# Abordagem para Aprendizado do Simulador gem5 para Pesquisadores Iniciantes

**Pedro Corrêa Rigotto, Henrique Cota de Freitas**

*Computer Architecture and Parallel Processing Team (CArT)*
Departamento de Ciência da Computação – Pontifícia Universidade Católica de Minas Gerais
Belo Horizonte – MG – Brasil

pcrigotto@sga.pucminas.br, cota@pucminas.br

***Abstract.*** *The gem5 simulator is a tool with many features for computer architecture research. It supports full-system simulations, the market's most used Instruction Set Architectures (ISAs), and many system configurations. Researchers mostly use it, but it could also be used to teach computer architecture to students. In this article, we propose a simple approach for beginners to get started with the simulator in a quick and easy-to-understand manner.*

***Resumo.*** *O simulador gem5 é uma ferramenta com muitos recursos para desenvolvimento de pesquisa em arquitetura de computadores. Ele suporta simulações de sistemas completos, as* Instruction Set Architectures *(ISAs) mais utilizadas no mercado e diversos tipos de configurações de sistema. É principalmente utilizado por pesquisadores, mas também pode ser utilizado para ensinar arquitetura de computadores a estudantes. Neste artigo, uma abordagem simples é proposta para que iniciantes comecem a utilizar o simulador de maneira rápida e fácil de entender.*

## 1. Introdução

A rápida evolução dos computadores demanda alternativas de suporte ao projeto e desenvolvimento. Desde as arquiteturas de computadores valvulados, transistorizados até os computadores quânticos, pesquisadores fazem uso de modelos analíticos e simulações em fases iniciais de projeto e teste. O estudo em cursos de graduação voltado para arquitetura de computadores concentra-se em pipelines escalares e superescalares, o que de certo modo, corresponde ao que mais está presente na índústria de processadores. Suportes a multiplas threads e o desenvolvimento de múltiplos núcleos aumenta de maneira considerável a complexidade de estudo, pesquisa e desenvolvimento de arquiteturas sob um ponto de vista, principalmente, do estudante iniciante em pesquisa. Com um leque de opções de arquteturas muito extenso, é necessário criar condições para que uma iniciação científica supere rapidamente uma barreira de aprendizado para que resultados não demorem a surgir.

Portanto, a pesquisa em arquitetura de computadores necessita de simuladores altamente precisos, para que inovações possam ser testadas de maneira rápida e confiável. Como dito por Akram and Sawalha [2019], a escolha de qual simulador utilizar depende muito do propósito da pesquisa, porém existe uma opção que atende diversas demandas

Pedro Rigotto é bolsista de iniciação científica FIP PUC Minas.

diferentes. Esse simulador é o gem5, que é uma aplicação que pode simular diversas arquiteturas e configurações [Binkert et al., 2011], e que será o simulador a ser utilizado para os propósitos deste artigo.

O simulador gem5 habilita a simulação de sistemas completos ou básicos, e isso nos permite utilizá-lo com a granularidade de precisão desejada. Para este artigo, não será utilizada a simulação completa (*full-system*), pois ela não é relevante para o aprendizado de iniciantes ao simulador, apesar de ser muito utilizada em pesquisas reais [Carmo et al., 2016; Penna and Freitas, 2013; Vasconcelos et al., 2014; Mishra, 2019].

O problema abordado por este artigo é que apesar de suportar várias configurações, o gem5 apresenta uma barreira para iniciantes, por ser tão complexo. A documentação existente falta atualizações necessárias e informações importantes. Isso significa que o iniciante deve abrir arquivos presentes no código fonte do simulador para descobrir a maneira correta de se utilizar alguma função. Além disso, o passo-a-passo disponível no site do gem5 para iniciantes[1], além de ser muito complexo, é muito extenso. Ele deveria ser utilizado apenas para referência após o usuário já ter obtido conhecimento prévio, para que o aprendizado inicial seja rápido, e específicos possam ser estudados quando necessário.

O objetivo deste artigo é apresentar uma maneira mais didática e fácil de se começar a estudar o simulador, para que pesquisadores iniciando a sua pesquisa com essa ferramenta possam acelerar o seu processo de ambientação, e para que seja possível ensinar arquitetura de computadores a alunos de forma simples, sem que eles gastem tempo precioso para aprender funções que não serão utilizadas em um meio didático.

A abordagem para essa aprendizagem será utilizando scripts de configurações de exemplo já existentes no código fonte do simulador, em vez de que sejam feitos novos scripts para cada configuração a ser testada. Desse modo, pode ser aprendida a funcionalidade do simulador de maneira simples e rápida, sem entrar em detalhes de como construir seus próprios objetos ou scripts de configuração. Assim, poderá ser explorada a funcionalidade do gem5 de executar diferentes arquiteturas, tipos de processador, e configurações do computador simulado, além de permitir a comparação de resultados entre essas simulações, utilizando o próprio sistema de estatísticas do simulador.

O restante deste artigo será apresentado da seguinte forma: Na Seção 2 são discutidos artigos relacionados que exploram simuladores de sistemas computacionais, e onde este artigo se difere da literatura existente. As simulações executadas e os passos necessários para repetí-las são abordados na Seção 3. A Seção 4 apresenta a avaliação dos resultados encontrados e o conhecimento que foi possível ser extraído das estatísticas encontradas a partir das simulações feitas. São expostas as conclusões encontradas na Seção 5, e também o que pode ser explorado em trabalhos futuros.

## 2. Trabalhos Correlatos

Simuladores de sistemas computacionais são muito importantes para a pesquisa e o aprendizado. Eles proporcionam *insights* que seriam inobtíveis de outras maneiras. O mercado está repleto desses simuladores, cada um com os seus recursos e limitações.

[1] https://www.gem5.org/documentation/learning_gem5/introduction/

Nikolic et al. [2009] fizeram uma pesquisa onde foram comparados diversos simuladores de sistemas de computadores em 2009. Os resultados encontrados mostram que nenhum simulador existente na época conseguia abordar todos os tópicos de um curso de arquitetura de computadores. Porém, entre os que tiveram melhores resultados, se encontra o simulador M5, que foi um dos dois simuladores que serviram de base para o desenvolvimento do gem5 [Binkert et al., 2011]. Isso indica que o gem5, por ter mais capacidades que o M5, é mais adequado para o ensino da arquitetura de computadores que os outros simuladores explorados pelos autores daquele artigo.

Simuladores de sistemas de computadores são usados para o ensino de arquitetura e organização de computadores [Ristov et al., 2013; Garcia et al., 2009; Soares et al., 2016; Radivojevic et al., 2018; Penna and Freitas, 2013], e existem benefícios do seu uso [Prasad et al., 2016]. Muitos desses simuladores sendo utilizados são visuais e simplificados. O gem5, apesar de não possuir interface gráfica, apresenta simulações de sistemas de maneira precisa. Ele consegue simular sistemas completos, incluindo sistema operacional, e obter estatísticas de desempenho que refletem o comportamento de sistemas reais. Estatísticas de execução de programas em sistemas físicos não são fáceis de se obter, e, ao utilizar o gem5, os estudantes podem ter contato com esses números que normalmente estariam fora do alcance.

Apesar de haver literatura disponível que explora simuladores, e o gem5 ser um dos mais utilizados na indústria [Lowe-Power et al., 2020], não foi encontrado nenhum artigo que tenha como objetivo facilitar a iniciação científica e sua exploração. Portanto, este artigo propõe uma abordagem que seja mais acessível e rápida para a ambientação no simulador. Assim, estudantes interessados em arquitetura de computadores e pesquisadores iniciantes na área poderão explorar essa ferramenta de maneira fácil e compreensível.

## 3. Método

Para os testes realizados, foram utilizadas três arquiteturas e três tipos de processador diferentes. As arquiteturas escolhidas foram as mais presentes no mercado, sendo elas ARM, x86 e RISC-V. Entre os tipos de processador disponíveis, foram testados o *TimingSimpleCPU*, *AtomicSimpleCPU* e *O3CPU*.

As CPUs *TimingSimpleCPU* e *AtomicSimpleCPU* são tipos de CPU simples. Isso quer dizer que elas não são muito detalhadas durante a sua execução, portanto, elas são utilizadas em momentos onde não é necessário um modelo mais detalhado. Elas executam as instruções em apenas um ciclo, exceto instruções de acesso à memória, portanto não há estatísticas de CPI para este tipo de CPU.

A *TimingSimpleCPU* é a mais detalhada das CPUs simples, e seu propósito é executar as instruções no tempo mais realístico, para maior precisão nas estatísticas. Já a *AtomicSimpleCPU* é uma simulação bem mais rápida, com estimações de tempo para requisições para acelerar a simulação. Portanto, a *AtomicSimpleCPU* é usada para *fast-forwarding* até um checkpoint pré-definido, ou para "aquecer" as caches, em vez de ser o carro-chefe da simulação.

A outra CPU utilizada, *O3CPU*, é um modelo bem mais detalhado de um processador. Ela permite a execução fora de ordem das instruções, e tenta simular o tempo das

requisições de forma mais precisa possível. Ela é utilizada quando a simulação demanda precisão em aspectos como os estágios do *pipeline*.

Os dados coletados foram o tempo total simulado, o número de *hits* e o número de *misses* na cache, a taxa de *misses*, e o CPI total da simulação. O sistema escolhido utiliza os valores padrão para todos os aspectos do processador, sendo eles número de CPUs igual a um, uma *thread*, tamanho da linha da cache igual a 64 bytes, associatividade da cache igual a 2, dois níveis de cache, *dcache* de 64kB, *icache* de 32kB, DRAM de 512MB. Todos esses valores podem ser alterados na configuração do simulador.

O arquivo de configuração utilizado foi o "se.py", disponível na pasta de configurações de exemplo presentes no código fonte do simulador. Ele utiliza o modo *Syscall Emulation* (SE), que não simula um sistema completo, portanto não há uso de um *kernel* de sistema operacional. O simulador foi desenvolvido para Linux, portanto, o sistema operacional utilizado para os testes foi o Ubuntu 22.04.2 LTS.

### 3.1. Passo-a-passo Para Repetir as Simulações

O primeiro passo para executar estas simulações é instalar o gem5. Os desenvolvedores disponibilizam um guia para isso[2], porém o comando para instalação dos pré-requisitos presente no site em Agosto de 2023 está desatualizado. Há uma necessidade de se alterar o comando de instalação do Python de `python` e `python-dev` para `python3` e `python3-dev`, respectivamente. O comando final é:

```
sudo apt install build-essential git m4 scons zlib1g
zlib1g-dev libprotobuf-dev protobuf-compiler libprotoc-dev
libgoogle-perftools-dev python3-dev python3
```

Após a instalação dos pré-requisitos é necessário obter o código do simulador, o que pode ser feito através do comando `git clone https://github.com/gem5/gem5`, executado na pasta escolhida para o simulador. Para compilar o gem5 com a arquitetura desejada, são usados os comandos seguintes na pasta onde o simulador foi instalado:

```
scons build/ARM/gem5.opt
scons build/X86/gem5.opt
scons build/RISCV/gem5.opt
```

É importante notar que o nome da arquitetura escolhida deve ser escrito com letras maiúsculas. O simulador tabém disponibiliza outras *Instruction Set Architectures* (ISAs), porém para este artigo foram utilizadas apenas aquelas presentes acima.

O programa sendo executado pelo sistema simulado é o arquivo "hello", presente no código fonte do gem5 na pasta "`gem5/tests/test-progs/hello/bin/<arquitetura escolhida>/linux`". O caminho difere para cada teste, portanto devemos alterar o texto `<arquitetura escolhida>` pelo nome da arquitetura instalada, utilizando letras minúsculas. Esse arquivo apenas imprime "*Hello world!*" na tela. Apesar de ser simples, é o suficiente para os propósitos deste artigo.

[2]https://www.gem5.org/documentation/learning_gem5/part1/building/

Após a compilação do simulador com as arquiteturas desejadas é possível realizar os testes. Para isso, foi utilizado o seguinte comando:

```
build/ARM/gem5.opt configs/example/se.py -c
tests/test-progs/hello/bin/arm/linux/hello
--cpu-type=TimingSimpleCPU --caches
```

É necessário alterar o nome da arquitetura e o tipo de CPU para repetir os testes, mantendo caixa alta após `build/` e caixa baixa após `/bin/`. Um exemplo com outros tipos de arquitetura e CPU é o seguinte:

```
build/X86/gem5.opt configs/example/se.py -c
tests/test-progs/hello/bin/x86/linux/hello
--cpu-type=X86O3CPU --caches
```

Esse comando contém informações importantes para a configuração do simulador. Com ele definimos a arquitetura, o arquivo de configuração, o binário a ser executado, e o tipo de CPU desejados, além do fato do sistema conter caches com configuração padrão. É possível incluir vários outros argumentos nesse comando para definir outros parâmetros do sistema, e obter mais informações com *debug flags*, porém, essas opções são mais avançadas, e fogem do escopo deste artigo. Mais informações estão disponíveis no site do simulador[3].

Para analisar os resultados, algumas estatísticas são necessárias. Elas podem ser encontradas no arquivo `gem5/m5out/stats.txt`. Nos testes realizados, foram verificados os valores presentes nas linhas que começam com os seguintes nomes:

```
simSeconds
system.cpu.dcache.overallHits::total
system.cpu.dcache.overallMisses::total
system.cpu.dcache.overallMissRate::total
system.cpu.cpi
```

Caso haja necessidade de se verificar algo que não está presente na estatística padrão, é possível adicionar seus próprios valores ou *debug flags*[4].

## 4. Avaliação de Resultados

Com os dados presentes na Figura 1, podemos verificar algumas características de cada arquitetura e tipo de CPU. Como descrito na Seção 3, a *AtomicSimpleCPU* executa cerca de 2,6 a 10 vezes mais rápido que as outras duas, devido a algumas aproximações e estimações que ela implementa para acelerar a sua execução. As duas CPUs *TimingSimpleCPU* e *AtomicSimpleCPU* apresentam exatamente a mesma quantidade de *hits* e *misses* na cache em todos os testes, por serem muito similares e implementadas a partir da mesma CPU base. É notável uma taxa de *misses* muito maior que as outras na *O3CPU*, exceto quando é utilizada a arquitetura x86. Também é visível que há uma diferença de *Cycles Per Instruction* (CPI) entre as ISAs, sendo a arquitetura RISC-V a mais eficiente

[3]https://www.gem5.org/documentation/learning_gem5/part1/example_configs/

[4]https://www.gem5.org/documentation/learning_gem5/part2/debugging/

**Figure 1. Resultados dos testes com três tipos de CPU e arquitetura**

| Tipo de CPU | Arquitetura | *Hits* | *Misses* | *Miss rate* (%) | Tempo (em µs) | CPI |
|---|---|---|---|---|---|---|
| Timing | ARM | 1833 | 142 | 7,18 | 29 | - |
| | x86 | 1890 | 135 | 6,66 | 31 | - |
| | RISC-V | 1935 | 238 | 6,52 | 30 | - |
| O3 | ARM | 2292 | 519 | 18,46 | 14 | 5.764 |
| | x86 | 2313 | 200 | 7,95 | 16 | 5.655 |
| | RISC-V | 2204 | 527 | 19,29 | 15 | 5.258 |
| Atomic | ARM | 1833 | 142 | 7,18 | 3 | - |
| | x86 | 1890 | 135 | 6,66 | 6 | - |
| | RISC-V | 1935 | 135 | 6,52 | 3 | - |

nesse quesito. Os resultados foram obtidos a partir de uma única execução de cada teste, já que a simulação é determinística no modo SE.

Os resultados indicam que há conhecimento a ser adquirido através dessas estatísticas, mesmo quando são testados apenas sistemas com componentes e ISAs já existentes e conhecidos. Isso pode ser utilizado por professores de arquitetura de computadores para ensinar aos seus estudantes conceitos como a importância das caches e a diferença de CPI entre arquiteturas ou programas. Pesquisadores com mais experiência testando alguma nova tecnologia podem acelerar sua introdução ao simulador realizando os testes aqui apresentados para se ambientar. Aquilo que pode levar dias para ser aprendido em uma primeira utilização, é possível de ser reduzido a uma simples execução de uma linha de comando pré-montada.

## 5. Conclusões

O simulador gem5 é uma ferramenta que promove diversos benefícios a estudantes e pesquisadores que o utilizam. Ele apresenta várias funcionalidades que englobam grande parte dos aspectos da arquitetura de computadores, porém, por possuir tantos recursos, ele se torna difícil de se ambientar e utilizar, para iniciantes. Este artigo propõe uma abordagem complementar e mais simplificada para o primeiro uso do simulador, com objetivo de diminuir a sua barreira de entrada. Em vez de seguir o passo-a-passo disponibilizado pelos desenvolvedores do simulador, foi proposto o uso de configurações pré-montadas, para que não seja necessário se aprofundar nos aspectos específicos do simulador em um primeiro uso.

Os testes realizados apontam que existe conhecimento a ser obtido mesmo sem a construção de arquivos de configuração customizados. O uso do simulador dessa maneira pode ser útil para o ensino de arquitetura e organização de computadores, além de proporcionar uma primeira experiência no gem5 com menos barreiras que o seu uso tradicional. Portanto, os objetivos propostos foram alcançados com sucesso. Porém, deve-se considerar que o simulador possui uma extensa documentação presente em seu site[5], que também contém o livro "*Learning gem5*", ambos fornecendo informações essenciais para a aprendizagem aprofundada de seu uso. Este artigo não deve ser a única fonte de informações para a ambientação no simulador, mas apenas um guia para o primeiro passo.

[5]https://www.gem5.org/documentation/

Trabalhos futuros poderão explorar aspectos mais complexos do simulador, para facilitar a utilização de recursos mais avançados que ele disponibiliza. Poderão ser exploradas a funcionalidade de simulação de sistemas completos, a criação de novos componentes a serem simulados, e a introdução de novas tecnologias. Também pode ser pesquisado o motivo de se encontrar mais *misses* ao utilizar a *O3CPU*, além da possibilidade de executar outros arquivos binários de algoritmos clássicos, explorando a diferença das ISAs e tipos de CPU, e os comportamentos dos algoritmos em cada configuração. Há a possibilidade de ser explorada a aplicação da metodologia aqui proposta para a educação em arquitetura de computadores, possibilitando a criação de novos scripts de configuração que exploram certos atributos do simulador, e a criação de exercícios que utilizam o gem5 e esses scripts para melhorar o ensino. Serão realizadas mais buscas por trabalhos correlatos, inclusive em simuladores diferentes, para maior detalhamento e comparação com a abordagem proposta neste artigo.

## Agradecimentos

Os autores agradecem à FAPEMIG, ao CNPq e a PUC Minas pelo suporte parcial na execução desta pesquisa.

## Referências

Ayaz Akram and Lina Sawalha. A survey of computer architecture simulation techniques and tools. *IEEE Access*, 7:78120–78145, 2019. doi: 10.1109/ACCESS.2019.2917698.

Nathan Binkert, Bradford Beckmann, Gabriel Black, Steven K. Reinhardt, and et al. The gem5 simulator. *SIGARCH Comput. Archit. News*, 39(2):1–7, aug 2011. ISSN 0163-5964. doi: 10.1145/2024716.2024718. URL `https://doi-org.ez93.periodicos.capes.gov.br/10.1145/2024716.2024718`.

Daniel Carmo, Matheus Souza, and Henrique Freitas. Avaliação de topologias de redes-em-chip usando simulação de sistemas completos e aplicações paralelas. In *Anais do XVII Simpósio em Sistemas Computacionais de Alto Desempenho*, pages 109–120, Porto Alegre, RS, Brasil, 2016. SBC. doi: 10.5753/wscad.2016.14252. URL `https://sol.sbc.org.br/index.php/wscad/article/view/14252`.

M. Isabel Garcia, Santiago Rodriguez, Antonio Perez, and Antonio Garcia. p88110: A graphical simulator for computer architecture and organization courses. *IEEE Transactions on Education*, 52(2):248–256, May 2009. ISSN 1557-9638. doi: 10.1109/TE.2008.927690.

Jason Lowe-Power, Abdul Mutaal Ahmad, Ayaz Akram, Mohammad Alian, and et al. The gem5 simulator: Version 20.0. *arXiv.org*, 2020. ISSN 2331-8422.

Debadatta Mishra. gemos: Bridging the gap between architecture and operating system in computer system education. In *Proceedings of the Workshop on computer architecture education*, WCAE'19, pages 1–8. ACM, 2019. ISBN 1450368425.

Bosko Nikolic, Zaharije Radivojevic, Jovan Djordjevic, and Veljko Milutinovic. A survey and evaluation of simulators suitable for teaching courses in computer architecture and organization. *IEEE Transactions on Education*, 52(4):449–458, 2009. doi: 10.1109/TE.2008.930097.

PH Penna and Henrique C Freitas. Análise e avaliação de simuladores de sistemas completos para o ensino de arquitetura de computadores. *Int. Journal of Computer Architecture Education*, 2(1):13–16, 2013.

P. W. C. Prasad, Abeer Alsadoon, Azam Beg, and Anthony Chan. Using simulators for teaching computer organization and architecture. *Computer applications in engineering education*, 24(2):215–224, 2016. ISSN 1061-3773.

Zaharije Radivojevic, Zarko Stanisavljevic, and Marija Punt. Configurable simulator for computer architecture and organization. *Computer applications in engineering education*, 26(5):1711–1724, 2018. ISSN 1061-3773.

Sasko Ristov, Marjan Gusev, Blagoj Atanasovski, and Nenad Anchev. Using educache simulator for the computer architecture and organization course. *International Journal of Engineering Pedagogy (iJEP)*, 3(3):pp. 47–56, Jun. 2013. doi: 10.3991/ijep.v3i3.2784. URL `https://online-journals.org/index.php/i-jep/article/view/2784`.

Jorge Fernando Maxnuck Soares, Luís Tadeu M. Raunheitte, and Takato Kurihara. The use of marie cpu simulator in computer architecture course: A case study of student's perception of learning and performance. *Journal of systemics, cybernetics and informatics*, 14(7):7–13, 2016. ISSN 1690-4524.

Leonardo BA Vasconcelos, Max V Machado, and Henrique C Freitas. Ambiente para estudo de computaçao paralela baseado no simulador completo gem5 e em algoritmos de ordenaçao escritos com openmp. *International Journal of Computer Architecture Education (IJCAE)*, 3(1):1–4, 2014.