# TRABAJO 2: EJECUCION DE LENGUAJE PROGRAMACION DE PSEUDOCODIGO EN ESPAÑOL

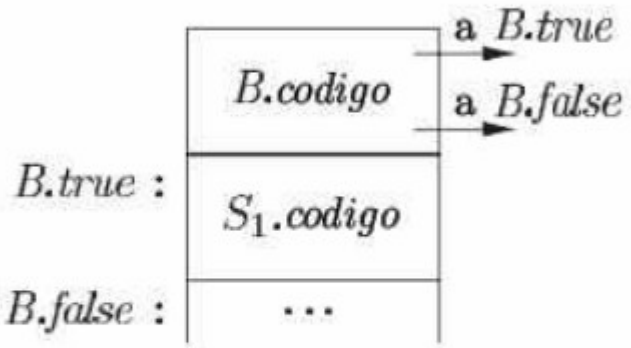
Universidad Nacional de Ingeneria

Alumnos : Tomas J. Casas Rodriguez
Luis Carhuarica Aguilar

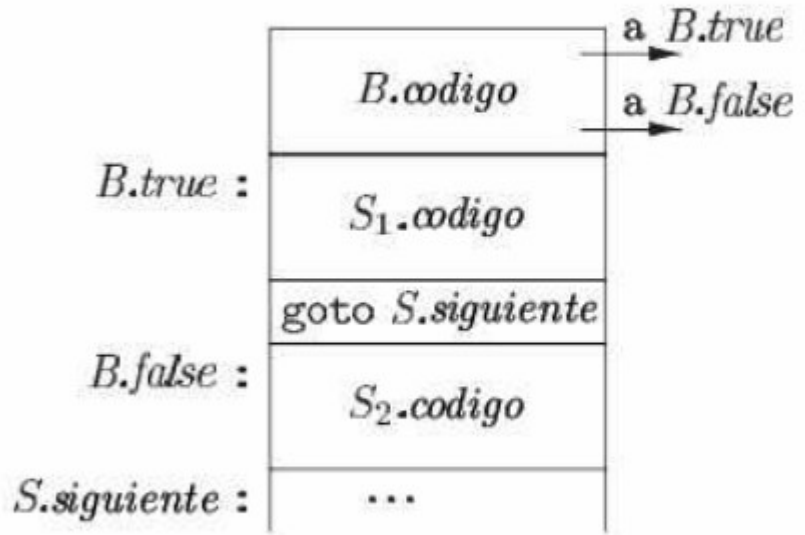
Prof .: Jaime Osorio Ubaldo

# IF_ELSE

La traducción de **if** $(B)$ $S_1$ consiste en $B.codigo$ seguida de $S_1.codigo$, como se muestra en la figura . Dentro de $B.codigo$ hay saltos con base en el valor de $B$. Si $B$ es verdadera, el control fluye hacia la primera instrucción de $S_1.codigo$, y si $B$ es falsa, el control fluye a la instrucción que sigue justo después de $S_1.codigo$.




(a) if




(b) if-else

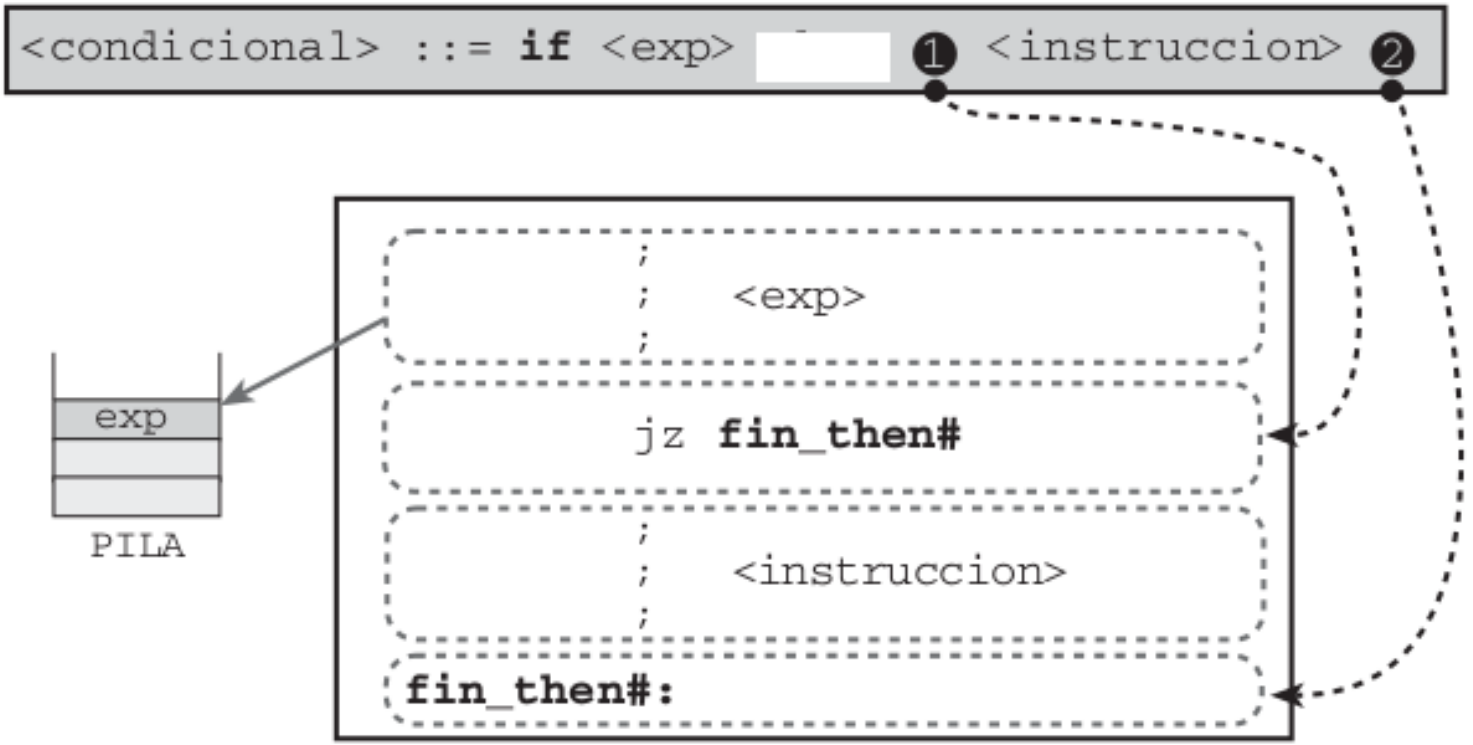



**Figura** Generación de código para la instrucción `if-then`.

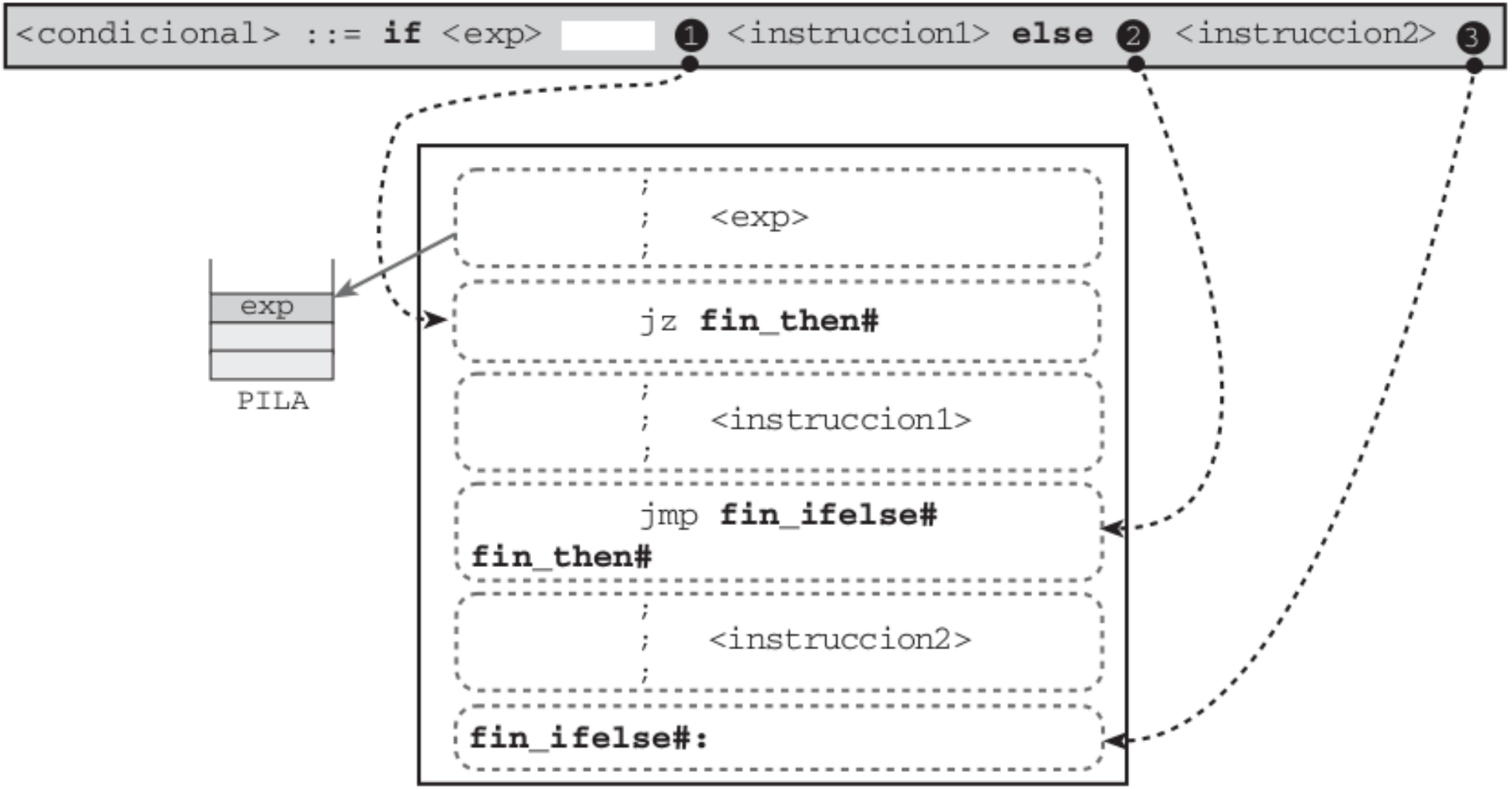



**Figura** Generación de código para la instrucción `if-then-else`.

```
INICIOPROGRAMA principal
    Z:a,b;
    S: r;
    R: m;
    ......;
    ......;
    SI (condicion)
        .........;
    SISINO (condicion)
        ...........;
    SINO
        ...........;
    FINSI
        ...........;
FINPROGRAMA
```

- (jmp) a la etiqueta fin_ifelse# para evitar que se ejecuten las instrucciones de la parte else

  tras ejecutar las instrucciones de la parte then.
- Además se genera una línea que define la eti-

  queta fin_then#.
- Por último, la tercera acción semántica genera únicamente una línea que contiene la etiqueta fin_ifelse#.

# FOR-WHILE

## Instrucciones iterativas (bucles)

Casi todos los lenguajes de programación proporcionan instrucciones iterativas con una sintaxis parecida a la de la siguiente regla de producción:

```
<bucle> ::= while <expresion> do <instruccion>
```

Estas instrucciones presentan las mismas peculiaridades que las condicionales, tanto en lo relativo a las etiquetas, como en lo referente a las comprobaciones de tipo de la condición.

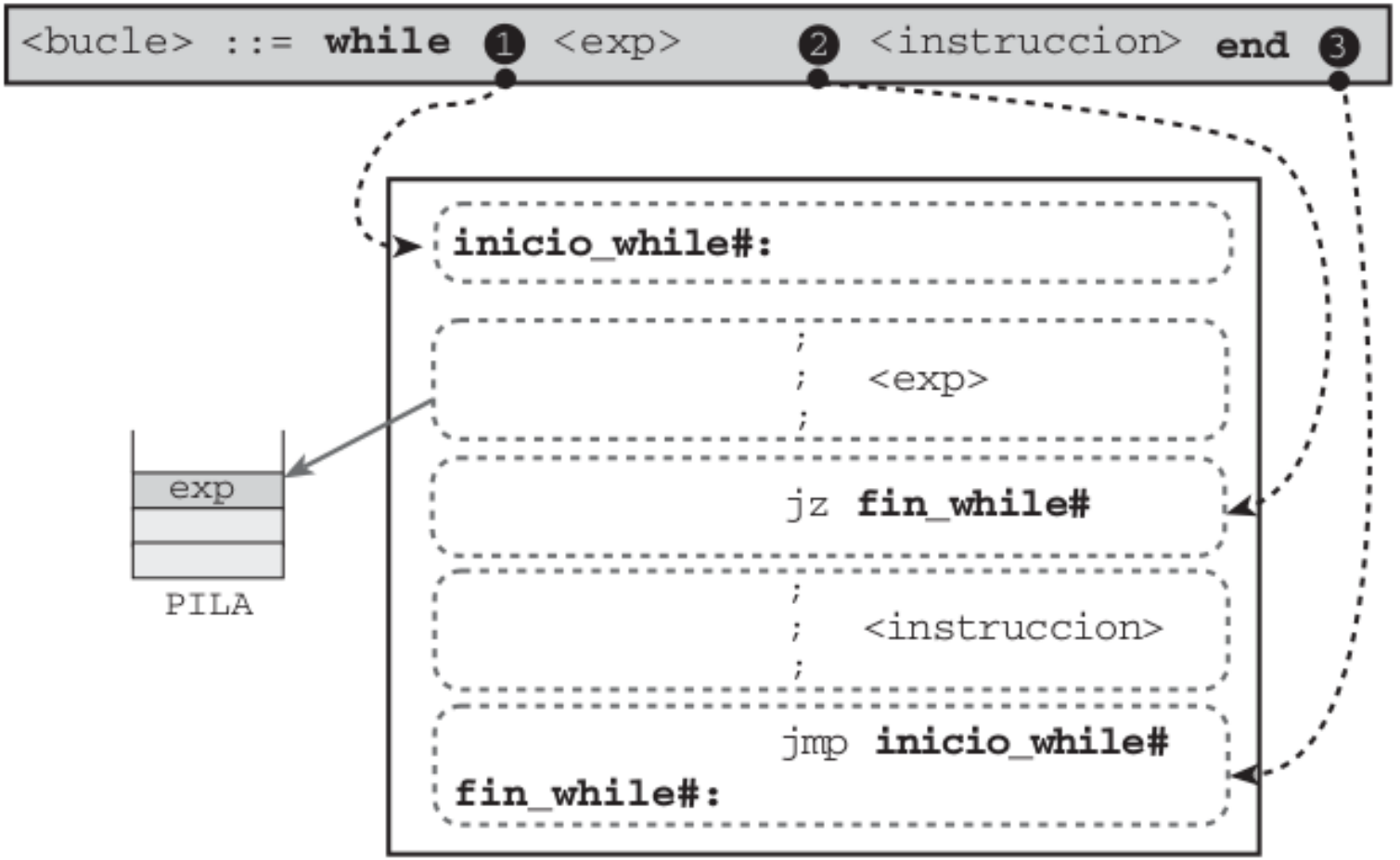



**Figura** Generación de código para la instrucción `while`.

-La primera acción semántica genera únicamente una línea que contiene la etiqueta inicio_while#. La segunda acción semántica genera un código exactamente igual al generado por la primera acción semántica de las Figuras IF-ELSE.

-El efecto de este código es salir del bucle si el valor de la expresión es igual a 0, es decir, si la expresión es falsa.
La tercera acción semántica genera una instrucción de salto incondicional (jmp) a la etiqueta inicio_while#, para continuar con la siguiente iteración del bucle.

-Además genera una línea que contiene la etiqueta fin_while#.

```
    MIENTRAS(condicion)
          .........;
          .........;
    FINMIENTRAS


tmp :=n;
    MIENTRAS(tmp>0)
              ..........;
              ..........;
            tmp := tmp-i;
    FINMIENTRAS
```

# FOR

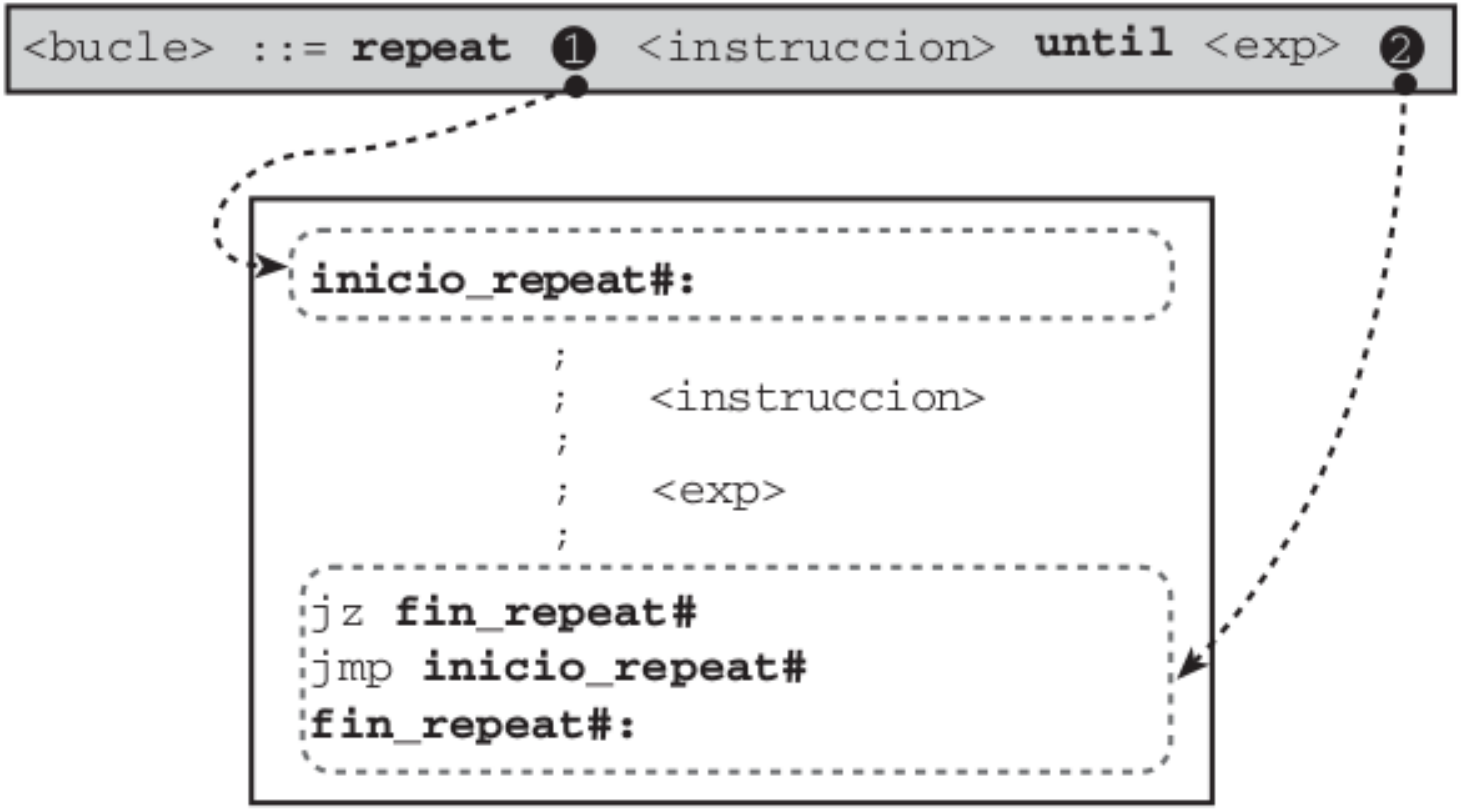



**Figura** Generación de código para la instrucción `repeat`.

# FOR

-La primera acción semántica genera únicamente una línea que contiene la etiqueta
inicio_repeat#.

-La segunda acción semántica genera un código similar al generado por la primera acción semántica de las Figuras IL-ELSE

- El efecto de este código es salir del bucle si el valor de la expresión es true, es decir, si la expresión es verdadera. En caso contrario, se ejecuta un salto incondicional (jmp) a la etiqueta inicio_repeat#, para continuar con la siguiente iteración del bucle.

- Además se genera una línea que contiene la etiqueta fin_repeat#.

```
    PARA (inicializacion; condicion;incremento o decremento)
        .........;
        .........;
    FINPARA


PARA (i := 2; i < n; i := i + 1)
        ........;
        ........;
FINPARA
```

```
INICIOPROGRAMA principal
   /*Declaracion Variables*/
    Z: a,n;
    n := 6;
   ......................;
   ......................;
    a :=nombreFuncion(parametros);
FINPROGRAMA

INICIOFUNCION  nombreFuncion(parametros)
    Z: mp;
     ..............;
    ..............;
    RETORNAR tmp;
FINFUNCI
```

```
maquina@JDC-HP-ProBook-6460b:~$ cd trabajo2
maquina@JDC-HP-ProBook-6460b:~/trabajo2$ ./main
INICIOPROGRAMA principal
    Z:a,b;
    a := 3;
    b := 1;
    SI (a > b)
        b := 7;
    SISINO (a < b)
        b := 0;
    SINO
        SI (a == b)
            b := 5;
        SINO
            b := 8;
        FINSI
    FINSI
        IMPRIMIR b;
FINPROGRAMA
programa valido!
PROGRAMA: principal
```

```
Tabla de Codigos
OPCODE  OP1     OP2     OP3
begpr   principal
asign   a       3
asign   b       1
label   $begif_1
mayor   $tmp_0  a       b
salt0   $tmp_0  $else_1_1
asign   b       7
salta   $endif_1
label   $else_1_1
menor   $tmp_1  a       b
salt0   $tmp_1  $else_1_2
asign   b       0
salta   $endif_1
label   $else_1_2
label   $begif_2
igual   $tmp_2  a       b
salt0   $tmp_2  $else_2_3
asign   b       5
salta   $endif_2
label   $else_2_3
asign   b       8
label   $endif_2
label   $endif_1
print   b
endpr   principal
7
PROGRAMA: principal

Stack: main
----------------------
```

```
maquina@JDC-HP-ProBook-6460b:~/trabajo2$ ./main
INICIOPROGRAMA principal
    Z: a,n;
    n := 6;
    a := factorial(n);
    IMPRIMIR a;
FINPROGRAMA

INICIOFUNCION factorial(Z:n)
    Z: i,tmp;
    tmp := 1;
    PARA (i := 2; i < n; i := i + 1)
        tmp := tmp*i;
    FINPARA
    RETORNAR tmp;
FINFUNCION
programa valido!
PROGRAMA: principal
```

```
Tabla de Codigos
OPCODE  OP1      OP2      OP3
begfx   factorial
asign   tmp      1
asign   i        2
label   $begfor_1
menor   $tmp_0   i        n
salt0   $tmp_0   $endfor_1
salt0   $tmpfor_1         $bfor_1
sumar   $tmp_1   i        1
asign   i        $tmp_1
label   $bfor_1
asign   $tmpfor_1         1
multi   $tmp_2   tmp      i
asign   tmp      $tmp_2
salta   $begfor_1
label   $endfor_1
return  tmp
endfx   factorial
720
PROGRAMA: principal
```

```
maquina@JDC-HP-ProBook-6460b:~/trabajo2$ ./main
INICIOPROGRAMA principal
    Z: a,n;
    n := 6;
    a :=otraFuncion(n);
FINPROGRAMA

INICIOFUNCION otraFuncion(Z:n)
    Z: i,tmp,x;
    tmp :=n;
    MIENTRAS(tmp>0)
        PARA (i := 0; i < n; i := i + 1)
          IMPRIMIR tmp;
          tmp := tmp-i;
          x:=tmp;
        FINPARA
    FINMIENTRAS
    x:=x*x;
     IMPRIMIR x;
    RETORNAR tmp;
FINFUNCION
programa valido!
PROGRAMA: principal
```

```
Tabla de Codigos
OPCODE  OP1      OP2      OP3
begfx   otraFuncion
asign   tmp      n
mayor   $tmp_0   tmp      0
asign   i        0
label   $begfor_1
menor   $tmp_1   i        n
salt0   $tmp_1   $endfor_1
salt0   $tmpfor_1         $bfor_1
sumar   $tmp_2   i        1
asign   i        $tmp_2
label   $bfor_1
asign   $tmpfor_1         1
print   tmp
resta   $tmp_3   tmp      i
asign   tmp      $tmp_3
asign   x        tmp
salta   $begfor_1
label   $endfor_1
multi   $tmp_4   x        x
asign   x        $tmp_4
print   x
return  tmp
endfx   otraFuncion
```

```
return   tmp
endfx    otraFuncion
6
6
5
3
0
-4
-9
225
PROGRAMA: principal
```

- GRACIAS……..!