UNIVERSIDAD NACIONAL

AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES

ARAGÓN

INGENIERÍA EN COMPUTACIÓN

LAB.MICROPROCESADORES Y

MICROCONTROLADORES

PROF: ROBERTO AGUSTIN GARCIA

CASTREJON

PREVIO 4

ALUMNA: CELIS ALONSO NELI

XIMENA

Previo

1. Defina qué es un retardo

Un retardo en microprocesadores y microcontroladores es una pausa intencionada en la ejecución de un programa durante un período específico de tiempo, que puede ser generado mediante software o hardware . Su propósito principal es sincronizar tareas, permitir la espera de eventos o introducir un tiempo de espera necesario para el correcto funcionamiento de un sistema electrónico.
Los retardos de Software se implementan creando un bucle de instrucciones que se ejecuta un número específico de veces para consumir una cantidad determinada de tiempo.
Los retardos de Hardware se utilizan dispositivos físicos llamados temporizadores o contadores, que son circuitos integrados dentro del microcontrolador. El programador configura el temporizador para contar un número de pulsos del reloj y, una vez que se completa la cuenta, genera una interrupción que detiene o pausa temporalmente la ejecución del programa.

2. Investigue y detalle las instrucciones: CPSE, CPI, SBRS y SBRC.

CPSE:*Compare, Skip if Equal*

Sintaxis: CPSE Rd, Rr
Función: Compara Rd con Rr y **salta la siguiente instrucción** si Rd == Rr.
Flags: no modifica banderas (Z, C, etc.).
Ciclos:

- Condición falsa (no salta): 1 ciclo.
- Condición verdadera (salta): salta la siguiente instrucción; costo total habitual 2 ciclos (si la instrucción saltada es de 16 bits como LDS/STS, el salto consume 3).

Uso típico: comparar con **R1 = 0** (en AVR suele mantenerse en cero) para saber si un contador llegó a 0 sin tocar flags.

CPI: *Compare with Immediate*

Sintaxis: CPI Rd, K (16 ≤ Rd ≤ 31)
Función: Compara Rd con una constante K. **No cambia** Rd.
Flags afectadas: Z, N, V, S, H, C (útil para saltos condicionales posteriores).
Ciclos: 1.

Uso típico: CPI R18, 0 para saber si un contador está en cero.

SBRS:*Skip if Bit in Register is Set*

Sintaxis: SBRS Rr, b (0 ≤ b ≤ 7)
Función: **Salta la siguiente instrucción** si el **bit b = 1** en Rr.
Flags: no modifica.
Ciclos: 1 si no salta; si salta, normalmente 2 (3 si la instrucción saltada es de 16 bits).

Uso típico: pruebas rápidas de bits de registros o I/O leídos a un registro.

SBRC: *Skip if Bit in Register is Clear*

Sintaxis: SBRC Rr, b
Función: **Salta la siguiente instrucción** si el **bit b = 0** en Rr.
Flags: no modifica.
Ciclos: igual que SBRS.

Uso típico: saltar si un bit está en 0 (por ejemplo, esperar a que un pin suba/baje).

3. Con las instrucciones anteriores realice un retardo.

Retardo simple con CPSE

```
; Retardo corto ~ (255 * cuerpo) ciclos
; Usa: R18 como contador, R1=0
; Frec. ejemplo: 16 MHz

      .equ    N = 255

delay_cpse:
      LDI     R18, N          ; contador
      CLR     R1              ; asegura R1=0 (por si acaso)

loop_cpse:
      CPSE    R18, R1         ; ¿R18 == 0? si sí, salta el RJMP siguiente
      RJMP    do_body         ; si R18 != 0, ejecutar cuerpo
      RJMP    end_delay       ; si R18 == 0, se saltó "do_body" y cae aquí

do_body:
      NOP                     ; cuerpo: 1 ciclo (ajusta agregando NOPs)
      DEC     R18
      RJMP    loop_cpse

end_delay:
      RET
```

Retardo con CPI y SBRS/SBRC

```
; Retardo doble bucle con SBRS: salta si el bit 0 del contador interno está en 1
; (Solo como demostración del uso de SBRS dentro del retardo)

        .equ    OUTER = 200
        .equ    INNER = 250

delay_sbrs:
        LDI     R20, OUTER          ; bucle externo
outer_loop:
        LDI     R21, INNER          ; bucle interno
inner_loop:
        SBRS    R21, 0              ; si bit0 de R21 está en 1, salta un NOP
        NOP                         ; consumo condicional (1 o 0 NOP)
        DEC     R21
        CPSE    R21, R1             ; ¿R21 == 0?
        RJMP    inner_loop

        DEC     R20
        CPSE    R20, R1             ; ¿R20 == 0?
        RJMP    outer_loop
        RET
```

4. Realice un diagrama de flujo para un parpadeo en PD0.