Ejercicio 1.1

La ejecución de las **7 instrucciones ocupó 16 ciclos (CPI = 2,286).**
Respecto a atascos, hubo **4 tipo RAW y 2 estructurales.**

Un primer atasco RAW se da en la instrucción (3), porque espera al writeback de (2).
Dos atascos RAW se dan en la instrucción (5) por la espera de la suma FP (3).
En (5) también se da un **atasco estructural** porque (3) se encuentra en etapa MEM.

Un atasco RAW se da en la instrucción (6) por la espera de la multiplicación FP (4).
En (6) también se da un atasco estructural porque (4) se encuentra en etapa MEM.

```
.code
(1)     l.d     f1, n1(r0)
(2)     l.d     f2, n2(r0)
(3)     add.d   f3, f2, f1          ; Atasco RAW en la etapa EX
(4)     mul.d   f4, f2, f1
(5)     s.d     f3, res1(r0)        ; 2 RAW y 1 estructural, ambos en EX
(6)     s.d     f4, res2(r0)        ; 1 RAW y 1 estructural, ambos en EX
(7)     halt
```

Ejercicio 1.2 – 24 ciclos

Los atascos WAR se producen al intentar escribir en un registro pendiente de lectura, por eso **write after read**: se demora la escritura hasta que se haga la lectura.

En este caso, la instrucción (4) tiene como objetivo guardar una multiplicación en f1, y este mismo registro está pendiente de lectura en (3) durante un ciclo. Aunque no se produzca un error (la multiplicación tarda 7 ciclos) lo realiza por precaución.

```
.code
(1)     l.d     f1, n1(r0)
(2)     l.d     f2, n2(r0)
(3)     add.d   f3, f2, f1
(4)     mul.d   f1, f2, f1          ; Atasco WAR en la etapa ID
(5)     mul.d   f4, f2, f1          ; 6 RAW en la etapa EX
(6)     s.d     f3, res1(r0)        ; 1 estructural en EX
(7)     s.d     f4, res2(r0)        ; 4 RAW en ID, 5 RAW y 1 Str en EX
(8)     halt
```

Agregando un NOP entre las instrucciones (2) y (3) se elimina el atasco RAW en (3) porque se realiza en tiempo el writeback de (2) para su uso en la suma FP de (3).

Esto produce que la lectura de f1 en (3) se realice un ciclo antes, por lo que también implica la eliminación del atasco WAR en la instrucción (4). ~~¡Dos pájaros de un tiro!~~

Ejercicio 2 – Instrucciones de conversión

mtc1 r_f, f_d	Copia los 64 bits del registro entero r_f al registro f_d de punto flotante
mfc1 r_d, f_f	Copia los 64 bits del registro f_f de punto flotante al registro r_d entero
cvt.d.l f_d, f_f	Convierte a punto flotante el valor entero copiado al registro f_f, dejándolo en f_d
cvt.l.d f_d, f_f	Convierte a entero el valor en punto flotante contenido en f_f, dejándolo en f_d

Ejercicio 3 – 40 ciclos

```
.data
BASE:       .double 5.85        ; en cm
ALTURA:     .double 13.47       ; en cm
SUPERFICIE: .double 0.0         ; en cm cuadrados

.code
l.d     f0, BASE(r0)
l.d     f1, ALTURA(r0)
daddi   r2, r0, 2               ; divisor entero
mul.d   f2, f0, f1              ; base x altura
mtc1    r2, f3
cvt.d.l f4, f3                  ; convertir a punto flotante
div.d   f5, f2, f4              ; dividir por 2.0
s.d     f5, SUPERFICIE(r0)
halt
```

Ejercicio 4 – 62 ciclos en el peor caso

```
; ATENCIÓN:  F0 ES DE PROPOSITO GENERAL
; VALORES:   45 KG PARA INFRAPESO, 60 KG PARA NORMAL
;            84 KG PARA SOBREPESO, 96 KG PARA OBESO

; LA DIVISION FP PRODUCE UN GRAN ATASCO

.data
PESO:       .double 96.4        ; En Kg
ALTURA:     .double 1.73        ; En metros (cte)
TABLA:      .double 18.5, 25, 30
ESTADO:     .word 0
IMC:        .double 0.0

.code
l.d     f2, ALTURA(r0)
l.d     f1, PESO(r0)
mul.d   f4, f2, f2              ; denominador
div.d   f3, f1, f4              ; calculo de imc

; REGISTROS PARA TRABAJAR CON EL RESULTADO ANTERIOR
dadd    r1, r0, r0              ; para desplazamiento
daddi   r2, r0, 3               ; cant max comparaciones
daddi   r3, r0, 1               ; valor de estado (min 1)

loop:   l.d     f0, TABLA(r1)
        c.lt.d  f3, f0          ; menor a un IMC?
        bc1t    fin             ; salta si FP = 1
        daddi   r2, r2, -1      ; dec r2 (delay slot)
        daddi   r1, r1, 8       ; offset
        bnez    r2, loop
        daddi   r3, r3, 1       ; inc estado (delay slot)

fin:    s.d f3, IMC(r0)
        sd r3, ESTADO(r0)
        halt
```