UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INFORMÁTICOS

DEPARTAMENTO DE ARQUITECTURA Y TECNOLOGÍA DE SISTEMAS INFORMÁTICOS

(Arquitectura de Computadores)

“Proyecto de Entrada/Salida mediante interrupciones”

Autores:

Víctor Nieves Sánchez *X150375*

Daniel Morgera Pérez *X150284*

“Proyecto de Entrada/Salida mediante interrupciones”

Autores:

Víctor Nieves Sánchez *X150375*

Daniel Morgera Pérez *X150284*

Resumen:

*El objetivo del proyecto consiste en la realización de **operaciones de Entrada/Salida** en un periférico mediante interrupciones en lenguaje ensamblador del **Motorola MC68000**. El dispositivo elegido es la **DUART MC68681** operando ambos puertos serie mediante interrupciones.*

*Además, existe una única rutina de tratamiento de las interrupciones de los puertos que será la encargada transferir la información a o desde los mencionados búfferes internos. El proyecto implica la programación de la **rutina de tratamiento de las interrupciones** (RTI) así como de las subrutinas que constituyen la interfaz.*

Fecha:

13 de mayo de 2019, Campus Montegancedo (Boadilla del Monte),

Universidad Politécnica de Madrid.

Índice

1 Objetivos

Programar en ensamblador del *Motorola MC68000* las rutinas necesarias para poder realizar las operaciones de Entrada/Salida de un periférico, en este caso el *DUART MC68681*.

Testear nuestras propias rutinas para comprobar el funcionamiento correcto del programa.

Realizar un informe que incluya la bitácora del trabajo, así como lo aprendido y destacable durante el desarrollo del proyecto.

2 Metodología de trabajo

Ambos miembros del grupo realizaron el trabajo de manera conjunta y presencial en varios días

(detalles al final del apartado). Para cada rutina del proyecto, el procedimiento era el siguiente:

1. Leer detenidamente la función de la rutina y marcar los puntos claves para su funcionamiento.
2. Dibujar el estado de la pila y de la memoria desde la cual se trabaja.
3. Codificar en papel la subrutina, comentando el cometido de cada instrucción máquina.
4. Pasar el programa en papel a código en el ordenador.
5. Realizar casos de prueba para testear el funcionamiento de la rutina.
6. Testeo de la rutina y, en caso de que no funcione correctamente, volver a repasar el código, poner los puntos de ruptura que fueran necesarios y seguir testeando.

Cabe destacar que las dudas que surgían en la elaboración del proyecto, se contó con los profesores de la asignatura *Santiago Rodríguez de la Fuente* y *Antonio Pérez Ambite*, quienes nos aconsejaban la manera en la cual afrontar y solucionar nuestros problemas.

El tiempo total invertido en el proyecto, por nuestra parte fue de ***56 horas*** aproximadamente.

Respecto a los *test* que debe pasar nuetro programa, de los 59 totales, pasamos ***55 test***, los test que fallan son: *55, 56, 57, 58* y *59.*

A continuacíon se muestra la bitácora de trabajo detalladamente:

05/03/19:

Se han inicializado los datos del proyecto (velocidad, puertos, buffers, ...) y se ha creado la estructura del programa.

No se ha realizado ninguna entrega.

Tiempo: 3 horas.

7/03/19:

Se ha terminado la subrutina LEECAR supuestamente.

No se ha realizado ninguna entrega.

Tiempo: 2 horas.

12/03/19:

Se ha cambiado LEECAR. Aún no funciona.

Se ha empezado ESCCAR, aún estamos pensando en el algoritmo principal del buffer.

No se ha realizado ninguna entrega.

Tiempo: 2 horas.

14/03/19:

Se ha modificado ESCCAR, creemos que funciona correctamente pero falta testear.

Se ha modificado LEECAR, creemos que funciona correctamente pero falta testear.

No se ha realizado ninguna entrega.

Tiempo: 4.45 horas.

15/03/19:

Se ha modificado ESCCAR y se ha testeado un poco.

Se ha modificado LEECAR y se ha testeado un poco.

Se ha realizado la primera entrega. Hay fallos cuando se quiere escribir más de 2000 B.

Tiempo: 3 horas.

18/03/19:

Se ha modificado ESCCAR y LEECAR y se han testeado. Aparentemente funciona correctamente.

Se ha empezado con LINEA. Se ha testeado algunos casos de prueba.

Se ha enviado otra correción.

Tiempo: 4 horas.

Se ha superado el primer hito.

25/04/19:

Se ha realizado SCAN.

No se ha probado.

No se ha realizado ninguna entraga.

Tiempo: 2 horas.

26/04/19:

Se ha mejorado SCAN.

Tenemos algunas dudas que le preguntaremos al profesor.

No se ha realizado ninguna entrega.

Tiempo: 1.30 horas.

29/04/2019:

Se ha empezado RTI y se sigue trabanando en SCAN.

No se ha realizado ninguna entrega.

Tiempo: 5 horas.

30/04/2019:

Se sigue trabanajdo en RTI.

SCAN no se ha testeado (con RTI).

No se ha realizado ninguna entrega.

Tiempo: 5h.

06/05/2019:

Se ha mejorado RTI y SCAN.

Se han testeado ambas.

Se ha realizado una entrega.

Tiempo: 5h.

Se han pasado las pruebas de SCAN.

07/05/2019:

Se ha realizado PRINT.

Se ha testeado PRINT.

Se ha realizado una entrega.

Tiempo: 5h.

08/05/2019:

Se ha arreglado PRINT, corrigiendo según los fallos en la entrega anterior.

Se ha realizado una entrega.

Tiempo: 5h.

09/05/2019:

Se ha intentado arreglar los problemas de concurrencia.

Se ha realizado una entrega.

Tiempo: 6h.

10/05/2019:

Se ha intentado arreglar los problemas de concurrencia.

Se ha realizado la última entrega.

Tiempo: 4h.

Se destaca que, se pasó el primer *hito evaluable* se pasí con éxito el día *18/03/2019.*

3 Introduccíon

El microprocesador *MC68000* fue introducido en 1979 y es el primer microprocesador de la familia *M68000 de Motorola*. Es un procesador *CISC*, aunque posee un juego de instrucciones muy ortogonal, tiene un bus de datos de 16 bits y un bus de direcciones de 24 bits,

El *MC68681* es un módulo de entrada/salida perteneciente a la familia *M68000 de Motorola*. Su función es controlar dos líneas series con capacidad de transmisión y recepción asíincrona, a este tipo de dispositivo se le conoce con el nombre de *DUART* (*Dual Universal Asynchronous Receiver/Transmiter*).

4 Pseoudocódigo de los algoritmos utilizados

Todo el código que se ha escrito (salvo los programas principales) tienen un comentarío explicando que se está haciendo. Leyendo detenidamente esos comentarios, se puede ver el algoritmo utilizado en cada una de las subrutinas.

5 Conjunto de casos de prueba

Nosotros mísmos creabamos nuestros propios casos de prueba.

Para las subrutinas del *hito evaluable* (ESCCAR, LEECAR y LINEA) se elaboraba un programa principal que, de manera secuencial ejecutara pruebas para cada subrutina de manera individual, y tras pasar esas pruebas, se hacían pruebas de manera conjunta (primero llamar a ESCCAR y luego a LEECAR). Un ejemplo de programa principal sería:

```
INICIO:MOVE.L #0,D0
       MOVE.L      #$1,D1
       BSR    ESCCAR
       MOVE.L      #2,D1
       BSR    ESCCAR
       MOVE.L      #$3,D1
       BSR    ESCCAR
       MOVE.L      #4,D1
       BSR    ESCCAR
```

Este es un ejemplo de la llamada a ESCCAR por el puerto de recepción de A para que escribiera el carácter ‘1’.

Se hicieron pruebas de este estilo para todos los buffers de las subrutinas del *hito evaluable*.

Las pruebas para el resto de subrutinas (SCAN, PRINT y RTI) eran algo más complejas, ya que para poder testear SCAN o PRINT, era necesario el tener RTI de manera correcta. Un ejemplo de programa principal sería:

```
esp dc.L $100000
buffer    dc.L    $5000
```

```
dirpar  dc.L    0
INICIO: BSR     INIT
        MOVE.L          BUFFER,DIRPAR
        MOVE.W          #$2000,SR
        MOVE.L          #0,D5
        MOVE.L          #$5000,A0
BUC:    MOVE.W          #3002,-(A7)
        MOVE.W          #0,-(A7)
        MOVE.L          DIRPAR,-(A7)
        BSR     SCAN
        MOVE.L          (A7)+,A0
        MOVE.W          (A7)+,D7
        MOVE.W          (A7)+,D7
        ADD.L   D0,DIRPAR
        ADD.L   D0,D5
        CMP.L   #3002,D5
        BNE     BUC
        BREAK
        MOVE.L          BUFFER,DIRPAR

BUCP:   MOVE.W          #3002,-(A7)
        MOVE.W          #0,-(A7)
        MOVE.L          DIRPAR,-(A7)
        BSR     PRINT
        MOVE.L          (A7)+,A0
        MOVE.W          (A7)+,D7
        MOVE.W          (A7)+,D7
        MOVE.L          ESP,D1
BESP:   SUB.L #1,D1
        BNE BESP
        BREAK
```

En este ejemplo primero se escribe con SCAN en el puerto de A y posteriormente se imprime mediante PRINT en el mismo puerto A.

Pruebas de este estilo se realizaron, variando los puertos, la cantidad de caracteres, e intentando forzar posibles errores.

Se realizaron también programas principales en base a los resultados fallaidos en los test de la correción del proyecto.

6 Resultados

El programa funciona correctamente salvo en los casos de las pruebas anteriormente mencionadas (*55, 56, 57, 58* y *59*).

Hemos aprendido a programar en ensamblador, tanto en papel como en el mismo ensamblador, a testear nuestras propias subrutinas y tambíen gracias al proyecto, se han asentado muchos de los conocimientos aprendidos en la parte teórica de la asignatura.

7 Observaciones

Durante el transcurso de la práctica, el principal problema ha sido la concurrencía, ya que, en muchos casos, en nuestros ordenadores todo funcionaba correctamente, pero a la hora de que el corrector los ejecutara, este fallaba.

8 Dificultades encontradas

Como se menciona en el apartado anterior, el principal problema ha sido la concurrencía entre las subrutinas.

Ha sido especialmente dificil el depurar la ejecución del programa cuando, en nuestras máquinas no se producía ningún fallo y en el corrector sí.

9 Comentarios

Agradecer a los profesores del proyecto, *Santiago Rodríguez de la Fuente* y *Antonio Pérez Ambite* los consejos ofrecidos y la ayuda proporcionada durante la realización de esta práctica.