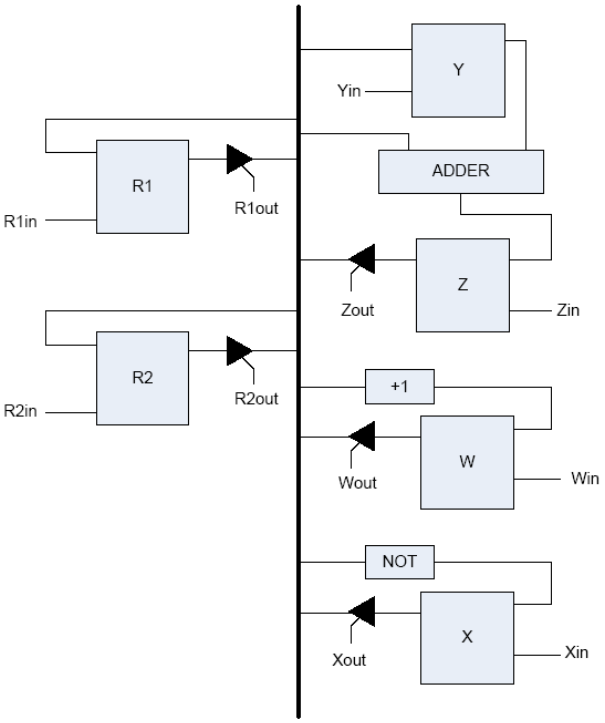
ESTRUCTURA DE COMPUTADORES
PROBLEMAS DE LA UNIDAD 5.- EL PROCESADOR III: LA RUTA DE DATOS MULTICICLO

5.1. Utilizando la arquitectura facilitada en la figura, se pide indicar ciclo a ciclo la ruta de datos así como en cada caso, las microinstrucciones necesarias para ejecutar la instrucción:

a) R1 ← R1 - R2 + 1

b) R2 ← 2*R2 + 2*R1 + 2

Describir ciclo a ciclo cada operación necesaria señalando el movimiento de datos entre registros (descripción RTL, *Register Transfer Level*).
Nota: para cada uno de los registros del sistema las variables: Yin, Zin, Win, Xin, R1in y R2in, habilitan en cada caso al registro para la escritura. Por su parte, las señales XXout habilitan el triestado correspondiente al ponerse a '1'.



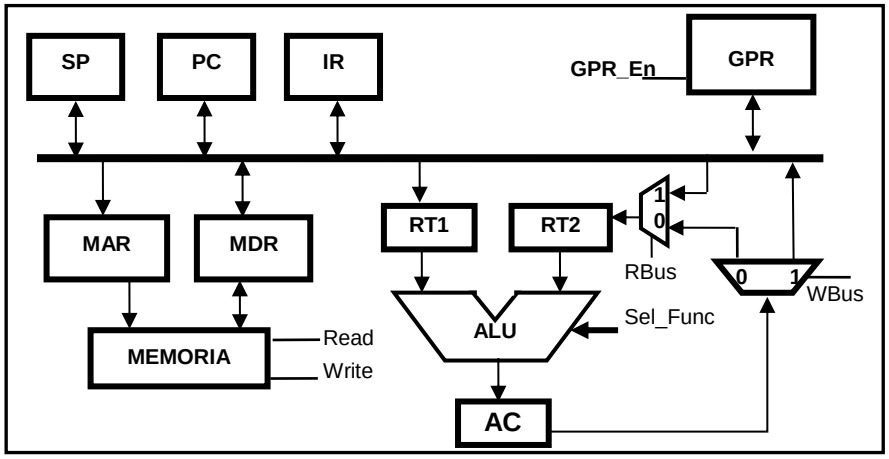
Solución:

a) OPERACIÓN: R1 ← R1 - R2 + 1			
CICLO	**Descripción RTL**	**Microinstrucción (sólo señales a '1')**	**Operación y comentario**
T1	**[X] ← / [R2]**	**R2out; Xin**	**X <= NOT(R2); en X guarda el C1(R2)**
T2	**[W] ← [X] + 1**	**Xout; Win**	**W <= X + 1; en W guarda el C2(R2)**
T3	**[Y] ← [W]**	**Wout; Yin**	**; en Y guarda el C2(R2)**
T4	**[Z] ← [R1] + [Y]**	**R1out; Zin**	**; en Z guarda (R1-R2)**
T5	**[W] ← [Z] + 1**	**Win; Zout**	**W <= Z + 1; en W guarda [(R1-R2) +1]**
T6	**[R1] ← [W]**	**Wout; R1in**	**; En R1 guarda el resultado pedido**

b) OPERACIÓN: R2 ← 2*R2 + 2*R1 + 2			
CICLO	**Acción (en lenguaje RTL)**	**Microinstrucción (sólo las señales a '1')**	**Comentario**
T1	**[Y] <= [R2]**	**R2out; Yin**	**Y = R2**
T2	**[Z] <= [R1] + [Y]**	**R1out; Zin**	**Z = R2+R1**
T3	**[W] <= [Z] + 1**	**Zout; Win**	**W = R2+R1+1**
T4	**[Y] <= [W]**	**Wout; Yin**	**Y = R2+R1+1**
T5	**[Z] <= [W] + [Y]**	**Wout; Zin**	**Z = (R2+R1+1)+(R2+R1+1)**
T6	**[R2] <= [Z]**	**Zout; R2in**	**R2 = 2*R2 + 2*R1 + 2**

ESTRUCTURA DE COMPUTADORES
PROBLEMAS DE LA UNIDAD 5.- EL PROCESADOR III: LA RUTA DE DATOS MULTICICLO

5.2. En la figura adjunta se muestra la ruta de datos de la arquitectura de un determinado ordenador de 16 bits, con las siguientes características: banco de registros de propósito general (GPR) con 16 registros de 16 bits; como registros específicos dispone de un puntero de pila (SP), un contador de programa (PC) y un registro de instrucciones (IR) con funciones conocidas y todos ellos de 16 bits. También dispone de los registros MDR y MAR utilizados para registrar el código o dato leído/escrito desde/en memoria el primero y para guardar la dirección de acceso el segundo. La arquitectura dispone de tres registros temporales, RT1, RT2 y ACC utilizados como registros de entrada y salida de la ALU. Este último puede ser redireccionado al registro RT2 o al bus común. En la ALU se han definido cuatro operaciones: sumar, restar, sumar '1' y restar '1', estas dos últimas sobre cualquiera de los dos operandos.



Se pide: señalar utilizando la descripción funcional del mismo en base al movimiento de datos entre registros (descripción RTL, *Register Transfer Level*) brevemente comentada, el mínimo número de operaciones elementales necesarias para ejecutar las instrucciones dadas (señalar el ciclo en el que deben ser ejecutadas).

a) JMP Rk ; Salto incondicional a la dirección contenida en el registro Rk.
b) STORE Rk, #direccion ; [Rk] => MEM[direccion].
c) LOAD R3, #dat_inm ; MEM[(dat_inm)] => [R3].

Nota1: Como se muestra en la figura, los registros comparten un único bus, lo que hay que tener en cuenta para evitar colisiones de datos.
Nota2: Se debe tener en cuenta que es posible diseñar más de una operación elemental por ciclo.

Solución:

	a) JMP Rk	
ciclo	Operación Elemental (RTL)	Comentario
①	PC => MAR	Prepara la dirección de memoria para capturar la instrucción
	PC => RT1	Actualización del PC (1): Guarda PC en RT1
②	MEN [MAR] => MDR	Recibe la instrucción leída
	RT1 + 1 => ACC	Actualización del PC (2): Incrementa RT1 en 1
③	MDR => IR	Guardo en el IR la instrucción capturada
	ACC => RT2	Actualización del PC (3): Guarda parcial en RT2
④	[Rk] => PC	Termina la ejecución de la instrucción con el valor del salto en el PC
	b) STORE Rk, #direccion	
ciclo	Operación Elemental (RTL)	Comentario
①	PC => MAR	Prepara la dirección de memoria para capturar la instrucción
	PC => RT1	Actualización del PC (1): Guarda PC en RT1
②	MEM [MAR] => MDR	Recibe la instrucción leída
	RT1 +1 => ACC	Actualización del PC (2): Incrementa RT1 en 1
③	MDR => IR	Guardo en el registro IR la instrucción capturada
	ACC => RT2	Actualización del PC (3): Guarda parcial en RT2
④	IR (dirección) => MAR	Prepara la dirección de memoria para escribir
	RT2 +1 => ACC	Actualización del PC (4): Incrementa RT2 en 1
⑤	[Rk] => MDR	Prepara el dato para escribir
⑥	ACC => PC	Actualización del PC (5): (PC + 2 => PC)
	MDR => MEM[MAR]	Escribe el dato

	c) LOAD R3, #dat_inm	
Ciclo	Operación Elemental	Comentario
①	PC => MAR	Prepara la dirección de memoria para capturar la instrucción
	PC => RT1	// Actualización del PC (1): Guarda PC en RT1
②	MEM [MAR] => MDR	Recibe la instrucción leída
	RT1 +1 => ACC	// Actualización del PC (2): Incrementa RT1 en 1
③	MDR => IR	Guardo en el registro IR la instrucción capturada
	ACC => RT2	// Actualización del PC (3): Guarda parcial en RT2
④	IR (dirección) => MAR	Prepara la dirección de memoria para leer
	RT2 +1 => ACC	// Actualización del PC (4): Incrementa RT2 en 1
⑤	MEM[MAR] => MDR	Escribe el dato leído de memoria
	ACC => PC	// Actualización del PC (5): (PC + 2 => PC)
⑥	MDR => [R3]	Escribe el dato leído en el registro destino

ESTRUCTURA DE COMPUTADORES
PROBLEMAS DE LA UNIDAD 5.- EL PROCESADOR III: LA RUTA DE DATOS MULTICICLO

5.3. Sea un sistema procesador basado en MIPS con arquitectura multiciclo igual que el estudiado en clase. Se recuerda que las direcciones para el acceso a memoria son de 32 bits. Se quiere ejecutar el código que se adjunta, en donde se señala el inicio de la memoria de código y el de la memoria de datos.

RAM CÓDIGO	RAM DATOS
.text 0	.data 0x2000
lab: lw $t2, X($0)	.space 8
sllv $t3, $t2, $t2	X: 0x0002
addi $t1, $t3, 0x200C	Y: 0x00FF
and $t4, $t1, $t2	Z:
xor $s1, $t4, $t4	
bne $s1,$0, lab	
0xAC112010	
fin: j fin	

a) Con la ayuda de la tabla de códigos facilitada, encuentre el código máquina para la instrucción del programa anterior "**bne $s1, $0, lab"**

op	rs	rt	Imm
000101	10001	00000	1111111111111010

En hexadecimal: 0x1620FFFA

b) El código ensamblador que corresponde al código máquina dado en el programa anterior **0xAC112010** y señale brevemente, teniendo en cuenta los operandos la función que esta instrucción realiza.

sw $s1, Z($0)

Escribe el contenido del registro $s1 = 0x0000 en la posición de memoria con dirección 0x2010 (var Z)

c) Con la ayuda de los diagramas facilitados, complete la tabla adjunta con los valores que en cada ciclo de reloj correspondan a las líneas de control señaladas, para ejecutar completamente las dos primeras instrucciones del programa dado.

Nota: Señalar sólo las señales de control con valor '1'

	lw $t2, X($0)					sllv $t3, $t2, $t2			
Ciclo	**T1**	**T2**	**T3**	**T4**	**T5**	**T6**	**T7**	**T8**	**T9**
IorD	0	X	X	1	X	0	X	X	X
IRWrite	1	0	0	0	0	1	0	0	0
RegDest	X	X	X	X	0	X	X	X	1
MemWrite	0	0	0	0	0	0	0	0	0
MemtoReg	X	X	X	X	1	X	X	X	0
RegWrite	0	0	0	0	1	0	0	0	1
PC_Write	1	0	0	0	0	1	0	0	0
Branch	0	0	0	0	0	0	0	0	0
ALUScrA	0	0	1	X	X	0	0	1	X
ALUScrB$_{(1:0)}$	01	11	10	XX	XX	01	11	00	XX
PCSrc$_{(1:0)}$	00	XX	XX	XX	XX	00	XX	XX	XX

Señale los valores de los registros indicados una vez ejecutadas completamente las tres primeras instrucciones, es decir, inmediatamente antes del comienzo de la ejecución de la cuarta **and $t4, $t1, $t2**

$pc = 0x0000000C **$t1 = 0x00002014**

$t2 = 0x00000002 **$t3 = 0x00000008**

ESTRUCTURA DE COMPUTADORES
PROBLEMAS DE LA UNIDAD 5.- EL PROCESADOR III: LA RUTA DE DATOS MULTICICLO

5.4. Conocida la ruta de datos multiciclo para MIPS, se pide:

a) Utilizando la tabla inferior, indicar el valor para cada uno de los cuatro ciclos de ejecución el valor de las señales de control para ejecutar las instrucciones:

sw $t1, 0x2404($0)

beq $s1, $t1, 0x8000.

b) Señalar brevemente la función de cada una de las dos instrucciones anteriores, teniendo en cuenta el valor de sus operandos, (para bne, suponer que PC = 0xFFFF0 y Z = '0')

Solución:

a)

	sw $t1, 0x2404($0)				beq $s1, $t1, 0x8000		
	Ciclo 1	Ciclo 2	Ciclo 3	Ciclo 4	Ciclo 1	Ciclo 2	Ciclo 3
IorD	0	X	X	1	0	X	X
IRWrite	1	0	0	0	1	0	0
Branch	0	0	0	0	0	0	1
MemWrite	0	0	0	1	0	0	0
MemtoReg	X	X	X	X	X	X	X
RegWrite	0	0	0	0	0	0	0
PC_Write	1	0	0	0	1	0	0
ALUScrA	0	0	1	X	0	0	1
$ALUScrB_{(1:0)}$	01	11	10	XX	01	11	00
$PCSrc_{(1:0)}$	00	XX	XX	XX	00	XX	01
$ALUControl_{(2:0)}$	010	010	010	XXX	010	010	110

b)

sw $t1, 0x2404($0)

Accede a la posición de memoria {$0 + ext_sig (0x2404)} = 0x00002404 y escribe en memoria el dato que contiene el registro **$t1**.

beq $s1, $t1, 0x8000 (Z = '0', PC = 0xFFFF0):

Como Z = '1', salta a la posición que se obtiene de sumar el valor actual de PC+4 (0x000FFFF4) con el dato inmediato que acompaña a la instrucción, multiplicado por 4 y extendido en signo, es decir 0xFFFE0000. Es un salto hacia atrás de 0x8000 instrucciones, a la dirección PC = 0x000DFFF4.

ESTRUCTURA DE COMPUTADORES
PROBLEMAS DE LA UNIDAD 5.- EL PROCESADOR III: LA RUTA DE DATOS MULTICICLO

5.5. Se quiere añadir en la ruta de datos multiciclo de MIPS estudiada en clase, los elementos de hardware necesarios para ejecutar una instrucción que permita intercambiar el contenido de dos registros de propósito general cualesquiera. La sintaxis ensamblador sería: **swap $0, $X, $Y** y también **swap $X, $Y**; donde $X y $Y representan los registros a intercambiar. Esta nueva instrucción de formato R-Type e identificada por el código funct = "110000", debe ejecutarse en 4 ciclos incluyendo la etapa de captura. Para facilitar el diseño se adjunta la descripción funcional en base al movimiento de datos entre registros (RTL) para la nueva instrucción.

①	[Instr] <= MEM [pc] ; [pc] <= [pc] + 4
②	[A] <= [rs]; [B] <= [rt]
③	[rt] <= [A]
④	[rs] <= [B]

Se pide:

a) Escribir el código máquina para esta instrucción tomando para X e Y dos valores cualesquiera.
b) Señalar de forma esquemática los cambios necesarios en la ruta de datos que se proponen para ejecutar la instrucción (nuevos dispositivos y/o nuevos caminos).
c) Señalar, indicando cómo y cuándo se utilizan, las nuevas variables definidas para el control de la ruta de datos una vez incorporada la nueva instrucción.

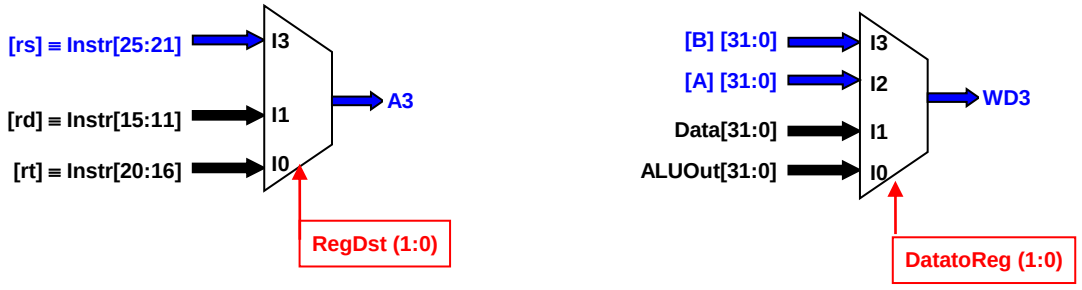
Solución:

a) El código de la instrucción **swap $s1, $t1** puede ser (se usa op de R-type y un funct libre):

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
op						rs					rt					rd					shamt					funct					
0	0	0	0	0	0	1	0	0	0	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0

b) Modificar la ruta de datos no supone cambiar los componentes hardware del sistema, tan solo hacer compatible el camino de los datos entre estos elementos para cada una de las instrucciones definidas, añadiendo nuevos caminos y nuevos dispositivos MUX o modificando los existentes.

La solución más sencilla, no es la única, implica sustituir los multiplexores de las entradas que controlan la escritura en banco de registros, A3 y WD3, por otros dos como los descritos a continuación y mostrados en la figura.



Además se deben implementar dos nuevos caminos o buses:

- ✓ De 5 bits desde el registro de instrucciones a la entrada (I3) del MUX de la entrada A3
- ✓ De 32 bits desde los registros A y B a las entradas I2 e I3, del MUX de la entrada WD3.

c) En esta solución, se han definido en dos nuevas variables (vectores) de control (ver figura):

RegDst$_{(1:0)}$. RegDst(0) equivale a la antigua señal de control RegDst y **s**e activa a:

- ✓ "00" cuando el registro donde se escribe se señala con el campo rt. Ocurre para lw en el C5, para las instrucciones I-Type con la ALU en C4 y para la nueva instrucción swap en C3.
- ✓ "01" cuando el registro donde se escribe se señala con el campo rd. Ocurre en el C4, para las instrucciones R-Type.
- ✓ "11" cuando el registro donde se escribe se señala con el campo rs. Ocurre para la nueva instrucción swap en C4.

DatotoReg$_{(1:0)}$, DatotoReg(0) equivale a la antigua señal de control MemtoReg y **s**e activa a:

- ✓ "00" cuando se escribe el resultado de la ALU, ocurre en C4 para las instrucciones aritmético-lógicas.
- ✓ "01" cuando se escribe el dato leído de memoria en C5 de lw.
- ✓ "10" cuando se escribe el contenido del registro A, ocurre para la nueva instrucción swap en C3.
- ✓ "11" cuando se escribe el contenido del registro B, ocurre para la nueva instrucción swap en C4.

ESTRUCTURA DE COMPUTADORES
PROBLEMAS DE LA UNIDAD 5.- EL PROCESADOR III: LA RUTA DE DATOS MULTICICLO

5.6. En el caso de rutinas anidadas en MIPS, se debe tener la precaución de guardar en la pila la dirección de retorno ($ra) antes de realizar una nueva llamada con la instrucción **jal**. Para ello el proceso normal consiste en descontar 4 posiciones el puntero de pila (**add $sp, $sp, -4**) y a continuación guardar la dirección de retorno en la pila (**sw $ra, 0($sp)**). Se quiere añadir al juego de instrucciones de MIPS una nueva instrucción denominada **savelink** que sustituya a las dos anteriores. Esta instrucción se considera del tipo Memoria y se comporta de forma similar a store (**sw**), aunque en este caso todos los operandos son implícitos y por tanto no es necesario indicarlos en la instrucción ensamblador, aunque sí hay que tenerlos en cuenta al codificar la instrucción. Sintaxis equivalentes de esta nueva instrucción son **savelink** sin operandos y también **savelink $ra, 0($sp)**. El código op identificativo de esta nueva instrucción es "101100". Se pide:
a) Escribir el código máquina (único) que corresponde a esta instrucción.
b) Realizar la descripción RTL de los distintos ciclos de reloj que permiten describir la nueva instrucción **savelink.**
c) Describir de forma esquemática los cambios, si los hubiera, que habría que realizar en la arquitectura del MIPS para poder ejecutar la nueva instrucción **savelink**.
d) Completar las señales de control necesarias para controlar la ejecución de la instrucción **savelink**.

Solución:

a) La solución supone codificar **savelink** con su código OP correspondiente, como si se tratara de una instrucción I-type como store (**sw**) con los operandos implícitos: rt = $ra = 31, rs = $sp = 29 e imm = - 4, es decir:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
op						rs					rt					imm																
1	0	1	1	0	0	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0

b) Realizar la descripción RTL de los distintos ciclos de reloj que permiten describir la nueva instrucción savelink.

① **Captura (fetch) de la instrucción desde memoria. Actualización del PC**

[Instr] <= MEM [pc]; [pc] <= [pc] + 4

② **Registro de los operandos desde los campos rs y rt a los registros A y B**

[A] <= [rs]; [B] <= [rt]

③ **Suma de los operandos y registro de la dirección efectiva a memoria en ALUOut**

[ALUOut] <= [A] + Ext_sig (Instr[15:0])

IF {Intr[31] = ..1..} (lw, sw, savelink)

④ **IF (Intr[29:26] = ..1011..) THEN** **!** Instrucción **lw**

MEM[ALUOut] => Data[31:0];

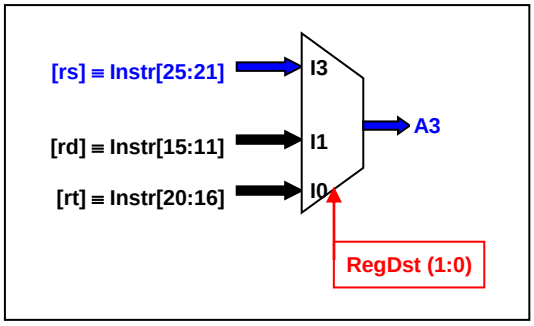
ELSE IF (Intr[29:26] = 1011) THEN **!** Instrucción **sw**

[B] => MEM[ALUOut];

ELSE IF (Intr[29:26] = .. 1100..) THEN **!** Instrucción **savelink**

[B] => MEM[ALUOut]; [ALUOut] => [rs];

c) Para ejecutar la ruta de datos propuesta, es preciso:
Aumentar el número de entradas en el MUX situado en la entrada A3 del banco de registros para la dirección indicada por el campo rs (Instr[25:21]= 29) en **savelink**.



d) No hay novedad con respecto a load (lw) o store (sw) en los tres primeros ciclos. La nueva señal RegDst(1:0) = "11" debe estar activa en el cuarto.

	IorD	IRWrite	$RegDest_{(1:0)}$	MemWrite	MemtoReg	RegWrite	PC_Write	Branch	ALUScrA	$ALUScrB_{(1:0)}$	$PCSrc_{(1:0)}$	$ALUControl_{(2:0)}$
T1	**0**	**1**	**XX**	**0**	**X**	**0**	**1**	**0**	**0**	**01**	**00**	**010**
T2	**X**	**0**	**XX**	**0**	**X**	**0**	**0**	**0**	**0**	**11**	**XX**	**XXX**
T3	**X**	**0**	**XX**	**0**	**X**	**0**	**0**	**0**	**1**	**10**	**XX**	**010**
T4	**1**	**0**	**11**	**1**	**0**	**1**	**0**	**0**	**X**	**XX**	**XX**	**XXX**

ESTRUCTURA DE COMPUTADORES
PROBLEMAS DE LA UNIDAD 5.- EL PROCESADOR III: LA RUTA DE DATOS MULTICICLO

5.7. En el juego de instrucciones de MIPS, se desea incorporar una nueva instrucción denominada **lli (**load lower immediate**)**, similar a la especificada en el juego de instrucciones lui pero en esta caso se completan los 16 lsb del registro destino implicado. La descripción funcional en base al movimiento de datos es para los tres primeros ciclos similar a lw o sw y para el cuarto y último:

④ IF {Instr[31:26] = 110000} THEN [rt] ⇐ Ext_cero (imm)

Se pide:

a) Describir de forma breve, utilizando un ejemplo, el funcionamiento de esta nueva instrucción.
b) Escribir el código máquina que corresponde a la instrucción definida anteriormente en el ejemplo.
c) Dada la ruta de datos para la arquitectura multiciclo del procesador MIPS estudiado en teoría, describir de forma esquemática los cambios, si los hubiera, que habría que realizar en la arquitectura para poder ejecutar la nueva instrucción.
d) Señalar el mínimo número de ciclos necesarios para ejecutar esta nueva instrucción e indicar las líneas de control antiguas y nuevas en cada ciclo, si las hubiera, necesarias para controlar la ejecución de la nueva instrucción.

SOLUCION

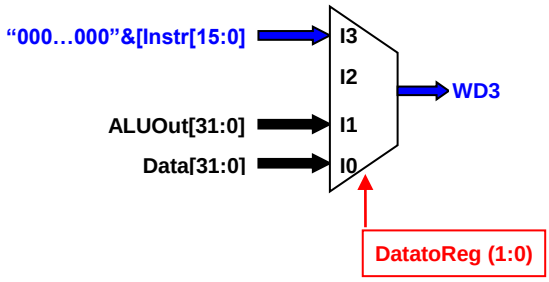
a) Sea la instrucción **lli $t4, 0x80A0($0).** Los 16 bits del dato inmediato 0x80A0 se guardan extendidos a ceros hasta 32 bits, 0x000080A0 en el registro $t4, señalado en el campo rt de la instrucción.

b) El código máquina solicitado para **lli $t4, 0x80A0($0)**:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
op						rs					rt					imm																
1	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0

c) Los nuevos elementos hardware a incorporar son:

1. Un circuito para la extensión de ceros de 16 a 32 bits.
2. Modificar el multiplexor a la entrada WD3 del GPR para permitir la escritura del nuevo operando. Una solución puede ser la mostrada en la figura.



d) La nueva señal de control $DatatoReg_{(1:0)}$ = "11" selecciona la opción para la nueva instrucción lli.

	IorD	IRWrite	RegDest	MemWrite	$DatotoReg_{(1:0)}$	RegWrite	PC_Write	Branch	ALUScrA	$ALUScrB_{(1:0)}$	$PCSrc_{(1:0)}$	$ALUControl_{(2:0)}$
T1	0	1	X	0	XX	0	1	0	0	01	00	010
T2	X	0	X	0	XX	0	0	0	0	11	XX	XXX
T3	X	0	0	0	11	1	0	0	X	XX	XX	010

ESTRUCTURA DE COMPUTADORES
PROBLEMAS DE LA UNIDAD 5.- EL PROCESADOR III: LA RUTA DE DATOS MULTICICLO

5.8. El procesador MIPS multiciclo visto en teoría ejecuta el código que se muestra a continuación. Se pide escribir una tabla con la evolución a lo largo de los ciclos de reloj de diversas señales, sabiendo que en el ciclo 1 de reloj se está ejecutando el primer ciclo de la instrucción **sw,** hasta que se ejecute completamente la instrucción **or**.
Nota: Si se accede a memoria a una posición cuyo valor no se puede conocer por otros datos del enunciado se supondrá que el contenido de dicha posición de memoria es 0xCC.

```
            sw $s1, A($s2)
            jal  func
            add $s2, $s2, $s2
        .text 0x0100
func:       lw    $s3, B($s2)
            or $s1, $s3, $s1
        .data 0x2000
A:          0x0020
B:
```

Nota: Escribir en la tabla sólo cuando se produzca un cambio con respecto al valor en el ciclo anterior.

	sw				jal			lw					or				
Ciclo	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**
pc	**0x00**	0x04	=	=	=	0x08	=	0x100	0x104	=	=	=	=	0x108	=	=	0x108
$s1	**0x08**	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=	0xCC
$s2	**0x40**	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=
$s3	**0x00**	=	=	=	=	=	=	=	=	=	=	=	0xCC	=	=	=	=
$ra	**0x00**	=	=	=	=	=	=	0x08	=	=	=	=	=	=	=	=	=
IorD	**0**	**X**	**X**	**1**	**0**	**X**	**X**	**0**	**X**	**X**	**1**	**X**	**0**	**X**	**X**	**X**	
IRWrite	**1**	**0**	**0**	**0**	**1**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	
RegDest	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**0**	**X**	**X**	**X**	**1**	
MemWrite	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	
MemtoReg	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**1**	**X**	**X**	**X**	**0**	
RegWrite	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**1**	
PC_Write	**1**	**0**	**0**	**0**	**1**	**0**	**1**	**1**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	
ALUScrA	**0**	**0**	**1**	**X**	**0**	**0**	**X**	**0**	**0**	**1**	**X**	**X**	**0**	**0**	**1**	**X**	
ALUScrB$_{(1:0)}$	**01**	**11**	**10**	**XX**	**01**	**11**	**XX**	**01**	**11**	**10**	**XX**	**XX**	**01**	**11**	**00**	**XX**	
PCSrc$_{(1:0)}$	**00**	**XX**	**XX**	**XX**	**00**	**XX**	**10**	**00**	**XX**	**XX**	**XX**	**X**	**00**	**XX**	**XX**	**XX**	
ALUControl$_{(2:0)}$	**010**	**010**	**010**	**XX**	**010**	**010**	**XX**	**010**	**010**	**010**	**XX**	**XX**	**010**	**010**	**001**	**XX**	

ESTRUCTURA DE COMPUTADORES
PROBLEMAS DE LA UNIDAD 5.- EL PROCESADOR III: LA RUTA DE DATOS MULTICICLO

5.9. Cuando en MIPS se trabaja con funciones o subrutinas, es posible hacer uso de la pila para pasar los parámetros necesarios y para obtener el resultado de ejecución de la función. Para simplificar esta tarea se quiere añadir una nueva instrucción al procesador denominada **push** que se emplea para poner (escribir) un dato en la pila, de modo que sustituya al conjunto de instrucciones necesario para realizar esta tarea. Esta nueva instrucción se considera de tipo memoria y por tanto se comporta de manera similar a sw, aunque en este caso sólo tiene un operando (**push $rx**), donde $rx representa al registro cuyo contenido se quiere guardar en la pila. Se pide:

a) Escriba las instrucciones a las que sustituye esta instrucción.

push $rx: **addi $sp, $sp, - 4**
sw $rx, 0($sp)

b) Escriba el código máquina de la instrucción **push $rx** suponiendo que su op = "111000". (Utilice un valor cualquiera para el operando registro $rx).

La instrucción push $t1 ($t1 = $9), recibe como registro destino el registro que se quiere guardar en la pila y se guarda en memoria en la posición marcada por rs = $sp + [Ext_sig (Imm)].

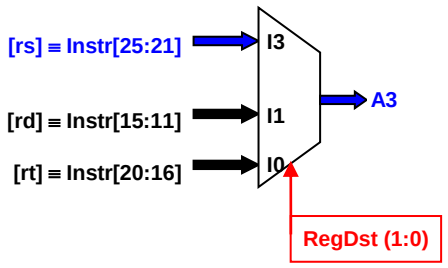
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
op						rs					rt					imm																
1	1	1	0	0	0	1	1	1	0	1	0	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0

c) Dada la ruta de datos para el procesador MIPS multiciclo visto en teoría, describa de forma esquemática los cambios, si los hubiera y si fueran posibles, que habría que realizar en la arquitectura para poder ejecutar la nueva instrucción. Indique las señales de control necesarias para ejecutar esta nueva instrucción.

La instrucción push sí es posible.

Con la codificación señalada push actúa exactamente igual que store, sw $rx, -4($sp), por lo que no es necesario modificar nada de la ruta de datos sino tan solo señalar al controlador que debe tratar push como si fuera sw.

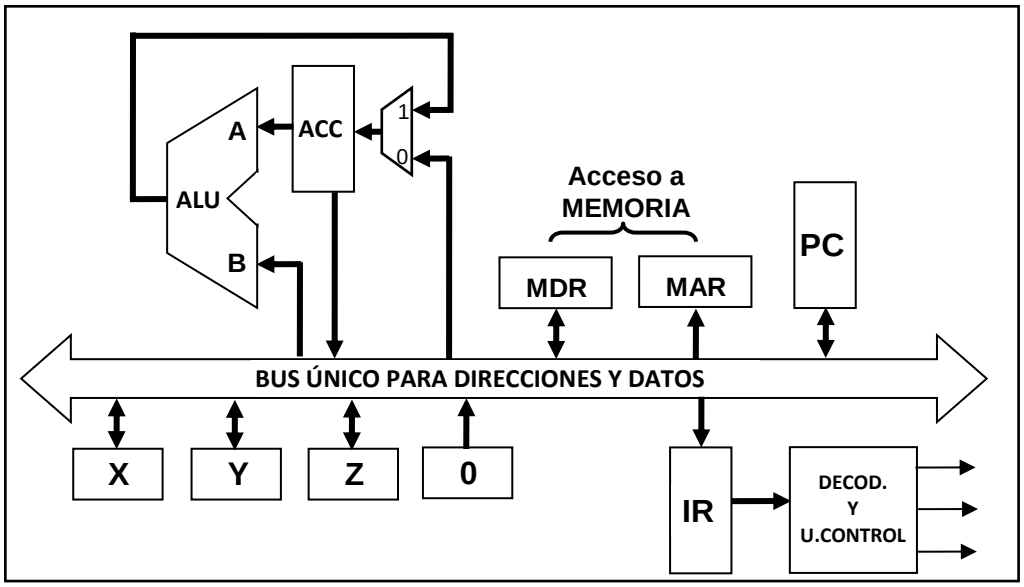
La otra parte de la funcionalidad es actualizar $sp <= $sp – 4. Para lo cual, se debe direccionar A3 en la entrada del campo rs y WD3 con el valor ($sp – 4), que se encuentra en el registro ALUOp a la salida de la ALU.



	IorD	IRWrite	$RegDest_{(1:0)}$	MemWrite	MemtoReg	RegWrite	PC_Write	Branch	ALUScrA	$ALUScrB_{(1:0)}$	$PCSrc_{(1:0)}$	$ALUControl_{(2:0)}$
T1	0	1	XX	0	X	0	1	0	0	01	00	010
T2	X	0	XX	0	X	0	0	0	0	11	XX	010
T3	X	0	XX	0	X	0	0	0	1	10	XX	010
T4	1	0	11	1	0	1	0	0	X	XX	XX	XXX

ESTRUCTURA DE COMPUTADORES
PROBLEMAS DE LA UNIDAD 5.- EL PROCESADOR III: LA RUTA DE DATOS MULTICICLO

5.10. En la figura adjunta se muestra la ruta de datos de la arquitectura tipo Von Neumann de un procesador que no es MIPS.



En la figura se observan los siguientes elementos: 3 registros de propósito general (X, Y, Z), un registro que siempre contiene cero y cinco registros específicos: contador de programa (PC), registro de instrucciones (IR), un registro acumulador (ACC) que se emplea, tal y como indica el esquema como registro de entrada y de salida de la ALU, un registro para señalar la dirección de acceso a memoria (MAR) y un registro para guardar el dato leído/escrito desde/en memoria (MDR). Todos los registros mencionados son de 16 bits. Todos los registros acceden a un bus común y disponen del control necesario para leer y/o escribir en el citado bus único. Estos controles no están señalados en el esquema son suministrados por la Unidad de Control. En la ALU se han implementado cuatro operaciones: sumar (A+B), restar (A-B), incrementar una unidad (A+1) y la función XOR (A⊕B), donde A y B señalan las entradas de la ALU. Se pide:
a) Utilizando la descripción funcional en base al movimiento de datos entre registros (descripción RTL, *Register Transfer Level*), el mínimo número de operaciones elementales necesarias para:
a) La captura de cualquier instrucción y la actualización del registro PC.
b) Los ciclos siguientes para la ejecución de la instrucción: **SUB X, Y, Z** (Y - Z => X)
c) Los ciclos siguientes para la ejecución de la instrucción: **STORE X, Y** (X => MEM [Y])

Nota: Es preciso señalar el ciclo en el que debe ser ejecutada cada operación de los apartados anteriores.

Solución:

CICLO	a) Captura de la Instrucción y Actualización del PC	
C1	PC => MAR PC => ACC	; Direcciona la memoria ; Prepara la actualización del PC
C2	MEM[MAR] => MDR ACC + 1 =ACC	; Recibe la instrucción de memoria ; Incrementa en 1 el Acumulador (PC + 1)
C3	MDR => IR ACC + 1 =ACC	; Registra la instrucción en IR ; Incrementa en 1 el Acumulador (PC + 2)
C4	ACC => PC	; Actualiza el PC con al valor siguiente PC = PC + 2.
CICLO	b) SUB X, Y, Z	
C5	Y => ACC	; Escribe el minuendo en el ACC.
C6	ACC => ACC - Z	; Realiza la resta y escribe el resultado en el ACC.
C7	ACC => X	; Escribe el resultado en el registro destino
CICLO	c) STORE X, Y	
C5	Y => MAR	; Envía la dirección de memoria donde escribir
C6	X => MDR	; Envía el dato que se quiere escribir en memoria
C7	MDR => MEM[MAR]	; Escribe el contenido de MDR en memoria

ESTRUCTURA DE COMPUTADORES
PROBLEMAS DE LA UNIDAD 5.- EL PROCESADOR III: LA RUTA DE DATOS MULTICICLO

5.11. Sea un sistema procesador basado en MIPS con arquitectura multiciclo igual que el estudiado en clase. Se quiere ejecutar el código que se adjunta:

```
        .text 0x0000
            lw $s1,B($0)
            beq $s1,$s3, fin
            jal  etiq
            lui $s1,0x1CAA
        .text 0x0028
fin:        j fin
        .data 0x2000
A:      0xE3
B:      0xBF
```

Se pide completar la siguiente tabla con la evolución a lo largo de los ciclos de reloj de diversas señales y registros, sabiendo que en el ciclo 1 de reloj se está ejecutando el primer ciclo de la primera instrucción del programa:

Instrucción	**lw**					**beq**			**j**			
Ciclo	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
pc	**0x00**	0x04	=	=	=	=	0x08	=	0x28	0x2C	=	0x28
$s1	**0x0A**	=	=	=	=	0xBF	=	=	=	=	=	=
$s2	**0x02**	=	=	=	=	=	=	=	=	=	=	=
$s3	**0xBF**	=	=	=	=	=	=	=	=	=	=	=
$ra	**0x00**	=	=	=	=	=	=	=	=	=	=	=
IorD	**0**	**X**	**X**	**1**	**X**	**0**	**X**	**X**	**0**	**X**	**X**	
IRWrite	**1**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**1**	**0**	**0**	
RegDest	**X**	**X**	**X**	**X**	**0**	**X**	**X**	**X**	**X**	**X**	**X**	
MemWrite	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	
MemtoReg	**X**	**X**	**X**	**X**	**1**	**X**	**X**	**X**	**X**	**X**	**X**	
RegWrite	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	
PC_Write	**1**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**1**	**0**	**1**	
Branch	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	
ALUSrcA	**0**	**0**	**1**	**X**	**X**	**0**	**0**	**1**	**0**	**0**	**X**	
$ALUSrcB_{(1:0)}$	**01**	**11**	**10**	**XX**	**XX**	**01**	**11**	**00**	**01**	**11**	**XX**	
$PCSrc_{(1:0)}$	**00**	**XX**	**XX**	**XX**	**XX**	**00**	**XX**	**01**	**00**	**XX**	**10**	

ESTRUCTURA DE COMPUTADORES
PROBLEMAS DE LA UNIDAD 5.- EL PROCESADOR III: LA RUTA DE DATOS MULTICICLO

5.12. Sea un sistema procesador basado en MIPS con arquitectura multiciclo igual que el estudiado en clase. Se quiere ejecutar el código que se adjunta:

Código

```
        .text 0x0000
          lw $s1, 5($s2)
          and $s2, $s1, $s3
          beq $s1,$s2,salto
          sw $s2, -4($s3)
          j fin
        .text 0x001C
salto:  jal fin
          or $s3,$s1,$s2
fin:    and $s2,$s1,$s3
          sw $s3,B($s2)
```

```
      .data 0x2200
A:  0x00000002
B:  0x00002200
C:  0x0000EA17
```

Contenido final de la memoria:

.data 0x2200

A: **0x00000002**

B: **0x00002200**

C: **0x0000004C**

Se pide completar la tabla facilitada con los valores de las señales y de los registros indicados, desde el comienzo (primer ciclo) de la ejecución de la instrucción etiquetada con "salto", hasta que se ejecute completamente el programa. Rellene además el contenido final de las posiciones de memoria indicadas (cuadro superior).

Instrucción	**jal**			**and**				**sw**				
Ciclo	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
pc	0x1C	0x20	=	0x24	0x28	=	=	=	0x2C	=	=	0x2C
$s1	**0xA4**	=	=	=	=	=	=	=	=	=	=	0xA4
$s2	**0x02**	=	=	=	=	=	=	0x04	=	=	=	0x04
$s3	**0x4C**	=	=	=	=	=	=	=	=	=	=	0x4C
$ra	**0x00**	=	=	0x20	=	=	=	=	=	=	=	0x20
IorD	**0**	**X**	**X**	**0**	**X**	**X**	**X**	**0**	**X**	**X**	**1**	
IRWrite	**1**	**0**	**0**	**1**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	
RegDest	**X**	**X**	**X***	**X**	**X**	**X**	**1**	**X**	**X**	**X**	**X**	
MemWrite	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	
MemtoReg	**X**	**X**	**X***	**X**	**X**	**X**	**0**	**X**	**X**	**X**	**X**	
RegWrite	**0**	**0**	**1**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	
PC_Write	**1**	**0**	**1**	**1**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	
Branch	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	
ALUSrcA	**0**	**0**	**X**	**0**	**0**	**1**	**X**	**0**	**0**	**1**	**X**	
$ALUSrcB_{(1:0)}$	**01**	**11**	**XX**	**01**	**11**	**00**	**XX**	**01**	**11**	**10**	**XX**	
$PCSrc_{(1:0)}$	**00**	**XX**	**10**	**00**	**XX**	**XX**	**XX**	**00**	**XX**	**XX**	**XX**	

*** Se desconoce la implementación en la arquitectura para la operación (PC+4) => $ra ($31)**

ESTRUCTURA DE COMPUTADORES
PROBLEMAS DE LA UNIDAD 5.- EL PROCESADOR III: LA RUTA DE DATOS MULTICICLO

5.13. Sea un sistema procesador basado en MIPS con arquitectura multiciclo igual que el estudiado en la asignatura. Se quiere ejecutar el código que se adjunta:

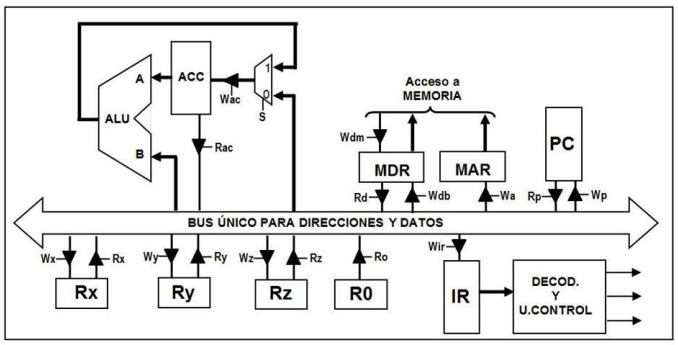
```
        .text 0x0000
            j eti
            beq $s1,$s3, fin
eti:        sw $s2, B($0)
fin:        j fin
        .data 0x2000
A:      0x01
B:      0x02
```

Se pide completar la siguiente tabla con la evolución a lo largo de los ocho primeros ciclos de reloj de diversas señales, registros y contenido de memoria, sabiendo que en el ciclo 1 de reloj se está ejecutando el primer ciclo de la primera instrucción del programa.

Instrucción	**j**	**j**	**j**	**sw**	**sw**	**sw**	**sw**	**j**
Ciclo	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
pc	**0x00**	0x04	=	0x08	0x0C	=	=	=
$s1	**0x08**	=	=	=	=	=	=	=
$s2	**0x09**	=	=	=	=	=	=	=
$s3	**0x0A**	=	=	=	=	=	=	=
A:	**0x01**	=	=	=	=	=	=	=
B:	**0x02**	=	=	=	=	=	=	0x09
IorD	**0**	**X**	**X**	**0**	**X**	**X**	**1**	**0**
MemWrite	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**
IRWrite	**1**	**0**	**0**	**1**	**0**	**0**	**0**	**1**
RegDest	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**
MemtoReg	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**
RegWrite	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
$PCSrc_{(1:0)}$	**00**	**XX**	**10**	**00**	**XX**	**XX**	**XX**	**00**
PCWrite	**1**	**0**	**1**	**1**	**0**	**0**	**0**	**1**
Branch	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
ALUSrcA	**0**	**0**	**X**	**0**	**0**	**1**	**X**	**0**
$ALUSrcB_{(1:0)}$	**01**	**11**	**XX**	**01**	**11**	**10**	**XX**	**01**

ESTRUCTURA DE COMPUTADORES
PROBLEMAS DE LA UNIDAD 5.- EL PROCESADOR III: LA RUTA DE DATOS MULTICICLO

5.14. En la figura adjunta se muestra la ruta de datos de la arquitectura tipo Von Neumann de un procesador que no es MIPS. En la figura se observan los siguientes elementos: tres registros de propósito general (Rx,Ry,Rz), un registro que contiene siempre el valor cero (R0) y cinco registros específicos: contador de programa (PC), registro de instrucciones (IR), un registro acumulador (ACC) que se emplea, tal y como indica el esquema como registro de entrada y de salida de la ALU, un registro para señalar la dirección de acceso a memoria (MAR) y un registro para guardar el dato leído/escrito desde/en memoria (MDR). Todos los registros mencionados son de 16 bits. Todos los registros acceden a un bus común y disponen del control necesario, suministrados por la Unidad de Control (UC), para leer y/o escribir en el citado bus único.



En la ALU se han implementado cuatro operaciones: sumar (A+B), restar (A-B), incrementar en una unidad (A+1) y la función XOR (A⊕B), donde A y B señalan las entradas de la ALU. Se pide:

a) Utilizando la descripción funcional en base al movimiento de datos entre registros (descripción RTL, *Register Transfer Level*), el mínimo número de operaciones elementales necesarias para ejecutar completamente, captura y actualización del PC incluidas, las instrucciones siguientes:

a1) LOAD X, Y ==> (MEM [Y] => X)

a2) JRAL RX ==> ([RX] => PC, [PC]+2 => RZ) Salta a RX y guarda dirección de vuelta en RZ

b) Para cada instrucción, indique para cada ciclo la palabra de control necesaria que debe generar la UC para ejecutar cada una de las operaciones definidas. Indicar sólo las señales de control que deben estar activas (valor = '1')

Nota: Utilice una fila de la tabla por cada ciclo de reloj. Añada o elimine filas en la tabla en función del resultado que se proponga.

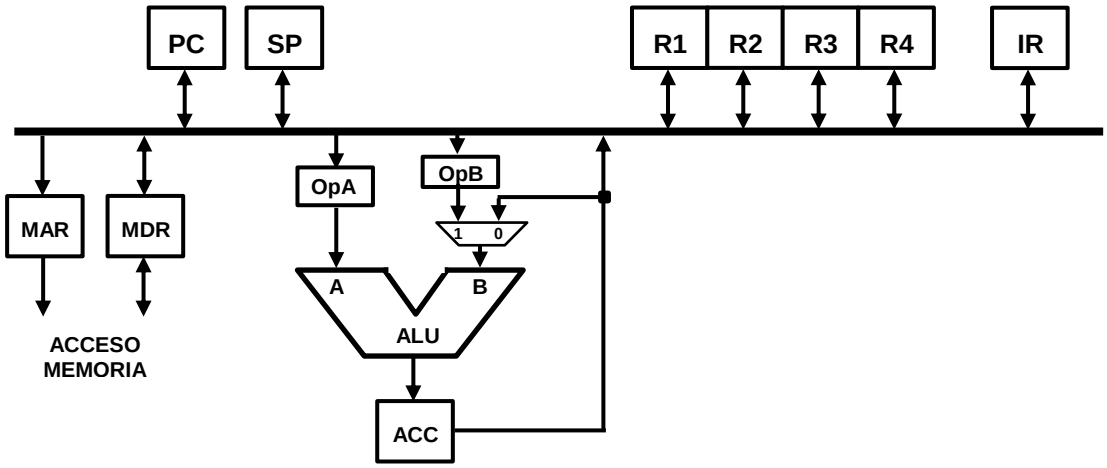
Ciclo	**a1) Seudocódigo RTL**	**b) Palabra de control** (señales con valor = '1')	**Comentario**
T1	PC => MAR PC => ACC	Rp, Wa Rp, Wac	- Direcciona la memoria - Prepara la actualización del PC
T2	MEM[MAR] => MDR ACC + 1 => ACC	Wdm S, Wac	- Recibe la instrucción de memoria - Incrementa en 1 el Acumulador (PC + 1)
T3	MDR => IR ACC + 1 => ACC	Rd, Wir S, Wac	- Registra la instrucción en IR - Incrementa en 1 el Acumulador (PC + 2)
T4	ACC => PC	Rac, Wp	- Actualiza el PC con al valor PC = PC + 2. (*)
T5	R_Y => MAR	Ry, Wa	- Envía la dirección de memoria donde leer
T6	MEM[MAR] => MDR	Wdm	- Lee el dato que se quiere escribir en registro
T7	MDR => R_X	Rd, Wx	- Escribe el contenido de MDR en el registro
Ciclo	**a2) Seudocódigo RTL**	**b) Palabra de control** (señales con valor = '1')	**Comentario**
T1	PC => MAR PC => ACC	Rp, Wa Rp, Wac	- Direcciona la memoria - Prepara la actualización del PC
T2	MEM[MAR] => MDR ACC + 1 => ACC	Wdm S, Wac	- Recibe la instrucción de memoria (opcode) - Incrementa en 1 el Acumulador (PC + 1)
T3	MDR => IR ACC + 1 => ACC	Rd, Wir S, Wac	- Registra el opcode en IR - Incrementa en 1 el Acumulador (PC + 2)
T4	ACC => RZ	Rac, Wz	- Guarda la dirección de vuelta (PC+2) en RZ
T5	RX => PC	Rx, Wp	- RX pasa al PC (**)

(*) Esta solución contempla la compatibilidad con otras instrucciones de este mismo sistema, en las cuales el registro acumulador podría ser usado en el primer ciclo de ejecución de dichas instrucciones. Si se considera esta instrucción de forma independiente, el enunciado no niega esta opción, la operación en el T4, ACC=>PC podría ejecutarse en el mismo ciclo que MEM[MAR] => MDR del T6, con lo que se ahorraría un ciclo.

(*) Los ciclos T4 y T5 pueden ejecutarse en cualquier orden.

ESTRUCTURA DE COMPUTADORES
PROBLEMAS DE LA UNIDAD 5.- EL PROCESADOR III: LA RUTA DE DATOS MULTICICLO

5.15. En la figura adjunta se muestra la ruta de datos de la arquitectura de un determinado ordenador de 8 bits, con las siguientes características: cuatro registros de propósito general (R1, R2, R3 y R4). Como registros específicos dispone de un puntero de pila (SP), un contador de programa (PC) y un registro de instrucciones (IR) con funciones conocidas y todos ellos de 8 bits. También dispone de los registros MDR y MAR utilizados para registrar el código o dato leído/escrito desde/en memoria el primero y para guardar la dirección de acceso el segundo. La arquitectura dispone de tres registros temporales, OpA, OpB y ACC, utilizados como registros de entrada y salida de la ALU respectivamente. Este último puede ser redireccionado a la propia ALU o al bus común. En la ALU se han definido las operaciones: A+B, A+1, B+1, A NAND B y A XOR B. Por claridad en el esquema no se incluyen las líneas de control necesarias para definir la ruta de datos.



Se sabe que en el instante inicial T0, el registro PC contiene la dirección de la instrucción "**comp R1**" que calcula el complemento a dos del contenido del registro señalado y lo guarda en el mismo registro.

Se pide utilizando expresiones a nivel de transferencia de registros (RTL), escribir ciclo a ciclo las operaciones necesarias para **a)** capturar la instrucción y **b)** ejecutar la instrucción. Considere también las operaciones necesarias para actualizar el PC, entendiendo que pueden ejecutarse en paralelo con cualquiera de las operaciones anteriores.

Ciclo	Operación	Comentario, si necesario
T1	[MAR] <= [PC] [OpA] <= [PC]	Dirección de la instrucción a memoria
T2	[MDR] <= MEM([MAR]) [Acc] <= [OpA] + 1	Lectura de la instrucción de memoria Incremento en una unidad para actualizar el PC
T3	[IR] <= [MDR]	Fin de la captura de la instrucción
T4	[PC] <= [Acc]	Fin de actualización del PC
T5	[OpA] <= [R1] [OpB] <= [R1]	Escribe R1 en ambos registros operando
T6	[Acc] <= [OpA] NAND [OpB]	El acumulador guarda el complemento de R1
T7	[Acc] <= [Acc] + 1	El acumulador guarda el complemento a dos de R1
T8	[R1] <= [Acc]	Guarda el resultado en el registro destino

ESTRUCTURA DE COMPUTADORES
PROBLEMAS DE LA UNIDAD 5.- EL PROCESADOR III: LA RUTA DE DATOS MULTICICLO

5.16. Sea un sistema procesador basado en MIPS con arquitectura multiciclo igual que el estudiado en la asignatura. El sistema se encuentra ejecutando el código que se adjunta.

Considere que en el ciclo de valor T, se acaba de ejecutar una determinada instrucción del código señalado. Con la información facilitada, debe poder identificar la instrucción o instrucciones que se ejecutarán en los ocho ciclos consecutivos siguientes del T+1 al T+8. El sistema tiene una única llamada a la subrutina *sub* desde la rutina principal *main*. Complete la tabla adjunta con los valores adecuados para las señales de control, e indique el valor de los registros que se hayan modificado durante la ejecución de los ciclos señalados. Complete toda la tabla, pero no añada más columnas aunque queden instrucciones sin ejecutar completamente.

Código

```
.text 0x0000
main: lw $s2, A($s0)
      jal sub
      sllv $s1, $s2, $s3
      …
      …
----------------------------
.text 0x0100
sub: beq $s2, $s3, eti2
      addi $s2, $0, 8
      jr $ra
-----------------------------
.data 0x2000
A:    0x00000008
```

Instrucción	**jr (1)**	**jr (2)**	**jr (3)**	**sllv (1)**	**sllv (2)**	**sllv (3)**	**sllv (4)**	**… (1)**
Ciclo	**T+1**	**T+2**	**T+3**	**T+4**	**T+5**	**T+6**	**T+7**	**T+8**
PC	**0x0108**	**0x010C**	**=**	**0x0008**	**0x000C**	**=**	**=**	**0x000C**
$s1	**0xA0A0**	**=**	**=**	**=**	**=**	**=**	**=**	**0x0080**
$s2	**0x0010**	**=**	**=**	**=**	**=**	**=**	**=**	**=**
$s3	**0x0003**	**=**	**=**	**=**	**=**	**=**	**=**	**=**
IorD	**0**	**X**	**X**	**0**	**X**	**X**	**X**	**0**
IRWrite	**1**	**0**	**0**	**1**	**0**	**0**	**0**	**1**
MemWrite	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
RegWrite	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**
MemtoReg	**X**	**X**	**X**	**X**	**X**	**X**	**0**	**X**
RegDst	**X**	**X**	**X**	**X**	**X**	**X**	**1**	**X**
ALUSrcA	**0**	**0**	**X**	**0**	**0**	**1**	**X**	**0**
$ALUSrcB_{(1:0)}$	**01**	**11**	**XX**	**01**	**11**	**00**	**XX**	**01**

ESTRUCTURA DE COMPUTADORES
PROBLEMAS DE LA UNIDAD 5.- EL PROCESADOR III: LA RUTA DE DATOS MULTICICLO

5.17. Sea un sistema procesador basado en MIPS con arquitectura multiciclo igual que el estudiado en la asignatura. El sistema se encuentra en el periodo T ejecutando el primer ciclo (captura) de una determinada instrucción, instrucción actual, del código que se adjunta. En este código, para el paso de parámetros entre rutinas se utilizan los registros específicos de MIPS ($ax y $vx).

Con la información facilitada de la tabla de control, debe poder identificar la instrucción actual y las dos instrucciones ejecutadas en los ciclos precedentes.

Código			

.text 0x0000
main: lw $s1, Num($0)
add $a0, $s1, $0
jal sub1
lw $s1, R($0)
fin: j fin
---------- 	
---------- 	
.data 0x2000
Num: 0x00000064
R: 0x00000000
	.text 0x0100
	sub1: addi $sp, $sp, -4
	sw $ra, 0($sp)
	addi $t1, $0, 100
	beq $a0, $t1, etiq
	jal mulx8
	j ret
	etiq: jal mulx4
	ret: lw $ra, 0($sp)
	addi $sp, $sp, 4
	sw $v0, R($0)
	jr $ra
		.text 0x0200
		mulx4: sll $v0, $a0, 2
		jr $ra
			.text 0x0300
			mulx8: sll $v0, $a0, 3
			jr $ra

Se pide:

Ciclo	**T-7**	**T-6**	**T-5**	**T-4**	**T-3**	**T-2**	**T-1**	**T**	**T+1**	**T+2**	**T+3**	**T+4**
PCWrite	1	0	0	0	1	0	1	1	**0**	**0**	**0**	**0**
IorD	0	X	X	1	0	X	X	0	**X**	**X**	**1**	**X**
IRWrite	1	0	0	0	1	0	0	1	**0**	**0**	**0**	**0**
MemWrite	0	0	0	1	0	0	0	0	**0**	**0**	**0**	**0**
RegWrite	0	0	0	0	0	0	0	0	**0**	**0**	**0**	**1**
MemtoReg	X	X	X	X	X	X	X	X	**X**	**X**	**X**	**1**
RegDst	X	X	X	X	X	X	X	X	**X**	**X**	**X**	**0**
ALUSrcA	0	0	1	X	0	0	X	0	**0**	**1**	**X**	**X**
ALUSrcB$_{(1:0)}$	01	11	10	XX	01	11	XX	01	**11**	**10**	**XX**	**XX**

a. Indique el valor en hexadecimal del registro **pc** en los ciclos T-1, T y T+1

La identificación de la instrucción actual se basa en:

1. En los ciclos T-3, T-2 y T-1 se ejecuta un salto (j, jal o jr), identificado por la activación de PCWrite en los ciclos primero y tercero.
2. En el ciclo T-4, se ejecuta una escritura en memoria (sw), identificada por la activación de MemWrite.
3. En consecuencia la instrucción actual sólo puede ser **lw $s1, R($0)** instrucción retorno de la subrutina sub1.

%pc (T-1) = 0x0000012C	**%pc (T) = 0x0000000C**	**%pc (T+1) = 0x00000010**

b. En la tabla adjunta, complete con los valores adecuados las señales de control necesarias para ejecutar completamente la instrucción actual (utilice las columnas que considere necesarias).

c. Complete el valor en hexadecimal del registro señalado y de la posición de memoria señalados, una vez ejecutada completamente la instrucción actual.

%ra = 0x0000000C

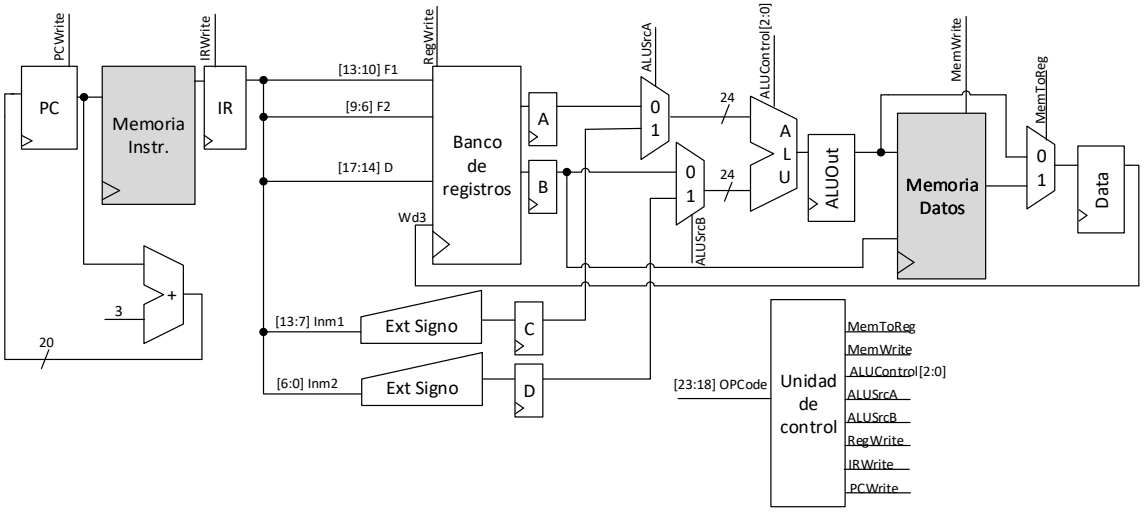
MEM[R] = 0x00000190

d. En la ruta de datos de MIPS, los registros intermedios A y B, se utilizan para registrar los operandos de la ALU en el caso que se necesiten y puede cambiar cada ciclo su valor. Sabiendo que la codificación de la instrucción incluida en el código **jr $ra, al final de la rutina mulx4**, es **0x03E00008**, se pide señalar el valor en hexadecimal de los registros A y B después de completar los dos últimos ciclos necesarios para ejecutar la citada instrucción, es decir un poco después del flanco de reloj correspondiente a los citados ciclos.

Ciclo 2º:	**A = 0x0000011C**	**B = 0x00000000**
Ciclo 3º:	**A = 0x0000011C**	**B = 0x00000000**

ESTRUCTURA DE COMPUTADORES
PROBLEMAS DE LA UNIDAD 5.- EL PROCESADOR III: LA RUTA DE DATOS MULTICICLO

5.18. En la figura que se adjunta se puede observar el esquemático de un procesador multiciclo distinto al MIPS. En cada ciclo de reloj sólo se ejecutará la ruta de datos comprendida entre registros (desde lectura en un registro hasta escritura en otro registro o en memoria). Se conocen tres tipos de instrucciones en las que, aparte de registro de destino (D), los operandos fuentes pueden ser: dos registros (F1, F2), dos datos inmediatos (Inm1, Inm2), o un registro (F1) y un dato inmediato (Inm2).



Se pide completar el cronograma con las señales de control necesarias, ciclo a ciclo, para ejecutar las siguientes instrucciones, indicando además qué instrucción se ejecuta en cada ciclo.

or $D, $F1, $F2 # OR lógica del contenido de los registros F1 y F2, dejando el resultado en el registro D

add $D, Inm1, Inm2 # Suma el contenido de los datos inmediatos Inm1 e Inm2, dejando el resultado en D

add $D, $F1, $Inm2 # Suma el contenido del registro F1 y el dato inm. Inm2, dejando el resultado en D

lw $D, $F1, $F2 # Carga desde memoria el contenido de la posición indicada por la suma de los registros # F1 y F2, dejando el dato leído en D.

Instrucción	or $D, $F1, $F2						add $D, Inm1, Inm2						lw $D, $F1, $F2						add $D, $F1, $Inm2				
Ciclo	**1**	**2**	**3**	**4**	**5**		**1**	**2**	**3**	**4**	**5**		**1**	**2**	**3**	**4**	**5**		**1**	**2**	**3**	**4**	**5**
PCWrite	1	0	0	0	0		1	0	0	0	0		1	0	0	0	0		1	0	0	0	0
IRWrite	1	0	0	0	0		1	0	0	0	0		1	0	0	0	0		1	0	0	0	0
RegWrite	0	0	0	0	1		0	0	0	0	1		0	0	0	0	1		0	0	0	0	1
ALUSrcA	X	X	0	X	X		X	X	1	X	X		X	X	0	X	X		X	X	0	X	X
ALUSrcB	X	X	0	X	X		X	X	1	X	X		X	X	0	X	X		X	X	1	X	X
MemWrite	0	0	0	0	0		0	0	0	0	0		0	0	0	0	0		0	0	0	0	0
MemToReg	X	X	X	0	X		X	X	X	0	X		X	X	X	1	X		X	X	X	0	X

ESTRUCTURA DE COMPUTADORES
PROBLEMAS DE LA UNIDAD 5.- EL PROCESADOR III: LA RUTA DE DATOS MULTICICLO

5.19. En el mismo circuito procesador del problema anterior, suponga que los registros del banco de registros se numeran $0, $1, $2, etc. Se ejecuta el siguiente código constituido por 2 instrucciones

add $2, $1, 5 # Suma el contenido del registro $1 y el inmediato 5, dejando el resultado en $2

lw $2, $1, 0 # Lee la dirección de memoria [$1]+0 y lo escribe en $2

Se pide completar ciclo a ciclo un cronograma de tiempo, desde el ciclo inicial T hasta el ciclo T+11, indicando el contenido de las señales de control y de los registros señalados.

Registros:	**PC**	**$1**	**$2**				
Control:	**PCWrite**	**IRWrite**	**RegWrite**	**ALUSrcA**	**ALUSrcB**	**MemWrite**	**MemToReg**

Se sabe que en el ciclo T, comienza la primera instrucción con los siguientes valores para los registros:

PC = 0x00; $1 = 0x03 y $2 = 0x02.

Suponga que el contenido inicial de todas las posiciones de la memoria de datos es 0x07. Utilice el valor XX, para señalar el contenido de los aquellos registros en los que dicho contenido sea desconocido).

Instrucción	**add $2, $1, 5**						**lw $2, $1, 0**					**?**	**?**
Ciclo	**T**	**T+1**	**T+2**	**T+3**	**T+4**		**T+5**	**T+6**	**T+7**	**T+8**	**T+9**	**T+10**	**T+11**
Registro PC	**0x0**	0x3	=	=	=		=	0x6	=	=	=	=	0x9
Registro $1	**0x3**	0x3	=	=	=		=	=	=	=	=	=	0x3
Registro $2	**0x2**	0x2	=	=	=		0x8	=	=	=	=	0x7	0x7
Registro ALUOut	**NP**	**NP**	**NP**	0x8	X		X	X	X	0x3	X	X	X
Registro Data	**NP**	**NP**	**NP**	**NP**	0x8		X	X	X	X	0x7	X	X
IRWrite	1	0	0	0	0		1	0	0	0	0	1	0
RegWrite	0	0	0	0	1		0	0	0	0	1	0	0
MemToReg	X	X	X	0	X		X	X	X	1	X	X	X

ESTRUCTURA DE COMPUTADORES
PROBLEMAS DE LA UNIDAD 5.- EL PROCESADOR III: LA RUTA DE DATOS MULTICICLO

5.20. Sea un sistema procesador basado en MIPS con arquitectura multiciclo igual que el estudiado en la asignatura. Se quiere ejecutar el código que se adjunta:

```
.text 0x0000
    lw $s2, 0x2000($s1)
    beq $s2, $s3, fin
    sw $s2, 0x2000($0)
fin: j fin
.data 0x2000
X: 0x0A
Y: 0x0C
```

Se pide completar la siguiente tabla con la evolución a lo largo de los nueve primeros ciclos de reloj de diversas señales y registros, sabiendo que en el ciclo 1 de reloj se está ejecutando el primer ciclo de la primera instrucción del programa.

Instrucción	**lw**	**lw**	**lw**	**lw**	**lw**	**beq**	**beq**	**beq**	**j**
Ciclo	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Registro pc	**0x00**	0x04	=	=	=	=	0x08	=	0x0C
Registro $s1	**0x04**	=	=	=	=	=	=	=	=
Registro $s2	**0x08**	=	=	=	=	0x0C	=	=	=
Registro $s3	**0x0C**	=	=	=	=	=	=	=	=
Registro A	X	X	0x04[1]	=	=	=	=	0x0C	=
Registro B	X	X	0x08[1]	=	=	=	0x0C[5]	0x0C	=
Registro ALUOut	X	0x0004[2]	0x8004[2]	0x2004	X	X	0x0008	0x000C[3]	0x0000[4]
RegWrite	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**
ALUSrcA	**0**	**0**	**1**	**X**	**X**	**0**	**0**	**1**	**0**
ALUSrcB$_{(1:0)}$	**01**	**11**	**10**	**XX**	**XX**	**01**	**11**	**00**	**01**

(1) En A y B siempre se leen los registros rs y rt respectivamente, independientemente de que se utilicen o no. Se leen durante ciclo 2, así que aparecen en el ciclo 3.

(2) La ALU siempre hace PC+4 durante ciclo 1 y el BTA durante ciclo 2, así que aparecen en el registro ALUOut en los ciclos 2 y 3 respectivamente.

(3) En esta instrucción beq el dato inmediato es 0x1, así que el BTA = (PC+4) + (SignImm x 4) = 0x000C, que es la dirección de la instrucción a la que se salta.

(4) En el tercer ciclo del beq se restan los datos en la ALU, que son iguales en este caso y por tanto la resta da 0x0000.

(5) En los dos primeros ciclos de cada instrucción se siguen escribiendo en A y B el contenido de los registros codificados como rs y rt de la instrucción anterior, debido a que el registro de instrucciones (Instr) de la instrucción actual no se actualiza hasta el ciclo 2. Como el contenido del registro $s2 (campo rt) tiene un nuevo valor (0x0C) en el ciclo 6, este valor aparece en el registro B, en ciclo 7 de la tabla.

ESTRUCTURA DE COMPUTADORES
PROBLEMAS DE LA UNIDAD 5.- EL PROCESADOR III: LA RUTA DE DATOS MULTICICLO

5.21. Sea un sistema procesador basado en MIPS con arquitectura multiciclo igual que el estudiado en clase. Se va a ejecutar el código que se adjunta.

Código
.text 0 lw $s1, 0x2000($s2) beq $s1, $s3, etiq1 j etiq2 .text 0x0100 etiq2: add $s3, $s1, $s2 etiq1: sw $s3, 0x2004($s2) etiq3: j etiq3

Contenido inicial de la memoria
.data 0x2000 A: 10 B: 20 C: 30

a) Se pide rellenar la tabla adjunta que contiene registros de propósito general, registros de la ruta de datos y señales de control. Se sabe que en el ciclo 1 se está ejecutando el primer ciclo de la primera instrucción, "lw $s1, 0x2000($s2)". Complete toda la información de la tabla que se pueda, pero no añada más columnas aunque queden instrucciones sin ejecutar. Los datos que no se puedan conocer indíquelos con una interrogante "?". En los registros puede poner "=" para indicar que queda igual que el ciclo anterior.

Instrucción	**lw**	**lw**	**lw**	**lw**	**lw**	**beq**	**beq**	**beq**	**sw**
Ciclo	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
$s1	**0**	**=**	**=**	**=**	**=**	**20**	**=**	**=**	**=**
$s2	**4**	**=**	**=**	**=**	**=**	**=**	**=**	**=**	**=**
$s3	**20**	**=**	**=**	**=**	**=**	**=**	**=**	**=**	**=**
Registro A	**?**	**?**	**4[1]**	**=**	**=**	**=**	**=**	**20**	**=**
Registro B	**?**	**?**	**0[1]**	**=**	**=**	**=**	**20[4]**	**=**	**=**
Registro ALUOut	**?**	**0x4[2]**	**0x8004[2]**	**0x2004**	**?**	**?**	**0x8[2]**	**0x104[2]**	**0[3]**
IorD	**0**	**X**	**X**	**1**	**X**	**0**	**X**	**X**	**0**
RegWrite	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**
MemToReg	**X**	**X**	**X**	**X**	**1**	**X**	**X**	**X**	**X**
PCEn	**1**	**0**	**0**	**0**	**0**	**1**	**0**	**1**	**1**

(1) En A y B siempre se leen los registros rs y rt respectivamente, independientemente de que se utilicen o no. Se leen durante ciclo 2, así que aparecen en el ciclo 3.

(2) La ALU siempre hace PC+4 durante ciclo 1 y el BTA durante ciclo 2, así que aparecen en el registro ALUOut en los ciclos 2 y 3 respectivamente.

(3) En la instrucción beq la ALU resta en ciclo 3, así que aparece el resultado de la resta al ciclo siguiente en ALUOut aunque no se utilice.

(4) En A y B siguen llegando los datos de la instrucción antigua durante los dos primeros ciclos de cada instrucción, pero en el ciclo 7 aparece un 20 en lugar del 0 porque el registro $s1 cambió en el ciclo 6.

ESTRUCTURA DE COMPUTADORES
PROBLEMAS DE LA UNIDAD 5.- EL PROCESADOR III: LA RUTA DE DATOS MULTICICLO

b) Rellene el contenido de la siguiente tabla en la que se pide el contenido del registro PC a lo largo de la ejecución del programa. Se sabe que en el ciclo 1 se está ejecutando el primer ciclo de la primera instrucción, "lw $s1, 0x2000($s2)". Complete toda la información de la tabla que se pueda, pero no añada más columnas aunque queden instrucciones sin ejecutar.

Instrucción	lw	lw	lw	lw	lw	beq	beq	beq	sw
Ciclo	1	2	3	4	5	6	7	8	9
PC	0x000	0x004	=	=	=	=	0x008	=	0x104

Instrucción	sw	sw	sw	j	j	j	j	j	j
Ciclo	10	11	12	13	14	15	16	17	18
PC	0x108	=	=	=	0x10C	=	0x108	0x10C	=

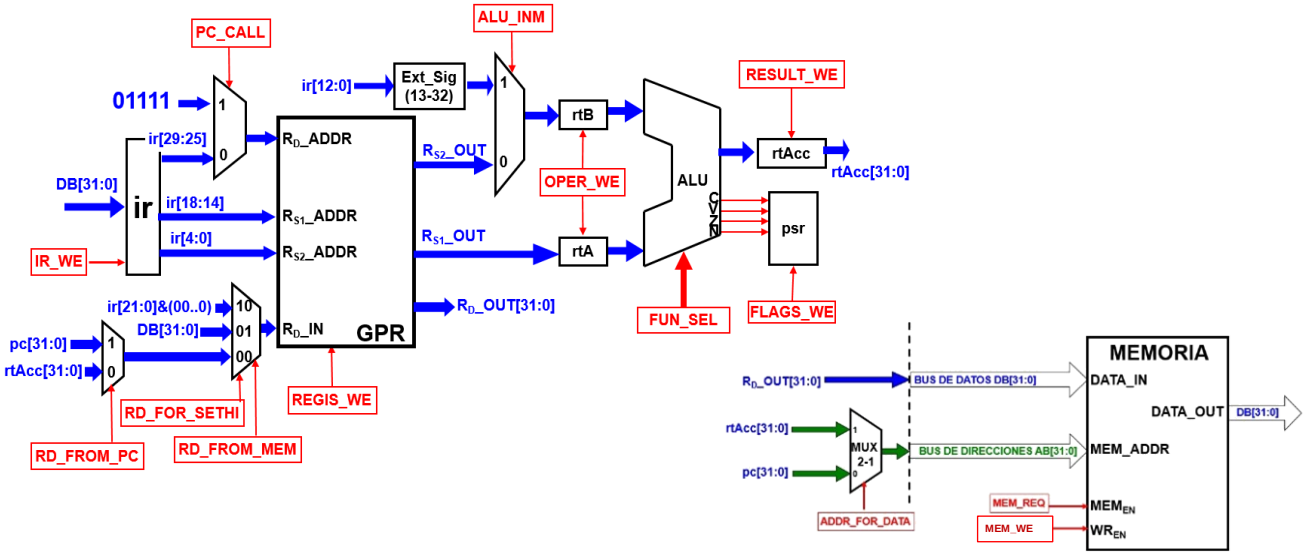
c) Complete la tabla adjunta a partir de la información que ya figura en dicha tabla. Además indique en el hueco qué instrucción se está ejecutando en el ciclo T.

Ciclo	T-1	T
PC	0x104	0x108
ALUSrcA	0	0
$ ALUSrcB	01	11
PCWrite	1	0
IRWrite	1	0
RegWrite	0	0

Instrucción
sw $s3, 0x2004($s2)

ESTRUCTURA DE COMPUTADORES
PROBLEMAS DE LA UNIDAD 5.- EL PROCESADOR III: LA RUTA DE DATOS MULTICICLO

5.22. En el esquema adjunto se muestra la ruta de datos para la arquitectura multiciclo del procesador ARC (A Risc Computer), incluyendo la ruta para el acceso a la memoria única de instrucciones y datos. En ARC el tamaño para las instrucciones y datos es de 32 bits. En el esquema se identifican, de izquierda a derecha de la imagen, el registro de instrucciones ir, el banco de registros GPR, los registros temporales rtA y rtB a la entrada de la ALU y el registro acumulador rtAcc a la salida de esta esta unidad. Las señales de control son las que aparecen enmarcadas en el esquema.



La siguiente instrucción de ARC hace la suma de dos operandos fuente y escribe el resultado en un registro: **addcc %r10, - 100, %r15**, (en MIPS esta instrucción es equivalente a: **addi $15, $10, - 100**). Con la información de la que se dispone, se pide:

a) La instrucción máquina equivalente. Seleccione agrupando en bits los campos identificados y marque con 'x' los bits para los que no se dispone de información y que corresponden con el campo o con los campos destinados al código de operación.

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Campo	OP		R_D					OP						R_{S1}					OP	Inm												
Valor	X	X	0	1	1	1	1	X	X	X	X	X	X	0	1	0	1	0	X	1	1	1	1	1	1	0	0	1	1	1	0	0

b) La ruta de datos multiciclo de ARC utiliza 4 ciclos para ejecutar esta instrucción, complete la tabla adjunta indicando el valor para las señales de control señaladas. Las señales terminadas en _WE, habilitan la escritura en el circuito síncrono correspondiente. El resto son señales de control cuyo valor debe saber identificar para ejecutar la instrucción demandada. Complete la tabla con los valores '0', '1' ó 'X' según corresponda a cada ciclo. **Nota**: Señalar que en el esquema se ha omitido la ruta de actualización para el PC y por tanto no hay que actualizarlo.

Señal	Ciclo 1	Ciclo 2	Ciclo 3	Ciclo 4
MEM_WE	0	0	0	0
ADDR_FOR_DATA	0	X	X	X
IR_WE	1	0	0	0
PC_CALL	X	X	X	0
ALU_INM	X	1	X	X
OPER_WE	0*	1	0*	0*
RESULT_WE	0*	0*	1	0*
RD_FROM_PC	X	X	X	0
RD_FOR_SETHI	X	X	X	0
RD_FROM_MEM	X	X	X	0
REGIS_WE	0	0	0	1

* Se considera "0" como la mejor solución, pero se admite "X, don't care" ya que no afecta al resultado.

ESTRUCTURA DE COMPUTADORES
PROBLEMAS DE LA UNIDAD 5.- EL PROCESADOR III: LA RUTA DE DATOS MULTICICLO

5.23. En una arquitectura MIPS multiciclo se ejecuta un programa en el cual de forma consecutiva se ejecutan las dos instrucciones es siguientes: **sllv $s1, $s1, $s1** y **sw $s1, 0x0040($s1**). Se pide:

a) En el ciclo T, la instrucción **sllv $s1, $s1**, $s1, se está ejecutando en su segundo ciclo. Complete para los ciclos señalados de la tabla adjunta, sólo las columnas en blanco, el valor de los registros indicados. Si no tiene información para conocer el valor pedido, escriba NA.

	2ºsllv	3ºsllv	4ºsllv	1ºsw	2ºsw	3ºsw	4ºsw	1º¿?	2º¿?
Ciclos Registros	**T**		**T+2**		**T+4**	**T+5**			**T+8**
PC	0x0104		**=**		**0x0108**	**=**			**0x010C**
Instr	0x02318804		**=**		**0xAE310040**	**=**			**NA**
$s1	0x0008		**=**		**0x0800**	**=**			**0x0800**
A	NA		**0x0008**		**=**	**0x0800**			**0x0800**
B	NA		**0x0008**		**=**	**0x0800**			**0x0800**
ALUOut	NA		**0x0800**		**0x0108**	**0x0208**			**0x010C**

b) Complete para los ciclos señalados en la tabla adjunta, el valor de las señales de control indicadas.

	2ºsllv	3ºsllv	4ºsllv	1ºsw	2ºsw	3ºsw	4ºsw	1º¿?
Ciclos Control	**T**		**T+2**			**T+5**	**T+6**	**T+7**
IorD	**X**		**X**			**X**	**1**	**0**
IRWrite	**0**		**0**			**0**	**0**	**1**
MemWrite	**0**		**0**			**0**	**1**	**0**
MemtoReg	**X**		**0**			**X**	**X**	**X**
RegWrite	**0**		**1**			**0**	**0**	**0**
PC_Write	**0**		**0**			**0**	**0**	**1**
ALUScrA	**0**		**X**			**1**	**X**	**0**
ALUScrB$_{(1:0)}$	**11**		**XX**			**10**	**XX**	**01**

c) En la arquitectura de MIPS multiciclo estudiada en clase, suponer las siguientes latencias:

MEM	Mux	ALU	BancoReg	Ext_signo	Despl. x 2
375 ps	25 ps	150 ps	200 ps	20 ps	2 ps

Si se desprecian los tiempos para el setup y delay (T_{SETUP}, T_{DELAY}) de los registros implicados, se pide, **justificando necesaria y brevemente la respuesta**, calcular la frecuencia máxima de operación y el tiempo de ejecución para cada una de las dos instrucciones señaladas en el enunciado.

De las 5 fases de ejecución vistas en MIPS, la más lenta y en consecuencia la fase que marca el ciclo de reloj es la primera o captura de instrucción, da igual la instrucción que sea. En el esquema dado, en esta fase interviene un multiplexor para seleccionar la dirección desde el PC y una operación de lectura en memoria. Es decir $T_{MAX} = T_{CICLO}$ = 25 ps + 375 ps = 400 ps.

En consecuencia la frecuencia máxima de operación es de **$F = 1/T_{CICLO}$ = 2,5 GHz**

Como tanto sllv como sw tardan 4 ciclos, ambas tardarán lo mismo es decir 4 x 400 ps = **1.600 ps**