**FUNDAMENTOS DE MICROPROCESADORES**
**PROBLEMAS DE LA UNIDAD 4.- EL PROCESADOR II: LA RUTA DE DATOS y SU CONTROL**

**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?

**Solución:**

| **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.

**FUNDAMENTOS DE MICROPROCESADORES**
**PROBLEMAS DE LA UNIDAD 4.- EL PROCESADOR II: LA RUTA DE DATOS y SU CONTROL**

**4.2.** Las distintas entidades digitales tienen latencias distintas (latencia: tiempo necesario para hacer su trabajo). La latencia mínima de una instrucción está determinada por la latencia de los bloques a lo largo de su camino crítico (el de mayor latencia). En la arquitectura de MIPS de ciclo único estudiada en clase, suponer las siguientes latencias:

| I-MEM | Add | Mux | ALU | BancoReg | D-MEM | Control | Ext_signo | Despl_iz-2 | And |
|---|---|---|---|---|---|---|---|---|---|
| 500 ps | 150 ps | 30 ps | 180 ps | 220 ps | 1000 ps | 65 ps | 90 ps | 20 ps | 20 ps |

**a.** ¿Cuál es el camino crítico para una instrucción **`and`**? ¿Cuál es la duración del ciclo de reloj si el único tipo de instrucción admitida son las instrucciones en la ALU (**`add`**, **`and`**, etc.)?
**b.** ¿Cuál es el camino crítico para una instrucción de carga **`lw`**? ¿Cuál es la duración del ciclo de reloj si sólo se admiten instrucciones de carga (**`lw`**)?
**c.** ¿Cuál es el camino crítico para una instrucción de salto condicional **`beq`**?
**d.** ¿Cuál es la duración del ciclo de reloj si se admiten las instrucciones **`add, beq, lw y sw`**?

**Solución:**

**a)** El camino para **and** es:
(leer instrucción, 500), (Control, 65 // leer banco registros, 220), (Mux, 30), (AND en ALU,180), (Mux, 30)

$T_{CRITICO1}$ = 500 + 220 + 30 + 180 + 30 = 960 ps (**camino crítico**)

En paralelo se produce la actualización del PC: (Suma+4, 150), (Mux, 30), (Mux, 30).

$T_{CRITICO2}$ = 150 + 30 +30 = 210 ps

* El tiempo necesario para la escritura en el banco de registros y en el PC, forma parte del ciclo siguiente.

El ciclo de reloj en este caso debe ser $T_{CICLO} \geq 960$ ps, Freq ≤ 1,04 GHz

**b)** El camino para **lw** es:
(leer instrucción, 500), (Control, 65 // leer banco registros, 220 // extensión signo y Mux, 90+30), (SUMA en ALU, 180), (leer dato, 1000), (Mux, 30)

$T_{CRITICO1}$ = 500 + 220 + 180 + 1000 + 30 = 1.930 ps (**camino crítico**)

En paralelo se produce la actualización del PC: (Suma+4, 150), (Mux, 30), (Mux, 30).

El ciclo de reloj en este caso debe ser $T_{CICLO} \geq 1930$ ps, Freq ≤ 518 MHz

**c)** El camino para **beq** es:
(leer instrucción, 500), (Control, 65 // leer banco registros 220), (Mux, 30), (RESTA en ALU, 180), (And del Zero, 20), (Mux, 30), (Mux, 30)

$T_{CRITICO1}$ = 500 + 220 + 30 + 180 + 20 + 30 + 30 = 1010 ps (**camino crítico**)

En paralelo se produce la actualización del PC: (Suma+4, 150 // leer instrucción, 500; SigExt, 90; <<2, 20), (SumaBranch, 150), (Mux, 30), (Mux, 30).

$T_{CRITICO2}$ = 500 + 90 + 20 + 150 + 30 + 30 = 820 ps

**d)** El reloj del sistema completo lo señala el valor del camino crítico mayor (caso b). Por tanto:

El ciclo de reloj en este caso debe ser $T_{CICLO} \geq 1930$ ps, Freq ≤ 518 MHz

**FUNDAMENTOS DE MICROPROCESADORES**
**PROBLEMAS DE LA UNIDAD 4.- EL PROCESADOR II: LA RUTA DE DATOS y SU CONTROL**

**4.3.** Si partimos como base de la arquitectura de MIPS de ciclo único estudiada en clase, se pide incluir de forma independiente en la ruta de datos tres nuevas instrucciones no implementadas:

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.** En el caso de que los haya, ¿cuáles de los bloques ya existentes pueden utilizarse para la nueva instrucción?
**b.** En el caso de que se pudiera, ¿qué bloques funcionales nuevos se necesitarían para la nueva instrucción?
**c.** En el caso de que las haya, ¿qué nuevas señales de control es necesario añadir a la unidad de control para poder ejecutar la nueva instrucción?

**Solución:**

**1a)** Esta instrucción utiliza I-MEM, un puerto de lectura y el de escritura del banco de registros y la ALU modificada (para que incluya la operación desplazamiento).

**1b)** Despl es el campo shamt de las instrucciones R-type, bits [10:6] de Instr. Este campo ha de llegar al primer operando de la ALU, por lo que se necesita un nuevo multiplexor a la entrada del primer operando de la ALU que elija entre [rs] que viene por RD1 o shamt, que son los bits [10:6] de Instr a los que se añaden 27 ceros por la izquierda para llegar a 32 bits en total (a la ALU llegan 32 bits).

**1c)** El nuevo multiplexor necesita una señal de control (que podemos llamar shift), aparte de que por ALUControl[2:0] hay que poner un código que la ALU interprete como desplazamiento.

**2a)** Esta instrucción utiliza I-MEM, el puerto de escritura del banco de registros y la lógica de construcción de la JTA ya utilizada para la instrucción j.

**2b)** Sólo necesita añadir dos nuevos Mux, uno a la entrada del puerto A3 del banco de registros, para facilitar la dirección del registro $ra <= "11111", para esta nueva instrucción. El segundo a la entrada del puerto WD3 para incorporar el valor actual PC+4 como dirección de retorno.

**2c)** Se necesita una nueva línea de control, que en activo, habilite la entrada de los mux incorporados a la ruta de datos.

**3a)** Esta instrucción utiliza I-MEM, los dos puertos de lectura y el de escritura del banco de registros y la ALU modificada.

**3b)** Se necesita aumentar en uno el número de puertos de lectura al banco de registros (para leer rx, desde el campo shamt de 5 bits de una instrucción de R-Type). Se necesitará una nueva Unidad de Suma a continuación de la ALU actual o diseñar una nueva ALU con tres operandos de entrada.

**3c)** Sólo se necesita una nueva palabra de control ALUControl[2:0] para seleccionar la nueva operación.

**FUNDAMENTOS DE MICROPROCESADORES**
**PROBLEMAS DE LA UNIDAD 4.- EL PROCESADOR II: LA RUTA DE DATOS y SU CONTROL**

**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?

**Solución:**

**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.

**FUNDAMENTOS DE MICROPROCESADORES**
**PROBLEMAS DE LA UNIDAD 4.- EL PROCESADOR II: LA RUTA DE DATOS y SU CONTROL**

**4.5.** Dada la arquitectura de ciclo único de MIPS, se señalan dos instrucciones en lenguaje máquina 0x8C430010 y 0x1023000C. Se conoce que todos los datos en memoria valen 0x0FF y que el contenido de los registros señalados en la tabla adjunta es:

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

Para cada una de las instrucciones anteriores, se pide:

- **a.** ¿Cuál es la salida de la extensión de signo y la salida de la unidad "Despl.Izq.2" ubicado en la ruta de datos del salto incondicional?
- **b.** ¿Cuáles son los valores en binario de las entradas de la unidad de control de la ALUControl[2:0]?
- **c.** ¿Cuál es la nueva dirección del registro PC después de ejecutar la instrucción?
- **d.** Mostrar los valores en hexadecimal de las salidas de cada multiplexor durante la ejecución. Si se desconoce el valor indicarlo con una X.
- **e.** ¿Cuáles son los valores en hexadecimal de las entradas de la ALU y de las 2 unidades de suma?
- **f.** ¿Cuáles son los valores de las entradas del banco de registros?. Señale en binario las entradas de 5 bits y en hexadecimal la de 32 bits.

**Solución:**

**a)** Para 0x8C430010 => lw $v1, 16($v0)
* A la salida de la extensión de signo (32b): 0x00000010
* A la salida del Desplz2 de la ruta del salto incondicional (28b): 0x10C0040

Para 0x1023000C => beq $at, $v1, 0x000C
* A la salida de la extensión de signo (32b): 0x0000000C
* A la salida del Desplz2 de la ruta del salto incondicional (28b): 0x08C0030

**b)** Para 0x8C430010 => lw $v1, 16($v0)
* ALUControl[2:0] = "010" (suma, se trata de un instrucción lw)

Para 0x1023000C => beq $at, $v1, 0x000C
* ALUControl[2:0] = "110" (resta, se trata de un instrucción beq)

**c)** Para 0x8C430010 => lw $v1, 16($v0)
* PC <= PC + 4 (secuencia de ejecución normal, se trata de un instrucción lw)

Para 0x1023000C => beq $at, $v1, 0x000C
* PC <= PC + 4 (la instrucción es: beq $at, $v1, 0x000C, salta a BTA si $at = $v1, como 1≠ 3, no salta). Si hubiera saltado, el valor sería 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 y 0x00000010 | PC y 4 | PC+4 y 0x00000040 |
| 0x1023000C | 0xFFFFFFF0 y 0xFFFFFFFD | PC y 4 | PC+4 y 0x00000030 |

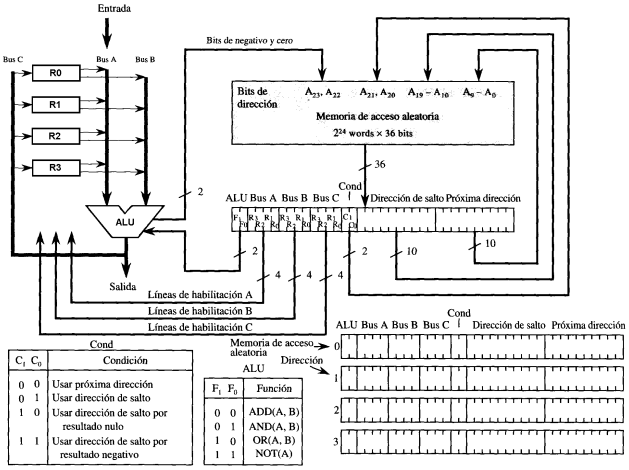
**f)**

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

**FUNDAMENTOS DE MICROPROCESADORES**
**PROBLEMAS DE LA UNIDAD 4.- EL PROCESADOR II: LA RUTA DE DATOS y SU CONTROL**

**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

**Solución:**

**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 |

**FUNDAMENTOS DE MICROPROCESADORES**
**PROBLEMAS DE LA UNIDAD 4.- EL PROCESADOR II: LA RUTA DE DATOS y SU CONTROL**

**4.7.** En la primera columna de la tabla adjunta se muestra un programa escrito para MIPS que se ejecuta en la arquitectura uniciclo estudiada. Se pide indicar los valores y las señales de los registros indicados en la tabla, al ejecutar el código dado.

**Nota:** En la tabla se incluyen cuatro palabras almacenadas en la memoria de datos, indicándose dirección y valor de la palabra contenida en dicha dirección:

| Código | Señal/registro | T | T+1 |
|---|---|---|---|
| .text 0x0800<br>lw $s1, B($0)<br>lw $s2, C($0)<br>and $s3, $s1, $s2<br>sw $s3, D($0)<br>add $s4, $s1, $s2<br>lw $s5, D($0)<br>fin: j fin | pc | 0x0800 | |
| | Jump | 0 | |
| | MemtoReg | 1 | |
| | MemWrite | 0 | |
| | Branch | 0 | |
| | ALUSrc | 1 | |
| | RegDst | 0 | |
| | RegWrite | 1 | |
| .data 0x2000<br>A: 0x00000005<br>B: 0x0000000C<br>C: 0x00000007<br>D: 0x0000002F | | | |
| | $s1 | 0x0000 | |
| | $s2 | 0x0000 | |
| | $s3 | 0x0000 | |
| | $s4 | 0x0000 | |
| | $s5 | 0x0000 | |

**Solución:**

| Código | Señal/registro | T | T+1 | T+2 | T+3 | T+4 | T+5 | T+6 | T+7 |
|---|---|---|---|---|---|---|---|---|---|
| .text 0x0800<br>lw $s1, B($0)<br>lw $s2, C($0)<br>and $s3, $s1, $s2<br>sw $s3, D($0)<br>add $s4, $s1, $s2<br>lw $s5, D($0)<br>fin: j fin | 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** |
| .data 0x2000<br>A: 0x00000005<br>B: 0x0000000C<br>C: 0x00000007<br>D: 0x00000004 | | | | | | | | | |
| | $s1 | 0x0000 | 0x0C | | | | | | 0x0C |
| | $s2 | 0x0000 | 0x00 | 0x07 | | | | | 0x07 |
| | $s3 | 0x0000 | 0x00 | | 0x04 | | | | 0x04 |
| | $s4 | 0x0000 | 0x00 | | | | 0x13 | | 0x13 |
| | $s5 | 0x0000 | 0x00 | | | | | 0x04 | 0x04 |

**FUNDAMENTOS DE MICROPROCESADORES**
**PROBLEMAS DE LA UNIDAD 4.- EL PROCESADOR II: LA RUTA DE DATOS y SU CONTROL**

**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 | | |

**Solución**:

| 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 | ? |

**FUNDAMENTOS DE MICROPROCESADORES**
**PROBLEMAS DE LA UNIDAD 4.- EL PROCESADOR II: LA RUTA DE DATOS y SU CONTROL**

**4.9.** En la figura adjunta se muestra un sistema con dos ALUs independientes, cuyos operandos se leen/escriben de/en un banco de 4 registros. Se adjunta una tabla para el control del sistema en donde se señalan las 4 operaciones que cada ALU puede ejecutar.

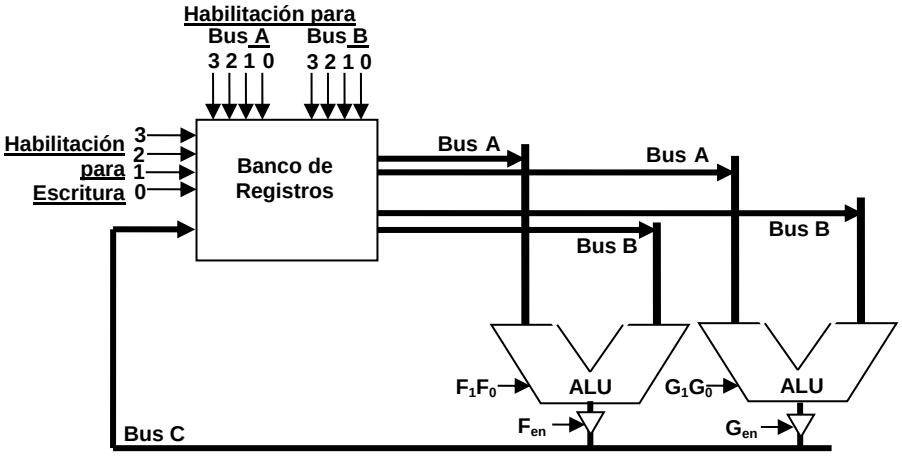
**a)** Escribir el seudocódigo para ejecutar una secuencia de control que calcule:

**a.1.** La función XOR de los contenidos de los registros $R_0$ y $R_1$ y deje el resultado en el registro $R_0$.
($R_0 \leftarrow R_0 \oplus R_1$)

**a.2.** La diferencia entre los contenidos de los registros $R_1$ y $R_2$ y deje el resultado en el registro $R_3$
($R_3 \leftarrow R_1 - R_2$)

**a.3.** El producto de los contenidos de los registros $R_0$ y $R_1$ y deje el resultado en el registro $R_2$. Suponer inicialmente que $R_2 = 0$ y $R_3 = 1$. Suponer que existen las mismas instrucciones de salto (brach y jump) vistas en MIPS.
($R_2 \leftarrow R_0 \times R_1$)

**b)** Identificar las señales de control para realizar las instrucciones pertinentes en cada uno de los programas anteriores. En el caso 3, no indicar el control para las señales de salto utilizadas.




| Código $F_1$ $F_0$ | Función | Sintaxis Instrucción | Código $G_1$ $G_0$ | Función | Sintaxis Instrucción |
|---|---|---|---|---|---|
| 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) |

**Nota:** Para leer/escribir de/en uno de los 4 registros, se debe aplicar una palabra de control en el bus que corresponda (para $R_0$ será "0001" y para $R_3$ será "1000"). Para activar la salida de una de las dos ALUs, se debe activar '1' la señal de habilitación correspondiente $F_{en}$ o $G_{en}$.

**a1) Código ensamblador:**

| Nº | Instrucción | Comentario/descripción |
|---|---|---|
| 1 | XOR ($R_0$, $R_1$, $R_0$) | $R_0$ <= $R_0$ xor $R_1$; Operación directa en la ALU |

**b1) Señales de control**

| 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 |

**FUNDAMENTOS DE MICROPROCESADORES**
**PROBLEMAS DE LA UNIDAD 4.- EL PROCESADOR II: LA RUTA DE DATOS y SU CONTROL**

**a2) Código ensamblador:**

| Nº | Instrucción | Comentario/descripción |
|---|---|---|
| 1 | NAND ($R_2$, $R_2$, $R_0$) | $R_0$ <= /$R_2$ ; invierte el substraendo |
| 2 | INC ($R_0$, 1, $R_0$) | $R_0$ <= $R_0$ + 1 ; calcula el C'2 del substraendo |
| 3 | ADD ($R_1$, $R_0$, $R_3$) | $R_3$ <= $R_1$ + $R_0$ ; hace la resta solicitada |

**b2) Señales de control**

| 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) Código ensamblador:**

| Nº | Instrucción | | Comentario/descripción |
|---|---|---|---|
| 1 | | NAND ($R_2$, $R_2$, $R_3$) | $R_3$ <= $R_2$ nand $R_2$; "1111....111" (-1) en $R_3$ |
| 2 | bucle: | ADD ($R_1$, $R_3$, $R_1$) | $R_1$ <= $R_1$ + $R_3$; Resto 1 al multiplicador |
| 3 | | beq $R_1$, $R_3$, fin | Salta si $R_1$ = $R_3$; finaliza si el multiplicador es -1 |
| 4 | | ADD($R_2$, $R_0$, $R_2$) | $R_2$ <= $R_2$ + $R_0$ ; Incremento $R_2$ con el multiplicando |
| 5 | | j bucle | Nueva iteración |
| 6 | fin: | j fin | Fin del programa |

**b3) Señales de control**

| 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 | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- |

**FUNDAMENTOS DE MICROPROCESADORES**
**PROBLEMAS DE LA UNIDAD 4.- EL PROCESADOR II: LA RUTA DE DATOS y SU CONTROL**

**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**

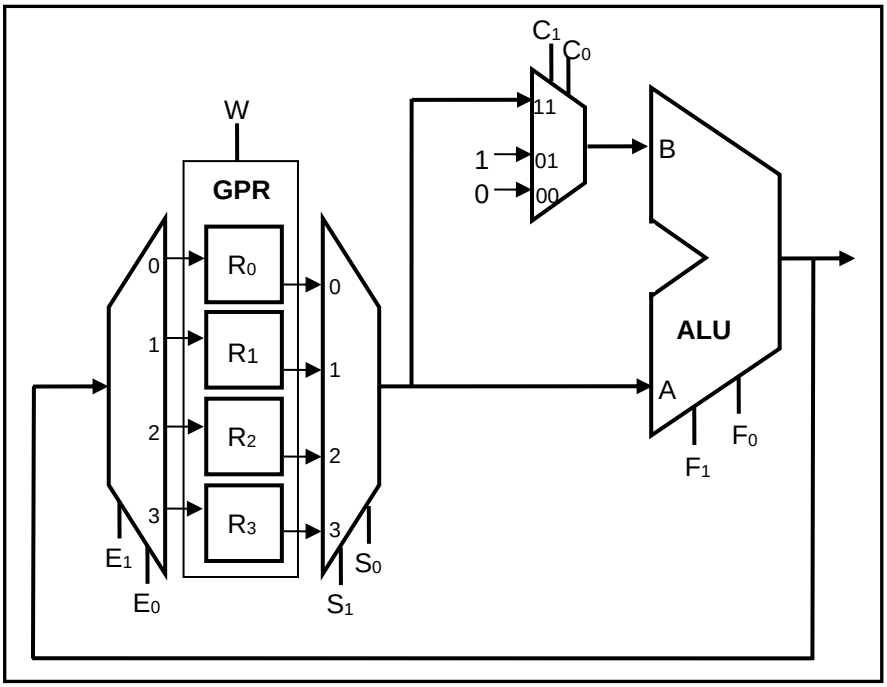
D: **0x00002104**

| 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 | ? |

**FUNDAMENTOS DE MICROPROCESADORES**
**PROBLEMAS DE LA UNIDAD 4.- EL PROCESADOR II: LA RUTA DE DATOS y SU CONTROL**

**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) |

**FUNDAMENTOS DE MICROPROCESADORES**
**PROBLEMAS DE LA UNIDAD 4.- EL PROCESADOR II: LA RUTA DE DATOS y SU CONTROL**

**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 | |

**FUNDAMENTOS DE MICROPROCESADORES**
**PROBLEMAS DE LA UNIDAD 4.- EL PROCESADOR II: LA RUTA DE DATOS y SU CONTROL**

**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

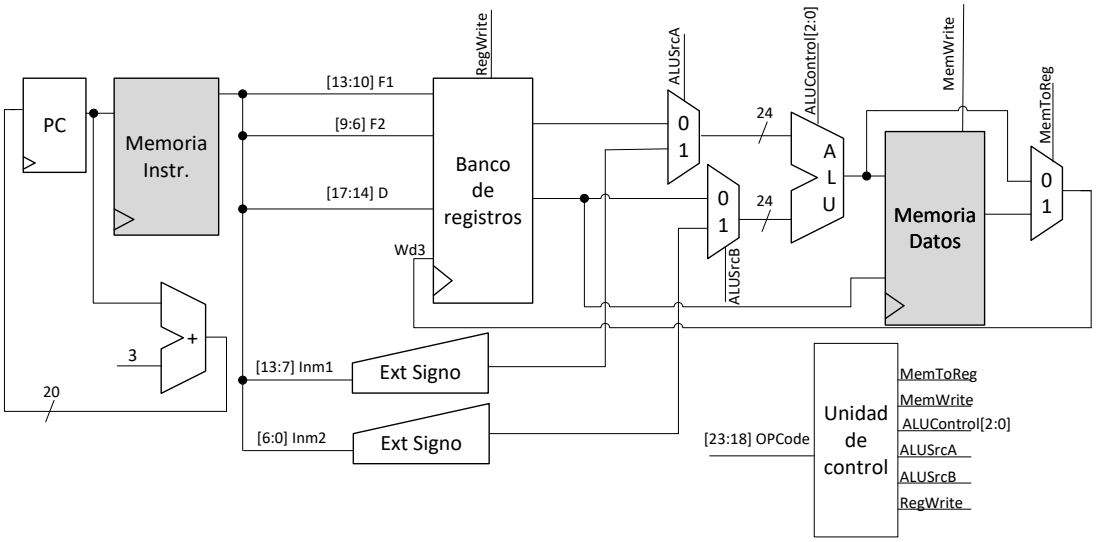
```
        .data 0x2000
A:      0x00000001
B:      0x00000002
C:      0x00000003
```

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** | **?** |

**FUNDAMENTOS DE MICROPROCESADORES**
**PROBLEMAS DE LA UNIDAD 4.- EL PROCESADOR II: LA RUTA DE DATOS y SU CONTROL**

**4.14.** En la figura que se adjunta se puede observar el esquemático de un procesador no MIPS. Se conocen tres tipos de instrucciones en las que siempre hay un registro de destino (D), pero donde los operandos fuentes pueden ser: dos registros (F1, F2), dos datos inmediatos (Inm1, Inm2), o un registro (F1) y un dato inmediato (Inm2).




Sabiendo esa información conteste a las siguientes preguntas.

a) ¿Qué formato tiene una instrucción con dos registros fuente como operandos fuente? Utilice los nombres del esquemático (D, F1…) e indique cuántos bits ocupa cada campo.

| **Tamaño** | **6 bits** | **4 bits** | **4 bits** | **4 bits** | **6 bits** |
|---|---|---|---|---|---|
| **Campo** | **OPCode** | **D** | **F1** | **F2** | **Otros usos** |
| **Bits implicados** | **23..18** | **17..14** | **13..10** | **9..6** | **5..0** |

b) ¿Qué formato tiene una instrucción con dos datos inmediatos como operandos fuente? Utilice los nombres del esquemático (D, F1…) e indique cuántos bits ocupa cada campo.

| **Tamaño** | **6 bits** | **4 bits** | **7 bits** | **7 bits** |
|---|---|---|---|---|
| **Campo** | **OPCode** | **D** | **Inm1** | **Inm2** |
| **Bits implicados** | **23..18** | **17..14** | **13..7** | **6..0** |

c) ¿Qué formato tiene una instrucción con un registro y un dato inmediato como operandos fuente? Utilice los nombres del esquemático (D, F1…) e indique cuántos bits ocupa cada campo.

| **Tamaño** | **6 bits** | **4 bits** | **4 bits** | **3 bits** | **7 bits** |
|---|---|---|---|---|---|
| **Campo** | **OPCode** | **D** | **F1** | **Otros usos** | **Inm2** |
| **Bits implicados** | **23..18** | **17..14** | **13..10** | **9..7** | **6..0** |

d) ¿Qué tamaño de palabra tiene este procesador? Justifique brevemente la respuesta.

**El tamaño de los buses de la ALU es 24 bits, por lo que tiene 24 bits de tamaño de palabra.**

e) ¿Cuántos bytes ocupa una instrucción? Justifique brevemente la respuesta.

**3 bytes. Se puede ver en el sumador de la ruta del PC, que se incrementa en 3 unidades en cada ciclo de reloj.**

f) ¿Cuál es la máxima capacidad de la memoria de instrucciones, expresada en bytes? Justifique brevemente la respuesta.

**Ya que la señal PC tiene 20 bits, se pueden acceder a $2^{20}$ direcciones distintas. Como se accede a nivel de byte, el máximo espacio direccionable es $2^{20}$ bytes, es decir, 1 MB.**

g) ¿Cuántos registros pueden ser direccionados con este procesador? Justifique brevemente la respuesta.

**Se puede observar que las direcciones D, F1 y F2 tienen 4 bits, por lo que se pueden direccionar $2^4$ registros, es decir, 16 registros.**

h) ¿Cuántas operaciones diferentes como máximo puede realizar la ALU? Justifique brevemente la respuesta.

**Se puede observar que el código de operación de la ALU (ALUControl) tiene 3 bits. Por tanto, se pueden codificar $2^3$ operaciones, es decir, 8 operaciones distintas.**

**4.15.** Durante la ejecución de un cierto código ensamblador para MIPS, en el ciclo de reloj de valor T, el valor de las señales de control, el de alguno de los registros y el de algunas posiciones de la memoria de datos, son

los señalados en las tablas facilitadas. Se pide:

**a)** Analizando las señales de control para el ciclo de reloj actual de valor T, identificar la instrucción que se está ejecutando y completar el código restante con el valor en binario de las señales de control indicadas.

**b)** Completar con su valor en hexadecimal el de los registros y el de las posiciones de memoria indicadas (A, B, C) al ejecutar el código facilitado.

| **Código** |
|---|
| .text 0x0000<br>addi $s1, $0, 0x2000<br>lw $s2, 8($s1)<br>add $s1, $s2, $s3 |
| .text 0x0020<br>and $s2, $s1, $s3<br>sw $s2, -4($s3)<br>beq $s1, $s2, etiq<br>lw $s4, 4($s2)<br>addi $s3, $s1, -1<br>or $s2, $s1, $s2 |
| fin: j fin |
| etiq: lw $s3,C($0)<br>slt $s3, $s2, $s1<br>add $s2, 4($s1) |

| **Memoria datos (actual, T)** | **Memoria datos (final, T+5)** |
|---|---|
| .data 0x2000 | .data 0x2000 |
| A: 0x0001 | A: **0x0001** **¿?** |
| B: 0xFF00 | B: **0xFF00** **¿?** |
| C: 0x0400 | C: **0x2000** **¿?** |
| **Valor Registros (actual, T)** | **Valor Registros (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** |

**FUNDAMENTOS DE MICROPROCESADORES**
**PROBLEMAS DE LA UNIDAD 4.- EL PROCESADOR II: LA RUTA DE DATOS y SU CONTROL**

**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").

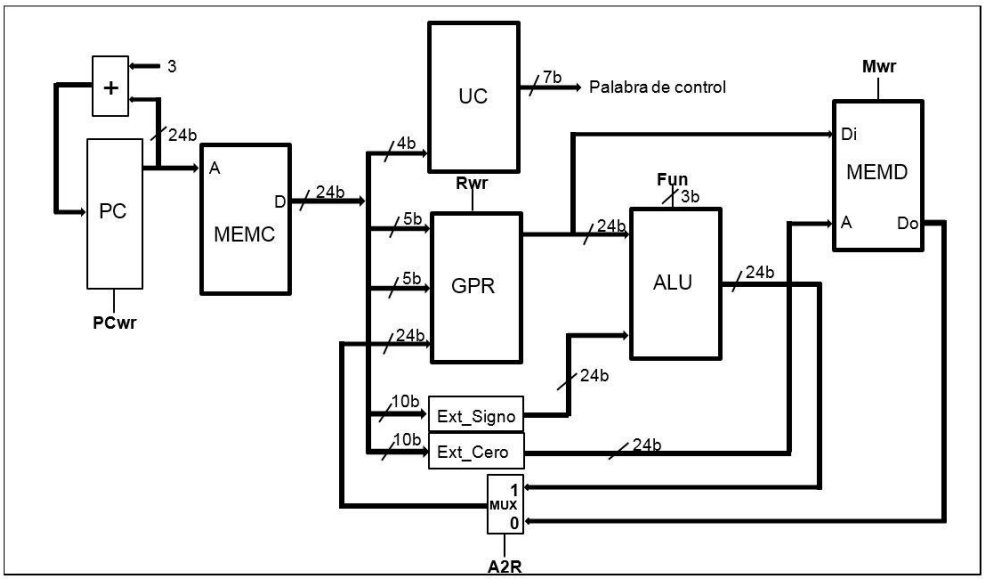
| **Instrucción** | **beq** | **addi** | **sub** | **sw** |
|---|---|---|---|---|
| **Ciclo** | **T** | **T+1** | **T+2** | **T+3** |
| **MemWrite** | **0** | **0** | **0** | **1** |
| **RegWrite** | **0** | **1** | **1** | **0** |
| **MemtoReg** | **X** | **0** | **0** | **X** |
| **RegDst** | **X** | **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** |

**FUNDAMENTOS DE MICROPROCESADORES**
**PROBLEMAS DE LA UNIDAD 4.- EL PROCESADOR II: LA RUTA DE DATOS y SU CONTROL**

**4.17.** En la figura adjunta se muestra el esquema de un determinado procesador uniciclo. En la figura se distinguen cinco bloques fundamentales para el procesador, las dos unidades de memoria (MEMC, MEMD), el banco de registros (GPR), la unidad aritmético-lógica (ALU) y la unidad de control (UC). Asimismo el esquema presenta otros elementos digitales conocidos que junto a los bits de control, se emplean para configurar la ruta de datos de este sistema.
En base a la información facilitada en el esquema, justificando de forma breve cada respuesta, se pide:




**a)** Señale el tamaño de palabra en bits del procesador.

| | | |
|---|---|---|
| **Tamaño en bits: 24** | | **Justificación:** El tamaño del procesador lo marca la capacidad de operación de la ALU y en este caso son dos operandos de 24 bits. |

**b)** Señale el máximo número de instrucciones diferentes para este procesador.

| | | |
|---|---|---|
| **Máximo nº Instr.: 16** | | **Justificación:** La entrada de direccionamiento en el GPR a la UC es de 4 bits. En consecuencia el máximo número de instrucciones diferentes serán $2^4 = 16$ instrucciones. |

**c)** Señale el máximo número de registros en el GPR.

| | | |
|---|---|---|
| **Máximo nº Reg: 32** | | **Justificación:** Los dos únicos campos de la instrucción que acceden al GPR, son de 5 bits. En consecuencia la capacidad de identificar registros como operandos es de $2^5 = 32$ registros. |

**d)** Escriba la palabra de control para la instrucción **addi r1, r2, 8** # ( r1 = r2 + 8)

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

**e)** Escriba la palabra de control para la instrucción **sw r1, 8** # ( r1 => MEMD(8))

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

**f)** Suponiendo que la ALU tenga incluida las operaciones necesarias, señale las razones que impiden ejecutar las siguientes instrucciones.

| | | |
|---|---|---|
| **add r1, r2, r3**<br>(r1 = r2 + r3) | | **Justificación:** El GPR no tiene tres puertos, dos de lectura y uno de escritura, necesarios para ejecutar esta instrucción en un sólo ciclo. |
| **lw r1, 5(r2)**<br>r1 = MEMD[(5+r2)] | | **Justificación:** La dirección de la memoria de datos sólo se genera desde el circuito de extensión de cero, luego no existe una ruta para obtener la dirección propuesta. |