

---

# Lenguajes de Programación y Procesadores de Lenguajes

(2º parcial)

13 de enero de 2016

---

1. Dado el siguiente programa C:

```
-----
float f (float p, float q)
{ float i=1.0;
  if (p > q) return f(p-i, q);
  else return p+q;          // <--- A
}
int main ()
{ int i=0; float x, y = 2.0;
  { int i=2;
    x=(float) i + y;          // <--- B
  }
  x = f(x, y);
}
-----
```

- a) (0,75 ptos.) Considerando que la talla de enteros es 2 y la de los reales es 4, y que la talla del segmento de enlaces de control es 8, mostrad el contenido completo de la TDS en los puntos de control **A** y **B**.
- b) (0,5 ptos.) Indicad el nivel y el desplazamiento relativo de las variables  $x$ ,  $i$  e  $y$  que aparecen en la instrucción del punto de control **B**.
- c) (0,75 ptos.) Mostrad el contenido (en términos de Registros de Activación) del estado de la pila de ejecución cada vez que se pasa por el punto de control **A** (antes del **return**).
2. (1,5 ptos.) Contestad brevemente a las siguientes cuestiones:
- a) Para la clase de Lenguajes con Estructura de Bloques, y desde el punto de vista del acceso a los objetos en los Registros de Activación de las funciones en la pila, ¿qué diferencias hay entre los lenguajes que permiten el anidamiento de funciones (p.ej. **PASCAL**) y los que no (p.ej. **C**)?
- b) Considerando que se ha completado la fase de declaración de los objetos, diseñad un ETDS para la comprobación de tipos y la generación de código intermedio para la regla:

$$E \rightarrow \text{id} [ E ]$$

- c) Describid en que consiste la optimización de código intermedio denominada “*Cálculo previo de constantes*”, cómo se detecta y como se aplicaría al siguiente segmento de código:

(100)	$t_1 = 4$	(102)	$t_2 = 10$	(104)	$t_4 = 1$	(106)	$t_6 = t_3 + t_5$
(101)	$x = t_1$	(103)	$t_3 = t_2 * x$	(105)	$t_5 = t_4 * y$	(107)	$x = t_6$

3. (3.5 ptos.) Diseñad un ETDS que genere código intermedio para el siguiente fragmento de una gramática:

$$\begin{array}{ll} I \rightarrow \text{for } id \text{ in range } ( E^1, E^2 ) : I & I \rightarrow I ; I \\ I \rightarrow \text{break} & I \rightarrow id := E \end{array}$$

Donde la regla del `for` representa un bucle que ejecutara las instrucciones de  $I^1$  asignando a la variable `id` en cada iteración los valores enteros comprendidos entre  $E^1$  y  $E^2$ . La ejecución de la instrucción `break` supone la salida inmediata del bucle.

4. (1 pto.) Dado el siguiente fragmento de código intermedio de un bloque básico, aplicad las optimizaciones locales a partir de su GDA. A la salida del bloque solo estarán activas las variables: `a`, `x`.

(100) $t_1 := 0$	(107) $t_7 := t_5 + t_6$	(114) $t_{14} := t_{13} + 10$
(101) $t_2 := x + t_1$	(108) $t_8 := t_7 * N$	(115) $t_{15} := t_{14} * N$
(102) $x := t_2$	(109) $t_9 := z$	(116) $t_{16} := x$
(103) $t_3 := 5$	(110) $t_{10} := t_8 + t_9$	(117) $t_{17} := t_{15} + t_{16}$
(104) $t_4 := x + t_3$	(111) $t_{11} := a[t_{10}]$	(118) $a[t_{17}] := t_{11}$
(105) $t_5 := t_4 * N$	(112) $t_{12} := t_4$	
(106) $t_6 := 10$	(113) $t_{13} := t_{12} * N$	

5. Dado el siguiente fragmento de código intermedio:

(101) $c := ini + 5$	(106) $t_5 := t_3 + t_4$	(111) $c := c + 10$
(102) $t_1 := k * 2$	(107) $a[t_5] := 0$	(112) $k := k + 2$
(103) $t_2 := t_1 + 3$	(108) <b>if</b> $c > 50$ <b>goto</b> 111	(113) <b>if</b> $k \leq 100$ <b>goto</b> 102
(104) $t_3 := t_2 * 4$	(109) $c := c + 5$	
(105) $t_4 := N * 2$	(110) <b>goto</b> 112	

- a) (0.5 ptos.) Determinad los bloques básicos que forman el bucle. Extraed el código invariante e indicad las variables de inducción y sus ternas asociadas.
- b) (0.75 ptos.) Aplicad el algoritmo de reducción de intensidad.
- c) (0.75 ptos.) Aplicad el algoritmo de eliminación de variables de inducción.

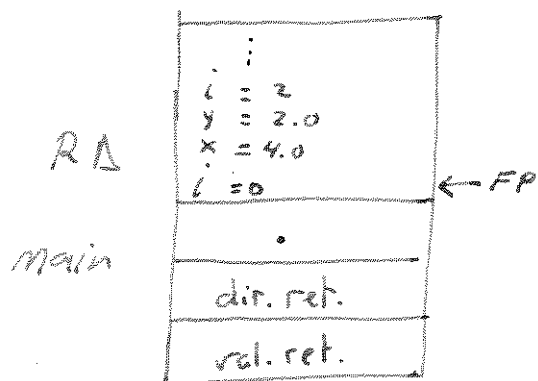
1.a)

nombre	n	s	tipo	
f	"función"	0	-	tfunción $\boxed{\rightarrow (treal, treal) \rightarrow treal}$
p	"parametro"	1	-12	treal $\boxed{\rightarrow}$
q	"parametro"	1	-16	treal $\boxed{\rightarrow}$ $\Leftarrow$ Pto. A
i	"var. local"	1	0	treal $\boxed{\rightarrow}$
<hr/>				
f	"función"	0	-	tfunción $\boxed{\rightarrow (treal, treal) \rightarrow treal}$
main	"	0	-	tfunción $\boxed{\rightarrow (tvacio) \rightarrow tentero}$
i	"var. local"	1	0	tentero $\boxed{\rightarrow}$
x	"	1	2	treal $\boxed{\rightarrow}$ $\Leftarrow$ Pto. B
y	"	1	6	treal $\boxed{\rightarrow}$
i	"	2	10	tentero $\boxed{\rightarrow}$

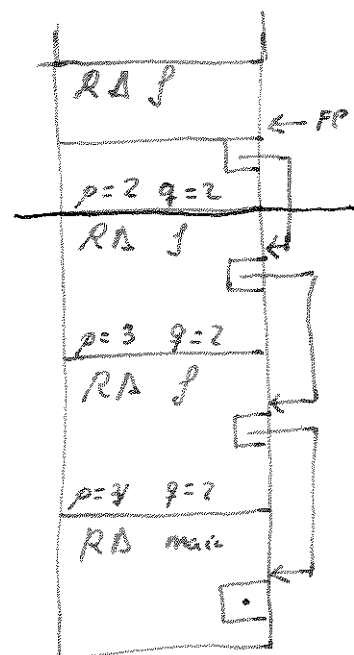
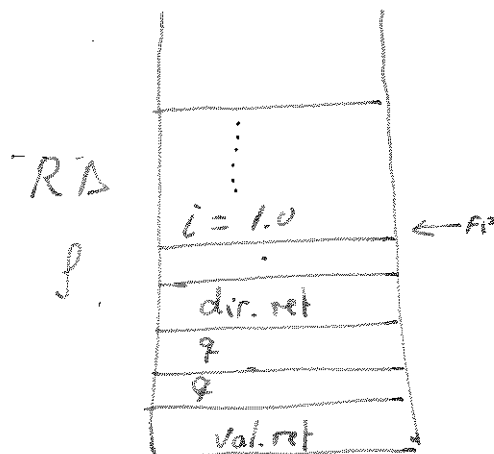
2.b)

(1, 2) (2, 10) (1, 6)

$$x = (\text{float}) i + y$$



2.c)



2.a) Lenguajes SIN anidamiento de funciones. - El acceso a gestiona con un solo registro (FP "Frame Pointer"). Para un cierto objeto  $x$ , se supone conocido (en la TDS) el desplazamiento relativo  $\delta_x$ , su dirección física será:

$$d_x = FP + \delta_x$$

Lenguajes con anidamiento de funciones. - El acceso a gestiona con una estructura (display) indexada por el nivel de anidamiento. Para un cierto objeto  $x$ , se supone conocido (en la TDS) el nivel de anidamiento  $n_x$  y su desplazamiento relativo  $\delta_x$ , su dirección física será:

$$d_x = display[n_x] + \delta_x$$

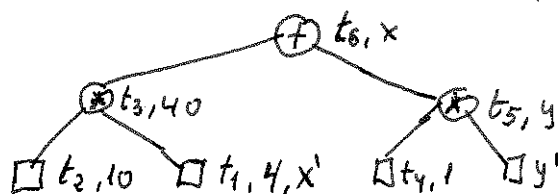
2.b)

$E \rightarrow id [E] \quad \underline{p_i} \mid [obtenerTds(id.nom, id.t, id.pos) \wedge (id.t = tvector(id.nel, E.t)) \wedge (E'.t = tvector)] \mid \{ E.t = tenor; \text{NewError}(); \}$

Emite ( $E'.pos \leftarrow E'.pos + talla(E.t)$ );  $E.pos \leftarrow \text{CreaVarTemp}(E.t)$ ;

Emite ( $E.pos \leftarrow id.pos [E'.pos]$ );

2.c) Evaluación en tiempo de compilación de las operaciones donde los ~~datos~~ operandos sean constantes o se permita alguna simplificación algebraica.

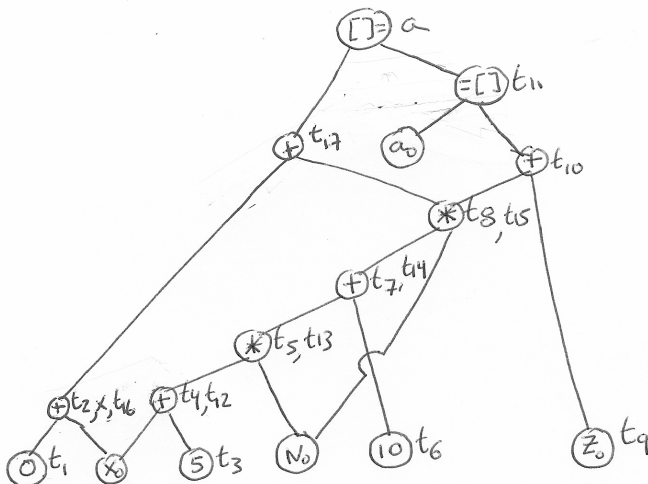


$$\Rightarrow x \leftarrow 40 + y$$

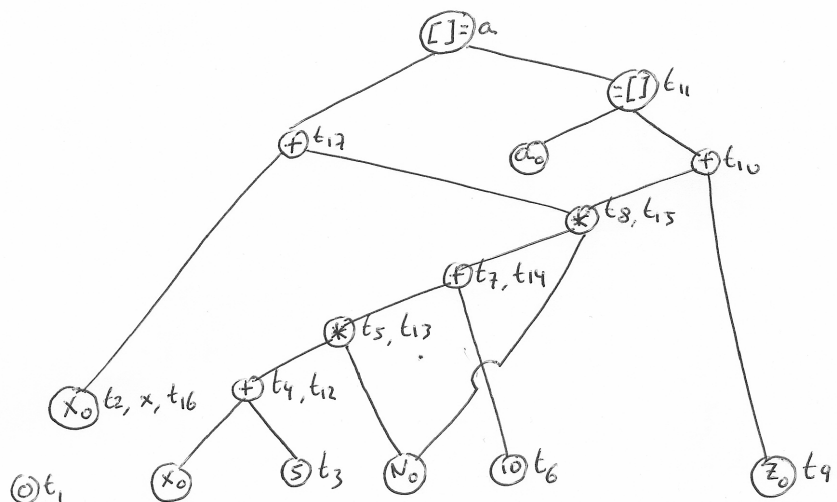
3.-

$I \rightarrow \text{for id in range}(E_1, E_2):$  $I_1$	<pre>{ id.pos = BuscaPos(id.nom);   emite(id.pos ':' E1.pos);   cond = SIGINST ;   fin = CreaLans(SIGINST);   emite( 'if' id.pos '&gt;' E2.pos 'goto' --);  {  emite( id.pos ':' id.pos '+ 1');   emite( 'goto' cond);   CompletaLans (fin, SIGINST);   CompletaLans (I1.salir, SIGINST);   I.salir = NULL ; }</pre>
$I \rightarrow I_1 ; I_2$	<pre>{ I.salir = FusionaLans( I1.salir, I2.salir); }</pre>
$I \rightarrow \text{break}$	<pre>{ I.salir = CreaLans(SIGINST);   emite ('goto' --); }</pre>
$I \rightarrow \text{id} := E$	<pre>{ id.pos = BuscaPos(id.nom);   emite (id.pos ':' E.pos);   I.pos = id.pos ;   I.salir = NULL ; }</pre>

4.-

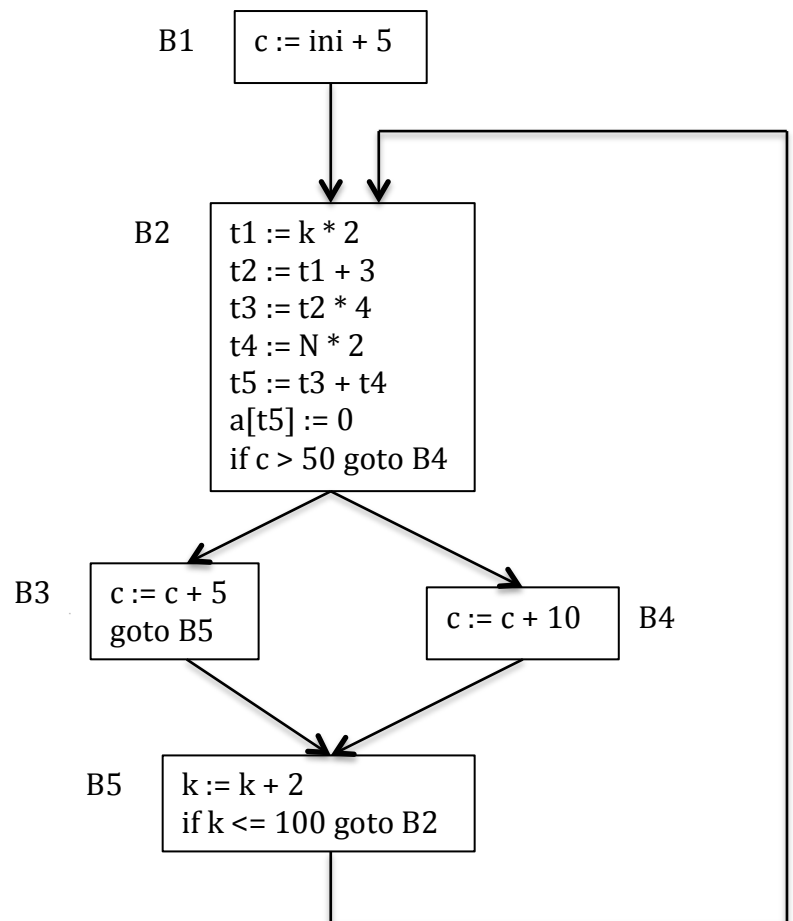


```
t4 := x + 5
t5 := t4 * N
t7 := t5 + 10
t8 := t7 * N
t10 := t8 + z
t11 := a [t10]
t17 := x + t8
a[t17] := t11
goto 1
```



5.-

```
(101) c := ini + 5
(102) t1 := k * 2
(103) t2 := t1 + 3
(104) t3 := t2 * 4
(105) t4 := N * 2
(106) t5 := t3 + t4
(107) a[t5] := 0
(108) if c > 50 goto 111
(109) c := c + 5
(110) goto 112
(111) c := c + 10
(112) k := k + 2
(113) if k <= 100 goto 102
```

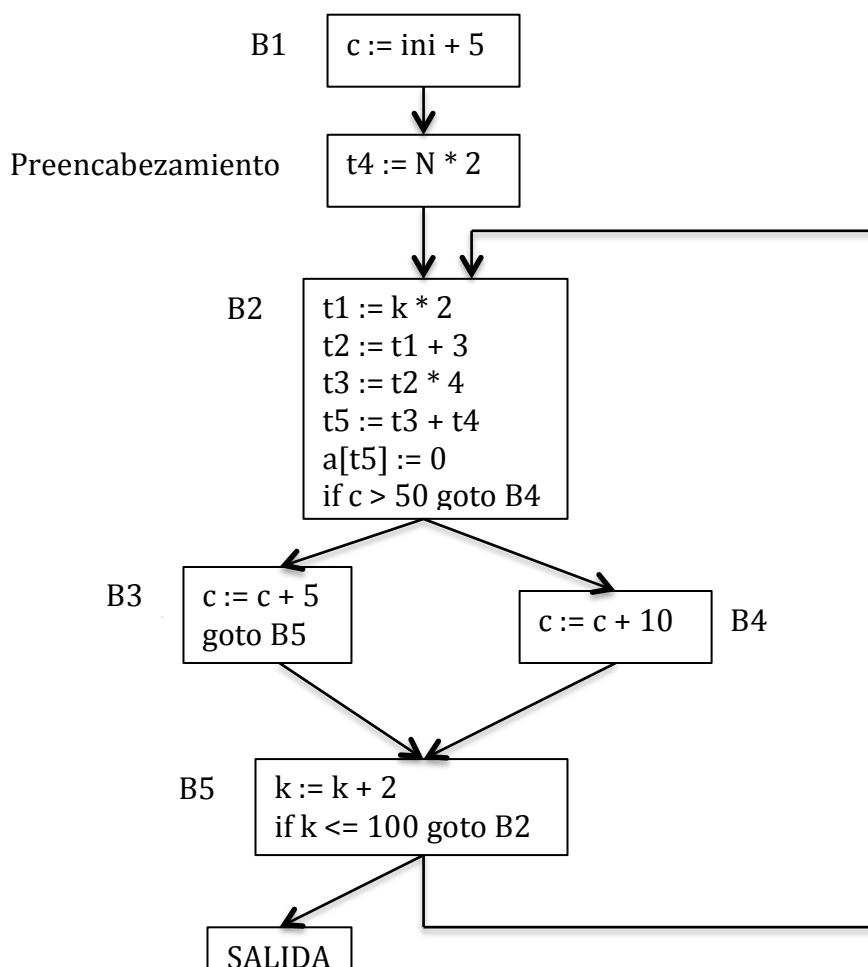


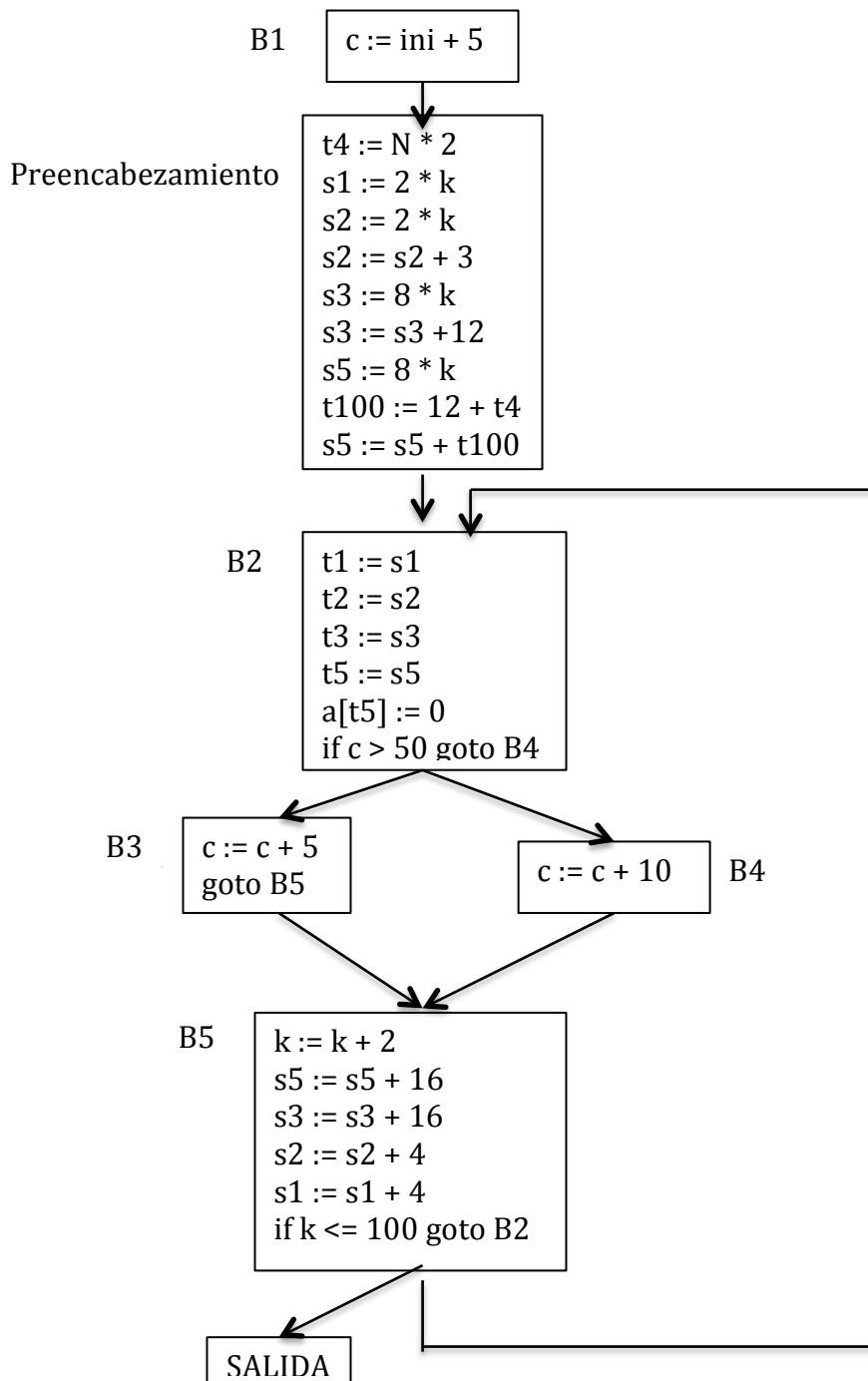
Arista de retroceso: B5->B2

Bucle natural: B2, B3, B4 y B5

Código invariante:  $t4 := N * 2$

Variables de Inducción:  $k(k,1,0)$  ;  $t1(k,2,0)$  ;  $t2(k,2,3)$  ;  $t3(k,8,12)$  ;  $t5(k,8,12+t4)$





Variables de Inducción:  $k (k,1,0)$  ;  $t1 (k,2,0)$ ;  $t2 (k,2,3)$ ;  $t3 (k,8,12)$ ;  $t5(k,8,12+t4)$

$s5 = s5 + 2 * 8$   
 $s3 = s3 + 2 * 8$   
 $s2 = s2 + 2 * 2$   
 $s1 = s1 + 2 * 2$

Suponiendo que k no está activa a la salida del bucle. Si está activa, no se eliminará.

