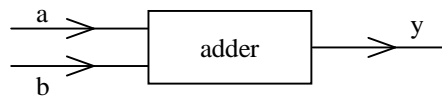
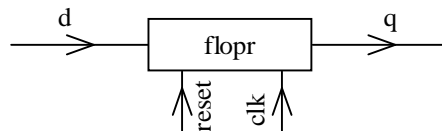


Para todos los ejercicios se pide que respeten los nombres de las entidades, entradas y salidas. Representación de datos siempre en *std\_logic* o *std\_logic\_vector*. Internamente, se les da la libertad para que utilicen el diseño que consideren más adecuado siempre y cuando la salida sea la esperada.

Ejercicio 1) Diseñe un sumador de 32 *bits* en VHDL según el diagrama.

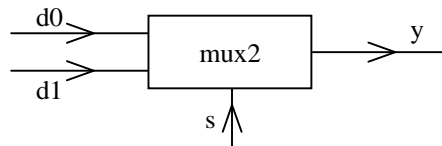


Ejercicio 2) Diseñe un *flipflop* con *reset* asíncrono de 32 *bits* en VHDL según el diagrama.



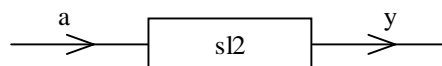
Investigue como puede escribir código genérico en VHDL, utilícelo para generalizar su diseño para registros de *N bits*. (ej. *N*=1, *N*=5, *N*=32,...).

Ejercicio 3) Diseñe un multiplexor de dos entradas de 32 *bits* según el diagrama.

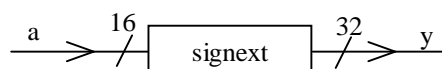


[Recuerde el funcionamiento del multiplexor: Si *s* = 0 => *y* = *d0*; si *s* = 1 => *y* = *d1*.]

Ejercicio 4) Diseñe un desplazador de 2 *bits* a izquierda (entrada y salida de 32 *bits*) según el diagrama.



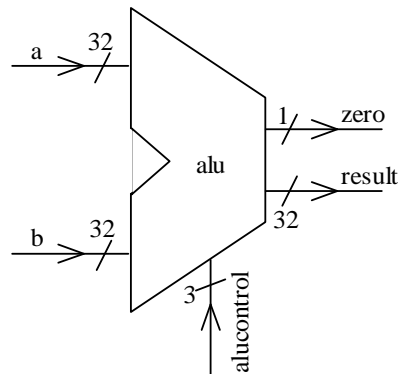
Ejercicio 5) Diseñe un módulo que tome un vector de 16 *bits* y lo convierta en un vector de 32 *bits*. Los primeros 16 *bits* del nuevo vector deben completarse con '1' o '0' dependiendo del valor del bit más significativo del vector original. Respete *In/Out* según el diagrama.



# Arquitectura de Computadoras 2013

## Práctico N° 1

**Ejercicio 6)** Diseñe una ALU según el diagrama y la tabla.

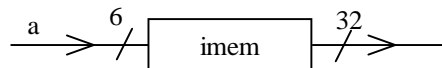


<i>alucontrol</i>	<i>result</i>
000	a and b
001	a or b
010	a + b
011	Sin uso
100	a and not(b)
101	a or not(b)
110	a - b
111	slt

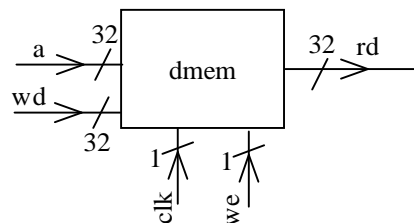
**Notas:**

- *zero* = 1, cuando *result* = 0.
- Slt (*set less than*): si  $a < b$ , *result* = 1 cc, *result* = 0.

**Ejercicio 7)** Diseñe una memoria ROM que pueda direccionar 64 palabras de 32 bits según el diagrama.

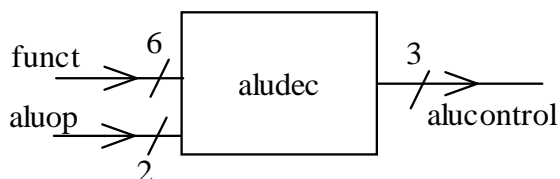


**Ejercicio 8)** Diseñe una memoria RAM según el diagrama. (Internamente descarte los *bits* 32..8 y 1..0; es decir, solo direcciona 64 palabras según los *bits* 7 a 2 de la dirección.



**Nota:** el dato contenido en 'wd' va a guardarse en la dirección 'a' siempre que 'we' sea uno.

**Ejercicio 9)** Diseñe un módulo decodificador según el diagrama y la tabla.



<i>aluop</i>	<i>funct</i>	<i>alucontrol</i>
00	-----	010
01	-----	110
1-	100000	010
1-	100010	110
1-	100100	000
1-	100101	001
1-	101010	111