



Microcontroladores

Retardos mediante Software

Retardo por Software

Retardo por Software puede ser creado por ciclos como:

```
for(i=0; i<100; i++);
```

o bien:

```
i=100;  
while(--i);
```

En ambos casos el retardo dependería de:

- 1) la veces que se realiza el ciclo (ya sea el for ó while).
- 2) del código que genera el compilador.
- 3) de la velocidad del procesador.

Retardo por Software

Como pueden ver, ustedes tienen control sobre el punto **1)** dado que se define un valor de 100 en los ejemplos a) y b), pero para los otros puntos NO. En el caso de **2)** depende de las opciones de optimización del compilador y en **3)** de la velocidad de la computadora.

Nota: mediante prueba y error se podría llegar a un retardo en tiempo específico pero al llevar el código a otra computadora con otras características el retardo ya no se garantiza.

Una manera de tener control de **2)** es diseñar el retardo en ensamblador, esto convirtiendo el código de alto nivel a ensamblador manualmente. Luego el código se introduce en lenguaje C de la forma **inline** de manera que el compilador no lo traduce y la pasa tal como fue escrito.

Retardo por Software

```
:  
i=100;  
while(--i);  
:
```

Algoritmo:

inicio

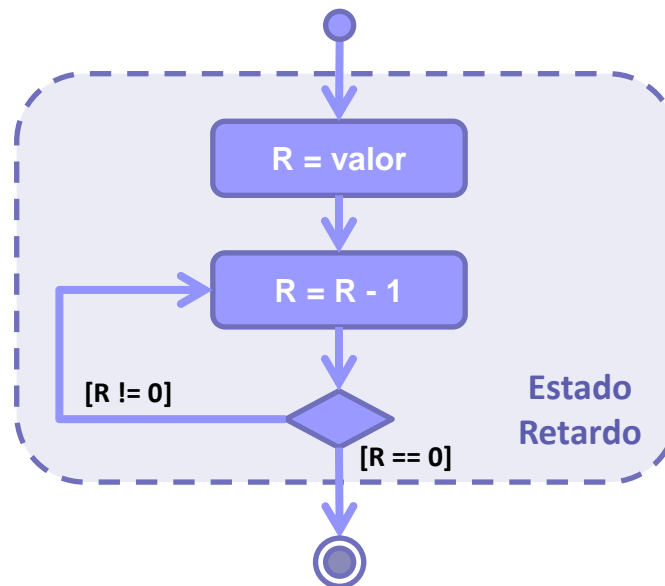
paso 1: $\text{Reg} \leftarrow \text{valor}$

paso 2: $\text{Reg} \leftarrow \text{Reg} - 1$

paso 3: Si Reg no es cero ir al paso 2 de lo contrario terminar

fin

Diagrama de Actividad:



```
      :  
      ldi R24,0x05      ; R24=5  
nxt:  nop              ; no operación  
      dec R24           ; R24=R24-1  
      brne nxt         ; salta a nxt si no es cero  
      :                ; fue cero continua aquí
```

Si queremos saber cuanto tiempo consume la ejecución de esta sección de código, es necesario hacer un análisis del código y ver **cuantos ciclos de reloj** son necesarios para cada instrucción y contabilizar las **veces que se ejecuta** cada instrucción.

análisis:

[**x**]: veces se ejecuta la instrucción

(**y**): número de ciclos de la instrucción

```
      :  
      ldi R24, 0x05      ; [1] * (1)  
nxt:  nop                ; [5] * (1)  
      dec R24            ; [5] * (1)  
      brne nxt           ; [4] * (2) y [1] * (1)  
      :                  ;
```

Ahora, para conocer el **tiempo total** es necesario calcular el total de ciclos de reloj del código, por tanto tenemos:

En total tenemos $1+5+5+(4*2)+1 = \mathbf{20 \text{ ciclos}}$ y ahora podemos determinar el tiempo total si conocemos la frecuencia con que opera el procesador. Por ejemplo si se opera a **8MHz** tenemos que un ciclo de reloj tiene un período de $1/8\text{Mhz} = \mathbf{125\text{nS}}$, por tanto el tiempo total de la secuencia es $125\text{uS} \times 20 = \mathbf{2.5 \text{ uS}}$.