Informe escrito Laboratorio #2 parte 2
Sistemas operativos

Estudiantes:

Cristian Daniel Muñoz Botero

Profesor:

Danny Alexandro Múnera Ramírez

Departamento de Ingeniería de Sistemas

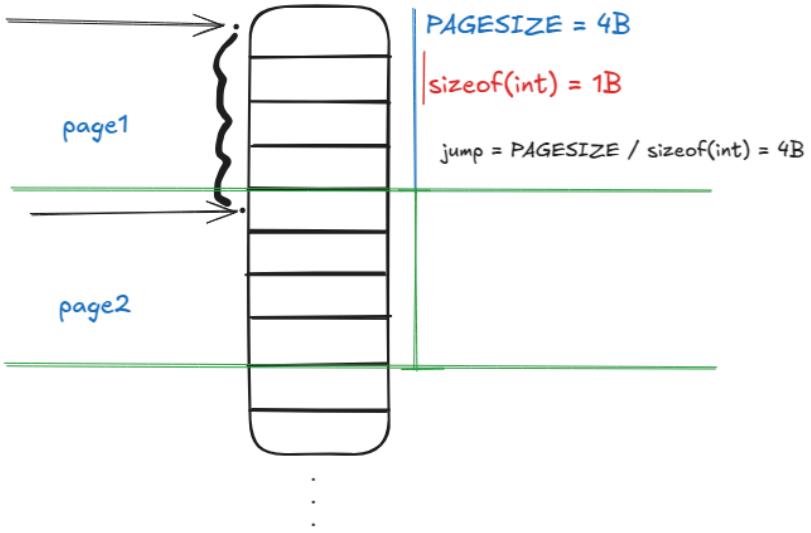
Facultad de Ingeniería

Universidad de Antioquia

2025

1. Se ejecuta el comando **lscpu** para identificar el modelo de la CPU y se recolecta la siguiente información relevante.
   - **1.1. Modelo:** AMD Ryzen 5 3600 6-Core Processor
   - **1.2. Tamaño de direcciones:** 43 bits physical, 48 bits virtual

2. En el código del programa se nos pide inicializar las variables: jump, pages y trials. El tamaño del salto debe ser la cantidad de enteros que caben en una página, porque con esto podemos determinar cuántos espacios de memoria debemos saltar para siempre caer en una página diferente. Por lo que
   jump = PAGESIZE / sizeof(int);



   En los argumentos que recibe el programa, se puede observar que hay un comentario en el cual indica que después del nombre del programa, va el número de pages y la cantidad de trials que se ejecutará el programa. Por lo que
   - 2.1. **pages** = atoi(argv[1]);
   - 2.2. **trials** = atoi(argv[2]);

3. Se crea un script en bash el cual ejecuta un número definido de intentos (10000). Luego ejecuta un ciclo for en el cual el número de páginas lo varía por potencias de 2, desde 1 hasta 14, quedando con los siguientes números de páginas:
   [2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384].
   Luego los resultados los pasa a un csv para su posterior análisis.

nvim run_tlb_test.sh

```
#!/bin/bash

TRIALS=1000

echo "pages,time_ns" > results.csv

for i in {1..14}; do
  PAGES=$((2**$i))   # 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384
  echo "Testing with $PAGES pages..."

  TIME=$(./tlb $PAGES $TRIALS)

  echo "$PAGES,$TIME" >> results.csv
done
```

4. Se ejecuta el script de bash.

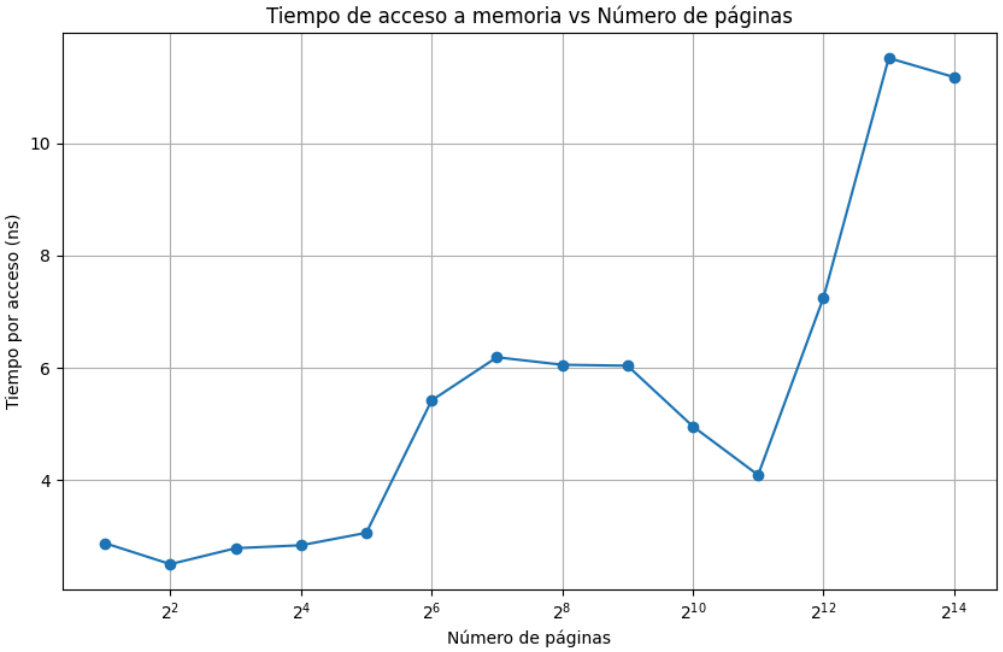
```
[cristian@my-arch lab02-parte2]$ ./run_tlb_test.sh
Testing with 2 pages...
Testing with 4 pages...
Testing with 8 pages...
Testing with 16 pages...
Testing with 32 pages...
Testing with 64 pages...
Testing with 128 pages...
Testing with 256 pages...
Testing with 512 pages...
Testing with 1024 pages...
Testing with 2048 pages...
Testing with 4096 pages...
Testing with 8192 pages...
Testing with 16384 pages...
```

5. Se crea un notebook de Jupyter con el cual observamos el csv que generó la ejecución del programa múltiples veces.

[32]:

	pages	time_ns
0	2	2.880500
1	4	2.507250
2	8	2.791500
3	16	2.843500
4	32	3.065125
5	64	5.420813
6	128	6.189430
7	256	6.056477
8	512	6.039182
9	1024	4.960429
10	2048	4.093968
11	4096	7.249316
12	8192	11.515024
13	16384	11.176144

6. Se grafican los datos:



7. Se pueden observar dos saltos en el tiempo de ejecución del algoritmo.
El primer salto se presenta al pasar de 32 páginas, donde el tiempo de acceso sube de aproximadamente 2 ns a alrededor de 6 ns. Esto indica que el TLB de primer nivel (L1), que tiene 32 entradas, ya no puede almacenar todas las traducciones, lo que produce fallos (misses) en L1 y delega al TLB de segundo nivel (L2).
El segundo se produce después de 2048 páginas, cuando el tiempo sube drásticamente a más de 6 ns y luego al rededor de 11.5 ns. Este comportamiento sugiere que también se llena el TLB de segundo nivel, que tiene una capacidad de 2048 entradas. A partir de aquí, los fallos en TLB requieren acceder a la tabla de páginas en memoria principal, lo cual es mucho más costoso.

L1 TLB con 32 entradas.
L2 TLB con 2048 entradas.