ESTRUCTURA DE COMPUTADORES

Ejercicios Tema 1: El procesador

1. Modifica la unidad de control vista en clase para dar soporte a la siguiente instrucción:

```
addi rt, rs, inm    # rt = rs + inm
```

Para realizar el ejercicio, completa la tabla de verdad de la unidad de control:

La ruta de datos vista en clase permite ejecutar esta instrucción. Solo hay que modificar el control tal y como indica la tabla siguiente:

				Banco Registros	**ALU**	**Mem. Datos**		**Multiplexores Configuración Ruta de Datos**			
Instrucción	Form	Código Op.	Función	EReg	OpALU	LMem	EMem	MxPC	MxALU	MxDst	MxER
add rd, rs, rt	R	000000	100000	1	010	0	0	0	0	1	0
sub rd, rs, rt	R	000000	100010	1	110	0	0	0	0	1	0
and rd, rs, rt	R	000000	100100	1	000	0	0	0	0	1	0
or rd, rs, rt	R	000000	100101	1	001	0	0	0	0	1	0
lw rt, desp(rs)	I	100011		1	010	1	0	0	1	0	1
sw rt, desp(rs)	I	101011		0	010	0	1	0	1	X	X
beq rs, rs, etiq	I	000100		0	110	0	0	Z	0	X	X
addi rt, rs, inm	I	001000		1	010	0	0	0	1	0	0

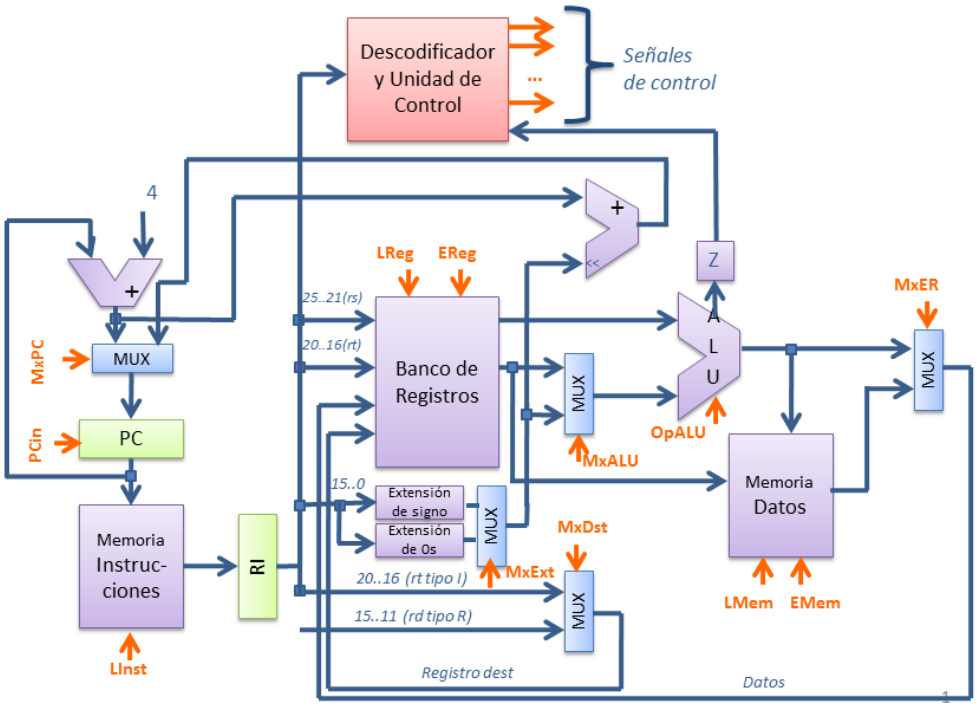
2. Modifica la ruta de datos y unidad de control vista en clase para dar soporte a la siguiente instrucción:

ori rt, rs, inm # rt = rs v inm (extendiendo a 0s)

En este caso no es posible ejecutar la instrucción en la ruta de datos vista en clase. Es necesario modificarla añadiendo un nuevo circuito que permita extender a 0 los 16 MSB del operando inmediato.

La figura presenta una posible solución:

Ruta de datos incluyendo ORI



Será necesario además añadir una nueva señal de control que actúe sobre el nuevo Multiplexor para seleccionar el dato inmediato (MxExt)

				Banco Registros	ALU	Mem. Datos		Multiplexores Configuración Ruta de Datos				
Instrucción	Form	Código Op.	Función	EReg	OpALU	LMem	EMem	MxPC	MxALU	MxDst	MxER	MxExt
add rd, rs, rt	R	000000	100000	1	010	0	0	0	0	1	0	0
sub rd, rs, rt	R	000000	100010	1	110	0	0	0	0	1	0	0
and rd, rs, rt	R	000000	100100	1	000	0	0	0	0	1	0	0
or rd, rs, rt	R	000000	100101	1	001	0	0	0	0	1	0	0
lw rt, desp(rs)	I	100011		1	010	1	0	0	1	0	1	0
sw rt, desp(rs)	I	101011		0	010	0	1	0	1	X	X	0
beq rs, rs, etiq	I	000100		0	110	0	0	Z	0	X	X	0
addi rt, rs, inm	I	001000		1	010	0	0	0	1	0	0	0
ori rt, rs, inm	I	001101		1	001	0	0	0	1	0	0	1

3. Codifica las siguientes instrucciones:

```
add $t2, $t1, $t0
li $t0, 0x10002000
```

Puedes hacer uso del simulador PCSpim (o del QtSpim) para realizar el ejercicio. Ten en cuenta que las pseudo-instrucciones se desdoblan en un conjunto de instrucciones antes de su codificación. Justifica la respuesta (indicando los diferentes campos de cada instrucción: COp, Rs, Rt, …)

Solución:

```
add $10, $9, $8      -> 0x01285020
lui $1, 4096         -> 0x3c011000
ori $8, $1, 8192     -> 0x34282000

0x01285020 -> 0000 0001 0010 1000 0101 0000 0010 0000
C.Op = 0x00 -> Función = 0x20 (add) tipo R
Rd = 0x0A ($t2); Rt = 0x08 ($t0); Rs = 0x09 ($t1)

0x3c011000 -> 0011 1100 0000 0001 0001 0000 0000 0000
C.Op = 0x0F (lui) tipo I; Rt = 0x01 ($at); Inm = 0x1000

0x34282000 -> 0011 0100 0010 1000 0010 0000 0000 0000
C.Op = 0x0D (ori) tipo I; Rt = 0x08 ($t0); Rs = 0x01 ($at)
Inm = 0x2000
```

4. Analiza la operación que realizan las instrucciones codificadas de la siguiente forma a partir de la dirección 0x00400000.

Dirección	Contenido
0x00400000	0x3c081000
0x00400004	0x8d090000
0x00400008	0x8d0a0004
0x0040000C	0x012a5820
0x00400010	0xad0b0008

Para realizar este ejercicio debemos tener presente la codificación de las instrucciones MIPS.

Solución:

1a inst: 0x3c081000 0011 1100 0000 1000 0001 0000 0000 0000
COp = 0x0f (lui) -> tipo I; Rt = 0x08 ($t0); Inm = 0x1000
Instrucción: lui $t0, 0x1000

2a inst: 0x8d090000 -> 1000 1101 0000 1001 0000 0000 0000 0000
COp = 0x23 (lw) tipo I; Rt = 0x09 ($t1); Rs = 0x08 ($t0)
Inm = 0x0000
Instrucción: lw $t1, 0($t0)

3a inst: 0x8d0a0004 -> 1000 1101 0000 1010 0000 0000 0000 0100
COp = 0x23 (lw) tipo I; Rt = 0x0a ($t2); Rs = 0x08 ($t0)
Inm = 0x0004
Instrucción: lw $t2, 4($t0)

4a inst: 0x012a5820 -> 0000 0001 0010 1010 0101 1000 0010 0000
COp = 0x00 -> Función = 0x20 (add) tipo R; Rd = 0x0b ($t3)
Rt = 0x0a ($t2); Rs = 0x09 ($t1)
Instrucción: add $t3, $t2, $t1

5a inst 0xad0b0008 -> 1010 1101 0000 1011 0000 0000 0000 1000
COp = 0x2b (sw) tipo I; Rt = 0x0b ($t3); Rs = 0x08 ($t0)
Inm = 0x0008
Instrucción: sw $t3, 8($t0)

El código resultante es:

```
lui $t0, 0x1000
lw $t1, 0($t0)
lw $t2, 4($t0)
add $t3, $t2, $t1
sw $t3, 8($t0)
```

Este código lee dos palabras en las direcciones 0x10000000 y 0x10000004, las suma, y deposita el resultado en la dirección 0x100000008.

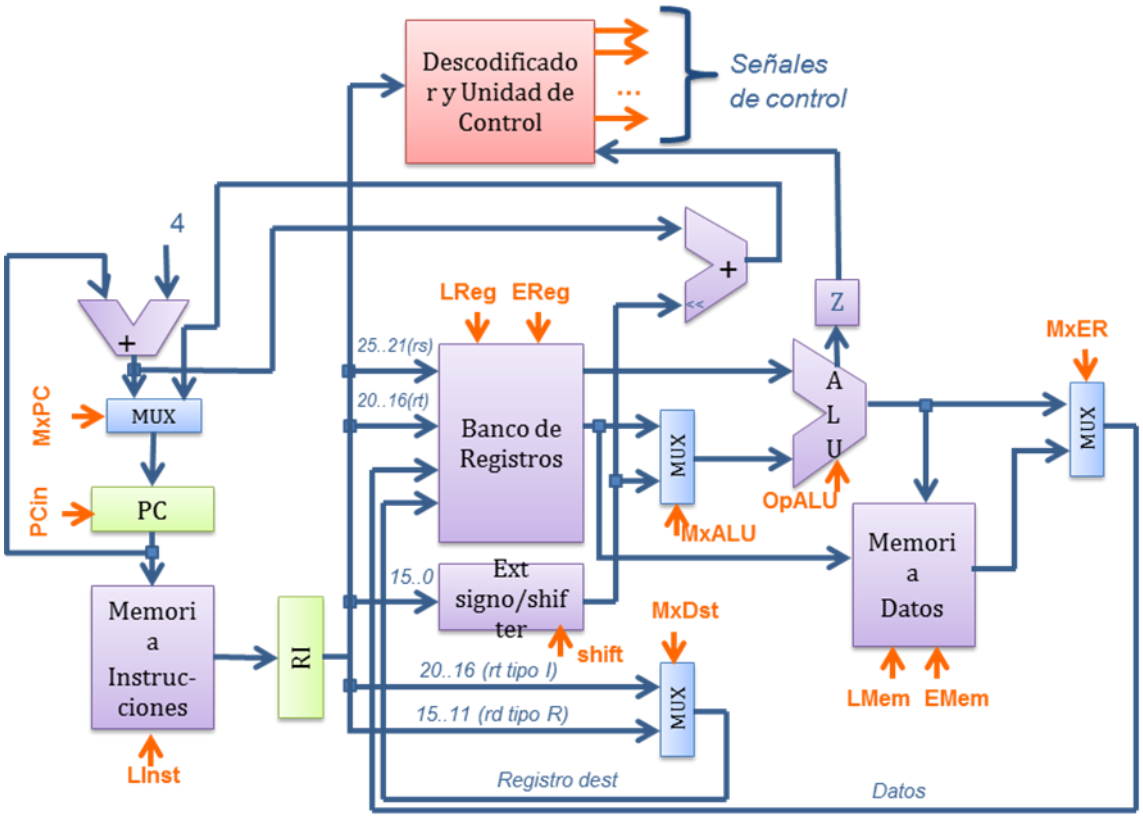
5. Modifica la ruta de datos para dar soporte a la instrucción

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

Para ello se decide modificar el recurso "Extensión de signo" con el fin de que tenga una funcionalidad adicional. El nuevo recurso se denomina ahora `Ext Signo/Shifter` y se configura con una nueva señal de control denominada `shift`. Cuando la señal `shift` vale 0 el recurso realiza la extensión de signo de la entrada. Cuando la señal `shift` vale 1, el recurso genera una salida de 32 bits a partir de la entrada de 16 bits, donde los 16 bits de la entrada aparecen en los 16 bits de mayor peso en la salida y los 16 bits de menor peso de la salida tienen el valor 0.

Indica los cambios a realizar tanto en la ruta de datos que se muestra a continuación, así como los cambios a realizar en la unidad de control.

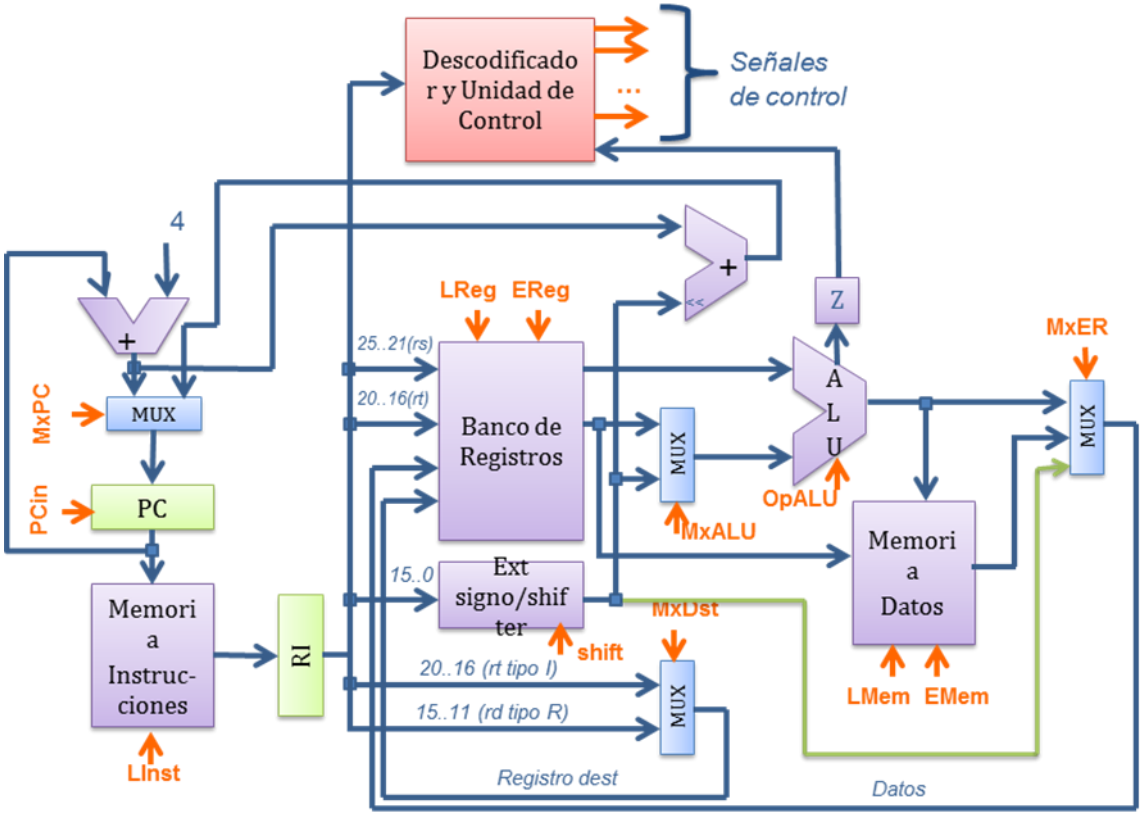
				Ext. Signo/ shifter	Banco Registros	ALU	Mem. Datos		Multiplexores Configuración Ruta de Datos			
Instrucción	Form	Código Op.	Función	shift	EReg	OpALU	LMem	EMem	MxPC	MxALU	MxDst	MxER
add rd, rs, rt	R	000000	100000		1	010	0	0	0	0	1	0
sub rd, rs, rt	R	000000	100010		1	110	0	0	0	0	1	0
and rd, rs, rt	R	000000	100100		1	000	0	0	0	0	1	0
or rd, rs, rt	R	000000	100101		1	001	0	0	0	0	1	0
lw rt, desp(rs)	I	100011			1	010	1	0	0	1	0	1
sw rt, desp(rs)	I	101011			0	010	0	1	0	1	X	X
beq rs, rs, etiq	I	000100			0	110	0	0	Z	0	X	X
lui rt, inm												



Solución:

La ruta de datos necesita de un nuevo cable que permita llevar el resultado del recurso Ext Signo/Shifter al banco de registros para su escritura. Para ello, debemos extender el multiplexor MxER para soportar una tercera entrada. La

señal de control del MxER debe configurarse de forma adecuada a la ejecución de la instrucción lui. La ruta de datos resultante es la siguiente:



La unidad de control resultante sería:

				Ext. Signo/ shifter	Banco Registros	ALU	Mem. Datos		Multiplexores Configuración Ruta de Datos			
Instrucción	Form	Código Op.	Función	shift	EReg	OpALU	LMem	EMem	MxPC	MxALU	MxDst	MxER
add rd, rs, rt	R	000000	100000	X	1	010	0	0	0	0	1	0
sub rd, rs, rt	R	000000	100010	X	1	110	0	0	0	0	1	0
and rd, rs, rt	R	000000	100100	X	1	000	0	0	0	0	1	0
or rd, rs, rt	R	000000	100101	X	1	001	0	0	0	0	1	0
lw rt, desp(rs)	I	100011		0	1	010	1	0	0	1	0	1
sw rt, desp(rs)	I	101011		0	0	010	0	1	0	1	X	X
beq rs, rs, etiq	I	000100		0	0	110	0	0	Z	0	X	X
lui rt, inm	I	001111		1	1	XXX	0	0	0	X	0	2