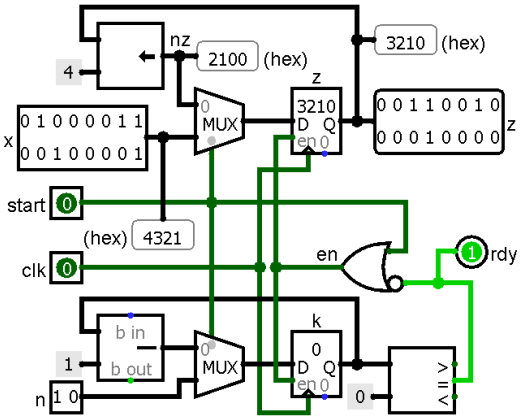
Pregunta 1

I. Considere el circuito de la figura. **Complete** la tabla de abajo con los valores que toma cada señal hasta el ciclo 6 del reloj. El valor de la entrada *n* se mantiene constante en 2 (en binario 10) y la entrada *x* es 0x4321 (en binario 0100 0011 0010 0001). Las columnas para *z* y *nz* deben estar en hexadecimal y la columna *k* en decimal.



ciclo	*start*	*k*	*z*	*rdy*	*nz*	*en*
1	0	0	3210	1	2100	0
2	1	0	3210	1	2100	1
3	0		4321			
4	0					
5	0					
6	0					

II. **Modifique** el circuito de manera que cuando la salida *rdy* es 1, la salida *z* sea el resultado de rotar la entrada *x* en *n* posiciones hexadecimales hacia la izquierda. Por ejemplo si *start* es 1, *x* es 0x**4**321 y *n* es 1, cuando rdy sea 1 nuevamente *z* debe ser 0x321**4**. Si *n* es 3, *z* será 0x1432. Dibuje solo la parte que necesita modificar, más una que otra compuerta para mostrar el contexto en donde introduce la modificación.

Pregunta 2

A) La figura muestra un extracto del estado actual de un *caché* de 4 KB (2^{12} bytes) de 1 grado de asociatividad con 256 líneas de 16 bytes. Por ejemplo en la línea 4c del caché (en hexadecimal) se almacena la línea de memoria que tiene como etiqueta b4c (es decir, la línea que va de la dirección b4c0 a la dirección b4cf).

línea cache	*etiqueta*	*contenido*
15	915	
4c	b4c	
e0	5e0	

Un programa accede a las siguientes direcciones de memoria (en hexadecimal): 5e00, 9158, b4cc, 5150, b4c0, de00, 5158, b4c4 y 9150. **Indique** qué accesos a la memoria son aciertos en el caché, cuáles son desaciertos y **rehaga la figura** mostrando el estado final del caché. Por ejemplo el acceso 5e00 es un acierto.

B) La tabla de abajo muestra un programa en assembler Risc-V y su ejecución.

Programa		*Ejecución*		
		Fetch	Dec	Exe
A addi s1,t3,1	...	1 AB		
B and t0,s1,t4	U addi s1,t0,15	2 CD	AB	
C sub a1,t0,a2	V ori s4,t6,31	3		A
D blt t0,s1,U	W ...	4 UV	CD	B
E add a3,t1,s2	X ...	4 WX	UV	CD
F andi a4,t2,15	Y ...	5 YZ	WX	UV
G add a5,a4,a3	Z ...			
H or t4,a5,t3				

Conteste lo siguiente: (i) Determine el tipo de arquitectura a partir de la columna *Ejecución*. (ii) ¿Por qué las instrucciones C y D se ejecutan en paralelo, pero no A y B? (iii) Explique si esta arquitectura tiene o no predicción de saltos. (iv) Rehaga la columna *Ejecución* para un arquitectura de un solo pipeline sin predicción de saltos. El salto D sí se produce.

Pregunta 3

La función *monedas* del cuadro de la derecha recibe un arreglo *a* con *n* valores de muchas monedas. Cada valor solo puede ser 5, 10, 50, 100 y 500. También recibe un arreglo *sumas* de 6 enteros. Esta función contabiliza el número de monedas de 5 pesos en *sumas[0]*, las de 10 pesos en *sumas[1]*, etc. **Programe** la función *monedasPar* con la misma funcionalidad de *monedas*, pero realizando la contabilización en paralelo con *p* procesos creados con *fork*. El encabezado es:

```
void monedas(int *a, int n,
             int *sumas) {
  for (int k= 0; k<6; k++)
    sumas[k]= 0;
  for (int i= 0; i<n; i++){
    if (a[i]==5)
      sumas[0]++;
    else if (a[i]==10)
      sumas[1]++;
    else if (a[i]==50)
      sumas[2]++;
    else if (a[i]==100)
      sumas[3]++;
    else if (a[i]==500)
      sumas[4]++;
    else
      sumas[5]++;
} }
```

*void monedasPar(int *a, int n, int *sumas, int p);*

Ayuda: Invoque *fork* para crear *p* nuevos procesos. Cada proceso contabiliza monedas en un subintervalo del arreglo *a*. Use un pipe para que cada hijo envíe al padre sus resultados. El padre se encarga de sumar los resultados de los hijos al arreglo *sumas*. Por simplicidad suponga que *n* es múltiplo de *p*. **¡Revise que haya paralelismo!**