Boletín de problemas COMPUTADOR SIMPLE

Problema 1.

Deduzca el valor de los registros x5, x6, x7 y x8 tras ejecutar el siguiente fragmento de programa.

```
addi x5,x0,4
addi x6,x0,0x7ff
addi x7,x0,0x800
addi x8,x5,0
```

Solución:

x5 = 4
x6 = 0x7ff (2047)
x7 = 0x800 (-2048)
x8 = 4

Problema 2.

a) Escriba un fragmento de programa que realice secuencialmente lo siguiente:
   - asigna el valor 7 al registro x2
   - asigna al registro x3 el valor del registro x2+5
   - asigna al registro x4 el valor de x3-5
   - pone a cero el registro x5
   - incrementa el registro x5
   - resta 2 unidades al registro x5
   - asigna el valor 30 al registro x6

b) Deduzca el valor final de todos los registros modificados

Solución:

```
add x2,x0,7
addi x3,x2,5
addi x4,x3,-5    o   addi x4,x3,0xffb
add x5,x0,x0    o   addi x5,x0,0
addi x5,x5,1
addi x5,x5,-2    o   addi x5,x5,0xffe
addi x6,x0,30
```

x2 = 7, x3 = 12, x4 = 7, x5 = -1, x6 = 30

Problema 3.
Considere las siguientes instrucciones para el RISC Y y los contenidos iniciales de los registros que se muestran. Tras cada instrucción muestre el valor que van tomando los registros.

```
add x1,x1,x2
addi x3,x2,2
sub x4,x3,x0
andi x2,x3,0xf0
sll x4,x2,x5
or x1,x1,x2
add x2,x0,x4
and x5,x1,x3
```

```
x1=0x00000016

x2=0x00000054

x3=0xffffffff

x4=0x00000000

x5=0x00000004
```

Solución:

```
add x1,x1,x2  → x1=0x0000006a
addi x3,x2,2  → x3=0x00000056
sub x4,x3,x0  → x4=0x00000056
andi x2,x3,0xf0 → x2=0x00000050
sll x4,x2,x5 → x4=0x00000500
or x1,x1,x2  → x1=0x0000007a
add x2,x0,x4 → x2=0x00000500
and x5,x1,x3 → x5=0x00000052
```

Problema 4.
Indique por qué las siguientes instrucciones no son válidas:

```
addi x3,3,x2
add x3,x2,0(x1)
slli x3,x3,40
lw x8,-4000(x1)
```

Solución:

```
addi x3,3,x2  → los campos 2 y 3 deben ser registro y dato respectivamente
posible corrección:  addi x3,x2,3

add x3,x2,0(x1)  → add no puede operar con memoria
posible corrección:
                     lw x7,0(x1)
                     add x3,x2,x7
```

```
slli x3,x3,40  → sí es válida
sería equivalente a slli x3,x3,8 ya que solo se usan los 5 bits menos
significativos

lw x8,-4000(x1) → -4000 excede los límites de 12 bits: [-2048,+2047]
posible solución:
                    addi x7,x1,-4000
                    lw x8,0(x7)
```

Problema 5.
Escriba un programa que borre el registro x2, sume 3 unidades al registro x2 (10 veces) y copie x2 en x1.

```
Solución:

add x2,x0,x0
addi x3,x0,10
addi x2,x2,3
addi x3,x3,-1
bne x3,x0,8
add x1,x2,x0
stop
```

Problema 6.
Escriba un programa que comprueba si el número entero almacenado en la dirección 0x000000e0 es cero. Si es así, escribe el dato 0xffffffff en esa misma dirección.

```
Solución:
lw x1,0xe0(x0)
bne x1,x0,0x10
addi x1,x0,-1
sw x1,0xe0(x0)
stop
```

Problema 7.
A partir de la dirección 0x00000000 se encuentra almacenada una tabla de números enteros.

a) Escriba un fragmento de programa que calcule la suma de los 2 primeros elementos de la tabla y escriba el resultado en la dirección 0xe0
b) Escriba un fragmento de programa que calcule la suma de los 10 primeros elementos de la tabla y escriba el resultado en la dirección 0xf0.

```
Solución:
a)
lw x1,0(x0)
lw x2,4(x0)
add x3,x1,x2
sw x3,0xe0(x0)

b)
addi x1,x0,10   (x1 será el contador de 10 elementos)
addi x2,x0,0    (x2 va a contener la diección de memoria del dato actual)
addi x15,x0,0   (x15 acumulará la suma)
lw x3,0(x2)
add x15,x15,x3
addi x2,x2,4
addi x1,x1,-1
bne x1,x0,0xc
sw x15,0xf0(x0)
```

Problema 8.
Escriba un fragmento de programa que escriba el dato 0xff en las primeras 32 palabras de la memoria.

```
Solución:
addi x1,x0,32   (x1 contador)
addi x2,x0,0    (x2 puntero)
addi x3,x0,0xff (x3 dato a escribir)
sw x3,0(x2)
addi x2,x2,4
addi x1,x1,-1
bne x1,x0,0xc
```

Problema 9.
Escriba un programa que escriba los 20 primeros números enteros impares positivos a partir de la dirección 0x00000000.

```
Solución:
addi x1,x0,20   (x1 contador)
addi x2,x0,0    (x2 puntero)
addi x3,x0,1    (x3 dato a escribir, 1 primer impar)
sw x3,0(x2)
addi x3,x3,2    (siguiente impar)
addi x2,x2,4
addi x1,x1,-1
bne x1,x0,0xc
stop
```

Problema 10.
Escriba un programa que traslade una tabla de 20 números enteros desde su ubicación inicial en la posición 0x0 a una nueva ubicación en la posición 0x200.

```
Solución:
addi x1,x0,20   (x1 contador)
addi x2,x0,0    (x2 puntero 1)
addi x3,x0,0x200 (x3 puntero 2)
lw x4, 0(x2)
sw x4,0(x3)
addi x2,x2,4
addi x3,x3,4
addi x1,x1,-1
bne x1,x0,0xc
stop
```

Problema 11.
A partir de la dirección 0x0 se encuentran almacenada una lista de 16 números enteros. Escriba un fragmento de programa que cargue en el registro x5 el menor de los elementos de la lista.

```
Solución:
addi x1,x0,16   (x1 contador)
addi x2,x0,0    (x2 puntero 1)
lw x5,0(x2)     (x5 almacena el menor hasta el momento)
addi x2,x2,4
addi x1,x1,-1
beq x1,x0,0x24
lw x3,0(x1)
blt x3,x5,8
beq x0,x0,0xc
stop
```

Problema 12.
Considere el siguiente programa para el RISC Y y la porción de la memoria de datos que se muestra. Tras cada instrucción muestre el valor que van tomando los registros. Señale también los cambios en la memoria de datos.

```
lw x4, 0x704(x0)
xori x5, x4, 0xfff
sw x5, 0x704(x0)
ori x4, x5, 0xf00
sw x4, 0x710(x0)
slli x5, x4, 20
srai x4, x5, 28
lw x5, 0x70c(x0)
srli x4, x5, 28
stop
```

POS	CONTENIDO
0x700	0xcafecafe
0x704	0xabcdef06
0x708	0x456caacb
0x70c	0x8765411a
0x710	0xa523ef10

Solución:

```
lw x4, 0x704(x0)    → x4 = 1010 1011 1100 1101 1110 1111 0000 0110 =
                         = 0xabcdef06
xori x5, x4, 0xfff → x5 = 0101 0100 0011 0010 0001 0000 1111 1001 =
                         = 0x543210f9  (0xfff se extiende en signo)
sw x5, 0x704(x0)    → memdat(0x704)= 0101 0100 0011 0010 0001 0000 1111 1001=
                         = 0x543210f9
ori x4, x5, 0xf00   → x4 = 1111 1111 1111 1111 1111 1111 1111 1001 =
                         = 0xfffffff9
sw x4, 0x710(x0)    → memdat(0x710) =1111 1111 1111 1111 1111 1111 1111 1001=
                         = 0xfffffff9
slli x5, x4, 20     → x5 = 1111 1111 1001 0000 0000 0000 0000 0000 =
                         = 0xff900000
srai x4, x5, 28     → x4 = 1111 1111 1111 1111 1111 1111 1111 1111 =
                         = 0xffffffff
lw x5, 0xf0c(x0)    → x5 = 1000 0111 0110 0101 0100 0001 0001 1010 =
                         = 0x8765411a
srli x4, x5, 28     → x4 = 0000 0000 0000 0000 0000 0000 0000 1000 =
                         = 0x8
```

Problema 13.

Realice un programa que calcule el Ca1 y el Ca2 de 27 y los almacene en las direcciones 0xa0 y 0xa4 de la memoria de datos respectivamente.

Solución:

Haremos las siguientes operaciones:

x14 ← 27

x15 ← Ca1(27), memdat(0xa0) <- x15

x10 ← Ca2(27), memdat(0xa4) <- x14

Para ello las instrucciones serán:

```
addi x14,x0,27                              27 = 0b11011=0x0000001b
xori x15, x14, -1
sw x15,(0xa0)x0
addi x10,x15,1          (también podría ser sub x10,x0,x14)
sw x10,(0xa4)x0
stop
```

explicación: xori x15,x14,-1 hace x15 ← x14 xor 0xffffffff (= not x14) = ca1 (x14)

addi x10,x15,1 hace x10 ← x15 + 1 = ca1(x14) +1 = ca2 (x14)

sub x10,x0,x14 hace x10 ← 0-x14= ca2(x14)

Problema 14.
Realice un programa que calcule el producto MXN de dos números positivos M y N.

Solución:
Para realizar el programa sumaremos N tantas veces como indique M. Para ello escribiremos N en x15, M en x7, y acumularemos la suma en x14.

Haremos el ejemplo con M=16 y N=5

```
      addi x14,x0,0
      addi x15,x0,16    pongo x15 a 16 (N)
      addi x7,x0,5      pongo x7 a 5  (M)
      add  x14,x15,x14
      addi x7,x7,-1
      bne  x7,x0,0xc
      stop
```

Problema 15.
Realice un programa que calcule N términos de la sucesión de Fibonacci.
0, 1, 1, 2, 3, 5, 8, 13, 21, …

Solución:
Escribimos N en X15, por ejemplo N=7

```
      addi x15,x0,7
      addi x6,x15,0    copio X15 a X6 (será el contador de elementos)
      addi x13,x0,0           inicializo el primer elto
      addi x14,x0,1           inicializo el segundo elto
(ahora empieza el bucle)
      sw x14,0xf4(x0)    uso la memoria para almacenar x14 que es a[i-1],
                         para que después sea a[i-2] (es decir x13)
      add x14,x14,x13      aqui hago a(i) = a(i-1)+a(i-2)
                           (sobreescribo x14 pero ya lo he salvado antes)
      lw x13,0xf4(x0)    Meto el antiguo x14 en x13 para que ahora sea a[i-2]
                           y el nuevo x14 ya es a[i-1]
      addi x6,x6,-1
      bne x6,x0,0x10
```

Problema 16.
Se quiere incorporar al juego de instrucciones del RISCY la instrucción de salto incondicional, su sintaxis es: "j inm 12" y su operación a nivel RT es:

"pc ← sext(inm12)".

Determine si esto es posible y en caso de que sea necesario indique qué cambios implicaría en la unidad de datos. Proporcione también el formato que asignará a la nueva instrucción, su código de operación y la secuencia de microoperaciones implicadas.

Solución:
Sí, es posible incorporar esta instrucción. No será necesario realizar modificaciones a la unidad de datos pues existe un camino de entrada a PC desde la ALU, camino que ya se utiliza para los saltos condicionales.

El formato que podemos utilizar es el de los saltos condicionales (S) donde no necesitaremos utilizar los códigos de función ni los campos asociados a registros. El código de operación puede ser cualquiera de los no usados, pr ejemplo, el 0000 1000.

Las microinstrucciones asociadas son:

1. IR ← CODMEM(PC), PC ← PC + 4 WIR,IPC (búsqueda)
2. PC ← sext(inm12) WPC,sel_inm,sel_op$_1$ (ejecución)

Problema 17.
Se quiere incorporar al juego de instrucciones del RISCY la instrucción de carga de dato inmediato, su sintaxis es: "li rd, inm12" y su operación a nivel RT es:

"rd ← sext(inm12)".

Determine si esto es posible y en caso de que sea necesario indique qué cambios implicaría en la unidad de datos. Proporcione también el formato que asignará a la nueva instrucción, su código de operación y la secuencia de microoperaciones implicadas.

Solución:
Sí, es posible incorporar esta instrucción. No será necesario realizar modificaciones a la unidad de datos pues existe un camino para llevar el dato extendido en signo hasta rd a través de la ALU, camino que ya se utiliza para las operaciones que usan dato inmediato como addi o xori.

El formato que podemos utilizar es el de las instrucciones con dato inmediato (I) donde no necesitaremos utilizar el campo asociado al registro fuente. El código de operación puede ser cualquiera de los no usados, por ejemplo, el 1000 1000.

Las microinstrucciones asociadas son:

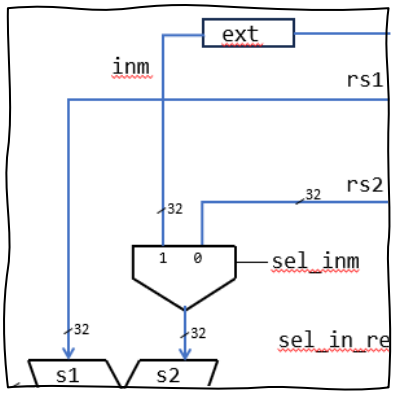
1. IR ← CODMEM(PC), PC ← PC + 4 WIR,IPC (búsqueda)
2. rd ← sext(inm12) WREG,sel_inm,sel_op$_1$

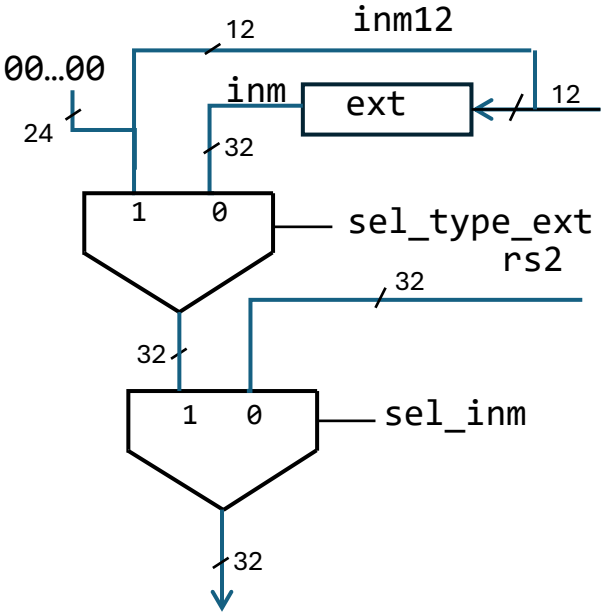
Problema 18.
En los problemas 16 y 17, así como en todas las instrucciones que trabajan con dato inmediato, el dato de 12 bits inm12 es extendido en signo hasta 32 bits, ¿qué modificación habría que incorporar en el RISCY para que en las instrucciones no aritméticas el dato inmediato se extendiera con 0 y no con el bit de signo?

Solución:

Opción 1.

Es necesario añadir un nuevo MUX2:1 que seleccionara entre inm extendido en signo o inm extendido en 0.
En la carta ASM para las instrucciones que precisaran la extensión con 0 habría que activar sel_type_ext. Para las instrucciones aritméticas no habría que modificar nada, sus microoperaciones seguirían siendo las mismas (ya que sel_type_ext no activada hace que la unidad de datos funcione como si el MUX no se hubiera introducido)

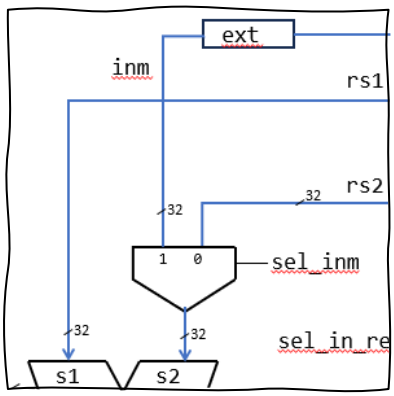


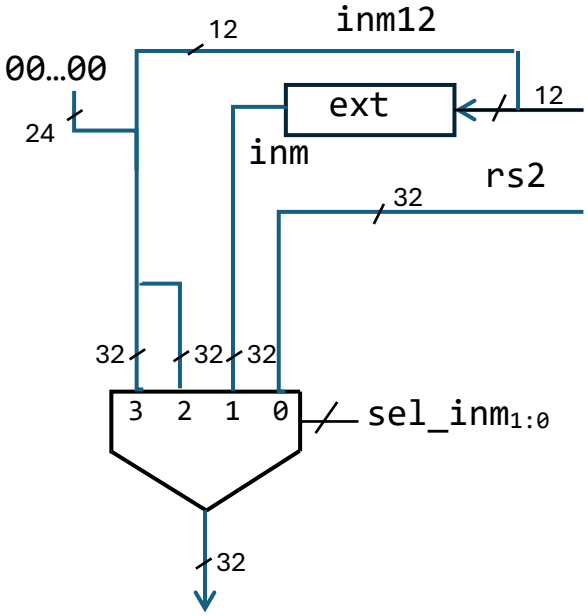


Opción 2 – NO recomendable.
Podríamos optar por modificar el multiplexor que está conectado a la entrada s2 de la ALU, para permitir una entrada más (un MUX4:1).

Esa entrada, de 32 bits, estaría conectada a la palabra {24'd0,$inm_{11:0}$).
La señal sel_inm debería ser una señal de dos bits: $sel\_inm_{1:0}$

Esta opción obliga a hacer cambios sobre instrucciones que ya teníamos antes de la modificación pues hemos cambiado la señal sel_inm de 1 bit a 2 bits. Es por esto que esta opción NO es adecuada. Intentaremos siempre hacer los menos cambios posibles y que estos no afecten a instrucciones que ya habían sido implementadas correctamente.


ext
inm
rs1
rs2
32
32
1 0
sel_inm
32
32
sel_in_re
s1
s2


00…00
12
inm12
24
ext
12
inm
rs2
32
32
32
32
3 2 1 0
sel_inm1:0
32