UNIVERSIDADE ESTADUAL DO PARANÁ - *CAMPUS* APUCARANA

Paloma de Castro Leite

RELATÓRIO TÉCNICO - AOC

APUCARANA – PR
2024

Paloma de Castro Leite

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
2024

SUMÁRIO

INTRODUÇÃO

A Arquitetura e Organização de Computadores (AOC) é o estudo do funcionamento interno, da estrutura e da implementação de sistemas de computador. A organização define como o sistema é estruturado, enquanto a arquitetura se concentra nas propriedades que afetam diretamente a execução do programa e são visíveis para o programador. Dessa maneira, AOC envolve o projeto de computadores, dispositivos de armazenamento de dados e componentes de rede que armazenam e executam programas, transferem dados e interagem entre computadores, redes e usuários.

Neste trabalho, iremos analisar especificamente o ciclo de instrução, também chamado de ciclo de busca e execução ou ciclo busca-execução. O ciclo de instrução é o período de tempo no qual um computador lê e processa uma instrução em linguagem de máquina da sua memória, além de compreender a sequência de ações que a CPU realiza para executar cada instrução em código de máquina num programa. Esse ciclo é fundamental para o funcionamento dos sistemas de computador, pois determina a eficiência e a velocidade com que as instruções são processadas.

CAPÍTULO 1
OBJETIVOS

Este trabalho tem por objetivo apresentar uma proposta de um simulador de ciclo de instrução, onde a CPU realiza várias etapas, incluindo a busca da instrução na memória, a decodificação da instrução para entender quais ações são necessárias, a execução da instrução e a gravação do resultado de volta na memória ou em registradores. Esse processo envolve o uso de diversos componentes do sistema, como o contador de programa (PC), o registrador de instrução (IR), o registrador de endereço de memória (MAR), o registrador de buffer de memória (MBR) e as flags de condição, como as flags de Zero e Negativo.

Portanto, compreender o funcionamento do ciclo de instrução é essencial para ver como os programas são executados em nível de hardware, permitindo maior desempenho no processador já que pode executar instruções na mesma sequência para diferentes programas. Isso evita que algumas ações do processador fiquem paradas, para isso é utilizada a pipelining, que consiste em reduzir o tempo de execução de um conjunto de instruções, sendo que o tempo para executar uma instrução continua o mesmo, mas a quantidade de instruções executadas por um período de tempo aumenta.

Neste trabalho, exploraremos as diferentes etapas do ciclo de instrução, analisando o papel de cada componente envolvido e os mecanismos que garantem a correta execução das instruções.

CAPÍTULO 2
MOTIVAÇÃO E RECURSOS UTILIZADOS

Tendo em vista o que foi descrito anteriormente, devemos explicitar os motivos para a realização do trabalho e seu objetivo final, além dos recursos utilizados para que o trabalho seja executado de maneira eficiente e concisa.

2.1 Motivação

Conforme mencionado no capítulo que trata dos objetivos do trabalho, a motivação é a execução de um código fonte funcional que demonstre e exemplifique o funcionamento de um ciclo de instruções. O código deve executar múltiplas instruções inseridas pelo usuário, simulando o ciclo de instruções e garantindo que o processador não pule nenhuma etapa, isso é necessário pois as instruções são armazenadas de forma sequencial e, se alguma etapa for perdida, o programa não conseguirá concluir a instrução corretamente, resultando em resultados incorretos ou em um erro.

O ciclo de instruções é um processo que requer várias etapas para ser executado. Primeiramente, o processador busca a instrução na memória, em seguida, ele decodifica a instrução e a executa de acordo com as especificações fornecidas. O comando pode ser uma operação de leitura/gravação de dados ou uma operação de lógica aritmética. Após a execução da instrução, o processador realiza uma operação de armazenamento para que os resultados possam ser guardados na memória.

Portanto, é necessário, com base nas informações pertinentes sobre os objetivos e motivações, detalhar os dados mais relevantes acerca da Estrutura de Dados, Linguagem de Programação e demais aspectos da implementação do código fonte em questão para uma melhor análise de suas funcionalidades, a fim de obter uma conclusão geral.

2.2 Estrutura de Dados

Este código em Java implementa um ciclo de instruções, onde o programa deve simular como os registradores PC, IR, MAR, MBR e as flags de condição de zero e negativas funcionam de maneira prática. As simulações devem realizar as etapas descritas na figura (1) a seguir:

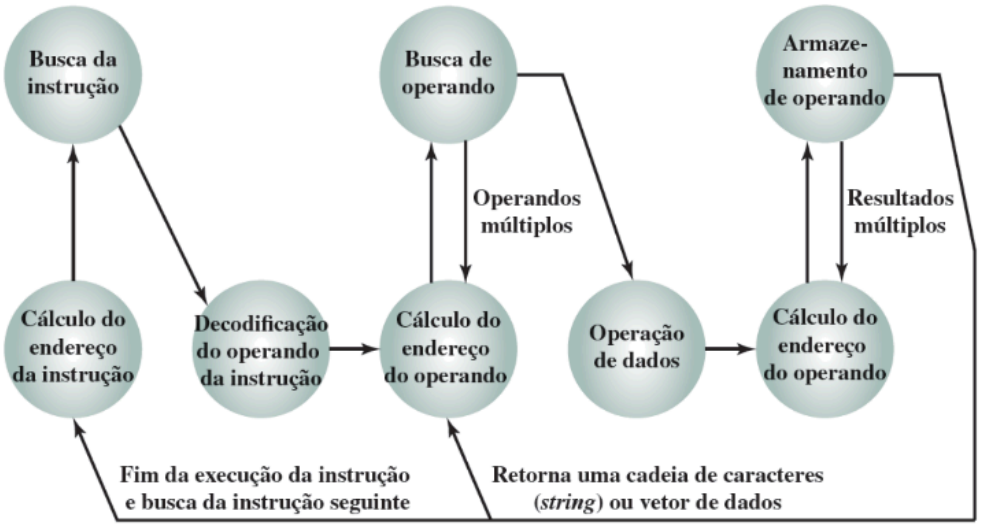

Figura 1 - Ciclo de Instruções

Inicialmente, o código foi separado nas classes principais *ControlaCicloInstrucao* e *UsarCicloInstrucao,* e em uma interface chamada *CicloInstrucao* para facilitar a compreensão e gerenciamento de cada parte do código, além de possibilitar novas implementações e alterações em suas funcionalidades sem afetar outras partes do sistema. A seguir serão explicadas o funcionamento de cada classe e da interface respectivamente para melhor entendimento:

a) ControlaCicloInstrucao: implementando a interface *CicloInstrucao,* inicialmente é incluído o pacote padrão *java.util.** que contém uma coleção de classes e interfaces utilitárias, além das variáveis de instância que armazenam os dados para

configurar e executar o Ciclo de Instrução. Essa classe é a implementação do simulador em que estamos trabalhando e que permite a entrada das instruções e seus respectivos operandos. Ela contém alguns métodos que serão explicados a seguir:

- *entradaUsuario():* permite que o usuário insira as instruções que deseja executar no simulador, seus operandos (se houver), e as armazena em uma lista chamada instruções. Se o usuário inserir '4', retorna ao menu de opções.
- *verInstrucoes()*: exibe as instruções que foram inseridas pelo usuário e uma tabela que descreve cada opcode, seus respectivos operandos e o resultado, como pode ser visto na figura a seguir:

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 - Instruções do simulador

- *executarTodasInstrucoes():* usa um loop while que permite executar todas as instruções inseridas pelo usuário e incrementa o contador de programa (PC) a cada iteração para obter a instrução atual.
- *executaInstrucao(String instrucao):* permite, através do switch case, identificar e executar uma instrução de

acordo com seu opcode (chama o método correspondente), após a execução da instrução é incrementado o PC para que o ciclo possa continuar.

- *exibeCiclo():* permite obter a instrução atual com base no PC, além de verificar se o PC está dentro do intervalo de instruções. Usa a estrutura de decisão switch case para exibir os detalhes do ciclo de instruções como cálculo do endereço, busca, decodificação, e execução. Para cada opcode, exibe passos específicos do ciclo de instrução.
- *verificaFlags():* permite atualizar as flags de condição denominadas flagZero e flagNegativa baseadas no valor de MBR.

A classe ControlaCicloInstrucao também contém métodos para executar as instruções especificadas na tabela de instruções (figura 2), onde cada uma realiza suas operações de acordo com o que foi determinado para aquela instrução.

b) UsarCicloInstrucao: é a classe principal que contém o método main e fornece um menu para o usuário interagir com o simulador de ciclo de instrução. Essa classe contém uma variável para armazenar a opção escolhida pelo usuário no menu, uma instância de objeto da classe ControlaCicloInstrucao que será utilizado para controlar o ciclo de instrução e um Scanner para ler as entradas do usuário. Também contém um do while que exibe as opções do menu (inserir as instruções, ver as instruções, executar todas as instruções ou sair do programa) para o usuário e um switch case para controlar esse menu.

c) CicloInstrucao: interface para representar o ciclo de instrução e suas operações, onde cada operação é uma instrução e cada instrução é um método, assim podemos apenas chamar o método para executar a instrução desejada e exibir o ciclo de instrução.

A execução do programa finaliza ao usuário digitar “4” no ambiente de menu,

garantindo que o programa feche apenas após termos obtido e visualizado todas as instruções desejadas e seus respectivos ciclos. Algumas outras maneiras de organizar e gerir o código fonte foram utilizadas, como o código ANSI para cores que foi empregado apenas para fins estéticos e de melhor visualização, porém apenas com o objetivo de facilitar e possibilitar o desempenho geral do mesmo.

2.3 Linguagem de Programação e demais informações

A linguagem de programação Java foi escolhida para o desenvolvimento do código, ela é abordada principalmente durante o segundo ano do curso, e utilizá-la para o desenvolvimento do programa permite conhecer mais sobre a linguagem e se desenvolver. Esta linguagem fornece a possibilidade de utilizar linguagem orientada a objetos, uma prática de programação que torna possível elaborar um software a partir da geração de objetos que se comunicam entre si, e o uso de classes para representar objetos do mundo real, onde declararmos atributos e métodos, que representam, respectivamente, as características e comportamentos desse objeto.

Dessa maneira, a linguagem de programação Java se tornou ideal para este caso específico, em que o código fonte foi feito para simular um Ciclo de Instruções, pois permite organizar de maneira mais fácil e lógica já que cada classe tem sua funcionalidade definida a fim de facilitar a compreensão e desenvolvimento do código, assim como realizar mudanças sem impactar toda a estrutura.

CAPÍTULO 3
RESULTADOS

Diante dos expostos apresentados ao longo do trabalho, a finalidade esperada seria o pleno funcionamento de um código que exemplifica como um simulador de Ciclo de Instruções funciona de maneira prática, demonstrando como cada registrador e as flags de condição age de acordo com a instrução especificada. Dessa maneira, com a implementação total e sua revisão, o objetivo principal do código fonte foi atingido, resultando em uma aplicação funcional no qual recebemos as instruções desejadas e seus respectivos operandos (se houver), e em seguida realizamos sua execução para visualizar o ciclo.

Abaixo, nas figuras 3, 4, 5, 6, 7, 8 e 9 podemos ver o funcionamento do ciclo de instruções a partir do exemplo dado pelo professor nas especificações do trabalho. Resultados:

```
--- Arquitetura e Organização de Computadores ---

Trabalho de AOC  - 2º Bimestre - C.C UNESPAR
Paloma de Castro Leite - 2ª Ano - 29.07.24

 --- SIMULADOR DE CICLO DE INSTRUÇÕES ---

================================================================================
= OPÇÕES: =
================================================================================
1. INSERIR
2. VER INSTRUÇÕES
3. EXECUTAR
4. SAIR DO PROGRAMA
================================================================================
Escolha uma opção: 1
Digite as instruções do programa (ou '4' para sair da inserção de dados):

Digite o código da instrução (ou '4' para voltar ao menu): 000010
Digite o primeiro operando: 251
Digite o segundo operando: 5

Digite o código da instrução (ou '4' para voltar ao menu): 000010
Digite o primeiro operando: 252
Digite o segundo operando: 10

Digite o código da instrução (ou '4' para voltar ao menu): 000010
Digite o primeiro operando: 253
Digite o segundo operando: 15
```

Figura 3

```
Digite o código da instrução (ou '4' para voltar ao menu): 000001
Digite o primeiro operando: 251
Digite o segundo operando: 0

Digite o código da instrução (ou '4' para voltar ao menu): 000011
Digite o primeiro operando: 252

Digite o código da instrução (ou '4' para voltar ao menu): 000011
Digite o primeiro operando: 253

Digite o código da instrução (ou '4' para voltar ao menu): 001111
Digite o primeiro operando: 254

Digite o código da instrução (ou '4' para voltar ao menu): 001100

Digite o código da instrução (ou '4' para voltar ao menu): 4

================================================================================
= OPÇÕES: =
================================================================================
1. INSERIR
2. VER INSTRUÇÕES
3. EXECUTAR
4. SAIR DO PROGRAMA
================================================================================
Escolha uma opção: 2
================================================================================
= INSTRUÇÕES: =
================================================================================
COD         OP1         OP2         RESULTADOS
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 <- sqrt(MBR)
001011      -           -           MBR <- -MBR
001111      #POS        -           #POS <- MBR
001100      -           -           NOP
================================================================================
```

Figura 4

```
================================================================================
Instruções que já foram inseridas:
000010 251 5
000010 252 10
000010 253 15
000001 251 0
000011 252
000011 253
001111 254
001100

================================================================================
= OPÇÕES: =
================================================================================
1. INSERIR
2. VER INSTRUÇÕES
3. EXECUTAR
4. SAIR DO PROGRAMA
================================================================================
Escolha uma opção: 3
================================================================================
EXECUTANDO
================================================================================
CÁLCULO DO ENDEREÇO DA INSTRUÇÃO:
PC: 000001

BUSCANDO A INSTRUÇÃO:
IR <OPCODE>: 000010
IR <OP1>: 251
IR <OP2>: 5

DECODIFICANDO A INSTRUÇÃO:
 POS <- #DADO
251 <- 5

CÁLCULO DO ENDEREÇO DO OPERANDO:
Endereço: 251

BUSCANDO O OPERANDO NA POSIÇÃO:
MAR: 251

CÁLCULO DO ENDEREÇO DO SEGUNDO OPERANDO:
Endereço: 5

BUSCANDO O SEGUNDO OPERANDO NA POSIÇÃO:
MAR: 5
```

Figura 5

```
OPERAÇÃO DE DADOS:
ARMAZENANDO: 5
NA POSIÇÃO: 251

CALCULANDO ENDEREÇO DO OPERANDO:
ENDEREÇO: 251

ARMAZENANDO O OPERANDO:
MAR: 251
O VALOR FOI ARMAZENADO!

================================================================================
CÁLCULO DO ENDEREÇO DA INSTRUÇÃO:
PC: 000002

BUSCANDO A INSTRUÇÃO:
IR <OPCODE>: 000010
IR <OP1>: 252
IR <OP2>: 10

DECODIFICANDO A INSTRUÇÃO:
 POS <- #DADO
252 <- 10

CÁLCULO DO ENDEREÇO DO OPERANDO:
Endereço: 252

BUSCANDO O OPERANDO NA POSIÇÃO:
MAR: 252

CÁLCULO DO ENDEREÇO DO SEGUNDO OPERANDO:
Endereço: 10

BUSCANDO O SEGUNDO OPERANDO NA POSIÇÃO:
MAR: 10

OPERAÇÃO DE DADOS:
ARMAZENANDO: 10
NA POSIÇÃO: 252
CALCULANDO ENDEREÇO DO OPERANDO:
ENDEREÇO: 252

ARMAZENANDO O OPERANDO:
MAR: 252
O VALOR FOI ARMAZENADO!
```

Figura 6

```
================================================================================
CÁLCULO DO ENDEREÇO DA INSTRUÇÃO:
PC: 000003

BUSCANDO A INSTRUÇÃO:
IR <OPCODE>: 000010
IR <OP1>: 253
IR <OP2>: 15

DECODIFICANDO A INSTRUÇÃO:
 POS <- #DADO
253 <- 15

CÁLCULO DO ENDEREÇO DO OPERANDO:
Endereço: 253

BUSCANDO O OPERANDO NA POSIÇÃO:
MAR: 253

CÁLCULO DO ENDEREÇO DO SEGUNDO OPERANDO:
Endereço: 15

BUSCANDO O SEGUNDO OPERANDO NA POSIÇÃO:
MAR: 15

OPERAÇÃO DE DADOS:
ARMAZENANDO: 15
NA POSIÇÃO: 253

CALCULANDO ENDEREÇO DO OPERANDO:
ENDEREÇO: 253

ARMAZENANDO O OPERANDO:
MAR: 253
O VALOR FOI ARMAZENADO!

================================================================================
CÁLCULO DO ENDEREÇO DA INSTRUÇÃO:
PC: 000004
BUSCANDO A INSTRUÇÃO:
IR <OPCODE>: 000001
IR <OP1>: 251
IR <OP2>: 0
```

Figura 7

```
DECODIFICANDO A INSTRUÇÃO:
 MBR <- #POS
5 <- 251

CÁLCULO DO ENDEREÇO DO OPERANDO:
Endereço: 251

BUSCANDO O OPERANDO NA POSIÇÃO:
MAR: 251

CÁLCULO DO ENDEREÇO DO SEGUNDO OPERANDO:
Endereço: 0

BUSCANDO O SEGUNDO OPERANDO NA POSIÇÃO:
MAR: 0

OPERAÇÃO DE DADOS:
EXECUTANDO OPERAÇÃO
VALOR DO MBR: 5
O VALOR FOI ARMAZENADO!

================================================================================
CÁLCULO DO ENDEREÇO DA INSTRUÇÃO:
PC: 000005

BUSCANDO A INSTRUÇÃO:
IR <OPCODE>: 000011
IR <OP1>: 252

DECODIFICANDO A INSTRUÇÃO:
  MBR <- MBR + #POS
MBR <- 15 + 252

CÁLCULO DO ENDEREÇO DO OPERANDO:
Endereço: 252

BUSCANDO O OPERANDO NA POSIÇÃO:
MAR: 252
OPERAÇÃO DE DADOS:
VALOR DO MBR: 15
VALOR DO CONTEÚDO NA POSIÇÃO: 10
VALOR DO MBR APÓS A OPERAÇÃO: 15 + 10 = 25
O VALOR FOI ARMAZENADO!

================================================================================
```

Figura 8

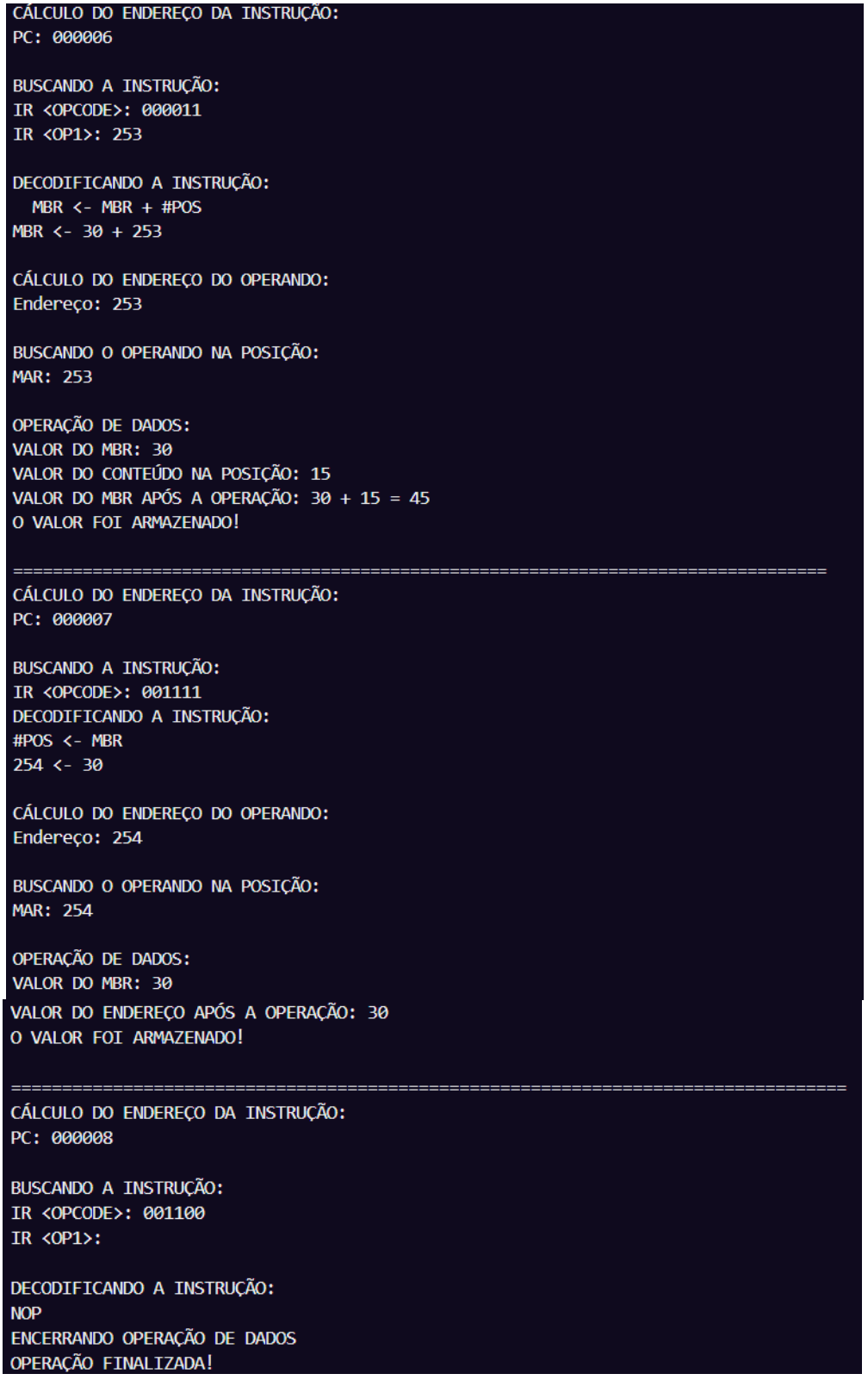
CÁLCULO DO ENDEREÇO DA INSTRUÇÃO:
PC: 000006
BUSCANDO A INSTRUÇÃO:
IR <OPCODE>: 000011
IR <OP1>: 253
DECODIFICANDO A INSTRUÇÃO:
MBR <- MBR + #POS
MBR <- 30 + 253
CÁLCULO DO ENDEREÇO DO OPERANDO:
Endereço: 253
BUSCANDO O OPERANDO NA POSIÇÃO:
MAR: 253
OPERAÇÃO DE DADOS:
VALOR DO MBR: 30
VALOR DO CONTEÚDO NA POSIÇÃO: 15
VALOR DO MBR APÓS A OPERAÇÃO: 30 + 15 = 45
O VALOR FOI ARMAZENADO!
CÁLCULO DO ENDEREÇO DA INSTRUÇÃO:
PC: 000007
BUSCANDO A INSTRUÇÃO:
IR <OPCODE>: 001111
DECODIFICANDO A INSTRUÇÃO:
#POS <- MBR
254 <- 30
CÁLCULO DO ENDEREÇO DO OPERANDO:
Endereço: 254
BUSCANDO O OPERANDO NA POSIÇÃO:
MAR: 254
OPERAÇÃO DE DADOS:
VALOR DO MBR: 30
VALOR DO ENDEREÇO APÓS A OPERAÇÃO: 30
O VALOR FOI ARMAZENADO!
CÁLCULO DO ENDEREÇO DA INSTRUÇÃO:
PC: 000008
BUSCANDO A INSTRUÇÃO:
IR <OPCODE>: 001100
IR <OP1>:
DECODIFICANDO A INSTRUÇÃO:
NOP
ENCERRANDO OPERAÇÃO DE DADOS
OPERAÇÃO FINALIZADA!

Figura 9

Agora, nas figuras 10, 11, 12, 13, 14 e 15 podemos ver o funcionamento do ciclo onde foram testadas várias funcionalidades que estão descritas abaixo:

000010 0 20 // Armazena 20 na posição de memória 0

000010 1 10 // Armazena 10 na posição de memória 1

000011 0 // Adiciona o valor da posição 0 ao MBR (MBR += 20)

000100 1 // Subtrai o valor da posição 1 do MBR (MBR -= 10)

001111 2 // Armazena o valor do MBR na posição de memória 2

000010 3 25 // Armazena 25 na posição de memória 3

000101 3 // Multiplica o valor do MBR pelo valor na posição 3 (MBR *= 25)

000110 1 // Divide o valor do MBR pelo valor na posição 1 (MBR /= 10)

001010 // Calcula a raiz quadrada do MBR (MBR = sqrt(MBR))

001011 // Nega o valor do MBR (MBR = -MBR)

Resultados:

```
================================================================================
= OPÇÕES: =
================================================================================
1. INSERIR
2. VER INSTRUÇÕES
3. EXECUTAR
4. SAIR DO PROGRAMA
================================================================================
Escolha uma opção: 3
================================================================================
EXECUTANDO
================================================================================
CÁLCULO DO ENDEREÇO DA INSTRUÇÃO:
PC: 000001

BUSCANDO A INSTRUÇÃO:
IR <OPCODE>: 000010
IR <OP1>: 0
IR <OP2>: 20

DECODIFICANDO A INSTRUÇÃO:
 POS <- #DADO
0 <- 20

CÁLCULO DO ENDEREÇO DO OPERANDO:
Endereço: 0

BUSCANDO O OPERANDO NA POSIÇÃO:
MAR: 0

CÁLCULO DO ENDEREÇO DO SEGUNDO OPERANDO:
Endereço: 20

BUSCANDO O SEGUNDO OPERANDO NA POSIÇÃO:
MAR: 20

OPERAÇÃO DE DADOS:
ARMAZENANDO: 20
NA POSIÇÃO: 0

CALCULANDO ENDEREÇO DO OPERANDO:
ENDEREÇO: 0

ARMAZENANDO O OPERANDO:
MAR: 0
O VALOR FOI ARMAZENADO!
```

Figura 10

```
================================================================================
CÁLCULO DO ENDEREÇO DA INSTRUÇÃO:
PC: 000002

BUSCANDO A INSTRUÇÃO:
IR <OPCODE>: 000010
IR <OP1>: 1
IR <OP2>: 10

DECODIFICANDO A INSTRUÇÃO:
 POS <- #DADO
1 <- 10

CÁLCULO DO ENDEREÇO DO OPERANDO:
Endereço: 1

BUSCANDO O OPERANDO NA POSIÇÃO:
MAR: 1

CÁLCULO DO ENDEREÇO DO SEGUNDO OPERANDO:
Endereço: 10

BUSCANDO O SEGUNDO OPERANDO NA POSIÇÃO:
MAR: 10

OPERAÇÃO DE DADOS:
ARMAZENANDO: 10
NA POSIÇÃO: 1

CALCULANDO ENDEREÇO DO OPERANDO:
ENDEREÇO: 1

ARMAZENANDO O OPERANDO:
MAR: 1
O VALOR FOI ARMAZENADO!
================================================================================
CÁLCULO DO ENDEREÇO DA INSTRUÇÃO:
PC: 000003

BUSCANDO A INSTRUÇÃO:
IR <OPCODE>: 000011
IR <OP1>: 0
IR <OP2>:

DECODIFICANDO A INSTRUÇÃO:
  MBR <- MBR + #POS
MBR <- 20 + 0

CÁLCULO DO ENDEREÇO DO OPERANDO:
Endereço: 0

BUSCANDO O OPERANDO NA POSIÇÃO:
MAR: 0

OPERAÇÃO DE DADOS:
VALOR DO MBR: 20
VALOR DO CONTEÚDO NA POSIÇÃO: 20
VALOR DO MBR APÓS A OPERAÇÃO: 20 + 20 = 40
O VALOR FOI ARMAZENADO!
```

Figura 11

```
================================================================================
CÁLCULO DO ENDEREÇO DA INSTRUÇÃO:
PC: 000004

BUSCANDO A INSTRUÇÃO:
IR <OPCODE>: 000100
IR <OP1>: 1
IR <OP2>:

DECODIFICANDO A INSTRUÇÃO:
MBR <- MBR - #POS
10 <- 10 - 1

CÁLCULO DO ENDEREÇO DO OPERANDO:
Endereço: 1

BUSCANDO O OPERANDO NA POSIÇÃO:
MAR: 1

OPERAÇÃO DE DADOS:
VALOR DO MBR: 10
VALOR DO CONTEÚDO NA POSIÇÃO: 10
VALOR DO MBR APÓS A OPERAÇÃO: 10 - 10 = 0
O VALOR FOI ARMAZENADO!
================================================================================
CÁLCULO DO ENDEREÇO DA INSTRUÇÃO:
PC: 000005

BUSCANDO A INSTRUÇÃO:
IR <OPCODE>: 001111
IR <OP1>: 2
IR <OP2>:

DECODIFICANDO A INSTRUÇÃO:
#POS <- MBR
2 <- 10

CÁLCULO DO ENDEREÇO DO OPERANDO:
Endereço: 2

BUSCANDO O OPERANDO NA POSIÇÃO:
MAR: 2

OPERAÇÃO DE DADOS:
VALOR DO MBR: 10
VALOR DO ENDEREÇO APÓS A OPERAÇÃO: 10
O VALOR FOI ARMAZENADO!
```

Figura 12

```
================================================================================
CÁLCULO DO ENDEREÇO DA INSTRUÇÃO:
PC: 000006

BUSCANDO A INSTRUÇÃO:
IR <OPCODE>: 000010
IR <OP1>: 3
IR <OP2>: 25

DECODIFICANDO A INSTRUÇÃO:
 POS <- #DADO
3 <- 25

CÁLCULO DO ENDEREÇO DO OPERANDO:
Endereço: 3

BUSCANDO O OPERANDO NA POSIÇÃO:
MAR: 3

CÁLCULO DO ENDEREÇO DO SEGUNDO OPERANDO:
Endereço: 25

BUSCANDO O SEGUNDO OPERANDO NA POSIÇÃO:
MAR: 25

OPERAÇÃO DE DADOS:
ARMAZENANDO: 25
NA POSIÇÃO: 3

CALCULANDO ENDEREÇO DO OPERANDO:
ENDEREÇO: 3

ARMAZENANDO O OPERANDO:
MAR: 3
O VALOR FOI ARMAZENADO!
```

```
================================================================================
CÁLCULO DO ENDEREÇO DA INSTRUÇÃO:
PC: 000007

BUSCANDO A INSTRUÇÃO:
IR <OPCODE>: 000101
IR <OP1>: 3
IR <OP2>:

DECODIFICANDO A INSTRUÇÃO:
MBR <- MBR * #POS
250 <- 250 * 3

CÁLCULO DO ENDEREÇO DO OPERANDO:
Endereço: 3

BUSCANDO O OPERANDO NA POSIÇÃO:
MAR: 3

OPERAÇÃO DE DADOS:
VALOR DO MBR: 250
VALOR DO CONTEÚDO NA POSIÇÃO: 25
VALOR DO MBR APÓS A OPERAÇÃO: 250 * 25 = 6250
O VALOR FOI ARMAZENADO!
```

Figura 13

```
================================================================================
CÁLCULO DO ENDEREÇO DA INSTRUÇÃO:
PC: 000008

BUSCANDO A INSTRUÇÃO:
IR <OPCODE>: 000110
IR <OP1>: 1
IR <OP2>:

DECODIFICANDO A INSTRUÇÃO:
 MBR <- MBR / #POS
25 <- 25 / 1

CÁLCULO DO ENDEREÇO DO OPERANDO:
Endereço: 1

BUSCANDO O OPERANDO NA POSIÇÃO:
MAR: 1

OPERAÇÃO DE DADOS:
VALOR DO MBR: 25
VALOR DO CONTEÚDO NA POSIÇÃO: 10
VALOR DO MBR APÓS A OPERAÇÃO: 25 / 10 = 2
O VALOR FOI ARMAZENADO!
```

```
================================================================================
CÁLCULO DO ENDEREÇO DA INSTRUÇÃO:
PC: 000009

BUSCANDO A INSTRUÇÃO:
IR <OPCODE>: 001010
IR <OP1>:
IR <OP2>:

DECODIFICANDO A INSTRUÇÃO:
MBR <- sqrt(MBR)
5 <- sqrt(5)

CÁLCULO DO ENDEREÇO DO OPERANDO:
Endereço:

BUSCANDO O OPERANDO NA POSIÇÃO:
MAR:

OPERAÇÃO DE DADOS:
VALOR DO MBR: 5
VALOR DO MBR APÓS A OPERAÇÃO: sqrt(5) = 2
O VALOR FOI ARMAZENADO!
```

Figura 14

```
================================================================================
CÁLCULO DO ENDEREÇO DA INSTRUÇÃO:
PC: 000010

BUSCANDO A INSTRUÇÃO:
IR <OPCODE>: 001011
IR <OP1>:
IR <OP2>:

DECODIFICANDO A INSTRUÇÃO:
MBR <- -MBR
-5 <- --5

CÁLCULO DO ENDEREÇO DO OPERANDO:
Endereço:

BUSCANDO O OPERANDO NA POSIÇÃO:
MAR:

OPERAÇÃO DE DADOS:
VALOR DO MBR: -5
VALOR DO MBR APÓS A OPERAÇÃO: --5 = 5
O VALOR FOI ARMAZENADO!
```

Figura 15

Durante os testes foi visto que a instrução 001010 e 001011 não exibem o endereço do operando e o MAR, mas realizam suas respectivas operações corretamente. Portanto o simulador de ciclo de instrução cumpre seu objetivo principal, porém, está sujeito a futuras alterações para entregar todas as funcionalidades previstas de maneira correta.

CONCLUSÃO

Conclui-se que a aplicação prática de um simulador de ciclo de instrução se mostra essencial para demonstrar como esses ciclos são utilizados para avaliar o desempenho de diferentes componentes de hardware, como processadores, unidades de armazenamento e memória. No contexto da disciplina de Organização e Arquitetura de Computadores, os ciclos de instrução fornecem métricas padronizadas que permitem comparar o desempenho de diferentes dispositivos em termos de velocidade, eficiência, custo e outros aspectos discutidos nas aulas.

Dessa maneira, o programa desenvolvido permite ver de maneira detalhada o funcionamento interno de um processador, demonstrando como as instruções são buscadas, decodificadas e executadas. Os resultados obtidos durante a execução do simulador podem ser utilizados para futuras otimizações no código, visando realizar diferentes tipos de testes para obter outros resultados a partir de diferentes instruções para se adaptar a cada situação.

Portanto, apesar de necessitar de mais testes e verificações no código fonte, o simulador de ciclo de instrução serve como base para experimentos e otimização que podem ser aplicados em cenários reais de hardware e software, alinhando-se perfeitamente com os objetivos da disciplina de AOC.

.

REFERÊNCIAS

HARDZONE. Así es como tu CPU ejecuta las instrucciones que le da el software. Disponível em: <https://hardzone.es/tutoriales/rendimiento/ciclo-instruccion-cpu/>.

No title. Disponível em: <http://tics.ifsul.edu.br/matriz/conteudo/disciplinas/aoc/ub/4.html>.

Disponível em: <https://github.com/GuilhermeNakahata/UNESPAR-2024/blob/main/Arquitetura%20e%20Organizacao%20de%20Computadores/1o%20Bimestre/Aulas/Aulas_ArquiteturaComputadores_Revisao.pdf>.