NOTAS PRELIMINARES

Borrad las carpetas descomprimidas si usáis los ordenadores del laboratorio porque alguien puede haber modificado los ficheros.

Paso 3)

~~Instrucción 2 (add r4,r1,#64): +2 ciclos de parada~~ ← Esta no cuenta porque está fuera del bucle

Instrucción 7 (add r12,r10,r12): +2 ciclos de parada
Instrucción 9 (add r14,r12,r14): +2 ciclos de parada
Instrucción 10 (sw 0(r3),r14): +2 ciclos de parada
Instrucción 15 (beqz r5,loop): (2+3) ciclos de parada

Instrucciones totales = 10
Total ciclos de parada = 11

Instrucciones Totales y ciclos de parada:
Intenta partirlo en: "antes del bucle + bucle + después del bucle"
Instrucciones Totales = 5+(16*10)+1 = 166
Ciclos de parada totales = 2+(16*11)+0 = 178

Para comprobar la memoria ejecutad por completo y en la pestaña "Memoria" comprobad los valores de "z", deben ir desde 90 hasta 105.

Estadísticas del simulador:
- Instrucciones Totales = 166
- Ciclos de parada totales = 178
- Ciclos totales = 348

Coinciden los resultados → Sí

CPI = ciclos / instrucciones → 348/166 = 2.096 CPI

Paso 4)

El código optimizado (más abajo) da 250 ciclos de ejecución, 80 ciclos de parada y 166 instrucciones.
CPI: 250 / 166 = 1,506

```
; z = a + x + y
; Tamaño de los vectores: 16 palabras
; Vector x
   .data
x: .word 0,1,2,3,4,5,6,7,8,9
   .word 10,11,12,13,14,15

; Vector y
y: .word 100,100,100,100,100,100,100,100
   .word 100,100,100,100,100,100,100,100

; Vector z
; 16 elementos son 64 bytes.
z: .space 64

; escalar a
a: .word -10

; El código
   .text

start:
   add r1,r0,x ; Copy pointer x to r1
   add r2,r0,y ; Copy pointer y to r2
   add r3,r0,z ; Copy pointer z to r3
   add r4,r1,#64 ; 16*4 add 64 (to the pointer), that is, 64 bytes or 16 int_32
   lw r10,a(r0) ; Load a to r10

loop:
   lw r12,0(r1) ; load x[0]
   lw r14,0(r2) ; load y[0]
   add r1,r1,#4 ; increment pointer x
   add r12,r10,r12 ; add a + x[0]
   add r14,r12,r14 ; add a + x[0] + y[0]
   add r2,r2,#4 ; increment pointer y
   seq r5,r4,r1 ; set equal pointer_x with pointer_x+64 (bytes)
   sw 0(r3),r14 ; store in z[0]
   add r3,r3,#4 ; increment pointer z
   beqz r5,loop ; branch if eq 0 to loop
   trap #0   ; Fin de programa
```