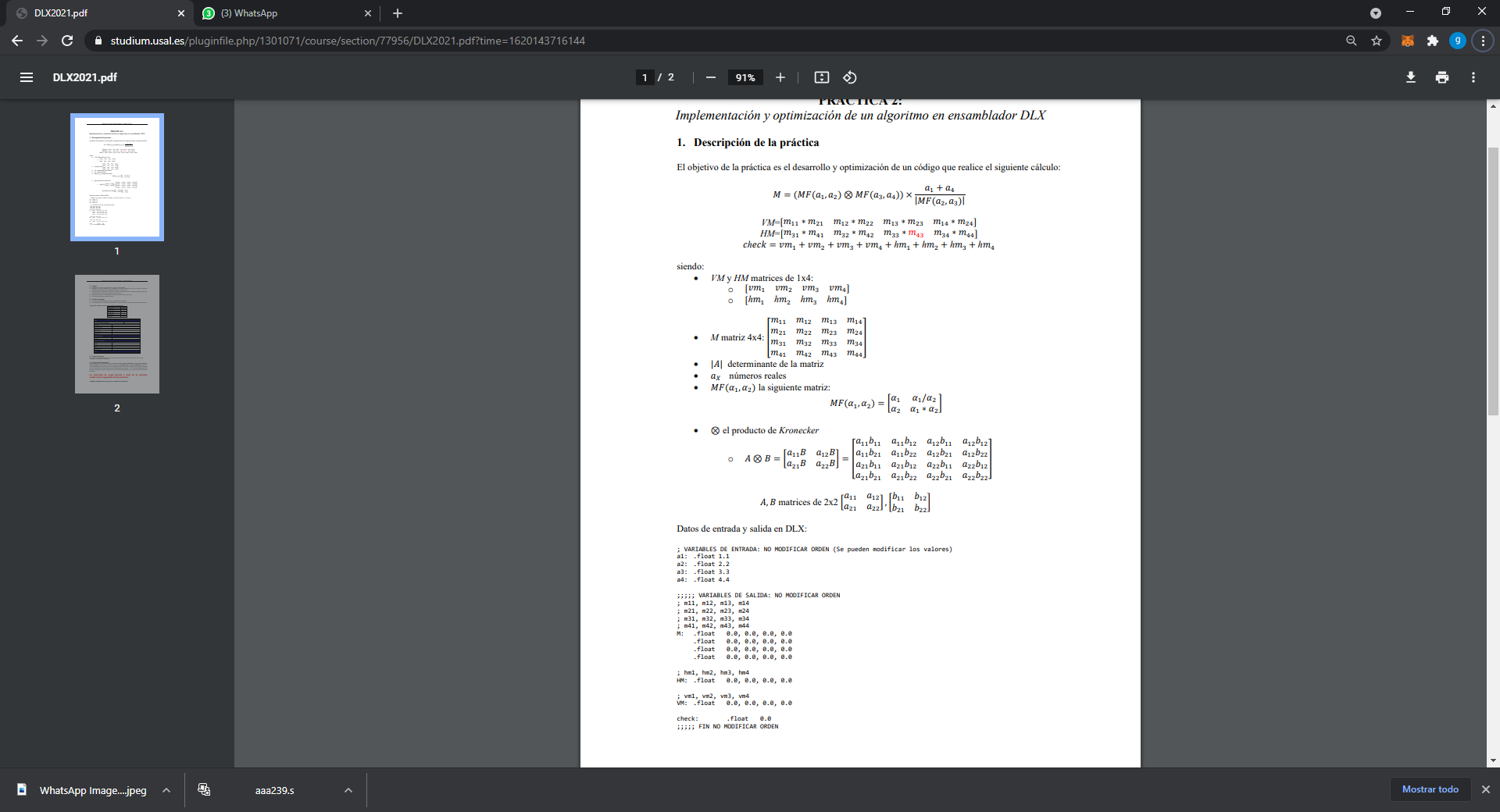
PRÁCTICA DLX

Guillermo Reguera Alfonso

Pablo Sánchez Losada

1. PROBLEMA DLX



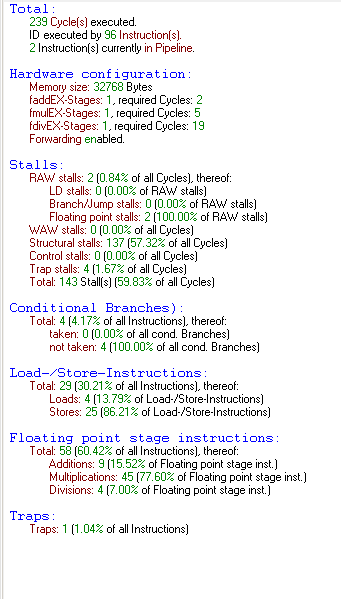
# aLGORITMO SIN OPTIMIZAR

Realizamos el algoritmo mediante pasos sin tener en cuenta ningún tipo de sistema de optimización de operaciones.

Podemos observar cómo después de la ejecución contamos con 325 ciclos, esto es debido a la gran cantidad de Stalls que se producen por no haber planificado el sistema operacional.

Entre los Stalls destacan los RAW stalls y los Floating points Stalls en los que nos centraremos en reducir a la hora de la optimización.

# ALGORITMO OPTIMIZADO



Comenzamos planificando las operaciones a realizar para evitar que ninguna de las ALUs se queden sin trabajo mientras que las demás estan procesando cálculos.

Tambien usamos la mayoría de registros para evitar dependencias entre operaciones.

Después de realizar todos estos cambios podemos observar como los ciclos se reducen a 239 y los RAW Stalls casi a 0, al igual que los Floating point stalls.

Además, añadimos mas conditional branches ya que en la primera version comprobábamos la división del determinante entre 0 pero no el resto de ellas.

Al final, la versión alfa final constaba de 245 ciclos, que, mediante un estudio exhaustivo

de las pipelines, ciclos de reloj y formas de eliminar stalls, se fue reduciendo de ciclo en ciclo hasta llegar a la versión de 239 que se ha obtenido finalmente. Por partes:

La versión de 245 se ha obtenido durante una semana completa de análisis de ciclos de reloj, viendo de qué manera encajarían las operaciones dentro del código sin producir RAW STALLS ni STRUCTURAL STALLS. Puesto que, según tenemos planteado nuestro código tendremos dos RAW STALLS al final.

Finalmente, recolocando ciertas porciones de códigos, pasamos de tener 245 a 243, 242, 240 y finalmente, 239, todo ello optimizando los ciclos de multiplicación.

# RESUMEN

No se podría optimizar mucho más. Si nos fijamos en los resultados del algoritmo optimizado, habría 45 multiplicaciones, 4 divisiones y 9 sumas. Las multiplicaciones son el gran problema y como se ha visto en la solución optimizada, se han reducido en una cantidad significativa los stallas structurales y se han reducido casi por completo los RAW, que únicamente se obitenen como resultado de no tener que realizar mas operaciones entre las sumas y los sf. Con esto, creemos que el código ha sido optimizado prácticamente al completo, considerando con las operaciones e instrucciones con las que trabajamos, y, con lo que creemos que ha sido la mejor solución con las operaciones que hemos empleado.