INFORME - DLX

21/09/2021

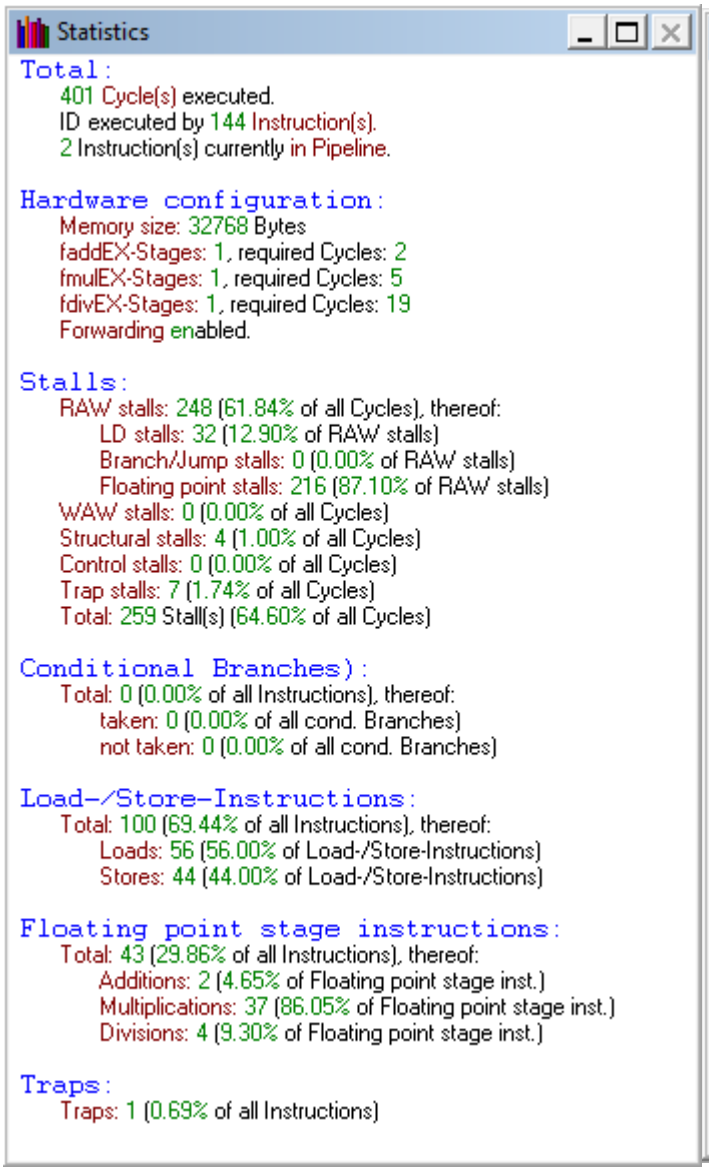
VNiVERSiDAD D SALAMANCA

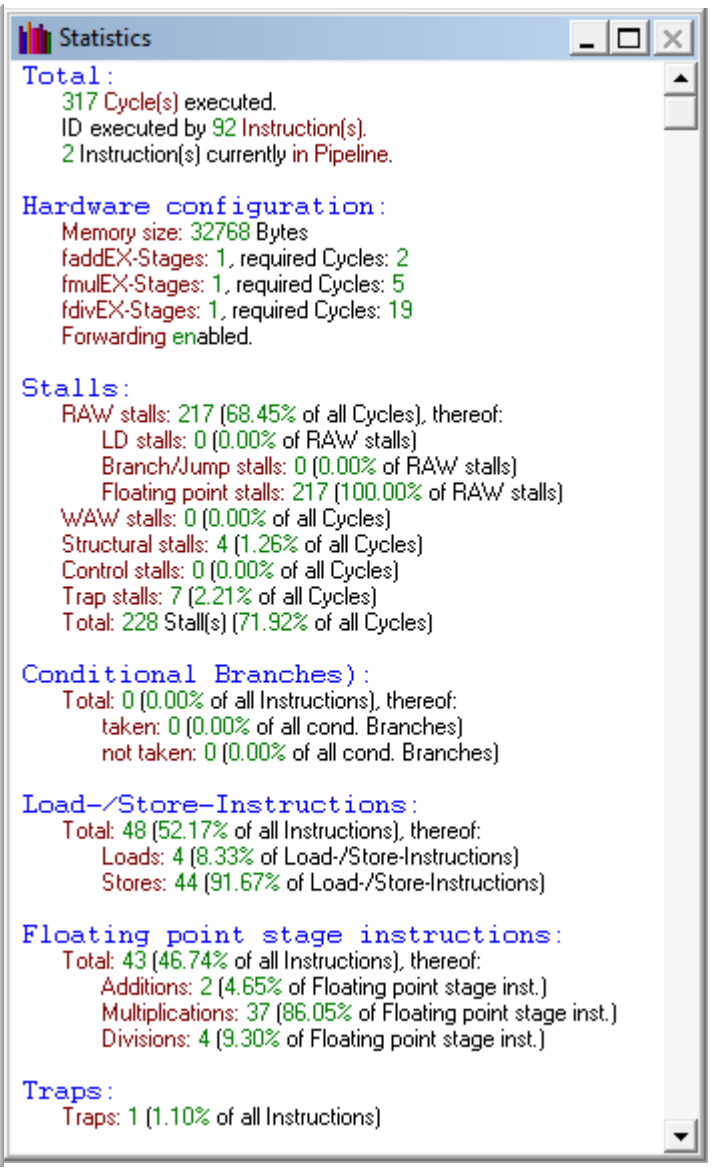
CAMPUS DE EXCELENCIA INTERNACIONAL

Sergio García González - 70921911P
Pablo Jesús González Rubio - 70894492M

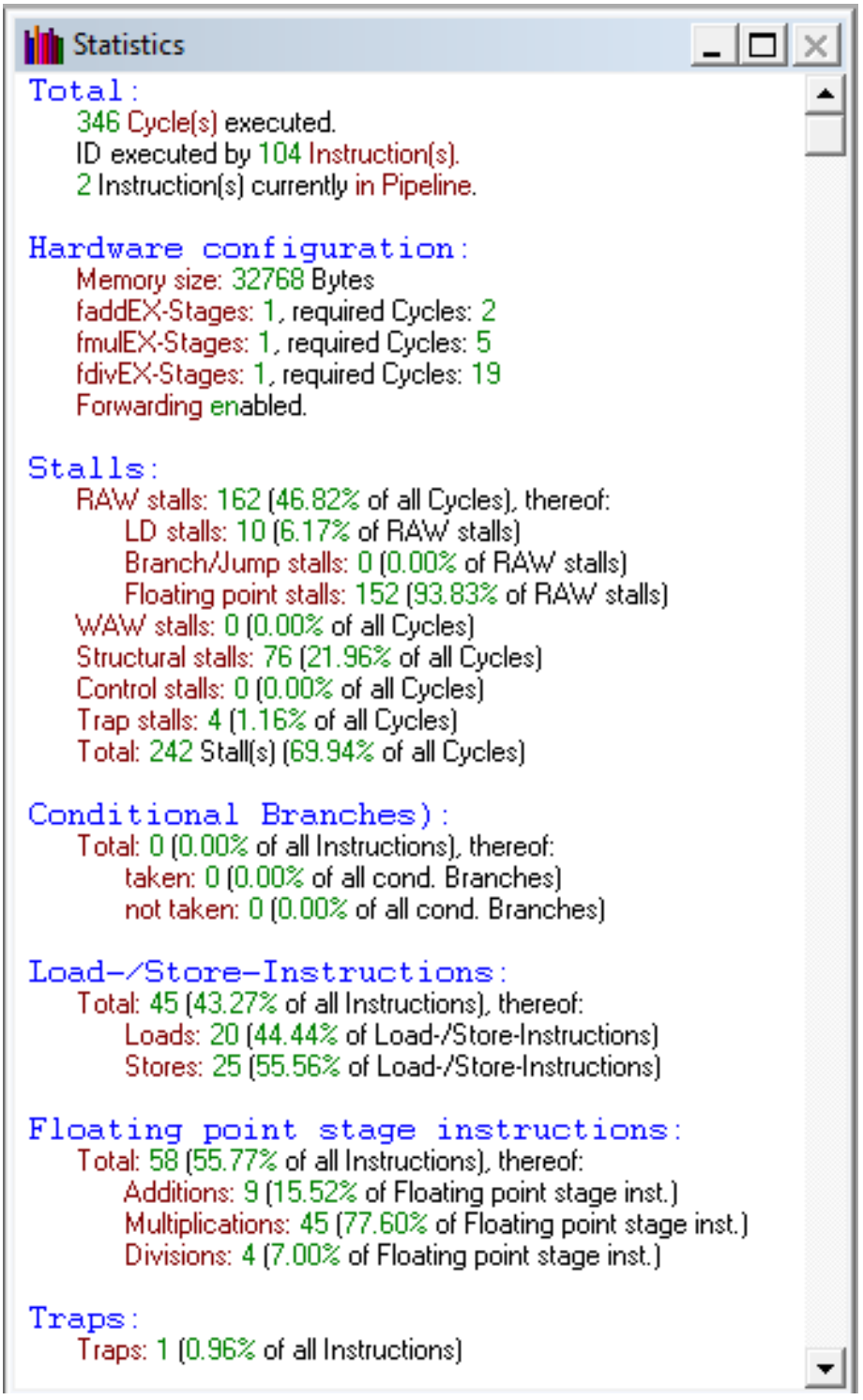
En un primer momento para realizar la versión no optimizada no nos preocupamos mucho por los ciclos que se iban a ejecutar ya que con obtener los resultados correctos nos valdría. Empezamos a realizar las operaciones sucesivamente y en orden.

En la primera versión del programa sin optimizar obtuvimos una ejecución con 401 ciclos (sin realizar el cálculo de la matriz VM, HM y la variable check), aunque antes de ponernos con la optimización eliminamos algunas variables que nos sobraban y sus correspondientes instrucciones “sf”, y pasamos a tener 317 ciclos.

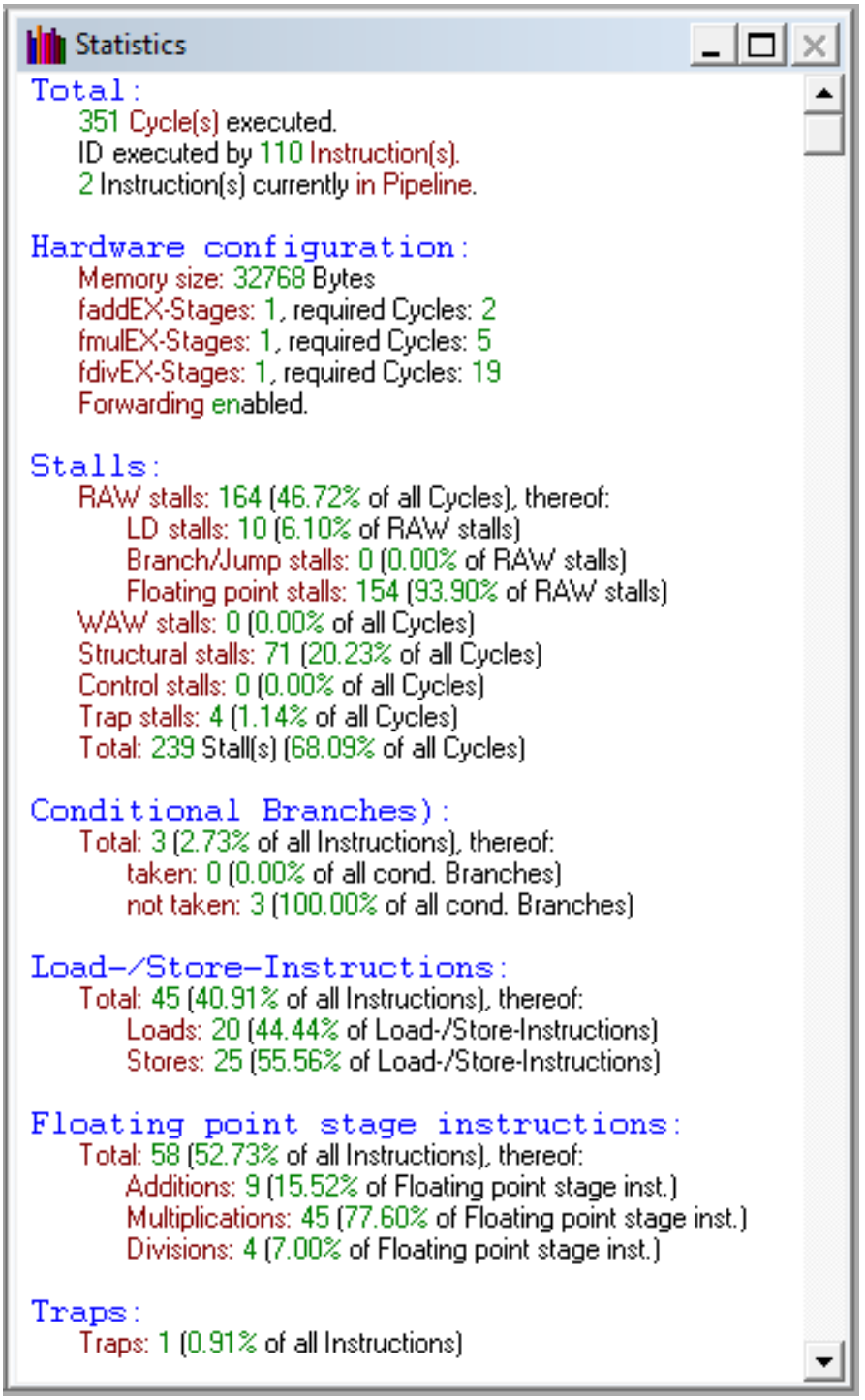




Una vez realizado este paso, realizamos todos los cálculos y obtenemos 346 ciclos sin optimizar el programa y con los resultados correctos.

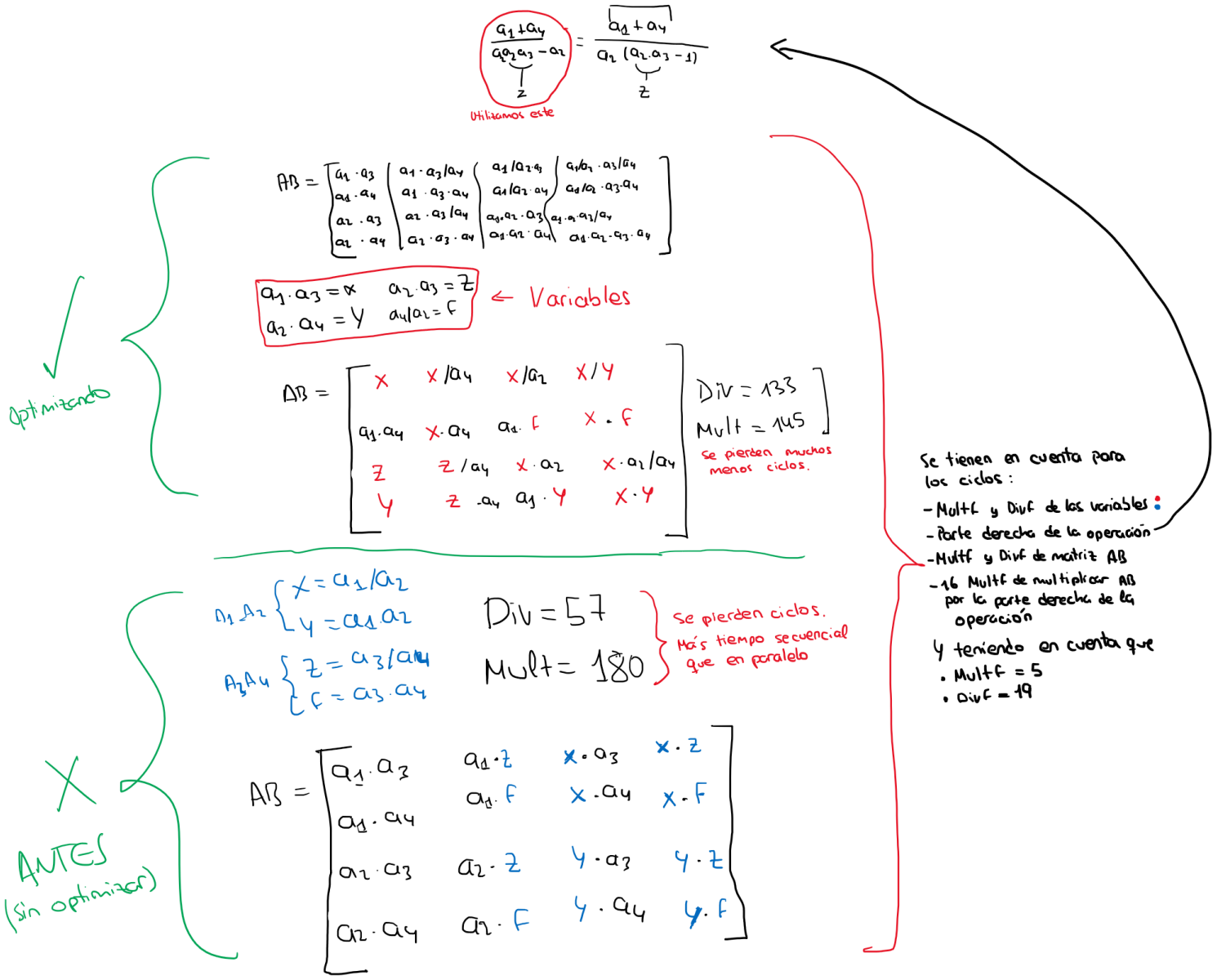


Incluyendo los saltos a fin y comprobaciones de divisiones por cero, aumenta un poco a 351 ciclos.

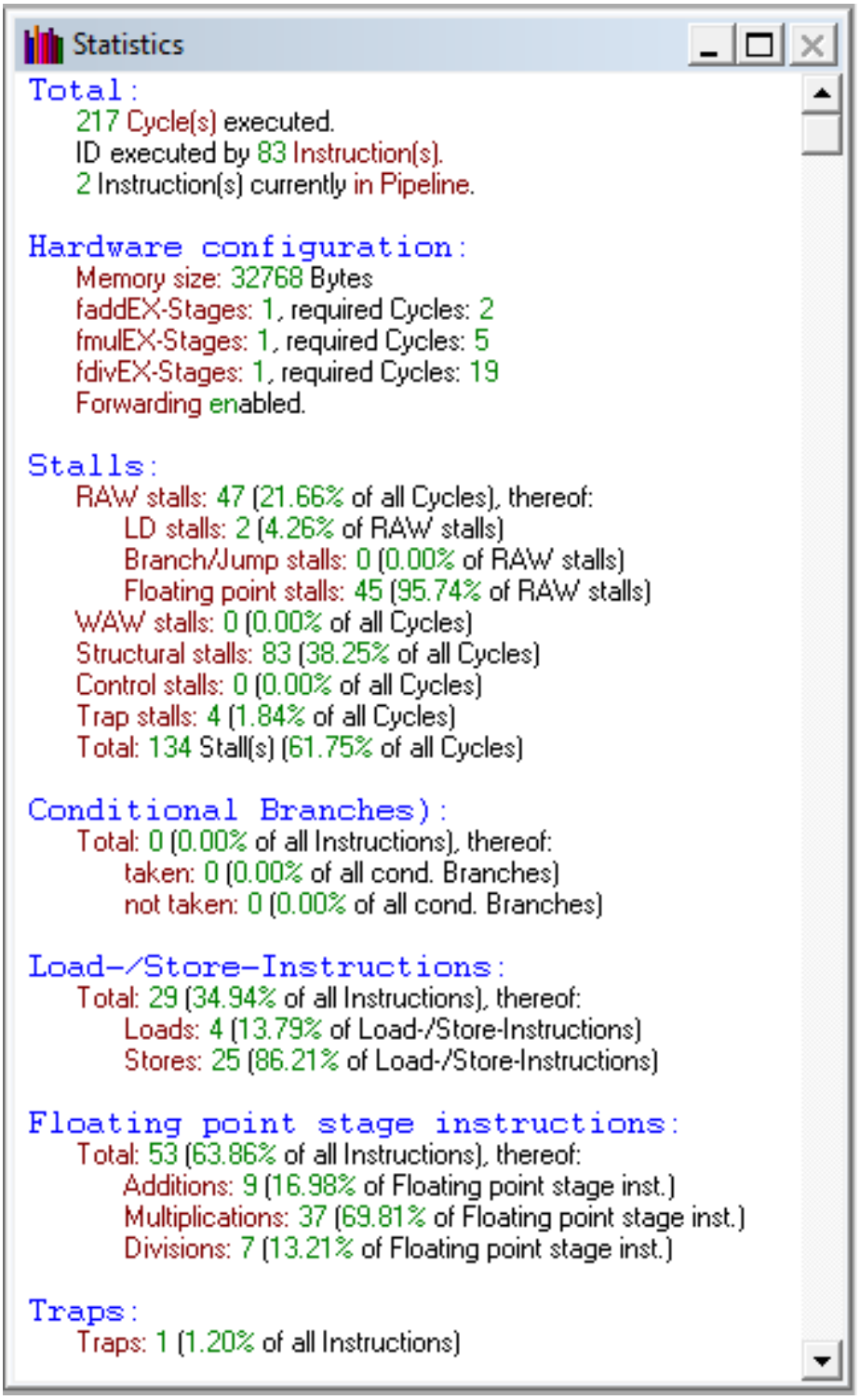


En la optimización del programa lo primero que hicimos fue eliminar todas las variables de almacenamiento que ya habíamos quitado. De esta forma conseguimos ahorrarnos muchas instrucciones de carga y almacenamiento. Lo segundo que hicimos fue hacer una serie de cálculos matriciales para intentar ahorrar algunas instrucciones a la hora de calcular la matriz. Esto lo conseguimos **haciendo factor común de multiplicaciones que se repetían muy frecuentemente** y utilizando variables "X", "Y", "Z" y "F" para renombrar los cálculos.

La siguiente imagen muestra el proceso de los cálculos que hicimos:



Con la realización de estas cuentas conseguimos 217 ciclos:



Lo último que hicimos fue utilizar la técnica de optimización del **reordenamiento de instrucciones** para conseguir que se ejecuten las divisiones, multiplicaciones, sumas y guardado de datos en variables en paralelo. De esta forma, prácticamente nunca se pierden ciclos pues se ejecutan en paralelo estas, siempre y cuando no sean operaciones del mismo tipo, pues estas se ejecutan secuencialmente. (De ahí la idea de optimizar este paralelismo de instrucciones para conseguir menos ciclos).

Una de las partes importantes al reordenar instrucciones era **evitar dependencias** entre instrucciones, como por ejemplo entre un "multf" y un "addf".

Y añadimos los saltos a fin en caso de división por cero (denominador es 0). Importante recordar que la comprobación de divisiones por cero no está puesta encima de todas las divisiones por que no es necesario. Simplemente si el denominador de la división ya se ha comprobado anteriormente no se vuelve a comprobar.

Últimamente conseguimos **203 ciclos**.