Práctica de Estructura de Computadores
Sistemas de Entrada/Salida: Entrada/Salida Programada

Curso 2010/2011

Antonio Pérez Ambite
Santiago Rodríguez de la Fuente

Departamento de Arquitectura y Tecnología de Sistemas Informáticos
Facultad de Informática
Universidad Politécnica de Madrid

Octubre 2010

1. Introducción y objetivos

La práctica consistirá en la programación, en ensamblador del Motorola 68000, de un conjunto de subrutinas que permitan la realización de operaciones de Entrada/Salida (E/S) sobre dos líneas serie utilizando el módulo DUART MC68681. La técnica de E/S que se utilizará para esta práctica el la **E/S programada** o por *polling*.

El objetivo de esta parte es que el alumno se familiarice con el entorno del simulador, el lenguaje ensamblador y la interfaz de los dispositivos que se desea utilizar.

Esta parte de la práctica, cuya estructura se muestra en la figura 1, presentará una interfaz constituida por las siguientes subrutinas:

INIT: Inicialización del dispositivo. Preparará la línea serie A para recibir y transmitir caracteres. Esta subrutina se le proporcionará construida al alumno.

SCAN: Lectura de un dispositivo. Las operaciones sobre la línea A se realizarán mediante muestreo (*polling*), es decir, mediante E/S programada.

PRINT: Escritura en un dispositivo. Las operaciones sobre la línea A se realizarán mediante muestreo (*polling*), es decir, mediante E/S programada.

Se aconseja la realización de este ejercicio en tres fases:

- Inicialización de los dispositivos.
- Lectura y escritura en un dispositivo mediante muestreo (*polling*), es decir, mediante E/S programada.
- Realización de programas principales que comprueben el funcionamiento correcto de las subrutinas. Por ejemplo, un programa que invocando a los módulos anteriores, realice una lectura de un conjunto de caracteres y posteriormente los escriba.

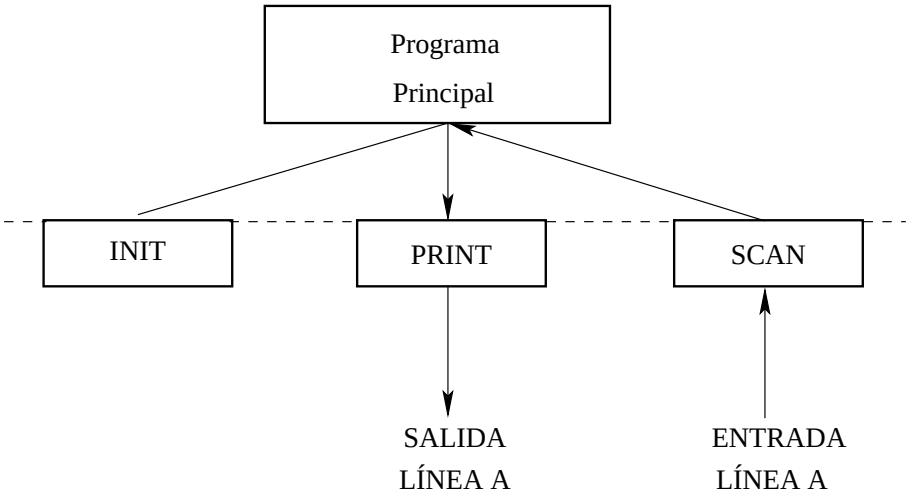

Figura 1: Estructura de la primera parte de la práctica.

2. Descripción de las Subrutinas

Todas las subrutinas reciben los parámetros en la pila y el valor de retorno, si lo tiene, se devuelve en el registro `D0`.

`INIT ( )`

Parámetros:
No tiene.

Resultado:
La línea A debe quedar preparada para la **recepción** y **transmisión** de caracteres. Al finalizar la ejecución de la instrucción `RTS`, el puntero de pila (*SP*) debe apuntar a la misma dirección a la que apuntaba antes de ejecutar la instrucción `BSR`. Debido a la particular configuración del emulador, esta subrutina no puede devolver ningún error y, por tanto, no se devuelve ningún valor de retorno en el registro `D0`.

Descripción:
La rutina `INIT` realiza la inicialización de la línea A del MC68681. Como resultado, la línea A debe quedar preparada para la recepción y transmisión de caracteres. Los parámetros de inicialización de esta subrutina son los siguientes:

- 8 Bits por carácter.
- No activar el *eco*.
- La velocidad de recepción y transmisión será de 38400 bits/s.
- Funcionamiento *Full Duplex*: deben estar habilitadas la recepción y la transmisión simultáneamente.

Se supondrá que el programa que invoca a esta subrutina no deja ningún valor representativo en los registros del computador salvo el puntero de marco de pila (A6).

`SCAN (Buffer, Tamaño)`

Parámetros:

- **Buffer:** 4 bytes. Es el buffer en el que se van a devolver los caracteres que se han leído del dispositivo. Se pasa por dirección. Es un parámetro de salida.
- **Tamaño:** 2 bytes. Es un número entero sin signo que indica el número de caracteres que se quieren leer del dispositivo. Es un parámetro de entrada.

Resultado:

- **D0:** 4 bytes. Se devuelve un código que indica el número de caracteres que se han leído.

Descripción:

La rutina `SCAN` realiza la lectura, de la línea A del MC68681, de un bloque de caracteres cuyo tamaño viene definido por el parámetro `Tamaño`. La lectura se deberá realizar de forma **bloqueante**, es decir, la subrutina finalizará cuando se haya completado la lectura del bloque completo.

El programa deberá devolver en la dirección apuntada por `Buffer` el bloque de caracteres leído. Esta subrutina debe dejar el puntero de pila (*SP*) apuntando a la misma posición de memoria a la que apuntaba antes de realizar la llamada a subrutina.

Se supondrá que el programa que invoca a esta subrutina habrá reservado espacio suficiente en el buffer que se pasa como parámetro (`Buffer`) y no deja ningún valor representativo en los registros del computador salvo el puntero de marco de pila (A6).

`PRINT (Buffer, Tamaño)`

Parámetros:

- **Buffer:** 4 bytes. Es el buffer en el que se pasa el conjunto de caracteres que se desea escribir en el dispositivo. Se pasa por dirección. Es un parámetro de entrada.
- **Tamaño:** 2 bytes. Es un número entero sin signo que indica el número de caracteres que se quiere escribir en el puerto A. Es un parámetro de entrada.

Resultado:

- **D0:** 4 bytes. Se devuelve un código que indica el número de caracteres que se han escrito.

Descripción:

La rutina `PRINT` realiza la escritura de un bloque de caracteres, cuyo tamaño viene definido por el parámetro `Tamaño`, por la línea A del MC68681. La escritura se deberá realizar de forma **bloqueante**, es decir, la subrutina no finalizará hasta que haya completado la escritura del bloque completo.

Esta subrutina debe dejar el puntero de pila (*SP*) apuntando a la misma posición de memoria a la que apuntaba antes de realizar la llamada a subrutina.

Se supondrá que el programa que invoca a esta subrutina no deja ningún valor representativo en los registros del computador salvo el puntero de marco de pila (A6).

Nota: como complemento a la descripción de estas subrutinas, en la sección Ejemplos se proporcionan distintos casos de uso.

3. Variables locales y paso de parámetros

El procesador MC68000 no dispone de un registro de propósito específico que realice las tareas de puntero de marco de pila (`FP`). No obstante, habitualmente se suele utilizar el registro de direcciones `A6` para que realice estas funciones. El procesador MC68000 dispone de dos instrucciones que ayudan a la creación y destrucción del marco de pila de una subrutina: `LINK` y `UNLK`. Estas instrucciones permiten gestionar fácilmente la creación y destrucción de las variables locales de una subrutina.

El espacio asignado para variables locales se reserva en el marco de pila de la correspondiente rutina. Para construir dicho marco de pila basta con salvaguardar el valor que tuviera el registro que actúa como puntero al marco de pila (`A6`), crear el nuevo marco de pila y reservar espacio en la pila para las variables locales. Obsérvese que todas estas funciones son realizadas por la instrucción `LINK`.

Supóngase que una rutina `SCAN` del ejercicio 2 necesita utilizar dos variables locales de 32 bits (i y j). La estructura de la pila se muestra en la figura 2. La reserva de este espacio de variables (8 bytes) se realizará al entrar en la rutina:

```
SCAN:     LINK A6, #-8      *Se crea el marco de pila
```

Si en el código posterior de la rutina `SCAN` se desea cargar en el registro **D2** la variable i y en **D3** la variable j se realiza con el siguiente código ensamblador:

```
SCAN:    LINK A6, #-8      *Se crea el marco de pila
         MOVE.L -8(A6),D2
         MOVE.L -4(A6),D3
```

Para deshacer el marco de pila creado a la entrada de la subrutina se deberán realizar las siguientes acciones:

- Se copia el valor del puntero de marco al puntero de pila.
- Se restaura el valor que tuviera el puntero de marco antes de entrar en la subrutina.
- Se retorna a la subrutina llamante.

Las dos primeras acciones son realizadas mediante la instrucción `UNLK`. Por tanto, el código de salida de una subrutina que utilice marco de pila se muestra a continuación.

```
UNLK A6      *Se destruye el marco de pila
RTS
```

Asignación de etiquetas y de memoria

Los puntos de entrada de las subrutinas deberán ir asociados a las etiquetas `INIT`, `SCAN` y `PRINT`.

El rango de direcciones **0 a la 0x00003FF** se reservarán para ubicar la tabla de vectores de interrupción. El alumno debe ubicar todo el código (datos y variables globales privadas a las subrutinas) a partir de la dirección hexadecimal 0x0000400 hasta la 0x00007FFF. La pila se situará en las **posiciones altas de memoria**.

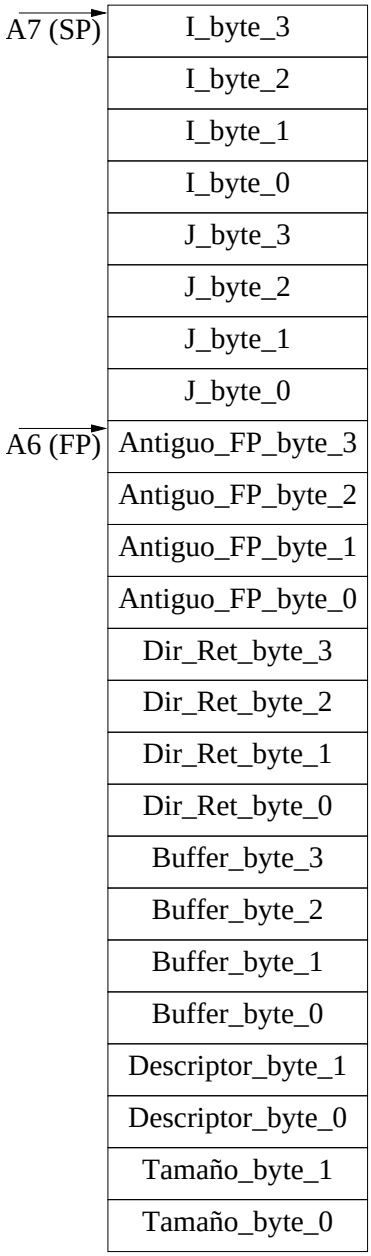

Figura 2: Gestión de variables locales.

Ejemplos

Como aclaración a la especificación de las subrutinas, a continuación se incluye una serie de ejemplos con los argumentos que se pasan a cada una de las subrutinas y direcciones de memoria que se modifican. Este conjunto de casos debe ser utilizado como ejemplo de la especificación a subrutinas, no como los casos de prueba con los que se evaluará la práctica. Puesto que en este procesador el direccionamiento es a nivel de byte, cada una de las direcciones que se muestran en este apartado contendrán un byte.

NOTA: Los números que comienzan con 0x están representados en hexadecimal.

E/S programada: INIT

Caso 1.

A7 = 32000	Direcciones de Memoria: **32000:** ??, ??, ??, ??,

Resultado:

A7 = 32000

Debe dejar la línea A preparada para la recepción y transmisión de caracteres.

E/S programada: SCAN

Caso 2.

A7 = 32000	Direcciones de Memoria: **32000:** 0x00, 0x00, 0x13, 0x88, **32004:** 0x00, 0x0C, **5000:** ??, ??, ??, ??, . . . , ??
Caracteres a la entrada de la línea A:	arquitectura de
Representación ASCII:	0x61, 0x72, 0x71, 0x75, 0x69, 0x74, 0x65, 0x63 0x74, 0x75, 0x72, 0x61, 0x20, 0x65, 0x64

Resultado:

A7 = 32000	Direcciones de Memoria:
D0 = 12	**5000:** 0x61, 0x72, 0x71, 0x75, 0x69, 0x74, 0x65,
	5007: 0x63, 0x74, 0x75, 0x72, 0x61
Caracteres pendientes de ser leídos:	`<blanco>` de
Representación ASCII:	0x20, 0x65, 0x64

Caso 3.

A7 = 32000	Direcciones de Memoria:
	32000: 0x00, 0x00, 0x13, 0x88,
	32004: 0x00, 0x01,
	5000: ??, ??, ??, ??, . . . , ??
Caracteres a la entrada de la línea A:	s
Representación ASCII:	0x73

Resultado:

A7 = 32000	Direcciones de Memoria:
D0 = 1	**5000:** 0x73
Caracteres pendientes de ser leídos:	`<vacío>`

E/S programada: PRINT

Caso 4.

A7 = 32000	Direcciones de Memoria:
	32000: 0x00, 0x00, 0x13, 0x88,
	32004: 0x00, 0x07
	5000: 0x70, 0x6C, 0x61, 0x6E, 0x20, 0x39, 0x36

Resultado:

A7 = 32000
D0 = 7

Caracteres a la salida de la línea A:	0x70, 0x6C, 0x61, 0x6E, 0x20, 0x39, 0x36
Representación ASCII:	plan 96

4. Normas

La práctica se realizará en grupos de dos personas. Estos grupos serán los mismos que realizarán el proyecto de E/S. El grupo tendrá que registrarse en el sistema de entrega de prácticas en `http://www.datsi.fi.upm.es/Practicas` al igual que en prácticas anteriores.

Se realizarán correcciones en todas las sesiones de prácticas cada 15 minutos. De esta forma cada grupo de prácticas puede evaluar el logro de objetivos de la práctica.

Se realizará una última corrección a las 20:00 del día de la última sesión.

En `http://www.datsi.fi.upm.es/docencia/Estructura_09/E_S/` puede encontrar información acerca de esta práctica.

5. Sesiones

Se han programado dos sesiones en las que los alumnos deberán realizar las tareas que se describen a continuación.

5.1. Sesión 1

En esta sesión se realizará una toma de contacto con el simulador `BSVC`. El profesor expondrá cómo se arranca, se configura, se carga un programa, se ponen puntos de ruptura y cómo se ejecuta.

En esta sesión se exige que el alumno programe la rutina SCAN que se ha expuesto anteriormente. El alumno utilizará la subrutina INIT que le han proporcionado los profesores de la práctica y completará el fichero proporcionado por los profesores con la rutina SCAN. A lo largo de esta sesión el sistema de entrega de prácticas corregirá las prácticas que se entreguen cada 15 minutos.

5.2. Sesión 2

En esta sesión se exige que el alumno programe la rutina PRINT. A lo largo de esta sesión el sistema de entrega de prácticas corregirá las prácticas que se entreguen cada 15 minutos. Si el alumno finaliza con la tarea asignada, debe comenzar a resolver el proyecto de E/S.