PLACA EMULADORA DE 680XX SOBRE MACINTOSH

L.Closas, L. y Mota P.
Departamento de Enginyeria Electrònica
Universitat Politècnica de Catalunya
c/ Gran Capitan s/n módulo C4 08034 Barcelona

RESUMEN

El presente trabajo presenta el desarrollo de una herramienta útil y versátil, para realizar la docencia en las Nuevas Titulaciones en el entorno Macintosh de una forma que sea agradable y pedagógica. Además, y como valor añadido el desarrollo del emulador nos permitirá de forma fácil y económica realizar diseños industriales, directanente sobre el prototipo, en el entorno Macintosh.

1.- INTRODUCCION

Es de gran interés en los desarrollos que emplean microprocesadores el disponer de un entorno de trabajo adecuado para poder diseñar de forma fiable y en corto espacio de tiempo. En este tipo de desarrollos es donde se emplean los emuladores o simuladores [1]. Mediante un emulador podemos diseñar y probar en tiempo real, es decir en la placa misma, nuestro propio diseño, por lo que podemos implementar nuestras realizaciones de forma eficiente.

2.- DESCRIPCION DEL EQUIPO

El desarrollo propuesto consta de dos partes, una de software y otra de hardware.

La parte de hardware necesita para su correcto funcionamiento una placa de Macintosh sín ROM-BIOS, y sin hacer falta ningún periférico. La placa de Macintosh deberá tener la posibilidad de expansión de la ROM o en su lugar un conector que permita poner la ROM de comunicaciones desarrollada por nosostros. También hace falta el disponer de un Macintosh (amo) que se comunicará con la placa de Macintosh (esclavo) via port seie tal como se muestra en la fígura 1 siguiente.

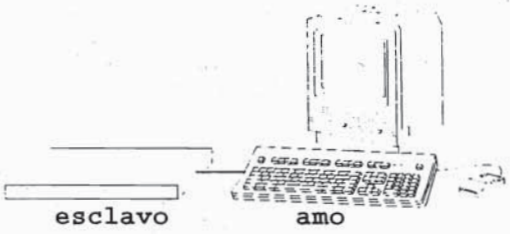

Fig. 1 Conexión Macintosh placa

La comunicación entre el amo y el esclavo se realiza por el puerto serie a 57600 bps, sín bits de stop ni de paridad. El formato de la comunicación incluye unas tramas con la estructura mostrada en la fígura 2.

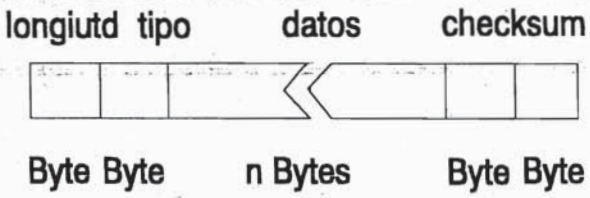

fig. 2 Formato

La trama puede tener una longitud máxima de 256 bytes. El byte que identifica el tipo de trama informa a la placa que información le llega (poner un breakpoint, reubicar un programa, ejecutar una instrucción etc.) y que posteriormente deberá ejecutar. El campo de datos contiene la información que enviamos. Por último, los bytes de checksum (suma de todos los bytes recibidos) que se utilizan para dar un cierto control para la detección y correción de errores. Sí el checksum no es correcto se solicita al amo el reenvio de la trama.

En el lado de la placa (esclavo) la recepción de las tramas se lleva a cabo mediante un polling continuo del chip SCC 8530, port serie, mirando sí hay algún byte en el buffer de recepción.

En el lado del ordenador (amo) la recepción de las tramas es asíncrona (utilizando la Toolbox). Cuando llega un carácter se produce una interrupción y el byte que ha llegado se queda en un buffer circular (parecido a una memoria FIFO) de 1 Kbyte de tamaño. En el programa del amo lee (checkea), cada cierto intervalo de tiempo, sí ha llegado la trama

completa la procesa. La ventaja de emplear este método es que el programa no se para en leer la trama, lo cual ralentizaría el proceso. Otra ventaja de esta forma de comunicación es que es multitarea. En efecto, bajo MultiFinder o con el sistema 7.0 podemos mandar un programa a la placa y volver al Finder u a otra aplicación, el programa seguirá en pantalla pero, algo más lento.

El software de desarrollo de aplicaciones se implementa a dos niveles. En el primer nivel, Macintosh amo, tendremos la edición, compilación en alto y bajo nivel reubicación de programas y programa de comunicaciones.

En un segundo nivel tenemos el software de control de la placa de Macintosh esclava, conectada al amo via port serie. En la ROM BIOS de la placa esclava hemos instalado un programa de comunicaciones que nos permite depurar un programa cargándolo desde el maestro y ejecutándolo sin interrupción o paso a paso en el esclavo.

El proceso de depuración nos permite visualizar en la pantalla del amo, mediante unos determinados comandos, como va evolucinado los contendios de los registros y posiciones de memoria del esclavo. En la figura 3 mostramos una breve descripción de los comandos que puede ejecutar el ordenador amo y que podemos visualizar en su pantalla.

COMANDO	SINTAXIS	DESCRIPCION
Ver y modificar memoria	vm a b,mm a b	Permite visualizar o modificar b bytes de memoria desde la posición a
Ver y modicar registros	<registro>,= valor hexa	Permite visulizar o modificar el contenido de os registros
Cargar programa	c <reubi>	Carga en la placa el programa reubicado a ejecutar
Paso a paso	p	Ejecuta una instrucción y visualiza los registros de la placa
Ejecutar	e	Ejecuta un programa hasta su finalización, para pararlo o hay error
Poner y quitar BreakPoints	pbr y qbr	Permite trabajar con BreakPoints
Desensambla	di a	Traduce intrucciones a partir de la dirección de memoria a
Ir al Toolox	ir	Va al sistema opertivo
Buscar	bu a b	Muestra por pantalla las posiciones de memoria entre a y b
Reset	reset	Inicializa la placa
Información	info	Muestra información de la placa

3.- POSIBILIDADES DEL EQUIPO

El equipo nos permite llevar a cabo desarrollos tanto en alto nivel, utilizando las posibilidades del Macintosh que actua de amo, como en ensamblador.

Para realizar una aplicación nos colocamos en el entorno MPW [2] del Macintosh amo, editanto un programa, para nuestra aplicación específica, que tenga en cuenta las posiciones de memoria de la placa esclava en donde se va a ejecutar. Seguidamente podemos ensamblar, compilar en C o Pascal y reubicar el programa ejecutable para que se pueda ejecutar a partir de una posición de memoria determinada de la placa esclava.

Posteriormente se envía el programa a ejecutar al esclavo, el cual vimos que está haciendo el polling del port serie. El programa de comunicaciones del esclavo nos permite poner el programa que nos envia el amo a partir de la posición de memoria del esclavo que queramos.

Finalmente, una vez instalado en la RAM del esclavo, pasamos a su ejecución sin parar hasta que encuentre una instrucción de break point o bíen a su ejecución paso a paso. Además y gracias al programa de comunicaciones podemos visualizar en el Macintosh amo, la evolución de los contenidos de los registros y posiciones de memoria del esclavo. Todo lo anterior nos permitirá, una fácil, depuración de nuestro programa de aplicación.

Mediante un ejemplo sencillo vemos como funciona todo el sistema, desde edición de un programa hasta la reubicación y depurado. Supongamos que pueremos hacer un programa en C que sume dos números. Dentro del entorno MPW abrimos una ventana y editamos el programa en C siguiente.

```
/* Miprograma*/

int a,b,c;
#pragma segment main
main()
{
        a=3;
        b=4;
        c=a+b;
}
```

La orden #pragma le dice al compilador que elcódigo va en el segmento Main. A continuación desde el mismo entorno MPW compilamos y linkamos el programa .

C Miprograma.c

Link Miprograma.c.a

Es importante no linkarlo con ninguna libreria , pues estas hacen llamadas al Toolbox del Macintosh y en la placa no podemos utilizar el Toolbox.

Seguidamente reubicamos el programa ejecutable para que pueda correr en las posiciones de memoria de la placa

Reubica Miprograma

Una vez que hemos depurado el programa de nuestra aplicación, gracias a la ayuda aportada por el Macintosh amo pasamos a hacer funcionar nuestra aplicación de forma autónoma. Para ello hemos desarrollado la posibilidad de grabar una EPROM con el contenido de la RAM del esclavo (debidamente reubicada), en formato Intel de programación de memorias EPROM . Instalamos la EPROM con el programa depurado en una placa de Macintosh mímima substituyendo su ROM-BIOS por la EPROM de nuestra con lo que la placa solo ejecutará nuestra aplicación en concreto.

4.- MAPA DE MEMORIA EN EL MACINTOSH Y SU GESTION

El Macintosh esclavo presenta un mapa de memoria caracteristico que deberemos tener en cuenta. Para el Macintosh PLUS/SE [3] tiene la distribución de la figura 6

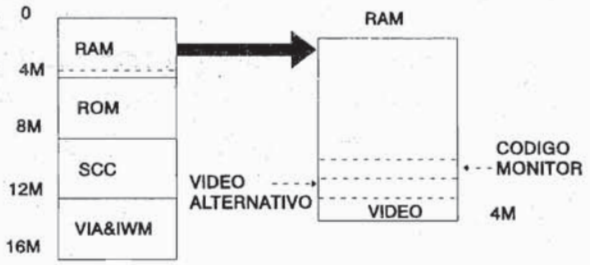

fig. 6 Mapa de memoria de um PLUS/SE

Este mapa nos será de utilidad al reubicar un programa ejecutable para que se pueda ejecutar en la placa.

El sistema operativo del Macintosh divide a los programas ejecutables en dos tipos [4]: a) Segmentos de código limitados a 32 Kbytes y b) Segmentos de datos de tamaño muy grande (2^{32} bytes).

Dentro de cada segmento las posiciones son relativas al origen con lo que se pueden reubicar a partir de cualquier posición de memoria. Además el sistema operativo crea también la tabla de salto que contiene las direcciones de salto entre segmentos, así como información de la dirección absoluta donde empieza cada módulo.

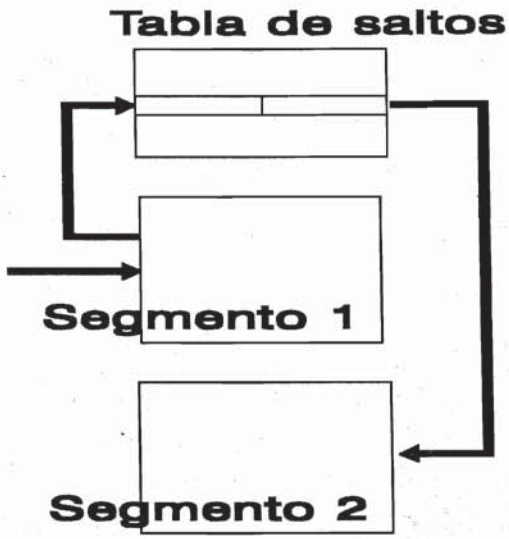

fig.7 Segmentos

Para reubicar, correctamente en la placa, el programa a ejecutar juntamos uno a continuación del otro todos los segmentos de código y volvemos a recalcular la tabla, poniendola al final.

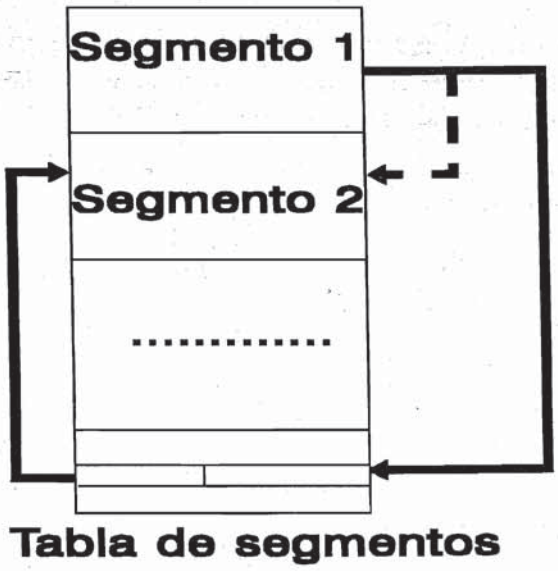

fig. 8 Ejecutable en la placa

Finalmente, se recalcula la tabla de saltos para que el programa juntado este situado a partir de la posición de memoria que queramos.

5.- CONCLUSIONES

En este trabajo hemos presentado un entorno de desarrollo para microprocesadores de la serie 680XX de Motorola, de una forma económica y eficaz. Pues, como placa emuladora nos sirve una placa de Macintosh de serie con la condición de que tenga la posibilidad de quitar la ROM-BIOS que lleva de fabrica para que pueda intercambiarse facilmente con la EPROM de comunicaciones de la placa que hemos diseñado, para ello las placas deben de tener las ROM-BIOS en zocalos de expansión o tener zocalos de expansión de la ROM-BIOS. Mediante la estructura propuesta se podra, facilmente, hacer diseños empleando placas estandar del Macintosh (Bus VME, SCCI, etc.).

6.- AGRADECIMIENTOS

Por ultimo, no nos queda más que agradecer la colaboración de J.Ayet de MacMito, G.Garcia y E.Santos de Apple Computer España.

7.- BIBLIOGRAFIA

[1]. J.Cabestany, A.Más.: Microprocesadores entorno didáctico de aplicación. Mundo Electrónico. Octubre 1989.

[2]. MPW 3.0 Macintosh Programmer's Workshop Development Environment. Apple Computer.

[3]. Guide to the Macintosh family hardware. By Apple Computer Addison Wesley, 1990

[4]. Inside Macintosh. Volume II. Addison Wesley 1985