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:

$$\mathrm{E} \rightarrow \mathrm{id} \; [\; \mathrm{E} \;]$$

   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:

I	→	`for` *id* `in range` (E , E) : I		I	→	I ; I
I	→	`break`		I	→	*id* := E

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)	`if` $c > 50$ `goto` 111	(113)	`if` $k \leq 100$ `goto` 102
(104)	$t_3 := t_2 * 4$	(109)	$c := c + 5$		
(105)	$t_4 := N * 2$	(110)	`goto` 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.

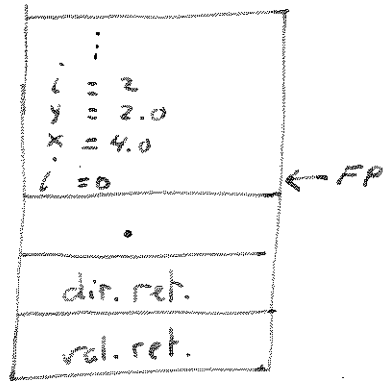
1. a)

nombre		n	δ	tipo	
f	"función"	0	–	tfunción	→ (treal, treal) → treal
p	"parámetro"	1	-12	treal	
q	"parámetro"	1	-16	treal	← Pto. A
i	"var. local"	1	0	treal	
f	"función"	0	–	tfunción	→ (treal, treal) → treal
main	"	0	–	tfunción	→ (tvacio) → tentero
i	"var. local"	1	0	tentero	
x	"	1	2	treal	← Pto. B
y	"	1	6	treal	
i	"	2	10	tentero	

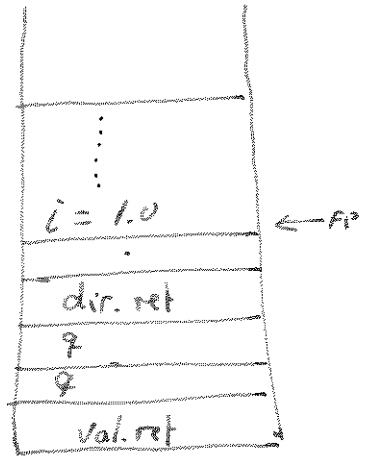
2. b)

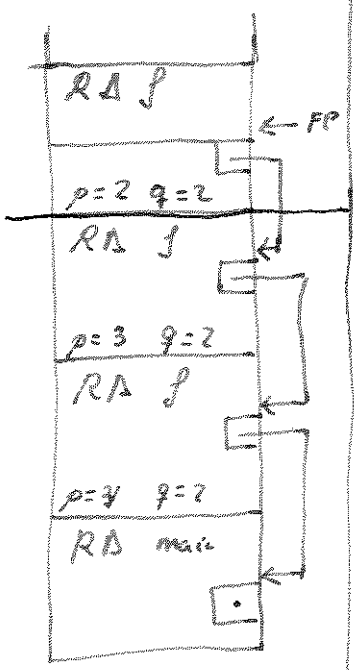
(1,2) (2,10) (1,6)

x = (float) i + y



2. c)





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

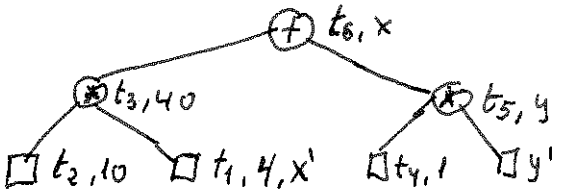
Lenguajes CON anidamiento de funciones. - El acceso a cualquier variable se hace con una estructura (display) indexada por el nivel de anidamiento. Para un cierto objeto x, se supone conocido (en la TDS) el nivel de anidamiento n_x y su desplazamiento relativo δx, su dirección física será:

$d_x = display[n_x] + \delta x$

2.b)

E → id [E] | Si] [obtenerTds(id.nom, id.t, id.pos) ^ (id.t = tVector(id.nel, E.t)) ^ (E'.t = tEntero)] } E.t = tError; MenuError();}
Emite(E'.pos ← E'.pos * talla(E.t)); E.pos ← CreaVarTemp(E.t);
Emite(E.pos ← id.pos[E'.pos]);

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



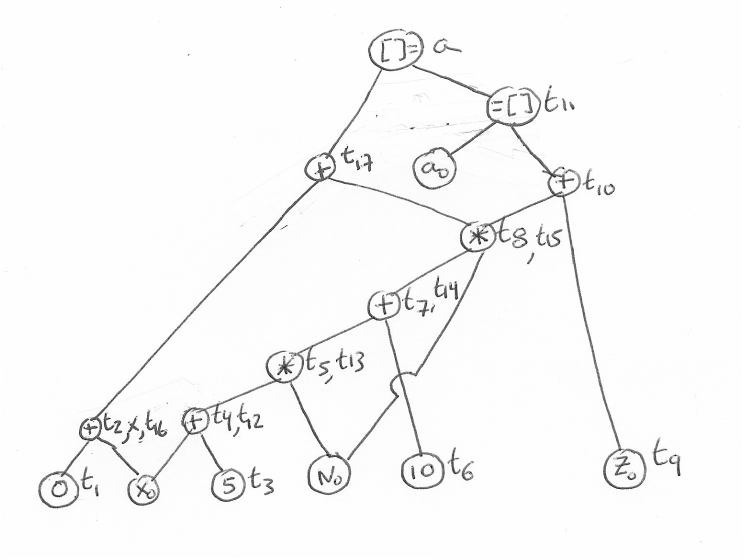
$\Rightarrow$ x ← 40 + y

3.-

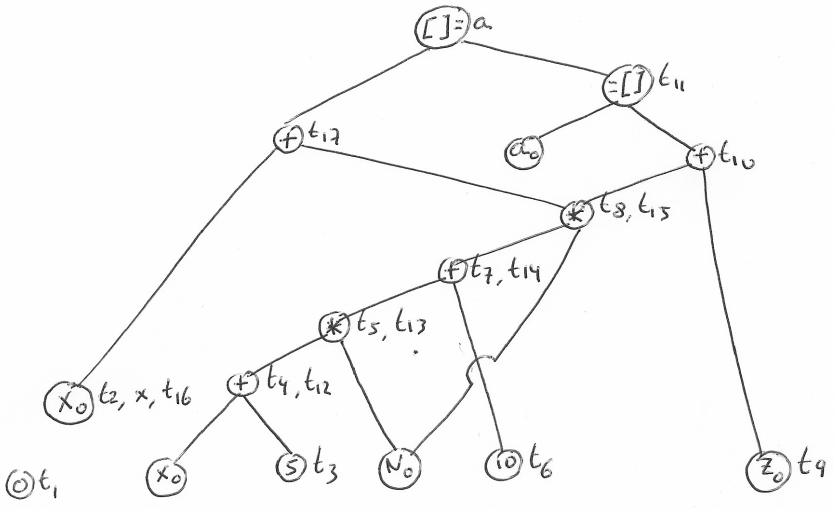
I -> **for id in range** (E_1, E_2):	{ id.pos = BuscaPos(id.nom);
	emite(id.pos ':=' E_1.pos);
	cond = SIGINST ;
	fin = CreaLans(SIGINST);
	emite('if' id.pos '>' E_2.pos 'goto' --);

I_1	{ emite(id.pos ':=' id.pos '+ 1'); emite('goto' cond); CompletaLans (fin, SIGINST); CompletaLans (I_1.salir, SIGINST); I.salir = NULL ; }
I -> I_1 ; I_2	{ I.salir = FusionaLans(I_1.salir, I_2.salir); }
I -> **break**	{ I.salir = CreaLans(SIGINST); emite ('goto' --); }
I -> **id :=** E	{ id.pos = BuscaPos(id.nom); emite (id.pos ':=' E.pos); I.pos = id.pos ; I.salir = NULL ; }

4.-

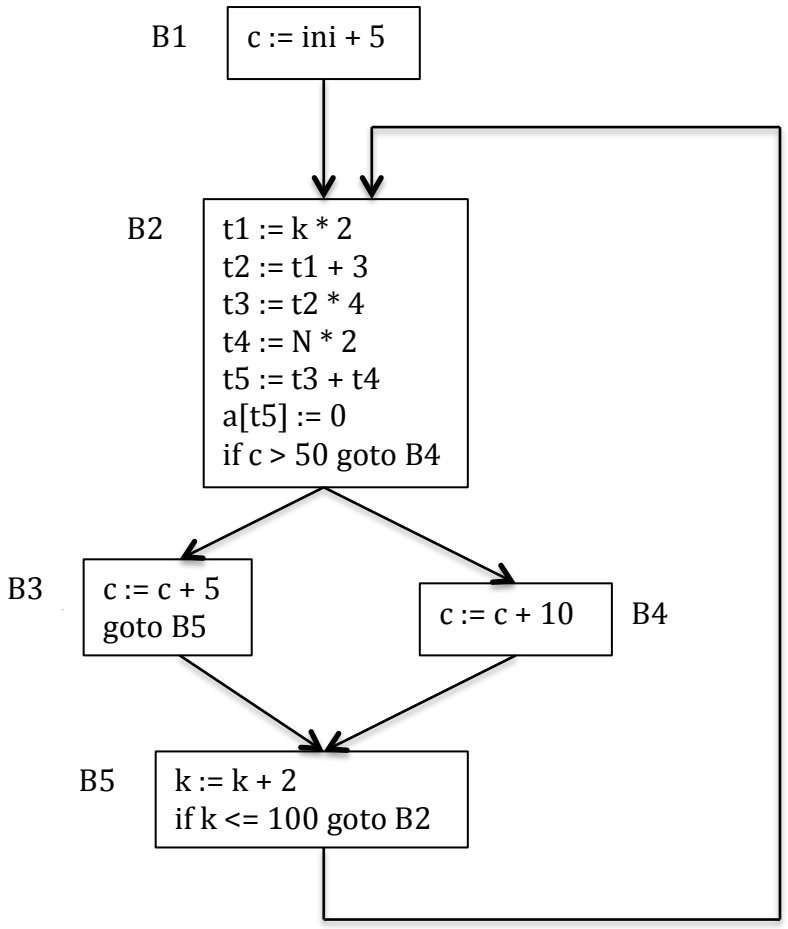


t4 := x + 5
t5 := t4 * N
t7 := t5 + 10
t8 := t7 * N
t10 := t8 + z
t11 := a [t10]
t17 := x + t8
a[t17] := t11
goto 1

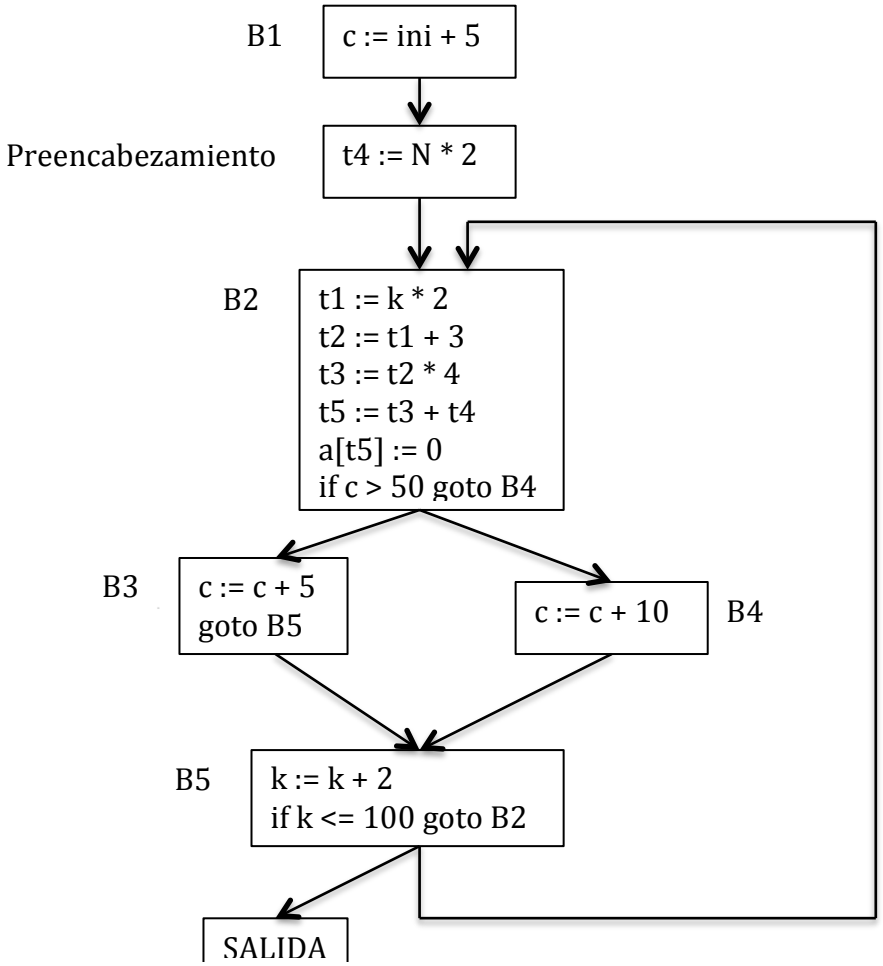


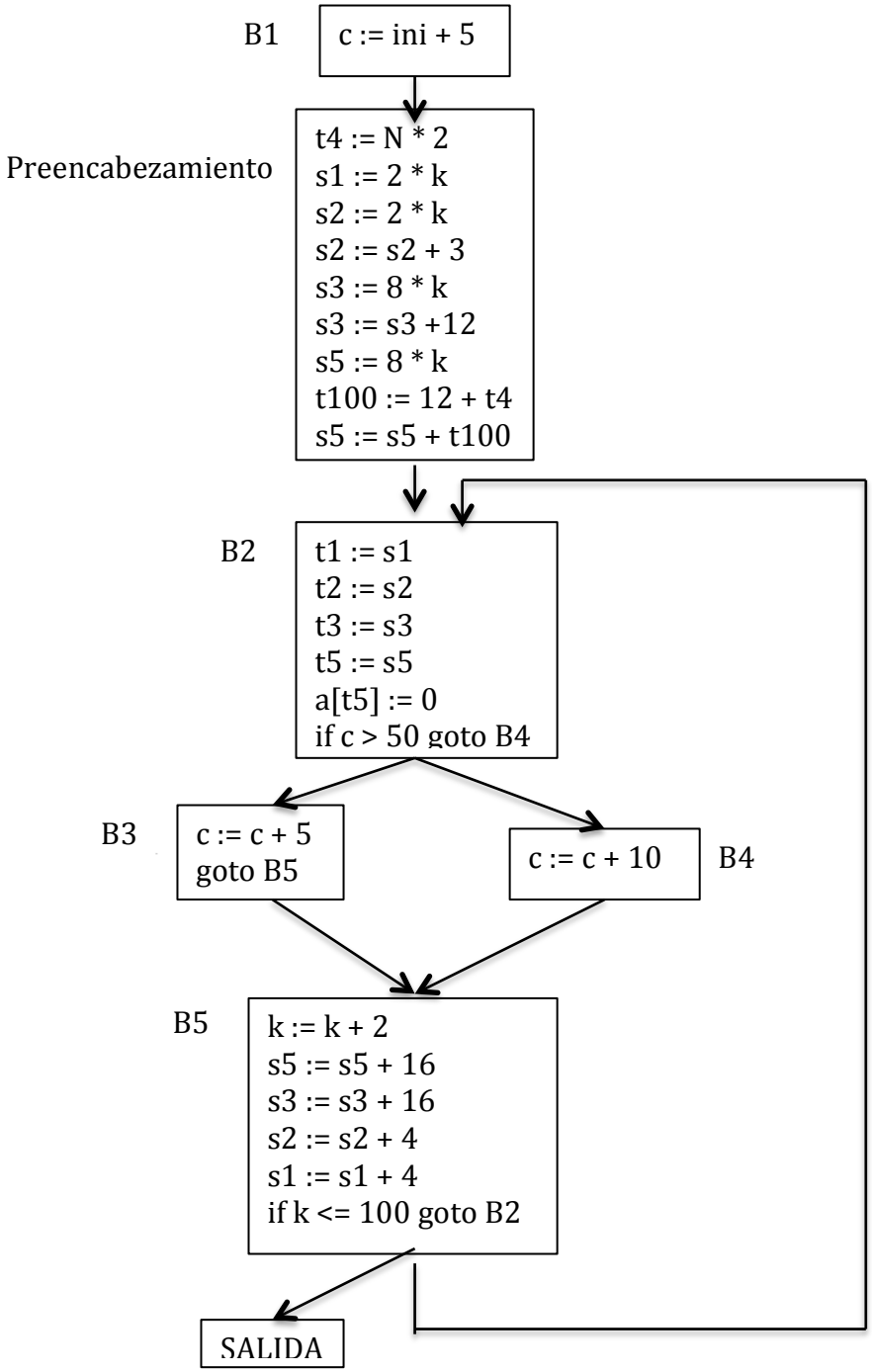
5.-

(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)




B1
c := ini + 5
Preencabezamiento
t4 := N * 2
s1 := 2 * k
s2 := 2 * k
s2 := s2 + 3
s3 := 8 * k
s3 := s3 +12
s5 := 8 * k
t100 := 12 + t4
s5 := s5 + t100
B2
t1 := s1
t2 := s2
t3 := s3
t5 := s5
a[t5] := 0
if c > 50 goto B4
B3
c := c + 5
goto B5
c := c + 10
B4
B5
k := k + 2
s5 := s5 + 16
s3 := s3 + 16
s2 := s2 + 4
s1 := s1 + 4
if k <= 100 goto B2
SALIDA

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