

## 7. Optimización de Código Intermedio

### 7.1. Introducción al problema de la optimización

- Motivación, criterios, fuentes y estrategias de optimización
- Boques Básicos y Grafo de Flujo
- Actividad de las variables

### 7.2. Optimizaciones Locales

- Transformaciones algebraicas
- Transformaciones que preservan la estructura
- Grafos Dirigidos Acíclicos (GDAS)
- Reconstrucción del código a partir de los GDA

### 7.3. Optimizaciones Globales

- Transformaciones que optimizan los saltos
- Detección de Bucles Naturales
- Extracción de código invariante a un bucle
- Reducción de intensidad y eliminación de variables de inducción



“La optimización prematura es la raíz de todos los males”.

D. Knuth

### ➤ Motivación

- Necesidad de una optimización de código independiente de la máquina.
- El 90 % del tiempo de ejecución de un programa se realiza en el 10 % del código.

### ➤ Ejemplo

$$a := x \uparrow 2 + y \quad \Rightarrow \quad \begin{array}{l} t_1 \leftarrow 2 \\ t_2 \leftarrow x \uparrow t_1 \\ t_3 \leftarrow t_2 + y \\ a \leftarrow t_3 \end{array} \quad \Rightarrow \quad \begin{array}{l} t_2 \leftarrow x * x \\ a \leftarrow t_2 + y \end{array}$$

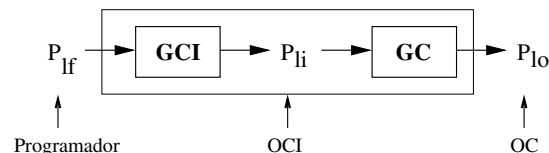
- Reducción del número de instrucciones
- Reducción de las variables temporales
- Eficiencia de los operadores

## OPTIMIZACIÓN DE CÓDIGO INTERMEDIO

### ➤ Criterios de optimización

- preservar el significado,
- acelerar (significativamente) la ejecución del código objeto,
- valorar el coste de la transformación.

### ➤ Fuentes de optimización



### ➤ Estrategias de optimización:

- Optimizaciones Locales.
- Optimizaciones Globales.

## EJEMPLO DE OPTIMIZACIÓN DE CÓDIGO (INTERMEDIO)

```

var A: array of [1..10,1..20] of integer; /* talla de enteros=2 */
...
while A[i,k] < M do k:=k-1;
    
```

	Cod. original	Op. locales	Op. globales
100	$t_1 := 10$	100	$t_2 := i * 10$
101	$t_2 := i * t_1$	101	$t_3 := t_2 + k$
102	$t_3 := t_2 + k$	102	$t_4 := t_3 * 2$
103	$t_4 := t_3 * 2$	103	$t_5 := A[t_4]$
104	$t_5 := A[t_4]$	104	$t_5 := A[t_4]$
105	if $t_5 < M$ goto 107	105	if $t_5 < M$ goto 107
106	goto 111	106	goto 111
107	$t_6 := 1$	107	$k := k - 1$
108	$t_7 := k - t_6$	108	
109	$k := t_7$	109	
110	goto 100	110	goto 100
111		111	

## BLOQUES BÁSICOS

### ALGORITMO: Detección de los Bloques Básicos

- Se define como **líder**, la primera instrucción de un Bloque Básico.
- Dada una secuencia de instrucciones de Código Intermedio inicial, el conjunto de líderes se obtiene como:
  - la primera instrucción del programa,
  - toda instrucción apuntada por una instrucción de salto,
  - toda instrucción siguiente de una instrucción de salto.
- Para cada uno de los líderes detectados, su **Bloque Básico** lo forman: el líder y la secuencia de instrucciones que le siguen hasta el siguiente líder (sin incluirlo) o el fin de programa.

## CÓDIGO EN C PARA EL *quicksort*

```
void quicksort (m, n)
int m, n;
{
    int i, j, v, x;
    if (n <= m) return;
    /*----- el fragmento comienza aquí -----*/
    i = m-1; j = n; v = a[n];
    while (1)
    {
        do i = i+1; while (a[i] < v);
        do j = j-1; while (a[j] > v);
        if (i >= j) break;
        x = a[i]; a[i] = a[j]; a[j] = x;
    }
    x = a[i]; a[i] = a[n]; a[n] = x;
    /*----- el fragmento termina aquí -----*/
    quicksort (m, j);
    quicksort (i+1, n);
}
```

## CÓDIGO INTERMEDIO DEL FRAGMENTO DEL *quicksort*

132	$t_1 := 1$	152	$t_{13} := 4 * i$
133	$t_2 := m - t_1$	153	$t_{14} := a[t_{13}]$
134	$i := t_2$	154	$x := t_{14}$
135	$j := n$	155	$t_{15} := 4 * i$
136	$t_3 := 4 * n$	156	$t_{16} := 4 * j$
137	$t_4 := a[t_3]$	157	$t_{17} := a[t_{16}]$
138	$v := t_4$	158	$a[t_{15}] := t_{17}$
139	$t_5 := 1$	159	$t_{18} := 4 * j$
140	$t_6 := i + t_5$	160	$a[t_{18}] := x$
141	$i := t_6$	161	goto 139
142	$t_7 := 4 * i$	162	$t_{19} := 4 * i$
143	$t_8 := a[t_7]$	163	$t_{20} := a[t_{19}]$
144	if $t_8 < v$ goto 139	164	$x := t_{20}$
145	$t_9 := 1$	165	$t_{21} := 4 * i$
146	$t_{10} := j - t_9$	166	$t_{22} := 4 * n$
147	$j := t_{10}$	167	$t_{23} := a[t_{22}]$
148	$t_{11} := 4 * j$	168	$a[t_{21}] := t_{23}$
149	$t_{12} := a[t_{11}]$	169	$t_{24} := 4 * n$
150	if $t_{12} > v$ goto 145	170	$a[t_{24}] := x$
151	if $i \geq j$ goto 162		

## BBS PARA EL FRAGMENTO DEL *quicksort*

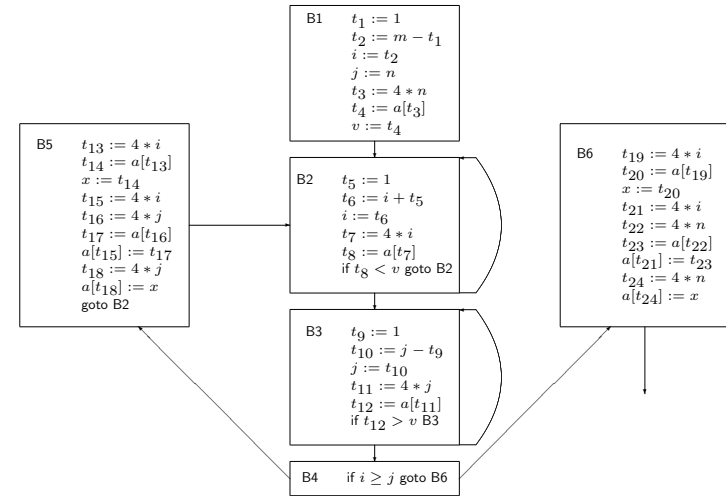
$B_1 \rightarrow$	132 $t_1 := 1$	$B_5 \rightarrow$	152 $t_{13} := 4 * i$
	133 $t_2 := m - t_1$		153 $t_{14} := a[t_{13}]$
	134 $i := t_2$		154 $x := t_{14}$
	135 $j := n$		155 $t_{15} := 4 * i$
	136 $t_3 := 4 * n$		156 $t_{16} := 4 * j$
	137 $t_4 := a[t_3]$		157 $t_{17} := a[t_{16}]$
	138 $v := t_4$		158 $a[t_{15}] := t_{17}$
$B_2 \rightarrow$	139 $t_5 := 1$		159 $t_{18} := 4 * j$
	140 $t_6 := i + t_5$		160 $a[t_{18}] := x$
	141 $i := t_6$		161 goto 139
	142 $t_7 := 4 * i$	$B_6 \rightarrow$	162 $t_{19} := 4 * i$
	143 $t_8 := a[t_7]$		163 $t_{20} := a[t_{19}]$
	144 if $t_8 < v$ goto 139		164 $x := t_{20}$
$B_3 \rightarrow$	145 $t_9 := 1$		165 $t_{21} := 4 * i$
	146 $t_{10} := j - t_9$		166 $t_{22} := 4 * n$
	147 $j := t_{10}$		167 $t_{23} := a[t_{22}]$
	148 $t_{11} := 4 * j$		168 $a[t_{21}] := t_{23}$
	149 $t_{12} := a[t_{11}]$		169 $t_{24} := 4 * n$
$B_4 \rightarrow$	150 if $t_{12} > v$ goto 145		170 $a[t_{24}] := x$
	151 if $i \geq j$ goto 162		

## GRAFO DE FLUJO

### ALGORITMO: Construcción del Grafo de Flujo

- Sea  $B_1, B_2, \dots, B_N$  el conjunto de BB, donde  $B_1$  es el bloque inicial;
- Para todo  $i := 1..N$  hacer
  - Si  $B_i$  no finaliza en ningún salto entonces crea\_arco ( $B_i \rightarrow B_{i+1}$ );
  - Si  $B_i$  finaliza en un salto incondicional a  $B_j$  entonces crea\_arco ( $B_i \rightarrow B_j$ );
  - Si  $B_i$  finaliza en un salto condicional a  $B_j$  entonces
    - crea\_arco ( $B_i \rightarrow B_{i+1}$ ) y crea\_arco ( $B_i \rightarrow B_j$ );
- Hecho.

## GF PARA EL FRAGMENTO DEL *quicksort*



## ACTIVIDAD DE LAS VARIABLES EN UN BB

- **Definidas:** aparecen en la parte izquierda de alguna asignación
- **Usadas:** aparecen en la parte derecha de alguna asignación o se emplean en el cómputo de una instrucción (p.ej. saltos condicionales);
- **Activas** en un BB  $B_i$ : si  $B_i$  está en un camino del GF desde el BB donde se definen hasta el BB donde se usan. Las variables activas son:
  - **de entrada:**  $entrada(i) = usadas^{\dagger}(i) \cup (salida(i) - definidas(i))$
  - **de salida:**  $salida(i) = \cup_{k \in sucesor(i)} entrada(k)$

$\dagger$  solo las usadas que no han sido definidas previamente en el BB.

## ACTIVIDAD DE LAS VARIABLES DEL FRAGMENTO DEL *quicksort*

	Definidas	Usadas	Activas	
			de entrada	de salida
B1	$t_1, \dots, t_4$ $i, j, v$	$t_1, \dots, t_4$ $m, n, a$	$m, n, a$	$i, a, v, j, n$
B2	$t_5, \dots, t_8$ $i$	$t_5, \dots, t_8$ $i, a, v$	$i, a, v, j, n$	$i, a, v, j, n$
B3	$t_9, \dots, t_{12}$ $j$	$t_9, \dots, t_{12}$ $j, a, v$	$j, a, v, i, n$	$i, j, a, v, n$
B4		$i, j$	$i, j, a, n, v$	$i, a, j, n, v$
B5	$t_{13}, \dots, t_{18}$ $x, a$	$t_{13}, \dots, t_{18}$ $i, a, j, x$	$i, a, j, v, n$	$i, a, v, j, n$
B6	$t_{19}, \dots, t_{24}$ $x, a$	$t_{19}, \dots, t_{24}$ $i, n, a, x$	$i, a, n$	—

## TRANSFORMACIONES ALGEBRAICAS

### 1. Simplificaciones algebraicas

#### > Expresiones de identidad

$x + 0 = 0 + x = x$        $x * 1 = 1 * x = x$        $x / 1 = x$   
 $x - 0 = x$        $0 - x = -x$        $-(-a) = a$   
 $false \text{ or } x = x$        $true \text{ and } x = x$        $not \ not \ x = x$

#### > Propiedades algebraicas

$x + (y + z) = (x + y) + z$        $x + y = y + x$   
 $x * (y * z) = (x * y) * z$        $x * y = y * x$   
 $x * (y + z) = (x * y) + (x * z)$

### 2. Reducción de intensidad de los operadores

$x^2 = x * x$        $x * 2 = x + x$        $x / 2 = x * 0.5$

## TRANSFORMACIONES ALGEBRAICAS

### 3. Cálculo previo de constantes

$y = 3$        $\Rightarrow$        $t_1 \leftarrow 3$        $\Rightarrow$        $y \leftarrow 3$   
 $x = 2 + y + 5 + z$        $y \leftarrow t_1$        $x \leftarrow 10 + z$   
 $t_2 \leftarrow 2$   
 $t_3 \leftarrow t_2 + y$   
 $t_4 \leftarrow 5$   
 $t_5 \leftarrow t_3 + t_4$   
 $t_6 \leftarrow t_5 + z$   
 $x \leftarrow t_6$

Precauciones en la aplicación de transformaciones algebraicas:

```
14 i <- 0
15 i <- i + 1
16 if i < N goto 15
```

## TRANSFORMACIONES QUE PRESERVAN LA ESTRUCTURA

### 1. Eliminación de subexpresiones comunes

$a \leftarrow b + c$        $\Rightarrow$        $a \leftarrow b + c$   
 $b \leftarrow a - d$        $b \leftarrow a - d$   
 $c \leftarrow b + c$        $c \leftarrow b + c$   
 $d \leftarrow a - d$        $d \leftarrow b$

### 2. Propagación de copias

$x \leftarrow t_1$        $\Rightarrow$        $x \leftarrow t_1$   
 $t_2 \leftarrow y$        $t_2 \leftarrow y$   
 $t_3 \leftarrow x$        $t_3 \leftarrow t_1$   
 $t_4 \leftarrow t_2$        $t_4 \leftarrow y$

### 3. Renombrado de variables temporales

## TRANSFORMACIONES QUE PRESERVAN LA ESTRUCTURA

### 4. Eliminación de código inactivo

$\forall s = (a \leftarrow \alpha) \in B_k : a \notin \text{de\_salida}(B_k) \wedge a \text{ no se usa en el resto del } B_k$

$\Rightarrow$  eliminar  $s$  de  $B_k$  y  $a$  de la tabla de actividad de las variables de  $B_k$

$a \leftarrow b + c$        $\Rightarrow$        $a \leftarrow b + c$        $\Rightarrow$        $a \leftarrow b * c$   
 $x \leftarrow a * k$        $a \leftarrow b * c$        $x \leftarrow b * k$   
 $a \leftarrow b * c$        $x \leftarrow b * k$   
 $x \leftarrow b * k$

### 5. Intercambio de instrucciones independientes adyacentes

$a \leftarrow b + c$        $\Rightarrow$        $x \leftarrow b * k$   
 $x \leftarrow b * k$        $a \leftarrow b + c$

## GRAFO DIRIGIDO ACÍCLICO

### Grafo Dirigido Acíclico

1. Hay un nodo (hoja) para cada uno de los valores iniciales de los operandos: variables o constantes.
2. Hay un nodo (interno) para cada una de las operaciones.
3. Cada nodo tiene asociado una lista (posiblemente vacía) de variables con el mismo valor semántico.

## ALGORITMO DE CONSTRUCCIÓN DEL GDA

Dado un BB  $B_i = (I, E, S, U, D)$ , su GDA se calcula como:

Para toda instrucción  $(x \leftarrow y \text{ op } z) \in I$

// Construcción de los nodos de los operandos:

$\text{nodo}(y) = \text{buscaHoja}(y)$ ;  $\text{nodo}(z) = \text{buscaHoja}(z)$ ;

// Construcción del nodo de la operación:

$n = \text{buscaSubarbol}(op, y, z)$ ,

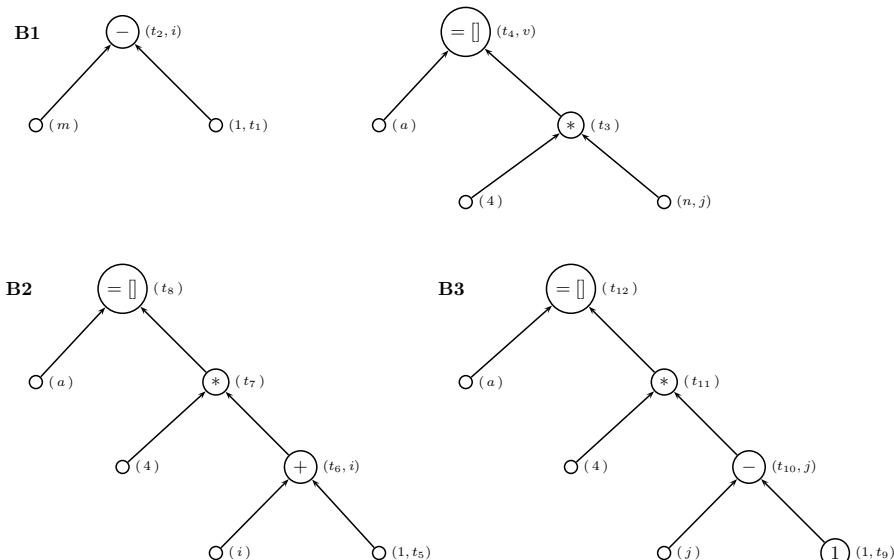
// Actualización de las listas:

- eliminar, si existe,  $x$  de la lista de variables del nodo  $\text{nodo}(x)$ ,
- añadir  $x$  a la lista de variables del nodo  $n$  y hacer  $\text{nodo}(x) = n$

**nodo**( $x$ ): enlaza con el nodo del último uso de  $x$ ; y por tanto  $x$  estará en su lista de variables.

**buscaHoja**( $x$ ): si el nodo existe, devuelve su enlace; y si no, lo crea y añade  $x$  a su lista de variables devolviendo su enlace. **buscaSubarbol**( $f(y, z)$ ): busca un subárbol cuya estructura y valores coincidan con el argumento de la operación; si existe, devuelve el enlace de la raíz del subárbol; si no lo crea y devuelve su índice.

## GDA PARA EL FRAGMENTO DEL *quicksort* 1/2



## ALGORITMO DE CONSTRUCCIÓN DEL GDA

### Consideraciones:

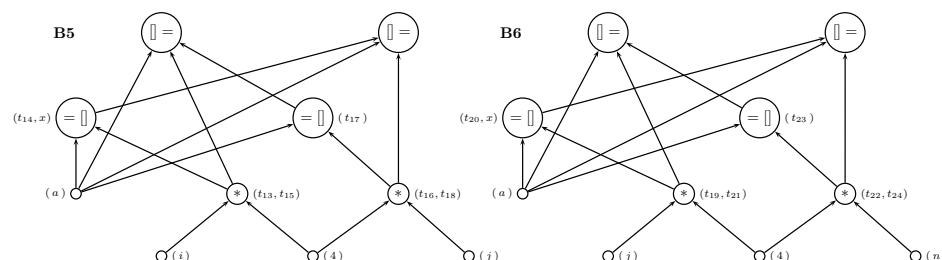
#### > Copia:

$y = x$  Se añade  $y$  a la lista de variables del nodo de  $x$

#### > Representación de los array

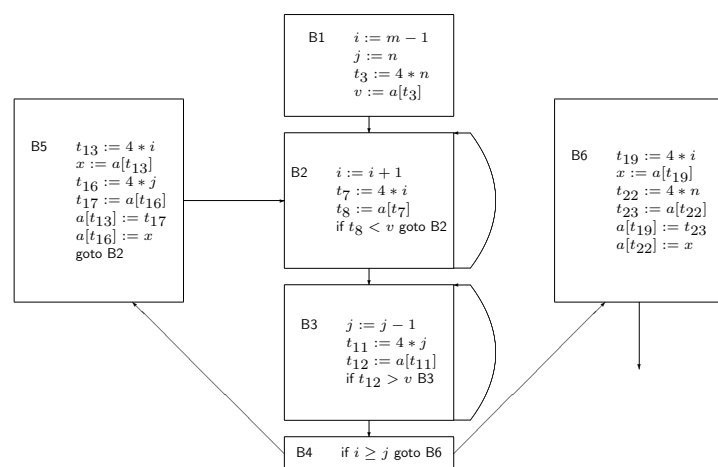
```
x <- a[i]
a[j] <- y
z <- a[i]
```

1. Para las instrucciones del tipo  $x \leftarrow a[i]$ , se crea un nodo interno  $= []$ , con dos hijos a los valores iniciales de  $a$ ,  $i$ . La variable  $x$  será la etiqueta del nuevo nodo.
2. Para las instrucciones del tipo  $a[j] \leftarrow y$ , se crea un nodo interno  $[] =$ , con tres hijos a los valores de  $a$ ,  $j$ ,  $y$ . Este nodo no tiene etiqueta. La creación de este nodo **bloquea** todos los nodos cuyos valores dependan de  $a$ . Un nodo bloqueado no puede recibir nuevas etiquetas; es decir, no puede ser una subexpresión común.

GDA PARA EL FRAGMENTO DEL *quicksort* 2/2

## ALGORITMO DE RECONSTRUCCIÓN APARTIR DE LOS GDA

1. La reconstrucción debe seguir el orden topológico.
2. La reconstrucción del código no es determinista.
3. Se debe de respetar el orden en caso de redefinición de variables.
4. Si todos los operandos de una instrucción son constantes o se pueden aplicar simplificaciones algebraicas entonces actualizar la lista de variables de su nodo interno.
5. Si existe un nodo (raíz) que no es argumento de otros nodos y su lista de variables está vacía entonces eliminar la estructura que define dicho nodo.
6. Si hay un nodo con más de una variable en su lista entonces:
  - a) si no hay ninguna variable activa a la salida del BB, se elige una de ellas al azar,
  - b) si una sola variable activa a la salida del BB, ésta es la que se elige,
  - c) si hay más de una variable activa a la salida del BB, se deberán generar tantas instrucciones de copia como sea necesario.

GF DEL *quicksort* DESPUÉS DE LAS OPTIMIZACIONES LOCALES:<sup>†</sup>

† eliminación de subexpresiones comunes, propagación de copias y eliminación de código inactivo

## OPTIMIZACIONES GLOBALES

- Transformaciones que optimizan los saltos:

a)	...	...		...	...
	goto 14	goto 28		if a<b goto 14	if a<b goto 28
	...	...		...	...
	14 goto 28	14 goto 28		14 goto 28	14 goto 28
	==>			==>	
<hr/>					
b)	...	...		...	...
	goto 14	14 ...		if a<b goto 14	if a>=b goto 28
	14 ...	...		goto 28	14 ...
	==>			==>	
				14 ...	

➤ Bucles naturales

1. un bucle natural debe tener un solo punto de entrada (encabezamiento),
2. en un bucle natural debe haber al menos una forma de iterar el bucle.

## BUCLES NATURALES

Dado un GF se pueden definir:

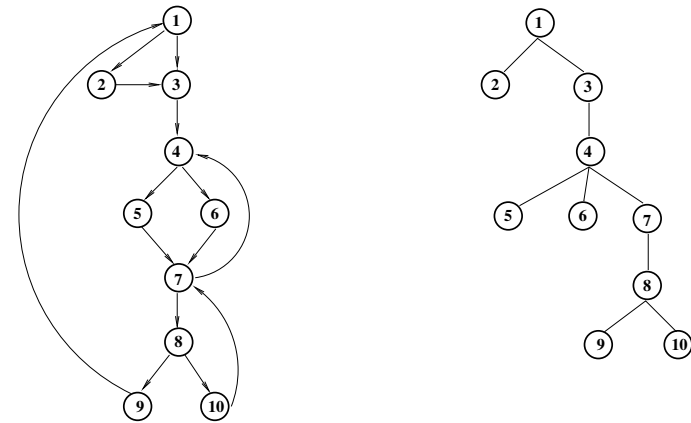
- **Árbol de dominación.**- Sean  $i, j$  dos nodos del GF,  $i$  **domina\_a**  $j$  si todo camino desde el nodo inicial del GF a  $j$  pasa por  $i$ .

Todo nodo se domina a sí mismo.

El árbol de dominación representa la relación de dominación entre los nodos del GF.

- **Arista de retroceso** es una arista  $j \rightarrow i$  en el GF tal que  $i$  **domina\_a**  $j$ .
- **Bucle natural.**- Dado una arista de retroceso  $j \rightarrow i$ , su bucle natural se define como el conjunto de todos los nodos que pueden alcanzar a  $j$  sin hacerlo a través de  $i$ . Los nodos  $i, j$  también pertenecen al bucle.

## EJEMPLO DE BUCLES NATURALES



$7 \rightarrow 4$  4, 5, 6, 7, 8, 10

$10 \rightarrow 7$  7, 8, 10

$9 \rightarrow 1$  1, 2, 3, 4, 5, 6, 7, 8, 9, 10

## EXTRACCIÓN DE CÓDIGO INVARIANTE A UN BUCLE

- **Detección del código invariante**

Marcar aquellas instrucciones cuyos operandos sean: todos constantes, tengan sus definiciones fuera del bucle, o tengan una sola definición dentro del bucle y esté marcada como invariante.

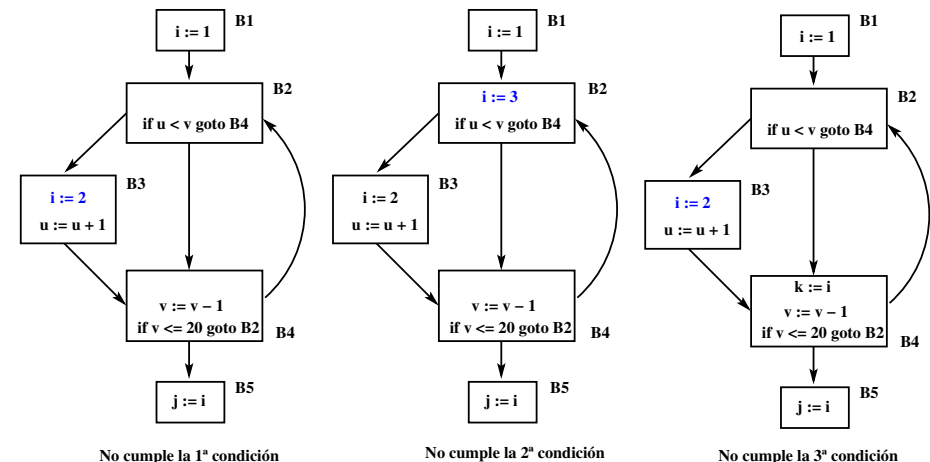
- **Traslado de código invariante**

para toda instrucción  $s$  que defina  $x$ , marcada como invariante, que cumpla:

1. que esté en un bloque que domine todas las salidas del bucle (o que no esté activa a la salida del bucle),
2. que  $x$  no se defina en otro lugar en el bucle,
3. que todos los usos de  $x$  en el bucle solo puedan ser alcanzados por la definición de  $x$  en la instrucción  $s$ .

hacer trasladar la instrucción  $s$  al bloque de preencabezamiento del bucle.

## EJEMPLOS DE TRASLADO ERRÓNEO DE CÓDIGO INVARIANTE



## VARIABLES DE INDUCCIÓN EN UN BUCLE

### > Variable de inducción

Durante la ejecución del bucle,  $x$  cambia su valor incrementándose o decrementándose en una constante.

### > Variable básica de inducción

Su única definición en el bucle es de la forma  $i = i + c$

	Aristas de retroceso	Bucle natural
	$B_2 \rightarrow B_1$	$\{B_1, B_2\}$
$B_1 \rightarrow$	$t_4 = i + 10$ $t_5 = t_4 * 2$ $t_6 = a[t_5]$ if $i \geq M$ goto $B_3$	
$B_2 \rightarrow$	$i = i - 1$ goto $B_1$	
$B_3 \rightarrow$	...	

	interacciones	ecuaciones
$i$	9 8 7	$i = 1 \cdot i + 0$
$t_4$	19 18 17	$t_4 = 1 \cdot i + 10$
$t_5$	38 36 34	$t_5 = 2 \cdot i + 20$

## VARIABLES DE INDUCCIÓN EN UN BUCLE

### > ALGORITMO: detección de variables de inducción

- > Detectar todas las variables básicas de inducción  $i$  y generar su terna  $i(i, 1, 0)$ .
- > Buscar todas las variables  $k$  con una sola definición en el bucle, de la forma:  
 $k = j * b$  ó  $k = j + b$ . Siendo  $b$  invariante al bucle o una constante.

(1) $j$ es variable básica de inducción	$j(j, 1, 0)$	$k = j * b$ $k(j, b, 0)$	$k = j + b$ $k(j, 1, b)$
(2) $j$ es variable de inducción <sup>†</sup>	$j(i, c, d)$	$k(i, c * b, d * b)$	$k(i, c, d + b)$

<sup>†</sup> Si ( $\exists$  definición de  $i$  entre la de  $j$  y la de  $k$ )  $\wedge$  ( $\exists$  definición de  $j$  que alcance la de  $k$ )

$B_1 \rightarrow$	$t_4 = i + 10$ $t_5 = t_4 * 2$ $t_6 = a[t_5]$ if $i \geq M$ goto $B_3$	
$B_2 \rightarrow$	$i = i - 1$ goto $B_1$	
$B_3 \rightarrow$	...	

Variable básica de inducción	$i(i, 1, 0)$
Variable de inducción	$t_4(i, 1, 10)$
Variable de inducción	$t_5(i, 2, 20)$

## REDUCCIÓN DE INTENSIDAD EN LAS VARIABLES DE INDUCCIÓN

Para toda variable de inducción  $j(i, c, d)$  hacer:

1. crear una variable temporal  $s$  con  $s(i, c, d)$ ,
2. sustituir las asignaciones a  $j$  por  $j = s$ ,
3. para toda  $i = i + n$  hacer añadir a continuación  $s = s + c * n$ ,
4. añadir en el preencabezamiento del bucle:  $s = c * i$  y  $s = s + d$ .

## ELIMINACIÓN DE VARIABLES DE INDUCCIÓN

1. Para toda variable de inducción no activa a la salida del bucle, y solo usada en su propia definición o en una instrucción de copia hacer:
  - > Eliminar la(s) instrucción(es) de su definición en el bucle y en el pre-encabezamiento.
2. Para toda variable de inducción  $i$  usada solo para calcular otra variable de inducción o saltos condicionales hacer:
  - > seleccionar una  $j(i, c, d)$ , con un cálculo simple,
  - > modificar cada comprobación de  $i$  para utilizar  $j$ .  
 $\text{if } i \text{ oprel } x \text{ goto } B \Rightarrow \text{if } j \text{ oprel } c * x + d \text{ goto } B$
  - > borrar todas las variables de inducción eliminadas en el bucle.
  - > ir al paso 1
3. Para toda variable de inducción  $j$  en la que se introdujo una instrucción  $j = s$  en el algoritmo anterior hacer:
  - > comprobar que no se define  $s$  entre  $j = s$  y los usos de  $j$ ,
  - > sustituir usos de  $j$  por los de  $s$ ,
  - > borrar  $j = s$ , si  $j$  no está activa fuera del bucle.



## EJEMPLO COMPLETO 1/8

```

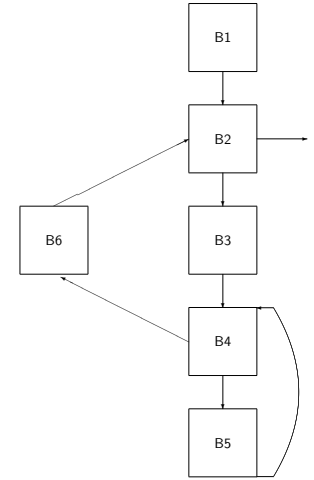
int A [10][10][10]; /* talla de enteros=2 */
int B [10][10]
. . .
for (int j=0; j<10; j++)
  for (int k=0; k<10; k++) B[7][k] = B[7][k] + A[7][j][k]

```

100 $t_1 := 0$	112 $t_9 := B[t_8]$	124 $t_{21} := t_{20} * 2$
101 $j := t_1$	113 $t_{10} := 7$	125 $B[t_{21}] := t_{17}$
102 $t_2 := 10$	114 $t_{11} := t_{10} * 10$	126 $t_{22} := 1$
103 if $j \leq t_2$ goto 134	115 $t_{12} := t_{11} + j$	127 $t_{23} := k + t_{22}$
104 $t_3 := 0$	116 $t_{13} := t_{12} * 10$	128 $k := t_{23}$
105 $k := t_3$	117 $t_{14} := t_{13} + k$	129 goto 107
106 $t_4 := 10$	118 $t_{15} := t_{14} * 2$	130 $t_{24} := 1$
107 if $k \leq t_4$ goto 130	119 $t_{16} := A[t_{15}]$	131 $t_{25} := j + t_{24}$
108 $t_5 := 7$	120 $t_{17} := t_9 + t_{16}$	132 $j := t_{25}$
109 $t_6 := t_5 * 10$	121 $t_{18} := 7$	133 goto 103
110 $t_7 := t_6 + k$	122 $t_{19} := t_{18} * 10$	134
111 $t_8 := t_7 * 2$	123 $t_{20} := t_{19} + k$	

## EJEMPLO COMPLETO 2/8

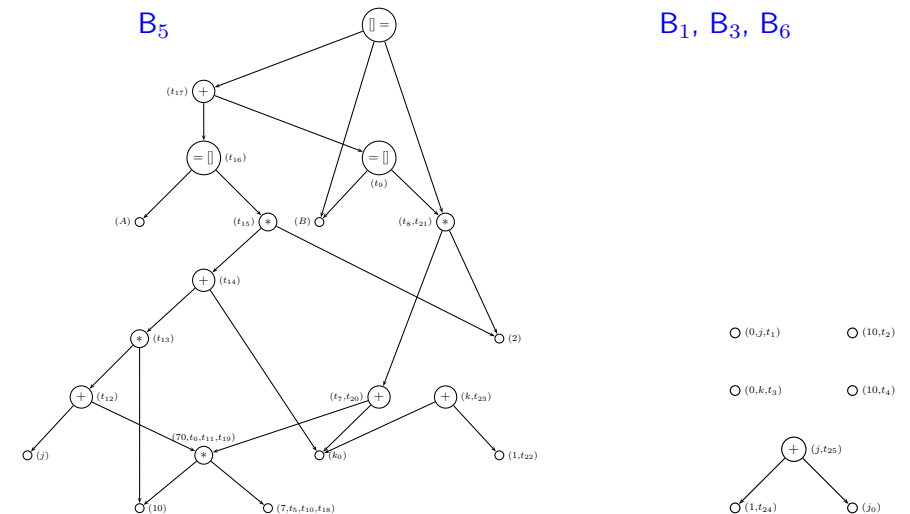
$B_1 \rightarrow 100$	$t_1 := 0$	117 $t_{14} := t_{13} + k$
101	$j := t_1$	118 $t_{15} := t_{14} * 2$
102	$t_2 := 10$	119 $t_{16} := A[t_{15}]$
$B_2 \rightarrow 103$	if $j \leq t_2$ goto 134	120 $t_{17} := t_9 + t_{16}$
$B_3 \rightarrow 104$	$t_3 := 0$	121 $t_{18} := 7$
105	$k := t_3$	122 $t_{19} := t_{18} * 10$
106	$t_4 := 10$	123 $t_{20} := t_{19} + k$
$B_4 \rightarrow 107$	if $k \leq t_4$ goto 130	124 $t_{21} := t_{20} * 2$
$B_5 \rightarrow 108$	$t_5 := 7$	125 $B[t_{21}] := t_{17}$
109	$t_6 := t_5 * 10$	126 $t_{22} := 1$
110	$t_7 := t_6 + k$	127 $t_{23} := k + t_{22}$
111	$t_8 := t_7 * 2$	128 $k := t_{23}$
112	$t_9 := B[t_8]$	129 goto 107
113	$t_{10} := 7$	$B_6 \rightarrow 130$
114	$t_{11} := t_{10} * 10$	$t_{24} := 1$
115	$t_{12} := t_{11} + j$	131 $t_{25} := j + t_{24}$
116	$t_{13} := t_{12} * 10$	132 $j := t_{25}$
		133 goto 103



## EJEMPLO COMPLETO 3/8

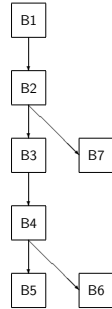
	Definidas	Usadas	Activas	
			de entrada	de salida
B1	$t_1, t_2, j$	$t_1$	$A, B$	$j, t_2, A, B$
B2	—	$t_2, j$	$j, t_2, A, B$	$A, B, j, t_2$
B3	$t_3, t_4, k$	$t_3$	$A, B, j, t_2$	$t_4, k, A, B, j, t_2$
B4	—	$t_4, k$	$t_4, k, B, A, j, t_2$	$A, B, k, j, t_2, t_4$
B5	$t_5 - t_{23}, B, k$	$t_5 - t_{23}, k, j, A, B$	$k, j, A, B, t_4, t_2$	$k, t_4, A, B, j, t_2$
B6	$t_{24}, t_{25}, j$	$t_{24}, t_{25}, j$	$j, t_2, A, B$	$j, t_2, A, B$

## EJEMPLO COMPLETO 4/8



## EJEMPLO COMPLETO 5/8

$B_1 \rightarrow$	$j := 0$ $t_2 := 10$
$B_2 \rightarrow$	if $j \leq t_2$ goto $B_7$
$B_3 \rightarrow$	$k := 0$ $t_4 := 10$
$B_4 \rightarrow$	if $k \leq t_4$ goto $B_6$
$B_5 \rightarrow$	$t_7 := 70 + k$ $t_8 := t_7 * 2$ $t_9 := B[t_8]$ $t_{12} := 70 + j$ $t_{13} := t_{12} * 10$ $t_{14} := t_{13} + k$ $t_{15} := t_{14} * 2$ $t_{16} := A[t_{15}]$ $t_{17} := t_9 + t_{16}$ $B[t_8] := t_{17}$ $k := k + 1$ goto $B_4$
$B_6 \rightarrow$	$j := j + 1$ goto $B_2$



Aristas de retroceso	Bucle natural
$B_6 \rightarrow B_2$	$\{B_2, B_3, B_4, B_5, B_6\}$
$B_5 \rightarrow B_4$	$\{B_4, B_5\}$

## EJEMPLO COMPLETO 6/8

Variable básica de inducción del bucle interno  $\{B_4, B_5\}$   $k(k, 1, 0)$

Variables de inducción  $t_7(k, 1, 70)$   $s_7$ ;  $t_{14}(k, 1, t_{13})$   $s_{14}$   
 $t_8(k, 2, 140)$   $s_8$ ;  $t_{15}(k, 2, 2 * t_{13})$   $s_{15}$

$Pre \rightarrow$	$k := 0$ $t_4 := 10$ $t_{12} := 70 + j$ $t_{13} := t_{12} * 10$ $s_7 := k + 70$ $s_8 := 2 * k$ $s_{14} := k + t_{13}$ $s_{15} := k * 2$ $t_{26} := 2 * t_{13}$ $s_{15} := s_{15} + t_{26}$ $t_{27} := 2 * t_4$ $t_{27} := t_{27} + 140$ if $s_8 \leq t_{27}$ goto $B_6$	$\leftarrow 1$ $\leftarrow 1$ $\leftarrow 1$ $\leftarrow 1$ $\leftarrow 1$ $\leftarrow 1$ $\leftarrow 2$
$B_4 \rightarrow$		

$B_5 \rightarrow$	$t_7 := s_7$ $t_8 := s_8$ $t_9 := B[t_8]$ $t_{14} := s_{14}$ $t_{15} := s_{15}$ $t_{16} := A[t_{15}]$ $t_{17} := t_9 + t_{16}$ $B[t_8] := t_{17}$ $k := k + 1$ $s_7 := s_7 + 1$ $s_8 := s_8 + 2$ $s_{14} := s_{14} + 1$ $s_{15} := s_{15} + 2$ goto $B_4$	$\leftarrow 1$ $\leftarrow 3$ $\leftarrow 3$ $\leftarrow 1$ $\leftarrow 3$ $\leftarrow 3$ $\leftarrow 2$ $\leftarrow 1$ $\leftarrow 1$ $\leftarrow 1$ $\leftarrow 1$
-------------------	--	--

## EJEMPLO COMPLETO 7/8

$B_1 \rightarrow$	$j := 0$ $t_2 := 10$
$B_2 \rightarrow$	if $j \leq t_2$ goto $B_7$
$B_3 \rightarrow$	$t_{12} := 70 + j$ $t_{13} := t_{12} * 10$ $s_8 := 140$ $s_{15} := 2 * t_{13}$ $t_{27} := 160$
$B_4 \rightarrow$	if $s_8 \leq t_{27}$ goto $B_6$
$B_5 \rightarrow$	$t_9 := B[s_8]$ $t_{16} := A[s_{15}]$ $t_{17} := t_9 + t_{16}$ $B[s_8] := t_{17}$ $s_8 := s_8 + 2$ $s_{15} := s_{15} + 2$ goto $B_4$
$B_6 \rightarrow$	$j := j + 1$ goto $B_2$

Bucle externo  $\{B_2, B_3, B_4, B_5, B_6\}$

Instrucciones invariantes

$s_8 \leftarrow$  NO es invariante

$t_{27} \leftarrow$  si es invariante

Variable básica de inducción

$s_8$  y  $s_{15} \leftarrow$  NO  $\exists$  otra definición en el bucle

$j(j, 1, 0)$

Variables de inducción

$t_{12}(j, 1, 70)$   $s_{12}$

$t_{13}(j, 10, 700)$   $s_{13}$

## EJEMPLO COMPLETO 8/8

$B_1 \rightarrow$	$j := 0$ $t_2 := 10$ $t_{27} := 160$ $s_{12} := j + 70$ $s_{13} := j * 10$ $s_{13} := s_{13} + 700$ $t_{28} := t_2 * 10$ $t_{28} := t_{28} + 700$	$\leftarrow 1$
$B_2 \rightarrow$	if $s_{13} \leq t_{28}$ goto $B_7$	$\leftarrow 2$
$B_3 \rightarrow$	$t_{12} := s_{12}$ $t_{13} := s_{13}$ $s_8 := 140$ $s_{15} := 2 * t_{13}$	$\leftarrow 1$ $\leftarrow 3$
$B_4 \rightarrow$	...	
$B_5 \rightarrow$	...	
$B_6 \rightarrow$	$j := j + 1$ $s_{12} := s_{12} + 1$ $s_{13} := s_{13} + 10$ goto $B_2$	$\leftarrow 2$ $\leftarrow 1$

$B_1 \rightarrow$	$t_{27} := 160$ $s_{13} := 700$ $t_{28} := 800$
$B_2 \rightarrow$	if $s_{13} \leq t_{28}$ goto $B_7$
$B_3 \rightarrow$	$s_8 := 140$ $s_{15} := 2 * s_{13}$
$B_4 \rightarrow$	if $s_8 \leq t_{27}$ goto $B_7$
$B_5 \rightarrow$	$t_9 := B[s_8]$ $t_{16} := A[s_{15}]$ $t_{17} := t_9 + t_{16}$ $B[s_8] := t_{17}$ $s_8 := s_8 + 2$ $s_{15} := s_{15} + 2$ goto $B_4$
$B_6 \rightarrow$	$s_{13} := s_{13} + 10$ goto $B_2$