# Notas generales:

* Cada entrada de FAT ocupa 4 bytes (uint\_32)
* La primera entrada FAT nunca se usa
* Todos los MOV\_IN / MOV\_OUT escriben/leen de a 4 bytes (uint32)
* F\_READ: lee un bloque del archivo y lo escribe en un marco de memoria
* F\_WRITE: lee un marco de memoria y lo escribe en un bloque del archivo
* MOV\_IN: lee 4 bytes en la direccion de memoria y lo escribe en un registro de CPU
* MOV\_OUT: escribe los 4 bytes de un registro de CPU en una direccion de memoria

# Prueba Memoria:

* **MEMORIA\_1:** 
  + Tiene 8 paginas (pero en la memoria real entran 4)
  + Escribe un 1 en las primeras 4 paginas del proceso
  + Lleva el contenido de la direccion 16 (pagina 1) al registro DX
  + Lleva el contenido de la direccion 32 (pagina 2) al registro CX
  + Escribe un 2 en las ultimas 4 paginas del proceso
  + Lleva las primeras 4 paginas que habia escrito antes a SWAP (porque en la memoria real entran 4)
* **MEMORIA\_2:** 
  + Tiene 4 paginas (y en la memoria real entran 4)
  + Itera del 20 al 1, y escribe el contador en las 4 paginas del proceso
* **NOTAS:**
  + Los bloques/paginas son de 16
  + Al finalizar MEMORIA\_1 (PID: 1), quedan los primeros 4 bloques del archivo bloques.dat con un 1 (correspondientes a lasp rimeras 4 paginas del proceso que fueron swapeadas)
  + La memoria tiene 4 marcos, por eso hay trashing (con 4 proceso corriendo, se intercalan por bloquearse por page faults)
  + Solicitudes de paginas:
    - Escritura – Pagina 0
    - Escritura – Pagina 1
    - Escritura – Pagina 2
    - Escritura – Pagina 3
    - Lectura – Pagina 1
    - Lectura – Pagina 2
    - Escritura – Pagina 4
    - Escritura – Pagina 5
    - Escritura – Pagina 6
    - Escritura – Pagina 7

**A grid of numbers and numbers on a graph paper

Description automatically generated**

# Prueba Filesystem:

* **FS\_D:**
  + Crea y abre en modo escritura el archivo “fibonacci”
  + Lo trunca de 0 a 160 BYTES (por lo cual le asigna 10 bloques en FS)
  + Iteracion (siempre escribe en la pagina 0 y despues escribe la pagina en un bloque X del archivo)
    - Iteracion 1: escribe un 0d 1d 1d 2d en la pagina y luego en el bloque logico 0 del archivo
    - Iteracion 2: escribe un 3d 5d 8d 13d en la pagina y luego en el bloque logico 2 del archivo
    - Iteracion 3: escribe un 21d 34d 55d 89d en la pagina y luego en el bloque logico 4 del archivo
    - Iteracion 4: escribe un 144d 233d 377d 610d en la pagina y luego en el bloque logico 6 del archivo
* **FS\_A:**
  + Crea y abre en modo escritura el archivo “consolas”
  + Lo trunca de 0 a 64 BYTES (por lo cual le asigna 4 bloques en FS)
  + Primero escribe 83d 79d 78d 89d en la pagina 0 y lo escribe en bloque 0 del archivo
  + Despues escribe 88d 66d 79d 88d en la pagina 1 y lo escribe en bloque 1 del archivo
* **FS\_B:**
  + Abre el archivo “consolas” en modo lectura
  + Se trae el primer bloque del archivo, lo escribe en memoria y lo lleva a registros del CPU
* **FS\_C:**
  + Abre el archivo “consolas” en modo lectura
  + Se trae el segundo bloque del archivo “consolas”, lo escribe en memoria y lo lleva a registros del CPU
* **FS\_E:**
  + Abre el archivo “consolas” en modo escritura
  + Se trae el primer bloque del archivo, lo escribe en memoria y lo lleva a registros del CPU
  + Primero escribe 78d 74d 78d 84d en la pagina 0 y lo escribe en bloque 0 del archivo
  + Despues escribe 69d 78d 69d 68d en la pagina 1 y lo escribe en bloque 1 del archivo
* **NOTAS:**
  + Tenemos 1024 bloques de SWAP, por lo cual la direccion de inicio de los bloques de archivos sera (1024d\*16d)+16d(1er bloque vacio)=16400d=**4010h** en el archivo bloques.dat.
* La tabla FAT debe quedar asi:
  + Primera entrada FAT vacia
  + 4 entradas FAT para el archivo “consolas”
  + 10 entradas FAT para el archivo “fibonacci”
* El archivo de bloques queda asi:
  + A partir de la 4010h, los datos del archivo “consolas”
  + A partir de la 4050h, los datos del archivo “fibonacci”