Prueba 4

1.1 Hay 4 zonas:

1ª zona: registros de propósito general (R0, R1 ... R32)
ocupan las direcciones (0x000 – 0x01F)

2ª zona: registros de I/O, hay 64
ocupan las direcciones (0x020 – 0x05F)

3ª zona: registros de I/O extendida, hay 160
ocupan las direcciones (0x060 – 0xDFF)

4ª zona: memoria de datos (SRAM), las hay de distinto tamaño según modelo
ocupa las direcciones (0x100 – RAMEND)
en el ATmega328P (0x100 – 0x8FF)

1.2

(a) MOV R0, R4 ó LDS R0, $4

(b) IN R0, 4 ó LDS R0, $24

(c) LDS R0, $64

(d) LDS R0, $104

1.3

```
MULTIPLICA:  PUSH  R1
             PUSH  R0
             LD    R4, X
             LD    R5, Y
             MULS  R5, R4
             MOV   R5, R1
             MOV   R4, R0
             POP   R0
             POP   R1
             RET
```

1.4

Si OCR1A = $0064 ⇒ VMAX = 100

Si TCCR1B = 0000 10 10 (5A) ⇒ modo CTC

10 → frecuencia — $f = \frac{clk}{8}$

como $f = 8Mhz$ ⇒ la frecuencia del temporizador con $\frac{clk}{8}$ sería 1 Mhz

Entonces en 1s —— 1.000.000 ciclo, para

1s — 100

llegar solo a VMAX = 100 ⇒ emplearía $\frac{1s}{10.000} = 0.1ms$

Se activa el banderín cada 0.1ms

2

```
.include "m328Pdef.inc"
.cseg
.org $0
jmp start
.org $16
jmp refrigera
.org $40
```

f = 1 Mhz

1 s — 1.000.000

30 s — 30.000.000

30 s — 29296 (f/1024)

OK

```
start: rcall iniciapuertos
       rcall iniciatimer
       ldi r16, 0b1101
       sts tccr1b, r16
       sei
fin:   rjmp fin
```

} puede ir dentro de la subrutina

```
iniciapuertos: clr r16
               out ddrb, r16
               sbi ddrd, 0
               ret
```

} puede obviarse (por defecto ya está a 0)

```
iniciatimer: clr r16
             sts tcnt1h, r16
             sts tcnt1l, r16
             ldi r16, high(29296)
             sts ocr1ah, r16
             ldi r16, low(29296)
             sts ocr1al, r16
             ldi r16, 2
             sts timsk1, 2
             ret
```

} puede obviarse (por defecto ya está a 0)

```
refrigera: in   r16,sreg   ---> no es obligatorio preservar sreg pues el programa principal está en el bucle fin: rjmp fin
           in   r17, pinb
           cpi  r17, 25
           brcc enciende      T≥25
           cpi  r17, 23
           brcs apaga         T<23
vuelve:    out  sreg, r16     /* si 23≤T<25 no hago nada */
           reti
enciende:  sbi  portb,0
           rjmp vuelve
apaga:     cbi  portb,0
           rjmp vuelve
```