20 AVR

```
LDI R16,$40
OUT DDRB,R16

NOP                 ;Si la frecuencia de reloj es de 4MHZ
                    ;entonces cada NOP consume 0.25 micro-seg
NOP

LDI R17,$38
MOV R13,R17
```

bién configuraremos los bits `ISC01` e `ISC00`:

Bit	7	6	5	4	3	2	1	0	
	PUD	SM1	SE	SM0	ISC11	ISC10	ISC01	ISC00	MCUCR
	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Valor inicial	0	0	0	0	0	0	0	0	

```
LDI R16,(1<<SE)|(1<<SM1)|(1<<SM0)|(0<<ISC01)|(0<<ISC00) ;HABILITAMOS
OUT MCUCR, R16;                  ;EL SLEEP EN MODO POWER-DOWN
                                 ;Y EL INT0 EN "LEVEL INTERRUPT"
```

Programa:

```
;ESTE PROGRAMA DORMIRÁ AL AVR, CON LA INTERRUPCIÓN-0
;SE DESPERTARÁ AL AVR PARA PRENDER UN LED

Encabezado para ATtiny2313

RJMP CUANDO_RESETEO           ;DIR $000
RJMP INTERRUPCION_INT0        ;DIR $001

CUANDO_RESETEO:
LDI R16,LOW(RAMEND)
OUT SPL,R16

LDI R16,$01
OUT DDRB,R16                  ;PB0 como salida del LED

;DAREMOS DE ALTA LA INTERRUPCIÓN-0 CON "LEVEL INTERRUPT" (VER
;MANUAL) PORQUE ES LA CONFIGURACIÓN QUE RESPETA LA "INT0"
;PARA EL SLEEP EN POWER-DOWN
```

```
LDI R16,(0<<SE)|(1<<SM1)|(1<<SM0)|(0<<ISC01)|(0<<ISC00)
OUT MCUCR,R16

LDI R16, 1<<INT0                ;Habilitamos INT0
OUT GIMSK,R16
SEI

IN R16,MCUCR
ORI R16,(1<<SE)
OUT MCUCR, R16

RCALL CUATRO_SEGUNDOS

SLEEP                           ;DORMIMOS AL AVR

ESPERANDO: RJMP ESPERANDO
;DESHABILITAREMOS EL SLEEP PARA QUE SÓLO SEA USADO UNA VEZ
INTERRUPCION_INT0:

;"SE" deshabilitado
LDI R16,(0<<SE)|(1<<SM1)|(1<<SM0)|(0<<ISC01)|(0<<ISC00)
OUT MCUCR, R16

LDI R16,$01
OUT PORTB,R16                   ;PRENDE LED
RETI
```

Esta sintaxis ayuda a activar un solo bit sin tener que escribir todos los demás

En este ejercicio desactivamos SE en esta línea, pero es factible activar nuevamente SE en otra parte del programa

Bit	7	6	5	4	3	2	1	0	
	-	-	-	WDCE	WDE	WDP2	WDP1	WDP0	**WDTCR**
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Valor inicial	0	0	0	0	X	0	0	0	

```
;SUBRUTINA WATCHDOG RESET
;SIRVE PARA RESETEAR EL AVR POR SOFTWARE POR SI ALGÚN
;PROCESO NO SE CUMPLIÓ

.INCLUDE "M8515DEF.INC"
.CSEG
.ORG 0
RJMP RESET          ;DIR 000
.ORG 6
RJMP OVER1          ;DIR 006

RESET:
LDI R16,LOW(RAMEND)
OUT SPL,R16
LDI R16,HIGH(RAMEND)
OUT SPH,R16

RCALL DOG_OFF       ;Deshabilita DOG

LDI R16,$01
OUT DDRA,R16        ;Testigo

LDI R16,$00
OUT PORTA,R16

OUT TCNT1L,R16      ;RESET al CONTADOR-1
OUT TCNT1H,R16

LDI R16,$80         ;Activa interrupción TIMER1 OVERFLOW
OUT TIMSK,R16

LDI R16,$80         ;Borra OVERFLOW1
OUT TIFR,R16

LDI R16,$04         ;PRE-ESCALAMIENTO CK/256 para TIMER1
OUT TCCR1B,R16      ;para que sea lento
SEI

CICLO:
RJMP CICLO

OVER1:
LDI R16,$01         ;Indica que ya entró a OVERFLOW
OUT PORTA,R16
```

```
RCALL DOG_ON        ;Prende el WATCHDOG y esperamos a que
                    ;se apague el LED
RETI

DOG_ON:             ;Subrutina para habilitar al WATCHDOG
WDR                 ;Resetea el DOG TIMER para evitar que
                    ;El DOG no empiece en cero

LDI R16,1<<WDE|1<<WDP2|1<<WDP1|1<<WDP0
OUT WDTCR,R16       ;El PRE-ESCALAMIENTO es el retardo para
                    ;que se dé el RESET

LDI R16,$00         ;Detenemos el TIMER1
OUT TCCR1B,R16
RET

DOG_OFF:            ;Subrutina para deshabilitar WATCHDOG
LDI R16,$18         ;pongo 1 en WDCE y WDE al mismo tiempo
OUT WDTCR,R16       ;dentro de los 4 ciclos de reloj (según
                    ;manual)

LDI R16,$10         ;Pongo a cero el WDE
OUT WDTCR,R16       ;ya se deshabilitó el WATCHDOG
NOP
RET
```

Ejemplo:

```
.INCLUDE "TN2313DEF.INC"
.CSEG
.ORG 0

LDI R16,LOW(RAMEND)
OUT SPL,R16
```

```
LDI R16,1
OUT DDRB,R16        ;SALIDA PARA LED-TEST

LDI R16,10
LDI R17,10
LDI R18,10

BREAK
LDI R20,1
OUT PORTB,R16       ;Prendemos EL LED para
                    ;visualizar el comportamiento de BREAK

FIN: RJMP FIN
```

En la simulación va a encontrar el error "`AVR Simulator: Invalid opcode 0x9598`", cuando ejecute `BREAK`, que corresponde al hexadecimal de la instrucción en el desensamblador. Verifique la ventana Disassembler