_procs(Dec)

recolecta_procs(dec_simple(Tipo,id)): noop

recolecta_procs(dec_type(Tipo,id)): noop

recolecta_procs(dec_proc(id,PFml,Decs,Is)):
    emit apila(sub_pendientes,$)
```

Índice de figuras