Ejercicio 1

Para cada una de las siguientes direcciones físicas en base 10, calcule el PF y el offset considerando un tamaño de página de 4 y 8 KB

- 20000
- 32768
- 60000

★ Si la página es de 4KB (2^12), se usarán 12 bits para indicar en qué parte de esa página estamos (offset). El resto de los bits indican el FP.

★ Si la página es de 8KB (2^13), se usarán 13 bits para indicar en qué parte de esa página estamos (offset). El resto de los bits indican el FP.

Páginas de 4 KB (2^12):

#PF Offset
20 000 en binario es: 0100 1110 0010 0000 → #PF (en base 10) = 4
#PF Offset
32 768 en binario es: 1000 0000 0000 0000 → #PF (en base 10) = 8
#PF Offset
60 000 en binario es: 1110 1010 0110 0000 → #PF (en base 10) = 14

Páginas de 8 KB (2^13):

#PF Offset
20 000 en binario es: 0100 1110 0010 0000 → #PF (en base 10) = 2
#PF Offset
32 768 en binario es: 1000 0000 0000 0000 → #PF (en base 10) = 4
#PF Offset
60 000 en binario es: 1110 1010 0110 0000 → #PF (en base 10) = 7

Ejercicio 2

Considere un sistema de swapping en el cual la memoria posee los siguientes espacios libres (en orden): 10, 4, 20, 18, 7, 9, 12 y 15 MB. ¿Qué espacio libre será ocupado por 3 solicitudes consecutivas de 12, 10 y 9 MB para cada una de los siguientes algoritmos?

- First fit
- Best fit
- Worst fit
- Next fit

10 MB | 4 MB | 20 MB | 18 MB | 7 MB | 9 MB | 12 MB | 15 MB
A B C D E F G H

1. First fit:

Se solicitan 12 MB → Se ocupa el espacio C de 20 MB
Se solicitan 10 MB → Se consume el A de 10 MB
Se solicitan 9 MB → Se consume el D de 18 MB

2. Best fit:

Se solicitan 12 MB → Se ocupa el espacio G de 12 MB
Se solicitan 10 MB → Se consume el A de 10 MB
Se solicitan 9 MB → Se consume el F de 9 MB

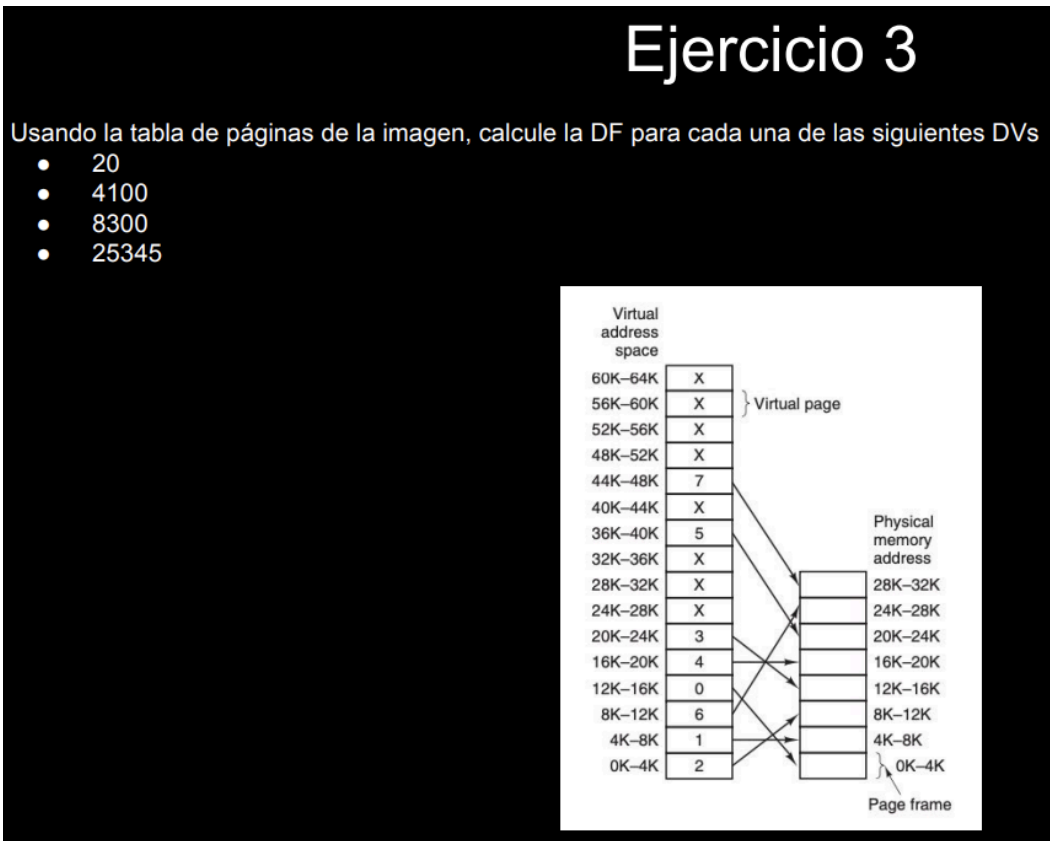
3. Worst Fit:

Se solicitan 12 MB → Se ocupa el espacio C de 20 MB
Se solicitan 10 MB → Se consume el D de 18 MB
Se solicitan 9 MB → Se consume el H de 15 MB

10 MB | 4 MB | 20 MB | 18 MB | 7 MB | 9 MB | 12 MB | 15 MB
A B C D E F G H

4. Next Fit: empezamos desde la primera

Se solicitan 12 MB → Se ocupa el espacio C de 20 MB
Se solicitan 10 MB → Se consume el D de 18 MB
Se solicitan 9 MB → Se consume el F de 9 MB



★ Páginas de 4 KB = 2^12, hay 16 páginas en el EDV y 8 páginas en el EDF

DV (virtual address space) = 20 → entrada 0 → VP = 2 → PF = 2 → La página fisica #2 empieza en la dirección 2^(12) tamaño cada página * 2 páginas previas = 8192 → le sumo el offset de 20 → Physical memory address = 8192 + 20 = 8212

DV (virtual address space) = 4100 → La primera dir de la entrada 1 es 2^12 = 4096 → entrada 1 → VP = 1 → PF = 1 → le sumo el offset de 4100 - 4096 = 4 → Physical memory address = 4096 + 4 = 4100

DV (virtual address space) = 8300 → La primera dir de la entrada 2 es 2^12 * 2 = 8192 → entrada 2 → VP = 6 → PF = 6 → La página fisica #6 empieza en la dirección 2^12 * 6 = 24576 → le sumo el offset de 8300 - 8192 = 108 → Physical memory address = 24576 + 108 = 24684

DV (virtual address space) = 25345 → La primera dir de la entrada 6 es 2^12 * 6 = 24576 → entrada 6 → VP = X → page fault.

Ejercicio 4

Considere un sistema que tiene un EDV de 48 bits y un EDF de 32 bits

- Si las páginas son de 4KB
  - ¿Cuántas entradas tendrá la tabla de páginas si tiene 1 solo nivel?
  - ¿Cuánto ocupa esta tabla considerando entradas de 32 bits?
- Considerando que este sistema tiene una TLB de 32 entradas y ejecuta un programa cuyas instrucciones caben en 1 página y lee secuencialmente un arreglo de enteros que abarca miles de páginas, ¿Qué tan efectiva será la TLB para este caso?

EDV 48 bits → 2^48 = 256 TB direcciones
EDF 32 bits → 2^32 = 4 GB direcciones

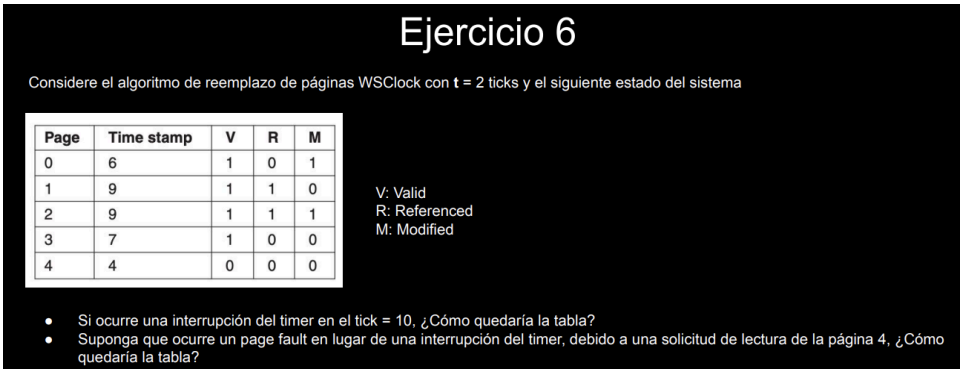
1. Páginas de 4 KB
   a. Tabla de páginas de un solo nivel tendrá tantas entradas como páginas virtuales. Es decir 256 TB / 4 KB = 2^(40 - 10) * 2^(8 - 2) = 2^36 entradas
   b. Como las entradas son de 32 bits = 4 B y hay 2^36 entradas, la tabla ocupa 4 B * 2^36 = 2^38 B = 256 GB
2. Considerando una TLB de 32 entradas (32 páginas virtual de rápida traducción), si se ejecuta un programa cuyas instrucciones caben en 1 página (buenisimo) y lee secuencialmente de un arreglo de enteros que abarca miles de páginas (no good), la TLB es efectiva para los accesos dentro de las mismas páginas, pero va a tener que estar cambiando las entradas constantemente. Como el arreglo se lee secuencialmente, va a usar toda una página con el arreglo antes de traer otras, osea cada página del arreglo en la TLB tiene 1 miss para traerla y después 1023 hits, por lo tanto la TLB es efectiva.

Ejercicio 5

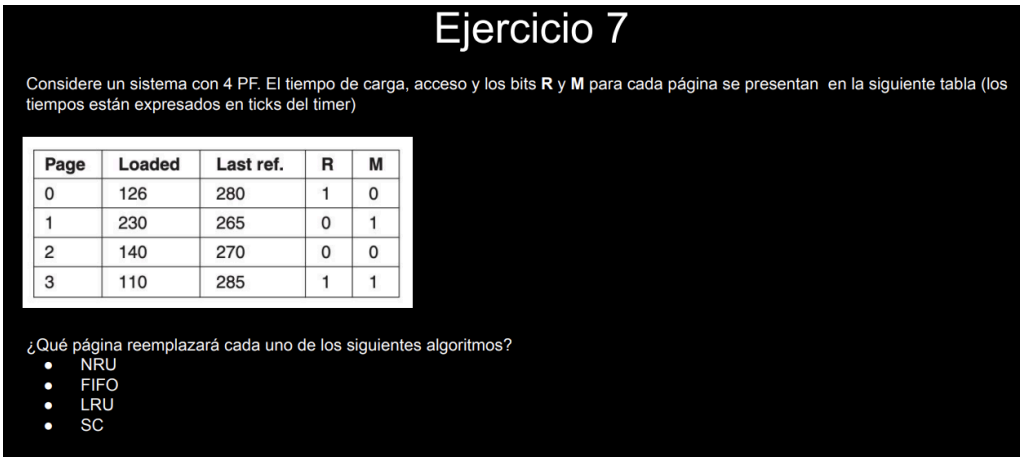
Usando el algoritmo de reemplazo de páginas FIFO en un sistema con 4 PF y 8 VP, ¿Cuántos page faults ocurrirán para la siguiente secuencia de accesos a páginas: 0172327103 si los 4 PF están inicialmente vacíos?
¿Y para LRU?

4 PF y 8 VP
Secuencia de accesos de páginas virtuales: 0 1 7 2 3 2 7 1 0 3 con los 4 PF vacíos
FIFO:
0 → page fault, lo mapea a la página física 0 (podría ser cualquiera, for the sake of simplicity las tomo en orden)
1 → page fault, lo mapea a la página fisica 1
7 → page fault, lo mapea a la página fisica 2
2 → page fault, lo mapea a la página fisica 3 - se lleno la RAM
3 → page fault, desaloja a la página virtual 0 pues fue la primera en ser mapeada, queda libre el espacio físico 0. Se mapea la VP#3 a la PF#0
2 → nada
7 → nada
1 → nada
0 → page fault, desaloja la página virtual 1 pues es la más antigua, queda libre el espacio físico 1. Se mapea la VP#0 a la PF#1
3 → nada
Hay 6 page faults.

LRU:
0 → page fault, lo mapea a la página física 0
1 → page fault, lo mapea a la página fisica 1
7 → page fault, lo mapea a la página fisica 2
2 → page fault, lo mapea a la página fisica 3 - se lleno la RAM
3 → page fault, desaloja a la página virtual 0 pues es la última de las recientemente referenciadas, queda libre el espacio físico 0. Se mapea la VP#3 a la PF#0
2 → nada
7 → nada
1 → nada
0 → page fault, desaloja la página virtual 3 pues es la última de las recientemente referenciadas, queda libre el espacio físico 0. Se mapea la VP#0 a la PF#0
3 → page fault, se desaloja la página virtual 2 pues es la última de las recientemente referenciadas, queda libre el espacio físico 2. Se mapea la VP#3 a la PF#2
Hay 7 page faults



No lo vimos, no entra en el parcial. Hacer para el final. // todo



1. NRU usa los bits R y M. El orden es (no R ^ no M) > (no R ^ M) > (R ^ no M) > R ^ M

El algoritmo NRU va a reemplazar la página 2

2. FIFO saca al que se mapeo primero (mayor tiempo de carga, menor cantidad de ticks del timer cuando fue mapeado)

El algoritmo FIFO va a reemplazar la página 3

3. LRU saca el que hace mas tiempo fue referenciado

El algoritmo LRU va a reemplazar la página 1

4. SC saca el primero mapeado que tiene el bit R en cero

El algoritmo SC va a reemplazar la página 2

Ejercicio 8

Considere la siguiente matriz bidimensional

```
int X[64][64]
```

Suponga que un sistema tiene 4 PFs y cada uno tiene 128 palabras (un número entero ocupa una palabra). Los programas que manipulan la matriz X caben exactamente en una página y siempre ocupan la página 0. Los datos se intercambian dentro y fuera de los otros 3 PFs. La matriz X se almacena de manera tal que X[0][1] sigue a X[0][0] en memoria. ¿Cuál de los dos fragmentos de código que se muestran a continuación generará la menor cantidad de fallas de página? Explique y calcule el número total de faltas de página.

```
//Fragmento A
for (int j = 0; j < 64; j++)
  for (int i = 0; i < 64; i++)
    X[i][j] = 0;

//Fragmento B
for (int i = 0; i < 64; i++)
  for (int j = 0; j < 64; j++)
    X[i][j] = 0;
```

4 PF con 128 ints cada una → 512 = 2^9 ints vs 4096 = 2^12 en la matriz
El programa cabe en una página, siempre va a la cero
Los datos se intercambian dentro y fuera de los otros 3 PFs
En memoria:

X[0][0] X[0][1] … …X[1][0]... X[1][63] (En cada PF entran dos filas de la matriz pues X[0][63] es la 64th 128 - 64 = 64)	
X[2][0] X[2][1] …	X[3][63]
X[4][0] X[4][1] …	X[5][63]

El fragmento A llena cada columna primero, osea que salta más entre páginas que el fragmento B, que llena primero toda una fila.
El fragmento B genera menos page faults.
Cálculo de page faults en total:
A: Hay dos filas por tabla de paginas, osea 32 páginas en total. Busca primero llenar las columnas. Hay 32 page faults por columna. Son 64 columnas, osea **2048 page faults**.

B: Por cada página hay dos filas, osea para 64 filas necesito 32 páginas → **32 page faults** porque todas generan un page fault para necesitar ser cargadas y no hay que volver a utilizarlas.