UNIVERSIDADE ESTADUAL DO PARANÁ - *CAMPUS* APUCARANA

João Vitor de Souza Ribeiro

RELATÓRIO TÉCNICO - AOC

APUCARANA – PR
2023

João Vitor de Souza Ribeiro

RELATÓRIO TÉCNICO – AOC

Trabalho apresentado à disciplina de Arquitetura e Organização de Computadores do curso de Bacharelado em Ciência da Computação.

Professor: Guilherme Henrique de Souza Nakahata;

APUCARANA – PR
2023

SUMÁRIO

INTRODUÇÃO

Como já é de conhecimento geral, as disciplinas do curso de Ciência da Computação se mostram muito ligadas à abstração, de modo com que se torne mais complexo e exaustivo o ensino e aprendizado das disciplinas que compõe a matriz curricular. Dessa forma, fazem-se necessários meios e projetos para a diminuição desta abstração, de maneira que se coloque métodos práticos para a exibição de conteúdos programaticamente teóricos, como na disciplina de Arquitetura e Organização de Computadores. Neste trabalho, por exemplo, mostraremos a aplicação de um ciclo de instruções, dado uma tabela finita com instruções pré-estabelecidas, de maneira prática, com aplicação em código funcional e interativo. Em geral, o programa recebe do usuário a sua intenção (instrução que deseja executar) e os operandos necessários para realizar tal instrução, seja ela de armazenamento, modificação ou jump. Após, realiza, de uma só vez, todas as instruções desejadas, incluindo os jumps, se existirem.

A necessidade de aplicações práticas em disciplinas como esta pode ser baseada, por exemplo, na metodologia *Peer Instruction*, criada na década de 90 pelo físico e professor da Universidade de Harvard, Eric Mazur. No método citado, utilizam-se de meios práticos, como a resolução de exercícios individuais, e revisões em grupo ou duplas, de modo que, segundo o físico, os estudantes tenham uma participação ativa na construção de conhecimento. As metodologias de ensino ativas basicamente se referem àquelas que fogem do habitual, como é o caso de uma aplicação em código de um conteúdo antes somente visto na teoria. O ciclo de instruções é uma teoria que exemplifica o funcionamento de uma memória interna e suas buscas por instruções, aplicando-os a um meio semelhante ao método de Mazur, como neste trabalho, possibilita que os estudantes interajam entre si, compartilhando conhecimentos e, ainda, criando relações mais sólidas com o conteúdo, por meio de uma atividade prática. Isso, porque, de acordo com Mazur (2015), muitas vezes, os estudantes podem ensinar o conteúdo entre si de maneira mais eficaz que o próprio professor do curso, seja por entenderem as dificuldades de seus semelhantes ou ainda pelos meios diversos de ensino que cada um deles pode desempenhar em um panorama geral.

O presente documento evidencia os principais pontos relevantes para o entendimento do algoritmo aplicado, além de esmiuçar os objetivos e resultados obtidos ao longo do processo. Em todo o seu decorrer, serão pontuadas as estruturas de dados utilizadas, como foram aplicadas e o que cada parte do código funcional têm de relação com o objetivo final, assim oferecendo um maior aproveitamento das informações gerais contidas no documento interativo.

CAPÍTULO 1
OBJETIVOS

O objetivo principal do código criado pode ser interpretado como estabelecer o funcionamento prático e fictício de um ciclo de instruções, como visto no primeiro bimestre da disciplina de Arquitetura e Organização de Computadores. Neste ciclo temos oito etapas, desde a busca de instrução – que inicia o ciclo –, até o armazenamento dos resultados e cálculo de endereçamento de instrução, caso exista uma próxima. As etapas citadas, além das outras, podem ser visualizadas na Figura 1.

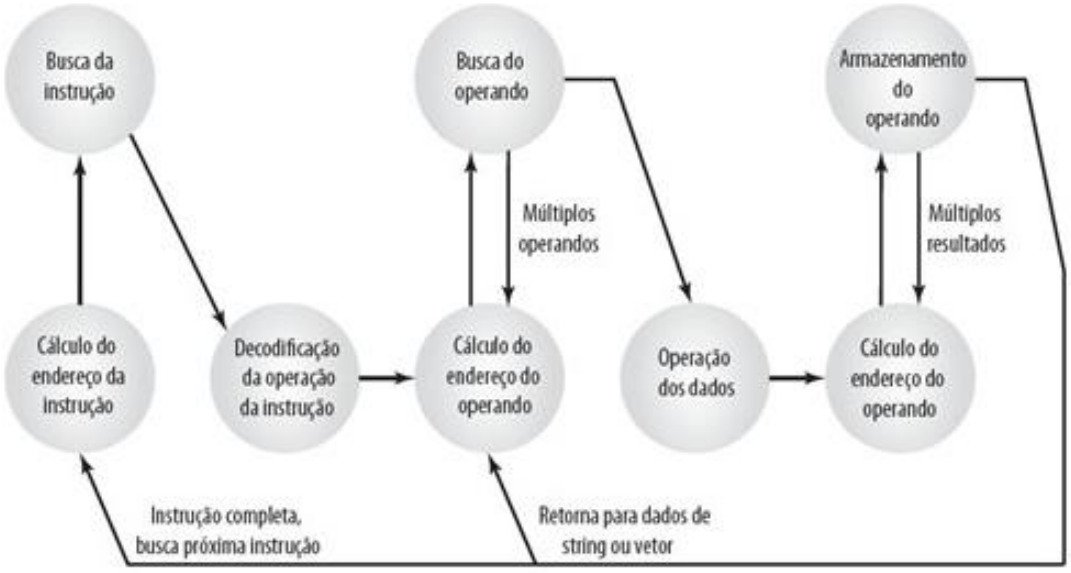

Figura 1

Dessa forma, de um modo geral, o código fonte têm como objetivo principal a recriação das etapas do ciclo de intruções, de maneira que, dado um opcode – instrução – e seu(s) operando(s), possa realizá-la com um desempenho agradável, demostrando o processo pelo qual os computadores realizam o mesmo ciclo, mas, claramente, de um modo mais didático e sem total fidelidade com a realidade, assim obtendo um trabalho realizado de maneira “fictícia”, somente como um material de demonstração visual, basicamente.

CAPÍTULO 2
MOTIVAÇÃO E RECURSOS UTILIZADOS

Baseando-se no exposto anteriormente, devemos explicitar os motivos para a realização do trabalho, ou seja, o objetivo final e os recursos utilizados para que isso seja cumprido.

2.1 Motivação

Como citado no capítulo que trata acerca dos objetivos do projeto em questão, a motivação também seria a realização de um código fonte funcional em que possamos demonstrar as etapas que um ciclo de instruções realiza para que uma instrução seja buscada, armazenada, efetuada e seu resultado armazenado em uma memória. De maneira análoga a Figura 1, na Figura 2, podemos entender um pouco das motivações para a criação do código supracitado. Na Figura em questão, presente logo abaixo, podemos visualizar o conjunto finito de instruções utilizadas para que o programa possa realizar as etapas do ciclo em cada uma delas, de modo a desempenhar a execução somente ao final do recebimento destas – após o opcode 001100 ser lido.

Código da Instrução	Operandos	Resultado
000001	#pos	MBR ← #pos
000010	#pos #dado	#pos ← #dado
000011	#pos	MBR ← MBR + #pos
000100	#pos	MBR ← MBR - #pos
000101	#pos	MBR ← MBR * #pos
000110	#pos	MBR ← MBR / #pos
000111	#lin	JUMP to #lin
001000	#lin	JUMP IF Z to #lin
001001	#lin	JUMP IF N to #lin
001010	-	MBR ← raiz_quadrada(MBR)
001011	-	MBR ← - MBR
001111	#pos	#pos ← MBR
001100	-	NOP

Figura 2

Assim, faz-se necessário, com as informações pertinentes acerca dos objetivos e motivações, esmiuçar-se àqueles dados relevantes à Estrutura de Dados, Linguagem de Programação e demais questões acerca da implementação do código em questão.

2.2 Estrutura de Dados

De modo geral, serão utilizados 4 (quatro) vetores, cada um com funções e atribuições diferentes, de maneira com que se consiga efetuar de maneira eficaz o desenvolvimento do projeto.

Para representar o vetor geral de acesso as memórias, foi criado um *array* ***vet*** de 250 posições úteis, para que, ao longo da execução, o usuário possa inserir dados nestas posições e operar estes dados da maneira que achar melhor. No momento do recebimento dos dados, estes serão armazenados em vetores individuais, de maneira que se possa controlar cada um deles na execução das instruções.

No caso do recebimento das instruções e dos operandos, três vetores serão utilizados, ***opcode*** (para armazenar os códigos de instrução), ***pos_lin*** (para armazenar os operandos correspondentes as posições ou linhas – no caso de jump) e ***dado*** (para armazenar os operandos correspondentes aos dados, no caso do opcode ser 000010).Ainda acerca das variáveis, o programa possui três variáveis de acesso global nas funções do arquivo, ***cont*** (responsável por contar a quantidade de instruções totais no recebimento – o PC), ***mbr*** (representando o registrador de buffer de memória, responsável por conter os valores a serem guardados) e ***inicio*** (utilizado para controlar o retorno do looping for quando um jump é encontrado).

A respeito da estrutura do código em si, o programa contém a função tradicional ***main*** (onde é executado o menu de escolha das opções – uso detalhado na documentação – criado o vetor principal e preenchido de valores *Integer.MIN_VALUE*), a função ***ReceberDados*** (responsável por receber os dados acerca das instruções e operandos de uma só vez, armazenando-os nos vetores mencionados), a função ***Executar*** (que executa as instruções de acordo com o armazenado nos vetores de recebimento, mostrando passo a passo do ciclo de instruções), a função ***Impressao*** (responsável por receber um “título” e adaptá-lo ao espaço escolhido para impressão no console, ou seja, editando o

visual da execução do programa, somente para meios visuais) a função *ImprimirDado* (que permite ao usuário visualizar se uma posição ainda está vazia ou qual o seu conteúdo, além do valor atual presente no *mbr*), além da função *Limpar* (permite que o usuário exclua os dados do vetor de posições e/ou limpe o conteúdo do mbr, zerando-o) e daquela intitulada *Instruções* (responsável apenas pela impressão da tabela de instruções completa, seguido do questionamento ao usuário se deseja ou não inserir dados a seguir).

Os demais recursos/variáveis utilizadas apenas corroboram a execução do código e das funções, como aquelas utilizadas dentro de laços de repetição ou auxiliares necessários para realizar o funcionamento real do programa.

2.3 Linguagem de Programação e demais informações

A linguagem escolhida para a implementação do código foi Java, de maneira que se utiliza-se a linguagem principalmente abordada no segundo ano de formação para implementar o ciclo de instruções. Java é uma linguagem de alto nível desenvolvida pela Sun Microsystems, orientada a objetos e amplamente utilizada em vários ramos da programação. A única biblioteca da linguagem utilizada para o funcionamento do código foi o Scanner, para recebimento dos dados pelo usuário, via console.

CAPÍTULO 3
RESULTADOS

Mediante os objetivos apresentados, o resultado esperado seria o pleno funcionamento de um código que exemplifique o ciclo de instruções de uma memória. Dessarte, com a implementação completa e revisada, os resultados foram atingidos, resultando em uma aplicação interativa e funcional, recebendo as entradas de uma só vez, uma a uma, e, após, executando-as. Nas figuras 3,4 e 5 podemos observar a inserção de dados e a execução das instruções ao final, após o opcode 001100.

```
Opção escolhida 1: INSERIR INSTRUÇÃO
================================================================================
PC: 1
Digite o opcode:
000010
Digite o operando 1 (pos):
250
Digite o operando 2 (dado):
5
================================================================================
================================================================================
PC: 2
Digite o opcode:
000001
Digite o operando 1 (pos):
250
================================================================================
================================================================================
PC: 3
Digite o opcode:
001100
Operações Encerradas
==================== FINALIZANDO RECEBIMENTO DAS INSTRUÇÕES ====================
--------------------------------------------------------------------------------
                                EXECUTANDO...
--------------------------------------------------------------------------------
======================== BUSCA DO ENDEREÇO DE INSTRUÇÃO ========================
Program Counter: 1
============================== BUSCA DA INSTRUÇÃO ==============================
Opcode: 000010
Posição (1ºoperando): 250
Dado (2ºoperando): 5
========================== DECODIFICANDO INSTRUÇÃO ==========================
#pos <-- #dado
#250 <-- 5
====================== CALCULANDO O ENDEREÇO DO OPERANDO ======================
Endereço: 250
```

Figura 3

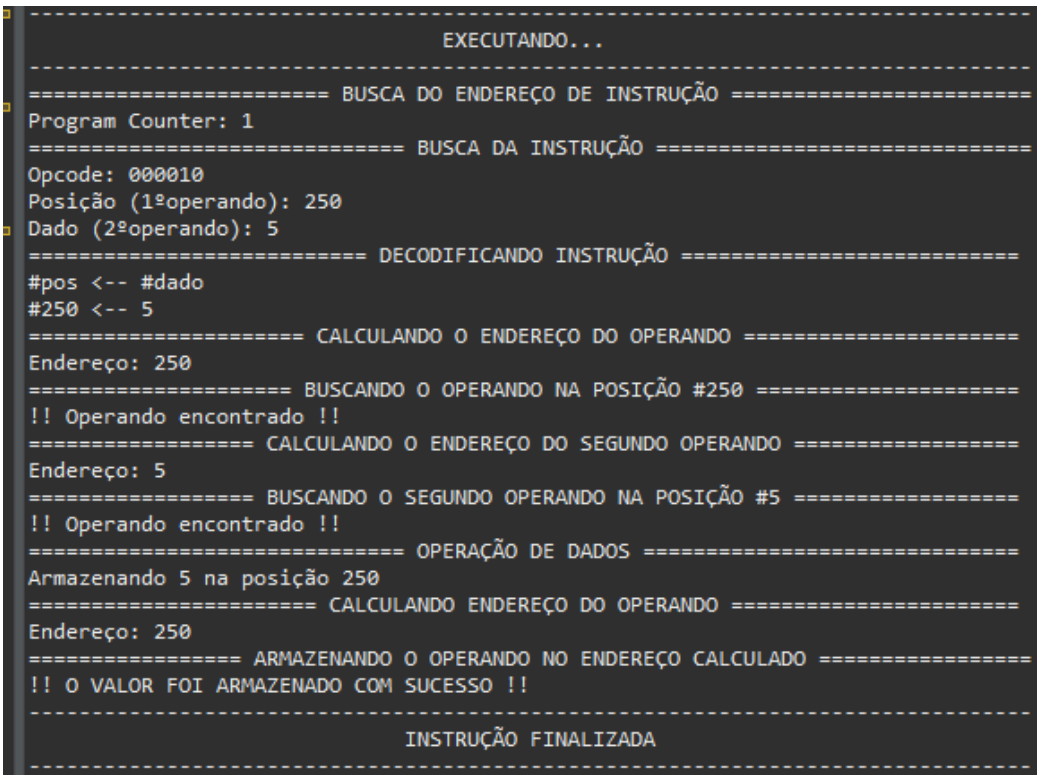

Figura 4

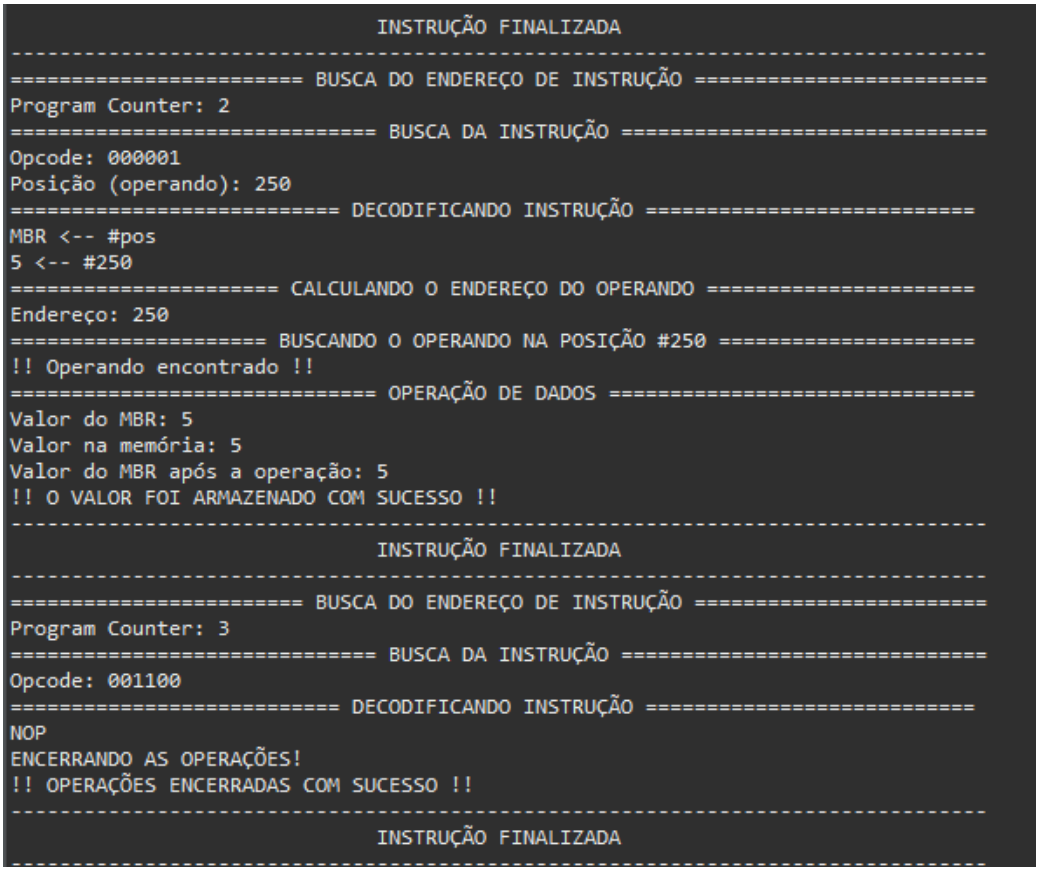

Figura 5

CONCLUSÃO

Com base no exposto anteriormente, podemos evidenciar que a diminuição da abstração ser faz extremamente necessária em disciplinas como esta, aplicando a necessidade a um código funcional observamos que o aprendizado prático se faz, em grande parte das vezes, um auxiliar vantajoso para as disciplinas teóricas, uma vez que transformam os conteúdos em atividades que devem ser interpretadas, entendidas e executadas.

Mazur (2015), já citado anteriormente, esclarece que métodos diferenciados de ensino podem ser mais interessantes que os comuns ao passo que influenciam os estudantes a buscarem os conhecimentos de outras formas, seja em trocas entre colegas ou com pesquisas adicionais em livros e sites de busca. Assim, entende-se que o código apresentado, por sua extensão, principalmente, desempenhou um papel interessante no processo de aprendizado, ao menos neste caso específico.

REFERÊNCIAS

MAZUR, Eric. **Peer Instruction -A Revolução da Aprendizagem Ativa**.Editora Penso. Ano 2015.

MOZZAQUATRO, M. C. Patricia; QUARESMA, R. T. Cindia; BILLING, B. G. Solange. 2018. **Aplicação do Método de ensino Peer Instruction para o Ensino de Lógica de Programação com acadêmicos do Curso de Ciência da Computação**. In Anais do 5º SENID (Seminário Nacional de Inclusão Digital) - Cultura Digital na Educação.

PLANTIER, M. Carla; AMBRÓSIO, Z. M. Ana Paula; ROSAN, C. G. Raquel; DE OLIVEIRA, S. Sidinei; APARECIDA, D. L. T. Adriana. **Peer Instruction: Metodologia Ativa de Ensino e Aprendizagem e suas Ferramentas de Interatividade Gratuitas**. Colloquium Humanarum, v. 14, n. Especial, p. 644–650, 2017.