29.- A continuación se presentan dos secuencias de código escrito en lenguaje C que hacen lo mismo: calcular la sumatoria de i desde 0 hasta n. El de la izquierda sigue un método recursivo mientras que el de la derecha sigue un procedimiento iterativo.

algoritmo recursivo	**algoritmo iterativo**

`int sumatorio (int n, int acumulado)`
`{`
`  if (n>0) return sumatorio (n - 1, acumulado + n);`
`  else return acumulado;`
`}`
	`int sumatorio (int n)`
	`{`
	`  int i, acumulado = 0;`
	`  for (i = n; i > 0; i--)`
	`  {`
	`  acumulado = acumulado + n;`
	`  }`
	`  return acumulado;`
	`}`

$\sum_{i=0}^{n} i = \sum_{i=0}^{n-1} i + n$	$\sum_{i=0}^{n} i = 0+1+2+\cdots+n$

Aunque ambas secuencias funcionan correctamente y generan el mismo resultado, la iterativa tiene un mejor rendimiento. ¿Por qué? Describa cómo quedan las instrucciones en Assembler y cuál es su comportamiento en tiempo de ejecución.

La iterativa tiene un mejor rendimiento debido a que en la recursiva, una función se llama a sí misma repetidamente, lo que implica ocupar más espacio de memoria y un mayor tiempo de procesador. Hay veces que al hacer un programa recursivo, el mismo no termina debido al espacio de memoria requerido, mientras que el iterativo termina en unos segundos(esto pasa cuando las iteraciones o las llamadas a sí mismo son muy altas)

Iterativo

```
!Este programa suma n numeros
!Uso de registros: %r1 - indice a sumar
!                  %r3 - Acumulado
             .begin
n            .equ          20
             .org 2048                        !El programa empieza en la direcc. 2048
main:         andcc        %r3, %r0, %r3     !Pone a 0 en el %r3
              add          %r0, n, %r1       !%r1 = n

for:          andcc        %r1, %r1, %r0     !Chequea elementos restantes
              be done                        !Si no hay mas elementos termina
              addcc        %r1, -4, %r1      !Actualiza en indice a sumar
              addcc        %r3, %r1, %r3     !Suma el nuevo indice a %r3
              ba           for               !Vuelve al for

done:         jmpl         %r15 + 4, %r0     !Vuelve al proceso invocante
             .end
```

Recursivo

```
!Este programa suma n numeros
!Uso de registros: %r1 - indice a sumar
!                  %r3 - Acumulado
!                  %r5 - guardar %r15
              .begin
              .org 2048                          !El programa empieza en la direcc. 2048
main:          andcc       %r3, %r0, %r3       !Pone a 0 en el %r3
               andcc       %r5, %r0, %r5
               ld          [n], %r1            !Cargo el contenido de n en %r1
               st          %r15, %r5
               call        sub_add
               ld          %r5, %r15
               ba          done

sub_add:       andcc       %r1, %r1, %r0       !Chequea elementos restantes
               be done                         !Si no hay mas elementos termina
               addcc       %r3, %r1, %r3       !Suma el nuevo indice a %r3
               addcc       %r1, -4, %r1        !Actualiza en indice a sumar
               ba          sub_add             !Vuelve a la funcion

done:          jmpl        %r15 + 4, %r0       !Vuelve al proceso invocante
n:             20
z:             0
              .end
```