Lenguajes de Programación y Procesadores de Lenguajes

(2º parcial) **13 de enero de 2014**

1. Dado el siguiente programa C:

```
----------------------------------------
float f1(int x, int y)
{
  if (x>y) return f1(x-1,y)
  else return x+y;                <=== A
}
void main ()
{ int a = 3; float c; int b = 2;
  { int a = 6;
    c = a / 2 * b;                <=== B
  }
  { int a = 2; float b = 1.0
    c = a / 2 * b;                <=== C
  }
  c = f1(a, b);
}
----------------------------------------
```

   a) (0,75 ptos.) Suponiendo que la talla de enteros es 2 y la de los reales es 4, mostrad el contenido completo de la TDS en el punto de control **C**.

   b) (0,5 ptos.) Indicad el desplazamiento relativo de las 4 variables de la instrucción `c = a/2 * b` en los dos puntos de control: **B, C**.

   c) (0,75 ptos.) Mostrad el contenido (en términos de Registros de Activación) del estado de la pila de ejecución en el punto de control **A** (antes del `return`).

2. (1,5 ptos.) Contestad brevemente a las siguientes cuestiones:

   a) ¿Qué acciones se deben realizar para comprobar que el tipo de los parámetros actuales de un llamada a una función coincide con el de los parámetros formales?

   b) ¿Cómo y dónde se reserva espacio para el valor de retorno de una función?

   c) ¿Cómo y dónde se reserva espacio en un Registro de Activación para las variables temporales?

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

$$\begin{array}{lcl} \text{I} & \rightarrow & \text{dependon E1 execute E2 times \{ B} \\ \text{B} & \rightarrow & \text{num : LI ; B} \;\mid\; \} \\ \text{LI} & \rightarrow & \text{I ; LI} \;\mid\; \epsilon \end{array}$$

   Donde la instrucción *dependon* ejecuta un bucle el número de veces indicado por $E2$. En cada iteración del bucle se ejecutarán todos los bloques de instrucciones precedidos del número cuyo valor coincide con el de la expresión $E1$. $E1$ y $E2$ no se modifican en el bucle.

Ejemplo La salida de este ejemplo será: `e e e`

dependon 5 execute 3 times {
  2: print(b);
  5: print(e);
  3: print(c); }

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 i`.

(100)	$t_0 := 0$
(101)	$i := t_0$
(102)	$y := k$
(103)	$x := 0$
(104)	$t_1 := y + x$
(105)	$y := t_1$
(106)	$t_2 := i * 4$
(107)	$t_3 := t_2 + 2$
(108)	$t_4 := t_3 * y$
(109)	$t_5 := A[t_4]$
(110)	$x := x + t_4$
(111)	$i := y$
(112)	$if\ \ i < x\ \ goto\ \ 200$

5. Dado el siguiente fragmento de código intermedio:

(100)	$s := 0$
(101)	$m := 1$
(102)	$t_1 := m * 2$
(103)	$ini := size * 2$
(104)	$t_2 := t_1 - ini$
(105)	$if\ \ x > y\ \ goto\ \ 108$
(106)	$s := a[t_2]$
(107)	$goto\ \ 109$
(108)	$s := b[t_2]$
(109)	$tot := tot + s$
(110)	$m := m + 2$
(111)	$if\ \ m < 100\ \ goto\ \ 102$
(112)	$print(s)$

a) (0.5 ptos.) Determinad los bloques básicos que forman el/los bucle/s. Extrae el código invariante. 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.

Lenguajes de Programación y Procesadores de Lenguajes

(2º parcial) **13 de enero de 2014**

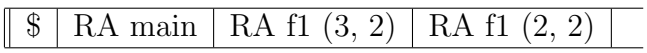
Respuesta 1-a:

nombre	categoría	nivel	desplazamiento	tipo
f1	función	0	–	tfunción((tentero,tentero)→treal
main	función	0	–	tfunción(tvacio→tvacio
a	variable	1	0	tentero
c	variable	1	2	treal
b	variable	1	6	tentero
a	variable	2	8	tentero
b	variable	2	10	tentero

Respuesta 1-b:

Pto.A c (1, 2); a (2, 8); b (1, 6)
Pto.B c (1, 2); a (2, 8); b (2, 10)

Respuesta 1-c:



Respuesta 2-a:

Dado la llamada a una cierta función f: `1)` calcular el dominio de los parámetros actuales, que será una lista ordenada (producto cartesiano) de los tipos de los paraámetros actuales; `2)` Obtener de la TDS el dominio de los parámetros formales asociado a la función f; y `3)` verificar que coinciden.

Respuesta 2-b:

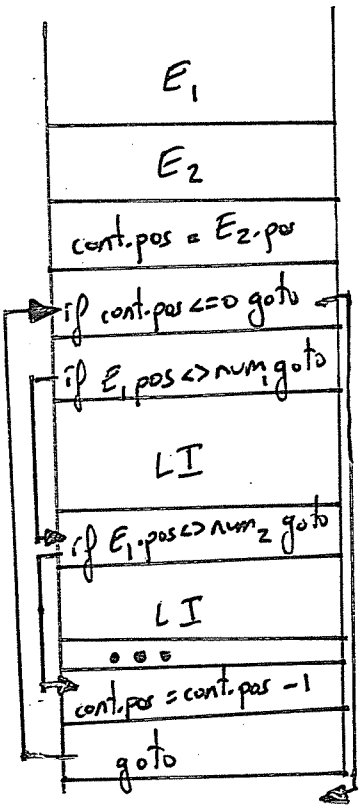
El valor de retorno asociado con una función f se almacenará en el primer campo su RA. Se hará simplemente con una operación `(SP = SP + talla_val_retorno)`, donde la talla del valor de retorno es la talla del rango de la funión f que se puede obtener de la TDS.

Respuesta 2-c:

Las variables temporales se situaran en el RA, en el (único) segmento de variables, después de la definición de las variables locales a la función. Dado que tanto las variables locales como las temporales comparten el mismo segmento y dicho segmento se gestiona con una misma variable (p.ej. "segvar") la reserva de espacio se puede realizar con una instrucción:
`(SP = SP + segvar)`

(3)

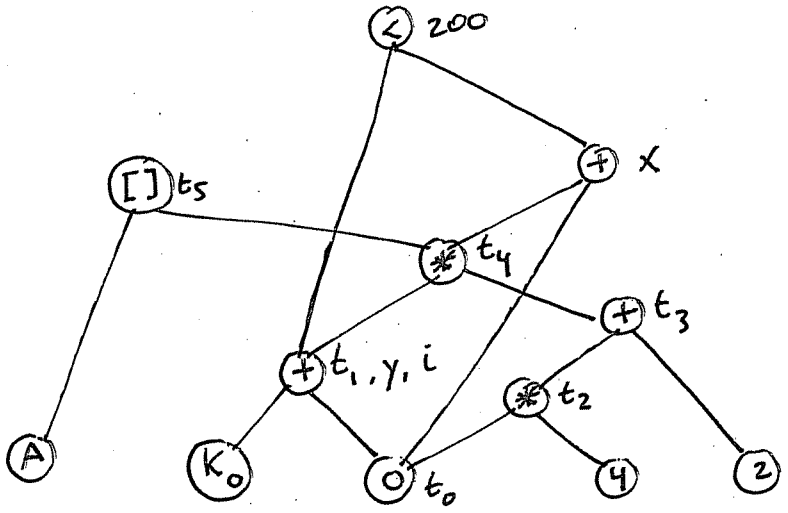
```
I → dependon E1 execute E2 times {
      {cont.pos = CrearVarTemp();
       emite (cont.pos := E2.pos);
       ini = SI;
       fin = CreaLans(SI)
       emite ('if' cont.pos '<=0 goto' -);
       B.exp = E1.pos
      }
   B  { emite (cont.pos := cont.pos - 1);
        emite ('goto' ini);
        CompletaLans (fin, SI);
      }
```

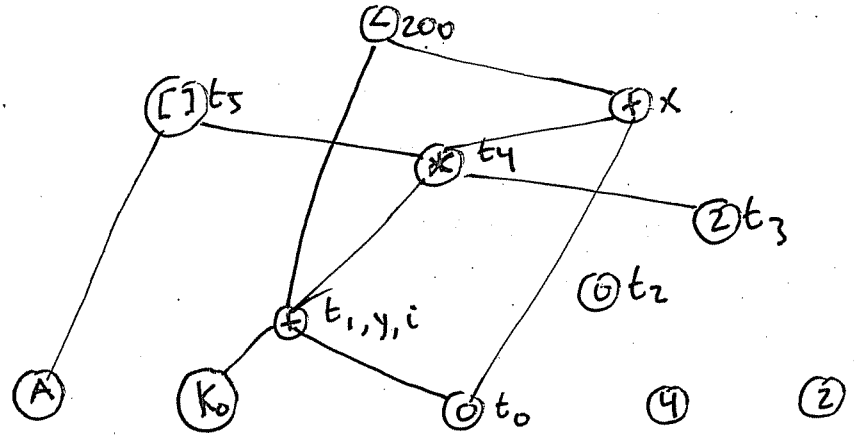



```
B → num:
      { sig = CreaLans (SI);
        emite ('if' B.exp <> num.val goto -);
      }
   LI;
      { CompletaLans (sig, SI);
        B1.exp = B.exp;
      }
   B
```

```
LI → I ; LI
   | ε
```

(4)



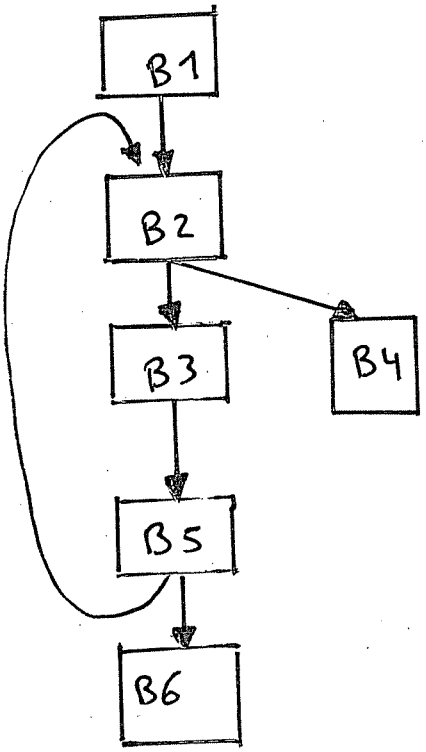



```
i = K
t4 = i * 2
x = t4
t5 = A [t4]
if i < x goto 200
```

si ni t_4 ni t_5 están activas a la salida:

```
i := K
t4 = i * 2
x := t4
if i < x goto 200
```

(5)



B1: 100-101
B2: 102-105
B3: 106-107
B4: 108
B5: 109-111
B6: 112

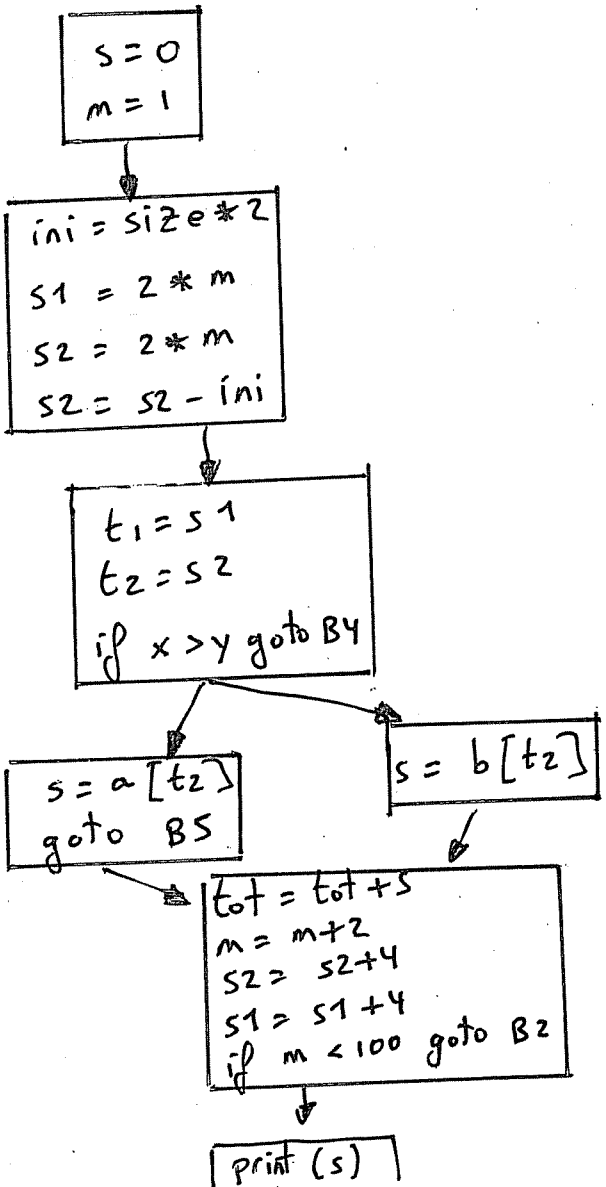
A. Retroceso: B5 → B2

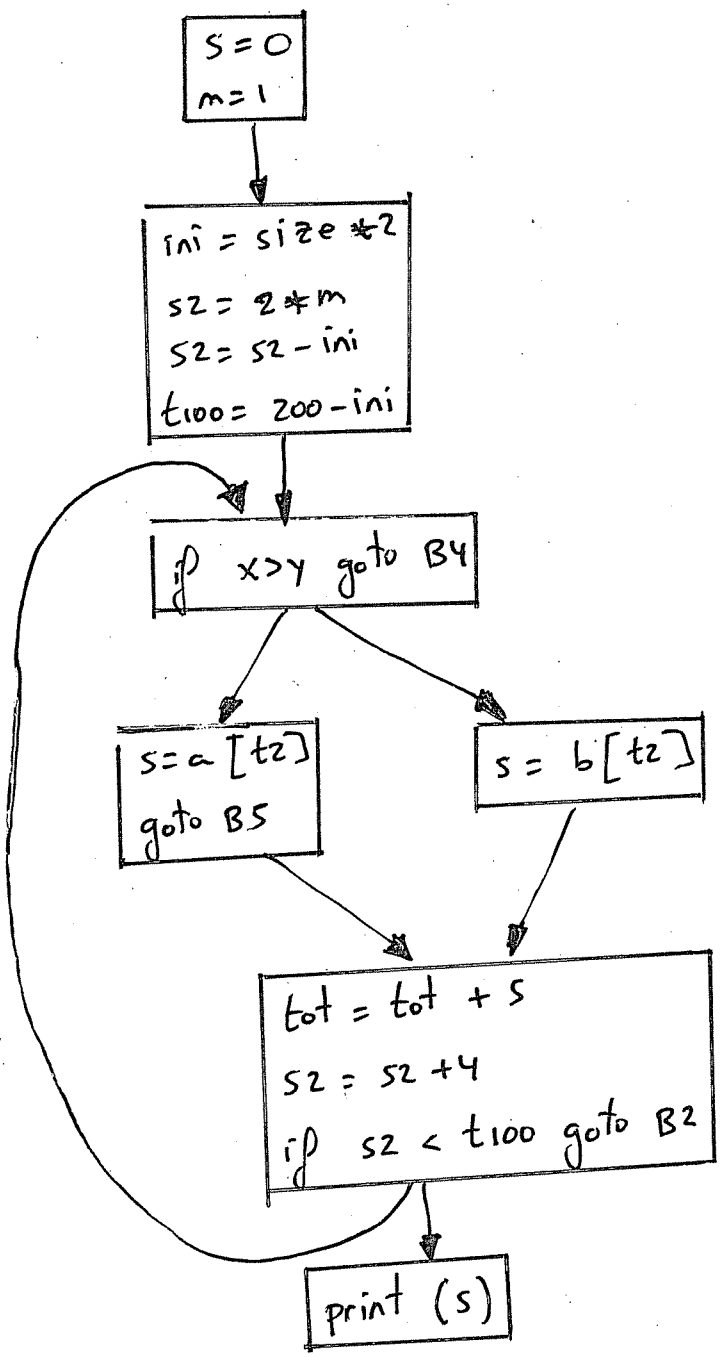
Bucle natural: B2, B5, B3, B4

Código invariante: ini = size * 2

Variables inducción:

m (m, 1, 0)
t_1 (m, 2, 0)
t_2 (m, 2, -ini)




S = 0
m = 1
ini = size * 2
s2 = 2 * m
s2 = s2 - ini
t100 = 200 - ini
if x > y goto B4
s = a [t2]
goto B5
s = b [t2]
tot = tot + s
s2 = s2 + 4
if s2 < t100 goto B2
print (s)