

Interfaz de vídeo del computador teórico

Área de Arquitectura y Tecnología de Computadores – Versión 1.0.403, 10/03/2020

Objetivos

En esta sesión se pretende que el alumno se familiarice con los conceptos de periférico, interfaz, mapa de direcciones y salida mapeada en memoria. Para esto se desarrollarán ejemplos sobre la interfaz de video del computador teórico.

Conocimientos y materiales necesarios

Para poder realizar esta sesión el alumno debe:


- Comprender el funcionamiento básico del computador teórico así como de su ensamblador.
- Repasar la teoría relativa a sistemas de memoria y Entrada/Salida mapeada en memoria.
- Repasar la teoría de la interfaz de vídeo del computador teórico.

Durante la sesión se plantearán una serie de preguntas que deben responderse en el correspondiente [cuestionario](#) del Campus Virtual. El cuestionario se puede abrir en otra pestaña del navegador pinchando en el enlace mientras se mantiene pulsada la tecla [Ctrl](#) .

1. El mapa de memoria y los periféricos

En esta práctica se va a conectar un periférico al computador teórico: una pantalla (de 8 filas por 15 columnas). Para imprimir caracteres en esta pantalla se deben escribir sobre la memoria de video de su interfaz, ^[1] que estará mapeada secuencialmente a partir de una dirección dada que podremos escoger.

¿Pero qué dirección escoger? Evidentemente, tiene que ser una dirección que pueda generar la CPU teórica, es decir, una dirección que esté en su *espacio de direcciones*. Como el bus de direcciones de la CPU teórica tiene 16 líneas,




- ¿Cuántas direcciones distintas podrá generar? Responde en el [cuestionario](#): pregunta 1.
- Si la posición más baja es la `0000h` , ¿cuál será la más alta? Responde en el [cuestionario](#): pregunta 2.

La pantalla necesitará 120 (8x15) de estas direcciones para mapear su memoria de vídeo. Además, tiene un registro de control que también deberá ser mapeado en el espacio de direcciones. Por tanto, la interfaz de video necesitará en total 121 direcciones. Recordarás de teoría que se debe utilizar un número de direcciones que sea potencia de 2. Por tanto, la interfaz de la pantalla necesitará 128 direcciones del espacio direccionable de la CPU.

En todas las prácticas anteriores hemos supuesto que podíamos escribir en cualquiera de las direcciones de memoria un dato o una instrucción. Esto era así porque todo el espacio de direcciones del computador teórico estaba lleno de módulos de memoria. Pero si tenemos que reservar algunas direcciones para la pantalla, tendremos que eliminar alguno de estos módulos. Si no, cuando la CPU generase un dato para una dirección que tuviesen asignada a la vez la interfaz de pantalla y un módulo de memoria, ¿quién cogería el dato?


El simulador de la CPU teórica permite *conectar* y *desconectar* módulos de memoria. Vamos a ver cómo está en estos momentos el sistema de memoria:



- Abre el simulador del computador teórico.
- Pulsa con el botón izquierdo del ratón sobre el dibujo etiquetado con **MEMORIA** en la esquina inferior izquierda del simulador.
- En el menú que te saldrá, escoge la opción **Configurar** .

Aparecerá una pantalla que te muestra, en la parte izquierda, el espacio de direcciones del computador teórico. En ella puedes comprobar si tu respuesta a las preguntas anteriores relativas al tamaño del espacio de direcciones fue acertada. Fíjate en que el computador tiene conectados dos módulos de memoria de 32K palabras, que cubren el espacio de direcciones de 64Ks sin dejar ningún hueco.


Vamos a desconectar un módulo de memoria para hacer sitio a la interfaz de pantalla:



- Pulsa sobre el dispositivo situado más abajo en la figura (el que cubre el rango de direcciones `8000h` hasta `FFFFh`). Ese módulo desaparecerá (será desconectado).

Ahora nuestro computador sólo tiene 32Ks de memoria instalados. Si intentaras escribir en una posición por encima de los primeros 32Ks (por ejemplo, la `8000h`), el simulador te daría un error porque ¡en esa posición no hay memoria!

Hemos dejado libres 32Ks para mapear la interfaz de pantalla, pero sólo necesitamos 128 direcciones, así que para tener más memoria vamos a conectar un módulo de 16Ks a continuación del de 32Ks:



- Haz clic en el módulo de 16Ks dibujado en la parte derecha de la pantalla. De esta forma pasará a estar seleccionado.
- Haz clic ahora en el hueco que hay entre las direcciones `8000h` y `9000h` . Se situará un chip de 16Ks a partir de la dirección `8000h` .

Con esto dejamos un hueco de 16Ks, suficiente para la interfaz de pantalla. Vamos a conectarla:

- Cierra la ventana de **Configuración de memoria** pulsando el botón [Aceptar](#) .
- Haz clic en el dibujo etiquetado con **ENTRADA/SALIDA** en la esquina inferior derecha de la ventana principal del simulador.
- Escoge la opción **Conectar Pantalla** .
- Te aparecerá una ventana en la que tendrás que introducir las características de la pantalla que quieres conectar. En el lugar reservado para el nombre escribe **Pantalla1** . Como dirección base (es decir, dirección a partir de la cual se mapearán las 128 direcciones de la interfaz de la pantalla) introduce la `F000h` .
- Pulsa el botón [Aceptar](#) . Deberás ver una ventana que representa la pantalla que acabas de conectar.

Con esto ya habrás equipado al computador elemental con una pantalla cuya interfaz estará mapeada a partir de la dirección `F000h` . Para no tener que repetir todo este proceso en sucesivas ocasiones, guarda la configuración actual del simulador en un archivo llamado `5-1pantalla.sim` .

2. Análisis de la salida mapeada en memoria

Para ver cómo se escribe en la pantalla vamos a hacer un sencillo programa que escriba un asterisco en la esquina superior izquierda de la pantalla del computador teórico.

La interfaz de la pantalla está mapeada de tal forma que a la dirección base (`F000h` en nuestro caso) se le asigna la posición de memoria de vídeo que representa la primera celda de la primera fila, a la siguiente dirección se le asigna la posición de memoria de vídeo que representa la segunda celda de la primera fila, etc. Después de todas las posiciones de memoria de vídeo está mapeado el registro de control.

Cada posición de la memoria de vídeo tiene la siguiente estructura:

Para comenzar a cargar en el registro **r0** la dirección donde vamos a guardar los datos, escriba el valor de la dirección de memoria que desea guardar los datos cuando *instale* la pantalla elegiste como dirección de memoria. En el registro **r0** ? Responde en el [questionario](#): pregunta 4. Escriba el valor de la dirección de memoria que desea guardar los datos para cargar ese valor en **r0**.

Después de haber cargado el valor en el registro **r1** el valor que queremos que aparezca en la pantalla. Escriba las instrucciones necesarias para que la parte baja de la pantalla se escriba en color negro y la parte alta los atributos para que se escriba en color blanco. Escriba la instrucción necesaria para que el asterisco aparezca en la pantalla de los registros anteriores.


Para representar los colores se usa el siguiente código RGB (Red, Green, Blue):

Código	Color
000	Negro
001	Azul
010	Verde
011	Cian
100	Rojo
101	Magenta
110	Amarillo
111	Blanco

Para escribir un asterisco en una posición de la pantalla habrá que poner el código ASCII del carácter asterisco en la parte baja de la dirección de la memoria de vídeo asociada a esa posición. Para obtener el código ASCII de un carácter en el ensamblador se debe escribir el carácter entre comillas simples. Por ejemplo: `'*'` .

En la parte alta de la dirección de la memoria de vídeo hay que fijar los atributos del carácter. Vamos a empezar siendo discretos: el asterisco deberá ser blanco sobre fondo negro. ¿Cuánto deberá valer el byte de atributo? Responde en el [cuestionario](#): pregunta 3.

Vamos a hacer el programa que imprime un asterisco en la esquina superior izquierda:



- Edita un archivo llamado `5-1ast1.ens` .
- El programa debe cargarse a partir de la dirección `0500h` .
- El código debe empezar cargando en el registro `r0` la dirección donde vamos a escribir el asterisco. Teniendo en cuenta que cuando *instale* la pantalla elegiste como dirección base `F000h` , ¿cuánto tiene que valer el registro `r0` ? Responde en el [cuestionario](#): pregunta 4. Escribe las instrucciones necesarias para cargar ese valor en `r0` .
- A continuación vamos a cargar en `r1` el valor que queremos que aparezca en la memoria de la interfaz de vídeo. Escribe las instrucciones necesarias para que la parte baja de `r1` contenga el código ASCII del asterisco y la parte alta los atributos para que se escriba en color blanco sobre fondo negro.
- Por último, escribe la instrucción necesaria para que el asterisco aparezca en pantalla. Para ello debes usar los dos registros anteriores.
- Finaliza el programa, guárdalo, sal del editor, ensambla el programa y cuando hayas obtenido el `.eje` , cárgalo en el simulador.


Sin perder la pantalla de vista, ejecuta el programa instrucción a instrucción comprobando que el asterisco aparece cuando debe hacerlo. Nota: Para que [F8](#) funcione, la ventana activa debe ser la del simulador y no la pantalla.

Vamos a ponerle un poco más de color al programa:



- Edita de nuevo el archivo `5-1ast1.ens` . Guárdalo con el nombre `5-1ast2.ens` .
- Modifica el programa para que el asterisco se escriba en rojo sobre verde. ¿Cuánto debe valer ahora el atributo? Responde en el [cuestionario](#): pregunta 5.
- Haz que el asterisco se escriba en la segunda celda de la segunda fila. ¿Cuál es la dirección de memoria asociada con esa celda? Responde en el [cuestionario](#): pregunta 6. (Recuerda que las dimensiones de la pantalla son 8 filas por 15 columnas.)
- Guarda, ensambla y ejecuta el programa para comprobar que funciona correctamente.

Vamos a ver ahora cómo funciona el programa a nivel de señales de control:



- Carga el archivo `5-1pantalla.sim` .
- Carga el archivo `5-1ast2.eje` .
- Ejecuta las cuatro primeras instrucciones con [F8](#) . La quinta instrucción del programa debería ser la que escribe el asterisco en pantalla, así que la vamos a ejecutar paso a paso.
- Ejecuta los tres primeros pasos con [F7](#) y comprueba que estás en la instrucción deseada.
- Antes de ejecutar el siguiente paso, contesta a estas preguntas: ¿Qué señales se activarán? Responde en el [cuestionario](#): pregunta 7. Pulsa [F7](#) y comprueba que has respondido adecuadamente.
- ¿Qué señales se activarán en el siguiente paso (T5)? Responde en el [cuestionario](#): pregunta 8. ¿Aparecerá en este paso el asterisco en pantalla? Responde en el [cuestionario](#): pregunta 9. ¿Por qué? Pulsa [F7](#) y comprueba si has respondido correctamente. Fíjate en la señal **WRITE** abajo a la derecha: como la dirección que hay en **MAR** pertenece al espacio de direcciones de E/S, la señal va dirigida al módulo de E/S y no al de memoria.
- ¿Qué señales se activarán en el siguiente paso (T6)? Responde en el [cuestionario](#): pregunta 10. ¿Aparecerá ahora el asterisco? Responde en el [cuestionario](#): pregunta 11. Pulsa [F7](#) y compruébalo.

3. Un programa más complejo: imprimir cadenas


Vamos a hacer ahora un programa que imprima una cadena de caracteres en la pantalla. El programa simplemente irá recorriendo la cadena e imprimiendo cada uno de sus caracteres en posiciones consecutivas de la pantalla hasta que se encuentre un cero (terminador de la cadena). Utilizaremos los siguientes registros:

Registro	Uso
r0	Apuntará a la dirección de pantalla donde escribir
r1	Apuntará al carácter de la cadena a escribir
r2	Contendrá temporalmente el carácter a escribir
r3	Contendrá un cero para hacer comparaciones

El pseudocódigo del programa será el siguiente:

```
Inicializaciones
bucle: Traer carácter a R2
Si es cero
  Salir
Si no
  Preparar atributos en R2
  Imprimir el carácter
  Avanzar la posición donde imprimir
  Avanzar a la siguiente posición de la cadena
  Ir a bucle
Fin Si
```

Sigue los siguientes pasos:



- Edita un fichero que se llame `5-1cad1.ens` .
- El programa se debe cargar a partir de la dirección `0500h` .
- Define en la sección de datos una cadena que sea `"Hola, mundo"` . Utiliza la etiqueta `cad` para apuntar a ella.
- Define a continuación un dato que valga cero. Como se ha comentado anteriormente indicará el final de la cadena.
- Comienza el código cargando en el registro `r0` la dirección base de la interfaz de la pantalla.
- A continuación, carga en el registro `r1` la dirección de la primera posición de la cadena.
- Después inicializa el registro `r3` a cero.
- Vas a comenzar ahora un bucle, así que márcalo con la etiqueta `bucle` .
- La primera instrucción del bucle debe ser copiar el carácter desde memoria al registro `r2` .

A continuación, para comprobar si ya se ha acabado la cadena o todavía quedan más caracteres, compara el carácter que acabas de traer con cero (que, como recordrás, está en el registro `r3`).

La siguiente instrucción tiene que ser un salto condicional. Si la comparación anterior determinó que hay un cero en el registro `r2` , debes saltar al final de programa, que vas a señalar convenientemente con una etiqueta llamada `final` .

Sigamos con las instrucciones que se ejecutan cuando todavía no hemos encontrado el cero. En primer lugar, debes colocar en la parte alta del registro `r2` un atributo que haga que se vea la cadena. Haz que la cadena aparezca en color verde sobre fondo negro.

Introduce ahora la instrucción para escribir el carácter en la pantalla. Recuerda que tienes el carácter en el registro `r2` y la posición en la que quieres escribir en el registro `r0` .

Escribe la instrucción para que el registro `r0` apunte a la siguiente posición y así cuando escribas la próxima letra se escriba a continuación de la anterior.

Escribe la instrucción para que el registro `r1` apunte al siguiente carácter de la cadena.

Escribe la instrucción para hacer un salto incondicional al principio del bucle.

Completa el programa poniendo la etiqueta `final` (si no lo has hecho ya) y la directiva de **FIN** .

Guarda el archivo, sal del editor, ensambla el programa, cárgalo en el simulador y comprueba que funciona correctamente. Para no tener que pulsar muchas veces [F8](#) , prueba a pulsar [F9](#) (como siempre, tienes que tener la ventana principal del simulador activa para que el simulador reciba estas órdenes). La CPU se pone en estado de ejecución, es decir, cada vez que acaba de ejecutar una instrucción, va a por la siguiente. Es similar a pulsar continuamente [F8](#) . Cuando hayas visto que la cadena se escribe en pantalla, vuelve a pulsar [F9](#) para que la CPU pare de ejecutar instrucciones (de nuevo, la ventana activa debe ser la principal del simulador).


Transforma el código del programa `5-1cad1.ens` en un procedimiento que permita escribir una cadena acabada en cero en cualquier posición de la pantalla. Debe recibir como parámetros la dirección de la cadena y la dirección donde debe comenzar a escribirla. Define dos cadenas en la sección de datos y haz que el programa principal imprima las dos cadenas en dos lugares distintos de la pantalla mediante dos llamadas al procedimiento.

Haz un protector de pantalla para nuestro computador teórico. Debe consistir en un programa que esté continuamente escribiendo caracteres cambiantes en la pantalla. Para ello, haz un bucle que vaya escribiendo un carácter e incrementando tanto el carácter a escribir como la posición. Comprueba después de cada incremento que la posición no se salga de la pantalla (no sea superior a la última dirección de la interfaz): cuando ocurra eso, haz que el puntero que apunta a la posición a escribir vuelva a apuntar a la primera celda de la pantalla. Ejecuta el programa con [F9](#) para comprobar que funciona.

Carga el archivo `5-1pantalla.sim` . Añade otra pantalla al computador teórico con dirección base `D700h` . Modifica el protector de pantalla para que vaya escribiendo en las dos pantallas a la vez.

4. Ejercicios adicionales

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



- Carga el archivo `5-1pantalla.sim` y vete a la opción de configurar memoria. Rellena con dispositivos de memoria todo el espacio posible hasta llegar a la dirección de la pantalla pero sin sobreescribirla. ¿Cuántos dispositivos has usado? Responde en el [cuestionario](#): pregunta 12. ¿De qué tamaño? Responde en el [cuestionario](#): pregunta 13. Carga el archivo `5-1cad1.eje` y comprueba que sigue funcionando con la nueva configuración.
- Escribe un programa que borre toda la pantalla. Para ello deberás escribir en el bit de peso 0 del registro de control un 1 en un bit del registro de control sin modificar el resto de bits debes leer primero el contenido del registro y luego, utilizando una máscara y una operación lógica, cambiarlo. Cuando hayas escrito y ensamblado el programa, carga el archivo `5-1pantalla.sim` y carga y ejecuta el archivo `5-1cad1.eje` para escribir algo en pantalla. Carga después el programa para borrar la pantalla y ejecútalo para comprobar que funciona correctamente.
- Transforma el código del programa `5-1cad1.ens` en un procedimiento que permita escribir una cadena acabada en cero en cualquier posición de la pantalla. Debe recibir como parámetros la dirección de la cadena y la dirección donde debe comenzar a escribirla. Define dos cadenas en la sección de datos y haz que el programa principal imprima las dos cadenas en dos lugares distintos de la pantalla mediante dos llamadas al procedimiento.
- Haz un protector de pantalla para nuestro computador teórico. Debe consistir en un programa que esté continuamente escribiendo caracteres cambiantes en la pantalla. Para ello, haz un bucle que vaya escribiendo un carácter e incrementando tanto el carácter a escribir como la posición. Comprueba después de cada incremento que la posición no se salga de la pantalla (no sea superior a la última dirección de la interfaz): cuando ocurra eso, haz que el puntero que apunta a la posición a escribir vuelva a apuntar a la primera celda de la pantalla. Ejecuta el programa con [F9](#) para comprobar que funciona.
- Carga el archivo `5-1pantalla.sim` . Añade otra pantalla al computador teórico con dirección base `D700h` . Modifica el protector de pantalla para que vaya escribiendo en las dos pantallas a la vez.

1. Se utilizan indistintamente las denominaciones de interfaz de vídeo e interfaz de pantalla.