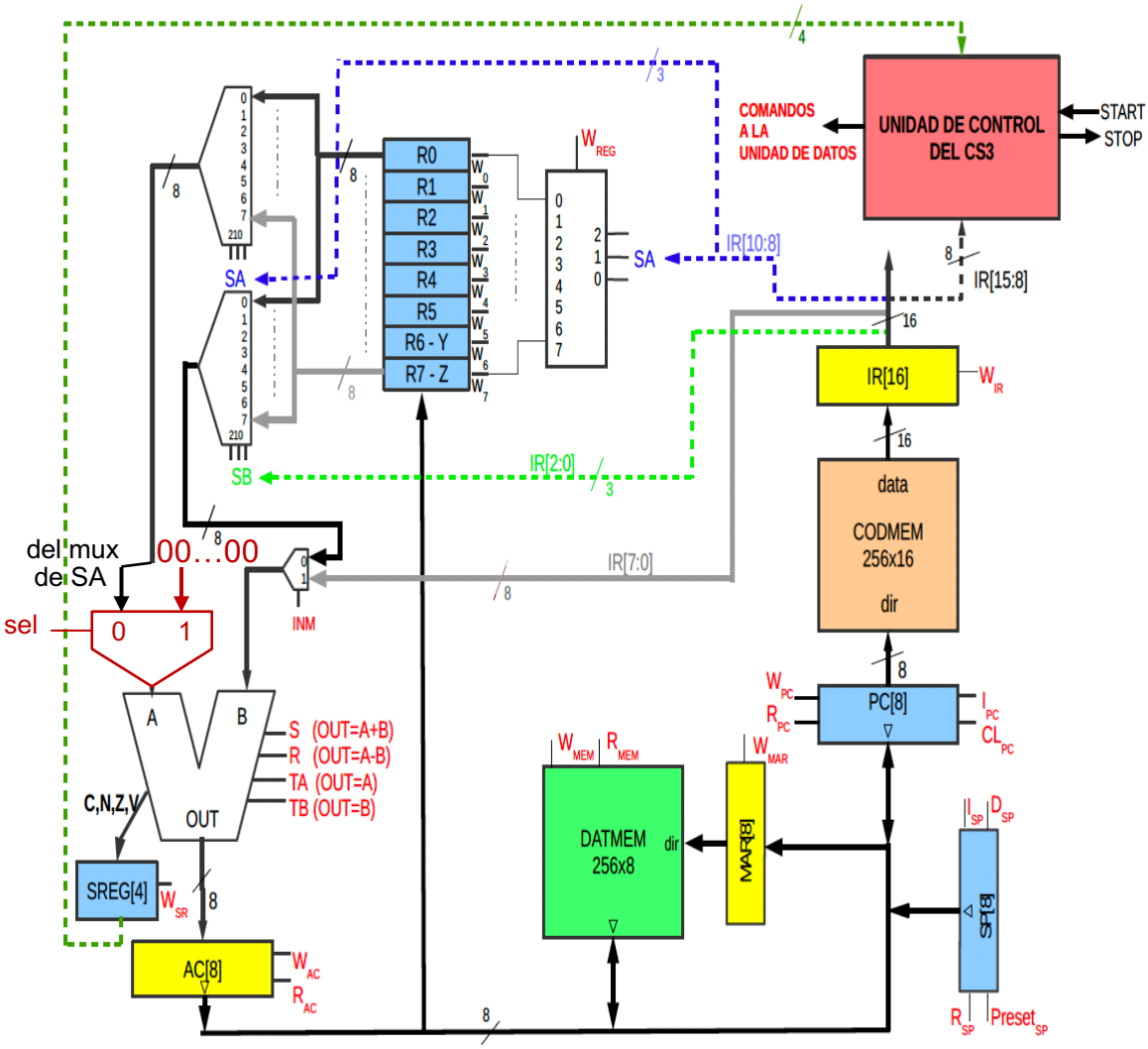
ALUMNO:_____

1. Se desea añadir la instrucción NEG Rd al CS3. Esta instrucción obtiene el complemento a 2 del dato contenido en Rd y lo almacena en Rd, sin modificar el contenido de ningún otro registro visible. A nivel RT, podríamos expresar la instrucción como Rd ←0 – Rd.

   - Explique si esto es posible (sin modificar la ALU) y describa en su caso los cambios que sería necesario hacer sobre la arquitectura original del CS3.

   Dado que queremos hacer la operación 0 – Rd, incluimos un MUX que nos permita elegir 00000000 como dato A de la ALU. Véase en la figura.



   - Describa el formato de instrucción y el código de operación elegidos.

   El formato sería el Formato A donde será necesario codificar Rd tanto en $IR_{10:8}$ como en $IR_{2:0}$ ya que este registro es fuente y destino simultáneamente. Utilizaremos como CODOP 10000 ya que no está usado por ninguna otra instrucción.

100000	Rd	- - - - -	Rd

- Escriba el código máquina de la instrucción NEG R5

La instrucción NEG R5 se codificaría, por tanto, como:
10000 101 00000 101, en hexadecimal: $8505

- Indique la secuencia de microoperaciones que deben realizarse durante la fase de ejecución de la nueva instrucción (muestre tanto las transferencias a nivel RT como las señales que debe activar la unidad de control)

La secuencia de microoperaciones y señales de control son:

1. $AC \leftarrow 0 - R(IR_{2:0})$, Wac, sel, R
2. $R(IR_{10:8}) \leftarrow AC$, Wreg, Rac

2. Se necesita medir, en ms, el tiempo de duración de un pulso que se produce en una señal digital conectada al pin PB2 del microcontrolador.
Cuando se detecta un cambio a 1 en dicha señal (comienzo del pulso), el micro podrá esperar 20 ms antes de volver a chequear el pin PB2. Con esto, damos por hecho que no habrá pulsos menores de 20 ms. Cuando la señal vuelva a 0 (el pulso termina), la duración total del pulso en ms debe mostrarse en los pines PD7:0 y seguir mostrándose hasta que una nueva medición se produzca. Tras cada medida, se volverá a iniciar la operación del sistema esperando un nuevo pulso en la señal conectada al pin PB2.

Realice un programa que se encargue de la tarea descrita, para ello no olvide:

(a) Configurar los puertos de entrada/salida.

```
cbi ddrb,2.          //PB2 entrada
ldi r16,$ff
out ddrd,r16         //PD7-0 salida
```

(b) Configurar el temporizador para que genere una interrupción cada ms sabiendo que la frecuencia del reloj del sistema es de 2Mhz.

A frecuencia 2Mhz → en 1s pasarían 2.000.000 de ciclos, en 1ms pasarían 2.000 haremos VMAX = 2000 (o 1999 que sería más exacto)

Haremos OCR1A ← 2000.

Seleccionaremos modo CTC ($TCCR1B_3$ = 1) y frecuencia ck/1 ($TCCR1B_{2-0}$= 0b001), aunque esto lo haremos en el programa principal.

Activaremos las interrupciones del temporizador haciendo $TIMSK1_1$= 1

```
.def tmp=r17
ldi tmp,high(2000)
sts ocr1ah,tmp
ldi tmp,low(2000)
sts ocr1al,tmp
ldi tmp,2          ;ldi tmp, 0b00000010
sts timsk1,tmp
ret
```

(c) Programar e instalar adecuadamente la rutina de servicio de interrupción.

```
.include "m328Pdef.inc”
.cseg
.org 0
jmp init
.org 0x16
jmp contadortiempo
```

```
                .def duración=r18

contadortiempo:  push r16
                 in r16,sreg

                 inc duración

                 out sreg,r16
                 pop r16
                 reti
```

(d) Desarrollar adecuadamente el programa completo.

```
init:   clr duracion
vale0:  sbic pinb,2
        rjmp vale0

        clr tmp
        sts tcnt1h,tmp
        sts tcnt1l,tmp

        sei
        ldi tmp, 0b1001
        sts tccr1b,tmp

vale1:  sbis pinb,2
        rjmp vale1

        out portd,duración
        rjmp init
```

3. (a) Para el CS3:

- Indique las diferencias existentes entre las instrucciones JMP y CALL.

  ver teoría

- Muestre las microoperaciones de los ciclos de búsqueda y ejecución para las instrucciones CALL dir, RET y LDI Rd,dato (especifique para cada ciclo de reloj tanto las transferencias a nivel RT como las señales que debe activar la unidad de control)

  ver teoría

- Obtenga el código máquina de las instrucciones CALL $BE, JMP 52 y LDI R4,25

  CALL $BE → 00100 000 1011 1110

  JMP 52 → 00111 000 00110100

  LDI R4,25→ 11111 100 00011001

(b)Para el AVR:

- escriba las instrucciones necesarias para realizar las siguientes tareas:
- escribir el contenido del registro R10 en el décimo byte de la memoria RAM

  STS 0X109,R10

- escribir en el registro R20 el contenido de la dirección $200 de la memoria de datos

  LD R20,$200

- escribir en el tercer registro de E/S el quinto byte de la memoria de datos

  OUT 3,R4

- inicializar el puntero X con el valor $2af

```
LDI XH,2
LDI XL,$AF
```

- suponiendo que el puntero Y contiene la dirección de memoria donde comienza una tabla de 16 bytes, cargar el décimo de dichos elementos en el registro R2

```
LDD R2,Y+9
```

- escribir el contenido del registro DDRC en el registro DDRB

```
IN R16,DDRC
OUT DDRB,R16
```

- poner a 0 el bit de overflow del registro SREG

```
CLV
```

- saltar a la etiqueta “etiq1” si el contenido del registro R2 es un número sin signo mayor que el contenido en R3

```
CP R2,R3
BRCC etiq1
```

- llamar a una subrutina cuya dirección de comienzo se encuentra almacenada en el puntero Z

```
ICALL
```