

“Para quem só sabe usar martelo, todo problema é um prego”
(Abraham Maslow).

Prática: Medindo Stalls

Paulo Lisboa de Almeida



Compile e execute

Entenda o que o programa loopComlf.c faz.

Compile e execute o programa.

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>

// Vetor ocupa 4GiB de DRAM
#define TAM 1073741824

int main() {
    ssize_t i;
    int *vetor = malloc(TAM * sizeof(int));
    if (vetor == NULL) return -1;

    for (i = 0; i < TAM; ++i) {
        if (i % 2)
            vetor[i] = 1;
        else
            vetor[i + 1] = 0;
    }
    // 10 primeiros
    for (int i = 0; i < 10; ++i) printf("%d ", vetor[i]);
    printf("... ");
    // 10 últimos
    for (i = TAM - 10; i < TAM; ++i) printf("%d ", vetor[i]);

    free(vetor);
    return 0;
}
```

Likwid

Para verificar os contadores disponíveis para sua CPU, utilize:

```
likwid-perfctr -a
```

Procure pelo contador `BRANCH`, ou similar.

Medindo Stalls

Execute o programa com o comando

```
./time likwid-perfctr -C 0 -g BRANCH ./loopComIf
```

Medindo Stalls

Como podemos melhorar o programa para que:

- Ele faça a mesma coisa.
- A quantidade de stalls diminua.
- Diminuir o tempo de execução.

Possível Solução

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>

// Vetor ocupa 4GiB de DRAM
#define TAM 1073741824

int main() {
    ssize_t i;
    int *vetor = malloc(TAM * sizeof(int));
    if (vetor == NULL) return -1;

    for (i = 0; i < TAM; i += 2) {
        vetor[i] = 0;
        vetor[i + 1] = 1;
    }
    // 10 primeiros
    for (i = 0; i < 10; ++i) printf("%d ", vetor[i]);
    printf("... ");
    // 10 últimos
    for (i = TAM - 10; i < TAM; ++i) printf("%d ", vetor[i]);

    free(vetor);
    return 0;
}
```

Possível Solução

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>

// Vetor ocupa 4GiB de DRAM
#define TAM 1073741824

int main() {
    ssize_t i;
    int *vetor = malloc(TAM * sizeof(int));
    if (vetor == NULL) return -1;

    for (i = 0; i < TAM; i += 2) {
        vetor[i] = 0;
        vetor[i + 1] = 1;
    }

    // 10 primeiros
    for (i = 0; i < 10; ++i) printf("%d ", vetor[i]);
    printf("... ");
    // 10 últimos
    for (i = TAM - 10; i < TAM; ++i) printf("%d ", vetor[i]);

    free(vetor);
    return 0;
}
```

Técnica é chamada de **Loop Unroll**.

Essa solução conta ainda com melhorias no despacho múltiplo dinâmico, e acesso a cache. Estudaremos adiante.

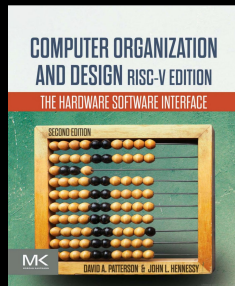


Exercícios

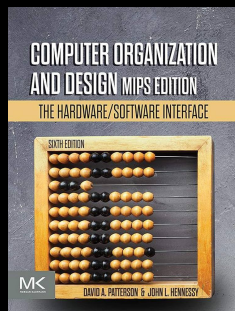
1. Meça novamente o tempo de execução e os stalls. Utilize agora a versão melhorada do programa.
2. Teste outros contadores de performance, e outros programas. Tente melhorar o tempo de execução de outros programas seus através dos conhecimentos que você adquiriu.

Referências

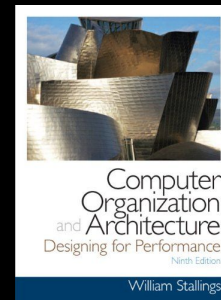
Patterson, Hennessy.
Computer Organization and
Design RISC-V Edition: The
Hardware Software
Interface. 2020.



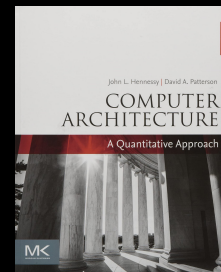
Patterson, Hennessy. Computer
Organization and Design MIPS
Edition: The Hardware/Software
Interface. 2020.



Stallings, W. Organização de
Arquitetura de Computadores.
10a Ed. 2016.



Hennessy, Patterson.
Arquitetura de Computadores:
uma abordagem quantitativa.
2019.



Licença

Esta obra está licenciada com uma Licença [Creative Commons Atribuição 4.0 Internacional](https://creativecommons.org/licenses/by/4.0/).