Lenguajes de Programación y Procesadores de Lenguajes

(2º parcial) **19 de enero de 2022**

1. `(1,5 ptos.)` Dado el siguiente fragmento de un programa en `C` y, suponiendo que la talla de enteros es 2, la de los reales es 4 y la del segmento de gestión (enlaces de control, dirección de retorno, etc) es 8, mostrad el contenido completo de la TdS en los puntos de control [1] y [2].

```
-------------------------------------------------
int M[100];
float func (int x, float y, int z) {
  int N[100];  int j;
  ....                                  // <---- [1]
}
struct{int a; float b; int c;} R;
int main () {
  int i;  float M[100];  float k;
  ....                                  // <---- [2]
}
-------------------------------------------------
```

2. `(2 ptos.)` Contestad brevemente a las siguientes cuestiones:

   a) `(0,5 ptos.)` Diseñad un ETDS para la comprobación de tipos y la generación de código intermedio para la regla:

   $$\mathrm{E} \rightarrow \texttt{not}\ \mathrm{E}$$

   b) `(0,75 ptos.)` Dado el siguiente código intermedio, obtened el código intermedio optimizado aplicando transformaciones locales mediante el correspondiente GDA (grafo dirigido acíclico). Solo la variable `x` está activa a la salida del bloque.

$t_1 = 10$	$t_4 = 0$	$t_6 = t_3 * t_5$
$t_2 = 2$	$x = 1$	$x = t_6$
$t_3 = t_1 * t_2$	$t_5 = t_4 + x$	

   c) `(0,75 pto.)` Construid el grafo de interferencias para el siguiente bloque de instrucciones.

Activas de entrada	{ a, d, f}
b ← d + f	
e ← a - b	
Activas de salida	{ f, e, b }

3. `(3,5 ptos.)` Diseñad un ETDS que genere código intermedio para el siguiente fragmento de una gramática:

$$\begin{array}{lcl} \mathrm{I} & \rightarrow & \mathrm{L} \mid \textbf{optloop}\ (\ \mathrm{E}\ )\ \mathrm{L} \mid \textbf{next}\ ; \\ \mathrm{L} & \rightarrow & \textbf{from}\ \ \textbf{id} = \mathrm{E}\ \ \textbf{to}\ \ \mathrm{E}\ \ \textbf{do}\ \ \mathrm{I}\ ; \end{array}$$

   Si la expresión E que sigue a **optloop** es cierta, se ejecuta el bucle que le sigue; en caso contrario, no se ejecuta el bucle. En el fragmento de gramática propuesto, **optloop** puede ir seguido de un bucle **from-to-do**, que comienza asignando el valor de la primera E a la variable `id` y, mientras `id` sea menor o igual que la segunda E, se ejecutará el cuerpo

del bucle I . La ejecución de la instrucción **next** implica saltar a la siguiente iteración del bucle.

Ejemplo: bool a=true; int b;
optloop (a) from b=3 to 6 { print(b); next; print(0); }

Imprimirá: 3, 4, 5, 6. Si no estuviera la instrucción **next**, imprimiría: 3, 0, 4, 0 , 5, 0, 6, 0.

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

(100)	$t_0 = 3$	(104)	$z = x * t_0$	(108)	$t_4 = t_3 + z$
(101)	$x = t_0$	(105)	$i = t_1$	(109)	$A[t_2] = t_4$
(102)	$y = 12$	(106)	$t_2 = z + j$	(110)	$i = i + j$
(103)	$t_1 = y + x$	(107)	$t_3 = A[t_2]$	(111)	if $i >= k$ goto 120

5. Dado el siguiente fragmento de código intermedio,

(101) $x = 0$	(105) $t_3 = t_2 + 10$	(108) goto 110	(111) $x = x - 2$
(102) $t_1 = x * 4$	(106) if $y < t_3$ goto 109	(109) t4 = t3 *2	(112) if $x > 0$ goto 102
(103) $y = e * 2$	(107) $t_4 = t_3$	(110) $z = z + t_4$	(113) $a = z$
(104) $t_2 = t_1 * 3$			

   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.