UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INFORMÁTICOS

DEPARTAMENTO DE ARQUITECTURA Y TECNOLOGÍA DE SISTEMAS INFORMÁTICOS

(Estructura de Computadores)

"Proyecto de programación en ensamblador"

Autores:

Víctor Nieves Sánchez *X150375*

Daniel Morgera Pérez *X150284*

“Proyecto de programación en ensamblador”

Autores:
Víctor Nieves Sánchez *X150375*

Daniel Morgera Pérez *X150284*

Resumen:
El proyecto consiste en la programación, en ensamblador del ***Motorola 88110****, de un conjunto de rutinas que realicen el* ***filtrado de una imagen*** *mediante un filtro programable.*

La imagen será una matriz de píxeles, cada uno de los cuales se representa mediante un byte sin signo que especifica su nivel de gris (0 equivale a negro y 255 a blanco).

El filtro está basado en una operación recursiva de convolución con un núcleo representado por una matriz de 3x3 valores enteros que define una matriz 3x3 de coeficientes fraccionarios:

Fecha:

15 de diciembre de 2018, Campus Montegancedo (Boadilla del Monte), Universidad Politécnica de Madrid.

Índice:

1.Objetivos:

Programar en ensamblador del *Motorola 88110* las rutinas necesarias para poder realizar el filtrado de una imagen.

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

Realizar un informe que incluya la bitácora de 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.

Destacar que salvo para las dos primeras subrutinas, si surgían dudas con el resto, se contó con la ayuda del profesor *Manuel M. Nieto*, quien nos aconsejaba la manera en la cual afrontar y encontrar nuestros problemas.

El tiempo total invertido en este proyecto, por nuestra parte fue de ***41 horas*** aproximadamente.

Respecto a los *test* que debe pasar nuestro programa, de los 93 totales, pasamos ***92* test.**

A continuación se muestra la bitácora de trabajo detalladamente.

25/10/2018:

Creamos subrutinas nFiltrados, Comp, y ActualizaFiltro.

Hacemos entrega para la corrección adicional y nFiltrados falla en una prueba.

Tiempo total de trabajo: 6 horas aprox.

26/10/2018:

Intentamos arreglar nFiltrados, hacemos entrega para el hito pero nFiltrados falla en tres pruebas.

Tiempo total de trabajo: 1 hora aprox.

08/11/2018:

Arreglamos los fallos de nFiltrados.

Creamos subrutinas ValorPixel y SubMatriz.

No se sube nada al servidor de pruebas.

Tiempo total de trabajo: 7 horas aprox.

12/11/2018:

Corregimos ValorPixel y creamos subrutina FilPixel.

Realizamos entrega adicional del segundo hito.

Tiempo total de trabajo: 5 horas aprox.

13/11/2018

Corregimos un pequeño fallo en SubMatriz.

Realizamos una entrega para el segundo hito.

Tiempo total de trabajo: 1'30 horas.

20/11/2018

Falló una prueba en submatriz.

Se volvió a corregir SubMatriz. Además se realizó la subrutina Filtro.

No se realizó ninguna entrega.

Tiempo de trabajo: 3'30 horas.

22/11/2018

Corregimos FIltro y creamos FiltRec, aunque esta última no funcionaba.

Realizamos primera entrega adicional, quedan 2.

Tiempo de trabajo: 6 horas aprox.

4/12/2018

Intentamos corregir FiltRec, sin éxito.

No se realizó ninguna entrega.

Tiempo de trabajo: 6 horas aprox.

12/12/2018

Corregimos FiltRec, éxito aparente.

Realizamos segunda entrega adicional, queda 1.

Tiempo de trabajo: 5 horas aprox.

Se quería destacar, que los alumnos realizaron las entregas para todos los *hitos evaluables*, pero en los 2 primeros hitos, fallaron las entregas a la hora de pasar todos los test, ya que hubo algún pequeño fallo en ambas entregas.

Para la entrega final, como ya se ha dicho, se pasan satisfactoriamente ***92 test*** de 93.

3. Introducción:

El *MC88110* es un microprocesador *RISC* superescalar que forma parte de la familia *88000* de *Motorola*. Es capaz de iniciar dos instrucciones cada ciclo de reloj, respetando siempre la apariencia de ejecución secuencial del programa a través del mecanismo de pipeline del secuenciador. Las instrucciones se despachan hacia diez unidades funcionales que trabajan en paralelo.

El simulador del *MC88110* que se utiliza en este proyecto permite configurar distintos parámetros de la memoria principal, de las memorias cache de instrucciones y datos y de la CPU.

4. Conjunto de casos de prueba:

Como se ha nombrado anteriormente, nosotros creábamos nuestros propios casos de prueba. La mayoría de ellos se pueden encontrar en el fichero fuente *filtror.ens* .

Para los casos de prueba se usaban los casos de ejemplo que nos proporcionaba la documentación, así como casos propios con datos elegidos por nosotros y casos de prueba que nos aconsejaba el profesor *Manuel M. Nieto.*

En el fichero principal se pueden ver todos los principales utilizados (uno por subrutina) y algunos datos (matrices) que se usaron.

Los puntos de ruptura se han eliminado para la entrega final, con motivo de que sea más fácil la lectura del código.

5. Resultados:

Salvo un solo fallo (respecto a los test que nos evalúan), el programa funciona correctamente.

Hemos aprendido a programar en ensamblador, tanto en papel como en el mismo ensamblador y a testear nuestras propias subrutinas.

6. Observaciones:

Durante el transcurso de la practica, tuvimos varios problemas a la hora de realizar *FiltRec*, ya que la función fallaba, la anterior entrega pasaba todos los test correspondientes al resto de subrutinas y el código de *FiltRec* era correcto, ya que los datos que se pasaban para llamar a las demás subrutinas se pasaban de forma correcta.

El error por el cual fallaba, era que en realidad la función *Comp* usaba valores sin signo, lo cual era incorrecto.

7. Dificultades encontradas:

Como ya se ha dicho en el apartado *6. Observaciones,* nuestro mayor problema fue el encontrar el fallo que se producía cuando se llamaba a *FiltRec,* ya que se probó cada subrutina por separado, y funcionaban correctamente, pero cuando se las llamaba en conjunto a través de *FiltRec* no.

Lo que más nos ha costado ha sido crear los casos de prueba, así como encontrar los errores una vez habíamos hecho el código de la subrutina.

8. Comentarios:

Lo primero te todo, agradecer a *Manuel M. Nieto* su apoyo y consejos durante la realización de esta práctica.

Un comentario el cual quería hacer constancia (de *Víctor Nieves)*, es que, los test que evalúan las subrutinas no me parecen correctos. Nosotros pasamos todos los test para las subrutinas (todas menos *FiltRec*), pero aún así, había un pequeño fallo en una de esas subrutinas que “estaba correcta”, lo cual nos complicó mucho la tarea de encontrar el fallo, ya que se daba por hecho que la que fallaba era *FiltRec*.