Simulador de Caches para o Processador RISC-V

Yago Martins Escada

Orientadora: Nahri Balesdent Moreano

Faculdade de Computação (FACOM)
Universidade Federal de Mato Grosso do Sul (UFMS)

Resumo. A utilização de simuladores educacionais é essencial no ensino de Arquitetura e Organização de Computadores, pois facilita a compreensão do comportamento dos sistemas modelados e a sua avaliação. Neste contexto, o Educational Cache Simulator (ECS) surge como uma ferramenta valiosa, especialmente após a sua migração para o RARS (RISC-V Assembler and Runtime Simulator). Essa transição representa um aprimoramento significativo, pois retém as funcionalidades do ECS na simulação de memórias cache, ao mesmo tempo que permite a execução de programas RISC-V. Com o ECS acoplado ao RARS, os alunos podem explorar diferentes configurações de cache, observar seu comportamento durante a execução de um programa RISC-V e analisar resultados de desempenho, através de uma interface gráfica intuitiva e interativa. Além disso, o simulador oferece representações gráficas detalhadas da organização da cache e a visualização dos acessos realizados, promovendo uma experiência estimulante de aprendizado.

Abstract. The use of educational simulators is essential in teaching Computer Architecture and Organization, as it facilitates the understanding of the behavior of the modeled systems and their evaluation. In this context, the Educational Cache Simulator (ECS) emerges as a valuable tool, especially after its migration to RARS (RISC-V Assembler and Runtime Simulator). This transition represents a significant improvement, as it retains ECS functionalities in simulating cache memories, while allowing the execution of RISC-V programs. With ECS coupled with RARS, students can explore different cache configurations, observe their behavior during the execution of a RISC-V program and analyze performance results, through an intuitive and interactive graphical interface. In addition, the simulator offers detailed graphical representations of the cache organization and the visualization of the accesses performed, promoting a stimulating learning experience.

1. Introdução

A hierarquia de memórias, especialmente as memórias cache, desempenha um papel crucial no desempenho de um computador [3]. Portanto, é essencial entender como as memórias cache funcionam e como sua organização afeta o desempenho. O uso de um simulador que possibilite a visualização dos eventos que acontecem durante o acesso à memória cache ao executar um programa pode ser uma valiosa ferramenta no processo de ensino e aprendizagem desses conceitos. Uma compilação e comparação detalhada de diversos simuladores de cache é apresentada em [1].

A ferramenta MARS (*MIPS Assembler and Runtime Simulator*) [13] foi criada por Pete Sanderson e Ken Vollmar enquanto eles eram professores nas Universidades de

Otterbein e Missouri, respectivamente, com o objetivo de ser um ambiente de desenvolvimento para a programação em linguagem de montagem MIPS, com o foco voltado para o uso educacional. Como conjunto de instruções MIPS foi amplamente usado por muitos anos como exemplo no ensino de Arquitetura e Organização de Computadores [8], tal ferramenta mostrou-se muito útil. O MARS possui o código aberto e permite que outras ferramentas sejam desenvolvidas e utilizadas de forma acoplada a ela. Assim, o ECS (*Educational Cache Simulator*) [7] foi desenvolvido, também com o foco de uso educacional, com o objetivo de permitir a simulação de uma memória cache durante a execução de um programa MIPS.

Após muitos anos de adoção do processador MIPS no ensino de Arquitetura e Organização de Computadores, um novo conjunto de instruções de máquina, mais moderno, foi criado na Universidade da Califórnia em Berkeley e passou a ser adotado como exemplo [9]. O conjunto de instruções RISC-V foi amplamente aceito pela comunidade acadêmica e pela indústria [12]. Em consequência, a ferramenta RARS (*RISC-V Assembler and Runtime Simulator*) [6] foi desenvolvida, usando como base o MARS. O RARS possui as mesmas finalidades e funcionalidades do seu antecessor, porém foi atualizado para a nova arquitetura.

Este projeto trata do desenvolvimento de uma nova versão do simulador ECS, para uso acoplado à ferramenta RARS e assim permitir a simulação de uma memória cache durante a execução de um programa RISC-V. Tal desenvolvimento envolve a migração do ECS e também a atualização e melhoria de suas funções. Existem outros simuladores do processador RISC-V que incorporam alguma simulação de memórias cache, como por exemplo as ferramentas QtRvSim [4, 5] e Ripes [10, 11]. Essas ferramentas, também de uso educacional, oferecem uma visualização bastante didática da implementação do processador, porém apresentam a simulação da memória cache com menos riqueza de detalhes que o ECS.

As seções a seguir detalham o projeto desenvolvido. A ferramenta ECS e suas principais funcionalidades são apresentadas na Seção 2. O processo de migração do ECS do MARS para o RARS, com reestruturações e correções no código, é descrito na Seção 3. A Seção 4 apresenta as modificações e melhorias desenvolvidas no ECS, assim como a documentação elaborada para a ferramenta. As Seções 5 e 6 ilustram o uso do ECS para a realização de experimentos com programas RISC-V e *traces* de endereços de memória, exemplificando como os estudantes podem estudar os conceitos de hierarquia de memórias através das simulações realizadas. Por fim, a Seção 7 apresenta a conclusão do trabalho e sugere trabalhos futuros.

2. O *Educational Cache Simulator* (ECS)

O *Educational Cache Simulator* (ECS) é uma ferramenta educacional acoplada ao simulador RARS, projetada para simular o comportamento de memórias cache durante a execução de programas RISC-V. Essa ferramenta permite que estudantes visualizem e compreendam o processo de acessos à memória cache em arquiteturas de computadores, tornando o aprendizado mais prático e interativo.

A ferramenta oferece diversas funcionalidades para acesso à cache de dados e/ou instruções, permitindo a configuração detalhada da cache, incluindo tipo, número de blocos, associatividade, tamanho do bloco e política de substituição. Além disso, exibe graficamente a organização da cache e os acessos realizados a cada instrução do programa, proporcionando um acompanhamento detalhado de acertos, falhas e

substituições de blocos. Os acessos à memória simulados pelo ECS podem ser oriundos da execução de um programa RISC-V no RARS, com o ECS atuando como um simulador *emulation-driven*. Alternativamente, também é possível simular acessos à memória oriundos de um *trace* de endereços de memória, fornecido pelo usuário, e nesse caso o ECS funciona como um simulador *memory trace-driven*.

O programa RISC-V desejado é carregado e montado, e então o ECS é acessado pelo menu *Tools*. Dentro do ECS, configura-se a organização da cache, podendo escolher entre caches unificadas, de dados ou de instruções. Após a configuração, o programa pode ser executado diretamente no RARS, com o acompanhamento detalhado dos acessos à cache.

O ECS possui várias abas que oferecem diferentes funcionalidades:

- *Settings*: Permite a configuração da cache a ser simulada e fornece informações detalhadas sobre a divisão do endereço de memória nos campos que permitirão o acesso à cache. A Figura 1 mostra esta aba.

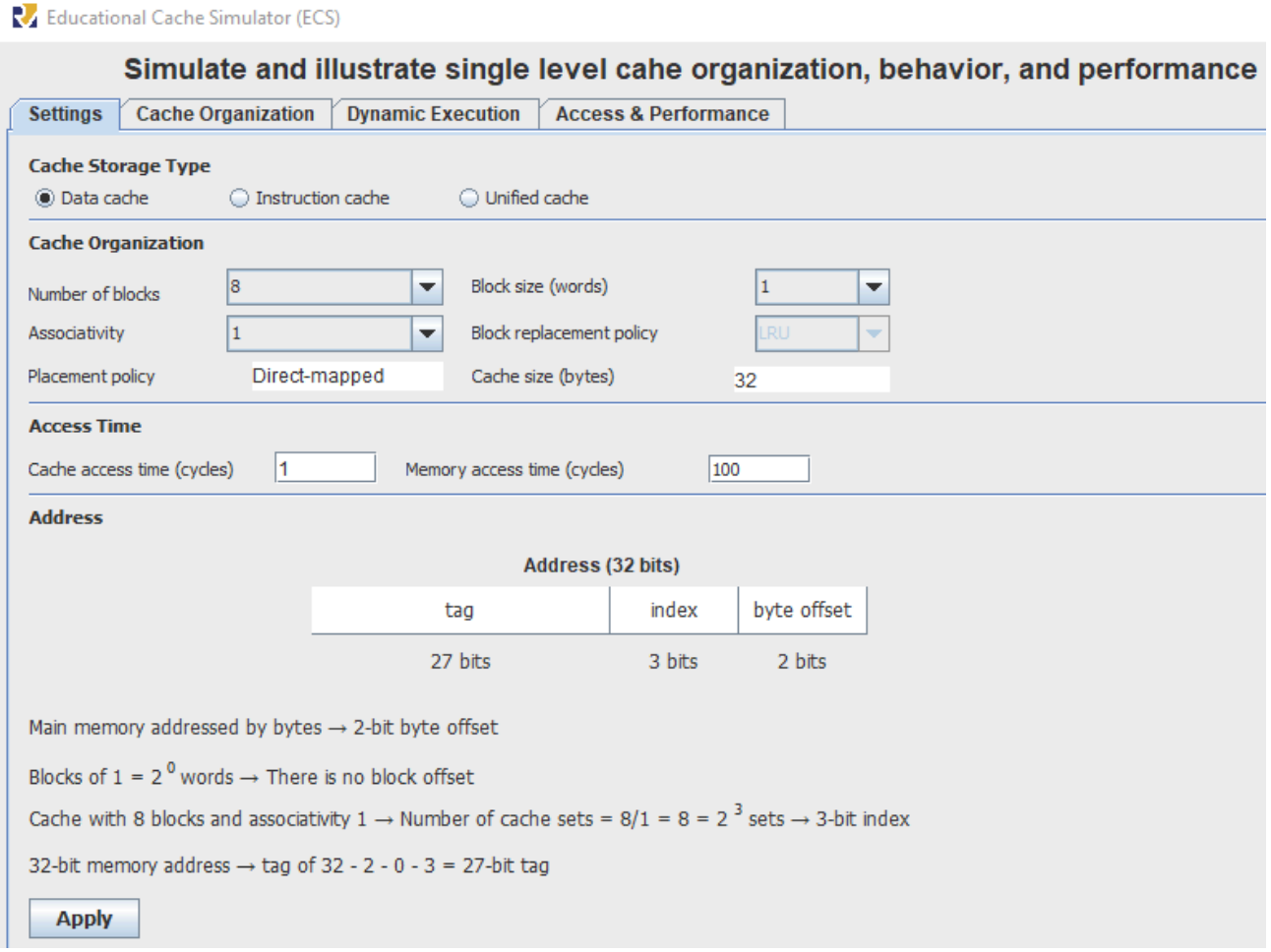

Figura 1. Aba *Settings* do ECS.

- *Cache Organization*: Exibe graficamente a organização da cache, cuja configuração foi escolhida na aba *Settings*, facilitando a compreensão de como os dados estão dispostos nos blocos e como os campos do endereço de memória são utilizados nos acessos à cache. A Figura 2 demonstra esta aba.

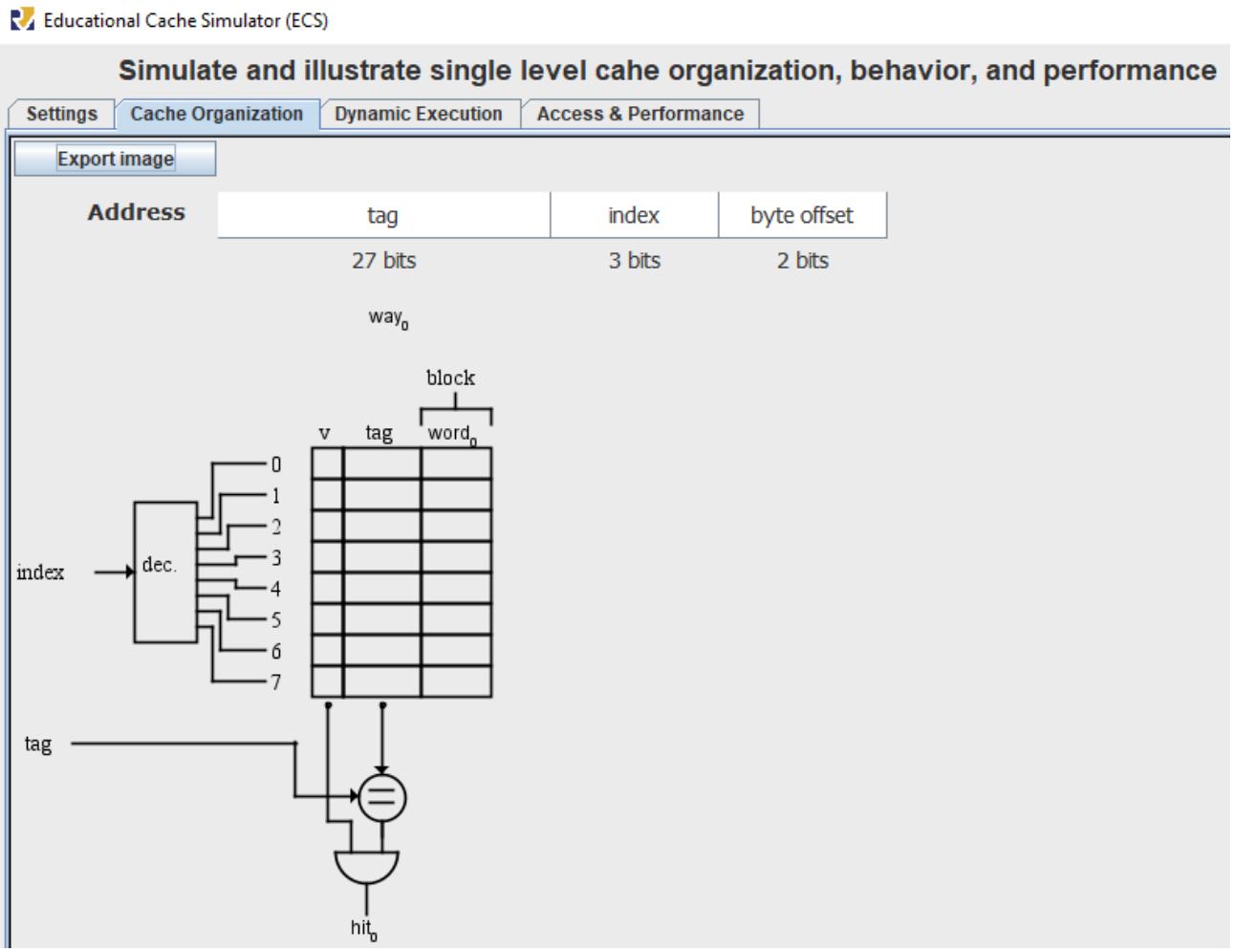

Figura 2. Aba *Cache Organization* do ECS.

- *Dynamic Execution*: Permite acompanhar, durante a execução do programa, cada acesso à cache, indicando se houve acerto ou falha, bem como o endereço de memória e a posição na cache acessada. Esta aba é demonstrada na Figura 3.
- *Access & Performance*: Exibe métricas de desempenho, como o número de acessos, acertos, falhas e substituições de blocos, além da taxa de falhas e o tempo médio de acesso à memória. Apresenta também uma tabela com informações sobre todos os acessos realizados. Também permite a carga e simulação de um *trace* de endereços de memória. Esta aba é apresentada na Figura 4.

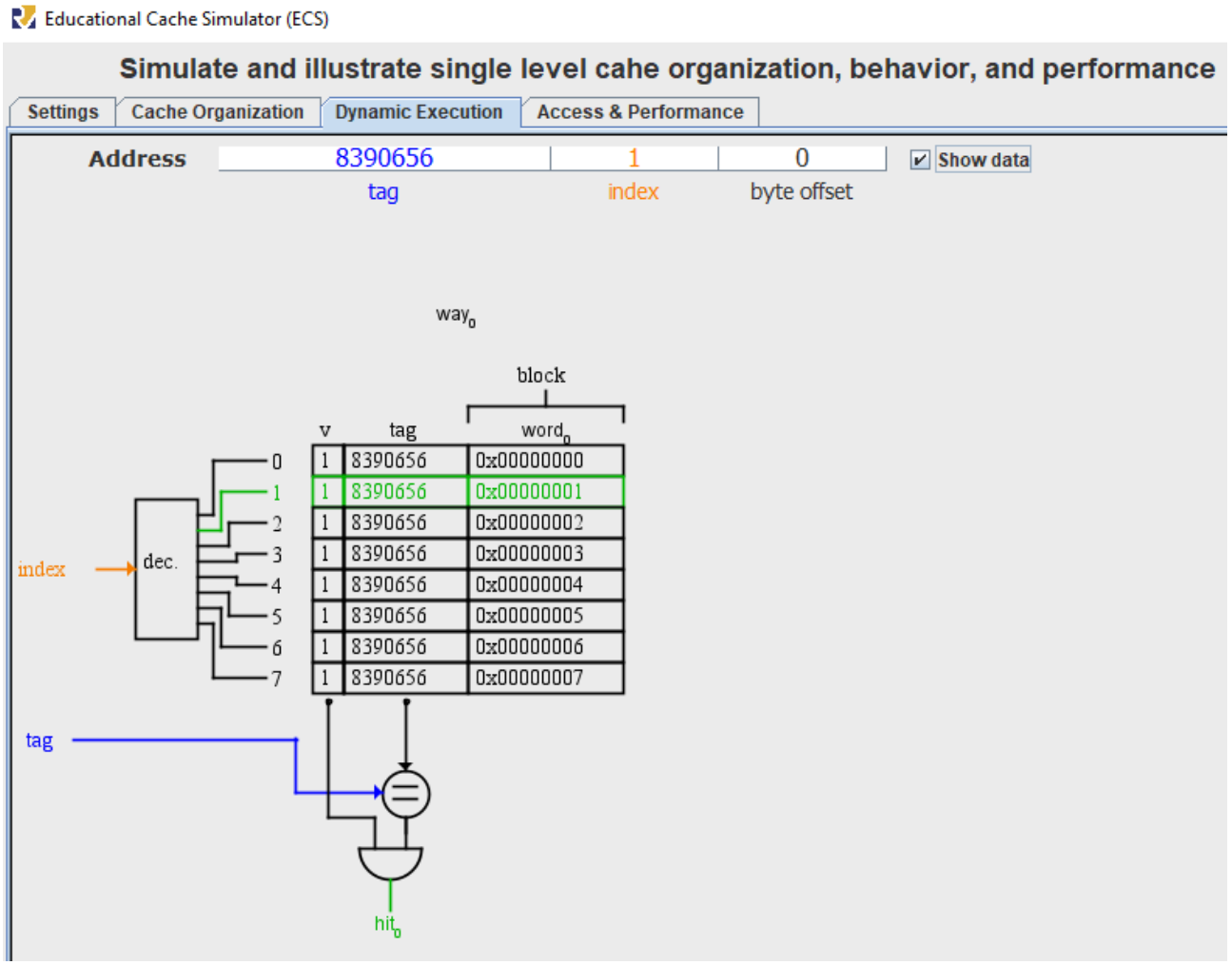

Figura 3. Aba *Dynamic Execution* do ECS.

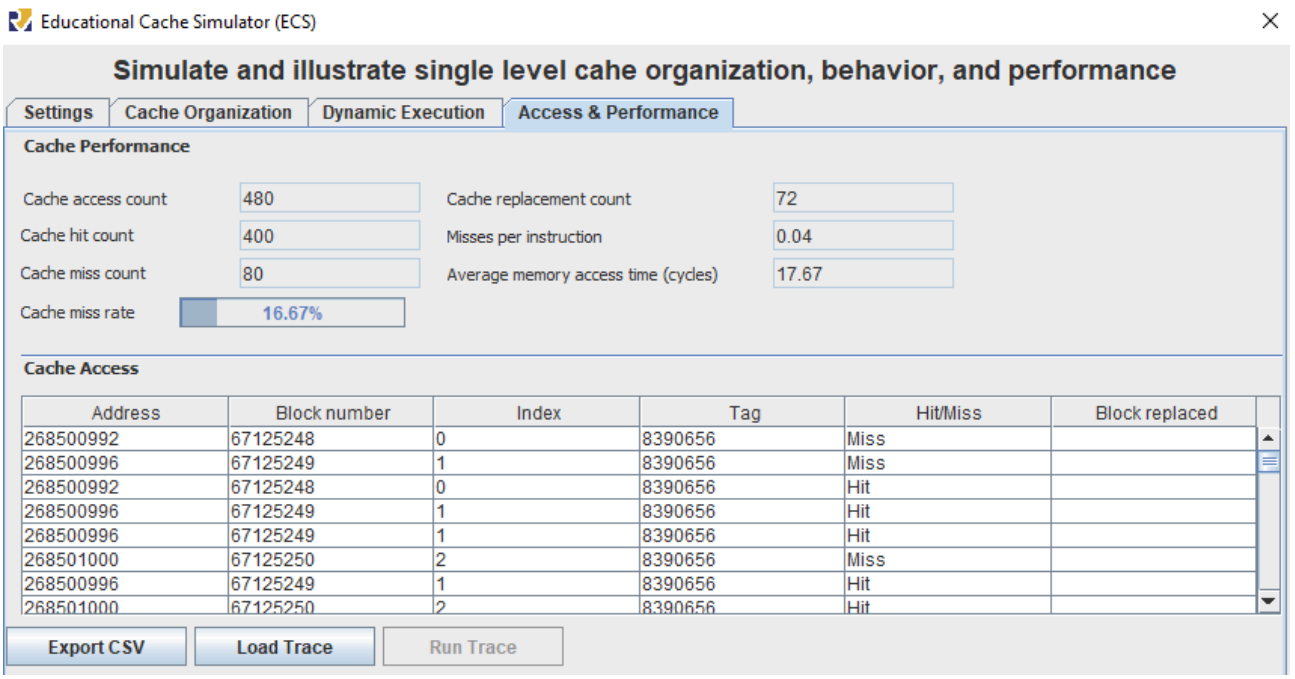

Figura 4. Aba *Access & Performance do ECS.*

A carga e execução de um arquivo de *trace* de endereços de memória permite a utilização do ECS na forma de um simulador *memory trace-driven*. Isso possibilita a simulação de *traces* longos e gerados com outras ferramentas, não ficando restrito à execução de programas RISC-V. O *trace* é fornecido em um arquivo texto, com um acesso por linha, contendo o tipo da operação (0, 1 ou 2, para leitura de instrução, leitura de dado e escrita de dado, respectivamente) e o endereço acessado, como exemplificado na Figura 5.

```
1  1856
1  2104
1  1920
1  2108
1  1984
2  0
```

Figura 5. Exemplo de um arquivo de *trace* de endereços de memória.

Uma maneira alternativa de utilizar a ferramenta ECS é através da sua execução *stand-alone*, com o RARS sendo executado apenas em *background* e sem usar o seu ambiente de desenvolvimento. Ela se dá através do terminal, usando o comando:

java -cp rars_1.6_ECS.jar rars.tools.EducationalCacheSimulator

diretamente na pasta em que o programa executável *rars_1.6_ECS.jar* se encontra. A Figura 6 demonstra como o ECS se apresenta com a execução *stand-alone* através do terminal.

O simulador ECS é uma ferramenta valiosa no ensino de Arquitetura e Organização de Computadores, pois oferece de uma maneira visual e interativa de entender o funcionamento das caches durante a execução de programas RISC-V. A possibilidade de configurar diferentes tipos de caches e observar seu impacto no desempenho do sistema é um diferencial importante para o aprendizado. Por meio da simulação e da análise dos resultados fornecidos pelo ECS, os alunos podem desenvolver uma compreensão mais aprofundada dos mecanismos de cache e sua importância no desempenho geral do computador.

3. Migração do ECS para o Simulador RARS

Ambas as ferramentas, o simulador MARS e o simulador RARS, possuem o código aberto e foram desenvolvidas em java. Dessa forma, o ECS também foi desenvolvido em java. O processo de migração do ECS do MARS para o RARS exigiu alterações em diversos pontos do projeto. As mudanças que foram necessárias durante o desenvolvimento são:

- Migração de pastas e mudanças de nomes;
- Separação do ECS e do EMCS;
- Alterações de classes do MARS para o RARS;
- Mudanças para os cálculos das métricas de desempenho.

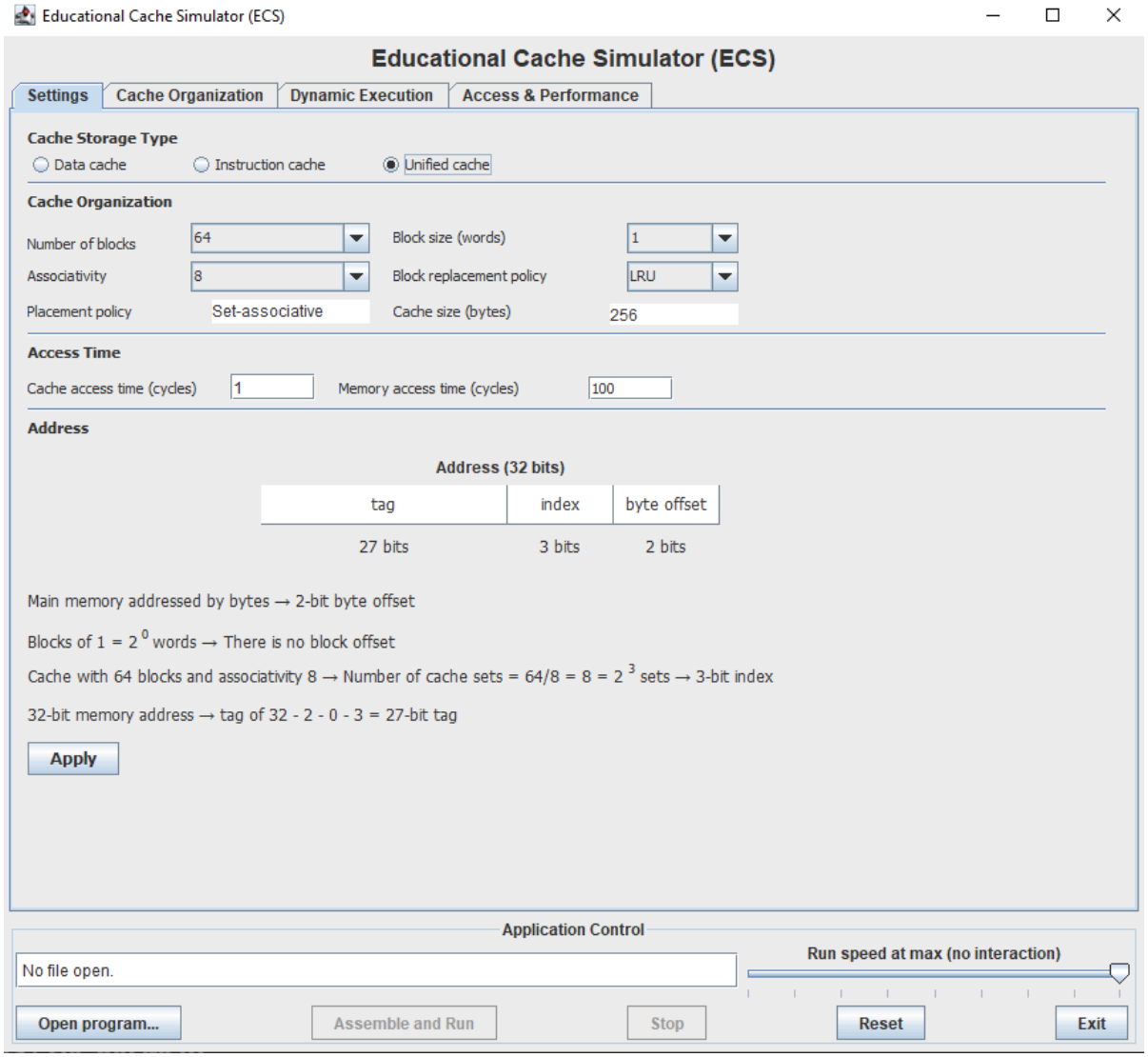

Figura 6. Aba *Settings* do ECS, na execução *stand-alone.*

A migração iniciou inserindo-se todos os arquivos que compõe a ferramenta ECS na estrutura de arquivos do RARS, sendo que o arquivo *SimpleCacheSimulator*, contendo a principal classe do ECS, foi para a pasta *tools* e a pasta *cache* foi renomeada para *ECS* e ficou dentro da pasta *rars*. Em seguida, foi necessária a mudança dos pacotes do MARS para os da ferramenta RARS, em todas as classes que fazem parte do ECS. Foi necessário corrigir alguns pacotes que mudaram de nome entre os simuladores, como por exemplo a classe *AbstractMarsToolAndApplication* que teve seu nome alterado para *AbstractToolAndApplication*. A Figura 7 demonstra a estrutura de pastas e arquivos do ECS ainda acoplado ao MARS, enquanto a Figura 8 apresenta a estrutura correspondente do ECS acoplado ao RARS. Toda ferramenta que é adicionada ao RARS precisa implementar a interface *rars.tools.Tool* e ser colocada na pasta *tools* do RARS.

Originalmente, além do ECS, outra ferramenta, o EMCS (*Educational Multilevel Caches Simulator*) [7], foi desenvolvida e acoplada ao MARS. Naquela versão, o ECS e o EMCS compartilhavam alguns módulos. Na migração para o RARS, optou-se por separar as duas ferramentas, dado que a especialização do código permitiria sua melhor estruturação. A separação entre o ECS e o EMCS tornou necessário o transporte de algumas classes, original e irregularmente associadas ao EMCS, para as pastas do projeto novo, em especial as classes que envolviam o tratamento dos arquivos *trace*.

Assim, classes como *RunTraceFile* e *TraceFile* foram criadas nas pastas do ECS e utilizadas na migração.

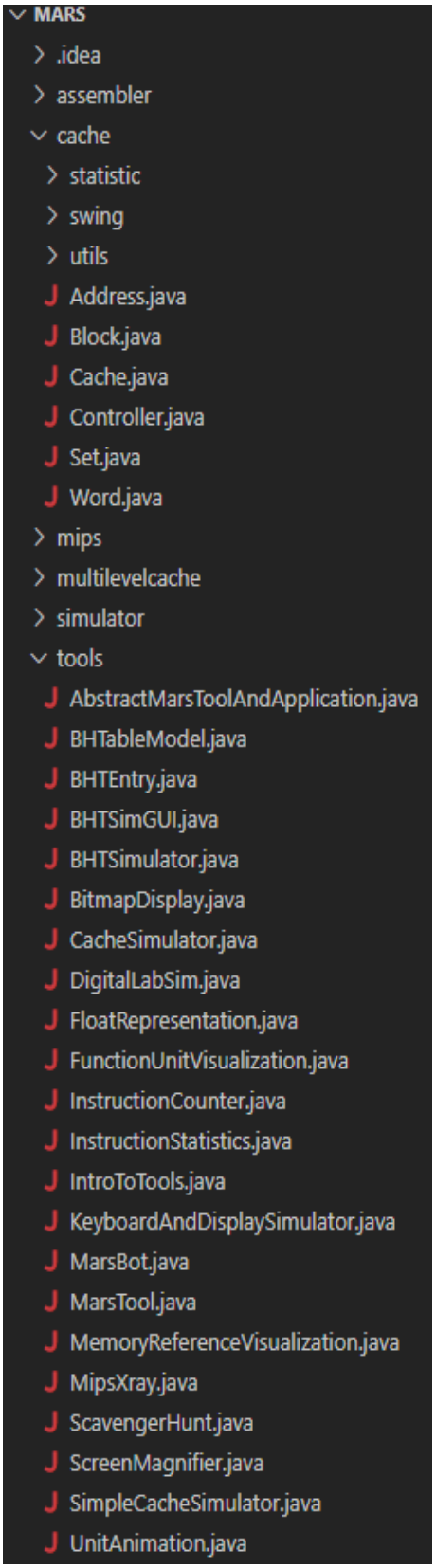

Figura 7. Estrutura do MARS com o ECS.

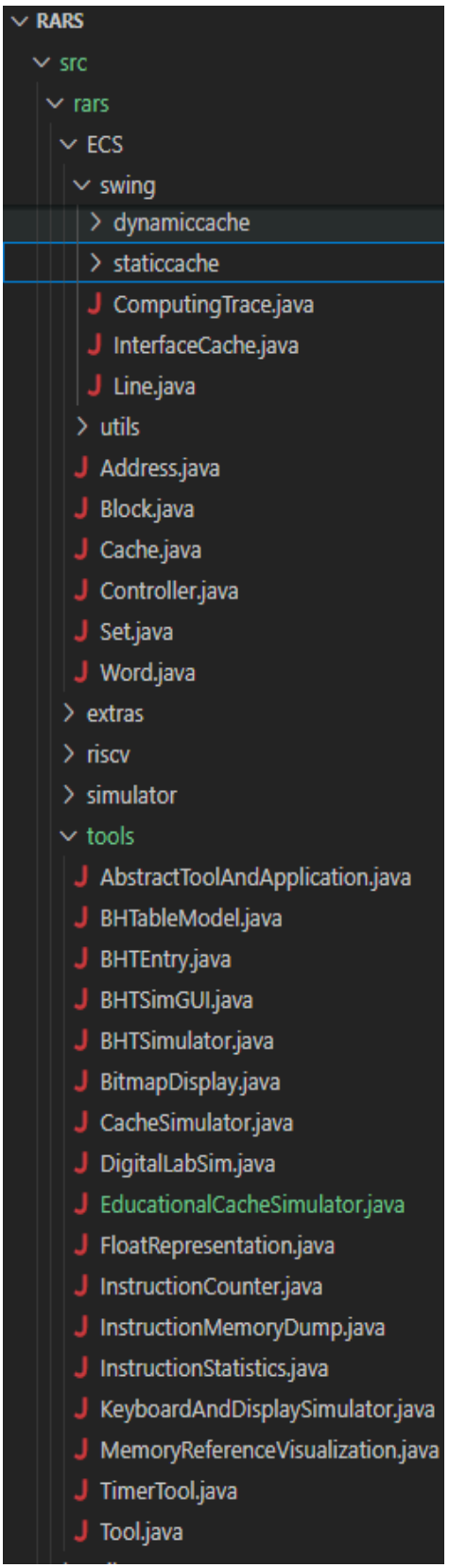

Figura 8. Estrutura do RARS com o ECS.

Todas as classes que eram relacionadas com o acesso de memória e hardware do MARS também foram alteradas na migração, assim tudo o que derivou do caminho *mars.mips* foi movido para *rars.riscv*. Essas mudanças afetaram diretamente partes do código em que eram usados métodos do MARS. Por exemplo, o método *InTextKernelTextSegment*, que descobria se o endereço passado como parâmetro estava no segmento de código, foi trocado pelo *inTextSegment*, que tem a mesma função mas está disponível na classe *rars.riscv.hardware.Memory*, assim como o comando *RunStopAction* que foi trocado para *RunStepAction* devido à mesma situação.

Por fim, para que as métricas de desempenho fossem calculadas corretamente foi necessário mudar a maneira que elas eram computadas, nos métodos *updateMetrics* e *updateMissInstruction* dentro da classe *Statistic*. Todos os valores tinham *casts* para o tipo *double*, o que não é aceito na estrutura do RARS, então foi necessária a criação de variáveis do tipo *double* que receberam os resultados dos cálculos todos feitos entre dados do tipo *double*.

Todas essas mudanças descritas até aqui foram necessárias para que o ECS funcionasse corretamente no RARS, da mesma que funcionava quando acoplado com o MARS. As modificações discutidas na próxima seção envolveram melhorias e ajustes no código do ECS, para o seu melhor funcionamento.

4. Modificações no ECS

Nesta seção são detalhadas as melhorias no código do ECS para que o seu uso se tornasse mais funcional e intuitivo. As mudanças realizadas no projeto foram:

- Arredondamento de métricas de desempenho;
- Remoção do valor da métrica falhas por instrução no uso de arquivos de *trace*;
- Padronização na pasta de busca de arquivos de *trace*;
- Ajustes nos avisos de execução de um arquivo de *trace*;
- Renomeação de pastas, arquivos e classes;
- Remoção da classe *TipoCache*;
- Remoção de código morto;
- Documentação com o doxygen.

O cálculo das métricas de desempenho que representam valores reais foi modificado para ter o arredondamento com precisão de duas casas decimais. Isso foi feito dentro dos métodos *updateMetrics* e *updateMissInstruction* dentro da classe *Statistic,* utilizando o método *Math.round* e também garantindo que a formatação da string apresentada em conformidade. Por exemplo, o método *updateMissInstruction* é mostrado na Figura 9. Outro processo de formatação foi a troca de todas as vírgulas decimais por pontos decimais, garantindo que o simulador estivesse de acordo com o resto da ferramenta RARS, que usa a língua inglesa.

```
public void updateMissInstrucion(int instruction)
{
    if(this.isCalculateMissesInstruction && instruction != 0) {
        double misscnt = missCount;
        double instr = instruction;
        double missperinstruction = misscnt/instr;
                String stringMisses = String.format(Locale.US, format:"%.2f", Math.round(missperinstruction*100.0)/100.0);
        txtMissesperInstruction.setText(stringMisses);
    }
}
```

Figura 9. Arredondamento das medidas de desempenho no método updateMissInstruction.

Quando um arquivo de *trace* é usado na simulação no ECS, a métrica de falhas por instrução não pode ser calculada, pois o número de instruções executadas não é conhecido. Entretanto, o ECS original algumas vezes apresentava um valor para essa métrica. Isso foi corrigido removendo da classe *RunTraceFile* a invocação do método *updateMissInstruction* da classe *Statistic*. Assim, no uso de um arquivo de *trace*, o campo que apresenta o valor da métrica fica sempre vazio.

A forma de busca de um arquivo de *trace* foi modificada, de forma a ficar padronizada com a forma como o RARS busca um programa fonte RISC-V para abertura. Para isso a classe *TraceFile* foi alterada, de forma que na primeira vez que o estudante for buscar um *trace* ele é levado à pasta no qual o programa principal do RARS se encontra. Após isso, o nome da pasta na qual o arquivo foi encontrado é armazenado em uma variável chamada *lastDirectory* e, a partir daí, nas próximas buscas de *trace*, o estudante será levado automaticamente à pasta do último *trace* executado.

No ECS original, não era muito intuitivo para o estudante solicitar a carga e simulação de um arquivo de *trace*. Para melhorar isso, a mensagem na janela de *pop-up* que informa a carga do *trace* foi modificada e foi incluída uma nova janela de *pop-up*, sinalizando a execução do *trace* e fechando automaticamente ao fim da simulação do mesmo. Desta maneira o estudante tem certeza de que a execução foi completada com sucesso. Assim, uma nova classe *ComputingTrace* foi desenvolvida, nela é criado um *JPanel* informando ao estudante para esperar o fim da execução do *trace*. No método *run* da classe *RunTraceFile*, após o *trace* terminar, o método *dispose* é invocado para fechar automaticamente a janela.

Diversas pastas, arquivos e classes foram renomeados para que ficasse mais claro para o estudante a função e o uso da ferramenta. Por exemplo, o nome da ferramenta no menu *Tools* do RARS foi mudado para *Educational Cache Simulator*. A ordem em que a ferramenta aparece nesse menu foi modificada para seguir a ordem alfabética padrão usada pelo RARS. Para isso, foi necessário remover o método *getOrderTool* da classe principal do ECS e atualizar a pasta *build* usada na execução dos *scripts* de compilação do RARS. A pasta *cache* com os demais arquivos do ECS foi renomeada para ECS, pois assim um estudante que observar o código saberá que ela pertence à estrutura dessa ferramenta.

A classe *TipoCache* que era importada em classes relacionadas com o uso de arquivos de *trace* foi removida, pois ela apenas criava variáveis que terminaram sendo implementadas diretamente nas classes em que eram utilizadas, assim seu uso se tornou desnecessário. Também foi feita a remoção de código morto de todos os arquivos do ECS, como variáveis criadas e calculadas mas não utilizadas e em algumas situações até métodos completos que não eram invocados, além de trechos de código em comentários. Um exemplo deste caso foi o método *createActionObjects*, que existia nas classes *Statistic* e *DynamicCache*, apesar de não ser utilizado.

Após todas as modificações, uma documentação de classes, variáveis e suas relações foi preparada utilizando o Doxygen [2], uma ferramenta *open source* que gera todos os documentos necessários, em um formato html, de maneira automatizada.

Essas modificações foram realizadas para que o uso do ECS se tornasse mais preciso e funcional para o estudante, e também para tornar o código do ECS mais limpo e mais padronizado com outras ferramentas que já existem dentro do RARS, facilitando futuras alterações por outros desenvolvedores.

5. Experimento de Simulação de Programas RISC-V

Como o ECS tem como objetivo principal o uso educacional, nesta seção é ilustrado um experimento com o uso da ferramenta durante a execução de programas RISC-V, para que o estudante compreenda melhor os conceitos sobre a organização e o funcionamento de uma memória cache.

Os programas RISC-V utilizados nas simulações são:

- soma de dois vetores;
- soma dos elementos de índice par de um vetor;
- ordenação bubblesort.

Cada programa se comporta de maneira diferente no uso das caches. Assim, o estudante pode observar diferentes situações que ocorrem no uso da cache durante a execução dos programa. Na Tabela 1 são apresentadas quatro configurações diferentes de caches que são simuladas neste experimento.

Caches de dados	Cache A	Cache B	Cache C	Cache D
Número total de blocos	32	32	32	32
Associatividade	1	1	4	4
Tamanho do bloco	1	2	1	4
Política de substituição	–	–	LRU	LRU

Tabela 1. Configurações utilizadas para os experimentos de simulações de programas RISC-V

Ao simular os programas de soma de dois vetores e soma dos elementos de índice par de um vetor, usando as caches A e B, o estudante pode observar que, para o primeiro programa, a cache A apresenta uma taxa de falhas bastante alta, enquanto a cache B reduz essa taxa, apresentando um desempenho melhor. No entanto, para o segundo programa, tal redução na taxa de falhas não é observada na cache B. O aluno é então é questionado a tirar conclusões sobre como o aumento do tamanho do bloco pode ter levado a taxas de falhas diferentes, sobre os programas apresentarem localidade espacial mais intensamente ou não, e como isso afeta a performance das caches.

Em seguida, simulando o programa bubblesort com as caches A e C, o aluno observa que a primeira apresenta uma taxa de falhas mais alta que a segunda, mesmo com ambas as caches possuindo o mesmo número total de blocos e diferenciando-se apenas na associatividade. O aluno pode analisar o comportamento do programa e a localidade temporal apresentada por ele, e como o aumento da associatividade leva à redução da taxa de falhas.

Também é interessante que o aluno observe, na ilustração da organização da cache fornecida na aba *Cache Organization* do ECS, como os circuitos auxiliares de acesso à cache mudam entre essas quatro configurações de cache, com a utilização ou não de multiplexadores. O aluno deve compreender qual é a função de cada multiplexador: selecionar a palavra do bloco acessada pelo processador (como nas caches B e D) ou selecionar a via em que ocorreu acerto (como nas caches C e D).

As conclusões extraídas desse experimento ajudam o estudante a compreender melhor a organização das memórias cache com diferentes configurações, como essa configuração determina a interpretação dos bits do endereço de memória e os campos extraídos desse endereço são usados no acesso à cache. Também permitem que o aluno identifique situações em que a cache explora a localidade temporal e/ou espacial apresentadas pelo programa para reduzir a taxa de falhas e assim melhorar o desempenho do programa.

6. Experimento de Simulação de *Trace* de Endereços

Outra maneira de utilizar o ECS é através de um arquivo de *trace* de endereços de memória. Esse arquivo contém a sequência de endereços de memória acessados durante a execução de um programa. Dessa forma, não é necessário realizar a simulação do programa, bastando simular os acessos realizados à cache a partir desses endereços. A utilização do ECS como simulador *memory trace-driven* permite simular caches com *traces* de endereços de memória gerados por outras ferramentas. Também possibilita a realização de experimentos com simulação de caches com um número muito grande de acessos.

Neste experimento é utilizado um *trace* de memória, usado em [7], contendo 55.986.879 acessos (a instruções e dados), correspondentes à execução do programa Rijndael encoder do *benchmark* MiBench. O aluno pode fazer simulações de maneira a considerar três cenários diferentes para a hierarquia de memórias:

a. Com uma única cache unificada, com instruções e dados;
b. Com duas caches, uma apenas com instruções e outra apenas com dados (caches *split*);
c. Sem memórias cache (ou seja, todo o acesso é feito diretamente à memória principal).

O aluno é instigado a analisar e comparar o desempenho obtido em cada um desses cenários. Usando as medidas de desempenho obtidas pelo ECS, o aluno pode calcular a taxa de falhas combinada das caches *split* do cenário (b) e compará-la com a taxa de falhas da cache unificada do cenário (a). Também pode calcular o tempo médio de acesso à memória (*Average Memory Access Time* – AMAT) dos cenários (b) e (c), usando a taxa de falhas combinada (para o cenário (b)) e os tempos de acessos definidos. Assim, o aluno pode comparar esses valores com o AMAT calculado pelo ECS para o cenário (a) e analisar o impacto, no desempenho da hierarquia de memórias, do uso de caches e da sua organização como cache unificada ou *split*.

7. Conclusão

Devido à adoção do processador RISC-V, em substituição ao processador MIPS, como processador exemplo usado no ensino de Arquitetura e Organização de Computadores em muitas instituições, diversas ferramentas de uso educacional, como montadores e simuladores, surgiram baseadas nesse novo processador. Assim, usando como base a ferramenta MARS, um montador e simulador do conjunto de instruções MIPS, a ferramenta RARS foi desenvolvida, como um montador e simulador do conjunto de instruções RISC-V. Em consequência, surgiu a necessidade de migrar para o RARS o ECS, um simulador de memórias cache de uso educacional desenvolvido originalmente como uma ferramenta acoplada ao MARS.

Para essa migração foram realizadas mudanças estruturais e otimizações no código do ECS, envolvendo diversos aspectos, desde a adaptação da ferramenta à arquitetura RISC-V até melhorias na usabilidade da mesma.

Como resultado deste trabalho, a ferramenta ECS é disponibilizada um recurso pedagógico adicional no ensino dos conteúdos relacionados a hierarquia de memórias e memórias cache nas disciplinas da área de Arquitetura e Organização de Computadores. O simulador pode ser usado pelos alunos em aulas práticas em laboratório ou no estudo extra-classe, através da realização de diversos experimentos. A ferramenta apresenta os seguintes recursos:

- Permite a configuração da cache a ser simulada: tipo (cache de dados, instruções ou unificada), número de blocos, associatividade, tamanho do bloco e política de substituição de blocos;
- Mostra como o endereço de memória acessado é dividido em campos e como eles são utilizados no acesso à cache;
- Apresenta uma representação gráfica visual detalhada da organização da cache e permite a exportação dessa imagem;
- Simula, durante a execução de um programa RISC-V no RARS, os acessos à cache relativos aos acessos realizados, pelo programa, a instruções e/ou dados na memória;
- Permite o acompanhamento e visualização de cada acesso realizado à cache durante a execução do programa RISC-V;
- Mostra o endereço acessado, o conteúdo da cache, o conjunto acessado, se ocorreu falha ou acerto e a via onde ocorreu o acerto;
- Apresenta informações de cada acesso realizado: o endereço acessado, o número do bloco de memória acessado, o *block offset*, a *tag* e o índice correspondentes, se ocorreu acerto ou falha e o número do bloco substituído; também permite a exportação dessas informações na forma de uma planilha;
- Simula os acessos à cache relativos aos acessos, a instruções e/ou dados na memória, oriundos de um *trace* de endereços de memória;
- Apresenta resultados de desempenho: número de acessos à cache, número de acertos e falhas, taxa de falhas, número de substituições de blocos realizadas, falhas por instrução e tempo médio de acesso à memória.

Como trabalhos futuros, novas melhorias podem ser desenvolvidas, como por exemplo:

- Integração do ECS com a ferramenta EMCS (*Educational Multilevel Cache Simulator*);
- Melhoria no desempenho do ECS, na atualização do conteúdo da cache a cada acesso realizado, na aba *Dynamic execution*;
- Possibilidade de realizar, em uma única execução, a simulação de caches de instruções e dados separadas (*split*); atualmente é necessário realizar duas execuções, uma para cada tipo de cache;
- Inclusão de uma maneira de exportar os resultados das métricas de performance para uma planilha.

8. Bibliografia

1. Brais, H., Kalayappan, R. e Panda, P. R. A Survey of Cache Simulators. ACM Computing Surveys, 53(1):19:1-19:32, 2020.
2. Heesch, D. V. Doxygen release 1.12.0, https://github.com/doxygen/doxygen, acessado em outubro de 2024.
3. Hennessy, J. L. e Patterson, D. A. Computer Architecture: A Quantitative Approach. Elsevier, 6a. edição, 2019.
4. Koci, K., Pisa, P., Dupak, J. e Hollmann, M. QtRvSim – RISC-V Architecture Simulator, https://comparch.edu.cvut.cz/qtrvsim/app/, acessado em outubro de 2024.
5. Koci, K., Pisa, P., Dupak, J. e Hollmann, M. QtRvSim – RISC-V CPU simulator for education, https://github.com/cvut/qtrvsim, acessado em outubro de 2024.
6. Landers, B. RARS – RISC-V Assembler and Runtime Simulator, https://github.com/TheThirdOne/rars, acessado em outubro de 2024.
7. Lima, D. P. e Moreano, N. ECS e EMCS: Simuladores de Caches para o Apoio Pedagógico no Ensino de Arquitetura de Computadores, Anais do XXIX Workshop sobre Educação em Computação, 2021.
8. Patterson, D. A. e Hennessy, J. L. Computer Organization and Design – The Hardware/Software Interface – MIPS Edition. Elsevier, 6a. edição, 2020.
9. Patterson, D. A. e Hennessy, J. L. Computer Organization and Design – The Hardware/Software Interface – RISC-V Edition. Elsevier, 2a. edição, 2021.
10. Petersen, M. B. Ripes, https://github.com/mortbopet/Ripes, acessado em outubro de 2024.
11. Petersen, M. B. Ripes: A Visual Computer Architecture Simulator, Anais do ACM/IEEE Workshop on Computer Architecture Education (WCAE), pg. 1-8, 2021.
12. RISC-V International. RISC-V, https://riscv.org/, acessado em outubro de 2024.
13. Sanderson, P. e Vollmar, K. MARS – MIPS Assembler and Runtime Simulator, https://github.com/dpetersanderson/MARS/, acessado em outubro de 2024.