SIMULACRO DE EXAMEN. PARCIAL 3.

P1. Se quiere ejecutar el código adjunto escrito para MIPS, en una implementación multiciclo, tal y como la estudiada en la asignatura. En la tabla facilitada, se indica las señales de control que corresponden al último ciclo (T) de la instrucción que se acaba de ejecutar.

```
.text 0x0000
    j    main
.text 0x0100
main: lw    $t0, A($0)
    lw    $t1, B($0)
    sll    $t0, $t0, 4
    beq   $t1, $t0, def
    sll   $t0, $t0, 2
    add   $t2, $t0, $sp
is0: sw    $0, 28($sp)
    j    done
 is2: addi  $t0, $0, 1
    sw    $t0, 32($sp)
    j    done
```

```
.text 0x200
def: addi  $t0, $0, -1
    sw    $t0, 28($sp)
    j    done
done:
```

```
.data 0x2000
A: 0x00FF
B: 0x0FF0
```

Con los datos de los que se dispone y en función de los valores de los registros implicados, se pide:

a. Señale, el valor en hexadecimal del registro $pc para **el ciclo actual T**.

La señal de control Branch = '1' en el ciclo T, implica que la instrucción es de tipo Branch. En este caso la única en el código es "beq $t1, $t0, def", por tanto en el ciclo actual T, el registro $pc señala a la siguiente instrucción "sll $t0, $t0, 2".

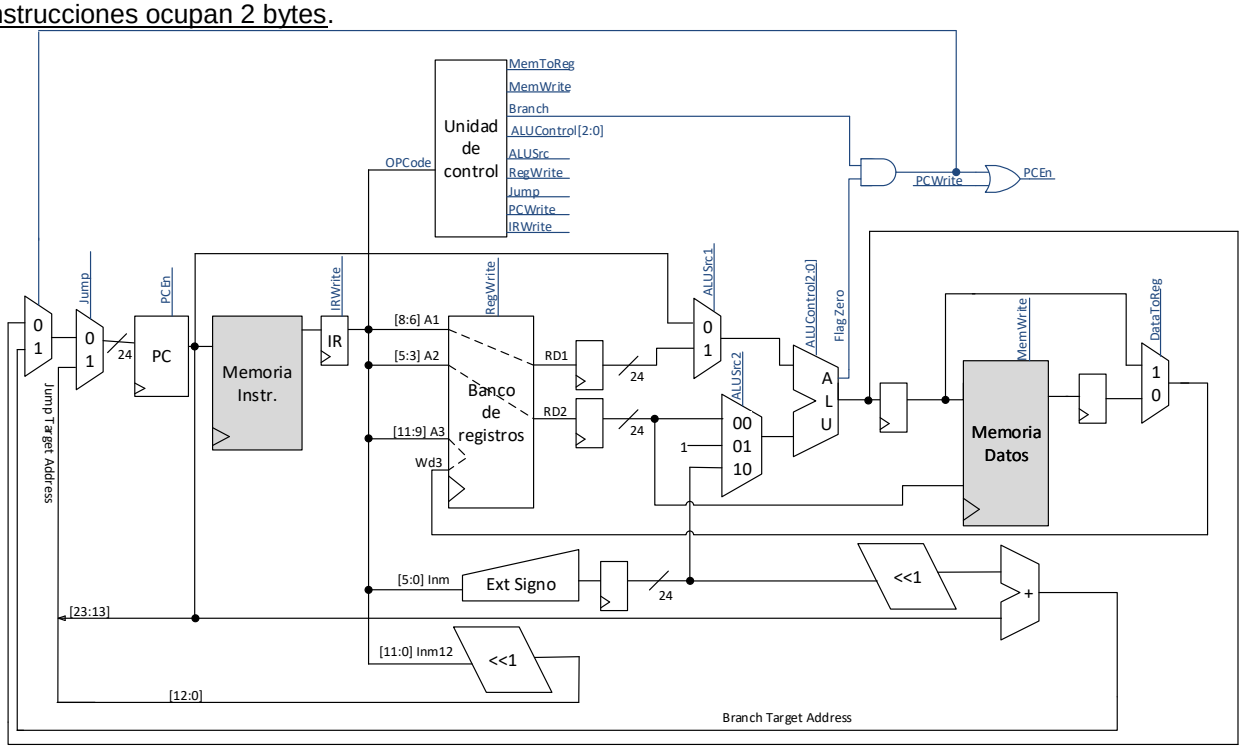
$pc (ciclo T) = 0x00000110

b. Complete la tabla para los once ciclos siguientes, del T+1 al T+11.
Al ejecutar la instrucción "beq $t1, $t0, def" el salto es efectivo, ya que se $t1 = $t0. Por tanto la siguiente instrucción a ejecutar en T+1 será "addi $t0, $0, -1"

	T	T+1	T+2	T+3	T+4	T+5	T+6	T+7	T+8	T+9	T+10	T+11
		addi				sw				j		
IorD	X	0	X	X	X	0	X	X	1	0	X	X
IRWrite	0	1	0	0	0	1	0	0	0	1	0	0
Branch	1	0	0	0	0	0	0	0	0	0	0	0
MemWrite	0	0	0	0	0	0	0	0	1	0	0	0
MemtoReg	X	X	X	X	0	X	X	X	X	X	X	X
RegDest	X	X	X	X	0	X	X	X	X	X	X	X
RegWrite	0	0	0	0	1	0	0	0	0	0	0	0
PC_Write	0	1	0	0	0	1	0	0	0	1	0	1
ALUScrA	0	0	0	1	X	0	0	1	X	0	0	X
$ALUScrB_{(1:0)}$	00	01	11	10	XX	01	11	10	XX	01	11	XX

SIMULACRO DE EXAMEN. PARCIAL 3.

P2. 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, por ejemplo, desde PC hasta IR). Se conocen dos tipos de instrucciones en las que, aparte de registro de destino (A3), los operandos fuentes pueden ser: dos registros (A1, A2), o un registro (A1) y un dato inmediato (Inm). Además, el procesador es capaz de ejecutar saltos incondicionales de tipo Jump, y condicionales de tipo Branch. Se sabe que todas las instrucciones ocupan 2 bytes.



a) Se pide completar el cronograma con las señales de control necesarias, ciclo a ciclo, para ejecutar las siguientes instrucciones. Si considera que una instrucción dura menos de 5 ciclos, no rellene los ciclos no utilizados.

xor $A3, $A1, $A2 # XOR lógica del contenido de los registros A1 y A2, dejando el resultado en el registro A3.

lw $A3, $A1, Inm # Lee la posición de memoria indicada por la suma del registro A1 y el dato inmediato Inm # extendido en signo a 24 bits, y el dato leído lo guarda en el registro $A3.

Instrucción	**xor**					**lw**				
Ciclo	**1**	**2**	**3**	**4**	**5**	**1**	**2**	**3**	**4**	**5**
PCWrite	**1**	**1**	**0**	**0**		**1**	**1**	**0**	**0**	**0**
IRWrite	**1**	**0**	**0**	**0**		**1**	**0**	**0**	**0**	**0**
RegWrite	**0**	**0**	**0**	**1**		**0**	**0**	**0**	**0**	**1**
MemWrite	**0**	**0**	**0**	**0**		**0**	**0**	**0**	**0**	**0**
Branch	**0**	**0**	**0**	**0**		**0**	**0**	**0**	**0**	**0**
Jump	**0**	**0**	**0**(1)	**0**(1)		**0**	**0**	**0**(1)	**0**(1)	**0**(1)
ALUSrc1	**0**	**0**	**1**	**X**		**0**	**0**	**1**	**X**	**X**
ALUSrc2	**01**	**01**	**00**	**XX**		**01**	**01**	**10**	**XX**	**XX**
DataToReg	**X**	**X**	**X**	**1**		**X**	**X**	**X**	**X**	**0**

(1) Aunque Jump='1' sólo el tercer ciclo de "j", en los casos señalados se considera válida la solución Jump='X', ya que en los casos señalados no está habilitada la escritura en el registro PC.

SIMULACRO DE EXAMEN. PARCIAL 3.

P3. Se desea diseñar un sistema electrónico basado en un procesador, con un bloque RAM otro ROM y varios bloques destinados a dispositivos periféricos. Se conoce que el procesador es capaz de manejar un bus de direcciones de 20 bits. Se ofrece su mapa de direcciones incompleto.

Componente	Bit Selección	Capacidad	Dirección inicial	Dirección final
ROM	CS1	64 kbytes	0x40000	0x4FFFF
Periférico 1 (P1)	$\overline{CS2}$	128 kbytes	0x00000	0x1FFFF
Periférico 2 (P2)	CS3	256 kbytes	0x80000	0xBFFFF
RAM	CS4	256 kbytes	0xC0000	0xFFFFF

a. Con estos datos complete la tabla con el mapa de direcciones adjunto teniendo en cuenta que todos los bloques deben estar alineados.

b. Indique las ecuaciones de los bits de selección de los tres bloques periféricos CS1, $\overline{CS2}$ y CS4**,** utilizando el **mapeo completo** de direcciones.

c. Indique las ecuaciones más reducidas posibles (**mapeo incompleto**) para los bits de selección $\overline{CS2}$ y CS3, entendiendo que deben ser ecuaciones válidas para seleccionar el mapa completo.

Nota: Para referirse a los bits de direcciones utilice la notación A_0, A_1, A_2….

b)

CS1 = $\overline{A_{19}} \cdot A_{18} \cdot \overline{A_{17}} \cdot \overline{A_{16}}$
$\overline{CS2} = A_{19} + A_{18} + A_{17}$
CS4 = $A_{19} \cdot A_{18}$

c)

$\overline{CS2} = A_{19} + A_{18}$
CS3 = $A_{19} \cdot \overline{A_{18}}$