# CALAOC - CALCULADORA DE ARQUITETURA E ORGANIZAÇÃO DE COMPUTADORES

**Anderson Cruz** [1]

**Ritiele Aldeburg** [2]

**Bruno Silveira Neves** [3]

**Resumo:**

A cadeira de Arquitetura e Organização de Computadores II (AOC II), presente em alguns cursos da área de Computação, é uma das cadeiras onde os acadêmicos apresentam dificuldades para assimilar elementos do conteúdo programático, tais como arquitetura de cache de um processador, barramentos e periféricos. Para que a taxa de reprovação não seja elevada, surge a necessidade de se usar de artifícios para auxiliar no aprendizado dos alunos. Na edição 2018-01 desta cadeira, foi proposto pelo professor, como trabalho da disciplina, o desenvolvimento de um software que auxilia na verificação dos cálculos realizados na disciplina. O objetivo é que o desenvolvimento deste software contribuísse na aprendizagem dos alunos. O software realiza diversos cálculos de memoria cache diretamente mapeada e de cache associativa, também calcula CPI final de processador e também simula o comportamento de HD no sistema RAID 3 e no sistema RAID 4 e por fim simula comunicação assíncrona (via protocolo handshake). Para facilitar o manuseio do usuario o software foi desenvolvido utilizando interface gráfica da linguagem Java. Os resultados preliminares sugerem que a atividade proposta contribuiu para o aumento da média da turma, considerando uma janela das quatro últimas edições da disciplina.

**Palavras-chave:** CALAOC; arquitetura de computadores; software educacional

**Modalidade de Participação:** Iniciação Científica

**CALAOC - CALCULADORA DE ARQUITETURA E ORGANIZAÇÃO DE COMPUTADORES**

[1] Aluno de graduação. andersonmcruz96@gmail.com. Autor principal

[2] Aluno de graduação. ritialdeburg@gmail.com. Co-autor

[3] Docente. brunoneves@unipampa.edu.br. Orientador

# CALAOC - CALCULADORA DE ARQUITETURA E ORGANIZAÇÃO DE COMPUTADORES

## 1. INTRODUÇÃO

A cadeira de Arquitetura e Organização de Computadores II (AOC II), presente em alguns cursos da área de Computação, é uma das cadeiras onde os acadêmicos apresentam dificuldades para assimilar elementos do conteúdo programático, tais como arquitetura de cache de um processador, barramentos e periféricos (PATTERSON, 2005; STALLINGS, 2002; HENESSY, 2006) . Para que a taxa de reprovação não seja elevada, surge a necessidade de se usar de artifícios para auxiliar no aprendizado dos alunos. Na edição 2018-01 desta cadeira, foi proposto pelo professor, como trabalho da disciplina, o desenvolvimento de um software que auxilia na verificação dos cálculos realizados na disciplina. O objetivo é que o desenvolvimento deste software contribuísse na aprendizagem dos alunos, sendo o mesmo dividido em sete módulos, propostos quinzenalmente, a seguir descritos na próxima seção.

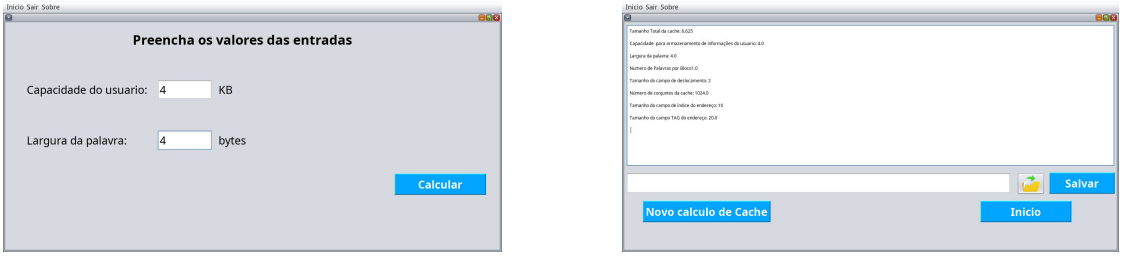
## 2. MÓDULOS PROPOSTOS

Assim que o trabalho foi proposto pelo professor e nos deu liberdade para escolhermos a linguagem de programação desejada. Optamos por desenvolver o mesmo em java (DEITEL, 2005; ORACLE, 2018), pois já existia experiência prévia com esta linguagem e devido também à facilidade de manuseio do software que a interface gráfica proporciona ao usuário.

O professor solicitou que no Módulo A fosse desenvolvido um software que calculasse o tamanho total em KB (kilo bytes) de uma cache direta mapeada, sendo que as informações de entrada que o usuário disponibilizaria para efetuar o cálculo seria a capacidade em KB da cache para o armazenamento de dados e instruções e a largura em bytes da palavra tanto para dados como endereços.

Para atender as necessidades solicitadas, criamos duas janelas de interface gráfica, uma para o usuário prover os dados de entrada, demonstrado na Figura 1(a) e outra para apresentar os resultados dos cálculos, demonstrado na Figura 1(b), também foram criadas classes adicionais para realização dos cálculos.

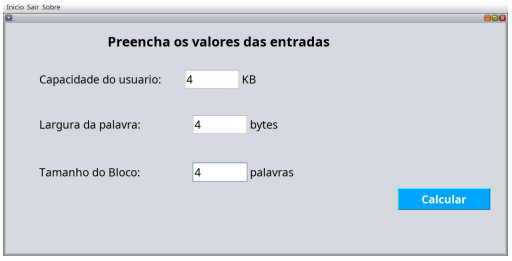
Figura 1 - Interfaces para entrada de dados (a) e exibição de resultados (b) no módulo A .



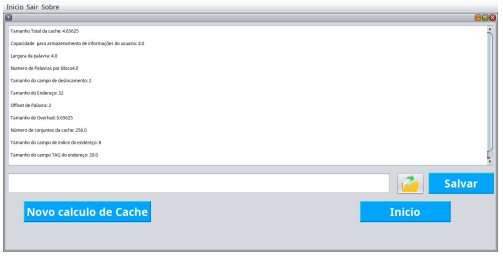

(a) (b)

Já o Módulo B foi dividido em duas partes, sendo a primeira era similar ao Módulo A, com a diferença de que o usuário também informará o tamanho do bloco de memória em número de palavras, pois no Módulo A o tamanho de palavra era restrito a uma única palavra. Nesse Módulo também foi criado uma interface de entrada de dados e outra para apresentar os resultados do mesmo, como ilustra a figura 2, e classes adicionais para efetuar os cálculos.

Figura 2 - Interfaces para entrada de dados (a) e exibição de resultados (b) no módulo B 1° parte.



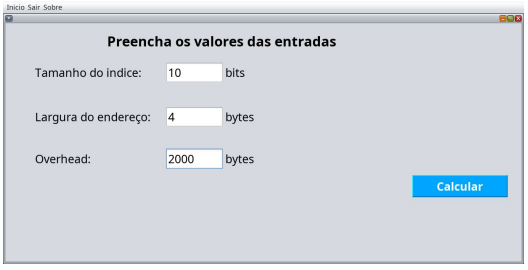

(a)




(b)

Na segunda parte do Módulo B, foi solicitado que o software fosse capaz de calcular o tamanho do bloco de memória em bytes, levando em conta que o usuário fornecerá, o número de bits do campo de índice do endereço, a largura do endereço em bytes e o overhead da cache em bytes, conforme ilustra a figura 3(a).

Figura 3 - Interfaces para entrada de dados (a) e exibição de resultados (b) no módulo B 2° parte.




(a)




(b)

O Módulo C solicita que o software seja capaz de calcular o CPI final de um processador sob influência das falhas de cache. Para realizar esse cálculo, o usuário deve fornecer os seguintes parâmetros: frequência de operação do processador em GHz (gigahertz); latência para acesso a bloco na memória principal (memória RAM), em nanosegundos; latência para acesso a cache de dados nível 2 em ns (nanosegundos); latência para acesso a cache de instruções nível 2 em ns CPI básico do processador, quando executa sem falhas de cache em ciclos; taxa de falhas na cache nível 1 e nível 2 de dados (expressa em porcentagem); taxa de falhas na cache nível 1 e nível 2 de instruções e percentual de instruções do Load/Store.

O software deve realizar os cálculos do Módulo C com e sem a presença da cache de nível 2 (tanto para instruções como para dados). Para isso, foi criado uma interface para o usuario digitar e configurar os parâmetros e outra para apresentar os resultados, conforme mostra a figura 4, sendo que esta última foi dividida para que os resultados pudessem ser apresentados com e sem a cache de nível. Os cálculos são realizados por classes diferentes para manter a organização e facilitar na manutenção do mesmo, caso haja necessidade.

Figura 4 - Interfaces para entrada de dados (a) e exibição de resultados (b) no módulo C.



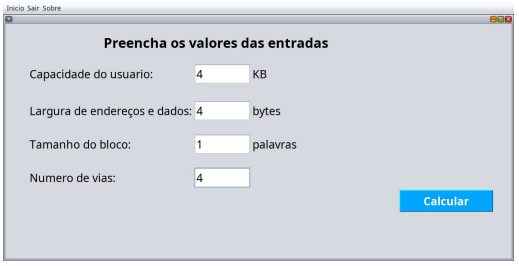

(a)



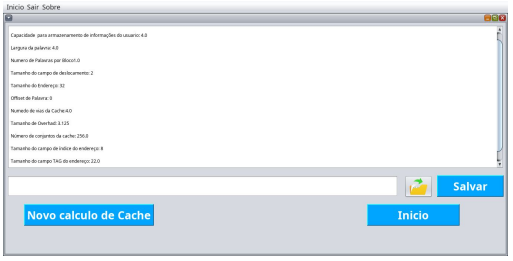

(b)

Para o Módulo D, foi solicitado que o software fosse capaz de calcular o tamanho total em KB de uma cache associativa por conjunto de n vias. O usuário deve fornecer a capacidade em KB da cache para armazenamento de informações do usuário, largura em bytes da palavra (tanto para dados como endereços), o tamanho do bloco de memória em número de palavras e o número de vias da cache, conforme mostra a figura 5.

Figura 5 - Interfaces para entrada de dados (a) e exibição de resultados (b) no módulo D.




(a)



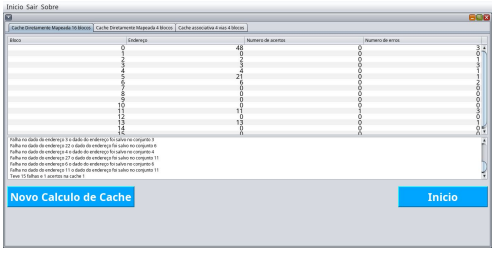

(b)

Para o Módulo E foi solicitado que o software contabilizasse o número de acertos e falhas de acesso para três tipos de cache diferentes cache diretamente mapeada com dezesseis blocos de uma palavra cada; cache diretamente mapeada com quatro blocos de quatro palavras cada e, por fim, uma cache associativa de quatro vias, com dezesseis blocos ao todo, sendo cada bloco de uma palavra. Os endereços ficam armazenados num arquivo txt, que será apontado pelo usuário na interface de entrada de dados, conforme ilustra a figura 6(a). Foram criadas três classes distintas para realizar os cálculos (uma para cada cache) e a interface de exibição dos resultados foi também dividida em três abas: a primeira apresenta o último endereço armazenado em cada posição de memória; a segunda apresenta o número de acertos e falhas para cada bloco, mais abaixo, apresenta uma descrição detalhada dos acontecimentos, como, por exemplo, se houve erro ele informa onde o dado do endereço foi armazenado. Estes elementos estão apresentados na figura 6(b).

Figura 6 - Interfaces para entrada de dados (a) e exibição de resultados (b) no módulo E.




(a)




(b)

A proposta para o Módulo F foi desenvolver um software para emular o comportamento dos sistemas RAID 3 e RAID 4, de modo que ambos os sistemas possuíssem quatro discos em um grupo de proteção e cada disco possuísse um único setor de 4 bytes (tamanho de uma variável inteira). Conforme mostrar a figura 7, existe apenas uma interface na parte superior, onde o usuário pode gravar um disco por vez (com um inteiro) ou gravar os 4 ao mesmo tempo(com um vetor de 4 posições de inteiro), porém ele precisa definir em qual sistema ele deseja gravar, RAID 3 ou RAID 4.

O sistema também tem a capacidade de simular um erro em um hd e, através do disco de proteção, ele consegue restaurar os dados do disco danificado. A parte de simular o erro fica contida ao centro da interface do módulo, onde o usuário seleciona o disco e o sistema (RAID 3 ou 4) para simulação do erro.

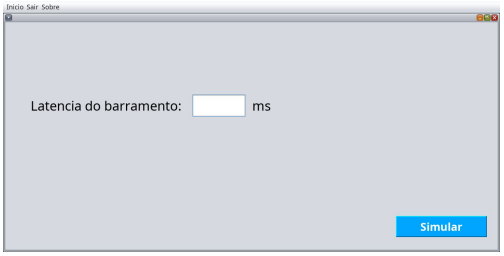
Na parte inferior da interface, o software apresenta os dados contidos em cada disco e o usuário pode escolher o sistema e solicitar para que sejam exibidos os dados contidos nos 4 discos e no disco de proteção. Durante a simulação de erro, o usuário pode gerar o erro e visualizá-lo também neste mesmo setor. Para realizar o controle dos sistemas RAID 4 e RAID 3, possui uma classe para gravação, uma para armazenar os dados gravados e uma que controla a simulação e correção de falha.

Figura 7 - Interfaces do módulo F.

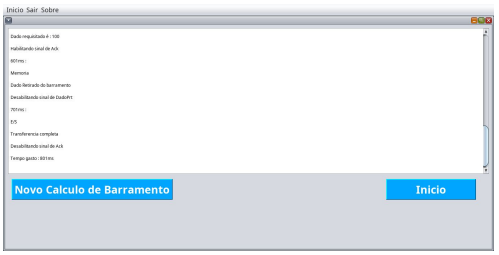



Com relação ao Módulo G, foi solicitado que o software simulasse uma comunicação assíncrona (via protocolo *handshake*) entre um dispositivo e a memória principal. Para isso, o usuário deve informar a latência do barramento e através disso é possível calcular o tempo total da comunicação através do protocolo. Como mostra a figura 8, a interface de resultados descreve passo a passo as ações do protocolo, exibindo as transições ocorridas sobre os sinais de controle. Para que isso fosse possível, a memória foi considerada uma classe e o dispositivo, outra. A comunicação foi modelada através da troca de informações entre essas classes Memória e Dispositivo, levando em consideração a latência do barramento, para calcular o tempo total. A cada passo ele informa o tempo e mostra o que está sendo realizado naquele passo, para que haja um melhor entendimento para o usuário, e por fim apresenta o tempo total gasto para realizar todos os passos da comunicação, levando em conta os atrasos de simulação.

Figura 8 - Interfaces para entrada de dados (a) e exibição de resultados (b) no módulo G.




(a)

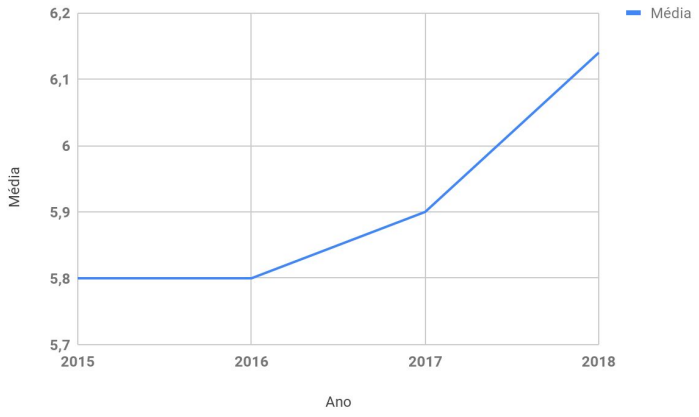



(b)

## 3. RESULTADOS E CONCLUSÕES

Conforme demonstra o gráfico da figura 9, um resultado parcial importante resultante do esforço de desenvolvimento do trabalho proposto foi o aumento da média de aprovação da turma, uma vez que, durante a elaboração do sistema, o aluno já estudava para a prova, visto que precisava de um conhecimento bastante amplo para implementar o software solicitado pelo professor. A perspectiva futura para esse trabalho é que o software seja utilizado pelas próximas turmas de AOC II, criando suporte motivacional e técnico devido a facilidade para rápida criação de gabaritos, para problemas formulados em aula, por meio de uma interface gráfica intuitiva. Espera-se também que os acadêmicos das futuras turmas desta disciplina possam também atuar no aprimoramento deste software, de modo a ampliar o seu rol de recursos para suporte ao aprendizado, facilitando, deste modo, ainda mais o estudo da disciplina Arquitetura e Organização de Computador II.

Figura 9 - Média dos alunos (sem realizar recuperação)




## 4. REFERÊNCIAS


PATTERSON, D.; HENESSY, J. L., **Organização e projeto de computadores: a interface hardware/software**. 3. ed. São Paulo: Elsevier, 2005. 484 p.

ORACLE, **The JVM Specification.** Disponível em: <https://docs.oracle.com/javase/specs/jvms/se7/html/>. Acesso em: 10 abril 2018.

STALLINGS, **W. Arquitetura e organização de computadores**. 5. ed. Prentice-Hall, 2002.

HENESSY, J., PATTERSON, **D. Computer architecture: a quantitative approach**. 4. ed. Morgan-Kaufman, 2006.

DEITEL, H. M.; DEITEL, **P. J. Java: Como Programar**. 6. ed. São Paulo: Pearson Prentice Hall, 2005.