

# Laboratório TLB

Neste laboratório, veremos o impacto dos TLB hit e miss na velocidade dos nossos programas.

Para descobrir o tamanho das páginas no seu sistema, use o comando:

```
getconf PAGESIZE (pode ser PAGE_SIZE dependendo da distribuição)
```

O comando `cpuid` pode dar diversas informações sobre o sistema, mas não deve estar instalado no laboratório.

Para verificar o impacto do tamanho da TLB e dos miss no acesso a dados, vamos fazer um programa que acesse um número específico de páginas na memória diversas vezes (como em um vetor, por exemplo).

Por exemplo: Vamos verificar o tempo necessário para acessar 100 páginas diferentes 1000 vezes:

Se nesse caso o `PAGESIZE` deu 4096 (4k), isso significa que um vetor de 100 páginas deve ter  $4096 \times 100 = 409600$  bytes, o que pode ser um vetor de inteiros (de 4 bytes) com 102400 posições.

Neste caso, para acessar uma página de cada vez, bastaria acessar um elemento a cada 1024 posições (  $4096/\text{sizeof}(\text{int}) = \text{número de inteiros por página}$  )

```
salto = PAGESIZE/sizeof(int);
npags = 100;
for(i=0; i< npags*salto; i+=salto){
    v[i]+=1;
}
```

Este trecho de código acessa 100 páginas diferentes em um mesmo vetor.

O que fazer:

Você deve:

1 - Fazer um programa em C que recebe dois parâmetros: número de páginas *P* e número de iterações *I*, e este programa deve acessar essas *P* páginas *I* vezes, e medir o tempo médio que este processamento leva **por acesso**. Para medir o tempo total, use a função `clock()` da biblioteca `time.h`, da seguinte forma:

```
clock_t inicio, fim;
inicio = clock()
... // faz o que se quer medir
fim = clock()
tempo_sec = (double)(fim-inicio)/CLOCKS_PER_SEC //em segundos
```

2 - Faça um script para rodar o seu programa com diferentes tamanhos de páginas, sempre com o número de iterações = 10000, e jogue esta saída para um arquivo.

3 - Use o programa `plota.py` para plotar seu resultado (`python3 plota.py nomeArquivoSaida`)

4 - Algo que pode ter acontecido é que seu processo trocou de CPU durante os escalonamentos (e cada CPU tem uma TLB diferente). Faça com que seu programa rode sempre no mesmo core, adicionando o seguinte comando no início do mesmo:

```
#include<sched.h>

// No inicio da main()
cpu_set_t mask;
CPU_ZERO(&mask);
CPU_SET(0, &mask);
sched_setaffinity(0, sizeof(cpu_set_t), &mask);
```

O que mudou?

Entregar: `tlb.c` ou `tlb.cc`, dois gráficos gerados, txt com as respostas