

Principios de Mecatrónica – SDI-11561

Ingeniería en Mecatrónica

Hugo Rodríguez Cortés

Departamento de Ingeniería Eléctrica y Electrónica

Instituto Tecnológico Autónomo de México

Agosto 2023

- MUL Rd, Rr Multiplicación de números sin signo, el resultado se almacena en R0 byte BAJO y R1 byte ALTO.
- MULS Rd, Rr Multiplicación de números con signo, el resultado se almacena en R0 byte BAJO y R1 byte ALTO.
- MULSU Rd, Rr Multiplicación de números con signo y números sin signo, el resultado se almacena en R0 byte BAJO y R1 byte ALTO.
- Ejemplo

```
LDI    R23, 0x25 ; —
LDI    R24, 0x65 ; —
MUL    R23, R25  ; 0x25 × 0x65 = 0xE99—
                ; R1 = 0x0E, R0 = 0x99—
```

- En 0x310 se encuentra el dato 0xFD, convertirlo a decimal guardando en 0x322, 0x323 y 0x324 los dígitos; en 0x322 el menos significativo.

```

.EQU    HEX_NUM=0x315      ;           —
.EQU    RMND_L=0x322      ;           —
.EQU    RMND_M=0x323      ;           —
.EQU    RMND_H=0x324      ;           —
.DEF    NUM=R20            ;           —
.DEF    DENOMINATOR = R21  ;           —
.DEF    QUOTIENT = R22     ;           —

        LDI    R16, 0xFD   ;           —
        STS    HEX_NUM, R16 ;           —
        LDS    NUM, HEX_NUM ;           —
        LDI    DENOMINATOR, 10 ;       —
        LDI    QUOTIENT, 0 ;           —

L1:     INC    QUOTIENT     ;           —
        SUB    NUM, DENOMINATOR ;       —
        BRCC   L1          ;   salta si C=0—
        DEC    QUOTIENT     ;           —
        ADD    NUM, DENOMINATOR ;       —
        STS    RMND_L, NUM  ;           —
        MOV    NUM, QUOTIENT ;           —
        LDI    QUOTIENT, 0 ;           —

L2:     INC    QUOTIENT     ;           —
        SUB    NUM, DENOMINATOR ;       —
        BRCC   L2          ;           —
        DEC    QUOTIENT     ;           —
        ADD    NUM, DENOMINATOR ;       —
        STS    RMND_M, NUM  ;           —

        STS    RMND_H, QUOTIENT ;       —
    
```

- Para representar el signo de un número con signo se utiliza el bit más significativo

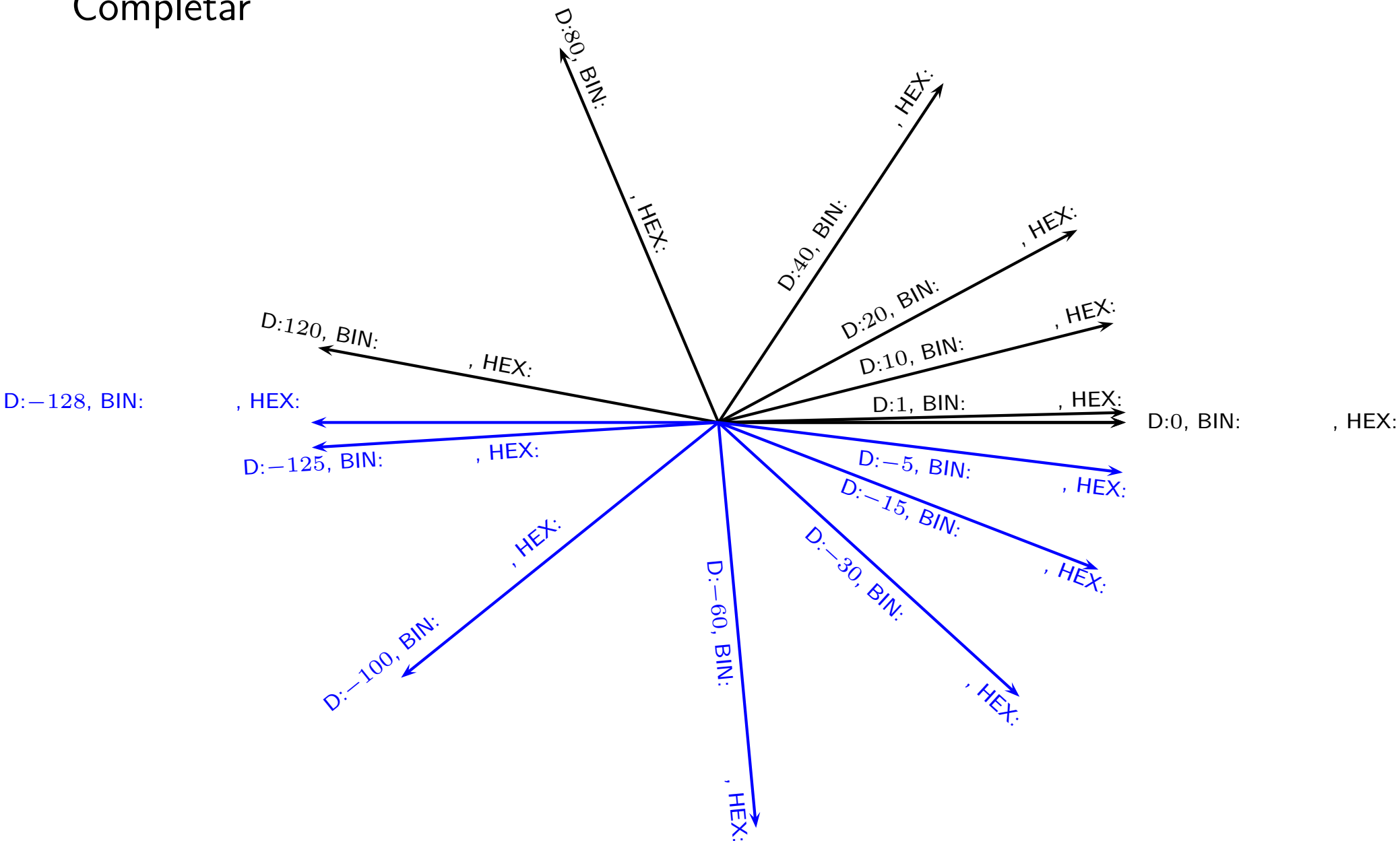
D7	D6	D5	D4	D3	D2	D2	D0
±	-	v	a	l	o	r	-

El rango de números positivos es de 0 a 127, si se requieren valores más grandes se deben utilizar operandos de 16 bits.

- En un número negativo $D7 = 1$ y el valor se representa con el complemento a 2.
- Ejemplo. Representar al número -5

0	0	0	0	0	1	0	1	5 en binario
1	1	1	1	1	0	1	0	complemento a 1
1	1	1	1	1	0	1	1	complemento a 2

Completar



- En operaciones con número con signo puede ocurrir un desbordamiento, el CPU indicará esta situación con la bandera V.
- Ejemplo.

```
LDI    R20, 0x60 ;      —  
LDI    R21, 0x46 ;      —  
ADD    R20, R21  ;      —
```

- En operaciones con número con signo puede ocurrir un desbordamiento, el CPU indicará esta situación con la bandera V.
- Ejemplo.

```
LDI    R20, 0x60 ;      —
LDI    R21, 0x46 ;      —
ADD    R20, R21   ;      —
```

$$0x60 + 0x46 = 0110\ 0000 + 0100\ 0110 = 1010\ 0110$$

El CPU indicará la situación con $V=1$. Notar que $N=1$ (resultado negativo).

- El rango de operación para números con signo son los números enteros $[-128, 127]$.

- En operaciones de 8 bits de números con signo la bandera V se asigna igual a 1 si alguna de las condiciones siguientes ocurre
 - ◆ Se lleva uno de D6 a D7 y no se lleva de D7 (C=0).
 - ◆ Se lleva uno de D7 (C=1) y no se lleva de D6 a D7
- Determine el valor de las banderas N y V en el siguiente código

LDI	R20, 0x80	;	—
LDI	R21, 0xFE	;	—
ADD	R20, R21	;	—

- En operaciones de 8 bits de números con signo la bandera V se asigna igual a 1 si alguna de las condiciones siguientes ocurre
 - ◆ Se lleva uno de D6 a D7 y no se lleva de D7 (C=0).
 - ◆ Se lleva uno de D7 (C=1) y no se lleva de D6 a D7
- Determine el valor de las banderas N y V en el siguiente código

```
LDI    R20, 0x80    ;      —
LDI    R21, 0xFE    ;      —
ADD    R20, R21     ;      —
```

$N = 0, V = 1$

Examine el valor de las banderas V y N

- Sumar -2 con -5.

LDI	R20, -2	;	—
LDI	R21, -5	;	—
ADD	R20, R21	;	—

Examine el valor de las banderas V y N

■ Sumar -2 con -5.

```
LDI    R20, -2    ;      —
LDI    R21, -5    ;      —
ADD    R20, R21   ;      —
```

1	1	1	1	1	1	1	0	-2	
1	1	1	1	1	0	1	1	-5	, V = 0, N = 1
1	1	1	1	1	0	0	1	-7	

- Sumar 7 con 18.

LDI	R20, 7	;	—
LDI	R21, 18	;	—
ADD	R20, R21	;	—

- Sumar 7 con 18.

```
LDI    R20, 7      ; —
LDI    R21, 18     ; —
ADD    R20, R21    ; —
```

```
0 0 0 0 0 1 1 1 7
0 0 0 1 0 0 1 0 18 , V = 0, N = 0
0 0 0 1 1 0 0 1 25
```

- En operaciones sin signo C determina el estado de la operación, mientras que en operaciones con signo lo hace V.

- AND Rd, Rr Aplica el operador lógico AND bit a bit entre los registros Rd y Rr. ANDI Rd, k con k un valor constante.

X	Y	X AND Y
0	0	0
0	1	0
1	0	0
1	1	1

Esta operación puede activar las banderas Z, S y N.

LDI	R20, 0x35	;	—
ANDI	R20, 0x0F	;	—

- AND Rd, Rr Aplica el operador lógico AND bit a bit entre los registros Rd y Rr. ANDI Rd, k con k un valor constante.

X	Y	X AND Y
0	0	0
0	1	0
1	0	0
1	1	1

Esta operación puede activar las banderas Z, S y N.

```
LDI    R20, 0x35 ; —
ANDI   R20, 0x0F ; —
```

```
0 0 1 1 0 1 0 1 0x35
0 0 0 0 1 1 1 1 0x0F , Z = 0, N = 0
0 0 0 0 0 1 0 1 0x05
```

- OR Rd, Rr Aplica el operador lógico OR bit a bit entre los registros Rd y Rr. ORI Rd, k con k un valor constante.

X	Y	X OR Y
0	0	0
0	1	1
1	0	1
1	1	1

Esta operación puede activar las banderas Z, S y N.

```
LDI  R20, 0x04 ; —
ORI  R20, 0x30 ; —
```


- OR Rd, Rr Aplica el operador lógico OR bit a bit entre los registros Rd y Rr. ORI Rd, k con k un valor constante.

X	Y	X OR Y
0	0	0
0	1	1
1	0	1
1	1	1

Esta operación puede activar las banderas Z, S y N.

```
LDI  R20, 0x04 ; —
ORI  R20, 0x30 ; —
```

```
0 0 0 0 0 1 0 0 0x04
0 0 1 1 0 0 0 0 0x30 , Z = 0, N = 0
0 0 1 1 0 1 0 0 0x34
```

- EXOR Rd, Rr Aplica el operador lógico XOR bit a bit entre los registros Rd y Rr.

X	Y	X XOR Y
0	0	0
0	1	1
1	0	1
1	1	0

Esta operación puede activar las banderas Z, S y N.

LDI	R20, 0x54	;	—
LDI	R20, 0x78	;	—
EOR	R20, R21	;	—

- EXOR Rd, Rr Aplica el operador lógico XOR bit a bit entre los registros Rd y Rr.

X	Y	X XOR Y
0	0	0
0	1	1
1	0	1
1	1	0

Esta operación puede activar las banderas Z, S y N.

```
LDI    R20, 0x54 ;      —
LDI    R20, 0x78 ;      —
EOR    R20, R21  ;      —
```

```
0 1 0 1 0 1 0 0 0x54
0 1 1 1 1 0 0 0 0x78 , Z = 0, N = 0
0 0 1 0 1 1 0 0 0x2C
```

- Analice el siguiente código

```
LDI    R20, 0xFF    ;    —
OUT    DDRC, R20    ;    —
LDI    R20, 0x00    ;    —
OUT    DDRB, R20    ;    —
OUT    PORTC, R20   ;    —
LDI    R21, 0x45    ;    —
HERE:
IN      R20, PINB    ;    —
EOR     R20, R21     ;    —
BRNE    HERE        ;    —
LDI     R20, 0x99    ;    —
OUT     PORTC, R20   ;    —
```

- COM Rd Esta instrucción complementa bit a bit el contenido de un registro. Se conoce también como el complemento a uno.
- NEG Rd Esta instrucción calcula el complemento a dos de contenido del registro.
- CP Rd, Rr Esta instrucción es una resta entre Rd y Rr excepto que el valor de Rd no cambia.
- CPI Rd, k comparación con una constante. CP y CPI se unen a operadores de saltos condicionales.

- BCD (binary coded decimal) es la representación binaria de los números del 0 al 9. Se asocia a dos conceptos BCD sin comprimir y BCD comprimido.
- En un BCD sin comprimir el nibble bajo representa al BCD y el nibble alto se llena con ceros. Por ejemplo, $9 = 00001001$, $5 = 00000101$. Se utiliza un byte de memoria para almacenarlo.
- En un BCD comprimido. Los nibbles alto y bajo contienen un número. Por ejemplo, $0x59 = 01011001$. Se requiere la misma cantidad de memoria para almacenarlo, un byte
- En teclados ASCII cuando se activa la tecla '0' se envía $0110000 = 0x30$ al CPU. De forma similar '2' $= 0110010 = 0x32$.

- Para convertir de BCD comprimido a ASCII. Primero el BCD se descomprime y luego se combina con 0x30.

BCD comprimido	BCD sin comprimir		ASCII	
0x29	0x02	0x09	0x32	0x39
0010 1001	0000 0010	0000 1001	0011 0010	0011 1001

- Código

```

LDI    R20, 0x29 ;
MOV    R21, R20  ;
ANDI   R21, 0x0F ;
ORI    R21, 0x30 ;

MOV    R22, R20  ;
SWAP   R22      ; intercambia nibbles—
ANDI   R22, 0x0F ;
ORI    R22, 0x30 ;
    
```

- Para convertir de ASCII a BCD comprimido. Primero el BCD se descomprime y luego se combina para comprimirlo.

Tecla	ASCII	BCD sin comprimir	BCD comprimido
'4'	0x34	00000100	
'7'	0x37	00000111	01000111=0x47

- Código

```

LDI    R21, '4'    ;    —
LDI    R22, '7'    ;    —
ANDI   R21, 0x0F   ;    —
SWAP   R21         ;    —

ANDI   R22, 0x0F   ;    —
OR     R22, R21    ;    —
MOV    R20, R22    ;    —
    
```


- El CPU acepta operaciones aritméticas del tipo

```
.EQU  ALFA = 50    ;      —  
.EQU  BETA = 40    ;      —  
  
LDI   R23, ALFA    ;      —  
STS   R24, ((ALFA-BETA)*2)+9 ;  —
```

- Lo mismo que operaciones lógicas como

```
.EQU  C1 = 0x50    ;      —  
.EQU  C2 = 0x10    ;      —  
.EQU  C3 = 0x04    ;      —  
  
LDI   R21, (C1&C2)|C3 ;      —
```

■ Operaciones aritméticas

Símbolo	Acción
+	Adición
—	Resta
*	Multiplicación
/	División
%	Módulo

■ Operaciones lógicas

Símbolo	Acción
&	AND bit a bit
	OR bit a bit
^	XOR bit a bit
~	NOT bit a bite

■ Operadores de corrimiento

Símbolo	Acción	Ejemplo
$X \ll Y$	Recorre hacia la izquierda a X Y número de lugares y viceversa	LDI R20, 0b101<<2 ;R20 = 0b10100
$X \gg Y$	Recorre hacia la derecha a X Y número de lugares y viceversa	LDI R20, 0b100>>1 ;R20 = 0b010

■ Ejemplo

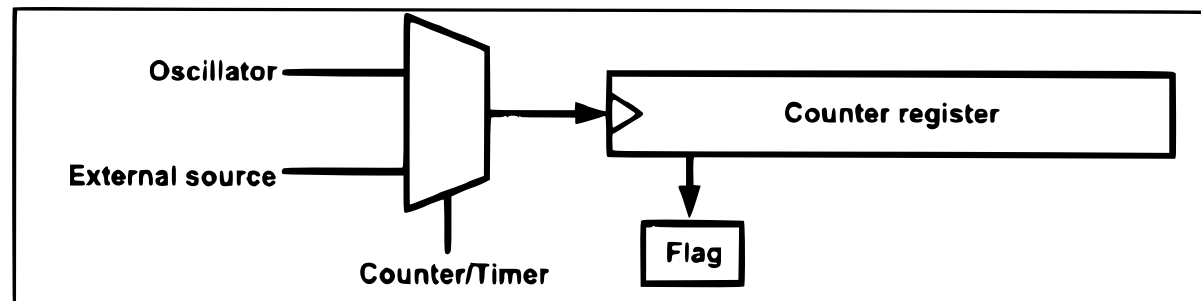
```
LDI R21, 0b00000111<<1 ; R21 = 0b00001110—
```

■ HIGH() y LOW() Obtienen el byte alto y bajo, respectivamente de un dato de 16 bits

```
LDI R21, LOW(0x4455) ; R21 = 0x55—
LDI R22, HIGH(0x4455) ; R22 = 0x44—
```

- Muchas aplicaciones necesitan contar eventos y/o generar tiempos de retardo. El microcontrolador tiene **registros de contadores** para estos propósitos.
- Para contar un evento se conecta la fuente del evento externo al pin del registro del contador. De esta forma, cuando el evento ocurre el contador se incrementa, así, el contenido del contador representa el número de veces que el evento ocurre.
- Para generar tiempo de retardo, se conecta el oscilador al pin del contador. Cada oscilación incrementa el contenido del contador. Como resultado, el contenido del registro del contador representará cuantas oscilaciones han ocurrido desde que limpió el contador.

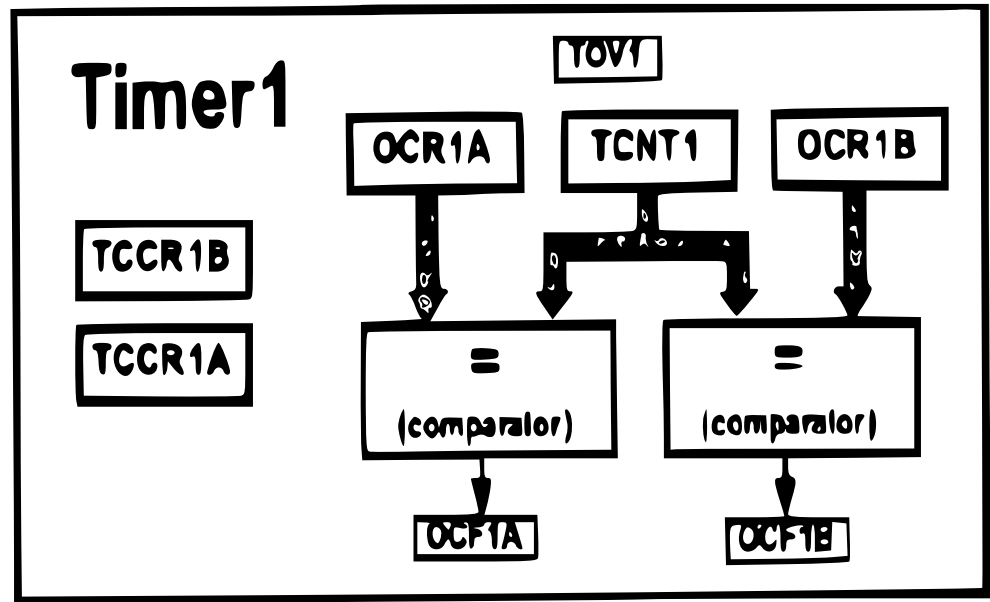
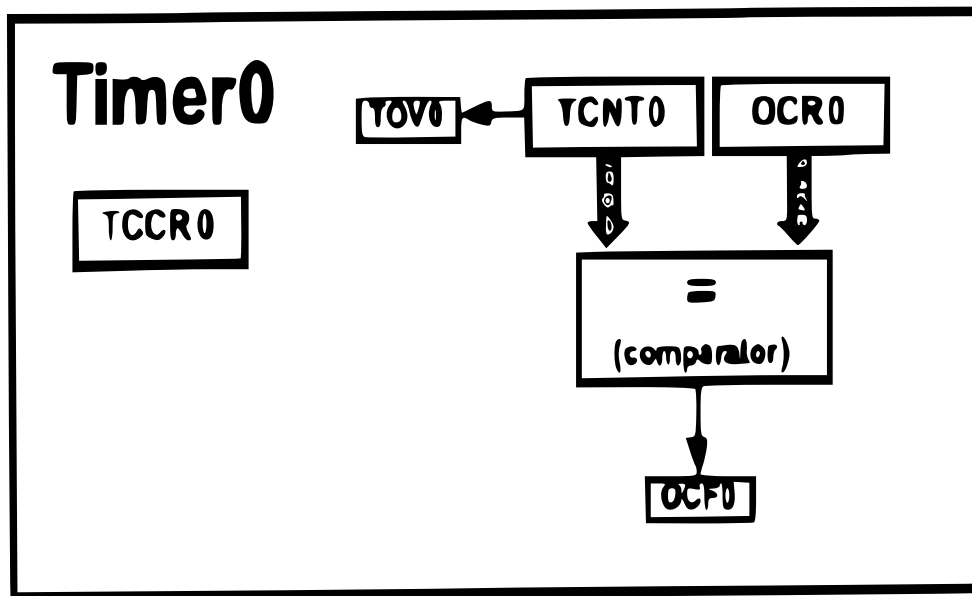
- Ya que la velocidad del oscilador, en un microcontrolador, se conoce, se puede calcular el periodo de la oscilación y del contenido del registro del contador se obtiene el tiempo transcurrido.
- De esta manera para generar un retardo, se borra el contador en el tiempo inicial y se espera hasta que el contador alcance un cierto número.



- Ejemplo, considere un microcontrolador con un oscilador de 1MHz; en el microcontrolador, el contenido del registro del contador incrementa una vez por cada microsegundo. Entonces, si se quiere un retardo de 100 microsegundos, debemos limpiar el contador y esperar hasta que sea igual a 100.
- En el microcontrolador, hay una bandera para cada uno de los contadores.
 - ◆ La bandera se activa cuando el contador se desborda, por software se borra.
 - ◆ El segundo método para generar un retardo es cargar el registro del contador y esperar hasta que el contador se desborde y la bandera se active.

- Ejemplo. En un microcontrolador, con oscilador de 1MHz y registro de contador de 8 bits. Si se requiere un retardo de 3 microsegundos, el registro del contador se carga con 0xFD, la bandera se activará después de tres ciclos. El contenido del registro se incrementará a 0xFE, después a 0xFF y en el tercer ciclo el contenido del registro se desborda a 0x00 y la bandera se activa.
- El AVR tiene de 1 a 6 timers dependiendo del tipo de familia. Ellos son referidos como 0, 1..5. Pueden ser usados como timers para generar retardo o como contadores para contar eventos sucediendo fuera del microcontrolador.
- El ATmega32, tiene 3 timers: Timer0, Timer1 y Timer2. Timer0 y Timer2 son de 8 bits, mientras que el timer T1 es de 16 bits.

- Cada Timer necesita un pulso de reloj para contar. La fuente del pulso puede ser interna o externa.
- Si se usa una fuente interna, entonces la frecuencia del oscilador alimenta al contador. Por lo tanto, al utilizarse para generar retardos se conoce como Timer.
- Si se elige la opción de un pulso externo, es necesario alimentarlo a través de uno de los pines del AVR. En este caso se tiene un contador.



TCNTn (timer/counter) registro del Timer/contador. TOVn (Timer Overflow) bandera de desbordamiento. TCCRn (timer/counter control register) registro para definir modos de operación. OCRn (Output Compare Register). El contenido de OCRn se compara con TCNTn, si son iguales la bandera OCFn (Output Compare Flag) se activa.

- TCNT0 El registro Timer/Contador tiene 8 bits

TCNT0 D7 D6 D5 D4 D3 D2 D1 D0

- TCCR0 Control/Modo de Operación del registro TCNT0

TCCR0	D7	D6	D5	D4	D3	D2	D1	D0
	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00
	W	RW	RW	RW	RW	RW	RW	RW
CI	0	0	0	0	0	0	0	0

FOC0: Se usa para generar ondas. WGM00-WGM01: Modo de Operación 0-0 normal, 0-1 Limpiar el Timer al comparar, 1-0 PWM fase correcta, 1-1 PWM rápido. COM00-COM01: Se utilizan en la generación de ondas. CS02-CS01-CS00: Configuración del reloj, 0-0-0 Timer/Contador detenido, 0-0-1 Reloj clk sin escalador, 0-1-0 clk/8, 0-1-1 clk/64, 1-0-0 clk/256, 1-0-1 clk/1024. 1-1-0 reloj en pin T0 flanco de bajada, 1-1-1 reloj en pin T0 flanco de subida.

- TIFR0 Registro de la bandera de interrupción del registro del Timer/Contador.

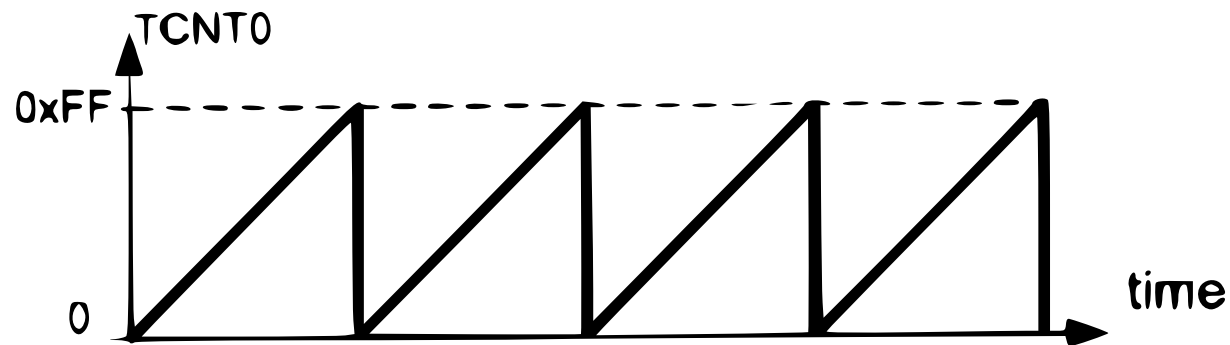
TIFR0	D7	D6	D5	D4	D3	D2	D1	D0
	OFC2	TOV2	ICF1	OFC1A	OFC1B	TOV1	OFC0	TOV0
	RW	RW	RW	RW	RW	RW	RW	RW
CI	0	0	0	0	0	0	0	0

TOV0: 0-1 no desbordamiento/desbordamiento Timer0, 0xFF→0x00. OFC0: 0-1 comparación ocurrió/no ocurrió. TOV1: 0-1 no desbordamiento/desbordamiento Timer1. OFC1B: Bandera de comparación de salida B del Timer1. OFC1A: Bandera de comparación de salida A del Timer1. ICF1: Bandera de captura de entrada. TOV2: Bandera de desbordamiento Timer2. OFC2: Bandera de comparación de salida.

- La bandera TOV0 es igual a uno cuando TCNT0 se desborda, decir va de 0xFF a 0x00. Para borrar la bandera se escribe uno en TOV.

```
LDI    R21, 0x01    ;
OUT    TIFR, R21    ;
```

- En este modo, el contenido del timer/contador aumenta cada ciclo de reloj. El contador se incrementa hasta alcanzar su máximo 0xFF, cuando él pasa de 0xFF a 0x00, la bandera TVO0 se activa. Esta bandera se puede leerse.



	LDI	R16, 1<<5	;	—
	SBI	DDRB, 5	;	—
	LDI	R17, 0x00	;	—
	OUT	PORTB, R17	;	—
BEGIN:	RCALL	DELAY	;	—
	EOR	R16, R17	;	—
	OUT	PORTB, R17	;	—
	RJMP	BEGIN	;	—
DELAY:	LDI	R20, 0xF2	;	—
	OUT	TCNT0, R20	;	—
	LDI	R20, 0x01	;	—
	OUT	TCCR0, R20	;	—
AGAIN:	IN	R20, TIFR	;	—
	SBRS	R20, TOV0	;	SBRS Verifica el estatus de un bit, salta si TVO0=1—
	RJMP	AGAIN	;	—
	LDI	R20, 0x00	;	—
	OUT	TCCR0, R20	;	—
	LDI	R20, (1<<TOV0)	;	—
	OUT	TIFR, R20	;	—
	RET		;	—

- ¿Cuál es el valor del retardo generado por el timer asumiendo que el cristal(XTAL) es de 8 MHZ?

- ¿Cuál es el valor del retardo generado por el timer asumiendo que el cristal(XTAL) es de 8 MHZ?
- ◆ La frecuencia del reloj que alimenta al Timer es de 8MHz, entonces cada ciclo de reloj tiene un periodo de $T = 1/(8MHz) = 0.125$ micro segundos. Esto significa que el Timer0 cuenta hacia arriba cada 0.125 microsegundos, esto es,
$$\text{Delay} = \text{número de cuentas} \times (0.125 \text{ microsegundos}).$$
- ◆ El contador inicia en 0xF2 y la bandera TOV0 se activa al llegar a 0x00, entonces, sumaríamos un ciclo extra ($0xFF - 0xF2 = 13 + 1$). Así, tendremos que el retardo es de 1.75 microsegundos.

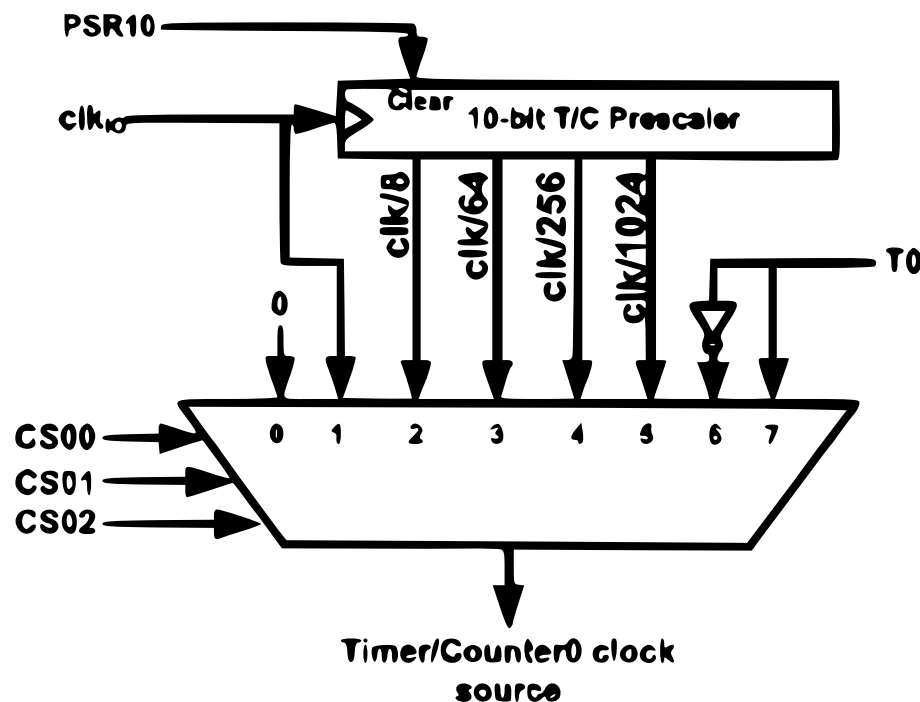
Ejemplo. Suponga que se tiene un cristal de 8MHz. Modificar el programa anterior para generar una onda cuadrada con periodo de $12.5 \mu s$ en el pin 3 del PORTB.

Ejemplo. Suponga que se tiene un cristal de 8MHz. Modificar el programa anterior para generar una onda cuadrada con periodo de $12.5 \mu s$ en el pin 3 del PORTB.

Para generar la onda cuadrada con $T = 12.5 \mu s$ se requiere un retardo de $6.25 \mu s$. Con el cristal de 8MHz se tiene un contador de $0.125 \mu s$. Por lo tanto, se requieren $6.25/0.125$ ciclos. Las modificaciones al programa anterior son las siguientes

	LDI	R16, 0x08	;	—
	SBI	DDRB, 3	;	—
	LDI	R17, 0x00	;	—
	OUT	PORTB, R17	;	—
DELAY:	LDI	R20, 0xCE	;	—

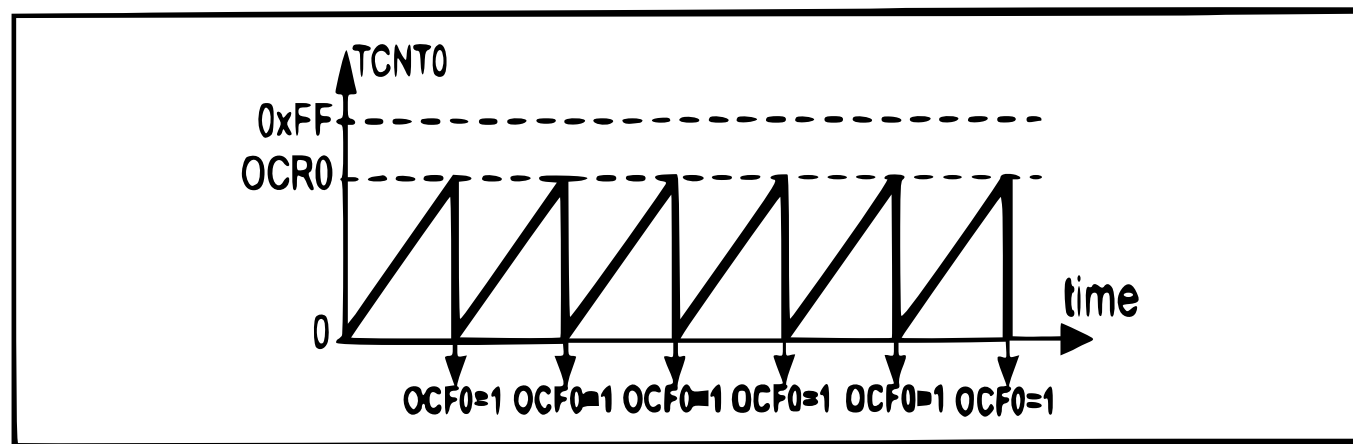
- El control del retardo recae en el registro de 8 bits TNCT0 y la frecuencia del Cristal XTAL.
- El AVR tiene la posibilidad de escalar la frecuencia del cristal por un factor de 8, 64, 256 y 1024.



Calcule el retardo generado por el siguiente programa con XTAL=8MHz

	LDI	R16, 0x08	;	—
	SBI	DDRB, 3	;	—
	LDI	R17, 0x00	;	—
	OUT	PORTB, R17	;	—
BEGIN:	RCALL	DELAY	;	—
	EOR	R16, R17	;	—
	OUT	PORTB, R17	;	—
	RJMP	BEGIN	;	—
DELAY:	LDI	R20, 0x00	;	—
	OUT	TCNT0, R20	;	—
	LDI	R20, 0x05	;	—
	OUT	TCCR0, R20	;	—
AGAIN:	IN	R20, TIFR	;	—
	SBRS	R20, TOV0	;	SBRS Verifica el estatus de un bit, salta si TVO0=1—
	RJMP	AGAIN	;	—
	LDI	R20, 0x00	;	—
	OUT	TCCR0, R20	;	—
	LDI	R20, (1<<TOV0)	;	—
	OUT	TIFR, R20	;	—
	RET		;	—

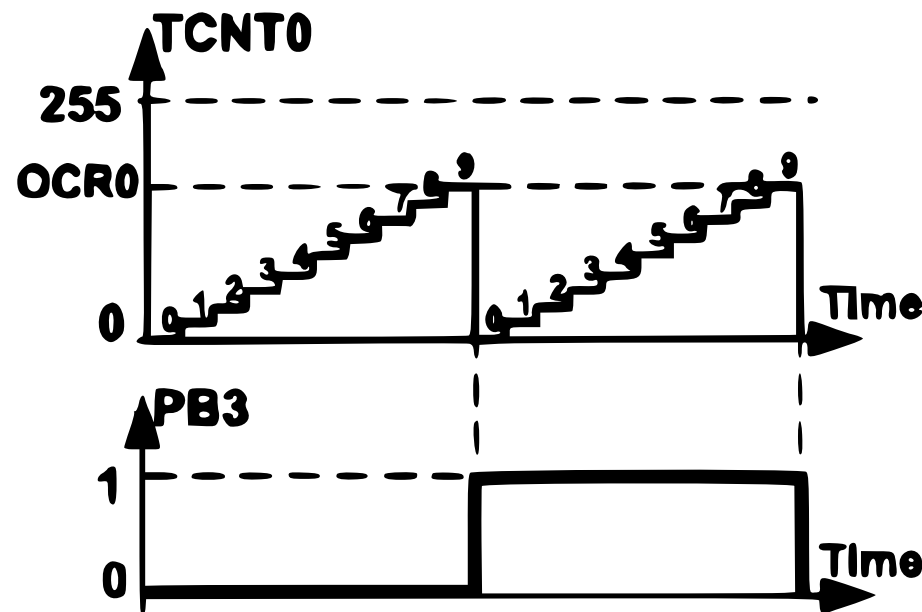
- El registro OCR0 se utiliza con el modo CTC. Al igual que en el modo normal, en el modo CTC, el timer se incrementa con el reloj, pero ahora cuenta hasta que el contenido del registro TCNT0 sea igual al contenido de OCR0.
- En esta punto la bandera OCF0 se activa en el siguiente ciclo de reloj. La bandera OCF0 esta localiza en el registro TIRF.



Determine la salida generada por el siguiente programa

	LDI	R16, 0x08	;	—
	SBI	DDRB, 3	;	—
	LDI	R17, 0x00	;	—
	OUT	PORTB, R17	;	—
BEGIN:	OUT	PORTB, R17	;	—
	RCALL	DELAY, R17	;	—
	EOR	R17, R16	;	—
	RJMP	BEGIN	;	—
DELAY:	LDI	R20, 0x00	;	—
	OUT	TCNT0, R20	;	—
	LDI	R20, 9	;	—
	OUT	OCR0, R20	;	—
	LDI	R20, 0x09	;	—
	OUT	TCCR0, R20	;	—
AGAIN:	IN	R20, TIFR	;	—
	SBRS	R20, OCFR0	;	—
	RJMP	AGAIN	;	—
	LDI	R20, 0x00	;	—
	OUT	TCCR0, R20	;	—
	LDI	R20, 1<<OCF0	;	—
	OUT	TIFR, R20	;	—
	RET		;	—

- OCR0 se carga con 9 y TCNT0 con 0. Después de 9 pulsos de reloj $TCNT0 = OCR0$. En el siguiente ciclo, la bandera OCF0 es igual a 1 y ocurre el reset. Esto significa que TCNT0 se limpia después de 10 ciclos.
- Ya que $XTAL = 8MHz$, el contador cuenta cada $0.125 \mu s$. Por lo tanto, tenemos que $10 \times 0.125 \mu s = 1.25 \mu s$ es el valor del retardo.



Aunque los Timer0 y Timer2 son de 8 bits, existen dos diferencias entre ellos.

- El Timer2 puede utilizarse como un contador de tiempo real conectando un cristal extra de 32.768KHz en los pines TOSC1 y TOSC2.
- Los bits en el registro TCCR tienen diferente significado.

- El Timer1 es de 16 bits, sus registros se dividen en dos bytes.
- TCNT1L-TCNT1H Bytes bajo y alto. TCCR1A y TCCR1B Registros de control del Timer/Contador. OCCR1A y OCCR1B. Los registros OCR están compuestos de dos bytes. Por ejemplo, OCR1A esta compuesto de OCR1AH (byte alto) y OCR1AL(byte bajo).
- Hay un registro auxiliar llamado ICR1 que se utiliza en operaciones de captura. ICR1 es un registro de 16 bits con ICR1H y ICR1L.

Determine la salida generada por el siguiente programa

	LDI	R16, 0x20	;	—
	SBI	DDRB, 5	;	—
	LDI	R17, 0x00	;	—
	OUT	PORTB, R17	;	—
BEGIN:	RCALL	DELAY	;	—
	EOR	R17, R16	;	—
	OUT	PORTB, R17	;	—
	RJMP	BEGIN	;	—
DELAY:	LDI	R20, 0xD8	;	—
	OUT	TCNT1H, R20	;	—
	LDI	R20, 0xF0	;	—
	OUT	TCNT1L, R20	;	—
	LDI	R20, 0x00	;	—
	OUT	TCCR1A, R20	;	—
	LDI	R20, 0x01	;	—
	OUT	TCCR1B, R20	;	—
AGAIN:	IN	R20, TIFR	;	—
	SBRS	R20, TOV1	;	—
	RJMP	AGAIN	;	—
	LDI	R20, 0x00	;	—
	OUT	TCCR1B, R20	;	—
	LDI	R20, 0x04	;	—
	OUT	TIFR, R20	;	—
	RET		;	—