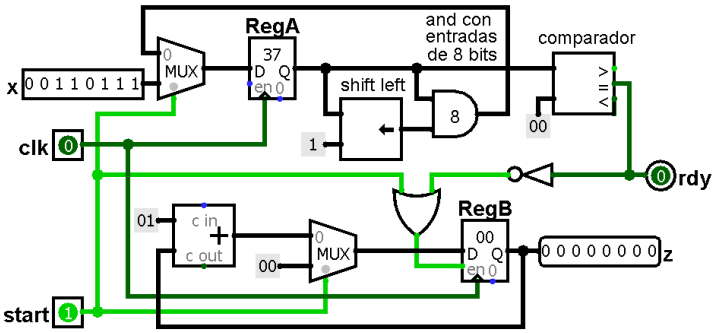
Pregunta 1

I.- La figura muestra un circuito con entradas *start*, *clk* y *x*, y con salidas *rdy* y *z*.



La entrada *x* es 0b110111 y se mantiene constante. Un ciclo del reloj inicia con el cambio de 1 a 0 de *clk* y termina con el siguiente cambio de 1 a 0 de *clk*. En el ciclo 1 del reloj *start* se pone en 1, en el ciclo 2 *start* se pone en 0 y luego se mantiene constante en 0. Indique los valores en binario del registro *RegA*, la salida de *shift left* y las salidas *rdy* y *z* en los ciclos 2, 3, 4 y 5.

Ayuda: Recuerde que un registro solo cambia de valor en la transición de 1 a 0 del reloj cuando la entrada *en* es 1 o simplemente está desconectada.

II.- Traduzca la función del cuadro de la derecha a assembler Risc-V. Puede optimizar el código en assembler para reducir la cantidad de instrucciones.

```
int f(unsigned int x) {
  int hi=31,lo=0,k= 16;
  while (lo!=k) {
    if ( (x>>k) == 0 )
      hi= k-1;
    else
      lo= k;
    k= (hi+lo+1)/2;
  }
  return k;
}
```

Pregunta 2

Programe la función *sinDigitos* con el siguiente encabezado:

```
void sinDigitos(char *s);
```

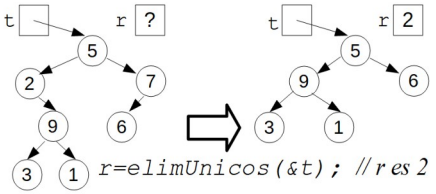
Esta función recibe como parámetro un string *s*. Ud. debe eliminar de *s* todos las cifras numéricas. Ejemplo:

```
char s[]= "ab+-43cd15AB9";
sinDigitos(s);  // s es "ab+-cdAB"
```

Restricciones: No puede invocar otras funciones predefinidas como *strlen*, *strcpy*, *etc*. No use el operador de subindicación de arreglos [ ] ni su equivalente **(p+i)*, use aritmética de punteros como *p*++ o *p*+*i*. No puede pedir memoria adicional con *malloc* ni declarar arreglos. Sí deberá declarar punteros adicionales.

Pregunta 3

```
typedef struct nodo {
  int x;
  struct nodo *izq, *der;
} Nodo;
int elimUnicos(Nodo **pa);
```



Programe la función *elimUnicos* con el encabezado del cuadro de la derecha. Esta función recibe en **pa* un árbol binario (no ordenado) y elimina todos los nodos que tengan un solo hijo. Debe retornar el número de nodos que fueron eliminados. En el ejemplo de la figura el tipo de *t* es *Nodo** y de *r* es *int*. El nodo con etiqueta 2 es eliminado porque su único hijo es el nodo de etiqueta 9. Por la misma razón se elimina el nodo de etiqueta 7. Los nodos 3, 1 y 6 no se eliminan porque no tienen hijos.

Metodología obligatoria: Sea *a*= **pa*. El caso en que *a* es el árbol vacío es trivial. Elimine recursivamente los nodos que tienen un solo hijo del subárbol izquierdo y del subárbol derecho. Si ahora *a* tiene un solo nodo hijo, haga que **pa* apunte hacia ese único nodo y libere con *free* la memoria ocupada por el nodo *a*. Calcule Ud. eficientemente el número de nodos eliminados.

Pregunta 4

```
double prodPunto(
    double *a,
    double *b, int n) {
  double sum= 0.0;
  for (int i=0;i<n;i++)
    sum += a[i]*b[i];
  return sum;
}
```

La función *prodPunto* del cuadro de la derecha calcula el producto punto de 2 vectores representados como arreglos. Recibe como parámetros los 2 arreglos *a* y *b*, y el número de elementos *n* de ambos arreglos. Reprograme la función *prodPunto* de manera que se use *fork* para realizar el cálculo con *p* cores. El proceso padre debe usar *fork* para lanzar *p* procesos. El encabezado de la función debe ser:

```
double prodPunto(double *a, double *b, int n, int p);
```

Ayuda: Por simplicidad considere que *n* es múltiplo de *p*.