Prácticas 10 y 11 de *Estructura de los Computadores*

Eduardo Espuch

Universitat d'Alacant
Universidad de Alicante

Resumen

Explicación para los ejercicios entregables de las prácticas 10 y 11 de *Estructura de los Computadores*, los archivos adjuntos vendrán en una carpeta aparte con un total de 4 archivos (caps.asm, read_string.asm, interrupciones.asm y rutinageneral.asm).

1. Ejercicios entregables de la práctica 10

1.1. caps.asm

Se pide iterar la lectura y escritura de caracteres por teclado y display, suponiendo que todos los caracteres introducidos son letras minúsculas ('a'=97 y 'z'=122), y que se impriman los mismos caracteres en mayúsculas ('A'=65 y 'Z'=90), considerando que el bucle finaliza cuando se introduce el carácter '/'. Sera tan sencillo como restar 32 al registro que contiene el carácter para obtener su carácter en mayúscula correspondiente. Solo introducimos letras minúsculas y '/' o como condición de salida, otro tipo de carácter no esta considerado y podría causar problemas.

1.2. read_string.asm

Partimos de que tenemos el código principal que hace llamamiento a las funciones *read_string* y *print_string*, y esta última ya esta hecha. Debemos de hacer que la función *read_string* almacene todos los valores que le pasemos por el teclado emulado a una cadena que se genera en memoria de unos 32 bytes. Para ello tendremos en cuenta los siguientes puntos:

1. Realizar un bucle con condición de salida tal que el carácter introducido sea un salto de linea ('\n'), esto permitirá introducir una cantidad de caracteres (que ocupan un byte) hasta que realicemos el salto de linea.

   Aunque hayamos dicho que la cadena es de 32 bytes, no pasara nada si acabamos introduciendo mas ya que, en este caso, el resto de la memoria esta a 0 y por tanto no accedemos a posiciones ajenas, se podría solucionar si forzamos que finalice al leer 32 caracteres, ademas la función *print_string* ira accediendo a la memoria hasta que lea un byte con valor '\0'.

2. Un bucle que debe de emular que el sistema esta ocupado realizando instrucciones(en este caso se ha dejado únicamente la instrucción andi para comprobar el flag) y el cual comprobara periódicamente si el flag del teclado esta activo para guardar el carácter introducido en el string 'cadena'.

3. Acceder en cada iteración de guardado de carácter al siguiente byte, debemos de trabajar con funciones save byte para almacenarlo en memoria o sino consumiría mucho espacio.

4. Entender que en $a0 se guarda la dirección de 'cadena' y que etiquetas como **ControlTeclado**, **BufferTeclado**,... guardan valores por los que se sustituirán, es decir, si hiciéramos 5+**BufferTeclado** se podría interpretar como 5+4. También saber que *print_string* imprimirá todo lo que encuentre desde el inicio de cadena hasta el primer '\0' que encuentre, en este caso hemos forzado que se guarde uno al leer un '\n'.

5. El programa si se ejecuta en velocidades de ejecuciones lentas, es posible que si se introducen caracteres no se le de tiempo a almacenarlos en memoria, a mayor velocidades de ejecución menor preocupación por la perdida de caracteres, si se ejecuta a la máxima velocidad no debemos de preocuparnos.

2. Ejercicios entregables de la práctica 11

2.1. interrupciones.asm

Se pide obtener una rutina de tratamiento de interrupciones que imprima en el display del transmisor el carácter introducido en el receptor, guardado este en el registro $v0, añadiendo ademas un programa principal el cual permita hacer pruebas y que finalice cuando introduzcamos un salto de linea.

1. Lo primero y mas sencillo, sera escribir el programa principal. En este debemos incluir al inicio un código para que acepte las peticiones de interrupciones, habilitando las interrupciones por teclado. El resto del código sera un bucle infinito que mantenga al sistema ocupado, en este caso, fuerza a comprobar si un numero es potencia de 2 de forma secuencial y cuando alcanza el valor máximo para 32 bits, el bucle reinicia los registros a los valores iniciales.

2. Escribimos la rutina, orientada únicamente a interrupciones por Hardware, con lo cual nos interesa que el código de excepción en el registro Cause tenga un único valor, el resto de valores los ignoramos y saltamos a finalizar la rutina.

3. Cuando el código de excepción vale 0, se da que la interrupción es causada por el hardware y, por lo tanto, hay un carácter que leer y el cual guardamos en el registro $v0. Si el carácter leído corresponde a un salto de linea, procedemos a finalizar el programa.

4. Si fuese cualquier otro tipo de carácter, comprobamos si el display esta listo para imprimir datos y, si lo esta, imprimimos el valor. A continuación saltamos a la finalización de la rutina.

5. Para esta rutina, en la cual solo consideramos interrupciones hardware, basta con reiniciar el registro Cause, y rehabilitar las interrupciones en el registro Status, EPC como es una interrupción lo mantenemos con el valor que tenia y restablecemos los valores de los registros usados en la rutina.

2.2. rutinageneral.asm

Se pide una rutina que funciones con tres tipos de excepciones, desbordamiento aritmético (Código de excepción, 12), error por lectura al tratar de acceder a una dirección invalida (código de excepción, 4) e interrupciones de teclado (código de excepción, 0). En los tres casos se debe de imprimir los mensajes de interrupción correspondiente por consola del MARS. Ademas un código de prueba para probar los tres casos.

1. Lo mas sencillo sera el código de prueba, si hacemos un bucle dejaremos al usuario que tenga tiempo para escribir por teclado. Si en el bucle, cuando el valor del cual depende alcanza un

valor, incluir un código el cual sepamos que va a causar algún tipo de error, ya sea por lectura del contenido de una dirección errónea o una operación en la que se produce desbordamiento. Ademas, en este código habrá que incluir al inicio un código el cual permita al programar aceptar interrupciones por teclado.

2. Reservar en la memoria del núcleo (.kdata) bytes para los registros necesarios ademas de almacenar los mensajes necesarios para interactuar con el usuario e indicar la interrupción producida.

3. Extraemos el código de excepción del registro Cause y según el valor que tenga saltamos a una de las posibles funciones consideradas, todas ellas imprimirán un mensaje para indicar que causa la excepción, pero la de interrupción por teclado ademas el carácter leído lo mostrara por display y, si el carácter es un salto de linea, finalizara el programa. En todas las funciones salvo la de la interrupción incrementamos por 4 la dirección en EPC.

4. Restaurar los registros Cause, VAddr y Status y devolver el valor que se había guardado previamente a los registros usados.