

Integración de la ALU en la CPU teórica

Área de Arquitectura y Tecnología de Computadores – Versión 1.0.894, 15/02/2022

Índice

Objetivos

Conocimientos y materiales necesarios

1. Introducción
 2. Registros
 3. Camino de datos
 4. Ejercicios adicionales
-

Objetivos

El objetivo de esta sesión práctica es la simulación por parte del alumno de operaciones aritméticas y lógicas dentro de la CPU del Computador Teórico (CT). Para no complicar en exceso la simulación se considera una CPU de sólo 4 bits, pero totalmente análoga a la del CT. Además, sólo se consideran los siguientes elementos de la CPU:


- Bus interno.
- ALU.
- Registro temporal de entrada.
- Registro temporal de salida.
- Registro de estado.
- Dos registros de propósito general.

Después de realizar la sesión práctica el alumno debería ser capaz de definir la secuencia de señales necesaria para realizar cualquier operación aritmética o lógica empleando la CPU del CT.

Conocimientos y materiales necesarios

Para aprovechar adecuadamente esta sesión de prácticas, el alumno necesita:

- Comprender el funcionamiento de los biestables D, los *buffers* triestado, los registros y la ALU del CT.
- Llevar a clase una memoria USB, o dispositivo análogo, para almacenar los circuitos que se desarrollarán en la práctica.

Durante la sesión se plantearán una serie de preguntas que deben responderse en el correspondiente [cuestionario](https://www.campusvirtual.uniovi.es/mod/quiz/view.php?id=143926) (<https://www.campusvirtual.uniovi.es/mod/quiz/view.php?id=143926>) del Campus Virtual. El cuestionario se puede abrir en otra pestaña del navegador pinchando en el enlace mientras se mantiene pulsada la tecla .

1. Introducción

En esta sesión se pretende simular lo que se denomina «el camino de datos» de la CPU del CT, cuyo circuito se muestra en la figura 1. El camino de datos está compuesto por una serie de elementos que permitirán almacenar datos en unos registros denominados «registros de propósito general» y hacer operaciones aritméticas y lógicas con los datos utilizando la ALU. Para conectar los registros con la ALU se utilizará un bus, denominado «bus interno» (IB , de *Internal Bus*). Un bus es un conjunto de líneas que funcionan de forma coordinada. En nuestro caso, serán cuatro líneas

ya que tanto los registros como la ALU serán de 4 bits. Como la ALU tiene dos entradas y no se pueden poner dos valores a la vez en un bus, será necesario introducir un registro temporal de entrada (TMPE) para almacenar uno de los datos. También será necesario un registro temporal de salida (TMPS) para almacenar el resultado de la ALU sin sobrescribir uno de los valores de entrada que se están utilizando. El último elemento que utilizaremos será otro registro, denominado «registro de estado» (SR , de *Status Register*) para almacenar los valores de los bits de estado (*flags*) calculados por la ALU.

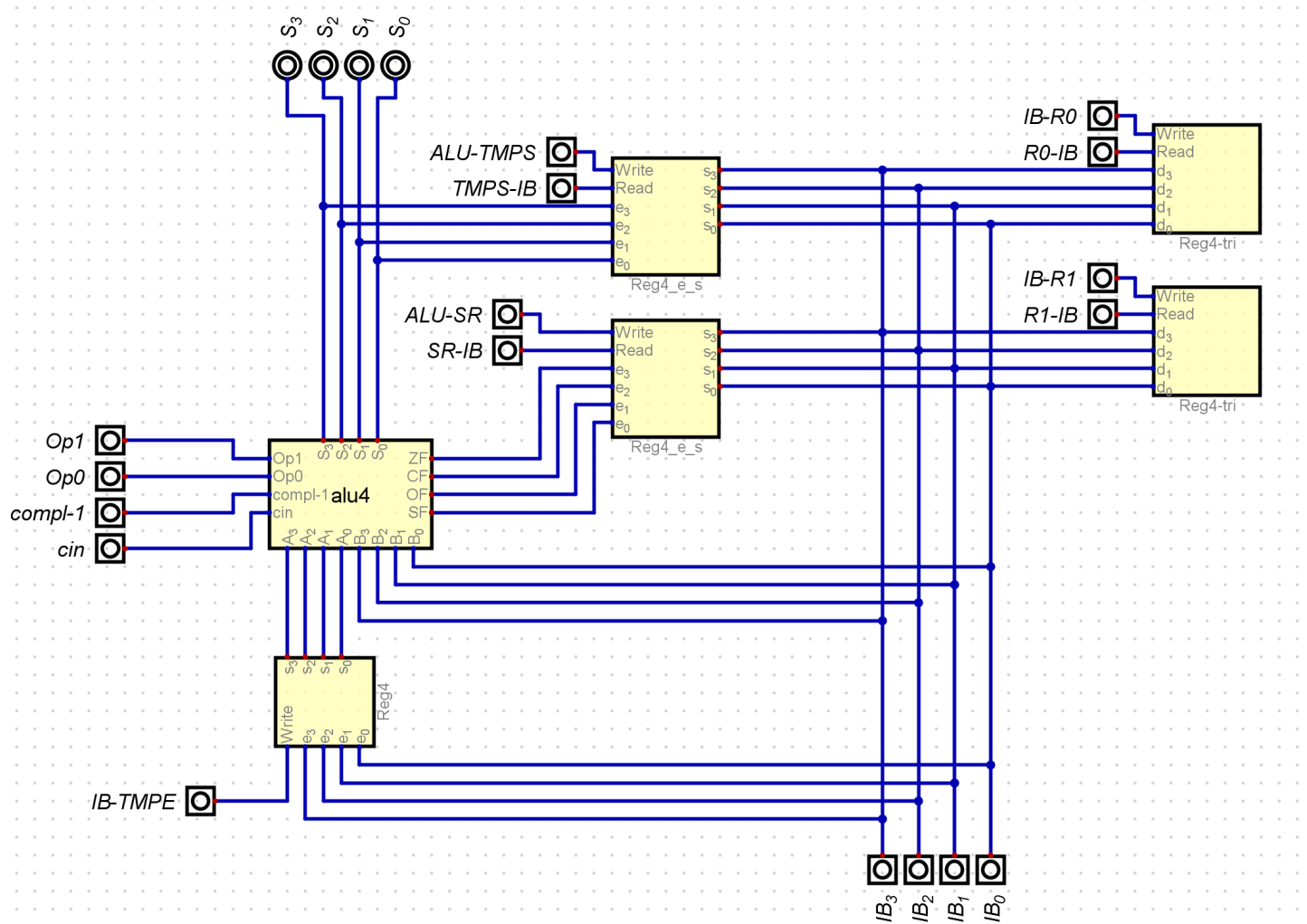


Figura 1. Camino de datos del CT

Fíjate que, en este circuito, las entradas conectadas a las entradas de control de los registros no se han llamado Read y Write, sino con unos nombres que indican cómo se mueve la información al activar esa entrada. Por ejemplo, la señal Read del registro R0, como vuelca su valor al IB, se ha llamado R0-IB y la señal Write de este registro, como escribe en el registro el valor que esté en IB, se ha llamado IB-R0. ¿Cómo se llama la entrada que hace que se almacene en el registro TMPS el valor generado por la ALU? Responde en el [cuestionario](#)

(<https://www.campusvirtual.uniovi.es/mod/quiz/view.php?id=143926>): pregunta 1

Como verás en el tema 5, estas entradas son señales de control que no son generadas por un humano, sino por otro circuito, denominado «Unidad de Control».

2. Registros

Para almacenar información dentro de la CPU se utilizan registros, pero no todos son iguales:

- Los registros de propósito general (R0 y R1) tienen líneas de datos d_i que actúan tanto de entrada como de salida y están conectadas al bus interno. Se utilizarán registros como los vistos en teoría.

- Los registros SR y TMPS, en cambio, tienen entradas y salidas separadas: las entradas están conectadas a salidas de la ALU (las de SR a los bits de estado y las de TMPS a las salidas de la operación) y sus salidas conectadas al bus interno.
- El registro TMPE tiene sus entradas y salidas separadas (las entradas conectadas al bus interno y las salidas a la entrada A de la ALU) pero, además, tiene la particularidad de que no hay otro elemento conectado a sus salidas (el resto de registros están conectados a un bus con otros registros), con lo que no necesita un *buffer* triestado en sus salidas.

En estas prácticas vamos a crear los registros de propósito general como los estudiados en teoría; los circuitos del resto de registros te serán proporcionados.



- Descarga el archivo `datapath.zip` y descomprímelo a tu carpeta de trabajo.
- Abre el simulador de circuitos digitales y crea un registro como el mostrado en la figura 2. Fíjate que usa biestables D (que puedes insertar a través de la opción **Componentes** > **Flip-Flops**) y *buffers* triestado (denominados «Drivers» en la opción **Componentes** > **Cables**). Como las entradas d_i tienen que actuar de entrada o salida en función de si la señal Read o Write está activa, tendrás que modificar sus opciones y en la solapa **Avanzado** seleccionar, tal y como se muestra en la figura 3, «Es una entrada de tres estados» y poner como valor por defecto «Z», que representa un estado de alta impedancia equivalente a no conectar ese elemento.
- Cambia el orden de las entradas (**Editar** > **Ordenar las entradas**) para que sea WRITE, READ, d_3 , d_2 , d_1 y d_0 .
- Guarda el circuito con el nombre `reg4-tri.dig` en la carpeta donde has descomprimido `datapath.zip`.

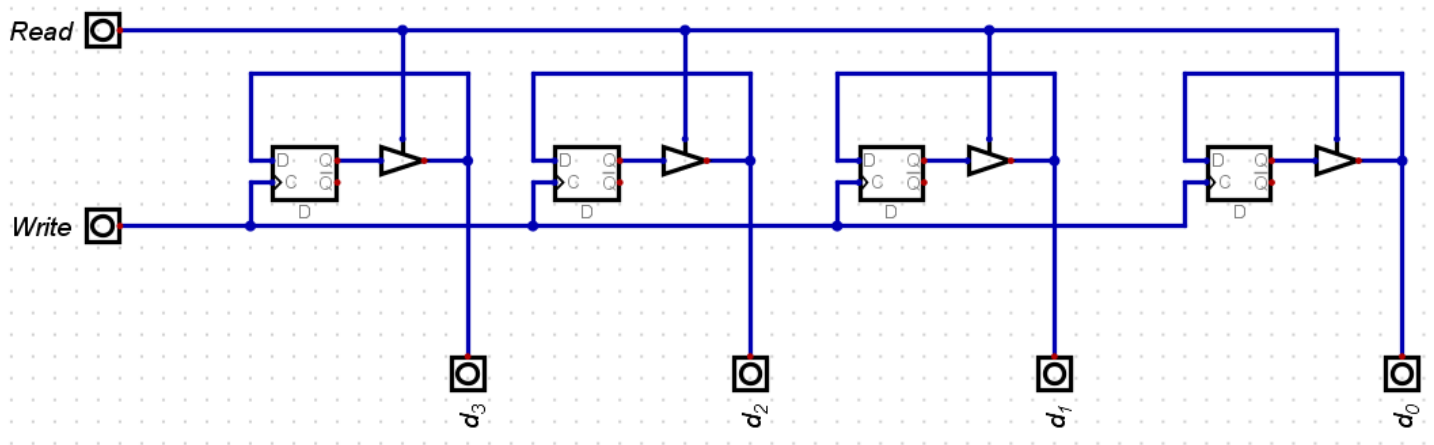


Figura 2. Registro que usa líneas de datos que funcionan como entrada y salida.

Figura 3. Opciones para las entradas tri-estado.

3. Camino de datos



- Carga en el simulador el proyecto `datapath.dig`.
- Complétalo añadiendo los componentes necesarios. Fíjate en que, aparte del registro que acabas de crear (`reg4-tri.dig`), hay otros dos tipos de registros: `reg4_e_s.dig`, que tiene entradas y salidas separadas, y `reg4.dig`, que además de tener entradas y salidas separadas, no utiliza un *buffer* triestado. Debes seleccionar el adecuado para cada registro. Además, es posible que debas rotar alguno de estos registros.

Ahora vamos a hacer una operación de carga (escritura) de un registro y vamos a comprobar que almacena correctamente su valor. En concreto, vamos a almacenar el valor 5 en el registro de propósito general superior (R0):



- Lanza la simulación. Fíjate en que algunos cables están en gris: si sitúas el ratón sobre uno de ellos, podrás comprobar que se corresponde con el estado de alta impedancia (Z). Las entradas a las que están conectados estos cables pueden ahora cambiarse entre tres estados: 0, 1 y Z.
- Sitúa el valor 5, codificado en binario natural con cuatro bits, en las entradas IB₃, IB₂, IB₁ e IB₀.
- Cambia el valor de la entrada IB-R0 de 0 a 1 para que se produzca un flanco de subida en los biestables D del registro R0. Eso hará que este registro tome el valor en sus líneas de datos y lo almacene en sus biestables.
- Pon a cero la entrada IB-R0 y en alta impedancia (color gris) las entradas IB₃, IB₂, IB₁ e IB₀ del bus interno. El bus en estos momentos no tiene ningún valor.
- Vamos a volcar el valor almacenado en R0 al bus, es decir, vamos a leer el registro. Para ello, pon a 1 la entrada R0-IB, que está conectada a la entrada Read del registro R0, que a su vez está conectada a sus *buffers* triestado, con lo que estos van a llevar la salida de sus biestables a las líneas d. Deberías ver en las líneas del bus interno el valor 5 que habías almacenado en R0.
- Detén la simulación.



Fíjate que en este tipo de operaciones con circuitos digitales **síncronos**, el orden de las operaciones es muy importante. Si cometes cualquier error, tienes que empezar a repetir todos los pasos desde el principio.

A continuación vamos a hacer una operación de resta:



- Analiza cómo se haría la resta de los operandos A=5 y B=4 en papel, tal y como se muestra en la figura 4.

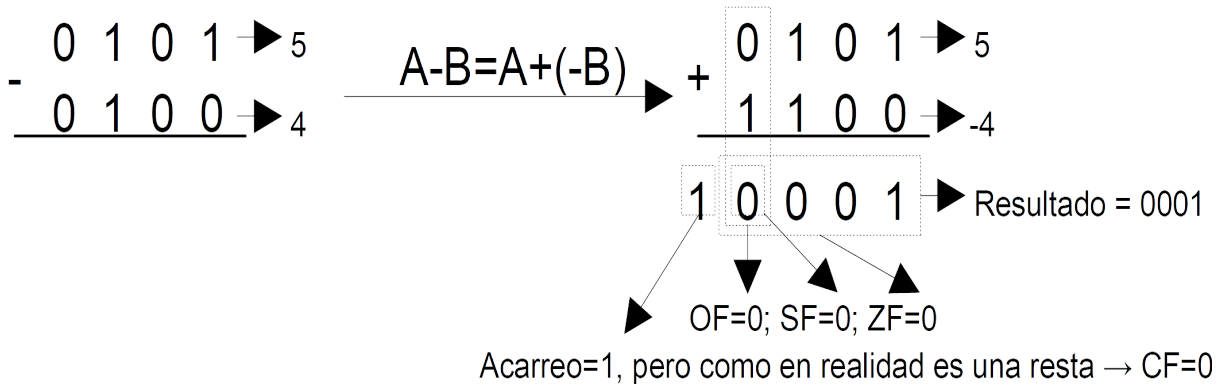


Figura 4. Resta de 5 - 4.

Seguidamente, vamos a simular esta operación con la herramienta. En este tipo de sesiones prácticas en las cuales se simulan circuitos digitales, es fundamental que trates de predecir el resultado de la simulación **antes** de llevarla a cabo. A continuación, debes comparar tu predicción con el resultado proporcionado por la simulación y extraer tus propias conclusiones.

Por ejemplo, si tu predicción no coincide con el resultado de la simulación podría suceder que no tengas claros los conceptos teóricos relacionados, o bien que hayas diseñado mal el circuito. En cualquier caso, debes dedicar el tiempo que sea necesario hasta conseguir que tus predicciones coincidan con los resultados de simulación.

En particular, en esta sesión práctica debes predecir correctamente el resultado y bits de estado correspondientes a cualquier operación **antes de realizar simulación alguna**. Además, con cada operación debes predecir correctamente el estado que tendrán todos los elementos del circuito **justo después de llevar a cabo cualquiera de los pasos anteriores**.

En primer lugar vamos a cargar los valores a restar en R0 y en R1, luego vamos a hacer la resta y vamos a llevar el resultado a R0, es decir, vamos a hacer $R0=R0-R1$. En cada uno de los pasos siguientes, si no se indica que una señal tiene que estar a 1 (activa), debería estar a 0. Si no lo haces así, se pueden producir cortocircuitos si se generan distintos valores sobre el mismo cable (pasaría por ejemplo si se activa R0-IB y R1-IB al mismo tiempo).



- Lanza la simulación.
- Carga, siguiendo los mismos pasos que hiciste antes, el valor 5 en `R0` . No hace falta que hagas la lectura.
- Carga el valor 4 en `R1` .
- Vuelca el valor de `R0` en el bus interno (`R0-IB`) y activa la entrada `IB-TMPE` para que se almacene en el registro `TMPE` . Fíjate que a la salida de este registro debería estar el mismo valor que se almacenó en `R0` .
- Vuelca el valor de `R1` en el bus interno y activa las señales, `Op1` , `Op0` , `compl-1` y `cin` de la ALU para que haga la operación de resta y, al mismo tiempo, activa las señales `ALU-TMPS` y `ALU-SR` para que se almacene el resultado y los bits de estado en `TMPS` y `SR` respectivamente. ¿Qué valor se almacena en `TMPS` ? Responde en el [cuestionario](https://www.campusvirtual.uniovi.es/mod/quiz/view.php?id=143926) (<https://www.campusvirtual.uniovi.es/mod/quiz/view.php?id=143926>): pregunta 2 ¿Y en `SR` ? Responde en el [cuestionario](https://www.campusvirtual.uniovi.es/mod/quiz/view.php?id=143926) (<https://www.campusvirtual.uniovi.es/mod/quiz/view.php?id=143926>): pregunta 3
- Ahora debes almacenar el valor de `TMPS` en `R0` . Para ello, vuelca el valor de `TMPS` al bus interno y lee ese valor desde el bus interno a `R0` .
- Por último, vuelca el valor de `R0` al bus interno para comprobar que se almacenó correctamente. ¿Qué valor aparece en el bus interno? Responde en el [cuestionario](https://www.campusvirtual.uniovi.es/mod/quiz/view.php?id=143926) (<https://www.campusvirtual.uniovi.es/mod/quiz/view.php?id=143926>): pregunta 4
- Detén la simulación.

4. Ejercicios adicionales

Estos ejercicios adicionales permiten al alumno reforzar los conocimientos adquiridos durante la sesión práctica.



- Para practicar la transferencia entre registros, carga el registro R0 con un valor a través de las entradas IB₀, IB₁, IB₂ e IB₃, y luego genera las señales necesarias para copiar la información de R0 a R1. Comprueba que R1 tiene el valor adecuado.
- Realiza las operaciones recogidas en la tabla 1 «en papel», es decir, sin utilizar el simulador. En el caso de las operaciones lógicas el valor de los bits de estado *carry* y *overflow*, aunque la ALU los genera, no son significativos y deben ignorarse.

Tabla 1. Operaciones propuestas

R0	R1	Operación	Resultado	ZF	CF	OF	SF
-3	7	OR					
5	3	AND					
-6	-8	AND					
4	-6	Suma					
-3	7	Resta					
-1	-5	XOR					
5	-4	AND					

- Lleva a cabo las operaciones sobre el simulador y verifica que tus respuestas son correctas.
- Abre los circuitos de los registros TMPE y TMPS y observa cómo están hechos. Estudia en qué se diferencian entre sí y con el circuito para los registros de propósito general. Analiza qué diferencias hay en su funcionamiento.