COMPUTER STRUCTURE
UNIT 4.- PROCESSOR II: DESIGN AND CONTROL OF THE DATAPATH. SINGLE-CYCLE

4.1. En la arquitectura de MIPS de ciclo único estudiada en clase, hay identificados varios elementos de hardware diferentes. Dadas las instrucciones:

1. `add $s1, $s2, $s2` y **2.** `lw $t1, Offset($t2)`

Se quiere saber para cada una:

a. ¿Cuáles son los valores de las señales de control generadas?
b. ¿Qué recursos o elementos hardware (excluyendo los multiplexores y el registro PC) hacen algo útil para esta instrucción?

Solution:

a)	Jump	MemtoReg	MemWrite	Branch	ALUControl	ALUSrc	RegDst	RegWrite
1.	**0**	**0**	**0**	**0**	**010**	**0**	**1**	**1**
2.	**0**	**1**	**0**	**0**	**010**	**1**	**0**	**1**

b)

1. Todos excepto la extensión de signo, la memoria de datos y los sumadores en la ruta del PC para saltos, condicionales e incondicionales.

2. Todos excepto los sumadores en la ruta del PC para saltos condicionales e incondicionales.

4.2. The different elements of the processor have their own delay (delay: time from a change in the input to a change in the output). The critical path for an instruction is given by adding the delays in the longest path. For the single-cycle MIPS architecture, consider the following delays:

I-MEM	Add	Mux	ALU	RegFile	D-MEM	Control	Sign-ext	ShiftLeft2	And
500 ps	150 ps	30 ps	180 ps	220 ps	1000 ps	65 ps	90 ps	20 ps	20 ps

a. Which is the critical path for an “and” instruction? Which would be the clock cycle if all the instructions were ALU instructions (“and”, “add”, etc)?
b. Which is the critical path for a “lw” instruction? Which would be the clock cycle if all the instructions were “lw” instructions?
c. Which is the critical path for a “beq” instruction?
d. Which would be the clock cycle if the instructions possible instructions are “and”, “beq” and “lw”?

Solution:

a) Critical path for **and** is:

(read instr, 500), (control, 65 // read RegFile, 220), (Mux, 30), (ALU,180), (Mux, 30)

T_{CRIT1} = 500 + 220 + 30 + 180 + 30 = 960 ps (**critical path**)

In parallel, the PC is updated, but it is a shorter path: (Add+4, 150), (Mux, 30), (Mux, 30).

T_{CRIT2} = 150 + 30 +30 = 210 ps

The clock cycle must be equal or greater than the longest path (critical path) $T_{CLOCK} \geq 960$ ps, Freq ≤ 1.04 GHz

b) Critical path for **lw** is:

(read instr, 500), (Control, 65 // read RegFile, 220 // sign-extend and Mux, 90+30), (ALU, 180), (read D-mem, 1000), (Mux, 30)

T_{CRIT1} = 500 + 220 + 180 + 1000 + 30 = 1930 ps (**critical path**)

In parallel, the PC is updated, but it is a shorter path: (Add+4, 150), (Mux, 30), (Mux, 30).

T_{CRIT2} = 150 + 30 +30 = 210 ps

The clock cycle must be equal or greater than the longest path (critical path) $T_{CLOCK} \geq 1930$ ps, Freq ≤ 518 MHz

c) Critical path for **beq** is:

(read instr, 500), (control, 65 // read RegFile, 220), (Mux, 30), (ALU, 180), (And with Zero, 20), (Mux, 30), (Mux, 30)

T_{CRIT1} = 500 + 220 + 30 + 180 + 20 + 30 + 30 = 1010 ps (**critical path**)

In parallel, the PC is updated: (Add+4, 150 // Read instr, 500; SignExt, 90; ShiftLeft2, 20), (Add in Branch, 150), (Mux, 30), (Mux, 30).

T_{CRIT2} = 500 + 90 + 20 + 150 + 30 + 30 = 820 ps

d) The clock cycle is equal to the longest critical path of all the instructions. The longest is “lw”, so:

The clock cycle must be equal or greater than $T_{CLOCK} \geq 1930$ ps, Freq ≤ 518 MHz

COMPUTER STRUCTURE
UNIT 4.- PROCESSOR II: DESIGN AND CONTROL OF THE DATAPATH. SINGLE-CYCLE

4.3. Starting from the single-cycle MIPS architecture studied in class, we want to include three new instructions (independently from each other):

1. `sll $rd, $rt, despl` # rd <= rt << despl
2. `jal address` # PC <= (PC+4)[31:28] & address & "00"
3. `add3 $rd, $rs, $rt, $rx` # rd <= rt + rs+ rx

a. Which are the blocks (if any) that can be used for the new instructions?
b. Which are the new blocks (if any) that are needed for each instruction?
c. Which are the new control signals (if any) that are needed for each instruction?

Solution:

1a) This instruction uses I-MEM, the register file (one read port and the write port) and the ALU (which has to be able to do the shift operation).

1b) Despl is the field “shamt” in the R-type instructions, bits [10:6] de Instr. This field has to go to the ALU through the first operand, so a new multiplexer for the first operand of the ALU. This multiplexer will chose among [rs], connected to RD1, or shamt, which is composed by the [10:6] bits of Instr and 27 zeros in the left to have 32 bits in total.

1c) The new multiplexer needs a control signal. We will name it “shift”. Furthermore, ALUControl[2:0] will have the code used for left shifting in this instruction.

2a) This instruction uses I-MEM, the register file (the write port) and the logic for generating JTA, already used for instruction “j”.

2b) Two new multiplexers are needed. One for the register file input A3, in order to connect the address of register $ra ("11111") to be fed into A3. The second register will be in the register file input WD3, in order to connect PC+4.

2c) A new control signal can control both multiplexers at the same time, using the new connections when at ‘1’ and letting the old ones when at ‘0’.

3a) This instruction uses I-MEM, the register file (both read ports and the write port) and the ALU (which may be modified).

3b) The number of read ports in the register file must be increased from 2 to 3. The address of the third source register can be mapped to the field “shamt”, which would be unused otherwise in this instruction. For completing the addition, a new Adder block can be added to the output of the ALU, or all the ALU can be modified in order to have three source operands.

3c) The control signals would be the same, but a new value for ALUControl[2:0] should be chosen for doing the three operands addition.

COMPUTER STRUCTURE
UNIT 4.- PROCESSOR II: DESIGN AND CONTROL OF THE DATAPATH. SINGLE-CYCLE

4.4. Dada la arquitectura uniciclo para MIPS estudiada en clase, suponer que una de las siguientes señales de control funciona mal: i**)** *RegWrite y* ii**)** *MemWrite*
a) Suponer que siempre vale '0' con independencia del valor que se quiera aplicar
b) Suponer que siempre vale '1' con independencia del valor que se quiera aplicar
Explicando en cada caso y para cada señal el motivo, ¿qué instrucciones del set estudiado funcionarán mal?

Solution:

a) Si *RegWrite* siempre vale '0' afectará a todas las instrucciones que deban escribir en el banco de registros a saber:

* Todas las que operan con la ALU, **add, addi, and, etc**.
* La instrucción **lw**.
* La instrucción **jal** (no incluida en clase)

Si *MemWrite* siempre vale '0' afectará a la única instrucción para la que debe valer '1' que es **sw**

b) Si *RegWrite* siempre vale '1' afectará a todas las instrucciones, excepto a las que deben escribir en el banco de registros, señaladas en el apartado a)

Si *MemWrite* siempre vale '1' afectará a todas las instrucciones excepto **sw,** ya que se modifican posiciones de memoria no deseadas.

COMPUTER STRUCTURE
UNIT 4.- PROCESSOR II: DESIGN AND CONTROL OF THE DATAPATH. SINGLE-CYCLE

4.5. The machine code instructions 0x8C430010 and 0x1023000C are going to be executed in a single-cycle MIPS. All memory addresses have the value 0x0FF and the initial content of the registers is shown in the following table:

$0	$at	$v0	$v1	$a0	$a1	$a2	$t0	$t4	$ra
0	-16	-2	-3	4	-10	-6	-1	8	-4

For each of the previous instructions answer the following questions:

a. Which is the output of the sign-extend block and the "Shift.Left.2" block in the unconditional jump datapath?
b. Which is the binary value of the ALU control signal ALUControl[2:0]?
c. Which is the new value of PC after the instruction?
d. Write the hexadecimal value of the output of each multiplexer during the execution. If the value can not be known, write X.
e. Write the hexadecimal values of the ALU inputs and the inputs of both adding blocks.
f. Write the values of all the inputs to the register file. Write it in binary for the 5-bit inputs and in hexadecimal for the 32-bit inputs.

Solution:

a) 0x8C430010 => lw $v1, 16($v0)
* Output of the sign-extend block (32b): 0x00000010
* Output of the "Shift.Left.2" block in the unconditional jump datapath (28b): 0x10C0040

0x1023000C => beq $at, $v1, 0x000C
* Output of the sign-extend block (32b): 0x0000000C
* Output of the "Shift.Left.2" block in the unconditional jump datapath (28b): 0x08C0030

b) 0x8C430010 => lw $v1, 16($v0)
* ALUControl[2:0] = "010" (add, in a lw the ALU is adding for getting the address)

0x1023000C => beq $at, $v1, 0x000C
* ALUControl[2:0] = "110" (subtract, in a beq the ALU subtracts for knowing if the operands are equal)

c) 0x8C430010 => lw $v1, 16($v0)
* PC <= PC + 4 (lw does not change the control flow, so next instruction)

0x1023000C => beq $at, $v1, 0x000C
* PC <= PC + 4 (beq $at, $v1, 0x000C, branches to BTA if $at = $v1, but 1≠ 3, so no branch). If it had branched, BTA = (PC+4) + 0x00000030

d)

	RegDst_Mux	ALU_Src_Mux	MemToReg_Mux	PCSrc_Mux	Jump_Mux
0x8C430010	00011	0x00000010	0x000000FF	PC + 4	PC + 4
0x1023000C	X	0xFFFFFFFD	X	PC + 4	PC + 4

e)

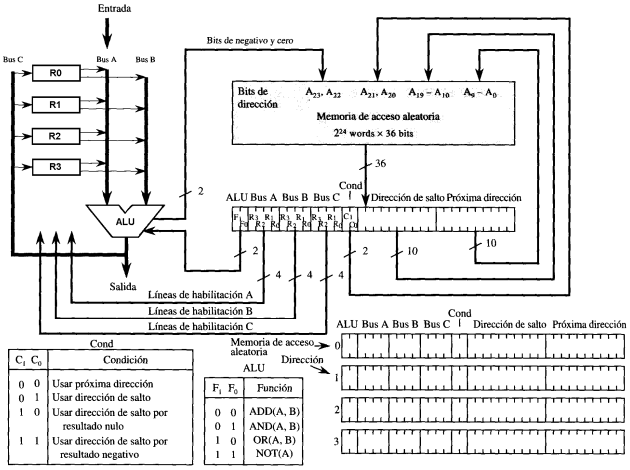
	ALU	Add (PC+4)	Add (Branch)
0x8C430010	0xFFFFFFFE and 0x00000010	PC and 4	PC+4 and 0x00000040
0x1023000C	0xFFFFFFF0 and 0xFFFFFFFD	PC and 4	PC+4 and 0x00000030

f)

	A1	A2	A3	WD3	RegWrite
0x8C430010	00010	00011	00011	0x000000FF	1
0x1023000C	00001	00011	X	X	0

4.6. En la figura adjunta se muestra una arquitectura particular. La ruta de datos tiene cuatro registros y una ALU. El control de la ruta de datos es de ciclo único y se realiza por medio de una memoria y de un registro que almacena la palabra de control (microinstrucción) en cada caso leída. Se pide:

a) Escribir en las 4 primeras posiciones de la memoria de control, la palabra necesaria para ejecutar las siguientes sentencias



M0: R1 ← ADD (R2 , R3)
M1: Saltar SI NEGATIVO (N=1) a 15_{10} ; (bneg 15)
M2: R3 ← AND (R1 , R2)
M3: Saltar siempre a 20_{10} ; (jump 20)

b) Escribir un seudo-código ensamblador y las palabras de control equivalentes que implementen la operación de multiplicar R2 <= R0xR1. Considerar que inicialmente R2 = 0 y R3=1. El programa debe terminar en un bucle infinito y el microcódigo debe estar almacenado a partir de la posición 0 de la memoria.

Nota: Si bien no hay líneas que así lo indiquen, debe interpretarse que los bits N y Z son ambos '0' cuando C_1C_0 son '00'. Esto significa que si no hay posibilidad de salto, las líneas A_{22} y A_{23} son ambas '0'.
Nota: Cada bit de los campos A, B y C corresponde directamente a un registro. Así, la palabra 1000 selecciona el registro R3, no el registro R8, inexistente

Solution:

a)

	ALU		BusA				BusB				BusC				C_1	C_2	Dirección Salto										Próxima Dirección									
0	0	0	0	1	0	0	1	0	0	0	0	0	1	0	0	0	x	x	x	x	x	x	x	x	x	x	0	0	0	0	0	0	0	0	0	1
1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	1	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	1	0
2	0	1	0	0	1	0	0	1	0	0	1	0	0	0	0	0	x	x	x	x	x	x	x	x	x	x	0	0	0	0	0	0	0	0	1	1
3	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	1	0	0	0	0	0	1	0	1	0	0	x	x	x	x	x	x	x	x	x	x

b)

Ensamblador:

```
        R3 <= not R2        ! -1 en C2 en R3 (aprovecho que -1 son todo unos en C2)
bucle:  R1 <= R1 + R3       ! Resto 1 al multiplicador
        bneg fin            ! Término si el multiplicador es 0.
                            ! La primera vez r2 = 0 cuando r1 es 0.
        R2 <= R2 + R0       ! Incremento r2 con el multiplicando
        jump bucle          ! Nueva iteración
fin:    jump fin            ! Fin
```

Microcódigo:

Nº	ALU		Bus A				Bus B				Bus C				C_1C_0		Dirección de Salto										Próxima Dirección									
0	1	1	0	1	0	0	x	x	x	x	1	0	0	0	0	0	x	x	x	x	x	x	x	x	x	x	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	1	0	0	0	1	0	1	0	0	0	0	0	x	x	x	x	x	x	x	x	x	x	0	0	0	0	0	0	0	0	1	0
2	x	x	x	x	x	x	x	x	x	x	0	0	0	0	1	1	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	1	0	1
3	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	x	x	x	x	x	x	x	x	x	x	0	0	0	0	0	0	0	1	1	0
4	x	x	x	x	x	x	x	x	x	x	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	x	x	x	x	x	x	x	x	x	x
5	x	x	x	x	x	x	x	x	x	x	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	1	x	x	x	x	x	x	x	x	x	x

4.7. The following program is executed in a single-cycle MIPS processor. Please, fill in the signals and registers values of the table when the code is executed.

Note: The table includes the initial content of four data memory elements.

Code

.text 0x0800
lw $s1, B($0)
lw $s2, C($0)
and $s3, $s1, $s2
sw $s3, D($0)
add $s4, $s1, $s2
lw $s5, D($0)
fin: j fin

.data 0x2000 A: 0x00000005 B: 0x0000000C C: 0x00000007 D: 0x0000002F

Signal/register	T	T+1
pc	0x0800	
Jump	0	
MemtoReg	1	
MemWrite	0	
Branch	0	
ALUSrc	1	
RegDst	0	
RegWrite	1	
$s1	0x0000	
$s2	0x0000	
$s3	0x0000	
$s4	0x0000	
$s5	0x0000	

Solution:

Code

.text 0x0800
lw $s1, B($0)
lw $s2, C($0)
and $s3, $s1, $s2
sw $s3, D($0)
add $s4, $s1, $s2
lw $s5, D($0)
fin: j fin

.data 0x2000 A: 0x00000005 B: 0x0000000C C: 0x00000007 D: 0x00000004

Signal/register	**T**	T+1	T+2	T+3	T+4	T+5	T+6	T+7
pc	0x0800	0x804	0x808	0x80C	0x810	0x814	0x818	0x818
Jump	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**1**
MemtoReg	**1**	**1**	**0**	**X**	**0**	**1**	**X**	**X**
MemWrite	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**
Branch	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
ALUSrc	**1**	**1**	**0**	**1**	**0**	**1**	**X**	**X**
RegDst	**0**	**0**	**1**	**X**	**1**	**0**	**X**	**X**
RegWrite	**1**	**1**	**1**	**0**	**1**	**1**	**0**	**0**
$s1	0x0000	0x0C						0x0C
$s2	0x0000	0x00	0x07					0x07
$s3	0x0000	0x00		0x04				0x04
$s4	0x0000	0x00				0x13		0x13
$s5	0x0000	0x00					0x04	0x04

4.8. En la arquitecura uniciclo del procesador MIPS, se ejecuta el código adjunto. Se pide definir en una tabla el valor de las señales de control indicadas, necesarias para ejecutar el código dado, así como el valor final de los registros señalados, sabiendo que en el ciclo 1 de reloj se está ejecutando la primera instrucción.

Nota: Si se accede a una posición de memoria cuyo valor no se puede conocer por los datos del enunciado, se supondrá que el contenido de dicha posición de memoria es 0x0A.

```
.text 0x0000
    add $s3, $s1, $s2
    beq $s1, $s2, eti
    jal eti
    lui $s1, 0x1000
.text 0x0100
eti: sw $s3, 4($s1)
    lw $s2, 4($s1)
    xor $s1, $s2, $s2
```

Ciclo	T1	T2
pc	0x0000	
$s1	0x208C	
$s2	0x2140	
$s3	0x2000	
$ra	0x0000	
MemWrite		
Jump		
MemtoReg		
RegWrite		
ALUSrc		
Branch		
PCSrc		

Solution:

Ciclo	1	2	3	4	5	6	7
pc	0x0000	0x0004	0x0008	0x0100	0x0104	0x0108	0x010C
$s1	0x208C	=	=	=	=	=	0x0000
$s2	0x2140	=	=	=	=	0x41CC	0x41CC
$s3	0x2000	0x41CC	=	=	=	0x41CC	0x41CC
$ra	0x0000	0x0000	0x0000	0x000C	=	0x000C	0x000C
Jump	0	0	1	0	0	0	?
MemtoReg	0	X	0	X	1	0	?
MemWrite	0	0	0	1	0	0	?
Branch	0	1	0	0	0	0	?
RegDst	1	X	X	X	0	1	?
RegWrite	1	0	1	0	1	1	?
ALUSrc	0	0	X	1	1	0	?
PCSrc	0	0	0	0	0	0	?

COMPUTER STRUCTURE
UNIT 4.- PROCESSOR II: DESIGN AND CONTROL OF THE DATAPATH. SINGLE-CYCLE

4.9. The following figure shows a single-cycle architecture with two independent ALUs whose operands are read/written from a 4 registers register file. The table shows the control signals of the ALUs and the operations that can be executed.

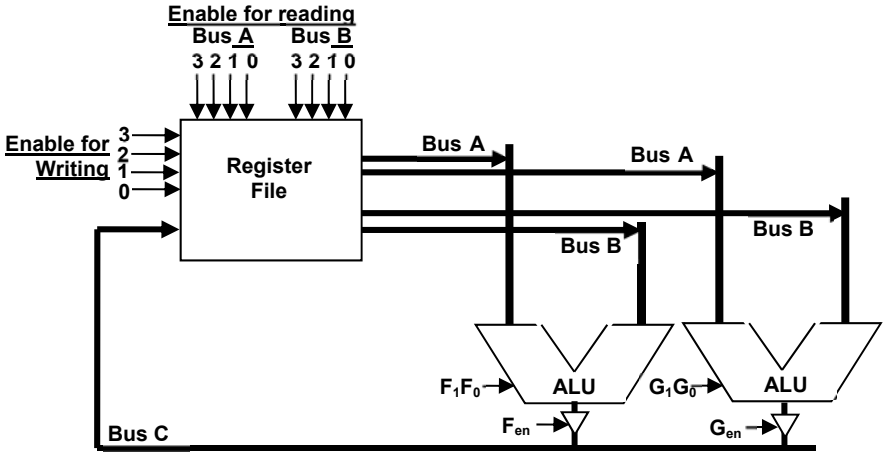
a) Write the code for a program that does the following:

a.1. XOR of the contents of the registers R_0 y R_1 storing the result in register R_0.
($R_0 \leftarrow R_0 \oplus R_1$)

a.2. The difference of registers R_1 and R_2 storing the result in register R_3
($R_3 \leftarrow R_1 - R_2$)

a.3. The product of registers R_0 and R_1 storing the result in register R_2. Suppose the initial values $R_2 = 0$ and $R_3 = 1$. Also suppose that the same control flow instructions of MIPS (branch and jump) are available.
($R_2 \leftarrow R_0 \times R_1$)

b) Write the control signals for the instructions of each of the previous codes. For case 3, ignore the control signals of the control flow instructions.



Code F_1 F_0	Function	Instruction Syntax	Code G_1 G_0	Function	Instruction Syntax
00	C = A + B	ADD (A,B,C)	00	C = A OR B	OR (A,B,C)
01	C = A + 1	INC (A,1,C)	01	C = A AND B	AND (A,B,C)
10	C = A * 2	MUL2 (A,2,C)	10	C = A XOR B	XOR (A,B,C)
11	C = A * 8	MUL8 (A,8,C)	11	C = A NAND B	NAND (A,B,C)

Note: In order to read/write in any of the 4 registers, the appropriate control word must be applied in the corresponding bus A, B or C (for R_0 it would be "0001" and for R_3 it would be "1000"). For enabling the output of one of the two ALUs, F_{en} or G_{en} must be set to '1'.

a1) Assembly code:

Nº	Instruction	Comment
1	XOR (R_0, R_1, R_0)	R_0 <= R_0 xor R_1; Direct operation in ALU

b1) Control signals

Nº	ALU F		ALU G		F_{en}	G_{en}	Bus A				Bus B				Bus C			
	F_1	F_0	G_1	G_0			R_3	R_2	R_1	R_0	R_3	R_2	R_1	R_0	R_3	R_2	R_1	R_0
1	X	X	1	0	0	1	0	0	0	1	0	0	1	0	0	0	0	1

a2) Assembly code:

Nº	Instruction	Comment
1	NAND (R_2, R_2, R_0)	R_0 <= /R_2 ; negates R2
2	INC (R_0, 1, R_0)	R_0 <= R_0 + 1 ; two's complement of R2
3	ADD (R_1, R_0, R_3)	R_3 <= R_1 + R_0 ; subtraction

b2) Control signals

Nº	ALU F		ALU G		F_{en}	G_{en}	Bus A				Bus B				Bus C			
	F_1	F_0	G_1	G_0			R_3	R_2	R_1	R_0	R_3	R_2	R_1	R_0	R_3	R_2	R_1	R_0
1	X	X	1	1	0	1	0	1	0	0	0	1	0	0	0	0	0	1
2	0	1	X	X	1	0	0	0	0	1	X	X	X	X	0	0	0	1
3	0	0	X	X	1	0	0	0	1	0	0	0	0	1	1	0	0	0

a3) Assembly code:

Nº	Instruction		Comment
1		NAND (R_2, R_2, R_3)	R_3 <= R_2 nand R_2; "1111....111" (-1) in R_3
2	loop:	ADD (R_1, R_3, R_1)	R_1 <= R_1 + R_3; subtract 1 from multiplying
3		beq R_1, R_3, fin	Branch if R_1 = R_3; end if multiplying reaches -1
4		ADD(R_2, R_0, R_2)	R_2 <= R_2 + R_0 ; Increment R_2 with multiplier
5		j loop	New iteration
6	end:	j end	End of program

b3) Control signals

Nº	ALU F		ALU G		F_{en}	G_{en}	Bus A				Bus B				Bus C			
	F_1	F_0	G_1	G_0			R_3	R_2	R_1	R_0	R_3	R_2	R_1	R_0	R_3	R_2	R_1	R_0
1	X	X	1	1	0	1	0	1	0	0	0	1	0	0	1	0	0	0
2	0	0	X	X	1	0	0	0	1	0	1	0	0	0	0	0	1	0
3	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
4	0	0	X	X	1	0	0	1	0	0	0	0	0	1	0	1	0	0
5	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
6	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

COMPUTER STRUCTURE
UNIT 4.- PROCESSOR II: DESIGN AND CONTROL OF THE DATAPATH. SINGLE-CYCLE

4.10. Se ejecuta el código adjunto escrito para MIPS con arquitectura uniciclo. En el código dado, también figura parte del contenido de memoria. Se pide definir en una tabla el valor de las señales de control indicadas, necesarias para ejecutar el código dado, así como el valor final de los registros señalados, sabiendo que en el ciclo 1 de reloj se está ejecutando la primera instrucción. Rellene además el contenido final de las posiciones de memoria indicadas.

Código

```
    .text 0x0000
    lw $s1, 4($s2)
    or $s3, $s1, $s2
    jal etiq
    add $s1, $s1, $s2
etiq: sw $s3, D($0)
    addi $s3, $s2, 4
    sw $s1, -4($s3)
```

```
    .data 0x2000
A: 0x00000001
B: 0x00000010
C: 0x00000100
D: 0x00001000
```

Contenido final de la memoria:

.data 0x2000

A: **0x00000001**

B: **0x00000100**

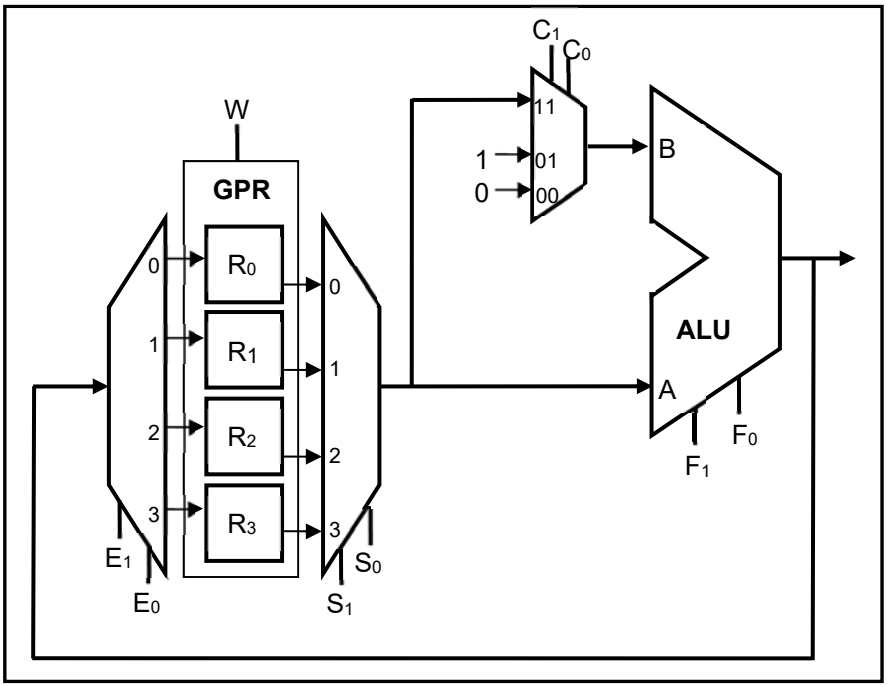
C: **0x00000100**

D: **0x00002104**

Ciclo	1	2	3	4	5	6	7
PC	0x0000	0x0004	0x0008	0x0010	0x0014	0x0018	0x001C
$s1	0x2000	0x0100					0x0100
$s2	0x2004						0x2004
$s3	0x0		0x2104			0x2008	0x2008
$ra	0x0			0x000C			0x000C
Jump	0	0	1	0	0	0	?
MemtoReg	1	0	X	X	0	X	?
MemWrite	0	0	0	1	0	1	?
Branch	0	0	0	0	0	0	?
RegDst	0	1	X	X	0	X	?
RegWrite	1	1	1	0	1	0	?
ALUSrc	1	0	X	1	1	1	?

4.11. En la tabla adjunta se muestran los bits de control de la ALU que figura en la ruta de datos de un cierto sistema digital. Los demás controles deben ser identificados por el estudiante, puesto que controlan a tres multiplexores y la línea W se utiliza para permitir la escritura en el banco de registros (GPR).

F_1	F_0	Función
0	0	A **+** B
0	1	A **OR** B
1	0	A **AND** B
1	1	A **NAND** B



Se quiere ejecutar una operación que intercambie los contenidos de los registros **R_3 y R_0**. Señale la palabra de control utilizando el menor número de ciclos que sean necesarios. En cada ciclo justificar brevemente la respuesta.
Nota: Todos los registros del GPR son de lectura/escritura y se puede utilizar cualquiera de ellos en la operación.

Ciclo	W	E_1	E_0	S_1	S_0	C_1	C_0	F_1	F_0	Comentario
1	1	0	1	1	1	0	0	0	1	Guardo R_3 en R_1 ($R_1 = R_3$ **OR** 0)
2	1	1	1	0	0	0	0	0	1	Guardo R_0 en R_3 ($R_3 = R_0$ **OR** 0)
3	1	0	0	0	1	0	0	0	1	Guardo R_1 en R_0 ($R_0 = R_1$ **OR** 0)

COMPUTER STRUCTURE
UNIT 4.- PROCESSOR II: DESIGN AND CONTROL OF THE DATAPATH. SINGLE-CYCLE

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

Código

```
        .text 0x0000
            lw $s1, 4($s2)
            and $s2, $s1, $s3
            beq $s1,$s2,etiq
            sw $s2, -4($s3)
            j fin
        .text 0x001C
etiq:   jal fin
            or $s3,$s1,$s2
fin:    addi $s3,$s1,5
            sw $s3,C($0)
        .data 0x2000
A:          0x00000002
B:          0x00000007
C:          0x00000013
```

Contenido final de la memoria

```
        .data 0x2000
A:          0x00000002
B:          0x00000007
C:          0x00000018
```

Se pide definir en una tabla el valor de las señales de control indicadas, necesarias para ejecutar el código dado, así como el valor final de los registros señalados, sabiendo que en el ciclo T de reloj se está ejecutando la primera instrucción. Rellene además el contenido final de las posiciones de memoria indicadas (cuadro superior).

Ciclo	T	T+1	T+2	T+3	T+4	T+5	T+6
PC	0x0000	0x0004	0x0008	0x001C	0x0024	0x0028	0x002C
$s1	0x3F41	0x0013					0x0013
$s2	0x2004		0x0013				0x0013
$s3	0x003F					0x0018	0x0018
$ra	0x0000				0x0020		0x0020
Jump	0	0	0	1	0	0	
MemtoReg	1	0	X	X	0	X	
MemWrite	0	0	0	0	0	1	
Branch	0	0	1	0	0	0	
RegDst	0	1	X	X	0	X	
RegWrite	1	1	0	1	1	0	
ALUSrc	1	0	0	X	1	1	

COMPUTER STRUCTURE
UNIT 4.- PROCESSOR II: DESIGN AND CONTROL OF THE DATAPATH. SINGLE-CYCLE

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

Código

```
.text 0x0
addi $s1, $0, 4
lw $s2, A($s1)
j eti1
.text 0x100
addi $s2, $0, 4
eti1: beq $s2, $s3, eti2
addi $s2, $0, 8
eti2: sw $s2, B($s1)
sllv $s1, $s2, $s3
```

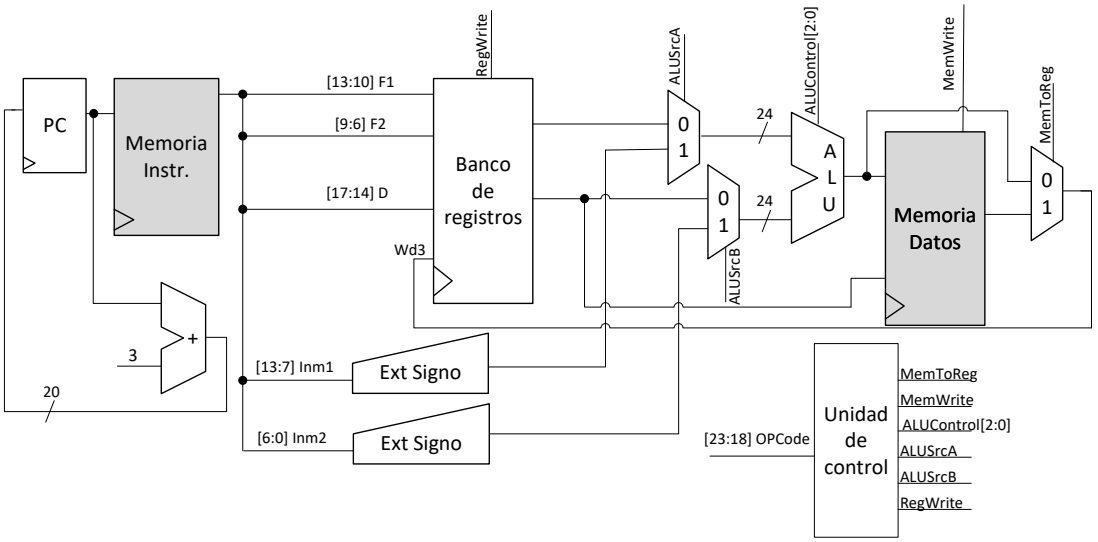
Contenido inicial de la memoria

```
        .data 0x2000
A:      0x00000001
B:      0x00000002
C:      0x00000003
```

Se pide definir en una tabla el valor de las señales de control indicadas, necesarias para ejecutar el código dado, así como el valor final de los registros señalados y el de las posiciones de memoria indicadas. Complete la tabla hasta el ciclo T+6, sabiendo que en el ciclo T de reloj se está ejecutando la primera instrucción. Complete toda la información de la tabla que se pueda, pero no añada más columnas aunque queden instrucciones sin ejecutar.

Instrucción	**addi**	**lw**	**j**	**beq**	**sw**	**sllv**	**?**
Ciclo	**T**	**T+1**	**T+2**	**T+3**	**T+4**	**T+5**	**T+6**
PC	**0x0000**	**0x0004**	**0x0008**	**0x0104**	**0x010C**	**0x0110**	**0x0114**
$s1	**0xABCD**	**0x0004**					**0x0008**
$s2	**0x1234**		**0x0002**				**0x0002**
$s3	**0x0002**						**0x0002**
A:	**0x0001**						**0x0001**
B:	**0x0002**						**0x0002**
C:	**0x0003**					**0x0002**	**0x0002**
MemWrite	**0**	**0**	**0**	**0**	**1**	**0**	**?**
RegWrite	**1**	**1**	**0**	**0**	**0**	**1**	**?**
MemtoReg	**0**	**1**	**X**	**X**	**X**	**0**	**?**
RegDst	**0**	**0**	**X**	**X**	**X**	**1**	**?**
ALUSrc	**1**	**1**	**X**	**0**	**1**	**0**	**?**
Branch	**0**	**0**	**0**	**1**	**0**	**0**	**?**
Jump	**0**	**0**	**1**	**0**	**0**	**0**	**?**

4.14. The figure shows the schematic of a microprocessor different from MIPS. There are three instruction types. In all the three types there is destination register (D), but the source operands can be: two registers (F1, F2), two immediates (Inm1, Inm2), or one register (F1) and an immediate (Inm2).



Please, answer the following questions.

a) Which is the format of an instructions with two registers as source operands? Use the names in the schematic (D, F1…) and indicate the number of bits for each field.

Size	6 bits	4 bits	4 bits	4 bits	6 bits
Field	OPCode	D	F1	F2	Other use
Bits used	23..18	17..14	13..10	9..6	5..0

b) Which is the format of an instructions with two immediates as source operands? Use the names in the schematic (D, F1…) and indicate the number of bits for each field.

Size	6 bits	4 bits	7 bits	7 bits
Field	OPCode	D	Inm1	Inm2
Bits used	23..18	17..14	13..7	6..0

c) Which is the format of an instructions with one register and one immediate as source operands? Use the names in the schematic (D, F1…) and indicate the number of bits for each field.

Size	6 bits	4 bits	4 bits	3 bits	7 bits
Field	OPCode	D	F1	Other use	Inm2
Bits used	23..18	17..14	13..10	9..7	6..0

d) Which is the word size of this processor? Justify briefly your answer.

The ALU input buses are 24-bits, so 24-bits is the word size.

e) How many bytes are necessary for the machine code of each instruction? Justify briefly your answer.

3 bytes. The PC is increased by 3 each clock cycle. Furthermore, the output of the instruction memory goes from 23 down to 0.

f) Which is the maximum capacity of the instruction memory (in bytes)? Justify briefly your answer.

Since PC has 20 bits, 2^{20} different addresses can be accessed. As the memory is byte addressable, the capacity is 2^{20} bytes, i.e. 1 MB.

g) How many registers are available in this processor? Justify briefly your answer.

The register addresses D, F1 and F2 have 4 bits, so 2^4 registers are available (16 registers).

h) How many different operations can be performed in the ALU? Justify briefly your answer.

The control input to the ALU (ALUControl) has 3 bits. Therefore, 2^3 operations can be codified (8 different operations).

4.15. During the execution of the following MIPS code, in the cycle T the values of some control signals, registers and memory addresses are as indicated in the tables.

a) Analysing the control signals at cycle T, identify the instruction being executed and fill in the rest of the control signals table after executing the rest of the code.

b) Fill in (in hexadecimal) the values of the registers and memory addresses of the table after executing the code (cycle T+5).

Code

.text 0x0000
addi $s1, $0, 0x2000
lw $s2, 8($s1)
add $s1, $s2, $s3

.text 0x0020
and $s2, $s1, $s3
sw $s2, -4($s3)
beq $s1, $s2, etiq
lw $s4, 4($s2)
addi $s3, $s1, -1
or $s2, $s1, $s2

fin: j fin
etiq: lw $s3,C($0) slt $s3, $s2, $s1 add $s2, 4($s1)

Data memory (current, T)	**Data memory (final, T+5)**
.data 0x2000	.data 0x2000
A: 0x0001	A: **0x0001** **¿?**
B: 0xFF00	B: **0xFF00** **¿?**
C: 0x0400	C: **0x2000** **¿?**
Registers (current, T)	**Registers (final, T+5)**
$pc = **0x0024** **¿?**	$pc = **0x0038** **¿?**
$s1 = 0x00FF	$s1 = **0x00FF** **¿?**
$s2 = 0x2000	$s2 = **0x20FF** **¿?**
$s3 = 0x200C	$s3 = **0x00FE** **¿?**
$s4 = 0x0000	$s4 = **0xFF00** **¿?**

Instruc.	**sw**	**beq**	**lw**	**addi**	**or**	**j**
Ciclo	**T**	**T+1**	**T+2**	**T+3**	**T+4**	**T+5**
ALUSrc	**1**	**0**	**1**	**1**	**0**	**X**
Jump	**0**	**0**	**0**	**0**	**0**	**1**
MemtoReg	**X**	**X**	**1**	**0**	**0**	**X**
MemWrite	**1**	**0**	**0**	**0**	**0**	**0**
PCSrc	**0**	**0**	**0**	**0**	**0**	**0**
RegDst	**X**	**X**	**0**	**0**	**1**	**X**
RegWrite	**0**	**0**	**1**	**1**	**1**	**0**

COMPUTER STRUCTURE
UNIT 4.- PROCESSOR II: DESIGN AND CONTROL OF THE DATAPATH. SINGLE-CYCLE

4.16. Se adjunta un código escrito para un sistema basado en MIPS, con arquitectura uniciclo igual que la estudiada en la asignatura. El programa comenzó su ejecución con el valor $pc = 0x0000, y se sabe que en el ciclo actual T, $pc = 0x00000104 y comienza la ejecución de una nueva instrucción.

Código

```
.text 0x0000
        lw $s1, A($0)
        lw $s2, B($0)
        and $s3, $s1, $s2
        j eti_1
.text 0x100
        add $s3, $s1, $s2
eti_1:  beq $s2, $s3, eti_2
        addi $s1, $s1, - 8
eti_2:  sub $s2, $s2, $s3
        sw $s2, A($s1)
        xor $s2, $s1, $s1
```

Contenido **actual** (**T**) memoria datos

```
.data 0x2000
A:      0x00000010
B:      0x000000FF
C:      0x000000AA
```

a. Con la información facilitada, se pide completar la tabla adjunta con los valores de las señales de control desde la instrucción actual (ciclo "T") hasta completar los 4 ciclos señalados (ciclo "T+3").

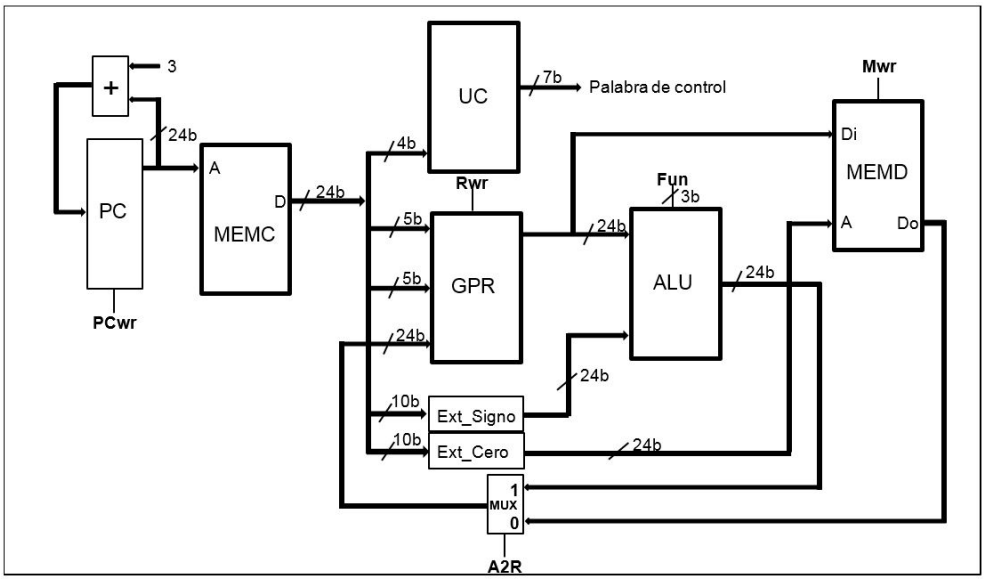
Instrucción	**beq**	**addi**	**sub**	**sw**
Ciclo	**T**	**T+1**	**T+2**	**T+3**
MemWrite	**0**	**0**	**0**	**1**
RegWrite	**0**	**1**	**1**	**0**
MemtoReg	**X**	**0**	**0**	**X**
RegDst	**X**	**0**	**1**	**X**
ALUSrc	**0**	**1**	**0**	**1**
Branch	**1**	**0**	**0**	**0**
Jump	**0**	**0**	**0**	**0**

b. Señalar el valor de los registros y de las posiciones de memoria indicados, antes de terminar el ciclo "T+4". Si se desconoce algún valor de los solicitados, escriba en la respuesta el valor 0xFFFF.

%pc	**$s1**	**$s2**	**$s3**	**%ra**	**A**	**B**	**C**
0x0114	**0x0008**	**0x00EF**	**0x0010**	**0xFFFF**	**0x0010**	**0x00FF**	**0x00EF**

COMPUTER STRUCTURE
UNIT 4.- PROCESSOR II: DESIGN AND CONTROL OF THE DATAPATH. SINGLE-CYCLE

4.17. The figure shows the architecture of a single-cycle processor. There are five main blocks: both memories (MEMC, MEMD), register file (GPR), arithmetic-logic unit (ALU) and control unit (UC). There are other known digital blocks that, along with the control signals, are used for configuring the datapath.
According to the shown architecture, answer the following questions briefly justifying each answer:



a) Indicate the word size of this processor.

Size in bits: 24		**Justification:** It is the word size of the ALU, which has 24 bits operands.

b) Maximum number of different instructions in this processor.

Max. # instr: 16		**Justification:** The input to the control unit is 4 bits. Therefore, there can be 2^4 = 16 different instructions.

c) Maximum number of registers in the GPR.

Max. # regs: 32		**Justification:** The address inputs to the GPR are 5 bits wide. Therefore, the number of registers is 2^5 = 32.

d) Write the control signals for the instruction **addi r1, r2, 8** # (r1 = r2 + 8)

PCwr	Mwr	Rwr	A2R	Fun(2:0)
1	0	1	1	XXX

e) Write the control signals for the instruction **sw r1, 8** # (r1 => MEMD(8))

PCwr	Mwr	Rwr	A2R	Fun(2:0)
1	1	0	X	XXX

f) Supposing the ALU can perform the necessary operations, indicate other reasons why the following instructions can not be performed.

add r1, r2, r3 (r1 = r2 + r3)		**Justification:** The GPR does not have three ports (two read ports and one write port), but only two ports (one read and one write port).
lw r1, 5(r2) r1 = MEMD[(5+r2)]		**Justification:** The memory address is only generated through the zero-extend circuit. There is no way of generating the necessary address and sending it to the data memory in one cycle.