MMX

```
MMX es un conjunto de instrucciones SIMD (Single Instruction Multiple Data) creadas por Intel. La tecnología MMX fue diseñada para mejorar de forma sustancial el rendimiento de las aplicaciones multimedia y de telecomunicaciones. Se basa en un nuevo juego de instrucciones, que se añadían a las ya existentes en la arquitectura 80×86, y en nuevos tipos de datos de 64 bits. Dichas instrucciones trabajaban en paralelo sobre múltiples elementos de datos empaquetados en cantidades de 64 bits.

https://docs.oracle.com/cd/E18752_01/html/817-5477/eojdc.html
```

SSE (Streaming SIMD Extensions)

```
Las SSE (SSE2, SSE3, SSE4) son unas extensiónes al grupo de instrucciones MMX, que también fueron desarrolladas por Intel, concretamente, para sus procesadores Pentium III (1999). Están especialmente pensadas para decodificación de MPEG2 (códec vinculado a los DVD), procesamiento de gráficos tridimensionales y software de reconocimiento de voz. Con la tecnología SSE, los microprocesadores x86 fueron dotados de setenta nuevas instrucciones y de ocho registros nuevos: del xmm0 al xmm7. Estos registros tienen una extensión de 128 bits (es decir que pueden almacenar hasta 16 bytes de información cada uno).

https://docs.oracle.com/cd/E26502_01/html/E28388/eojde.html
```

NET Bubble Sort SEE (comparando assembler + C)

- ¿Qué hace el programa?

  ```
    Ordena arrays con contenido aleatorio de valor double, usando el algoritmo BubbleSort implementado en C y en Inline Assembly. Se ordenan arrays de longitud 2 cuya longitud  crece expotencialmente despues de cada iteracion hasta tener la longitud 66537, y muestra  el tiempo transcurrido en cada caso, comparando ambas implementaciones.
  ```

- ¿Se utilizan extensiones MMX o SSE?

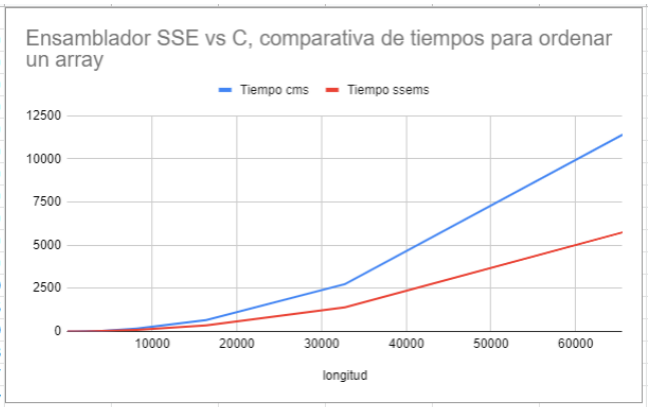
  ```
    SSE, ya que se pueden reconocer instrucciones del set SSE2 tales como minpd, maxpd, movapd...
  ```

- Comenta el funcionamiento de la función: void sortDoublesSSE(Int32 byteCount, double* values)
- ¿Qué ganancia obtenemos con el algoritmo MMX/SSE con respecto al algoritmo secuencial?

  ```
    A medida que la longitud del array va aumentando, podemos observar que la ganancia se va estabilizando en el 100%, siendo la implementacion en SSE el doble de rapida que en C.
  ```

- Realiza una batería de pruebas y muéstralo utilizando gráficas explicativas.

longitud	Tiempo cms	Tiempo ssems	Ganancia %
2	0	0	00,0
4	0	0	00,0
8	0	0	00,0
16	0	0	00,0
32	0	0	00,0
64	0	0	00,0
128	0	0	00,0
256	0	0	00,0
512	0	0	00,0
1024	1	1	00,0
2048	7	4	75,0
4096	34	21	61,9
8192	163	86	89,5
16384	662	358	84,9
32768	2741	1393	96,8
65536	11408	5742	98,7
			v
			(cms / ssems * 100 - 100)



NET Data Transfer (comparando assembler + C)

- ¿Qué hace el programa?
- Explica las siguientes instrucciones: shufpd, cmpltpd, movmskpd
- Explica el funcionamiento de la siguiente función: int DataTransferOptimised(int* piDst, int* piSrc, unsigned long SizeInBytes)
- ¿Qué ganancia obtenemos con el algoritmo optimizado mediante extensiones SIMD con respecto al algoritmo secuencial?
- Realiza una batería de pruebas y muéstralo utilizando gráficas explicativas.

NET Inner Product (comparando assembler + C)

- ¿Qué hace el programa?
- Comenta la siguiente función: float sse3_inner(const float* a, const float* b, unsigned int size)
- ¿Qué ganancia obtenemos con el algoritmo optimizado mediante extensiones SIMD con respecto al algoritmo secuencial?
- Realiza una batería de pruebas y muéstralo utilizando gráficas explicativas.

NET MatrixVectorMult (comparando assembler + C)

- ¿Que hace el programa?
- ¿Qué ganancia obtenemos con el algoritmo optimizado mediante extensiones SIMD con respecto al algoritmo secuencial?
- Realiza una batería de pruebas y muéstralo utilizando gráficas explicativas.