

01/10/2016

Práctica 0

Fundamentos de Arquitectura de Computadores

Práctica 0

El entorno de simulación j8085sim

Evaluación de la práctica

Es **obligatorio** rellenar y entregar, por correo electrónico, el cuadernillo asociado a este guion de prácticas.

Introducción

En esta práctica vamos a preparar nuestro equipo para trabajar con el programa j8085sim, que es un simulador del procesador 8085. Este software está escrito en Java, y se puede utilizar en entornos Windows, Linux y Mac OS sin problemas.

Paso 1: obtención del software necesario



Figura 1

página (marcado “1”). Si queremos descargarnos el JRE para un sistema operativo distinto del utilizado en el ordenador que estamos usando, tendremos que utilizar el enlace que nos permite consultar los archivos de descarga del JRE para otros sistemas operativos (marcado “2”).

Si hacemos clic en este segundo enlace, se nos muestra una lista de archivos, de tal forma que nos podremos descargar el JRE para Linux o para Mac OS sin problemas.

La instalación del JRE para Windows es sencilla: basta con hacer doble clic con el botón izquierdo del ratón en el fichero que nos acabamos de descargar, y seguir los pasos del asistente de instalación.

Al ser una aplicación Java, necesitamos tener instalado al menos el Java Runtime Environment (JRE) si no lo tuviéramos ya (los equipos con Linux y Mac OS suelen tenerlo instalado por defecto). Lo podemos descargar desde la web <http://www.java.com/es>. Como vemos en la Figura 1, basta con hacer clic en el botón que aparece en el centro de la página, y en la siguiente página (Figura 2), hacer de nuevo clic en el botón situado en el centro de la

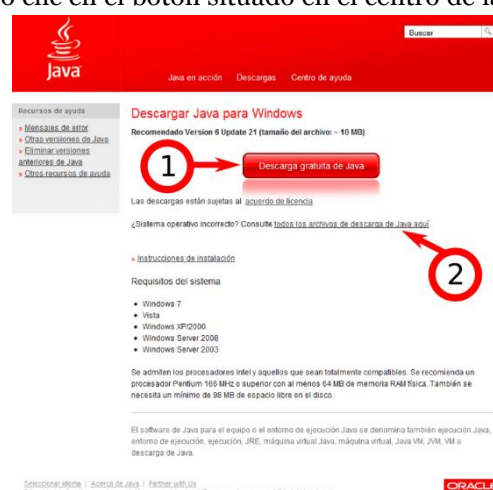


Figura 2

Como ya hemos comentado antes, el sistema operativo Mac OS lleva instalado por defecto un JRE, pero si queremos instalar uno nuevo, también basta con hacer doble clic sobre el fichero que nos hemos descargado para instalarlo.

En sistemas Linux también suele haber un JRE instalado por defecto. También podemos utilizar el gestor de paquetes (Synaptic, YaST, yum, ...) para instalar uno, o podemos descargar el fichero correspondiente de la web de Java y ejecutarlo (normalmente será necesario dar permisos de ejecución al fichero previamente).

Una vez que tenemos instalado el JRE, vamos a proceder a descargar el simulador. Éste está accesible a través de la web de Docencia Virtual (ILIAS) <http://dv.ujaen.es> (Figura 3).

Probablemente nuestro navegador nos proponga ejecutar el programa con el JRE que tenemos instalado, puesto que es un fichero JAR (una aplicación Java). Debemos cambiar la opción y elegir guardar el fichero descargado si queremos volver a utilizarlo en un futuro (Figura 4).

Una vez que tenemos el fichero descargado, lo colocaremos en un lugar que nos resulte apropiado en nuestro disco duro, y podremos ejecutarlo sin problemas.



Figura 3

Para la ejecución en entornos Windows, basta con hacer doble clic sobre el fichero JAR. El programa empezará su ejecución automáticamente.

En entornos Mac OS, también se ejecuta el fichero JAR haciendo doble clic sobre él.

En la mayoría de los entornos Linux, deberemos dar permisos de ejecución al fichero JAR (normalmente basta con hacer clic con el botón derecho del ratón, elegir la opción “propiedades” y activar la posibilidad de ejecutar el fichero), y una vez hecho esto, podremos ejecutarlo haciendo doble clic sobre él. Alternativamente, podemos ejecutarlo desde una terminal, con la instrucción `java -jar 8085sim.jar`. Esta instrucción hay que ejecutarla



Figura 4

desde el directorio en que se encuentra el fichero JAR. Así, si por ejemplo el fichero estuviera almacenado en `/home/usuario/primeropracticas/arquitectura`, habría que abrir una terminal, ejecutar `cd /home/usuario/primeropracticas/arquitectura`, y luego la instrucción `java -jar 8085sim.jar`.

Paso 2: conociendo el simulador

El j8085Sim es un entorno que incorpora un editor de textos, un compilador de lenguaje ensamblador y un simulador de una computadora equipada con un procesador Intel 8085. En este apartado vamos a hacer un recorrido rápido por sus componentes.



Figura 5

la computadora simulada (veremos más adelante su funcionamiento).

Una vez que lanzamos la ejecución del simulador, veremos una primera ventana de bienvenida, mientras se cargan e inician los componentes de la aplicación (Figura 5).

Después de unos instantes de carga, se abre la pantalla principal del simulador (Figura 6). En ella se distinguen los siguientes componentes (según la numeración de la figura):

1.- La barra de menús, con las opciones de:

Archivo (File), que nos permite crear un fichero en ensamblador, abrir un fichero en ensamblador ya existente, guardarlo, guardarlo con otro nombre y salir de la aplicación.

Editar (Edit), para deshacer o rehacer cambios, cortar, copiar y pegar texto, y seleccionar todo el texto del editor.

Administrar (Manage), que permite cambiar parámetros de la computadora simulada (veremos más adelante su funcionamiento).

Ejecutar (Run), con la que podemos compilar un código en ensamblador, ejecutarlo paso a paso o todo de golpe, y parar la ejecución de un programa que esté bloqueado.

Ayuda (Help), que nos da acceso a un pequeño manual de la aplicación, la lista de instrucciones del ensamblador que soporta el procesador Intel 8085, y una ventana de créditos, con información del autor.

2.- La barra de herramientas, con botones organizados en tres grupos: el primero de ellos incluye botones para crear un archivo nuevo, abrir un archivo ya existente y guardar el trabajo realizado hasta ahora; el segundo contiene botones para las tareas típicas de edición (deshacer, rehacer, cortar, copiar y pegar texto); por último, el tercer bloque de controles es para compilar un código en ensamblador, definir a partir de qué posición de memoria se inicia la ejecución, lanzar la ejecución completa o paso a paso de un programa, y detener la ejecución de un programa bloqueado.

3.- El editor de texto, donde escribiremos nuestros programas en lenguaje ensamblador

4.- Los registros de la CPU, donde podremos ver el contenido de cada uno de los registros de la CPU simulada. **Estos valores se muestran en formato hexadecimal.**

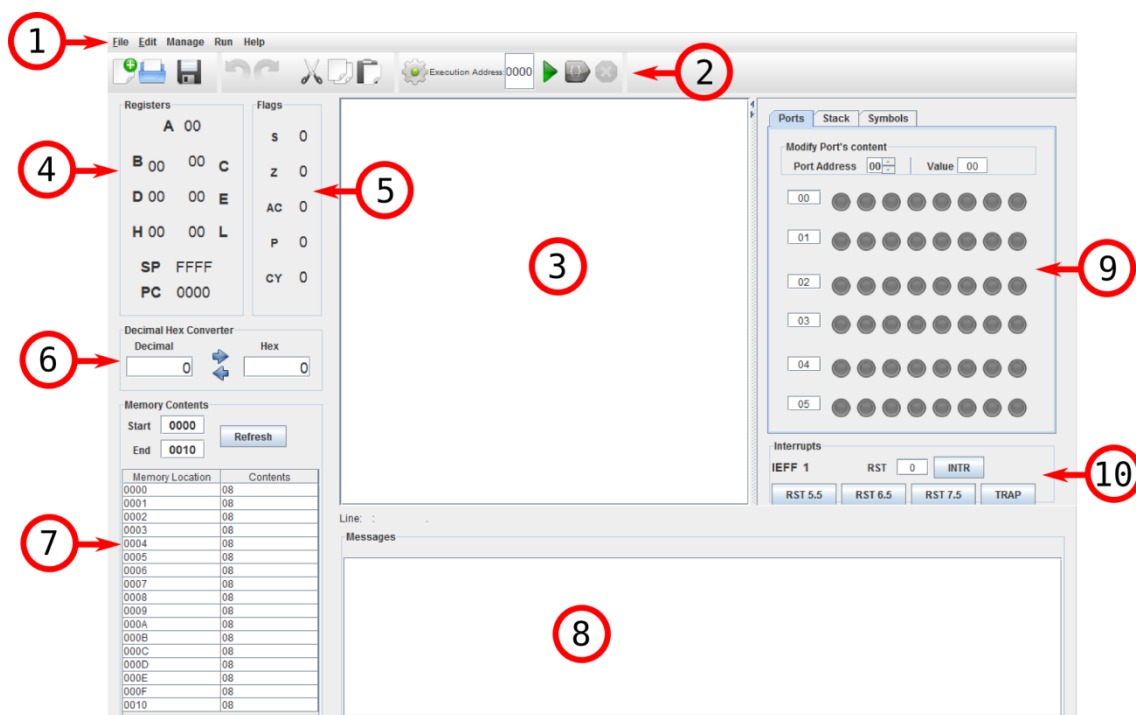


Figura 6

5.- Los flags de estado de la CPU simulada

6.- Una utilidad para convertir entre hexadecimal y decimal, para facilitar la comprensión de los valores mostrados en los registros, la memoria y los puertos. Pulsando los botones en forma de flecha, podemos convertir un valor de decimal a hexadecimal o viceversa.

7.- El estado de la memoria RAM. A través de esta ventana podemos ver qué datos hay guardados en cada posición de memoria de la computadora simulada. Por defecto se muestran las 17 primeras posiciones, pero basta con cambiar los valores de inicio y fin y pulsar el botón de actualizar (Refresh) para que se muestren las posiciones solicitadas. **Tanto los valores almacenados como las posiciones de memoria se muestran en formato hexadecimal.**

8.- La ventana de mensajes, con los errores observados durante la compilación y la ejecución del código, así como mensajes de estado de la computadora simulada.

9.- Puertos de entrada/salida. Utilizando esta ventana podemos introducir información en la computadora simulada, o mostrar información al exterior a través de instrucciones en ensamblador. Cada puerto de entrada/salida almacena un valor de 8 bits, y lleva incorporados una serie de luces LED que se iluminan mostrando el valor almacenado. Cada LED se corresponde con un bit; así, si el valor almacenado es 01001000 (48h), estarán encendidos el segundo y el quinto LED (Figura 7). Podemos utilizar los cuadros de texto de la parte superior de la ventana para asignar un valor a un puerto, o para consultar el valor almacenado. **Los valores almacenados en los puertos se muestran en formato hexadecimal; además, hasta que no se pulsa la tecla “Enter” tras introducir un valor en el cuadro “Value”, no se actualiza el puerto.**

10.- Botones para lanzar interrupciones. Con estos botones podemos activar una interrupción en la computadora simulada, para poder probar rutinas de tratamiento de interrupciones.

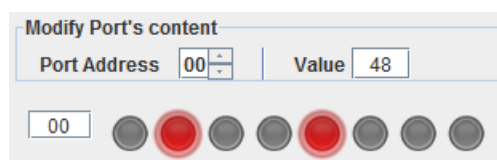


Figura 7

Paso 3: escribiendo y probando nuestro primer programa en ensamblador

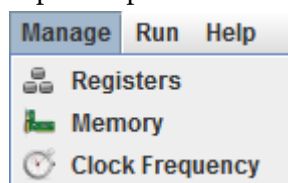
Ahora vamos a escribir nuestro primer programa en ensamblador, lo compilaremos y lo ejecutaremos. Al mismo tiempo, iremos viendo cómo usar el entorno de simulación.

Programa de ejemplo: toma dos valores de entrada a través de los puertos 00h y 01h, suma sus valores y muestra el resultado en el puerto 02h. No se tienen en cuenta situaciones de desbordamiento (overflow).

(En ensamblador, todo lo que va detrás de un punto y coma, y hasta el final de la línea, es comentario y se ignora)

```
in 00      ; almacena el contenido del puerto 00h en el acumulador
mov b, a   ; mueve el valor almacenado en el acumulador al registro B
in 01      ; almacena el contenido del puerto 01h en el acumulador
add b      ; suma al acumulador el valor almacenado en el registro B
out 02     ; actualiza el puerto 02h con el valor almacenado en el acumulador
hlt        ; fin del programa
```

El primer paso es escribir el código en ensamblador en la ventana del editor de texto del simulador.



Una vez hecho esto, deberemos inicializar los registros, la memoria y los puertos de entrada/salida para probar el programa, en caso de que los valores que tuvieran no nos interesasen. Esto lo hacemos utilizando el menú de **Administrar** (Manage) de la barra de menús (Figura 8). En este menú disponemos de tres opciones:

Figura 8

1.- Gestión de los registros. Al elegir esta opción, se abre una ventana (Figura 9) desde la que podemos cambiar los valores de los registros del procesador, así como de los flags del registro de estado. **Los valores de los registros son cantidades en hexadecimal.** Para cambiar un valor, basta con escribir el valor nuevo en la posición correspondiente.

2.- Gestión de la memoria. Esta opción activa una ventana (Figura 10) a través de la cual podemos establecer un valor para todas las posiciones de memoria en un rango que indiquemos. Para esto, indicaremos una posición de inicio, una de fin y un valor, y pulsaremos el botón de Inundación (Flood). **Tanto las direcciones de memoria como los datos almacenados son cantidades en hexadecimal.** A través de esta ventana también podemos consultar cualquier rango de posiciones de memoria, de la misma manera que hacemos con la ventana principal de la aplicación.

3.- Gestión de la frecuencia de reloj del procesador simulado. Cuando seleccionamos esta opción, se nos abre una ventana (Figura 11) en la que podemos indicar la frecuencia de reloj del procesador simulado. Cuanto mayor sea la frecuencia, más rápida será la ejecución de las simulaciones.

Figura 9




Figura 10

Figura 11

También podemos cambiar los puertos de entrada/salida a nuestro gusto, para que la ejecución tenga sentido. En el caso del ejemplo, introduciremos dos valores hexadecimales en los puertos 00h y 01h utilizando los controles situados encima de los puertos. La ventana del simulador, antes de empezar la ejecución, tiene que ser algo parecido a lo mostrado en la Figura 12.

Figura 12

Para poder probar un código, es imprescindible que no haya errores en el mismo. Si hubiera algún error, nos lo mostraría en la ventana de mensajes de la parte inferior, sin embargo, vemos en la Figura 12 que no hay errores, y por tanto, podemos continuar.

Una vez preparada la simulación, podemos ponerla en marcha paso a paso o de una sola vez con los botones de la barra de herramientas. Si pulsamos el botón , se ejecutará todo el programa, con mayor o menor velocidad según la frecuencia de reloj que hayamos fijado antes; sin embargo, si pulsamos el botón , iremos ejecutando el programa instrucción por instrucción (cada vez que pulsemos el botón, se ejecutará una instrucción). En cualquier momento de la ejecución de un programa, se marcará en amarillo la instrucción que se está ejecutando, y estará activo el botón para cancelar la ejecución ()

Al terminar la ejecución de este programa de ejemplo, los registros, flags y puertos de entrada/salida habrán cambiado, tal y como se ve en la Figura 13

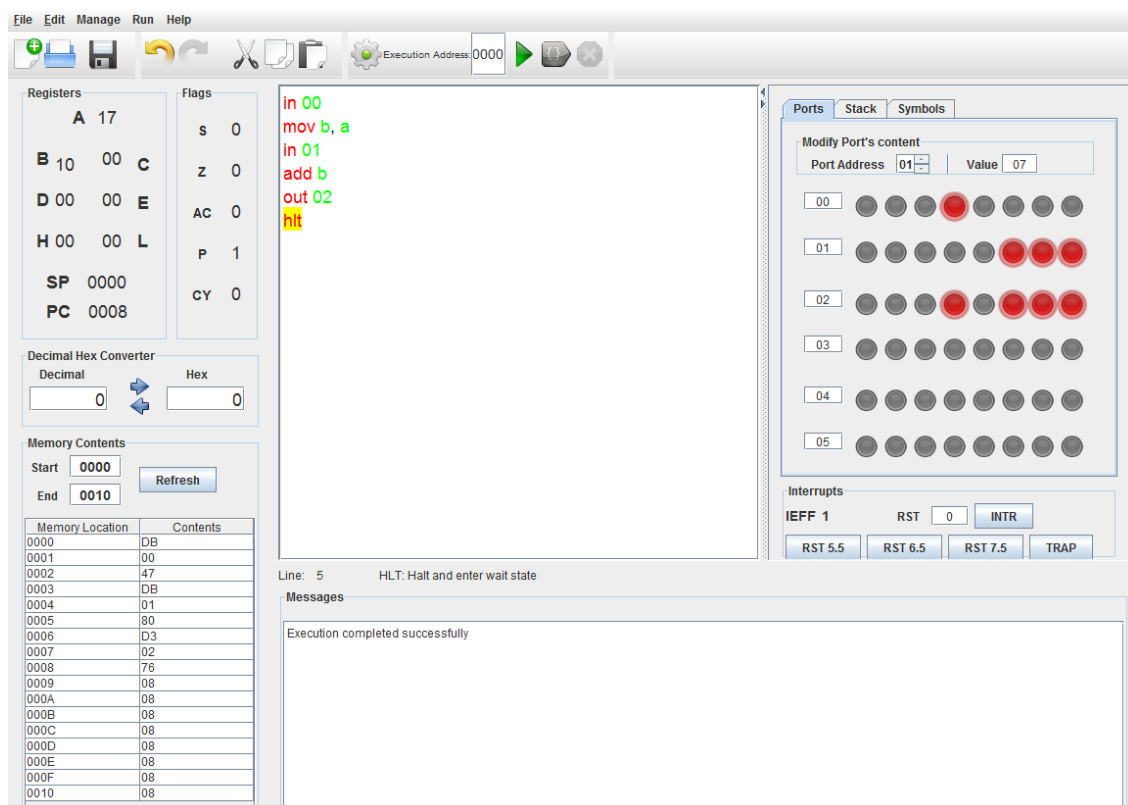


Figura 13