

# Introducción al simulador del Computador Teórico

Área de Arquitectura y Tecnología de Computadores – Versión 1.0.899, 18/02/2022

## Objetivos

En esta sesión se pretende que el alumno se familiarice con el uso del simulador del Computador Teórico.

En esta primera toma de contacto se aprenderá a manejar los registros y la memoria, se estudiará el funcionamiento básico del Computador Teórico (codificación y ejecución de instrucciones) y se introducirán dos de los tipos de archivos que puede manejar la aplicación: archivos de simulación y archivos de memoria.

Es importante que comprendas lo que está ocurriendo cada vez que realizas una acción y que puedas prever el resultado de las mismas antes de que llevarlas a cabo.

## Conocimientos y materiales necesarios


Para poder realizar esta sesión el alumno debe:

- Acudir al laboratorio con una copia del juego de instrucciones de la CPU que ya se ha suministrado en las clases teóricas. También debe conocer perfectamente el manejo de la tabla de codificación de las instrucciones de la CPU.
- Conocer el funcionamiento básico de una arquitectura Von Neumann.
- Durante la sesión se plantearán una serie de preguntas que puedes responder en el correspondiente [cuestionario](#) en el Campus Virtual. Puedes abrir el cuestionario en otra pestaña del navegador pinchando en el enlace mientras mantienes pulsada la tecla `Ctrl`.

## 1. Ejecución de instrucciones

Siguiendo las instrucciones de tu profesor de prácticas, arranca el programa simulador del Computador Teórico. Cuando lo hayas hecho, aparecerá la pantalla principal de la aplicación, en la que verás un diagrama de bloques de los principales elementos de una arquitectura Von Neumann: ALU, registros, memoria, módulos de E/S y buses. Los números de cuatro cifras que aparecen en diversos controles son números hexadecimales. Fíjate en los registros: todos están inicializados al valor `0000h` excepto uno. ¿Cuál es?


Vamos a modificar los registros `r3` y `r5` para que tengan los valores `2FE3h` y `1A5Bh` respectivamente. Para ello:



- Pulsa sobre el registro `r3`.
- En la ventana que aparecerá introduce el valor `2FE3h` y luego pulsa el botón `Aceptar` (no es necesario añadir el terminador `h` porque el simulador trabaja sólo con datos hexadecimales.)
- Siguiendo los mismos pasos, modifica el registro `r5` para que tenga el valor `1A5Bh`.


Fíjate en que los únicos valores que puedes modificar directamente de esta forma son los de los registros `r0` - `r7` y `PC`. El valor del resto de los registros (`MAR`, `MDR`, `IR` ...) lo asigna la aplicación durante la simulación.

Vamos a trabajar ahora con la memoria. Como sabes, la memoria se utiliza para almacenar instrucciones y datos. Vamos a utilizar la memoria para almacenar una instrucción: `mov r5, r3`.



- En primer lugar debes codificarla utilizando la tabla de codificación de instrucciones de la CPU. Tras la codificación obtendrás un número hexadecimal de 4 cifras.
- Vamos a colocar ese número en la posición de memoria a la que apunte `PC`. Mira el valor de `PC` y anótalo.
- Pulsa con el ratón sobre el dibujo etiquetado con **MEMORIA** (abajo a la izquierda). Aparecerá un menú en el que debes seleccionar la opción *Editor hexadecimal*.
- Se abrirá una ventana en la que se muestra dos listas de números. A la izquierda aparecen las direcciones de memoria y a la derecha el contenido de cada dirección. Se puede utilizar la barra de scroll para desplazarse por la memoria, pero para ir a una dirección concreta es más fácil utilizar el botón `Ir a dirección...`. Pulsa sobre él.
- En la pantalla que te aparecerá escribe el valor que habías apuntado del `PC`.
- Ahora en la fila superior del editor hexadecimal estarás viendo el contenido de la dirección de memoria a la que apunta el `PC`. Como es ahí donde queremos que esté la codificación de la instrucción `mov r5, r3`, haz doble clic sobre ese número y, en la ventana que verás, escribe la instrucción codificada.

Vamos a comprobar que has codificado bien la instrucción:



- Cierra la ventana del editor hexadecimal pulsando el botón `Aceptar`.
- Vuelve a pulsar sobre el dibujo de la memoria pero ahora escoge la opción *Desensamblador* en el menú que te sale. Aparecerá una ventana muy similar a la del editor hexadecimal pero con una columna más para cada posición de memoria que contiene el mnemónico de la instrucción que se corresponde con el contenido de cada posición de memoria.
- Vete a la posición a la que apunta `PC` y comprueba que el mnemónico y los operandos de la instrucción en esa posición son los que deseamos: `mov r5, r3`. Si no es así, repasa la codificación que has hecho, corrígela e introduce la nueva codificación en la posición a la que apunta `PC` (deberás usar de nuevo el editor hexadecimal).

Cuando llegues a este punto, `r3` tendrá el valor `2FE3h`, `r5` el valor `1A5Bh` y `PC` estará apuntando a la posición de memoria donde estará la codificación de la instrucción `mov r5, r3`. Vamos a hacer que la CPU ejecute esta instrucción.


Recuerda que la ejecución de una instrucción consiste en traer esa instrucción de memoria, decodificarla en `IR` y hacer lo que indique su codificación. La instrucción que se trae es precisamente aquella que haya en la dirección donde apunta `PC`; por lo tanto, en cuanto le digamos al simulador que ejecute una instrucción ejecutará exactamente la que queremos que ejecute. Antes de hacerlo, contesta a las siguientes preguntas:

- ¿Qué valor tendrán `r5` y `r3` después de la ejecución? Responde en el [cuestionario](#): pregunta 1.
- ¿Qué valor tendrá `IR`? Responde en el [cuestionario](#): pregunta 2.
- ¿Qué valor tendrá `PC`? Responde en el [cuestionario](#): pregunta 3.

Para ejecutar una instrucción puedes ir al menú **Ejecución > Instrucción completa**, que como verás tiene asignada la tecla rápida `F8`. Escoge esa opción. ¿Han quedado los registros como habías previsto?

## 2. Archivos .sim

El simulador permite en cualquier momento almacenar el estado de una simulación. Para ello utiliza archivos con extensión `.sim`. Vamos a probar esta característica.



- Escoge la opción de menú **Archivo > Guardar como**. Te saldrá el cuadro de diálogo típico para guardar. Escoge tu carpeta de trabajo y escribe el nombre `4-1cpu`. El programa le añadirá la extensión `.sim`.
- Fíjate en el valor de todos los registros de la CPU.
- Para comprobar que el archivo guarda correctamente los datos, vamos a reiniciar la CPU. En el menú del simulador, escoge la opción **Ejecución > Reiniciar**.
- Fíjate en que todos los registros han vuelto a cero, excepto `PC`, que tiene el valor que tenía inicialmente. Puedes comprobar también que la memoria tiene ceros donde habías puesto las instrucciones.
- Ahora vamos a hacer que el simulador vuelva al estado en el que se encontraba cuando guardamos el archivo `.sim`. Escoge la opción del menú **Archivo > Abrir**. Te saldrá el cuadro de diálogo típico para abrir un archivo. Escoge tu carpeta de trabajo y el tipo de archivo `.sim`. Elige a continuación el archivo `4-1cpu`.
- Fíjate que se ha modificado el estado de la CPU para que sea exactamente el mismo que cuando guardaste el archivo. Además de los registros, comprueba que en la memoria están las instrucciones que introdujiste al principio de la práctica.

Los archivos `.sim` son muy útiles para recuperar el estado total de la simulación en cualquier momento, lo que permite, entre otras cosas, interrumpir la práctica en cualquier instante y reanudarla más adelante.

## 3. Archivos .mem

El simulador puede cargar otro tipo de archivos, los *archivos de memoria*, con extensión `.mem`, que permiten cargar en una zona de memoria una serie de números (que pueden ser código máquina de instrucciones o datos). A diferencia de los `.sim`, que afectan a todo el computador (registros, memoria, buses, señales, dispositivos de E/S), los archivos `.mem` sólo modifican las posiciones de memoria que se le indiquen.

Los archivos de memoria son archivos de texto creados por el usuario con la ayuda de un editor de textos ASCII. Su contenido debe ser una lista de números de 16 bits en notación hexadecimal, separados por espacios en blanco, tabulaciones o saltos de línea, que representan las instrucciones o datos que se quieren cargar en memoria.

Por ejemplo, si quisieras escribir un archivo `.mem` que contenga el código correspondiente al siguiente listado:

```
mov r0, r0
mov r1, r2
not  r0
```


dicho fichero `.mem` debería tener el siguiente aspecto:

```
0800
0940
8000
```

Vamos a crear un archivo `.mem` que contenga el código máquina de las siguientes instrucciones y vamos a hacer que se cargue a partir de la dirección `0300h`:


```
sub r0, r5, r3
mov r5, r3
inc r6
xor r5, r5, r5
not r4
mov r1, r2
```

Sigue estos pasos:



- En un editor de texto, escribe la codificación de las instrucciones del listado anterior, separando cada valor hexadecimal de 16 bits por un salto de línea.
- Guarda el archivo en tu carpeta de trabajo con el nombre `4-1mem.mem` y sal del editor.

Vamos a comprobar qué ocurre cuando cargas este archivo.



- Reinicia el simulador.
- Pulsa sobre el dibujo que representa la memoria y escoge la opción *Cargar desde archivo* en el menú que aparecerá.
- Escoge el archivo `4-1mem.mem` que antes guardaste en tu carpeta de trabajo.
- Te saldrá ahora un cuadro de diálogo en el que te pregunta la dirección de carga del contenido del archivo, es decir, a partir de qué dirección quieres que se carguen las instrucciones o datos que contiene el archivo. Introduce el valor `0300h` y pulsa aceptar.
- Busca el código del programa en memoria con el desensamblador. Comprueba que se han cargado correctamente las instrucciones que escribiste en el archivo `4-1mem.mem`.
- Modifica ahora el contenido del registro `PC` para que se ejecute la segunda instrucción de las que has cargado.
- Pulsa `F8` y comprueba que se ejecuta la instrucción deseada.

## 4. Ejercicios adicionales

✓ Reinicia el simulador. Escribe en el registro `r3` el valor `103Bh` y en el registro `r5` el valor `F1ACh`. Coloca la codificación de la instrucción `add r0, r5, r3` en la posición de memoria `C000h` y haz que sea la siguiente instrucción a ejecutar. Antes de ejecutarla, responde a estas preguntas:

- ¿Qué valor tendrá el registro `r0` tras la ejecución? Responde en el [cuestionario](#): pregunta 4.
- ¿Qué valor tendrán los flags del registro de estado? Responde en el [cuestionario](#): pregunta 5.

Ejecuta ahora la instrucción y comprueba si tus previsiones fueron acertadas.

✓ Piensa ahora un valor que podrías escribir en el registro `r3` para que al hacer la operación anterior haya *overflow* y *carry*. Responde en el [cuestionario](#): pregunta 6. Escríbelo en el registro `r3` y modifica el registro `PC` para que se vuelva a ejecutar la instrucción anterior. Ejecútala y comprueba si has elegido bien el valor.

✓ ¿Qué valor tendrías que escribir en el registro `r3` para que el flag de cero estuviese a 1? Responde en el [cuestionario](#): pregunta 7. Introduce ese valor y comprueba si lo has elegido bien.