



**CAIO DE OLIVEIRA LOPES
ISMAEL MARTINS SILVA
LAYSE CRISTINA SILVA GARCIA
LUIZ FELIPE MONTUANI E SILVA**

RELATÓRIO

**LAVRAS - MG
2020**

CAIO DE OLIVEIRA LOPES
ISMAEL MARTINS SILVA
LAYSE CRISTINA SILVA GARCIA
LUIZ FELIPE MONTUANI E SILVA

RELATÓRIO

Relatório apresentado à disciplina GCC123 – Arquitetura de Computadores II,
como parte das exigências apresentadas no plano de curso.

Luiz Henrique Andrade Correia
Professor

LAVRAS - MG
2020

1 Introdução

Será apresentado no decorrer deste relatório as respostas aos exercícios solicitados pelo professor Luiz Henrique Andrade Correia, bem como indicar os respectivos códigos que geram cada uma das respostas ali colocadas (das que forem geradas por meio de códigos).

Na seção desenvolvimento serão apresentados os exercícios seguidos de suas respectivas resoluções.

2 Desenvolvimento

2.1 Exercícios e Respostas

1. Escreva um programa que suporte a entrada de três vetores A, B e C e realize a multiplicação e soma de seus elementos usando a instrução SIMD do AVX (`_mm256_fmadd_`). A entrada deve ser selecionada de acordo com o tipo de dados dos vetores double (4 x 64bits), float (8 x 32bits) ou integer (8 x 32 bits). Apresente o resultado convenientemente na tela.

Código que gera a resposta: Exercicio1.c

Resposta:

A entrada é dada pelo usuário no terminal, aqui está um exemplo:

Digite 3 vetores float com 8 elementos cada.

Cada elemento deve ser separado por um espaço em branco e cada vetor por uma quebra de linha.

```
7.0 7.0 7.0 7.0 7.0 7.0 7.0 7.0
```

```
2.0 2.0 2.0 2.0 2.0 2.0 2.0 2.0
```

```
5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0
```

Resultado da operação de floats:

```
19.000000 19.000000 19.000000 19.000000 19.000000 19.000000 19.000000 19.000000
```

Digite 3 vetores double com 4 elementos cada.

Cada elemento deve ser separado por um espaço em branco e cada vetor por uma quebra de linha.

```
7.000000 7.000000 7.000000 7.000000 7.000000 7.000000 7.000000 7.000000
```

```
2.000000 2.000000 2.000000 2.000000 2.000000 2.000000 2.000000 2.000000
```

```
5.000000 5.000000 5.000000 5.000000 5.000000 5.000000 5.000000 5.000000
```

Resultado da operação de double:

```
19.000000 19.000000 19.000000 19.000000
```

Digite 3 vetores de inteiros com 8 elementos cada.

Cada elemento deve ser separado por um espaço em branco e cada vetor por uma quebra de linha.

7 7 7 7 7 7 7 7

2 2 2 2 2 2 2 2

5 5 5 5 5 5 5 5

Resultado da operação de inteiros:

19 19 19 19 19 19 19 19

2. Escreva um programa que calcule a soma de dois vetores A e B do tipo double. Os vetores A e B devem ser lidos da memória usando a instrução `_mm256_load_pd`. O resultado da operação deve ser armazenado na memória usando a instrução `_mm256_store_pd`.

```
for (int i = 0; i < 4; i++)  
    c[i] = a[i] + b[i]
```

Código que gera a resposta: Exercicio2.c

Resposta:

A entrada é dada pelo usuário no terminal, aqui está um exemplo:

Digite 2 vetores double com 4 elementos cada.

Cada elemento deve ser separado por um espaço em branco e cada vetor por uma quebra de linha.

2.000000 2.000000 2.000000 2.000000

5.000000 5.000000 5.000000 5.000000

Resultado:

7.000000 7.000000 7.000000 7.000000

3. Ao adicionar ou subtrair vetores inteiros, é importante verificar a diferença entre as funções `_add_` e `_sub_` e as funções `_adds_` `_subs_`. O `s` extra significa saturação, que é produzida quando o resultado requer mais memória do que o o vetor pode armazenar. Crie um programa que exemplifique esta situação.000000o ao somarmos dois vetores A e B, no qual seus elementos são definidos como `_mm256_add_epi8` e `_mm256_subs_epi16`. Apresente o resultado convenientemente na tela.

Código que gera a resposta: Exercicio3.c

Resposta:

A entrada é dada pelo usuário no terminal, aqui está um exemplo:

Digite 2 vetores de inteiros com 32 elementos cada.

de linha.

1.000000 2.000000 3.000000 4.000000

5.000000 6.000000 7.000000 8.000000

Resultado:

3.000000 11.000000 7.000000 15.000000

5. Os valores dos vetores A, B e C são do tipo double e devem ser carregado na memória via terminal. Desenvolva um programa que dado que os vetores A, B e C foram carregados, e usando uma instrução FMA (Fused Multiply and Add) realize a operação:

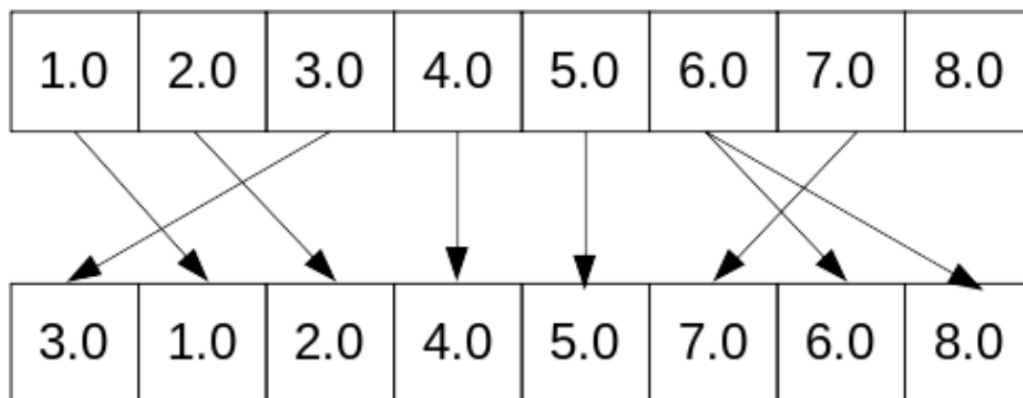
```
for (int i = 0; i < 4; i++) {  
    d[i] = a[i] * b[i]  
    d[i] = d[i] + c[i]  
}
```

Código que gera a resposta: Exercicio5.c

Resposta:

A entrada é dada pelo usuário no terminal, aqui está um exemplo:

6. Os intrínsecos `_permute_` aceitam dois argumentos: um vetor de entrada e um valor de controle de 8 bits. Os bits do valor de controle determinam qual dos elementos do vetor de entrada é inserido na saída. Usando a instrução intrínseca `_mm256_permute_ps` encontre o valor de controle de oito bits e realize a permuta conforme a figura abaixo. Apresente na tela o vetor de entrada, o valor do controle e o vetor de resultado.

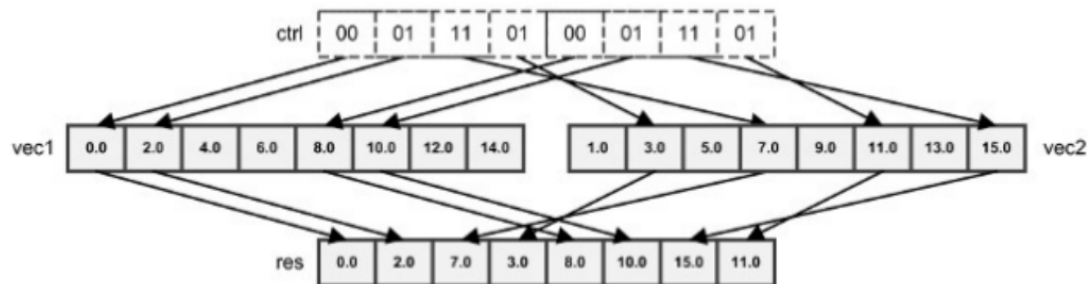


Código que gera a resposta: Exercicio6.c

Resposta:

A entrada é dada pelo usuário no terminal, aqui está um exemplo:

7. A função intrínseca `_shuffle_` seleciona elementos de entrada para um ou dois vetores de 256 bits e os colocam no vetor de saída de acordo com um terceiro elemento de 8 bits, que determina quais elementos devem ser colocados no vetor de saída. Encontre o valor do terceiro elemento e implemente um programa que realize o embaralhamento mostrado na figura abaixo, usando a instrução `_mm256_shuffle_ps (vec1, vec2, 0bxxxxxxx)`.



Código que gera a resposta: Exercício7.c

Resposta:

A entrada é dada pelo usuário no terminal, aqui está um exemplo:

8. Dado o exemplo de código para multiplicação de elementos complexos na página 14 do tutorial, seção 7. Implemente um código em que as entradas são inseridas via terminal, realize a operação solicitada e mostre o resultado na tela. **Explique como são realizados os cálculos.**

Código que gera a resposta: Exercício8.c

Resposta:

A entrada é dada pelo usuário no terminal, aqui está um exemplo:

3 Conclusão

4 Referências Bibliográficas

- Hennessy, John L. and Patterson, David A.. Computer Architecture, Fifth Edition: A Quantitative Approach. Morgan Kaufmann Publishers Inc., 2012.
- Luiz H. A.. Notas de aula da disciplina Arquitetura de Computadores 2. Capítulo 5. Universidade Federal de Lavras. Acessado em Agosto de 2020.
- SCARPINO, Matt. Crunching Numbers with AVX and AVX2. Code Project official page. <https://www.codeproject.com/Articles/874396/Crunching-Numbers-with-AVX-and-AVX>. Acessado em Agosto de 2020.

- Intel Corporation. Intrinsic Guide.
<https://software.intel.com/sites/landingpage/IntrinsicsGuide/>.
Acessado em Agosto de 2020.