Lenguajes de Programación y Procesadores de Lenguajes

(2º parcial) **15 de enero de 2019**

1. Dado el siguiente programa C:

```
-----------------------------------------------------
#include<stdio.h>
float f2(float, int ) ;
int r ;

int f1(int p11, int p12, int p13) {
   float r ;

   r = p12;
   while (r < 6) {
      r = f2(r, p13) ; }          // <--- TDS
   return r ; }

float f2(float p1, int p2) {
   float r ;

   r = p1 + p2 ;                    // <--- PILA
   return r ; }

int main () {
  int i = 1 ;

  r = f1(i, i*2, i*3) ;
  printf( "%d\n", r);  }
-----------------------------------------------------
```

   Suponiendo que la talla de enteros es 2 y la de los reales, enlaces de control y dirección de retorno es 4:

   a) `(0,75 ptos.)` Mostrad el contenido completo de la TDS en el punto de control TDS.

   b) `(0,5 ptos.)` Indicad el desplazamiento relativo de los parámetros p1 y p2 que aparecen en la instrucción del punto de control PILA.

   c) `(0,75 ptos.)` Mostrad el contenido de la pila de ejecución cada vez que se pasa por el punto de control PILA.

2. Contestad brevemente a las siguientes cuestiones:

   a) `(0,5 ptos.)` Enumerad los tipos de transformaciones algebraicas que se usan para optimizar código y pon un ejemplo de cada una de ellas.

   b) `(0,5 ptos.)` Definid grafo de interferencias e indicad su utilidad.

   c) `(0,5 ptos.)` Indicad cuál de las siguientes afirmaciones es FALSA:

1) Para poder asignar memoria estática es necesario conocer el tamaño del objeto en tiempo de compilación.
2) Durante la carga de un registro de activación, el enlace de control lo apila el bloque llamado.
3) Una tabla de dispersión es una buena estructura de datos para implementar una tabla de símbolos.
4) El "frame pointer" apunta al primer parámetro apilado en un "frame" o registro de activación.

3. `(3,5 ptos.)` Diseñad un ETDS que genere código intermedio para una instrucción `case` similar a la del C:

$$\begin{array}{lcl} \mathrm{I} & \rightarrow & \texttt{case}\ \mathrm{E}\ \texttt{of}\ \mathrm{L}\ \texttt{end} \\ \mathrm{L} & \rightarrow & \mathrm{C}\ :\ \mathrm{I}\ \mathrm{L} \mid \epsilon \\ \mathrm{C} & \rightarrow & \textbf{cte}\ ,\ \mathrm{C} \mid \textbf{cte} \end{array}$$

Si alguna de las constantes (**cte**), en la lista de constantes de C, coincide con la expresión E se debe ejecutar la instrucción I correspondiente y terminar el `case`. Si no, se debe seguir explorando la lista de ítems L.

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: `w, x, y`.

(100)	$t_1 = 0$	(104)	$t_5 = b$	(108)	$y = x + 10$
(101)	$t_2 = a$	(105)	$t_6 = a + t_5$	(109)	$x = a * b$
(102)	$t_3 = t_2 + t_1$	(106)	$x = t_6$	(110)	$y = b * 10$
(103)	$t_4 = t_3 + b$	(107)	$w = t_4 - x$		

5. Dado el siguiente fragmento de código intermedio, y sabiendo que a la salida del bucle solo estará activa la variable: `a`:

(100)	$k = 0$	(106)	$t_4 = a[t_3]$	(112)	$t_8 = t_7 * 4$
(101)	$L = 10$	(107)	$t_5 = t_4 + 10$	(113)	$t_9 = a[t_8]$
(102)	`if` $k > 5$ `goto` 110	(108)	$a[t_3] = t_5$	(114)	$t_{10} = t_9 + 10$
(103)	$t_1 = k * 20$	(109)	`goto` 116	(115)	$a[t_8] = t_{10}$
(104)	$t_2 = t_1 + L$	(110)	$t_6 = L * 20$	(116)	$k = k + 1$
(105)	$t_3 = t_2 * 4$	(111)	$t_7 = t_6 + k$	(117)	`if` $k < 10$ `goto` 102

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)

TDS

Lexema	Categoría	Tipo	Nivel	desp
f2	función	treal	0	-
r	variable	tentero	0	0
p11	parámetro	tentero	1	-10
p12	parámetro	tentero	1	-12
p13	parámetro	tentero	1	-14
f1	función	tentero	0	-
r	variable	treal	1	0

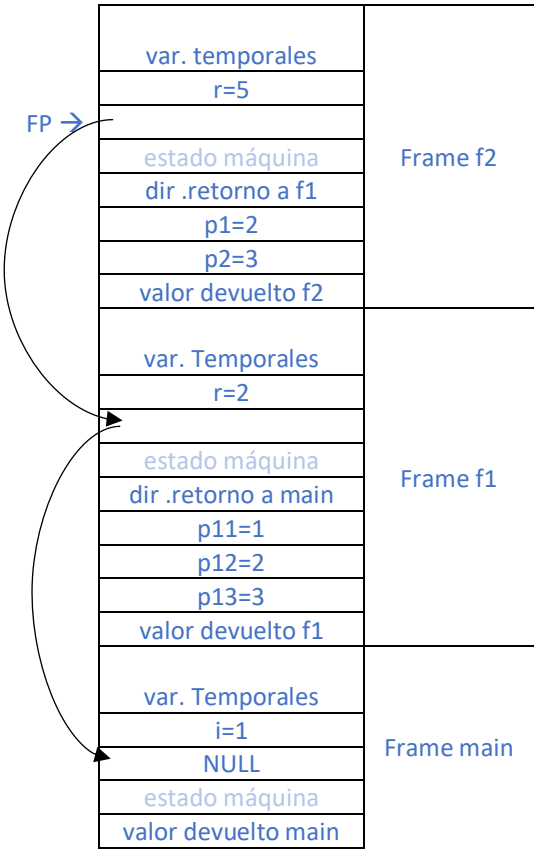
TdArgumento

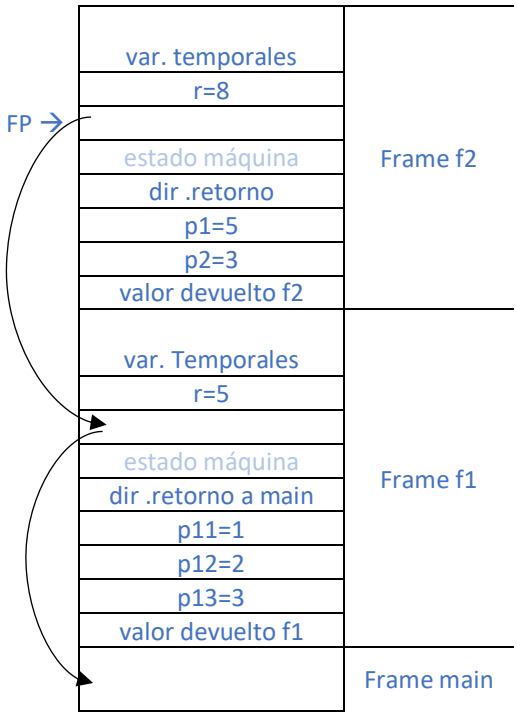
treal	tentero

tentero	tentero	tentero

b) $desp_{p1}$: -12 $desp_{p2}$: -14

c)





2.a.-

- Simplificaciones algebraicas:

  Expresiones de identidad: a * 1 → a
  Propiedad conmutativa: a+b → b+a
  Propiedad asociativa: a+(b+c) → (a+b) + c
  Operador θ_1 distributivo respecto a θ_2: a*(b+c) → a*b + a*c
  Operador unario autoinverso: --a → a

- Reducción de intensidad: 2*a → a+a
- Cálculo previo de constantes: PI= 3.14 ; a= 2*PI → a= 3.28

2.b.-

Llamamos *grafo de interferencias* a un grafo no dirigido donde:

- Cada nodo representa un valor (variable).
- Hay un arco entre los nodos t_1 y nodo t_2 (t_1, t_2) si los valores que representan ambos nodos no pueden asignarse al mismo registro porque representan variables activas al mismo tiempo.
- Si el procesador no puede producir el resultado de a:= b+c en el registror r_i añadir el arco (a, r_i).

Se emplea en los algoritmos de asignación de registros, ya que dos nodos adyacentes representan a dos variables activas al mismo tiempo, y por lo tanto no se podrán asignar al mismo registro.

2.c.-

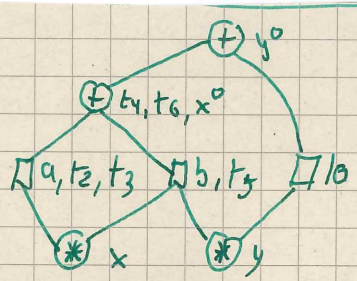
El “frame pointer” NO apunta al primer parámetro apilado en un “frame” o registro de activación. El “frame pointer” apunta a la base del área de datos locales.

I → case E of L end	• L.pos ← E.pos Completa lans (L.fin, Ω);
L → C : I L' / ε	• C.pos ← L'.pos ← L.pos • completa lans (C.v, Ω); I.fin ← crea lans (Ω); emite (goto □); completa lans (C.f, Ω); L.fin ← fusiona lans (I.fin, L'.fin); L'.fin ← nil
C → cte	• C.v ← crea lans (Ω); emite (if cte.val = E.pos goto □); • C.f ← crea lans (Ω); emite (goto □);
→ cte C'	• C'.pos ← C.pos • C.v ← fusiona lans (crea lans (Ω), C'.v); emite (if cte.val = C'.val goto □); • C.f ← fusiona lans (crea lans (Ω), C'.f); emite (goto □);

$t_1 \leftarrow 0$
$t_2 \leftarrow a$
$t_3 \leftarrow t_2 + t_1$
$t_4 \leftarrow t_3 + b$
$t_5 \leftarrow b$
$t_6 \leftarrow a + t_5$

$x \leftarrow t_6$
$w \leftarrow t_4 - x$
$y \leftarrow x + 10$
$x \leftarrow a * b$
$y \leftarrow b * 10$

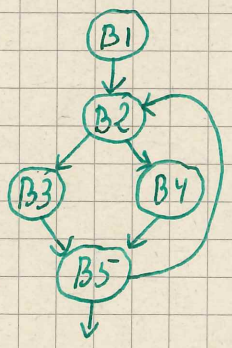
□ $t_1, 0, w$



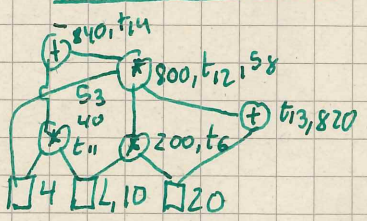
w, x, y

$w \leftarrow 0$
$x \leftarrow a * b$
$y \leftarrow b * 10$

$K \leftarrow 0$	B1
$L \leftarrow 10$	
if $K > 5$ goto B4	B2
$t_1 \leftarrow K * 20$	B3
$t_2 \leftarrow t_1 + L$	
$t_3 \leftarrow t_2 * 4$	
$t_4 \leftarrow a[t_3]$	
$t_5 \leftarrow t_4 + 10$	
$a[t_3] \leftarrow t_5$	
goto B5	
$t_6 \leftarrow L * 20$ ← *	B4
$t_7 \leftarrow t_6 + K$	
$t_8 \leftarrow t_7 * 4$	
$t_9 \leftarrow a[t_8]$	
$t_{10} \leftarrow t_9 + 10$	
$a[t_8] \leftarrow t_{10}$	
$K \leftarrow K + 1$	B5
if $K < 10$ goto B2	



$K \leftarrow 0$	B1
$L \leftarrow 10$	
$t_6 \leftarrow L * 20$	
$S1 \leftarrow 20 \cdot K$ ← 1	
$S2 \leftarrow 20 \cdot K$ ← 1	
$S2 \leftarrow S2 + L$ ← 1	
$S3 \leftarrow 80 \cdot K$	
$t_{11} \leftarrow 4 \cdot L$	
$S3 \leftarrow S3 + t_{11}$	
$S7 \leftarrow K + t_6$ ← 1	
$S8 \leftarrow 4 \cdot K$	
$t_{12} \leftarrow 4 \cdot t_6$	
$S8 \leftarrow S8 + t_{12}$	
$t_{13} \leftarrow 20 + t_{12}$	
$t_{14} \leftarrow 40 + t_{12}$	



S8, S3

K (K, 1, 0)
t1 (K, 20, 0) S1
t2 (K, 20, L) S2
t3 (K, 80, 4·L) S3
t7 (K, 1, t6) S7
t8 (K, 4, 4·t6) S8

B2	if $S8 > t_{13}$ goto B4 ← 2
B3	$t_1 \leftarrow S1$ ← 1
	$t_2 \leftarrow S2$ ← 1
	$t_3 \leftarrow S3$ ← 3
	$t_4 \leftarrow a[t_3]$ ←
	$t_5 \leftarrow t_4 + 10$
	$a[t_3] \leftarrow t_5$ ←
	goto B5
B4	t_6
	$t_7 \leftarrow S7$ ← 1
	$t_8 \leftarrow S8$ ← 3
	$t_9 \leftarrow a[t_8]$
	$t_{10} \leftarrow t_9 + 10$
	$a[t_8] \leftarrow t_{10}$
B5	$K \leftarrow K + 1$ ← 2
	$S1 \leftarrow S1 + 20$ ← 1
	$S2 \leftarrow S2 + 20$ ← 1
	$S3 \leftarrow S3 + 80$
	$S7 \leftarrow S7 + 1$ ← 1
	$S8 \leftarrow S8 + 4$
	if $S8 < t_{14}$ goto B2 ← 2

B1	$S3 \leftarrow 40$
	$S8 \leftarrow 800$
	$t_{13} \leftarrow 820$
	$t_{14} \leftarrow 840$

B2	if $S8 > t_{13}$ goto B4
B3	$t_4 \leftarrow a[S3]$
	$t_5 \leftarrow t_4 + 10$
	$a[S3] \leftarrow t_5$
	goto B5

B4	$t_9 \leftarrow a[S8]$
	$t_{10} \leftarrow t_9 + 10$
	$a[t_8] \leftarrow t_{10}$
B5	$S3 \leftarrow S3 + 80$
	$S8 \leftarrow S8 + 4$
	if $S8 < t_{14}$ goto B2