

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

Facultad de Ciencias

Integrantes:

Adrián Aguilera Moreno

Sebastián Alejandro Gutierrez Medina



Compiladores

Tarea 06

Varios lenguajes de programación, por ejemplo C, tienen definido el enunciado **switch**:

```
switch E
  begin
    case V1 : S1
    case V2 : S2
    ...
    case Vn1 : Sn1
  default: Sn
end
```

Describe una forma de traducir este enunciado a código de tres direcciones (puedes usar saltos y condicionales). Explica y justifica que el código de tres direcciones propuesto respeta el mismo comportamiento que el **switch**.

Solución: Especificamos a S_i y V_j , con $i \in [1, \dots, n]$ y $j \in [1, \dots, n-1]$, cómo una expresión en código de 3 direcciones (sino, basta con obtener el código de 3 direcciones para S_i y V_j). A continuación se muestra el código en tres direcciones de la expresión **switch**:

- Para un case con V_j y $S_i \neq S_n$ tenemos que

```
recorrer( $V_j$ ,  $n$ ):
  if ( $V_j$ ) goto  $S'_i$ ;                                con  $i = j$ .
  else: recorrer( $V_j$ ,  $n+1$ );
```

con n inicialmente en 0.

```
 $S'_i$ :
   $S_i$ ;
  break;
```

- Para el caso default tenemos que

```
default:
  goto  $S'_n$ ;
```

```
 $S'_n$ :
   $S_n$ ;
  break;
```

- Ahora, construyendo el switch tenemos que

```
switch ( $V_j$ ):
  if ( $j < n$ ) goto recorrer( $V_j$ , 0);
  else: goto default;
```

La definición anterior funciona, pues de manera recursiva comprobamos que siempre que $j < n$ buscamos el “salto” a la instrucción S'_i adecuado a j tal que $j = i$. Si $j > n$ entonces se acciona la instrucción default que es justamente cómo funciona la instrucción **switch**.

Considera el siguiente código:

```
int mul(int x, int y) {  
  if (x) return y - (0 - mul(x + 1, y));  
  else return 0; }
```

Proporciona una traducción a RTL suponiendo que ya se han seleccionado las instrucciones. Deberás mostrar las traducciones intermedias de subexpresiones del código.

Utilizando la tabla de traducción entre una representación intermedia lineal y las instrucciones de arquitectura MIPS, genera el código máquina para la siguiente secuencia:

```

d := c + 8
a := a + b/$^{last}$/
M[d/$^{last}$/] := a
IF a < c THEN L1 ELSE L2
LABEL L1

```

$t := r_s + k,$ $r_t := M[t^{last}]$	lw $r_t, k(r_s)$
$r_t := M[r_s]$	lw $r_t, 0(r_s)$
$r_t := M[k]$	lw $r_t, k(RO)$
$t := r_s + k,$ $M[t^{last}] := r_t$	sw $r_t, k(r_s)$
$M[r_s] := r_t$	sw $r_t, 0(r_s)$
$M[k] := r_t$	sw $r_t, k(RO)$
$r_d := r_s + r_t$	add r_d, r_s, r_t
$r_d := r_t$	add r_d, RO, r_t
$r_d := r_s + k$	addi r_d, r_s, k
$r_d := k$	addi r_d, RO, k
GOTO label	j label
IF $r_s = r_t$ THEN $label_t$ ELSE $label_f$, LABEL $label_f$	beq $r_s, r_t, label_t$ $label_f:$
IF $r_s = r_t$ THEN $label_t$ ELSE $label_f$, LABEL $label_t$	bne $r_s, r_t, label_f$ $label_t:$
IF $r_s = r_t$ THEN $label_t$ ELSE $label_f$	beq $r_s, r_t, label_t$ j $label_f$
IF $r_s < r_t$ THEN $label_t$ ELSE $label_f$, LABEL $label_f$	slt r_d, r_s, r_t bne $r_d, RO, label_t$ $label_f:$
IF $r_s < r_t$ THEN $label_t$ ELSE $label_f$, LABEL $label_t$	slt r_d, r_s, r_t beq $r_d, RO, label_f$ $label_t:$
IF $r_s < r_t$ THEN $label_t$ ELSE $label_f$	slt r_d, r_s, r_t bne $r_d, RO, label_t$ j $label_f$
LABEL label	label:

Secuencia Original	Código Maquina
$a := a + b^{last}$	<i>lw</i> \$1, <i>a</i>
<i>lw</i> \$2, b^{last}	<i>add</i> \$3, \$1, \$2
$d := c + 8$	
$M[d^{last}] := a$	<i>lw</i> \$4, <i>c</i>
<i>sw</i> \$3, 8(\$4)	
<i>IF</i> $a < c$ <i>THEN</i> <i>L1</i> <i>ELSE</i> <i>L2</i>	
<i>LABEL</i> <i>L1</i>	<i>L1</i> : <i>slt</i> \$5, \$3, \$4
	<i>beq</i> \$5, <i>R0</i> , <i>L2</i>

Hasta 1.5pts extra. Determina el código de tres direcciones de la siguiente expresión,

$$(a \text{ SUB } b) \text{ MOD } ((\text{MINUS } c) \text{ SUB } d)$$

usando las reglas semánticas siguientes.

Pueden omitir la explicación de la creación del árbol de sintaxis abstracta, pero hay que explicar los pasos del análisis semántico.

Synthesized Attributes

<i>E.code</i> :	Code sequence evaluating <i>E</i>
<i>E.sym</i> :	Symbol representing value of <i>E</i>
addop.op :	addition operator: ADD or SUB
mulop.op :	multiplication operator: MUL , DIV , or MOD

Grammar Rules Semantic Rules

$E \rightarrow E^1 \text{ addop } E^2$	<i>E.sym</i> := newtemp(); <i>AddInst</i> := new Inst(addop.op , <i>E.sym</i> , <i>E¹.sym</i> , <i>E².sym</i>); <i>E.code</i> := <i>E¹.code</i> + <i>E².code</i> + <i>AddInst</i> ;
$E \rightarrow E^1 \text{ mulop } E^2$	<i>E.sym</i> := newtemp(); <i>MulInst</i> := new Inst(mulop.op , <i>E.sym</i> , <i>E¹.sym</i> , <i>E².sym</i>); <i>E.code</i> := <i>E¹.code</i> + <i>E².code</i> + <i>MulInst</i> ;
$E \rightarrow \text{UnaryOp } E^1$	<i>E.sym</i> := newtemp(); <i>UnaryInst</i> := new Inst(<i>UnaryOp.op</i> , <i>E.sym</i> , <i>E¹.sym</i>); <i>E.code</i> := <i>E¹.code</i> + <i>UnaryInst</i> ;

Synthesized Attributes

<i>UnaryOp.op</i> :	Unary operator: PLUS , MINUS , NOT
id.name :	Identifier name
num.sym :	Literal symbol holding number value

Grammar Rules Semantic Rules

$E \rightarrow (E^1)$	<i>E.sym</i> := <i>E¹.sym</i> ; <i>E.code</i> := <i>E¹.code</i> ;
$E \rightarrow \text{id}$	<i>E.sym</i> := <i>idTable.lookup(id.name)</i> ; <i>E.code.first</i> := <i>E.code.last</i> = 0;
$E \rightarrow \text{num}$	<i>E.sym</i> := num.sym ; <i>E.code.first</i> := <i>E.code.last</i> = 0;
$\text{UnaryOp} \rightarrow \text{addop}$	if addop.op = ADD then <i>UnaryOp.op</i> := PLUS else <i>UnaryOp.op</i> := MINUS
$\text{UnaryOp} \rightarrow \text{not}$	<i>UnaryOp.op</i> := NOT

Solución: A continuación se muestra el código en tres direcciones para la expresión dada

Código 3 direcciones	Reglas Semánticas
$t_1 = a - b$	$E \rightarrow (E^1) \rightarrow (E^{11} \text{ addop } E^{12}) \rightarrow (E^{11}.\text{sym} \text{ addop.op } E^{12}.\text{sym}) \rightarrow (\text{id.name SUB id.name})$
$t_2 = -c$	$E \rightarrow (E^1) \rightarrow (\text{UnaryOp } E^{11}) \rightarrow (\text{MINUS } E^{11}.\text{sym}) \rightarrow (\text{MINUS id.name})$
$t_3 = t_2 - d$	$E \rightarrow (E^1) \rightarrow (E^{11} \text{ addop } E^{12}) \rightarrow (E^{11}.\text{sym} \text{ addop.op } E^{12}.\text{sym}) \rightarrow (\text{id.name SUB id.name})$
$t_4 = t_1 \text{ mod } t_3$	$E \rightarrow (E^1) \rightarrow (E^{11} \text{ addop } E^{12}) \rightarrow (E^{11}.\text{sym} \text{ addop.op } E^{12}.\text{sym}) \rightarrow (\text{id.name MOD id.name})$