- no tiene entregas obligatorias, pero habrá 2 entregables voluntarios solo.
- si no se especifica lo contrario, en estas prácticas hablaremos de optimización
  como mejora del tiempo de ejecución, fundamentalmente.
- tipicamente se suele medir el tiempo de ejecucion en los lazos, y suele ser
  útil desenrollarlos.
- la funcion que se utilizara sera gettimeofday(), como dice en la práctica. devuelve
  el tiempo que ha pasado desde una referencia del pasado hasta el actual, con
  precisión de microsegundos.
- un campo que devuelve es el numero de segundos y el segundo campo de la devolucion
  es el numero de segundos. estos se guardan en una estructura del tipo timeval.
- el funcionamiento de final - inicio aparece
- el tiempo a mayores, que hay que restar de esa resta, se denomina overhead.
- el overhead incluye el exceso inicial y final. hay que meterlo siempre en la
  cuenta.
- normalmente se hacen 3 llamadas a gettimeoday, pra medir el overhead en la
  última, aunque el overhead suela ser 0.
- para medir tiempo de ejecución hay que hacer medidas múltiples, ejecutando
  el mismo código muchas veces -> con una sola no es suficiente. la medida que
  obtengo de una sola vez puede que sea irreal.
- para ello, reportamos un estudio estadistico de rendimiento con maximo, minimo,
  media, varianza por lo menos.
- se podrían hacer 100 - 1000 medidas, aunque no se va a pedir tanta precisión.
- hay que redactar en el informe cuántas veces se ha ejecutado el script. el mínimo
  son 10 veces.
- para medir la E/S, evidentemente no se ponen scanfs.
- las operaciones aritméticas son mucho más rápidas que los accesos a memoria, que
  son a su vez mucho más rápidas que las operaciones de E/S, que no son prioritarias
  y es el procesador el que gestiona esto.
- las traducciones de código con gcc suelen ser muy fuertes, debido a las diferentes
  optimizaciones. pueden incluso cambiar las sumas en vez de con add con otras
  instrucciones. este tipo de cambios benefician la mejora de rendimiento.
- para realizar el codigo ensamblador en la practica, habra que hacerlo nosotros.
  si comprobamos que es el mismo que el generado por la herramienta online, tambien
  se podra usar este ultimo.
- es muy imporante tener en mente que compilando con la opcion -S no se genera el
  código ejecutable. Hay que volver a recompiar tras cada cambio con esa opción.
- la compilación estatica trae todo el codigo en el momento de la compilacion. la
  dinamica busca el codigo durante la ejecucion, y es la opcion por defecto.
- un codigo compilado de forma estatica ya es autocontenido. si se utiliza la compilacion
  lacion dinamica, hara falta que la ubicacion de todas las librerias del gcc, por

  ejemplo, sea la misma. si no hemos cambiado nada en la instalacion, deberia
  funcionar de igual manera.
- hay niveles de optimizacion para gcc, igual que hicimos en arqcomp.
- poner -o3 como opcion de compilacion no deberia ser una opcion recomendable, salvo
  ocasiones constastadas que estén bien verificadas. no se puede compilar siempre
  con -o3.
- usar -o0 es muy interesante y el codigo resultante de la traduccion se va a parecer
  mucho al ensamblador que nosotros podriamos escribir a mano.
- con la optimizacion o3 el codigo de las matrices es codigo muerto, y el ensamblador
  estara vacio.
- para el codigo de las matrices, la optimizacion os no es la mejor, luego no siempre se
  comporta la optimizacion como esperamos.
- el desenrollamiento es una tecnica casi utilizada por defecto, y de las mas utiles.