7. Optimización de Código Intermedio

7.1. Introducción al problema de la optimizaión

- 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

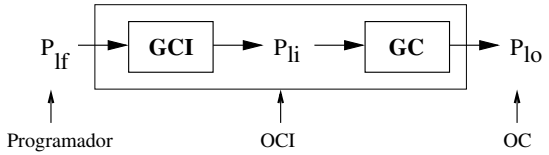
```
a:= x ↑ 2 + y   ⇒  t1 <- 2                ⇒  t2 <- x * x
                   t2 <- x ↑ t1              a <- t2 + y
                   t3 <- t2 + y
                   a <- t3
```

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

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.*


```
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		100	$t_2 := i * 10$
101	$t_2 := i * t_1$	101	$t_2 := i * 10$	101	$t_3 := t_2 + k$
102	$t_3 := t_2 + k$	102	$t_3 := t_2 + k$	102	$t_4 := t_3 * 2$
103	$t_4 := t_3 * 2$	103	$t_4 := t_3 * 2$	103	$t_5 := A[t_4]$
104	$t_5 := A[t_4]$	104	$t_5 := A[t_4]$	104	if $t_5 \geq M$ goto 107
105	if $t_5 < M$ goto 107	105	if $t_5 < M$ goto 107	105	$t_4 := t_4 - 2$
106	goto 111	106	goto 111	106	goto 103
107	$t_6 := 1$	107		107	
108	$t_7 := k - t_6$	108	$k := k - 1$		
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	$t_1 := 1$
133	$t_2 := m - t_1$
134	$i := t_2$
135	$j := n$
136	$t_3 := 4 * n$
137	$t_4 := a[t_3]$
138	$v := t_4$
139	$t_5 := 1$
140	$t_6 := i + t_5$
141	$i := t_6$
142	$t_7 := 4 * i$
143	$t_8 := a[t_7]$
144	if $t_8 < v$ goto 139
145	$t_9 := 1$
146	$t_{10} := j - t_9$
147	$j := t_{10}$
148	$t_{11} := 4 * j$
149	$t_{12} := a[t_{11}]$
150	if $t_{12} > v$ goto 145
151	if $i \geq j$ goto 162
152	$t_{13} := 4 * i$
153	$t_{14} := a[t_{13}]$
154	$x := t_{14}$
155	$t_{15} := 4 * i$
156	$t_{16} := 4 * j$
157	$t_{17} := a[t_{16}]$
158	$a[t_{15}] := t_{17}$
159	$t_{18} := 4 * j$
160	$a[t_{18}] := x$
161	goto 139
162	$t_{19} := 4 * i$
163	$t_{20} := a[t_{19}]$
164	$x := t_{20}$
165	$t_{21} := 4 * i$
166	$t_{22} := 4 * n$
167	$t_{23} := a[t_{22}]$
168	$a[t_{21}] := t_{23}$
169	$t_{24} := 4 * n$
170	$a[t_{24}] := x$

BBs para el fragmento del *quicksort*

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

Grafo de Flujo

ALGORITMO: Construcción del Grafo de Flujo

- Sea $B_1, B_2, \ldots B_N$ el conjuto 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` $(B_i \rightarrow B_{i+1})$;

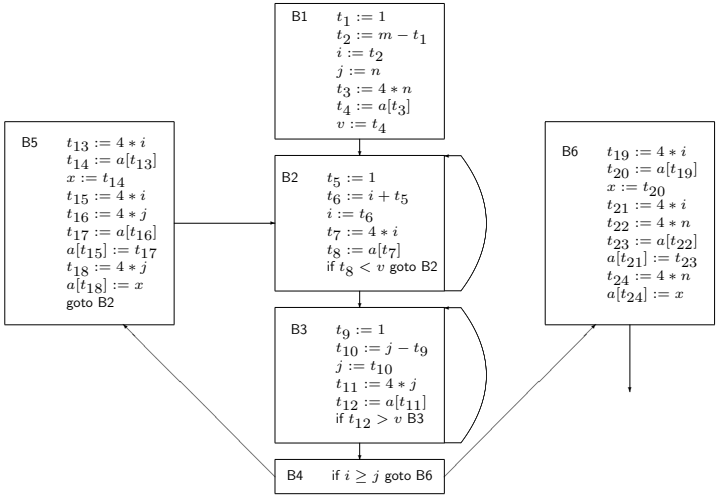
  Si B_i finaliza en un salto incondicional a B_j entonces `crea_arco` $(B_i \rightarrow B_j)$;

  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: $salida(i) = \cup_{k \in sucesor(i)} entrada(k)$

† 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				
			1	2	3	4	5	1	2	3	4	5
B1	$t_1, \ldots, t_4$ i, j, v	$t_1, \ldots, t_4$ m, n, a	m, n, a	–	–	–	–	i, a, v	j	–	n	–
B2	$t_5, \ldots, t_8$ i	$t_5, \ldots, t_8$ i, a, v	i, a, v	j	–	n	–	i, a, v, j	–	n	–	–
B3	$t_9, \ldots, t_{12}$ j	$t_9, \ldots, t_{12}$ 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	$t_{13}, \ldots, t_{18}$ x, a	$t_{13}, \ldots, t_{18}$ i, a, j, x	i, a, j	v	–	–	n	i, a, v	j	–	n	–
B6	$t_{19}, \ldots, t_{24}$ x, a	$t_{19}, \ldots, t_{24}$ 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 or x = x         true 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                  ⇒   t1 <- 3          ⇒   y <- 3
x = 2 + y + 5 + z          y <- t1              x <- 10 + z
                           t2 <- 2
                           t3 <- t2 + y
                           t4 <- 5
                           t5 <- t3 + t4
                           t6 <- t5 + z
                           x <- t6
```

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 subexpreseiones comunes

```
a <- b + c      ⇒   a <- b + c
b <- a - d          b <- a - d
c <- b + c          c <- b + c
d <- a - d          d <- b
```

2. Propagación de copias

```
x <- t1         ⇒   x <- t1
t2 <- y             t2 <- y
t3 <- x             t3 <- t1
t4 <- t2            t4 <- y
```

3. Renombrado de variables temporales

Transformaciones que preservan la estructura

4. Eliminación de código inactivo

$\forall$ `s = (a <- α)` $\in B_k : a \notin$ `de_salida`$(B_k) \wedge$ `a` 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 deB_k

```
a <- b + c   ⇒   a <- b + c   ⇒   a <- b * c
x <- a * k       a <- b * c       x <- b * k
a <- b * c       x <- b * k
x <- b * k
```

5. Intercambio de instrucciones independientes adyacentes

```
a <- b + c      ⇒   x <- b * k
x <- b * k          a <- 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 $(x \leftarrow y\ op\ z) \in I$

// Construcción de los nodos de los operandos:

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

// Construcción del nodo de la operación:

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

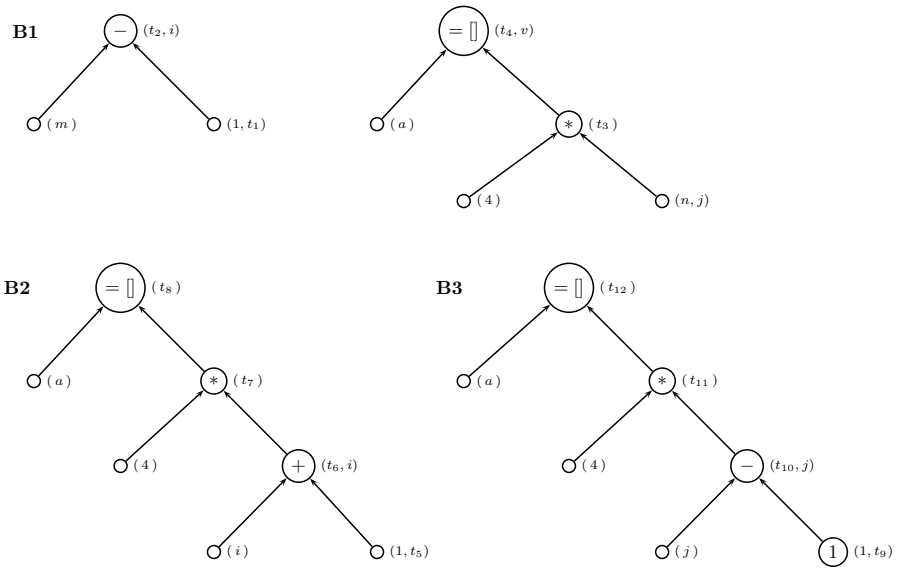
// Actualización de las listas:

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

---

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:

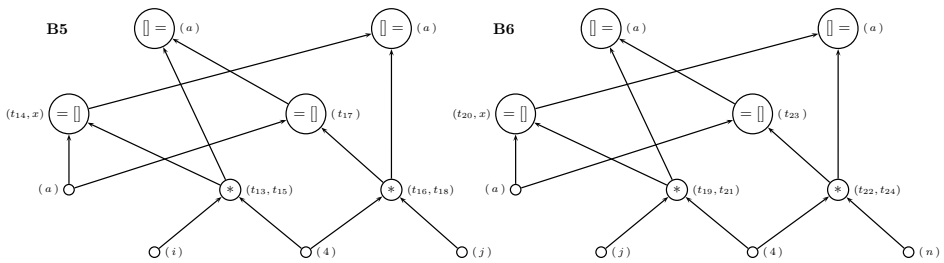
$y = x$ Se añade y a la lista de variables del nodo de x

➢ Representación de los `array`

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

1. Para las instruccines del tipo `x <- 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 instruccines del tipo `a[j] <- y`, se crea un nodo interno `[]=`, con dos hijos a los valores de `j, y`. Este nodo tiene la etiqueta `a`, indicando que ese objeto ha sido modificado y, por tanto, **bloqueado**. Un nodo bloqueado no puede recibir nuevas etiquetas; es decir, no puede ser una subexpresión común.

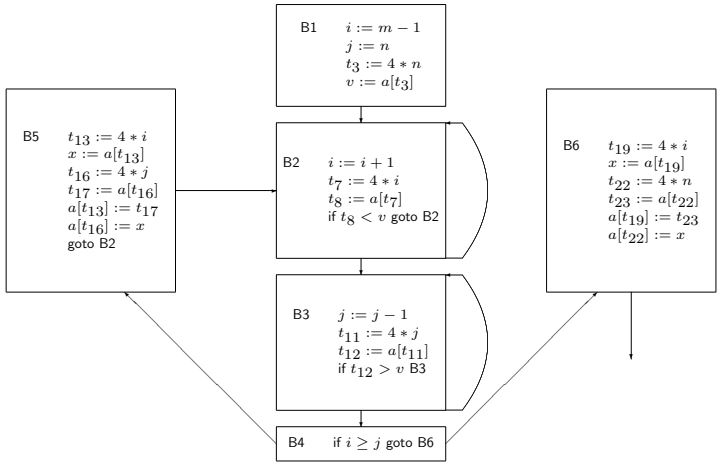
GDA para el fragmento del *quicksort* 2/2



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 (raiz) 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:†



† 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
-------------------------------------------------------------------
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**.

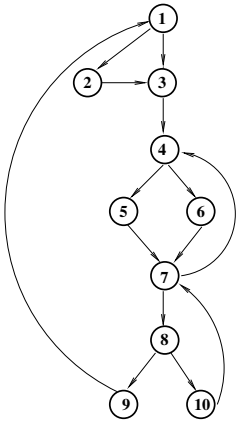
  Todo nodo se domina a sí mismo.

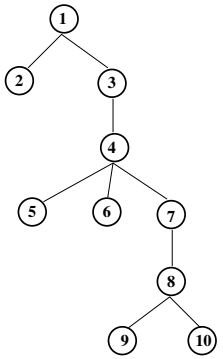
  El árbol de dominación representa la relación de dominación entre los nodos del GF.

- 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

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

10 $\rightarrow$ 7 7, 8, 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.

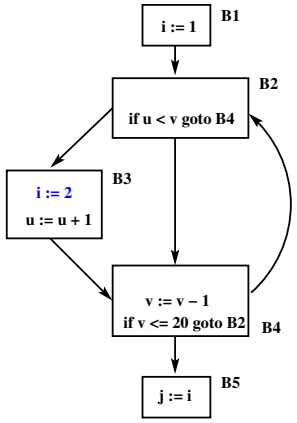
- Traslado de código invariante

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

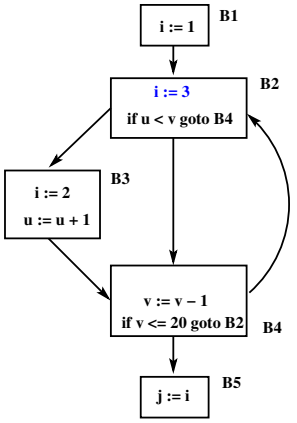
  1. que esté en un bloque que domine todas las salidas del bucle (o que no esté activa a la salida del bucle),
  2. que x no se defina en otro lugar en el bucle,
  3. que todos los usos de x en el bucle solo puedan ser alcanzados por la definición de x en la instrucción s.

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

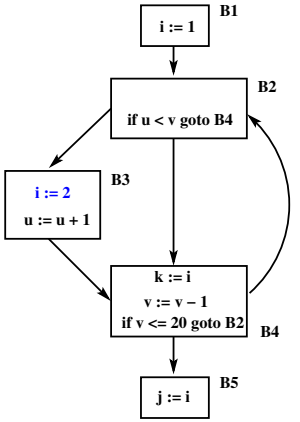
Ejemplos de traslado erróneo de código invariante



No cumple la 1ª condición



No cumple la 2ª condición



No cumple la 3ª condición

Variables de inducción en un bucle

➢ Variable de inducción

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

➢ Variable básica de inducción

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

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

$B_2 \rightarrow$ $i = i - 1$
goto B_1

$B_3 \rightarrow$ ...

Aristas de retroceso	Bucle natural
$B_2 \rightarrow B_1$	$\{B_1, B_2\}$

	interaciones			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	$t_5 = 2 \cdot i + 20$

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

Variables de inducción en un bucle

ALGORITMO: detección de variables de inducción

1. Detectar todas las variables básicas de inducción i y generar su terna $i(i, 1, 0)$.
2. Buscar todas las variables k con una sola definición en el bucle, de la forma:

$$k = j * b \quad \text{o} \quad k = j + b\,.$$

Donde b es una constante o invariante al bucle y j es una variable de inducción:

		$k = j * b$	$k = j + b$
a) j es variable básica de inducción	$j(j, 1, 0)$	$k(j, b, 0)$	$k(j, 1, b)$
b) j es variable de inducción†	$j(i, c, d)$	$k(i, c * b, d * b)$	$k(i, c, d + b)$

† Si ($\nexists$ definición de i entre la de j y la de k) $\wedge$ ($\nexists$ definición de j que alcance la de k)

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 definión 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.
     `if` i `oprel` x `goto` B $\Rightarrow$ `if` j `oprel` $c * x + d$ `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.

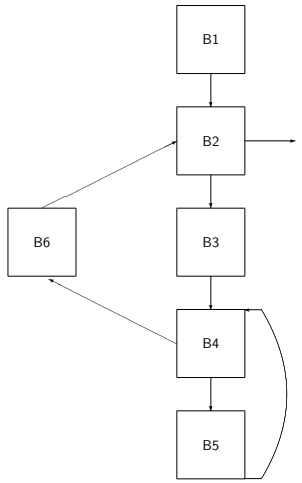
Ejemplo completo 1/8

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

Ejemplo completo 2/8

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

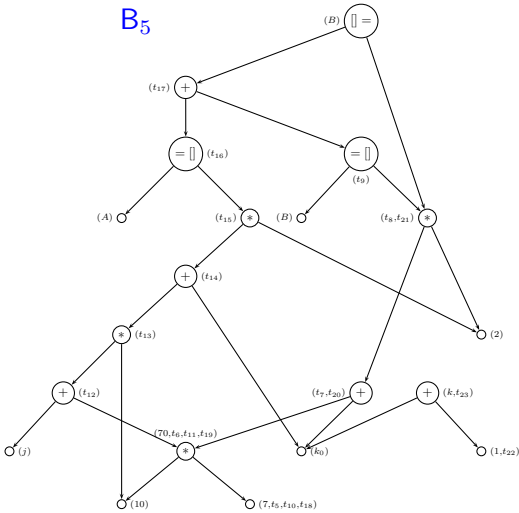


Ejemplo completo 3/8

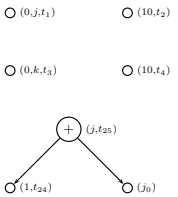
	Definidas	Usadas	Activas	
			de entrada	de salida
B1	t_1, t_2, j	t_1	A, B	j, t_2, A, B
B2	–	t_2, j	j, t_2, A, B	A, B, j, t_2
B3	t_3, t_4, k	t_3	A, B, j, t_2	t_4, k, A, B, j, t_2
B4	–	t_4, k	t_4, k, B, A, j, t_2	A, B, k, j, t_2, t_4
B5	$t_5 - t_{23}, B, k$	$t_5 - t_{23}, k, j, A, B$	k, j, A, B, t_4, t_2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

B_5

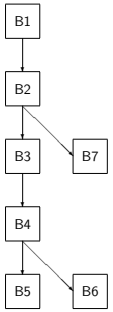


B_1, B_3, B_6



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	$\rightarrow$	$t_7 := 70 + k$
		$t_8 := t_7 * 2$
		$t_9 := B[t_8]$
	$* \rightarrow$	$t_{12} := 70 + j$
	$* \rightarrow$	$t_{13} := t_{12} * 10$
		$t_{14} := t_{13} + k$
		$t_{15} := t_{14} * 2$
		$t_{16} := A[t_{15}]$
		$t_{17} := t_9 + t_{16}$
		$B[t_8] := t_{17}$
		$k := k + 1$
		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}

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

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

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 $\{B_2, B_3, B4, B_5, B_6\}$

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

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	