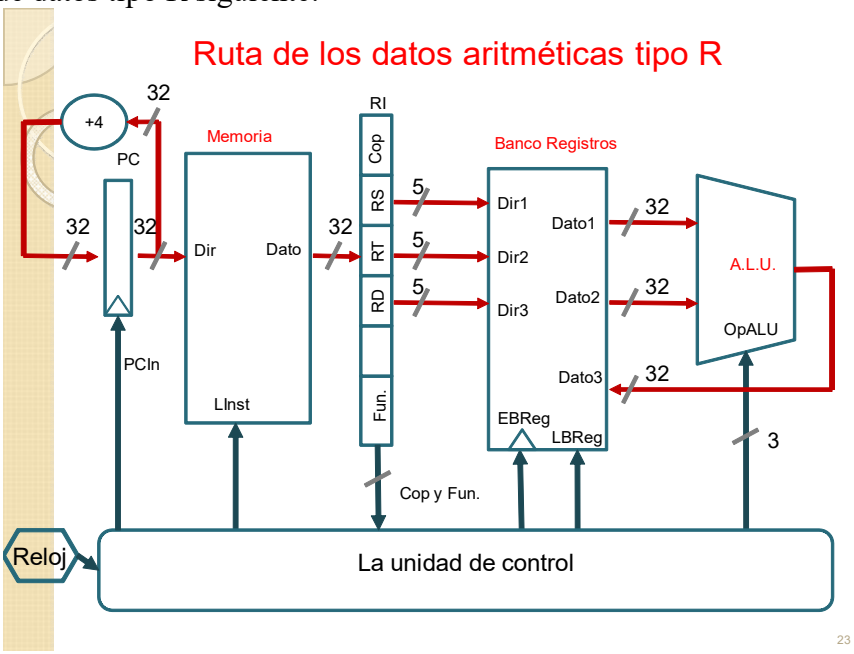
Ejercicio 1

Dada la ruta de datos tipo R siguiente:



ALU dispone de estas operaciones y su tabla de verdad es:

```
000    a ∧ b (and)
001    a ∨ b (or)
010    a + b (suma aritmética)
110    a − b (resta)
111     a < b (menor que)
```

a) Rellene la tabla de verdad de la unidad de control que ejecutaria las instrucciones aritméticas tipo R indicadas:

			PC	Memoria	Banco Registros		ALU
Instrucción	Código Operación	Función	PCin	LInst	EBReg	LBReg	OpALU
add rd, rs, rt	000000	100000	1	1	1	1	010
sub rd, rs, rt	000000	100010	1	1	1	1	110
and rd, rs, rt	000000	100100	1	1	1	1	000
or rd, rs, rt	000000	100101	1	1	1	1	001
slt rd, rs, rt	000000	101010	1	1	1	1	111

a) Calcule el tiempo de ciclo de la unidad de control. Asuma que Memoria, Banco de registros y UAL tardan 20ns por operación. Los registros y sumador de la dirección tiempo despreciable. $T_{MI} + T_{BR(leer)} + T_{ALU} + T_{BR(escribir)} = 4 \times 20 = 80$ ns

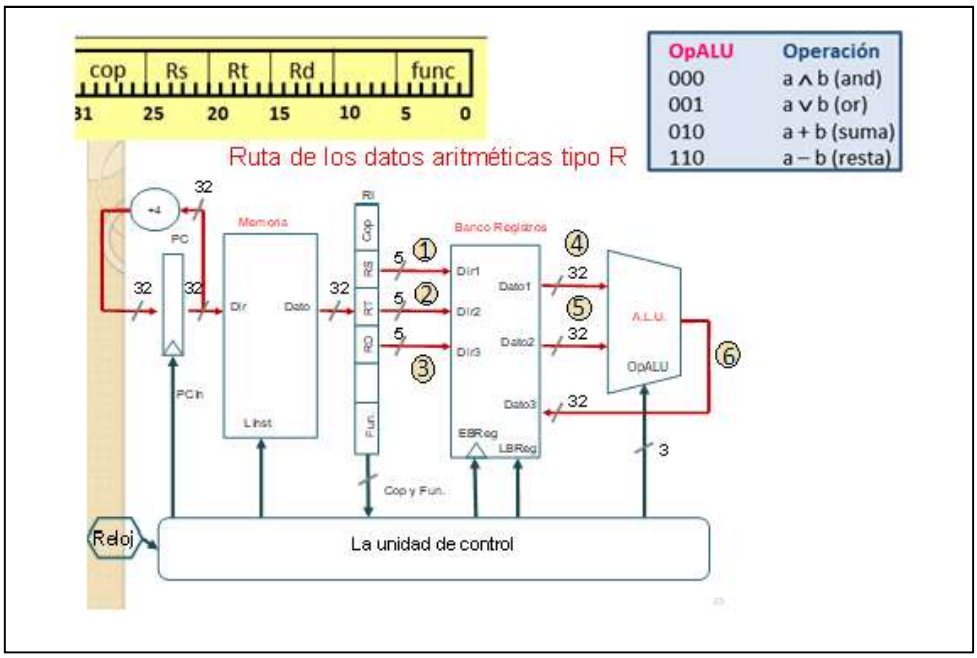
Ejercicio 2

Considere la siguiente instrucción de código en ensamblador del MIPS R2000, teniendo en cuenta que el contenido del registro $13 es 7 y del $10 es -4:

```
add $8, $10, $13
```

a) Indique el valor de las líneas marcadas con un círculo en la ruta de datos de la figura siguiente en el momento de la ejecución de la instrucción ***add***. Exprese los valores en decimal. El formato de la instrucción add se muestra a junto con la ruta de datos:

Punto	1	2	3	4	5	6
Valor	10	13	8	-4	7	3



b) Rellene el valor de las señales de control:

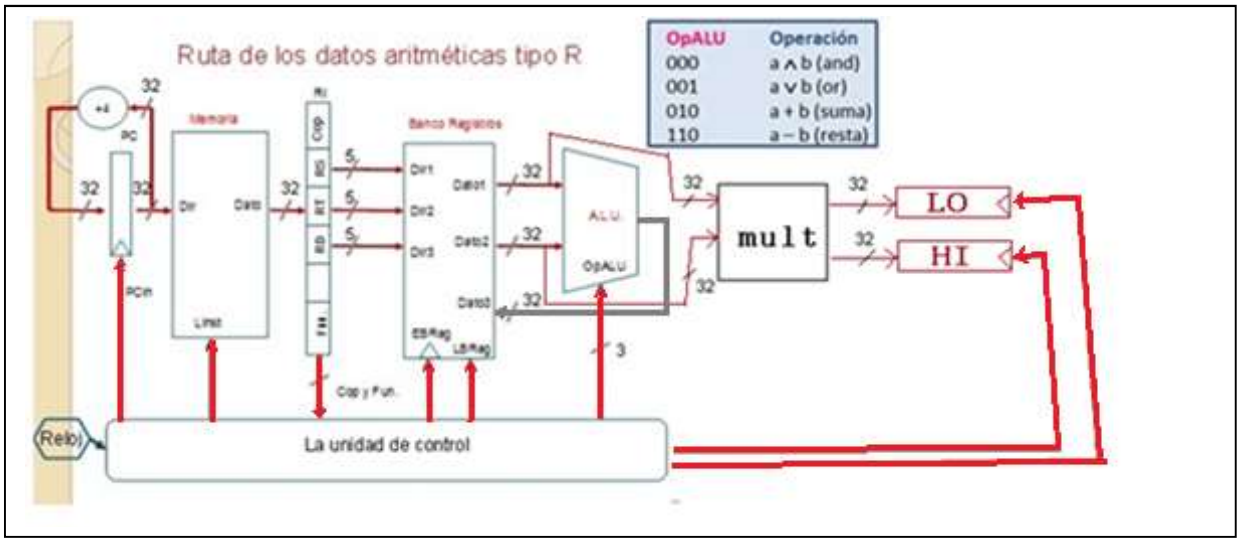
			PC	Memoria	Banco Registros		ALU
Instrucción	Código Operación	Función	PCin	LInst	EBReg	LBReg	OpALU
add $8, $10, $13	000000	100000	1	1	1	1	010

c) Supóngase los siguientes retardos en los elementos que componen esta ruta de datos: lectura/escritura en memoria 10 ns, lectura/escritura en banco de registros 5 ns, operación en la ALU 10 ns, el resto de retardos se considera despreciable. ¿Cuál sería la frecuencia de reloj máxima a la que puede trabajar este procesador? Justifique la respuesta.

$T_{ciclo} = T_{MI} + T_{BR(leer)} + T_{ALU} + T_{BR(escribir)} = 10+5+10+5 = 30$ ns

$F = 1/T_{ciclo} = 1/30 \times 10^{-9} = 1000/30$ MHz $= 33.3$ MHz

Ejercicio 3



Modifique la ruta de datos para ejecución de instrucciones tipo R para dar soporte a la instrucción

```
mult Rs, Rt    #  HI-LO = Rs × Rt
```

La instrucción es de formato R

COP	RS	RT	RD		Fun
000000	XXXXX	XXXXX	00000	00000	011000
31.....0					

1. Dibuje las modificaciones necesarias en la ruta de los datos e indique las características de los elementos funcionales necesarios para llevar a cabo su ejecución. Considere añadir elementos nuevos, no cambiar la ALU.

Habría que añadir una unidad multiplicadora, distinta de la ALU, y dos registros de 32 bits, el registro HI y el registro LO. La unidad de control tendría dos señales de salida nuevas, una para activar la escritura en registro LO y la otra para la escritura en HI.

2. Rellenar la tabla de verdad de la unidad de control para ejecutar las siguientes instrucciones: (añada las señales de control oportunas si fuera necesario)

			Banco Registros		ALU		
Instrucción	Código Operación	Función	EBReg	LBReg	OpALU	EHI	ELO
add $8, $10, $13	000000	100000	1	1	010	0	0
mult $8,$8	000000	011000	1	1	XXX	1	1

3. Si los tiempos de cálculo de cada unidad funcional fueran los que aparecen en la tabla siguiente, calcule la frecuencia máxima a la que podría funcionar esta ruta de los datos

antes de la modificación. (Observe que el banco de registros y la unidad de control pueden trabajar en paralelo)

Memoria	20ns
B.registros	15ns (ambas operaciones
ALU	15ns
U.Control	2ns
Resto de elementos	Tiempo despreciable

$T_{ciclo} = T_{MI} + T_{BR(leer)} + T_{ALU} + T_{BR(escribir)} = 20+15+15+15 = 65ns$

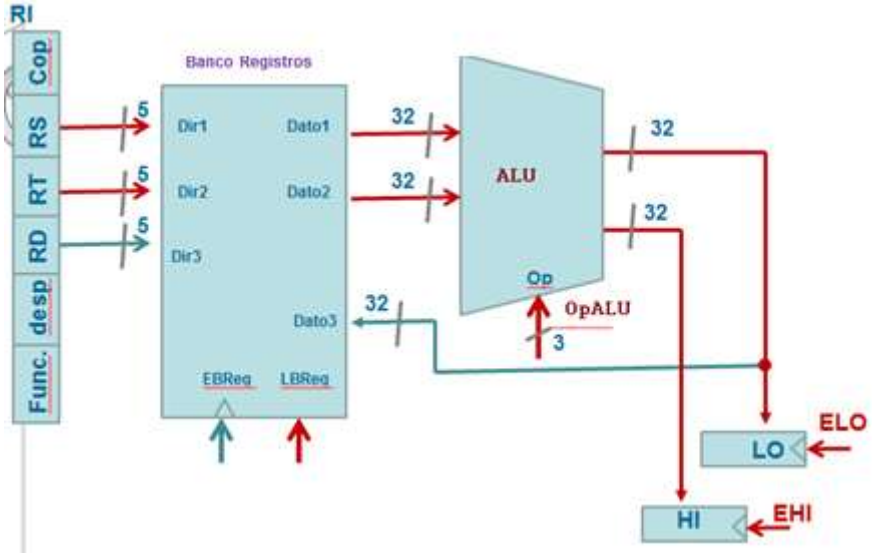
4. ¿Cuánto tiempo de cálculo como máximo podrá tener el elemento añadido para no modificar el tiempo de ciclo calculado en el apartado 2?

Tiempo MI	**Tiempo Leer BR**	**Tiempo ALU**	**Tiempo escribir BR**
|------20---------|	**|---------15--------|**	**|------15--------|**	**|---------15--------|**
		|------ Tiempo mult ----------|	

Como el circuito multiplicador es el que importa para la instrucción mult, y no haría falta almacenar en banco de registros ya que el resultado se queda en HI y LO, todo el tiempo disponible para la multiplicación sería 30ns. El nuevo circuito multiplicador podrá tardar hasta 30 ns si la escritura en los registros HI y LO fuera cero, y si fuera mayor habría que restárselo.

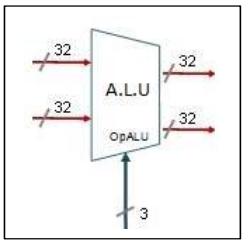
Ejercicio 4

Modifique la ruta de los datos para la ejecución de instrucciones aritméticas de tipo R add/sub/and/or/slt para incluir las operaciones mult y div.



Para ello se va a cambiar la ALU, a una que dispone de estas operaciones y su tabla de verdad es:

```
000    a ∧ b (and)
001    a ∨ b (or)
010    a + b (suma aritmética)
011     a *b (mult)
100     a/b  (div)
110    a - b (resta)
111    a < b (menor que)
```



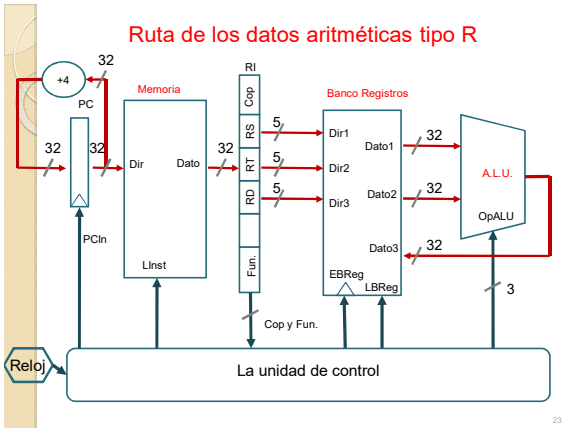
La salida de la ALU será en este caso de dos buses de 32 bits cada uno. Las cinco operaciones aritméticas anteriores aparecerían por las 32 patillas de menor peso de la ALU, mientras que para la mult/div se obtendría el resultado por ambos buses.

```
mult   rs, rt     #   HI-LO = rs * rt
div    rs, rt     #   HI = rs/rt y   LO = rs mod rt (resto)
```

Modifique sobre el dibujo anterior los elementos funcionales necesarios y complete la tabla de verdad de la unidad de control: (Añada las señales de control oportunas)

			PC	Memoria	Banco Registros		ALU		
Instrucción	Código Op.	Función	PCin	LInst	EBReg	LBReg	OpALU	ELO	EHI
mult rs, rt	000000	011000	1	1	0	1	011	1	1
div rs, rt	000000	011010	1	1	0	1	100	1	1
add rd,rs, rt	000000	100000	1	1	1	1	010	0	0

Ejercicio 5.1



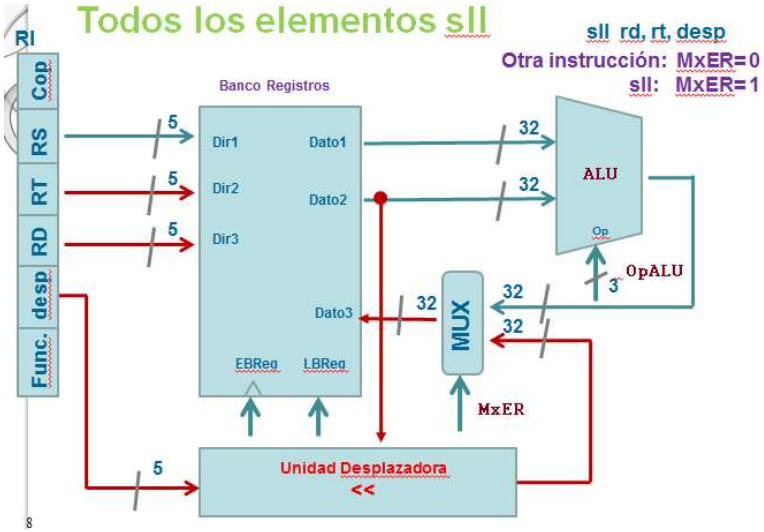
En la ruta de los datos anterior se desea incluir la instrucción:

```
sll rd,rt, desp5 #  rd ← rt << desp5, desplazamiento a
                       izquierdas. Se rellena con ceros.
```

COP	RS	RT	RD		Función
000000	00000	rt	rd	desp5	000000

Para ello se dispone de un operador combinacional llamado "Unidad desplazadora" que realiza la operación sll, para ello dispone de dos entradas, una de 32 bits, por donde se le introduce un dato a desplazar, y otra de 5 bits, desde la que se le indica la cantidad de bits a desplazar. La salida de dicho operador tendrá 32 bits.

a) Utilice este operador en la ruta de los datos anterior, marque sus conexiones y si fuera necesario añada otros elementos funcionales y señales de control adicionales en la unidad de control.

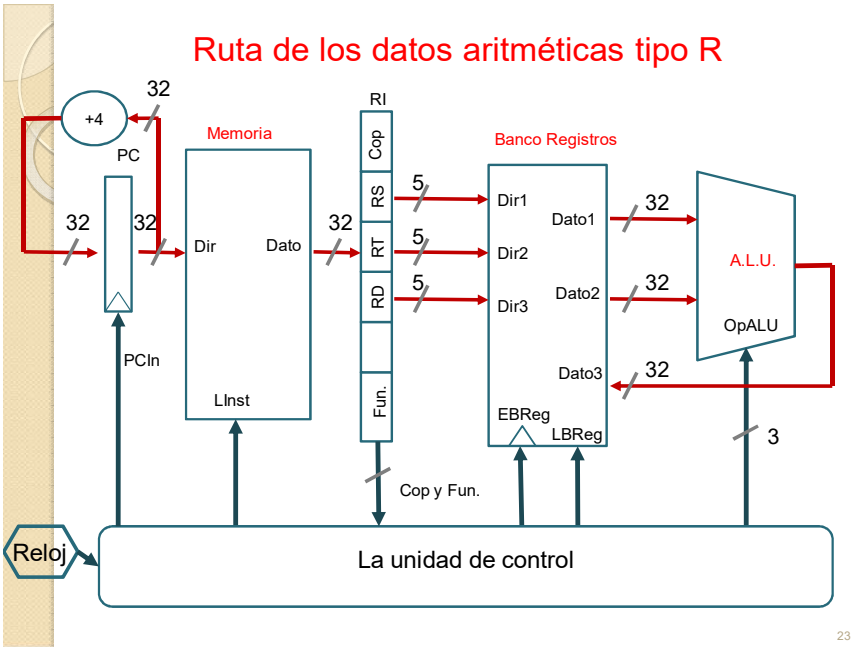


b) Rellene las señales de control de la siguiente tabla (la tabla de verdad de la ALU la misma del ejercicio 1) : (Añadir columnas si necesario)

NOTA: OpALU es xxx porque no se emplea la ALU con esa instrucción

				Banco Registros	ALU	
Instrucción	**Form**	**COP**	**Función**	**EBReg**	**OpALU**	**MxER**
add rd, rs, rt	R	000000	100000	1	010	0
sub rd, rs, rt	R	000000	100010	1	110	0
sll rt, rs,inm16	R	000000	000000	1	xxx	1

Ejercicio 5.2

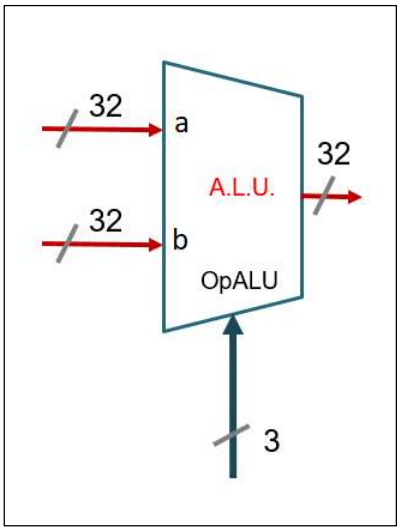


En la ruta de los datos anterior se desea incluir la instrucción:

```
sll rd,rt, desp5 #  rd ← rt << desp5, desplazamiento a
                        izquierdas. Se rellena con ceros.
```

COP	RS	RT	RD		Función
000000	00000	rt	rd	desp5	000000

Para ello se dispone de una nueva ALU que tiene la operación sll y funciona del siguiente modo:



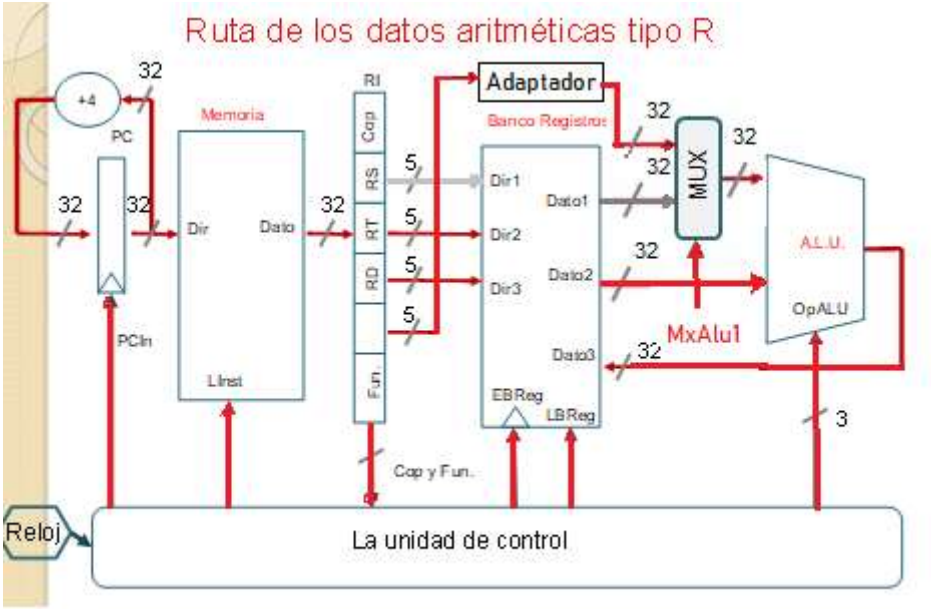
- Funcionamiento de la operación sll:

Por la primera entrada de datos, "a", toma los 5 bits de menor peso, y por la segunda entrada de datos, "b" toma los 32 bits entrantes. Por la salida se muestra el valor desplazado a izquierda tantos bits como se indique por la primera entrada de datos, introduciendo ceros por los bits menos significativos.

- Nueva tabla de verdad:

```
000     a ∧ b       (and)
001     a ∨ b       (or)
010     a + b       (suma aritmética)
011     b << a      (sll)
110     a − b       (resta)
111     a < b       (menor que)
```

a) Utilice este operador en la ruta de los datos anterior, marque sus conexiones y si fuera necesario añada otros elementos funcionales y señales de control adicionales en la unidad de control.

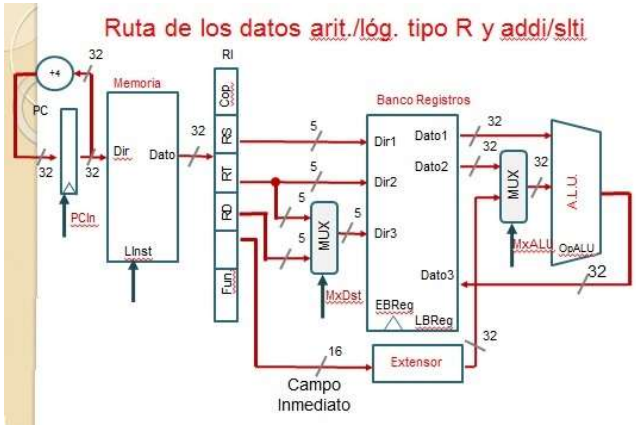


El Mux introducido está gobernado por la señal MxAlu1, y selecciona a 0 la entrada superior, y a 1 la inferior

b) Rellene las señales de control de la siguiente tabla: (Añadir columnas si necesario)

				Banco Registros	ALU	
Instrucción	**Form**	**COP**	**Función**	**EBReg**	**OpALU**	**MxAlu1**
add rd, rs, rt	R	000000	100000	1	010	1
sub rd, rs, rt	R	000000	100010	1	110	1
sll rt, rs,inm16	R	000000	000000	1	011	0

Ejercicio 6.1



NOTA: En los multiplexores, MUX, la entrada de arriba es la 0 y la de abajo es la 1.

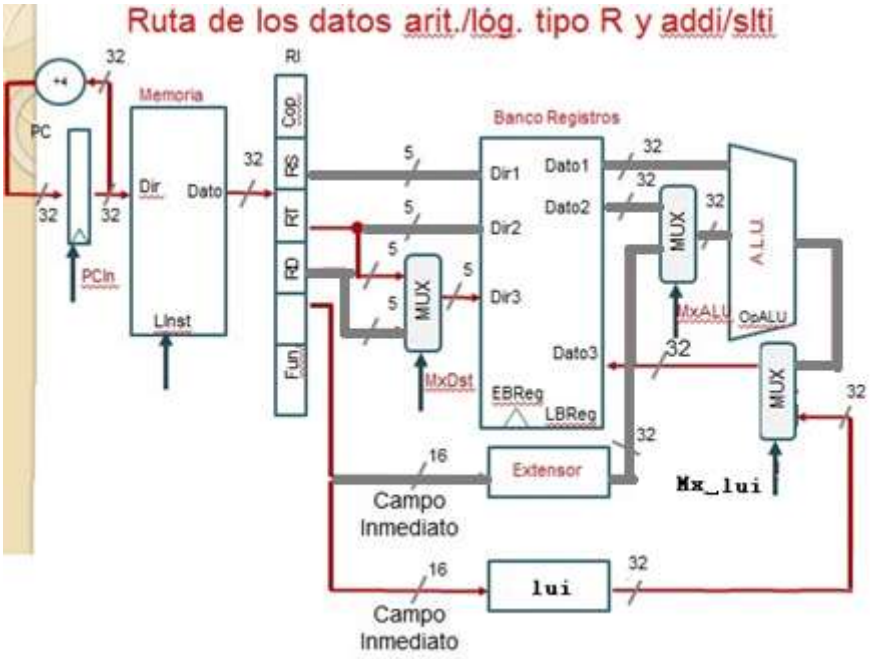
En la ruta de los datos anterior se desea incluir la instrucción de tipo I:

```
lui rt, inm16     # rt31..16 = inm16; rt15..0 = 0
```

COP	RS	RT	Inmediato 16
001111	00000	Rt	xxxxxxxxxxxxxxxx
31			0

Para ello suponga que se dispone de un operador combinacional llamado "lui" que recibe una entrada de 16 bits y proporciona una salida de 32 bits, dejando el dato entrante en los 16 bits de mayor peso y los 16 bits de menor peso a cero.

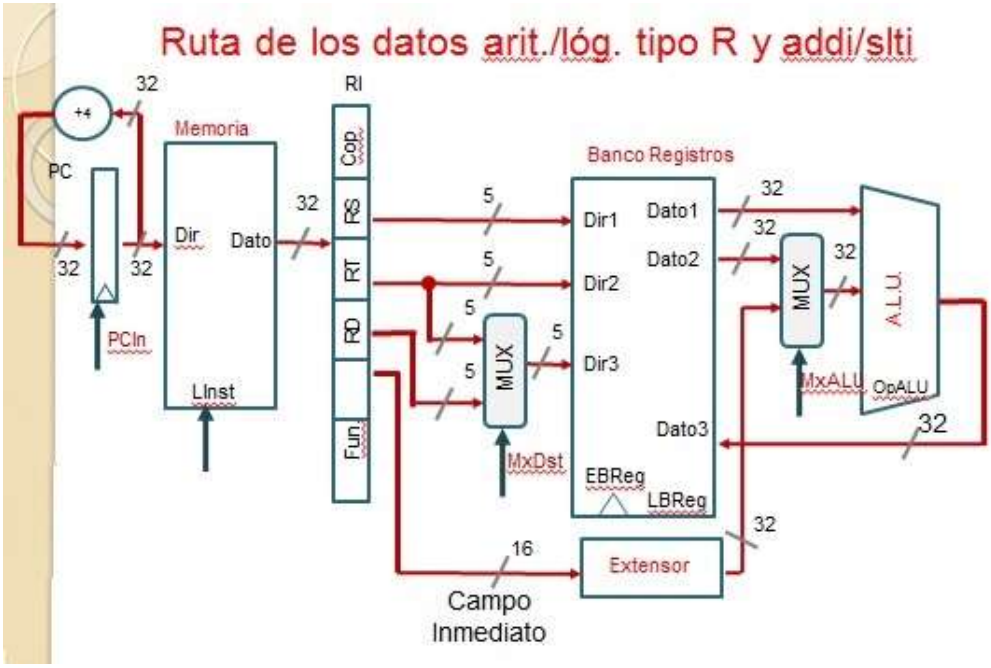
a) Incluya en la ruta de los datos anterior este operador y los elementos funcionales y señales de control necesarios, de modo que se pueda ejecutar en la misma la operación lui anteriormente descrita.



b) Rellene las señales de control de la siguiente tabla (la tabla de verdad de la ALU es la misma del ejercicio 1, y si hace falta añadir columnas)

Instrucción	Form	COP	Función	EBReg	OpALU	MxALU	MxDst	Mx_lui
add rd, rs, rt	R	000000	100000	1	010	0	1	0
lui rt, inm16	I	001111		1	xxx	x	0	1
addi rt, rs,inm16	I	001000		1	010	1	0	0

Ejercicio 6.2



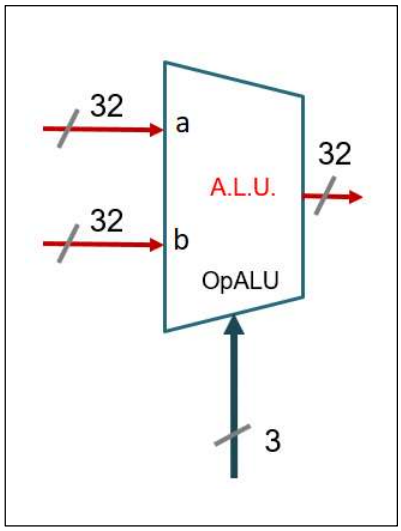
NOTA: En los multiplexores, MUX, la entrada de arriba es la 0 y la de abajo es la 1.

En la ruta de los datos anterior se desea incluir la instrucción de tipo I:

```
lui rt, inm16     # rt31..16 = inm16; rt15..0 = 0
```

COP	RS	RT	Inmediato 16
001111	00000	Rt	xxxxxxxxxxxxxxxx
31			0

Para ello se dispone de una nueva ALU que tiene la operación lui y funciona del siguiente modo:



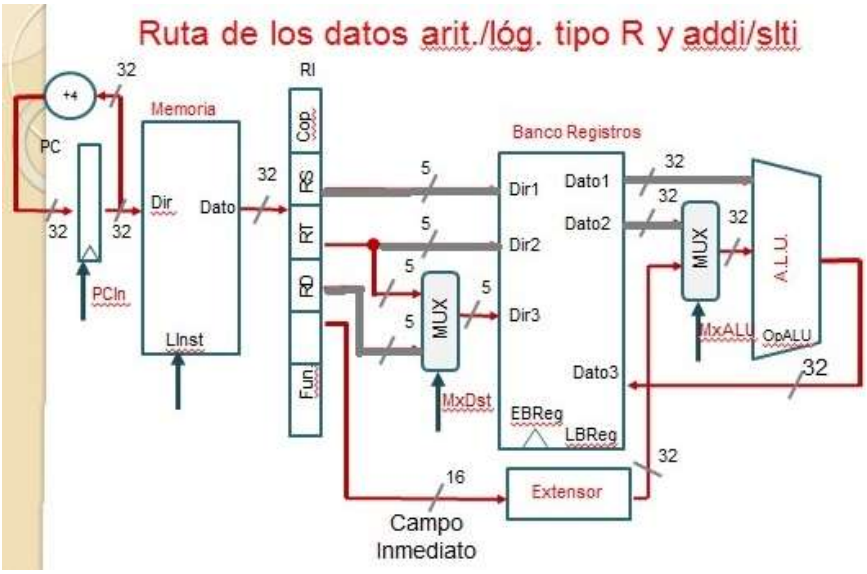
- Funcionamiento de la operación lui:

Por la segunda entrada de datos, "b", toma los 16 bits de menor peso y por la salida se muestra el valor entrante en los 16 bits de mayor peso y ceros por los 16 bits menos significativos.

- Nueva tabla de verdad:

```
000     a ∧ b       (and)
001     a ∨ b       (or)
010     a + b       (suma aritmética)
011     b || 0      (lui)
110     a − b       (resta)
111     a < b       (menor que)
```

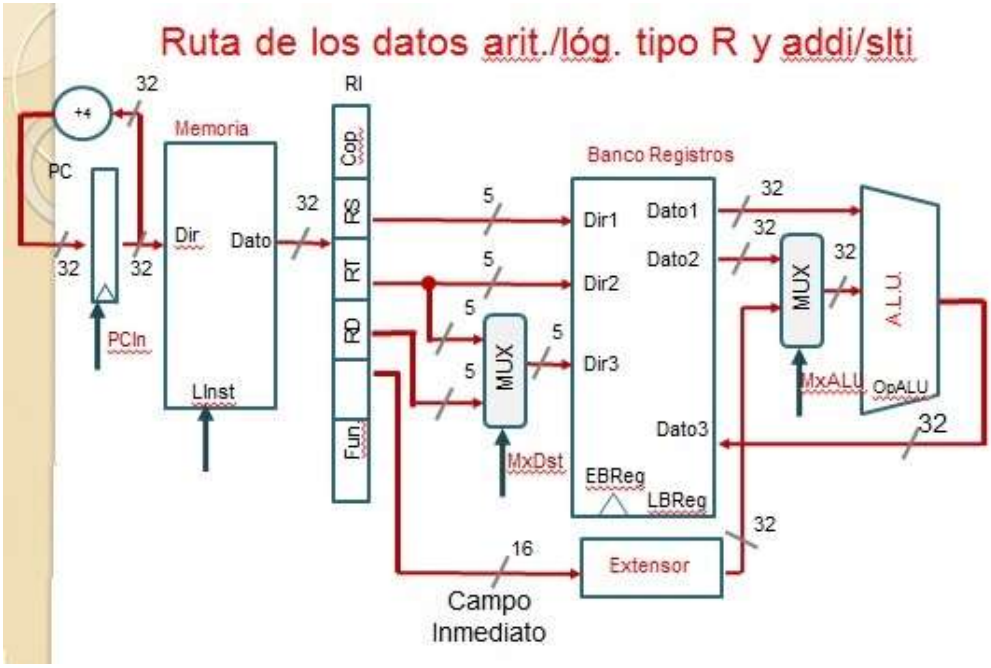
a) Incluya en la ruta de los datos anterior esta ALU y los elementos funcionales y señales de control necesarios, de modo que se pueda ejecutar en la misma la operación lui anteriormente descrita.



b) Rellene las señales de control de la siguiente tabla y si hace falta añadir columnas:

Instrucción	Form	COP	Función	EBReg	OpALU	MxALU	MxDst	
add rd, rs, rt	R	000000	100000	1	010	0	1	
lui rt, inm16	I	001111		1	011	1	0	
addi rt, rs,inm16	I	001000		1	010	1	0	

Ejercicio 6.3



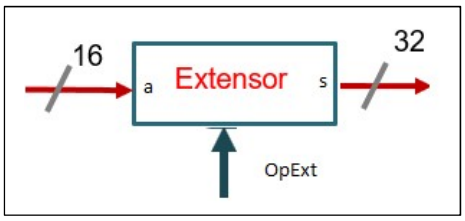
NOTA: En los multiplexores, MUX, la entrada de arriba es la 0 y la de abajo es la 1.

En la ruta de los datos anterior se desea incluir la instrucción de tipo I:

```
lui rt, inm16     # rt31..16 = inm16; rt15..0 = 0
```

COP	RS	RT	Inmediato 16
001111	00000	Rt	xxxxxxxxxxxxxxxx
31			0

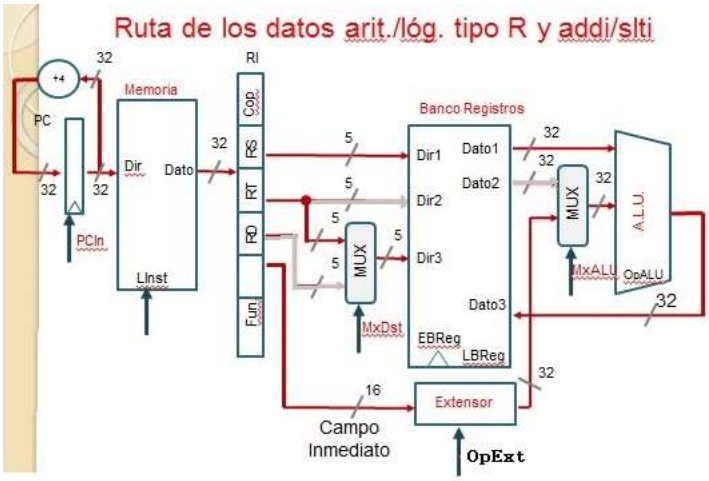
Para ello suponga que se dispone de un nuevo operador combinacional llamado "Extensor" que recibe una entrada de 16 bits y proporciona una salida de 32 bits en función del valor de la señal de entrada OpExt.
Se muestra la tabla de verdad del nuevo operador:



Entrada (OpExt)	Salida (s)
0	Signo || a (extensión signo)
1	a || 0 (operación lui)

Observe que el nuevo operador es un circuito multifunción, dependiendo de la entrada OpExt será la salida de 32 bits que se introducirá en la ALU.

a) Se propone cambiar el circuito "Extensor" actual por el nuevo operador gobernado por OpExt, sin hacer ningún otro cambio en la ruta de los datos. Piense de qué forma se podría emplear la ALU para dejar pasar el dato que le llega por la entrada inferior, teniendo dicha ALU las mismas operaciones que se describieron en el ejericio 1.
NOTA: fíjese en el campo RS de esta instrucción.



b) Rellene las señales de control de la siguiente tabla (la tabla de verdad de la ALU es la misma del ejercicio 1, y si hace falta añadir columnas)

Instrucción	Form	COP	Función	EBReg	OpALU	MxALU	MxDst	OpExt
add rd, rs, rt	R	000000	100000	1	010	0	1	X
lui rt, inm16	I	001111		1	010	1	0	1
addi rt, rs,inm16	I	001000		1	010	1	0	0

c) Se propone como ejercicio adicional pensar cómo modificar este nuevo circuito Extensor para que se puedan ejecutar en esta misma ruta de los datos las instrucciones de tipo I:

- addi/slti (ya descritas en clase)
- lui (ya descrita)
- ori/andi

```
andi rt,rs, inm16     # rt = rs and inm16(*)
```

COP	RS	RT	Inmediato 16
001100	Rs	Rt	xxxxxxxxxxxxxxxx

31.....0

```
ori rt, rs, inm16     # rt= rs or inm16(*)
```

COP	RS	RT	Inmediato 16
001101	Rs	Rt	xxxxxxxxxxxxxxxx

31.....0

inm16(*): En este caso no se extiende el signo a los 16 bits de más peso, se dejan a cero. Es distinto a las instrucciones addi/slti.

Se añadiría al circuito extensor una nueva función, que sería la extensión de ceros a la parte alta del dato de salida. En este caso se necesitarían dos bits para seleccionar una u otra opción de entre tres:

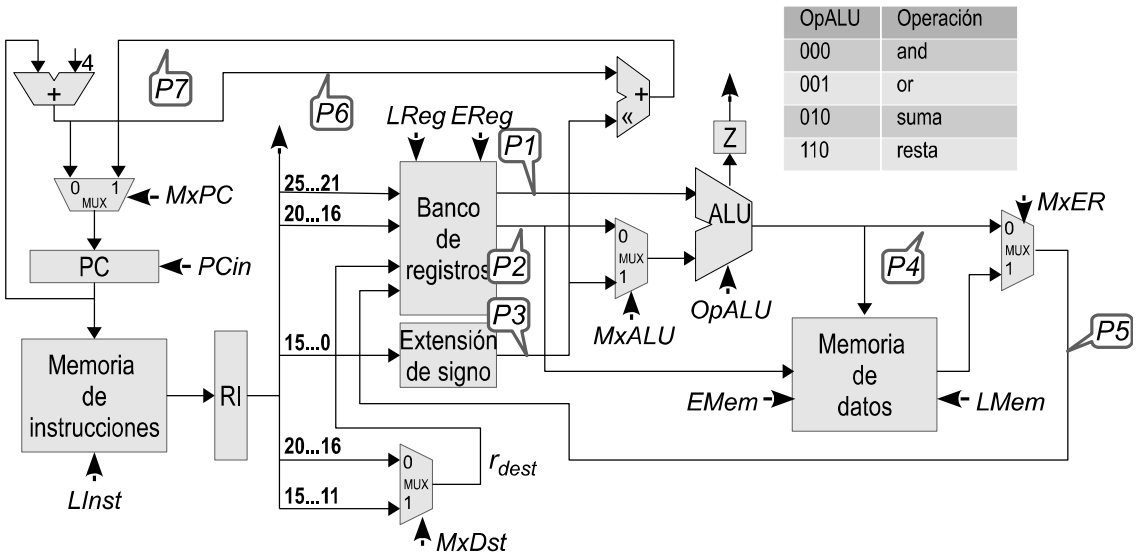
Entrada (OpExt)	Salida (s)
00	Signo || a (extensión signo)
01	0 || a (extensión ceros)
10	a || 0 (operación lui)

Se rellena a continuación cómo se activarían las señales en la ruta de los datos (se ha empleado la tabla de verdad del ejercicio 1 para la ALU:

Instrucción	Form	COP	EBReg	OpALU	MxALU	MxDst	OpExt
add rd, rs, rt	R	000000	1	010	0	1	XX
lui rt, inm16	I	001111	1	010	1	0	10
addi rt, rs,inm16	I	001000	1	010	1	0	00
andi rt, rs,inm16	I	001100	1	000	1	0	01
slti rt, rs, inm16	I	001110	1	111	1	0	00
ori rt, rs,inm16	I	001101	1	001	1	0	01

Ejercicio 7

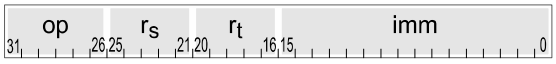
Considere la ruta de datos monociclo:



OpALU	Operación
000	and
001	or
010	suma
110	resta

a) La siguiente tabla muestra diversos estados de la ruta de datos definidos por los campos del registro RI y las señales de control que se aplican. Deduzca, en cada caso, a qué instrucciones del juego del MIPS se refieren. Todas son del formato I.

Formato I:



RI			Señales							
25...21	20...16	15..0	MxALU	OpALU	LMem	EMem	Ereg	MxDst	MxER	MxPC	Instrucción
2	4	8	1	010	0	0	1	0	0	0	`addi $4,$2,8`
2	4	8	1	010	0	1	0	0	0	0	`sw $4,8($2)`
2	4	8	1	010	1	0	1	0	1	0	`lw $4,8($2)`
2	4	8	0	110	0	0	0	0	0	0	`beq $2,$4,8`

b) Suponiendo que el contenido de algunos registros y de la memoria del MIPS es el mostrado a la derecha, indique (en decimal) el valor presente en los puntos rotulados en la ruta de dados como P1 a P7. Considere en todos los casos que la instrucción se encuentra en la dirección 1000 (decimal), o sea, que el PC contiene la dirección 1000. Si en algún caso falta información, indíquelo con "???"

Registros

Reg	Contenido
$2	200
$3	300
$4	400
$5	500

Memoria principal

Dir	Contenido
220	17

Instrucción	P1	P2	P3	P4	P5	P6	P7
`sw $3,20($2)`	*200*	*300*	*20*	*220*	*¿?*	*1004*	*1084*
`lw $3,20($2)`	*200*	*300*	*20*	*220*	*17*	*1004*	*1084*
`beq $2,$3,2004`	*200*	*300*	*2004*	*-100*	*¿?*	*1004*	*9020*

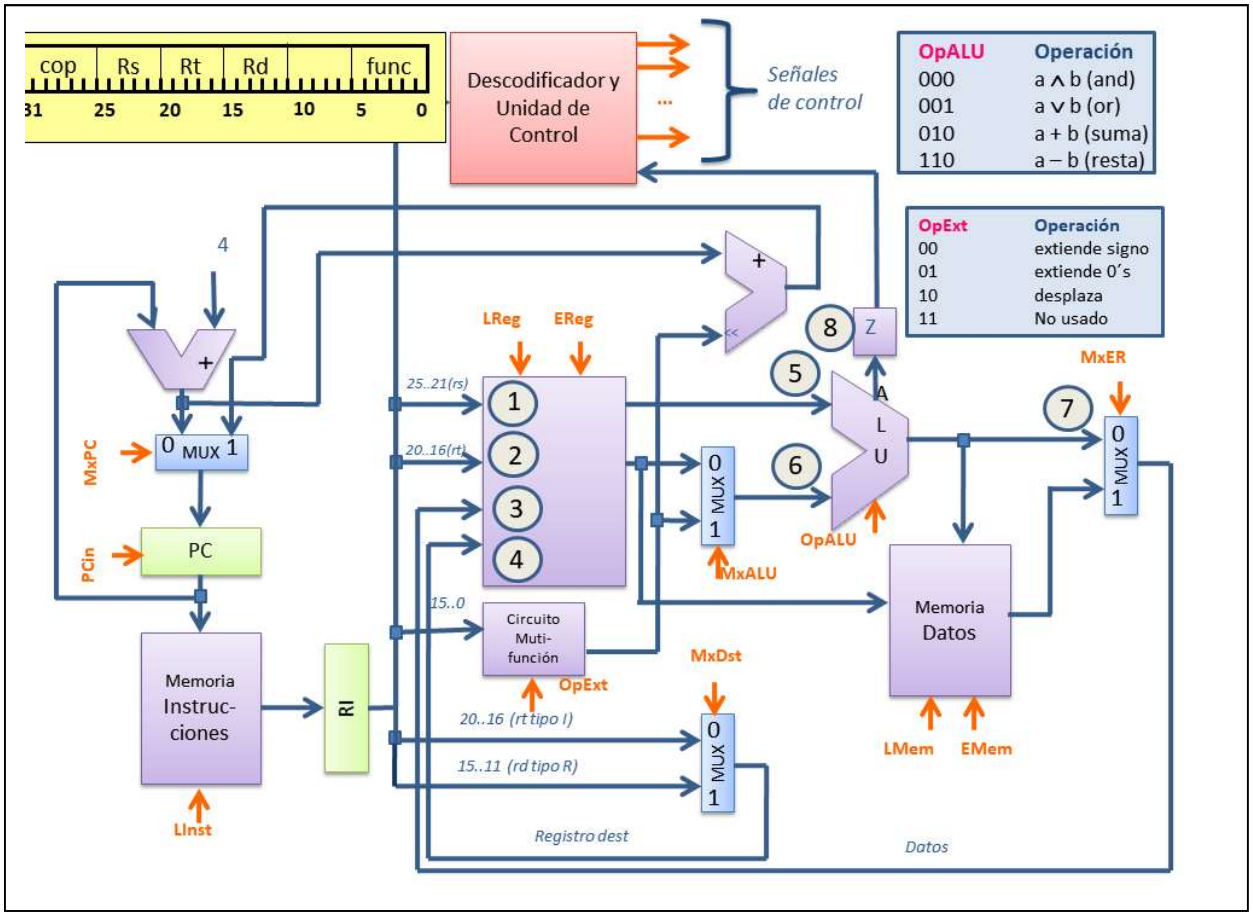
Ejercicio 8

Considere el siguiente fragmento de código en ensamblador del MIPS R2000:

```
or $8, $0, 2
or $9, $0, 25
or $8, $8, $9
```

a) Indique el valor de las líneas marcadas con un círculo en la ruta de datos de la figura 2 en el momento de la ejecución de la instrucción ***or***. Exprese los valores en decimal. El formato de la instrucción add se muestra a junto con la ruta de datos:

Punto	1	2	3	4	5	6	7	8
Valor	8	9	27	8	2	25	27	0

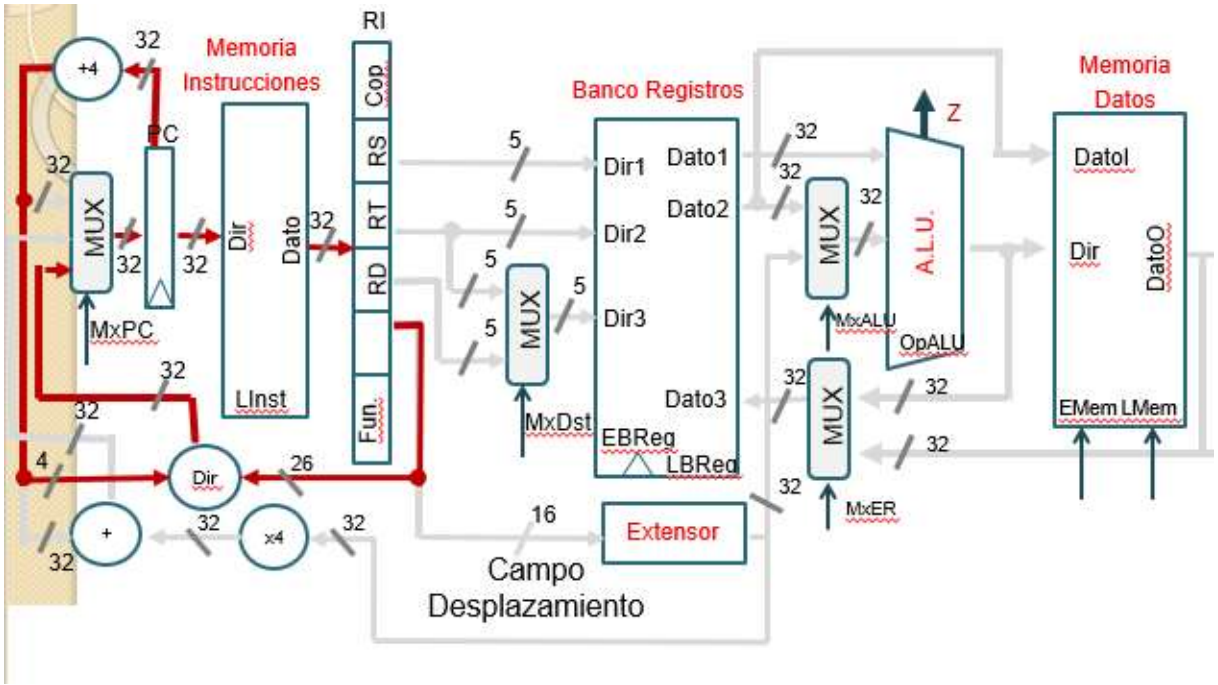


b) Supóngase los siguientes retardos en los elementos que componen esta ruta de datos: lectura/escritura en memoria 14 ns, lectura/escritura en banco de registros 5 ns, operación en la ALU 8 ns, circuito multifunción 6 ns, el resto de retardos se considera despreciable. ¿Cuál sería la frecuencia de reloj máxima a la que puede trabajar este procesador? Justifique la respuesta.

$Tc_{ciclo} = 14_{MI} + \max\{5_{BR}, 6_{multifuncion}\} + 8_{ALU} + 14_{MD} + 5_{BR} = 47$ ns

$F_{max} = 1/Tc_{ciclo} = 1000 / 47$ MHz $= 21.3$ MHz

Ejercicio 9



NOTA: En los multiplexores, MUX, la entrada de arriba es la 0 y la de abajo es la 1.

Dada la ruta de los datos anterior. Marque en el dibujo el camino que seguiría la instrucción

```
j dirección      #     PC ← PC31..28 || Destino
```

COP	Destino (26 bits menos peso de la dirección de salto)
000010	

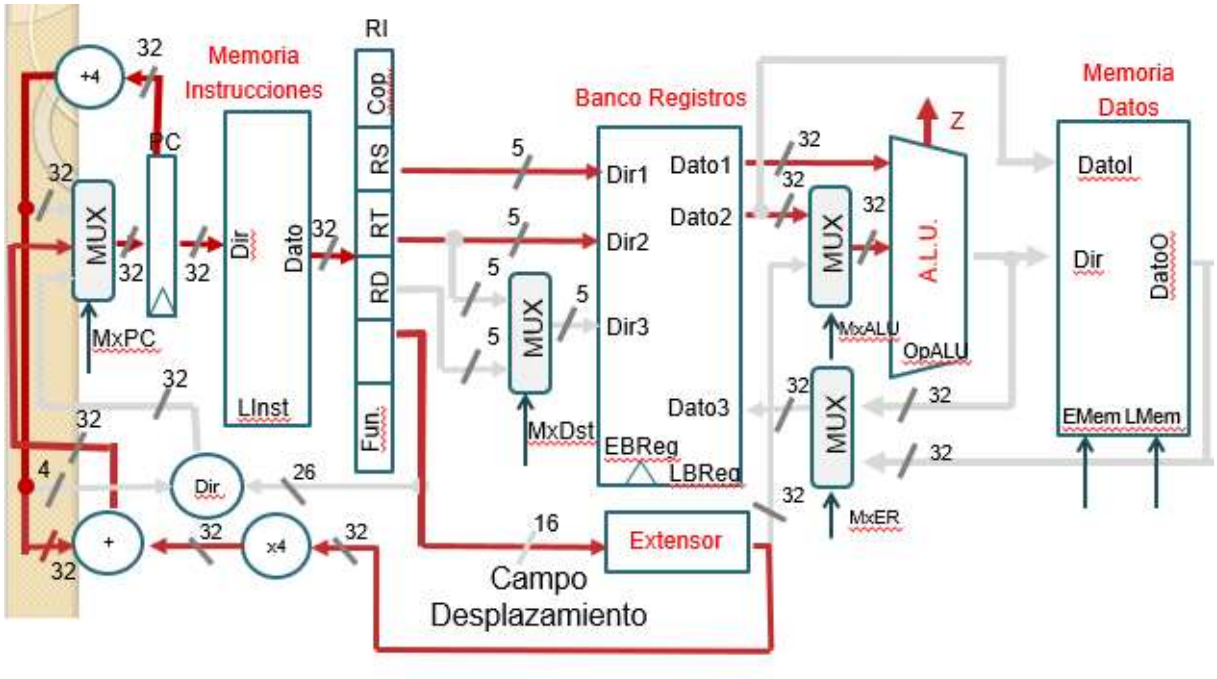
Y rellene las señales de control de la siguiente tabla

				Banco Registros	ALU	Mem. Datos		Multiplexores Configuración Ruta de Datos			
Instrucción	**Form**	**COP**	**Función**	**EReg**	**OpALU**	**LMem**	**EMem**	**MxPC**	**MxALU**	**MxDst**	**MxER**
j etiqueta	J	000010		0	xxx	0	0	10	x	x	x

¿Importa el valor del bit Z para la ejecución de esta instrucción? NO

¿Se emplea la ALU para la ejecución de esta instrucción? NO, por eso ponemos a X los valores de los multiplexores, no importa lo que valga esa entrada.

Ejercicio 10



NOTA: En los multiplexores, MUX, la entrada de arriba es la 0 y la de abajo es la 1. En el que hay tres, la 2 es la de abajo.

Dada la ruta de los datos anterior. Marque en el dibujo el camino que seguiría la instrucción

```
beq Rs,Rt,dirección # Si Rs< Rt PC= Inmediato16*4+PC Sino PC = Pc+4
```

COP	RS	RT	Inmediato 16
000100	xxxxx	xxxxx	xxxxxxxxxxxxxxxx

Y rellene las señales de control de la siguiente tabla

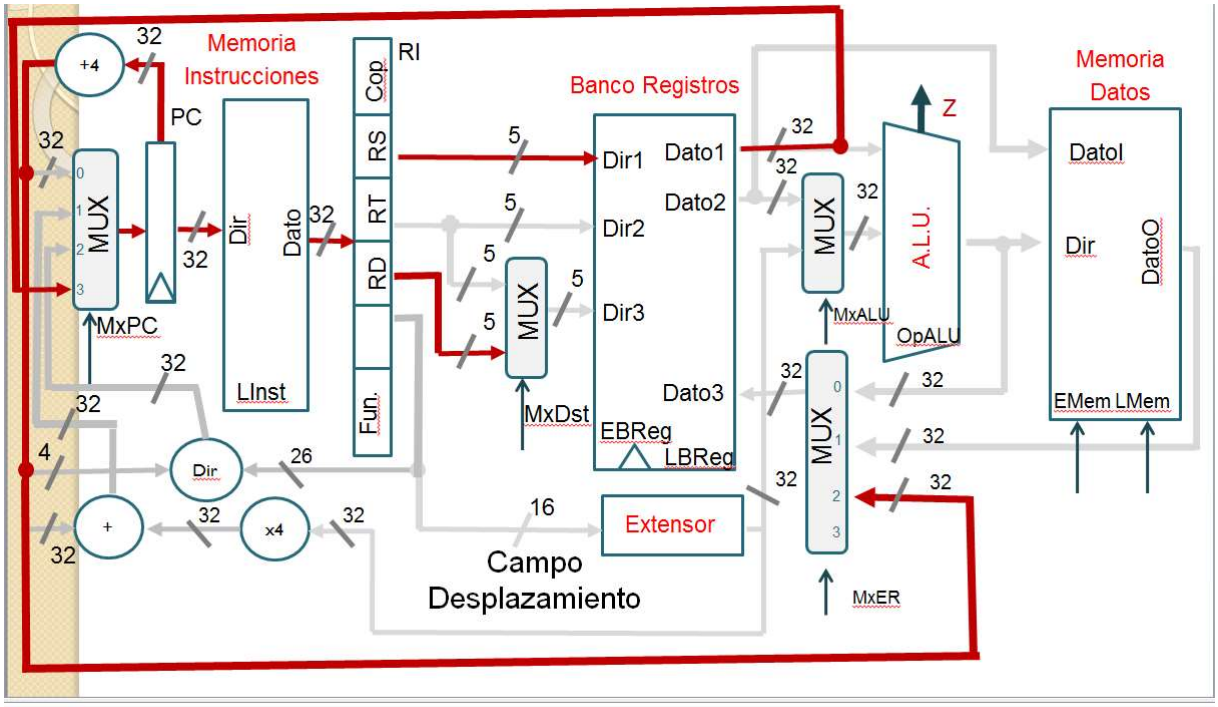
				Banco Registros	ALU	Mem. Datos		Multiplexores Configuración Ruta de Datos			
Instrucción	**Form**	**COP**	**Función**	**EReg**	**OpALU**	**LMem**	**EMem**	**MxPC**	**MxALU**	**MxDst**	**MxER**
j etiqueta	J	000010		0	xxx	0	0	10	x	x	x
beq rs,rt,etiqueta	R	000100		0	110	0	0	0Z	0	x	x

Nota: Emplear la misma tabla de verdad para la ALU que en el ejercicio 1.

¿Importa el valor del bit Z para la ejecución de esta instrucción? SI. Según su valor, se saltará o no se saltará. En caso de valer Z = 0, se elegirá PC = PC+4, mientras que si vale Z = 1, se elegirá PC = ruta del salto

¿Se emplea la ALU para la ejecución de esta instrucción? Si. Se resta el contenido de los registros Rs y Rt y así se sabe si son iguales. Si son iguales Z = 1, mientras que si son distintos, Z = 0.

Ejercicio 11



egar notas

NOTA: En los multiplexores, MUX, la entrada de arriba es la 0 y la de abajo es la 1. En el que hay tres, la 2 es la de abajo.

Se han introducido los elementos necesarios para ejecutar la jalr, y además se ha marcado la ruta de los datos que seguirá la instrucción jalr cuando se ejecute.

Incluya en la ruta anterior los elementos necesarios para ejecutar la instrucción:

```
jalr $rd, $rs      #  $rd ← PC+4, PC ← $rs
```

COP	RS	RT	RD		Función
000000	rs	00000	rd	00000	001001

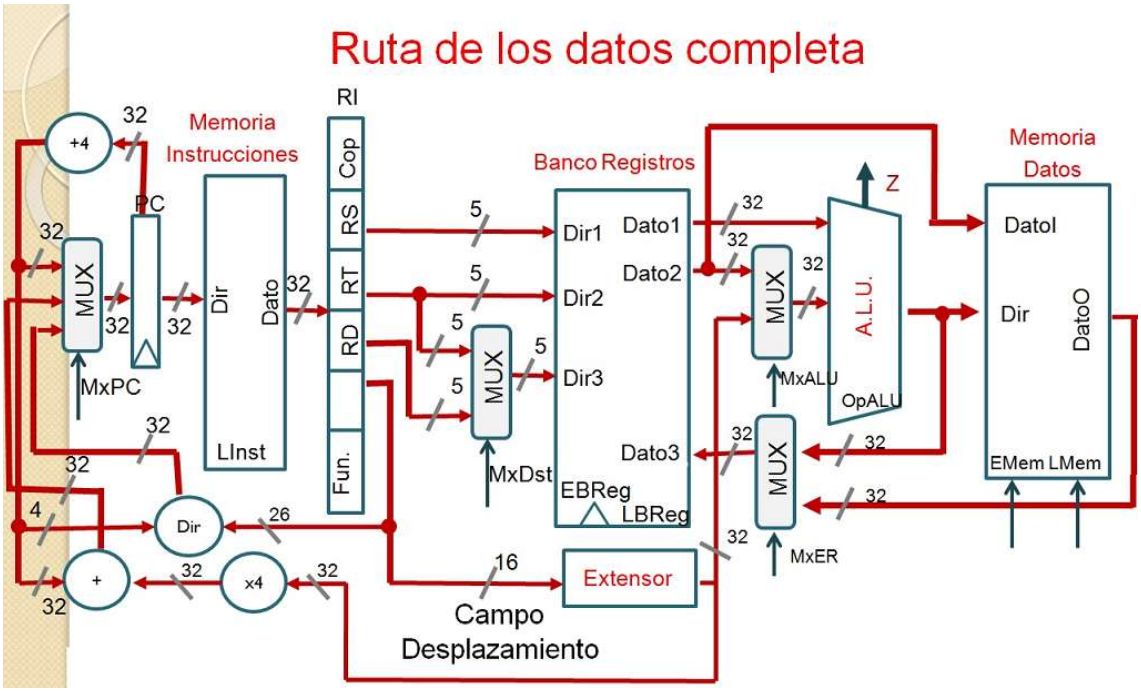
Y rellene las señales de control de la siguiente tabla:

				Banco Registros	ALU	Mem. Datos		Multiplexores Configuración Ruta de Datos			
Instrucción	**Form**	**COP**	**Función**	**EReg**	**OpALU**	**LMem**	**EMem**	**MxPC**	**MxALU**	**MxDst**	**MxER**
j etiqueta	J	000010		0	xxx	0	0	10	x	x	x
beq rs,rt, etiq	I	100011		0	110	0	0	0Z	0	x	x
jalr $rd,$rs	R	000000	001001	1	xxx	0	0	11	x	1	10

Nota: Emplear la misma tabla de verdad para la ALU que en el ejercicio 1.

OJO: para comparar en la beq se emplea la resta de los registros y el valor del bit Z es el que decide si se saltará o no (No se salta 00, si se salta 01)

Ejercicio 12



NOTA: En los multiplexores, MUX, la entrada de arriba es la 0 y la de abajo es la 1. En el que hay tres, la 2 es la de abajo.

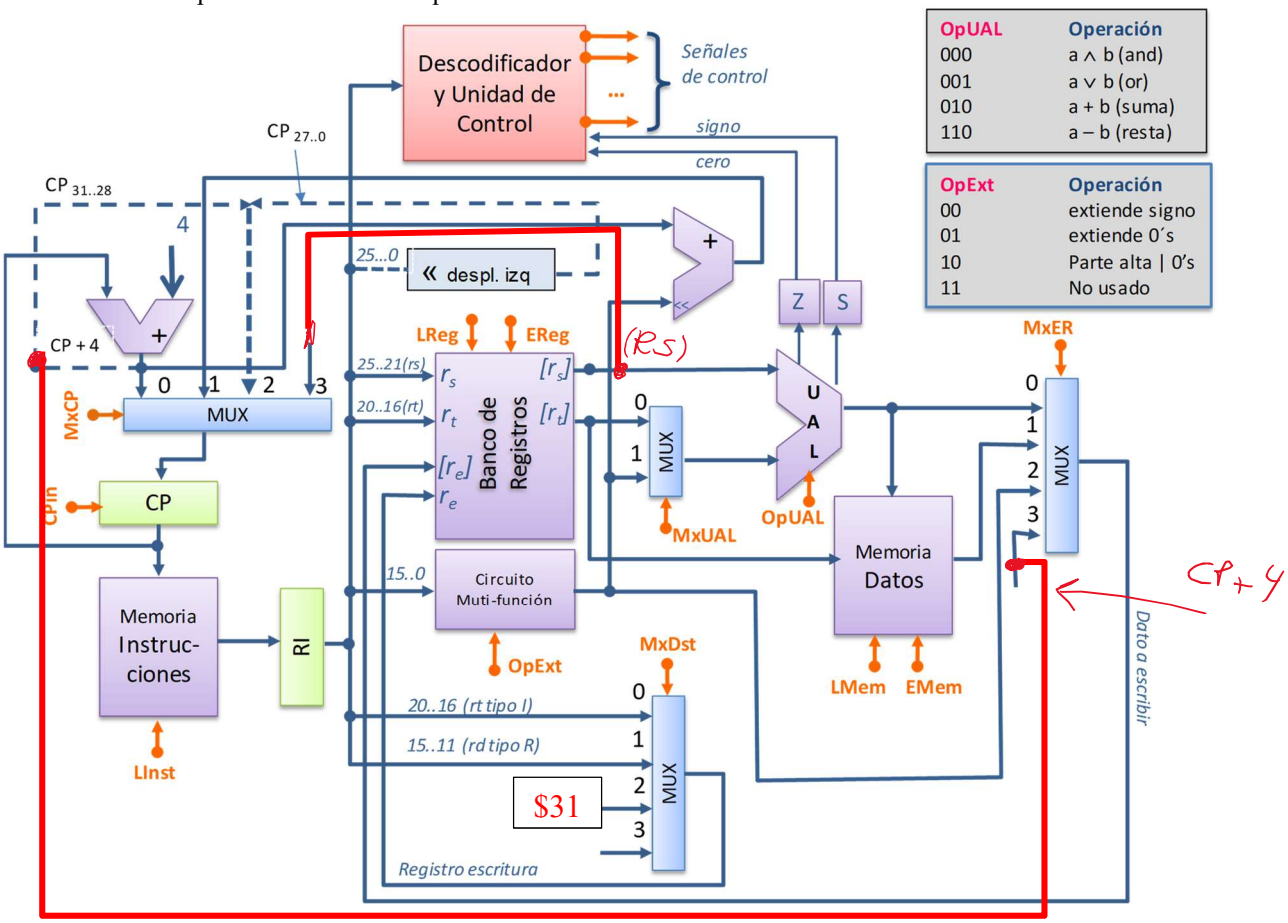
Rellene las señales de control de la siguiente tabla:

				Banco Registros	ALU	Mem. Datos		Multiplexores Configuración Ruta de Datos			
Instrucción	**Form**	**COP**	**Función**	**EReg**	**OpALU**	**LMem**	**EMem**	**MxPC**	**MxALU**	**MxDst**	**MxER**
slt rd, rs, rt	R	000000	101010	1	111	0	0	00	0	1	0
slti rt, rs,inm16	I	001110		1	010	0	0	00	1	0	0
j etiqueta	J	000010		0	xxx	0	0	10	x	x	x
beq rs,rt,etiqueta	R	000100		0	110	0	0	0Z	0	x	x
sw rd,desp($rs)	I	101011		0	010	0	1	00	1	x	x
lw rd,desp($rs)	I	100011		1	010	1	0	00	1	0	1
bne rs,rt,etiqueta	R	000101		0	110	0	0	0/Z	0	x	x
sub rd, rs, rt	R	000000	100010	0	110	0	0	00	0	1	0

Nota: Emplear la misma tabla de verdad para la ALU que en el ejercicio 1.

Ejercicio 13

La figura adjunta muestra una ruta de datos ampliada para el procesador monociclo visto en clase. Complete la tabla de las señales de control para la ejecución de las instrucciones indicadas. Si es preciso, añada a la figura las conexiones o componentes adicionales que considere necesarios.



				Banco Registros		ALU	Circuito multifunción	Memoria Datos		Multiplexores			
Instrucción	**Form**	**Co. Op**	**Función**	**LReg**	**Ereg**	**OpALU**	**OpExt**	**LMem**	**EMem**	**MxPC**	**MxALU**	**MxDst**	**MxER**
add rd, rs, rt	R	000000	100000	1	1	010	xx	0	0	00	0	01	00
sub rd, rs, rt	R	000000	100010	1	1	110	xx	0	0	00	0	01	00
and rd, rs, rt	R	000000	100100	1	1	000	XX	0	0	00	0	01	00
or rd, rs, rt	R	000000	100101	1	1	001	XX	0	0	00	0	01	00
addi rt, rs, inm_{16}	I	001000	xxxxxx	1	1	010	00	0	0	00	1	00	00
ori rt, rs, inm_{16}	I	001101	xxxxxx	1	1	001	01	0	0	00	1	00	00
andi rt, rs, inm_{16}	I	001100	xxxxxx	1	1	000	01	0	0	00	1	00	00
lui rt, inm_{16}	I	001111	xxxxxx	1	1	XXX	10	0	0	00	X	00	10
lw rt, desp(rs)	I	100011	xxxxxx	1	1	010	00	1	0	00	1	00	01
sw rt, desp(rs)	I	101011	xxxxxx	1	0	010	00	0	1	00	1	XX	XX
beq rs, rt, etiq	I	000100	xxxxxx	1	0	110	XX	0	0	0Z	0	XX	XX
bne rs, rt, etiq	I	000101	xxxxxx	1	0	110	XX	0	0	0Z*	0	XX	XX
j etiqueta	J	000010	xxxxxx	X	0	XXX	XX	0	0	10	X	XX	XX
jal etiqueta	J	000011	xxxxxx	X	1	XXX	XX	0	0	10	X	10	11
jr rs	R	000000	000001	1	0	XXX	XX	0	0	11	X	XX	XX
jalr rd, rs	R	000000	001001	1	1	XXX	XX	0	0	11	X	01	11