Lenguajes de Programación y Procesadores de Lenguajes

(2º parcial) 10 de enero de 2020

1. (1,5 ptos.) Dado el siguiente fragmento de un programa en C y, suponiendo que la talla de los enteros es 2, la de los reales 4 y la del segmento de gestión (enlaces de control, dirección de retorno, etc.) 6, mostrad el contenido completo de la TDB y de la TDS en los puntos de control [1] y [2]

```
-----------------------------------------
float A [100];
bool  x;
int f1(float p1, int p2)
{ int x;
  ...                              // <---- [1]
}
int y;
int main ()
{ float x, int z;
  ...                              // <---- [2]
}
-----------------------------------------
```

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

   a) ¿Porqué es necesario gestionar la memoria para las funciones mediante una pila de registros de activación (RA)? ¿Porqué el segmento de parámetros y el segmento de variables locales y temporales están separados en el RA de una función?

   b) Diseñad un ETDS para la comprobación de tipos y la generación de código intermedio para la regla:

   $$\text{E} \rightarrow \text{E oprel E}$$

   c) 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.

(100)	$t_1 = 1$	(102)	$t_3 = t_1 * t_2$
(101)	$t_2 = 10$	(103)	$x = t_3$

   d) Dado el siguiente fragmento de código y las variables activas de entrada y salida relacionadas con cada instrucción, proporcionad el grafo de interferencias asociado.

	{ e, d }
b ← e + d	{ e, d, b }
c ← e - 1	{ d, b, c }
a ← d * 2	{ b, c, a }

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

$$\begin{array}{lcl} \text{I} & \rightarrow & \texttt{for id}^1\ \texttt{in id}^2\ \text{S}\ \texttt{do}\ \text{I} \\ \text{S} & \rightarrow & \texttt{step E} \mid \epsilon \end{array}$$

La instrucción representa un bucle que recorre un array de enteros, id^2 (talla de los enteros = TallaEntero), asignando en cada iteración a la variable id^1 un elemento del array. Si no hay clausula `step`, se recorre el array de uno en un elemento (paso = 1). Si hay clausula `step` E (E es una expresión entera), se recorre el array con un paso igual al valor de E.

Ejemplo:

int a[]={10,20,30,40,50,60};

for x in a step 2 do print(x);

imprimirá: 10 30 50

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 estará activa la variable: s.

(100)	$t_0 = 0$	(105)	$d = t_1$	(109)	$t_5 = a + 5$
(101)	$a = t_0$	(106)	$t_2 = t_0 + 5$	(110)	$t_6 = t_5 * c$
(102)	$b = pi$	(107)	$t_3 = t_2 * c$	(111)	$s = s + t_4$
(103)	$c = a * s$	(108)	$t_4 = Z[t_3]$	(112)	`if` $t_6 < k$ `goto 200`
(104)	$t_1 = b * c$				

5. Dado el siguiente fragmento de código intermedio,

(101)	$a = 5$	(106)	`if` $t_0 > j$ `goto 109`	(111)	$suma = suma + t_5$
(102)	$b = a * 2$	(107)	$t_4 = R1[t_3]$	(112)	$a = a - 2$
(103)	$t_1 = j - 2$	(108)	`goto 110`	(113)	`if` $a > 100$ `goto 102`
(104)	$t_2 = b * 3$	(109)	$t_4 = R2[t_3]$	(114)	`print(` $suma$ `)`
(105)	$t_3 = t_2 + 5$	(110)	$t_5 = t_4 * 2$		

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.

I -> for id1 in id2
 	
S
 	
do I
	{ id1.pos = BuscaPos(id1.nom);
	id2.pos = BuscaPos(id2.nom);
	If (BuscaTipo(id2.nom) != tvector(tentero) yyerror("id2 no es array");
	I.nelem = Busca_NumElementos(id2.nom)
	I.cont = CrearVarTemp(); I.temp = CrearVarTemp();
	emite(I.cont ':= 0');
	
	I.inicio = SIGINST ;
	I.fin = CreaLans(SIGINST) ; emite('if' I.cont '>=' I.nelem goto --)
	emite(I.temp ':=' I.cont '*' TallaEntero);
	emite(id1.pos ':=' id2.pos '[' I.temp']');
	
	emite(I.cont ':=' I.cont '+' S.incr);
	emite('goto' I.inicio);
	CompletaLans(I.fin,SIGINST);

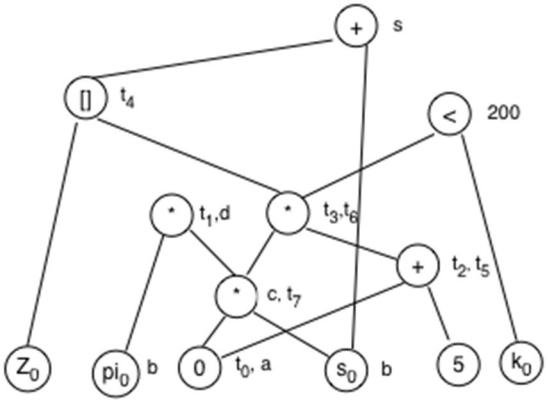
S -> step E	S.incr := E.pos
S -> epsilon	S.incr := CrearVarTemp(); emite (S.incr ':=' 1);

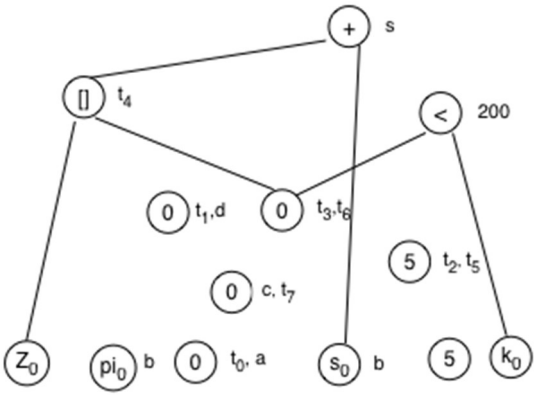
Otra alternativa:

I -> for id1 in id2
 	
S
 	
do I
	{ id1.pos = BuscaPos(id1.nom);
	id2.pos = BuscaPos(id2.nom);
	If (BuscaTipo(id2.nom) != tvector(tentero) yyerror("id2 no es array");
	I.nelem = Busca_NumElementos(id2.nom)
	S.cont = CrearVarTemp(); I.temp = CrearVarTemp();
	emite(S.cont ':= 0');
	emite(goto SIGINST '+' 2);
	I.inicio = SIGINST ;
	
	I.fin = CreaLans(SIGINST); emite('if' S.cont '>=' I.nelem goto --)
	emite(I.temp ':=' S.cont '*' TallaEntero);
	emite(id1.pos ':=' id2.pos '[' I.temp']');
	
	emite('goto' I.inicio);
	CompletaLans(I.fin,SIGINST);

S -> step E	emite(S.cont ':=' S.cont '+' E.pos);
S -> epsilon	emite(S.cont ':=' S.cont '+' 1);

4.-

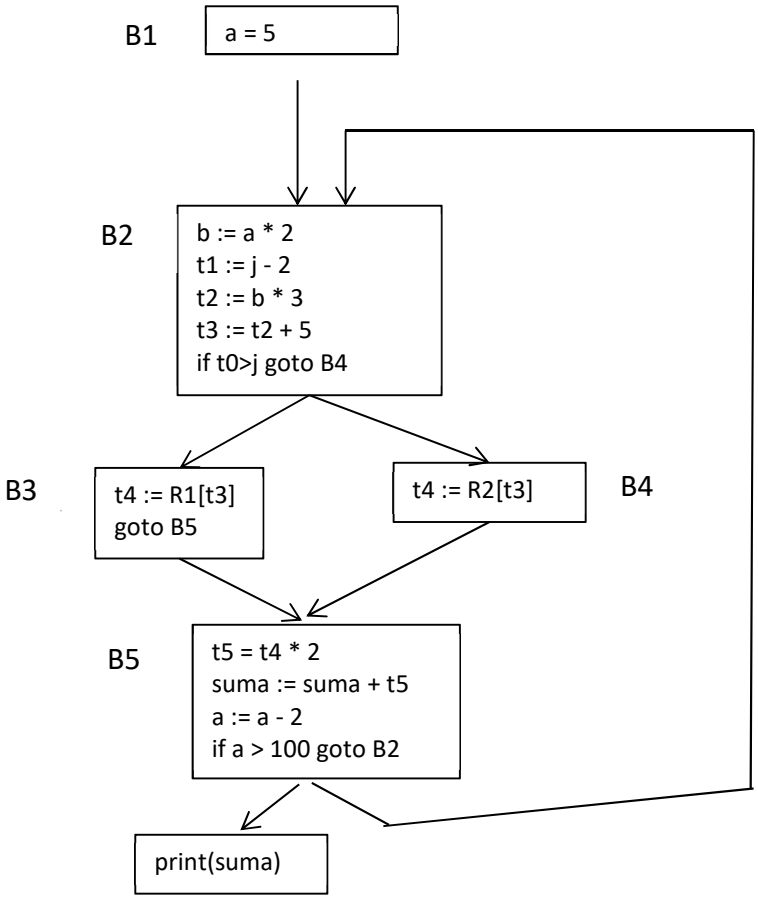




t4 := Z[0]
s := s + t4
if (0 < k) goto 200

5.-

(101 a = 5
(102) b := a * 2
(103) t1 := j - 2
(104) t2 := b * 3
(105) t3 := t2 + 5
(106) if t0>j goto 109
(107) t4 := R1[t3]
(108) goto 110
(109) t4 := R2[t3]
(110) t5 = t4 * 2
(111) suma := suma + t5
(112) a := a-2
(113) if a > 100 goto 102
(114) print (suma)

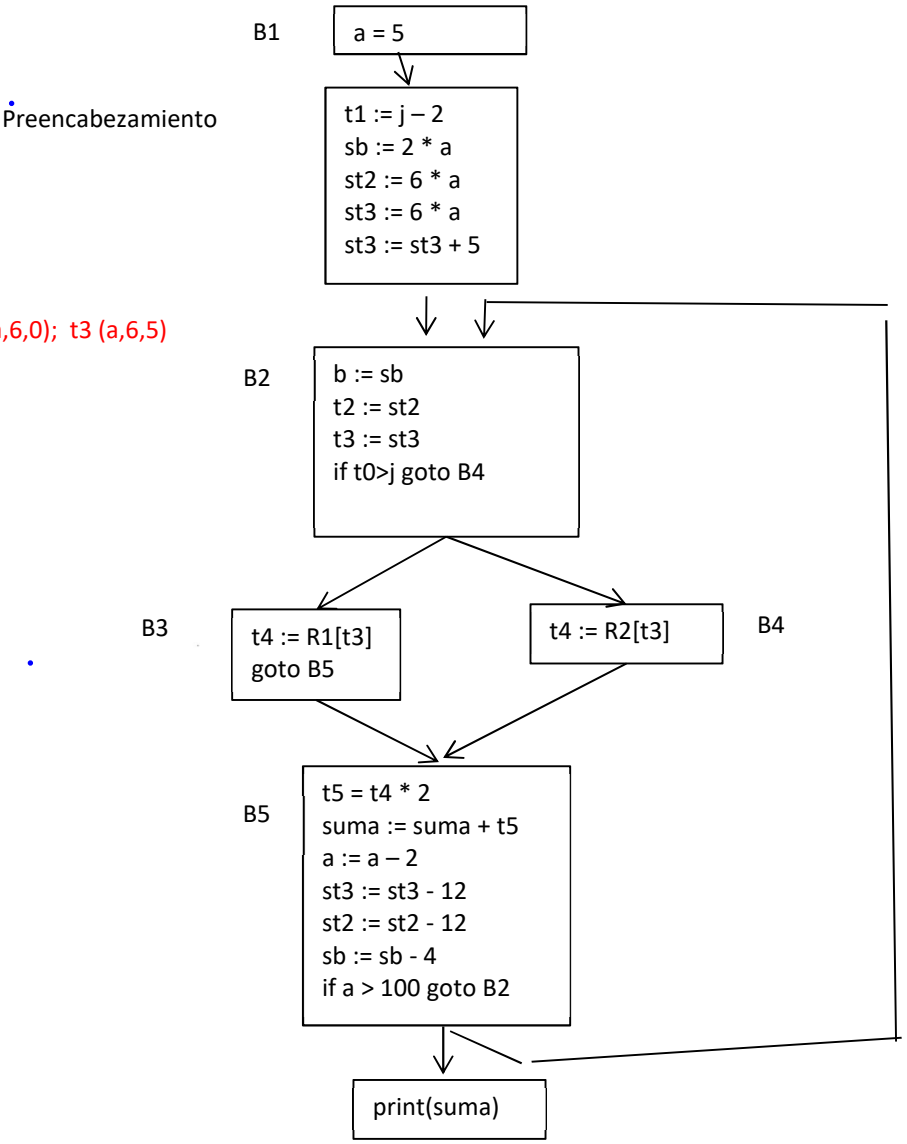


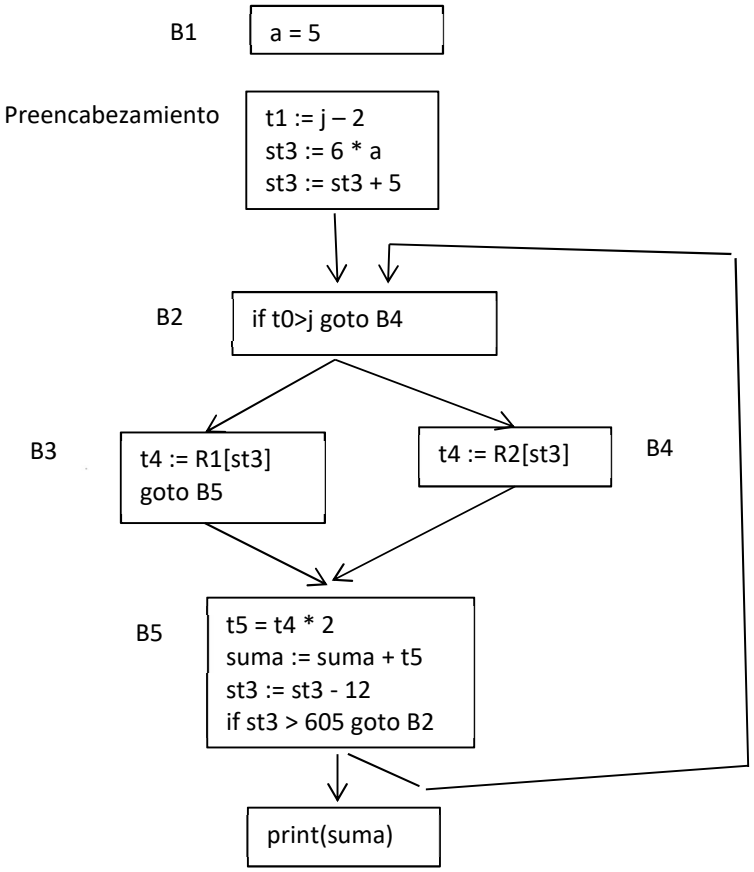
Arista de retroceso: B5->B2
Bucle natural: B2, B3, B4 y B5
Código invariante: t1:= j - 2
Variables de Inducción: a (a,1,0) ; b (a,2,0); t2 (a,6,0); t3 (a,6,5)

Preencabezamiento

Variables de Inducción: a (a,1,0) ; b (a,2,0); t2 (a,6,0); t3 (a,6,5)

a = a - 2 ->
st3 = st3 - 2 * 6 = st3 - 12
st2 = st2 - 2 * 6 = st2 - 12
sb = sb - 2 * 2 = sb - 4




B1
a = 5
Preencabezamiento
t1 := j – 2
st3 := 6 * a
st3 := st3 + 5
B2
if t0>j goto B4
B3
t4 := R1[st3]
goto B5
t4 := R2[st3]
B4
B5
t5 = t4 * 2
suma := suma + t5
st3 := st3 - 12
if st3 > 605 goto B2
print(suma)