Práctica 1 2021: Maquinari Mapat en Memòria

Alejandro Rodríguez Arguimbau
DNI: 45372127G
Sergi Mayol Matos
DNI: 45610262C

Índice

1. Introducción

En esta práctica, en lenguaje ensamblador del MC68000, trata de trabajar sobre los conceptos de maquinaria mapeada en memoria y las limitaciones de acceso a la periferia. El problema planteado en esta actividad, consiste en mostrar a través de los displays de siete segmentos, proporcionados por el hardware de EASy68k, un número dependiendo del botón pulsado, que puede variar entre el 0 y el 7, y en el caso de que no se pulse ninguno se enseñe “none”. El problema se debe resolver programando las cuatro subrutinas (MAPINIT, STR2SEG, BITPOS y MAPPRBIT) necesarias para el funcionamiento del programa principal (PRAC1.X68), estas subrutinas se encuentran divididas en dos ficheros externos al programa principal (AUXILIAR.X68 y MAP.X68).

2. Explicación subrutinas

2.1 MAPINIT

La subrutina “MAPINIT” es la encargada de iniciar el hardware del EASy68k y de guardar las direcciones de mapeo. Esta subrutina funciona de la siguiente manera, dependiendo del valor de los dos bits menos significantes la subrutina realizará una función específica. Por tanto, tenemos 4 combinaciones de bits que realizarán las siguientes funciones:

- Si los bits son 00: La subrutina no hace nada.
- Si los bits son 01: La subrutina muestra la ventana de hardware.
- Si los bits son 10: La subrutina guarda las direcciones de mapeo.
- Si los bits son 11: La subrutina muestra la ventana de hardware y guarda las direcciones de mapeo.

El valor de los bits se guarda en D0, y mediante unas instrucciones en el EASy68k podemos comparar el valor de los bits y saber que subrutina iniciar. Tenemos dos subrutinas una que es SHOWWIN que se encarga de iniciar la ventana de hardware (la cual se iniciará cuando los bits sean 01 o 11). La otra subrutina es GTHWADDR que se encarga de guardar las direcciones de mapeo (la cual se iniciará cuando los bits sean 10 o 11).

2.2 STR2SEG

La subrutina “STR2SEG” es la encargada de transformar un String de 8 caracteres a los display de 7 segmentos proporcionados por el hardware de EASy68k. STR2SEG funciona de la siguiente manera, a través del registro de direcciones A0, el cual contiene la dirección del string a mostrar por el display, se obtiene el valor de caracter del String apuntado por A0, se comprueba si está en mayúscula o minúscula, en el caso que sea minúscula se convierte en mayúscula restándole a su valor en ASCII 32, seguidamente se comprueba si es final de la frase, sino es así, se entra en una subrutina de decodificación del carácter, donde se obtiene su valor codificado en el display de 7 segmentos. Una vez se ha obtenido la codificación para el display, el carácter se muestra en los displays a través del registro de direcciones A1, el cual contiene la dirección de mapeo de los display de 7 segmentos. Este proceso se realizará para cada caracter de la frase, es decir, un total de 8 veces. Para hacer posible el recorrido del string y mostrar en cada uno de los displays el valor correspondiente, se debe aumentar la dirección tanto de A0 como A1 para apuntar al siguiente elemento de la frase y al siguiente display una vez obtenido el valor del caracter analizar. En el caso que la frase contenga menos de 8 caracteres solo se mostraran los que

contenga el String y el resto de los display no mostrarán nada. Y en el caso que la oración sea superior a 8 caracteres solo se enseñarán por los displays los 8 primeros.

2.3 BITPOS

La subrutina “BITPOS” es la encargada de devolver un número en D0 dependiendo del valor de los bits. En este caso tenemos que mirar el valor del bit más a la derecha que valga 1, dependiendo de eso devolveremos un número específico en D0. La subrutina funciona de la siguiente manera, al registro D0 le llega un valor que contiene 8 bits, estos bits se van comparando uno a uno a partir de la derecha. A medida que se va avanzando en la secuencia de bits, al encontrar un 1, en D0 se devolverá un número dependiendo de la posición en la que se haya encontrado ese 1.

2.4 MAPPRBIT

La subrutina “MAPPRBIT” es la encargada de administrar, dependiendo del botón pulsado, qué número en forma de String se debe mostrar por los displays de siete segmentos. MAPPRBIT funciona de la siguiente manera: el registro de direcciones A0 contiene la dirección de mapeo de los botones, por lo que accediendo a su contenido y pasándolo a la subrutina BITPOS, a través del registro de datos D0, esta devolverá en D0 qué botón ha sido pulsado. Seguidamente, se accede a una subrutina que devuelve un String del valor del botón pulsado en el registro A0. Una vez ya obtenido el String en A0, se pasa a la subrutina STR2SEG para mostrar el display por pantalla y se restauran las direcciones de A0 y A1.

3. VARS.X68

El fichero VARS.X68 contiene las variables empleadas durante el programa. En este fichero, podemos encontrar cuatro variables y una etiqueta. MAPADDR, la etiqueta, es empleada para la inicialización de las siguientes cuatro variables: MAPSEGAD que contiene la dirección de mapeo de los displays, LEDADDR que contiene la dirección de mapeo de los leds, STCHADDR que contiene la dirección de mapeo de los interruptores y MAPBUTAD que contiene la dirección de mapeo de los botones.

4. CONST.X68

El fichero CONST.X68 contiene las constantes empleadas durante el programa. En este fichero, podemos encontrar: MAPGETAD y MAPSHWHW que contienen los valores para poderlos emplearlos antes de llamar a MAPINIT, STRLNGTH que contiene la longitud máxima del String, la codificación en display de cada letra del abecedario incluyendo el espacio en blanco y finalmente los string correspondientes al valor de cada botón.

5. Conclusiones

Gracias a esta práctica hemos podido mejorar nuestros conocimientos en el EASy68K, además de darnos la oportunidad de aprender por nuestra cuenta. Nuestros mayores obstáculos han sido los siguientes: programar las subrutinas “STR2SEG” y “MAPPRBIT” ya que suponía tener que reflexionar un poco más a la hora de realizarlas. Además el

problema más grande lo tuvimos con “MAPPRBIT”, ya que teníamos que llamar varias subrutinas en una, y nos daba errores constantemente.