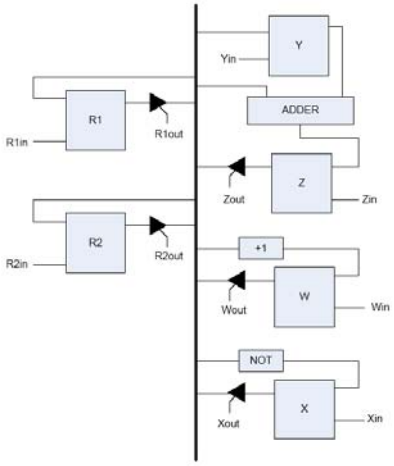
ESTRUCTURA DE COMPUTADORES
PROBLEMAS DE LA UNIDAD 5.- EL PROCESADOR III: LA RUTA DE DATOS MULTICICLO

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:

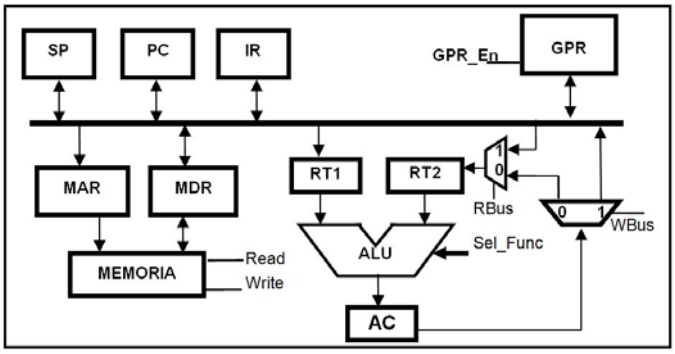
a) R1 ← R1 - R2 + 1

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'.

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.

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 para MIPS, encuentre el código máquina para la instrucción del programa anterior "**bne $s1, $0, lab"**

b) Señale el código ensamblador que corresponde al código dado en hexadecimal. Teniendo en cuenta los operandos, señale brevemente la función que esta instrucción realiza.

c) Para cada una de las líneas de control señaladas, señale los valores que en cada ciclo de reloj correspondan, para ejecutar completamente las dos primeras instrucciones del programa dado lw $t2, X($0) y sllv $t3, $t2, $t2

Señales de Control										
IorD	IRWrite	RegDst	MemWrite	MentoReg	RegWrite	PC_Write	Branch	ALUScrA	$ALUScrB_{(1:0)}$	$PCSrc_{(1:0)}$

Señale los valores de los registros $pc, $t1, $t2 y $t3, una vez ejecutadas completamente las tres primeras instrucciones, es decir, inmediatamente antes del comienzo de la ejecución de la 4ª: "and $t4, $t1, $t2".

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

a) Indicar en una tabla, el valor para cada uno de los ciclos de ejecución el valor de las señales de control indicadas para ejecutar las instrucciones: sw $t1, 0x2404($0) y beq $s1, $t1, 0x8000.

Señales de Control										
IorD	IRWrite	RegDst	MemWrite	MentoReg	RegWrite	PC_Write	ALUScrA	$ALUScrB_{(1:0)}$	$PCSrc_{(1:0)}$	$ALUControl_{(2:0)}$

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')

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. Se pide:

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

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.

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**.

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.

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 del valor de los registros y el de las señales de control indicados, 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**.

```
        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:
```

Registros:				
$pc	$s1	$s2	$s3	$ra

Datos iniciales: $pc = 0x00, $s1 = 0x08; $s2 = 0x040; $s3 = $s4 = 0x00
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.

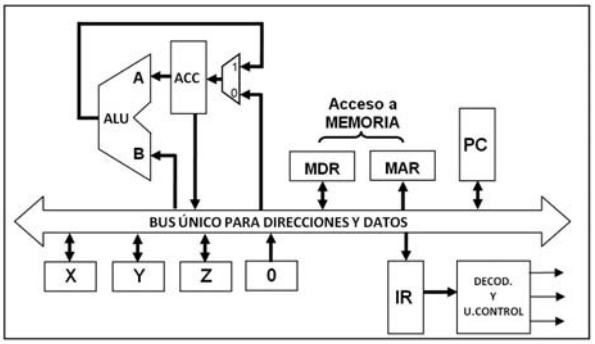
Señales de Control										
IorD	IRWrite	RegDst	MemWrite	MentoReg	RegWrite	PC_Write	ALUScrA	$ALUScrB_{(1:0)}$	$PCSrc_{(1:0)}$	$ALUControl_{(2:0)}$

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

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.

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:

Código

```
    .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 en una tabla con la evolución del reloj, el valor de los registros y de las señales de control indicadas, sabiendo que en el ciclo 1 de reloj se está ejecutando el primer ciclo de la primera instrucción del programa.

Registros:				
$pc	$s1	$s2	$s3	$ra

Datos iniciales: $s1 = 0x0A; $s2 = 0x02; $s3 = BF y $ra = 0x00

Señales de Control										
IorD	IRWrite	RegDst	MemWrite	MentoReg	RegWrite	PC_Write	Branch	ALUScrA	$ALUScrB_{(1:0)}$	$PCSrc_{(1:0)}$

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
```

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). Señale el contenido final de las posiciones de memoria indicadas (A, B y C).

Registros:				
$pc	$s1	$s2	$s3	$ra

Datos iniciales: $s1 = 0xA4; $s2 = 0x02; $s3 = 0x4C y $ra = 0x00

Señales de Control										
IorD	IRWrite	RegDst	MemWrite	MentoReg	RegWrite	PC_Write	Branch	ALUScrA	$ALUScrB_{(1:0)}$	$PCSrc_{(1:0)}$

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:
Se pide construir una 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.

Código

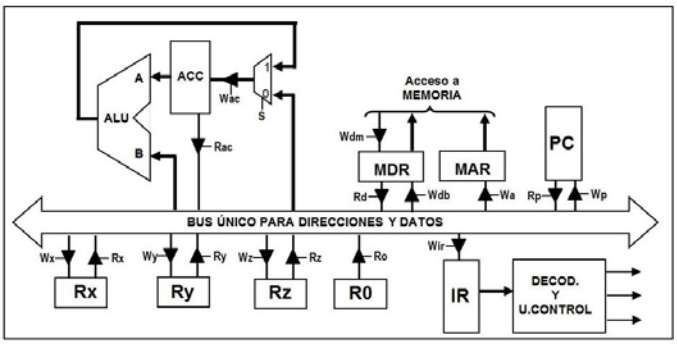
```
        .text 0x0000
          j eti
          beq $s1,$s3, fin
eti:    sw $s2, B($0)
fin:    j fin
        .data 0x2000
A:      0x01
B:      0x02
```

Registros/ Memoria					
$pc	$s1	$s2	$s3	A	B

Datos iniciales: $pc = 0x00, $s1 = 0x08; $s2 = 0x09; $s3 = 0x0A

Señales de Control										
IorD	MemWrite	IRWrite	RegDst	MentoReg	RegWrite	$PCSrc_{(1:0)}$	PC_Write	Branch	ALUScrA	$ALUScrB_{(1:0)}$

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:

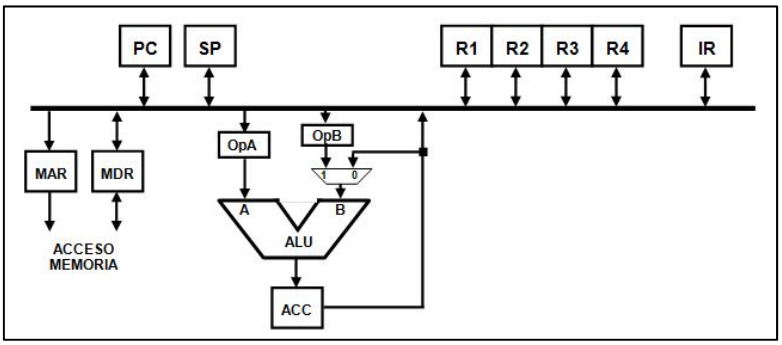
a1) LOAD X, Y ==> (MEM [Y] => X)

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')

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: 4 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.

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.

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
```

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.

Registros/ Memoria					
$pc	$s1	$s2	$s3	A	B

Datos iniciales (T+1): $pc=0x108, $s1=0xA0A0; $s2=0x0010; $s3=0x0003

Señales de Control							
IorD	IRWrite	MemWrite	RegWrite	MentoReg	RegDst	ALUScrA	ALUScrB$_{(1:0)}$

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				
IorD	0	X	X	1	0	X	X	0				
IRWrite	1	0	0	0	1	0	0	1				
MemWrite	0	0	0	1	0	0	0	0				
RegWrite	0	0	0	0	0	0	0	0				
MemtoReg	X	X	X	X	X	X	X	X				
RegDst	X	X	X	X	X	X	X	X				
ALUSrcA	0	0	1	X	0	0	1	0				
ALUSrcB$_{(1:0)}$	01	11	10	XX	01	11	XX	01				

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

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 **%ra** y de la posición de memoria dada por la etiqueta **R**, una vez ejecutada completamente la instrucción actual.

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.

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).
Dada las cuatro instrucciones:

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

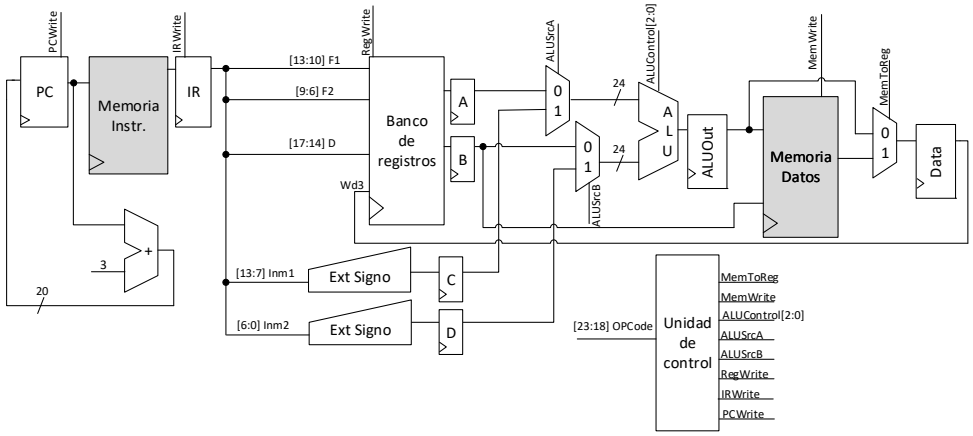
add $D, Inm1, Inm2 # Suma el contenido de los datos inmediatos Inm1 e Inm2, dejando el resultado en D

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.

Se pide completar ciclo a ciclo el cronograma con las señales de control a continuación señaladas,

PCWrite	IRWrite	RegWrite	ALUSrcA	ALUSrcB	MemWrite	MemToReg



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).

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
```

Datos iniciales: $s1 = 0x04; $s2 = 0x08; $s3 = 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.

Registros:	**PC**	**$s1**	**$s2**	**$s3**	**A**	**B**	**ALUOut**
Control:	**RegWrite**	**ALUSrcA**	**ALUSrcB$_{(1:0)}$**				

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 sabe que en el ciclo actual T se está ejecutando el primer ciclo de la primera instrucción, “lw $s1, 0x2000($s2)” y el contenido de los registros $s1 = 0; $s2 = 4 ty $s3 = 20. Escriba una tabla que incluya el ciclo actual T y los ochos ciclos siguientes, hasta T+8, indicando los valores de los registros y de las señales de control indicadas. Los datos que no se puedan conocer indíquelos con una interrogante “?”.

Registros:	**$s1**	**$s2**	**$s3**	**A**	**B**	**ALUOut**
Control:	**IorD**	**RegWrite**	**MemToReg**	**PCEn**		

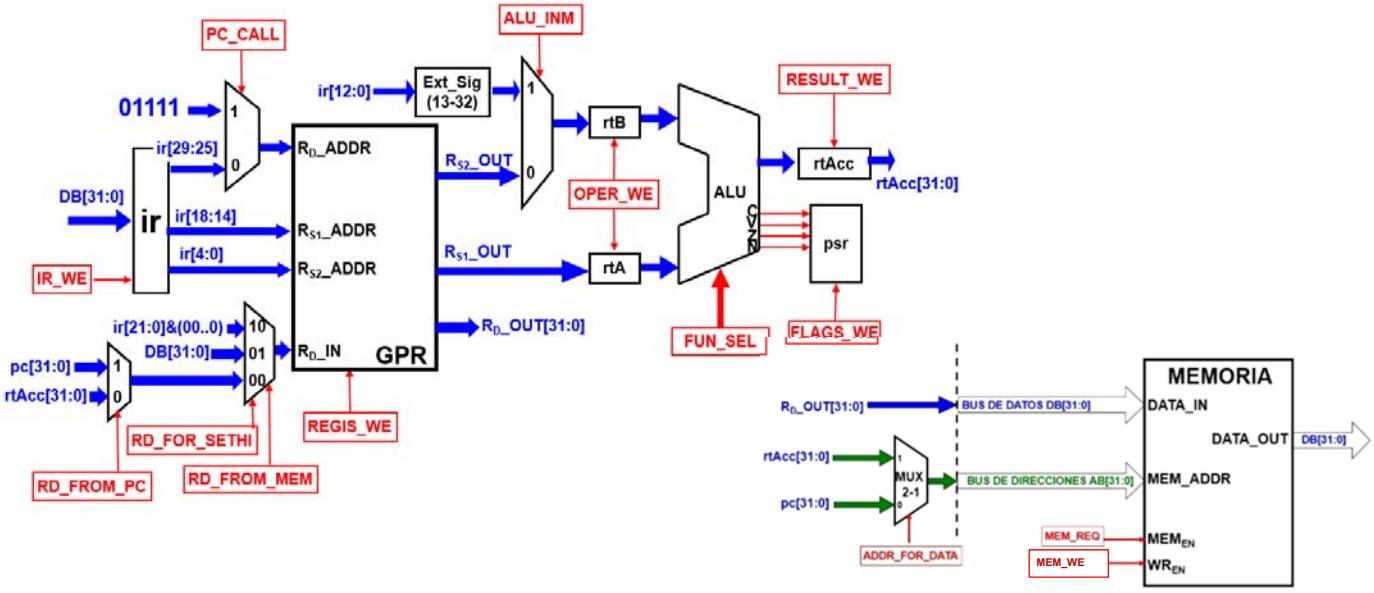
b. Se sabe que en el ciclo T se está ejecutando el primer ciclo de la primera instrucción, “lw $s1, 0x2000($s2)”. Señale el contenido del registro PC a lo largo de la ejecución del programa, desde T hasta T+17.

c. En un cierto ciclo T, valor distinto del anterior, el valor del registro PC y el de algunas señales de control son las que se indican. Calcule la instrucción que ahora se está ejecutando y complete las señales de control para el ciclo T-1.

Ciclo T	**PC**	**ALUSrcA**	**ALUSrcA**	**PCWrite**	**IRWrite**	**RegWrite**
Control:	**0x108**	**0**	**11**	**0**	**0**	**1**

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.

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.

Señales	MEM_WE	ADDR_FOR_DATA	IR_WE	PC_CALL	ALU_INM	OPER_WE	RESULT_WE
	RD_FROM_PC	RD_FOR_SETHI	RD_FROM_MEM	REGIS_WE			

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, y los registros señalados contienen los valores PC = 0x0104; Instr: 0x02318804; $s1 = 0x0008; A = B = ALUOut = NA. Complete para los ciclos T+2, T+4 T+5 y T+8 el valor de los mismos registros indicados. Si no tiene información para conocer el valor pedido, escriba NA.

b) Complete para los ciclos T, T+2, T+5, T+6, T+7, el valor de las señales de control IorD; IRWrite¸ MemWrite;

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.