

JengAOC

Vinícius Gonçalves, Larissa Lumi Hayakawa de Sá, Laura Carvalho Dalsan

Instituto de Ciência e Tecnologia (ICT-Unifesp) - São José dos Campos - SP - Brasil

vinicius.goncalves24@unifesp.br, lumi.larissa@unifesp.br, dalsan@unifesp.br

Resumo. O projeto **JengAOC** é uma adaptação do jogo Jenga para ensinar de forma lúdica o conteúdo de Arquitetura e Organização de Computadores. Os blocos de madeira do jogo são divididos em três grupos: memória, arquitetura de cache e endereçamento. O jogo envolve a retirada de blocos e a resposta a perguntas relacionadas ao terço escolhido, desta forma promovendo aprendizado ativo. O principal objetivo é reforçar conceitos de AOC, estimulando o raciocínio e a compreensão das funções de cada parte do sistema computacional.

1. Introdução

O ensino de Arquitetura e Organização de Computadores tem como objetivo capacitar os estudantes a compreender a estrutura interna dos sistemas computacionais e o funcionamento integrado de seus componentes. No entanto, por se tratar de um conteúdo muitas vezes abstrato, pode ser difícil assimilar os conceitos fundamentais. Essa dificuldade mostra a necessidade de recursos didáticos que aproximem a teoria da prática, tornando o aprendizado mais intuitivo.

Nesse contexto, o JengAOC foi desenvolvido como uma forma mais visual de entender o conteúdo, tendo como base o formato do jogo Jenga. O projeto irá contar com a utilização de blocos que representam os diferentes subsistemas computacionais como memória, arquitetura de cache e endereçamento, permitindo que os estudantes visualizem de forma física e simbólica as relações entre esses elementos.



Figura 1. Ilustração do jogo

Além de fortalecer a compreensão, o JengAOC promove a interação entre os participantes. A cada rodada, os alunos são desafiados a aplicar os conhecimentos teóricos, respondendo questões relacionadas aos componentes representados pelos blocos. Dessa forma incentivando o aprendizado sobre os elementos fundamentais da arquitetura de computadores.

Portanto, o JengAOC busca aliar teoria e prática em uma experiência de ensino que transforma conceitos abstratos em atividades acessíveis.

2. Fundamentação Teórica

A disciplina de Arquitetura e Organização de Computadores tem como propósito compreender como os sistemas computacionais são estruturados e como eles interagem para realizar seus processos principais. Segundo Patterson e Hennessy (2005), a arquitetura de computadores descreve a interação entre hardware e software, estabelecendo os mecanismos pelos quais os componentes do sistema se comunicam e cooperam para executar tarefas de processamento e armazenamento.

Entretanto, o ensino desses conceitos frequentemente apresenta desafios de entendimento, pois a abstração envolvida na representação dos processos internos do computador pode dificultar a compreensão. Nesse contexto, o uso de recursos como jogos concretos e interativos pode contribuir para tornar o aprendizado mais fácil.

Durante a dinâmica, os participantes removem blocos do conjunto e respondem a questões relacionadas aos componentes representados, estabelecendo uma conexão direta entre o jogo e o conteúdo teórico. Essa metodologia promove o aprendizado, em que o aluno deixa de ser um receptor passivo de informações e passa a construir o conhecimento por meio da experimentação e da resolução de problemas.

Além disso, a natureza competitiva e colaborativa do JengAOC estimula o raciocínio lógico e o reforço conceitual dos temas centrais de Arquitetura e Organização de Computadores como a hierarquia de memória, a função dos periféricos e as estruturas de armazenamento. Dessa forma, o jogo atua como um instrumento pedagógico muito eficiente, capaz de integrar teoria e prática, além de promover uma aprendizagem mais prazerosa.

Em resumo, a fundamentação teórica do JengAOC baseia-se nos princípios clássicos de arquitetura e organização de computadores apresentados por Patterson e Hennessy (2005) e Tanenbaum (2007), aliando-os a estratégias modernas de ensino que valorizam a experiência prática como meios de potencializar a compreensão dos sistemas abstratos.

3. Materiais e Métodos

O JengAOC é uma versão modificada do jogo Jenga, composta por blocos de madeira e cartas educativas. A seguir, descreve-se o funcionamento detalhado e os materiais utilizados.

3.1. Estrutura do jogo

Os blocos são divididos em três grupos, cada um com símbolos distintos:

- Primeiro terço: blocos com símbolos de memória;
- Segundo terço: blocos com símbolos de arquitetura de cache;
- Terceiro terço: blocos com símbolos de endereçamento.

Cada grupo contém três símbolos diferentes, representando diferentes tipos dentro daquela categoria nas cartas, os terços estão separados em diferentes cores azul-escuro para memória, magenta para arquitetura de cache e verde-escuro para endereçamento.

3.2. Dinâmica das rodadas

1. O jogador retira um bloco da torre normalmente.
2. O símbolo do bloco indica qual monte de cartas será utilizado.
3. O jogador à direita pega uma carta do monte correspondente e lê a pergunta.
4. As perguntas podem ser de três tipos, divididas em montes:
 - Estrutura
 - Funcionamento
 - Verdadeiro ou falso
5. Cada carta contém a resposta ao lado da pergunta entre parênteses;
6. Se o jogador responder corretamente, ganha um ponto e mantém a peça que foi retirada.

O jogo prossegue até a torre cair ou algum jogador juntar 7 fichas amarelas primeiro. Caso a torre caia o jogador com mais fichas é o que ganha, em caso de empate, vence quem tiver maior variedade de blocos de diferentes terços.

3.3. Materiais utilizados

- Conjunto Jenga de madeira com 45 peças;
- Tintas e canetas para marcação dos símbolos;
- Cartas feitas de papel com 180 de gramatura, contendo perguntas e respostas;
- Pontos amarelos que marcam pontos.

3.4 Perguntas avaliadas

3.4.1. MEMÓRIA

- **Estrutura (9 perguntas)**
 1. O que difere fisicamente a célula de memória SRAM da DRAM? (SRAM usa flip-flops; DRAM usa capacitores).
 2. O que caracteriza a volatilidade em memórias semicondutoras? (A perda de dados quando a alimentação elétrica é interrompida).
 3. Para que servem os pinos RAS e CAS em um chip DRAM? (Selecionar o endereço da Linha e da Coluna na matriz de memória).
 4. Qual a função física do Código de Hamming em memórias? (Detectar e corrigir erros de bit causados por falhas físicas ou interferência)
 5. O que é uma "soft failure" (falha de software) na memória? (Um evento aleatório e não destrutivo que altera o conteúdo de uma célula, como radiação).
 6. O que diferencia a SDRAM da DRAM convencional em termos de sinal? (A SDRAM sincroniza a transferência de dados com o clock do sistema).
 7. O que é a tecnologia DDR (Double Data Rate)? (Uma tecnologia que transfere dados tanto na borda de subida quanto na de descida do clock).
 8. Qual a função do sinal WE (Write Enable) no chip de memória? (Habilitar o modo de gravação de dados na célula endereçada).
 9. Por que a memória DRAM é mais densa que a SRAM? (Porque sua célula é menor e mais simples, usando apenas 1 transistor e 1 capacitor).
- **Funcionamento (9 perguntas)**

1. Por que a operação de "Refresh" é obrigatória na DRAM?
(Porque a carga elétrica nos capacitores vaza com o tempo e precisa ser restaurada).
2. Como o Código de Hamming corrige um erro de 1 bit?
(Comparando bits de paridade lidos com os calculados para identificar a posição do erro).
3. O que acontece fisicamente durante um ciclo de leitura na SDRAM? (O endereço é registrado na borda do clock e o dado é disponibilizado após a latência CAS).
4. Como a DDR alcança maior velocidade que a SDRAM comum?
(Enviando dois dados por ciclo de clock em vez de um).
5. O que causa uma falha de hardware (hard failure) na memória?
(Danos físicos permanentes na célula, como defeitos de fabricação ou desgaste).
6. Como a largura do barramento de dados afeta o chip de memória?
(Define quantos bits são transferidos fisicamente em paralelo a cada ciclo).
7. Por que a SRAM consome mais espaço físico no silício que a DRAM? (Porque cada célula exige mais transistores - geralmente 6 - para formar o flip-flop).
8. Qual a função do buffer de endereço na estrutura da DRAM?
(Armazenar temporariamente os endereços de linha e coluna multiplexados).
9. Como funciona o acesso em "Burst" na memória moderna? (A memória fornece uma sequência de dados a partir de um único endereço inicial).

- **Verdadeiro ou falso (9 perguntas)**

1. A memória SRAM precisa de circuitos de refresh constantes.
(Falso).
2. A memória DDR transfere dados apenas na borda de subida do clock.
(Falso).
3. O Código de Hamming consegue corrigir erros de 2 bits simultâneos.
(Falso)
4. A memória SDRAM opera de forma assíncrona em relação ao processador.
(Falso)
5. Uma "soft failure" danifica permanentemente o chip de memória.
(Falso)
6. Capacitores são os elementos de armazenamento da SRAM.
(Falso)
7. A latência CAS é o tempo entre o envio do endereço da coluna e a disponibilidade do dado.
(Verdadeiro)
8. A memória principal do computador é tipicamente volátil.
(Verdadeiro).

9. A DRAM é mais rápida e mais cara que a SRAM. (Falso)

3.4.2. ARQ. DE CACHE

- **Estrutura (9 perguntas)**

1. Qual a função estrutural da Cache na hierarquia? (Atuar como buffer rápido entre a CPU veloz e a memória principal lenta).
2. O que compõe uma "Linha de Cache"? (Uma tag para identificação e o bloco de dados da memória).
3. O que é o "Mapeamento Direto" na estrutura da cache? (Cada bloco da memória principal tem apenas uma linha fixa possível na cache).
4. O que diferencia o "Mapeamento Associativo"? (Um bloco de memória pode ser carregado em qualquer linha da cache, sem local fixo).
5. O que é uma Cache "Set-Associative"? (O cache é dividido em conjuntos; o bloco tem um conjunto fixo, mas pode usar qualquer linha dentro dele).
6. Para que serve o campo "Tag" no endereço? (Para verificar se o bloco armazenado na linha da cache é o do endereço requisitado).
7. O que é uma Cache de Nível 2 (L2)? (Uma memória maior e um pouco mais lenta que a L1, usada para capturar o que a L1 perdeu).
8. O que é a MMU (Memory Management Unit)? (A unidade de hardware que traduz endereços lógicos/virtuais em endereços físicos)
9. O que é uma Cache Unificada? (Uma cache que armazena tanto instruções quanto dados, equilibrando a carga entre eles).

- **Funcionamento (9 perguntas)**

1. O que define um "Cache Hit"? (Quando o dado requisitado pela CPU é encontrado dentro da memória cache).
2. O que diz o Princípio da Localidade Temporal? (Que os dados acessados agora provavelmente serão acessados novamente logo).
3. Como funciona a política de escrita "Write Through"? (A informação é escrita na cache e na memória principal simultaneamente).
4. Qual a vantagem da política "Write Back"? (Escreve apenas na cache, atualizando a memória principal só quando o bloco é removido, reduzindo tráfego).
5. O que faz o algoritmo de substituição LRU? (Substitui o bloco que está há mais tempo sem ser referenciado na cache).

6. Por que o Mapeamento Direto pode causar "thrashing"? (Porque se dois dados muito usados mapeiam pra mesma linha e ficam se expulsando).
7. O que acontece em um "Cache Miss"? (A execução para, o bloco é buscado na memória principal, copiado para a cache e entregue à CPU).
8. Como o tamanho do bloco afeta o desempenho da cache? (Blocos grandes aproveitam localidade espacial, muito grandes aumenta o miss).
9. Qual a vantagem de caches L1 separadas? (Permite que a CPU busque uma instrução e um dado no mesmo ciclo).

- **Verdadeiro ou falso (9 perguntas)**

1. No mapeamento direto, um bloco de memória pode ir para qualquer linha da cache. (Falso)
2. O Princípio da Localidade Espacial diz que acessaremos endereços vizinhos aos atuais. (Verdadeiro).
3. A política Write Through gera menos tráfego na memória que a Write Back. (Falso)
4. O algoritmo FIFO substitui o bloco que foi carregado há mais tempo. (Verdadeiro).
5. Caches L1 geralmente são maiores que caches L2. (Falso)
6. O mapeamento totalmente associativo exige mais comparadores de hardware que o direto. (Verdadeiro)
7. Uma cache "Split" (dividida) possui áreas separadas para instruções e dados. (Verdadeiro).
8. A MMU fica localizada entre o processador e a memória cache L1 física. (Verdadeiro)
9. Se ocorrer um Cache Miss na L1, o processador busca diretamente no Disco Rígido. (Falso)

3.4.3. ENDEREÇAMENTO

- **Estrutura (9 perguntas)**

1. O que é o Endereçamento Imediato? (O operando é uma constante que está na própria instrução)
2. O que caracteriza o Endereçamento Direto? (O campo de endereço contém o endereço exato da memória onde está o dado)
3. Como é estruturado o Endereçamento Indireto por Registrador? (A instrução aponta para um registrador, e este registrador contém o endereço da memória)
4. O que é o Endereçamento Relativo ao PC? (O endereço final é a soma do PC atual mais um deslocamento constante)

5. O que é um "Effective Address" (EA)? (É o endereço real/final de memória onde o operando está localizado)
6. No modo "Base + Deslocamento", quais são os dois componentes? (Um registrador e o deslocamento na instrução)
7. Para que serve o Endereçamento de Pilha (Stack)? (Para acessar dados no topo da pilha de forma implícita com push/pop)
8. O que é Endereçamento por Indexação? (Soma-se um endereço base a um registrador de índice para percorrer vetores/arrays)
9. O que é a arquitetura Load-Store? (Apenas instruções de carga e escrita acessam a memória; as outras operam em registradores)

- **Funcionamento (9 perguntas)**

1. Por que o Endereçamento Imediato é rápido? (Não requer acesso extra à memória para buscar o dado)
2. Qual a desvantagem do Endereçamento Direto? (Espaço de endereçamento limitado pelo tamanho do endereço na instrução)
3. Como o Endereçamento Relativo ajuda em loops? (Permite saltos curtos para frente ou para trás relativos à posição atual do código)
4. Por que o Endereçamento Indireto é mais lento? (Requer dois acessos à memória: um para pegar o ponteiro e outro para o dado)
5. Como o MIPS calcula o endereço de desvio (Branch)? (Multiplica o imediato por 4 e soma ao PC)
6. Para que serve o registrador \$sp? (Aponta para o endereço do topo da pilha atual na memória)
7. O que acontece no modo de auto incremento? (O endereço é calculado e o registrador de índice é atualizado automaticamente)
8. Como o modo "Load Upper Immediate" funciona no MIPS? (Carrega uma constante nos 16 bits superiores do registrador)
9. Em arquiteturas Big-Endian, como os bytes são ordenados? (O byte mais significativo fica no menor endereço de memória)

- **Verdadeiro ou falso (9 perguntas)**

1. O endereçamento imediato é usado para acessar grandes blocos de memória. (Falso)
2. No MIPS, apenas instruções Load e Store acessam a memória de dados. (Verdadeiro)
3. O endereçamento relativo ao PC tira proveito do princípio da localidade espacial. (Verdadeiro)
4. Endereçamento indireto permite acessar um espaço de memória maior que o direto. (Verdadeiro)
5. Registradores são mais lentos para acessar do que a memória cache. (Falso)

6. O modo de endereçamento de pilha requer que o endereço seja explícito na instrução. (Falso)
7. Little-endian armazena o byte menos significativo no menor endereço. (Verdadeiro)
8. Instruções RISC geralmente suportam muitos modos de endereçamento complexos. (Falso)
9. O cálculo de endereço efetivo pode envolver soma de registradores. (Verdadeiro)

3.5. Teste e avaliação

O protótipo será testado com estudantes da cadeira de Arquitetura e Organização de Computadores, analisando a compreensão dos conceitos e a cooperação entre os participantes. Serão aplicados questionários qualitativos para avaliar a eficácia da proposta no aprendizado dos temas após os jogos. A retenção de informações sobre o conteúdo deve aumentar significativamente após uma partida, já que o cérebro assimila melhor as informações se ele é迫使 a pensar sobre elas e ainda mais em modo de competição.

4. Resultados Esperados

Espera-se que o JengAOC proporcione uma forma mais atrativa para se estudar, auxiliando na fixação dos conceitos de AOC por meio da prática e da gamificação. A interação entre os alunos durante o jogo deve favorecer a discussão dos conceitos e a assimilação de termos técnicos, além de estimular o trabalho em grupo e o raciocínio lógico.

Como resultado esperado, o uso do JengAOC em sala de aula deverá contribuir para maior engajamento e melhoria na compreensão dos fundamentos de arquitetura de computadores.

5. Conclusão

O JengAOC constitui uma alternativa acessível para o ensino de Arquitetura e Organização de Computadores, ao converter conteúdos tradicionalmente teóricos em uma experiência mais lúdica. Por meio da combinação de competição e cooperação, o jogo estimula o aprendizado ativo, favorecendo a construção de conhecimento de forma tanto autônoma quanto colaborativa. Essa abordagem possibilita que os estudantes visualizem, de modo concreto, as relações entre os componentes de um sistema computacional, fortalecendo a compreensão dos princípios fundamentais de Arquitetura e Organização de Computadores.

Além disso, o JengAOC se destaca por ser fácil de reproduzir, mesmo sem um jenga original, é possível simplesmente anotar símbolos em papéis e então sortear a coleta desses símbolos para ganhar pontos, então as perguntas podem ser usadas como flashcards para estudo guiado em grupo, já que uma pessoa vai aprender pensando sobre a resposta e outra vai associar a pergunta com a resposta logo abaixo, e esse papel vai se alternando, assim fazendo com que todos assimilam muito melhor o conteúdo. A estrutura do jogo permite que ele seja facilmente ajustado para outras disciplinas também, por se tratar de um formato bem amplo, só é necessário adequar o conteúdo das cartas e desafios às especificidades de cada área. Dessa forma, o projeto não apenas contribui para o ensino de Arquitetura e Organização de Computadores, mas também se apresenta como um modelo pedagógico versátil, capaz de integrar teoria e prática em diferentes cadeiras.

Em resumo, o JengAOC demonstra que é possível aliar rigor conceitual e aprendizagem lúdica sem comprometer a profundidade dos conteúdos técnicos, promovendo um ensino mais engajador, inclusivo e alinhado às demandas contemporâneas da educação.

Referências Bibliográficas

1. PATTERSON, David A.; HENNESSY, John L. *Organização e projeto de computadores: a interface hardware/software*. 3. ed. Rio de Janeiro: Campus, 2005. 484 p. ISBN 9788535215212.
2. TANENBAUM, Andrew S. *Organização estruturada de computadores*. 5. ed. São Paulo: Pearson, 2007. 449 p. ISBN 9798576050673