PROYECTO DE ARQUITECTURA DE COMPUTADORES II
PRIMERA ENTREGA

INTEGRANTES

WILLIAM AGUIRRE ZAPATA
HECTOR DAVID AMATURE
JOSE DANILO MELGAREJO
JUAN CAMILO VANEGAS

FECHA ENTREGA

17/09/2018

PRESENTADO A

MARIBELL SACANAMBOY FRANCO

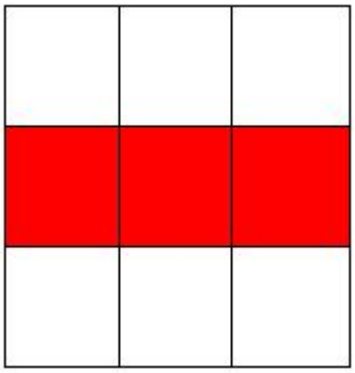
PONTIFICIA UNIVERSIDAD JAVERIANA

CALI-VALLE DEL CAUCA

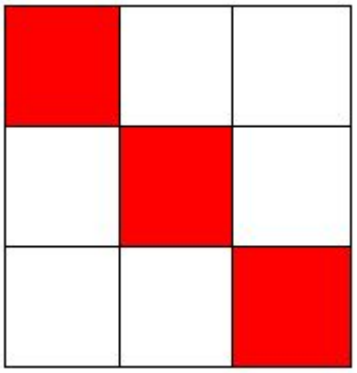
INTRODUCCIÓN

En nuestro proyecto planteamos hacer una máquina tragamonedas, la cual en nuestro caso recibe billetes con el propósito de realizar una apuesta definida por el usuario con el objetivo de duplicar o triplicar sus ganancias dependiendo de el caso, estos están definidos por una matriz 3 x 3 donde cada casilla tiene la posibilidad de dar un numero aleatorio entre 0 y 9, estos números están representados visualmente con un color cada uno, cuando se de uno de los siguientes casos de combinación de colores, se genera una victoria y dependiendo del caso el que tanto se incrementa la ganancia, partiendo desde la apuesta ya establecida, ahora a continuación se presentarán los casos de victoria:

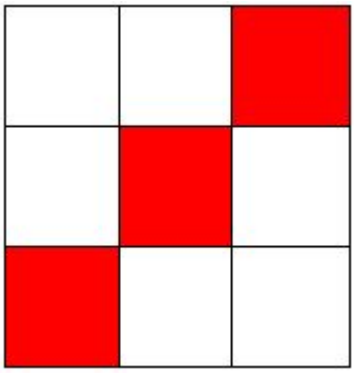
CASO 1



CASO 2



CASO 3

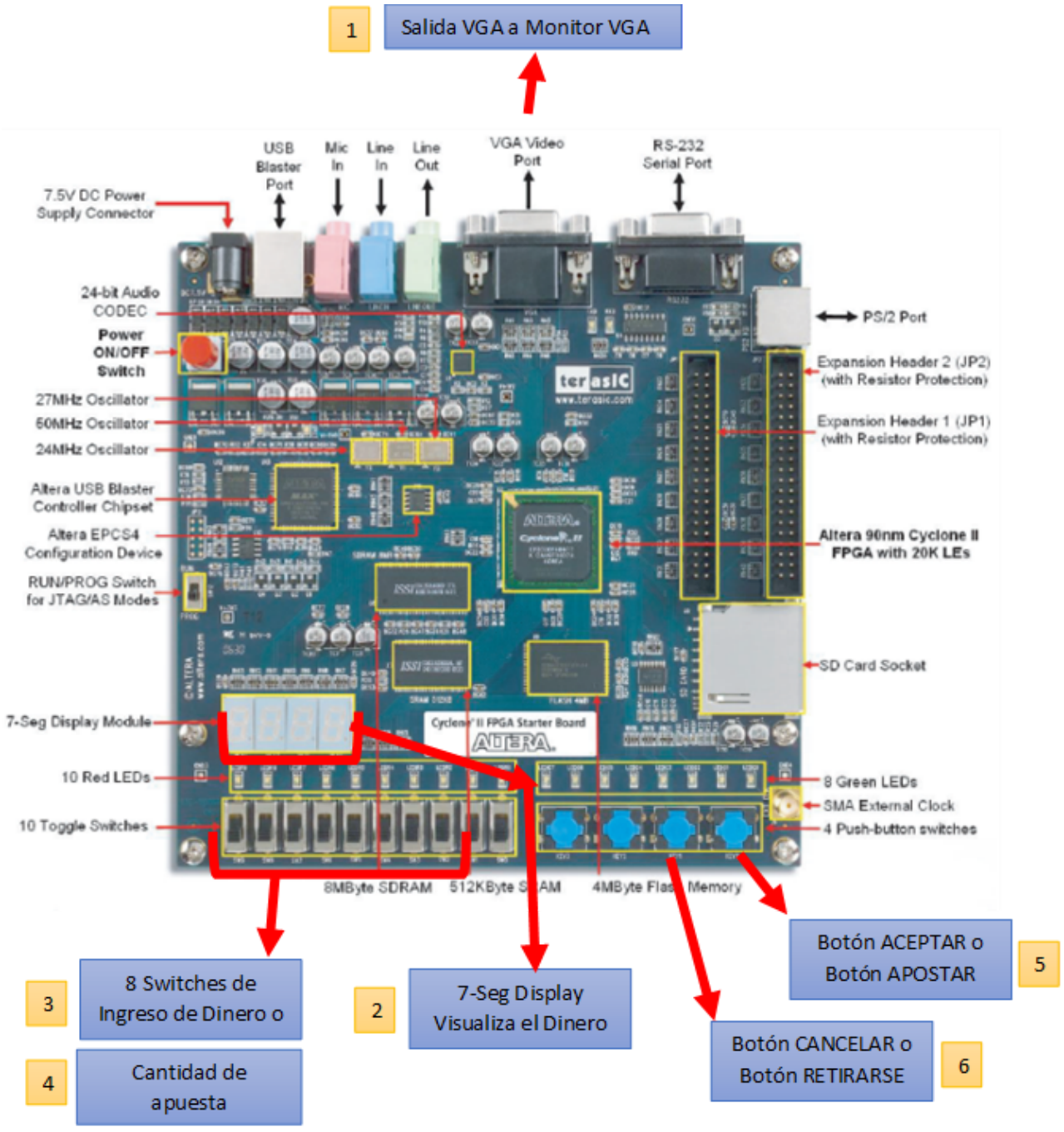


En cada unos de estos casos se genera el estado de victoria pero el caso uno genera un X3 en la apuesta hecha mientras que el caso 2 y 3 generarán un X2 en la apuesta hecha, en el caso que no se presente ninguno de estos la apuesta se pierde y si se tiene dinero aun, se tiene la posibilidad de poder volver a apostar.

El jugador puede retirarse en cualquier momento con tal de que este no este en medio de una jugada, y puede volver a ingresar dinero para volver a apostar cuando él lo desee, el ingreso de el dinero y las apuestas se manejan desde la FPGA y el apartado gráfico se tratara de representar en pantalla para mostrar si se genero una victoria o una pérdida.

CRITERIOS DE DISEÑO

A la hora de jugar se usarán los interruptores, los botones, y los display 7 segmentos con los que cuenta la FPGA y si es posible implementar una interfaz gráfica que haría necesario el uso de un monitor. Se ingresaría el dinero a apostar por medio de los interruptores, se usarían los botones para seleccionar la apuesta o si se desea retirarse y en los 7 segmentos se mostrarían todos los posibles valores numéricos involucrados mientras una persona juega (Saldo total, valor a apostar). Para el manejo de la interfaz gráfica se propone mostrar en pantalla una cuadrícula 3x3 donde se muestren colores en cada una de las celdas representando el resultado obtenido a la hora de apostar; si resulta imposible realizarlo de esta manera se acudiría a mostrar en pantalla algo que represente si el jugador ganó o perdió.



1) **Salida VGA a Monitor VGA:** Puerto en el que se podrá conectar un monitor externo para la visualización de las combinaciones en cada jugada.

2) **7-Seg. Display:** Se utiliza para una representación visual del valor numérico del Dinero y la Apuesta dada una combinación de los switches.

3) **8 Switches de Ingreso de Dinero:** Se utilizan 8 switches de tal manera que estén distribuidos de la siguiente forma (De izquierda a derecha), primer switch = 5*10^3, el segundo = 1*10^3, el tercero =5*10^2, el cuarto = 1*10^2, el quinto = 5*10^1, el sexto =1*10^1, el séptimo =5*10^0 y el octavo = 1*10^0. Se restringe a 8 el número de switches para que el valor máximo ingresado pueda ser mostrado en los 4 7-segmentos.

4) **8 Switches de Cantidad de Apuesta:** Por las mismas razones anteriores, pero con el propósito de ingresar la cantidad de la apuesta en cada jugada.

5) **Botón ACEPTAR o APOSTAR:** Después de ingresar la cantidad, sirven como una señal de apoyo a la unidad de control para permitir pasar el dato a los registros.

6) **Botón CANCELAR o RETIRARSE:** Es un botón que funciona como un reset. Para que vuelva a iniciar el juego desde cero.

Utilizaremos dos memoria, una para instrucciones y otra para datos, porque de esta manera podemos manejar anchos y largos diferentes en cada memoria y no tener que manejar datos excesivamente grandes para cantidades pequeñas

TAMAÑO DE LOS DATOS : El tamaño de los datos que tendrán la memoria de datos y los registros es de 14 bits porque todos los números que vamos a representar están incluidos en ese rue van desde -8191 a 8191 utilizando un bit de signo.

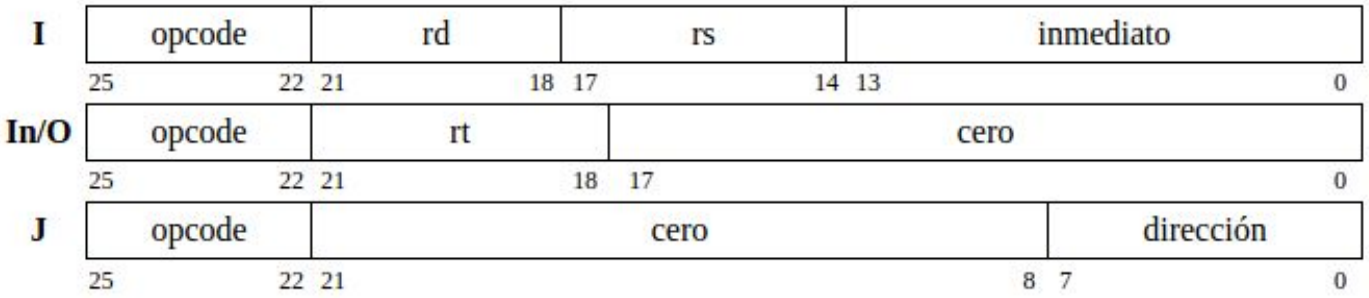
LARGO DE MEMORIA DE INSTRUCCIONES: El largo de la memoria de instrucciones es de 8 bits debido a que el programa a ejecutar no sobrepasa las 200 instrucciones.

ANCHO DE MEMORIA DE INSTRUCCIONES: El ancho de la memoria de instrucciones es de 26 bits debido a que ese es el tamaño de una instrucción completa.

LARGO DE MEMORIA DE DATOS: El largo de la memoria de datos es de 5 bits debido a que se van a guardar en memoria menos de 32 valores.

DESCRIPCIÓN Y FORMATO DE INSTRUCCIONES Y REGISTROS

FORMATO BÁSICO DE INSTRUCCIONES



OPERACIONES DE LA ALU

ALUOP	
+	00
-	01
X	10
0+	11

INSTRUCCIONES CON LAS RESPECTIVAS SEÑALES DE LA UNIDAD DE CONTROL

INSTRUCCIONES																												
SEÑALES	PARA TODAS		OM			OSS			INS		J	BEQ	BNQ	BGT	SM		LM			ADDI		ADD		MUL		SUB	
	Fetch	Decode	Dir	RM	Out	Dir	RM	Out	Input	WM					Cal.Dir	Guar.Dir	Cal.Dir	Leer.Mem	Gua.Mem	Execute	W.Register	Execute	W.Register	Execute	W.Register	Execute	W.Register
PCWRITE	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
BEQ	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
BNE	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
BGT	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
JUMP	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RMI	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
WIR	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
DG	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
WRF	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	1	0	1	0
ALUOP	00	00	11	00	00	11	00	00	11	00	00	01	01	01	00	00	00	00	00	00	00	00	00	10	00	01	00	
ALUSRCA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1	0	0	0	0	0	0	0	0
ALUOUT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1	0	1	0	1	0	1	0	1
DATSEL	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RMD	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
WMD	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
IN/OUT	00	00	00	00	11	00	0	10	01	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

CONJUNTO DE INSTRUCCIONES BÁSICAS

INSTRUCCIONES				OPCODE	TIPO
INS	$rt			0000	In/O
ADDI	$rd	$rs	i	0001	I
ADD	$rd	$rs		0010	I
MUL	$rd	$rs		0011	I
SM	$rd	$rs	i	0100	I
LM	$rd	$rs	i	0101	I
BEQ	$rd	$rs	i	0110	I
BNO	$rd	$rs	i	0111	I
BGT	$rd	$rs	i	1000	I
SUB	$rd	$rs		1001	I
J	i			1010	J
OSS	$rt			1011	In/O
OM	$rt			1100	In/O

NOMBRE DEL REGISTRO, NÚMERO, USO

NOMBRE	NÚMERO	USO
$0	0	Valor Constante 0
$r	1	Random
$2	2	Valor Constante 2
$3	3	Valor Constante 3
$s0	4	Valor del Dinero
$s1	5	Valor de la Apuesta
$s2	6	Valor de la Bonificación
$t0-$t8	7-14	Temporales

PROGRAMA A EJECUTAR POR EL PROCESADOR EN LENGUAJE ENSAMBLADOR

Versión funcional del programa en Lenguaje Ensamblador MIPS con funciones Syscall https://github.com/monowilliam/Tragamonedas.git

Versión del programa en Lenguaje Ensamblador para implementar en la FPGA

```
ingresarD:
    addi $s0,$cero,0
    addi $t0,$cero,0
    ins $t0
    lm $t0,$t0,0
    beq $t0,$cero,0
    addi $s0, $t0,0
    addi $t0,$cero,0
    oss $t0
jugarORet:
    addi $t0,$cero,3
    ins $t0
    lm $t0,$t0,0
    addi $t1,$cero,1
```

```
    beq $t0,$t1,10 #Va a realizar apuesta
    beq $t0,$cero,0  #Vuelve a pedir el dinero para iniciar
realizarA:
      addi $t0,$cero,1
      ins $t0
      lm $t0,$t0,0
      bgt $t0,$s0,9 #Vuelve a pedir la apuesta porque el numero ingresado es mayor
al dinero
      beq $t0,$cero,9 #Vuelve a pedir la apuesta porque el numero ingresado es cero.
      sm $t0,$cero,1
      addi $s1,$t0,0 #S1= A
      addi $t1,$cero,1
      oss $t1
      sub $s0,$s1 #Dinero=Dinero-A
      sm $s0,$cero,0
llenarM:
addi $t2,$cero,0 # i
addi $t4,$cero,0 # mem
# Cada salto de aquí en adelante estará con una etiqueta debido a que no sabemos
cuánto  espacio nos consumirá el código de random.
llenarfilas:beq $t2,$3,comparacion #Salta a comparacion
    addi $t3,cero,0 # j
        llenarcol: beq $t3,$3,fincol
        #En r estará el número random
        sm $r,$t4,4 #De la posición 4-12  de la memoria de datos se encontrará la
matriz
        addi $t5,$t4,4
        om $t5
        addi $t3,$t3,1
        addi $t4,$t4,1
        j llenarcol
    fincol:
    addi $t2,$t2,1
    j llenarfilas
comparacion:
    addi $s2,$cero,1 #s2=Bonificacion
    sm $s2,$cero,2
    addi $t7,$cero,4
    lm $t0,$t7,0
    lm $t1,$t7,2
    lm $t2,$t7,3
    lm $t3,$t7,4
```

```
    lm $t4,$t7,5
    lm $t5,$t7,6
    lm $t6,$t7,8
    bne $t0,$t3,falsed1
    bne $t3,$t6,falsed1
    mul $s2,$2
    falsed1:
    bne $t1,$t3,falsed2
    bne $t3,$t5,falsed2
    mul $s2,$2
    falsed2:
    bne $t2,$t3,falsecen
    bne $t3,$t4,falsecen
    mul $s2,$3
    falsecen:
    addi $t0,$cero,1
    sm $s2,$cero,2 #El valor de la bonificación se guarda en memoria
    bne $s2,$t0,win
    addi $t1,$cero,0
    oss $t1
    bne $s0,$cero,jugarORet
    j ingresarD
win:
     mul $s1,$s2 #A=A*Bonificacion
     add $s0,$s1 #Dinero=Dinero+A
     addi $t1,$cero,0
     oss $t1
     sm $s0,$cero,0 #El dinero actualizado se guarda en memoria
     sm $cero,$cero,1 #Se guarda en memoria 0 en el espacio donde está la apuesta
     j jugarORet
```

DIAGRAMA ESTRUCTURAL DE LA ARQUITECTURA DEL PROCESADOR

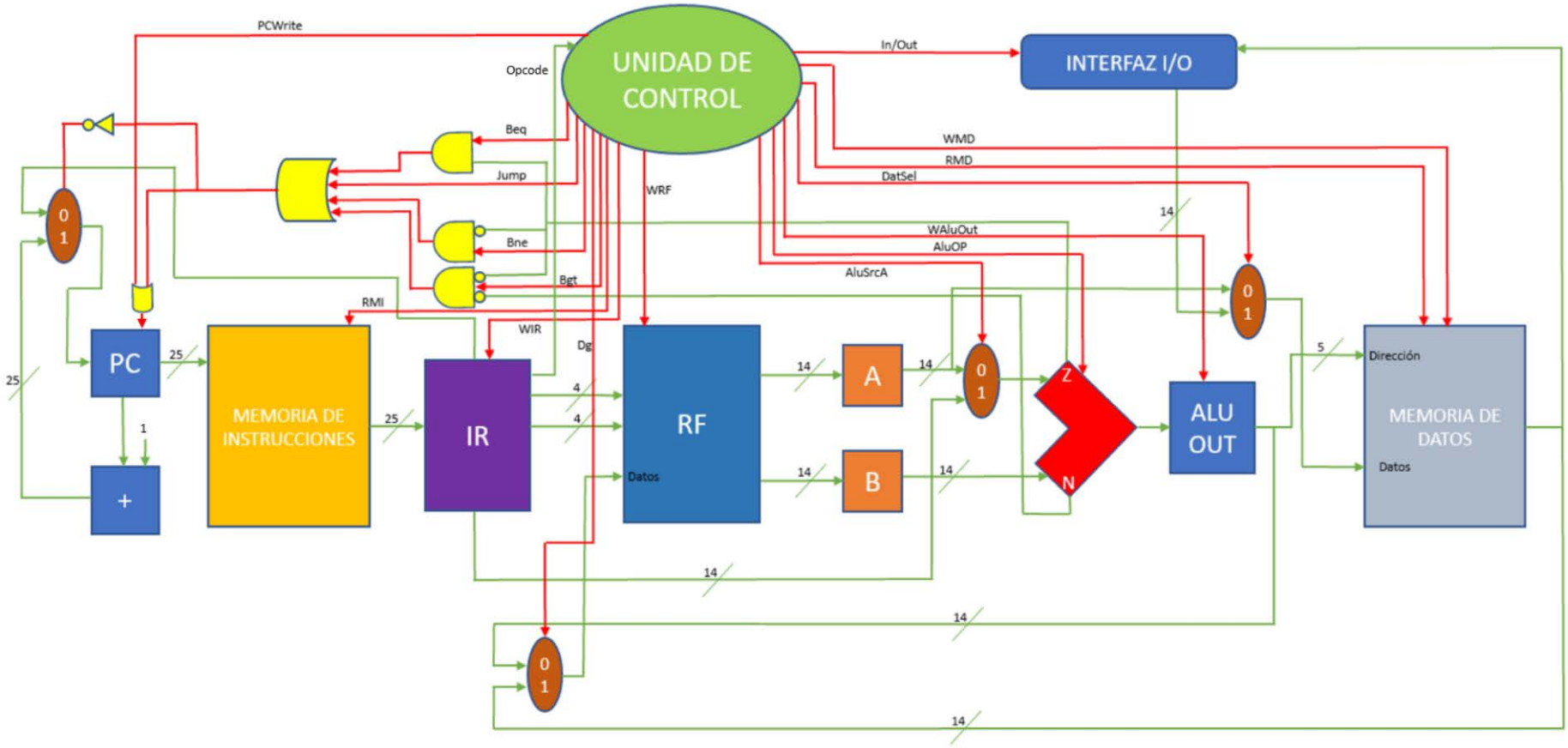


DIAGRAMA DE FLUJO DEL PROGRAMA A EJECUTAR POR EL PROCESADOR