# Trabalho prático de Arquitetura de Computadores

**Autor 1:** Renan Catini Amaral
**Autor 2**: Luis Renato Goulart
**Professor:** Eliseu César Miguel

---

## 1. Introdução

### 1.1 Objetivo do Projeto

Neste trabalho, foi implementado um processador multiciclo de 16 bits, com instruções de tamanho fixo, com capacidade de fazer operações aritméticas simples com 8 registradores.

A implementação foi feita na linguagem de descrição de Hardware (HDL) Verilog, com a finalidade de aprender a utilizar novas ferramentas para essa finalidade.

### 1.2 Especificações Atendidas

O processador implementado atende às especificações definidas no trabalho, incluindo:

- **Arquitetura**: Processador multiciclo que executa uma instrução a cada 4 ciclos de clock.
- **Largura de dados:** Cada instrução tem tamanho fixo de 16 bits para todas as operações definidas, registradores e imediatos.
- **Banco de registradores:** Foram definidos 8 registradores (R0-R7), de 16 bits cada.
- **Conjunto de instruções:** Instruções definidas em três formatos distintos.
- **ULA:** Implementa todas as operações aritméticas que foram definidas pelo trabalho (Soma, subtração e porta NAND).

# 2. Decisões de Arquitetura

## 2.1 Organização dos Módulos

Para a implementação, adotamos uma  abordagem modular, sendo o componente principal, ‘processor.v’, que é responsável por integrar todos os outros módulos do sistema. Entre eles está o módulo “Unidade de Controle”, que internamente instancia internamente dois submódulos próprios, que são o decodificador de instrução e o decodificador de registrador, únicos módulos que foram definidos fora do arquivo do componente principal do processador.

A abordagem escolhida foi decidida para facilitar a manutenção e depuração , permitindo que cada módulo seja desenvolvido e testado de forma isolada, com a finalidade de entendermos cada módulo de maneira mais simples e objetiva. Além disso, essa abordagem auxilia na reutilização de componentes em outros projetos ou em futuras versões do processador.

## 2.2 Unidade de Controle

Para a implementação da Unidade de Controle, foi utilizada a lógica combinacional baseada em estados com contador de 2 bits externo. Esse contador gera 4 estados distintos (de 0 a 3), permitindo que cada ciclo de execução seja bem definido e separado. A unidade de controle utiliza esses estados para ativar os sinais corretos a cada ciclo, guiando o processador em cada instrução.

Essa abordagem segue o modelo de uma arquitetura multiciclo, em que cada instrução é dividida em vários ciclos de clock. Isso permite uma execução mais flexível e otimizada, já que diferentes instruções podem reutilizar os mesmos ciclos, fazendo a Unidade de Controle gerar apenas o necessário para o estado atual.

Cada ciclo foi definido como:

- **Ciclo 0:** Decodificação da instrução.
- **Ciclo 1:** Seleção e carregamento do primeiro operando.
- **Ciclo 2:** Seleção do segundo operando e execução da operação.
- **Ciclo 3:** Writeback do resultado no registrador destino.

### 2.3 Multiplexador

O multiplexador responsável pela  seleção entre os registrador e o valor imediato foi implementado com 10 entradas: oito correspondem aos registradores, uma ao registrador R e a última ao valor imediato). O controle dessas entradas é feito a partir se um sinal seletor de 4 bits.

A lógica de seleção foi feita para aproveitar diretamente os códigos dos registradores fornecidos pela instrução. Para isso, o código de cada registrador é expandido com um bit 0 a esquerda. Já para as opções que não vem direto da instrução, o registrador R e o valor imediato,  foram atribuídos códigos exclusivos: 1000 para o imediato e 1001 para o R.

Essa implementação foi escolhida por simplificar o controle por parte da Unidade de Controle, para permitir que o sinal de controle seja gerado de forma direta e lógica a partir dos bits da instrução, sem que precise de lógica adicional ou decodificação complexa.

### 2.4 Registradores

Para a implementação dos registradores foi definido apenas um único módulo, qual padroniza todos os registradores, e declaramos ao registradores R0-R7, A e R no módulo principal do processador.

---

## 3. Implementação dos Módulos

### 3.1 Contador (counter.v)

**Função:** Gera os 4 estados da máquina do processador, ou seja, ciclos 0 a 3.
**Sinais:**

- **clock:** Clock do sistema
- **clear:** Reset assíncrono do contador.
- **out [1:0]:** Estado atual (2 bits)

**Comportamento:** Contador binário de 2 bits que incrementa a cada ciclo de clock e pode ser reiniciado através do sinal *clear.*

## 3.2 ULA (ULA.v)

**Operações implementadas:** Soma (ADD), Subtração (SUB), NAND e Transferência (REP).
**Códigos de controle:**

**3'b000**: Soma
**3'b001**: Subtração
**3'b010**: NAND
**3'b111**: Transferência (REP)

## 3.3 Decodificador (decodificador.v)

**Função:** Extrai os campos da instrução de 16 bits.
**Campos extraídos:** Opcode, RX e RY
**Implementação:** Utiliza assigns simples para extrair os bits correspondentes da instrução.

## 3.4 Decodificador de Registrador

**Função:** Converte o número do registrador (3 bits) em sinal de habilitação one-hot (8 bits).
**Mapeamento:**

*000 -> 10000000 (R0)*
*001 -> 01000000 (R1)*
*010 -> 00100000 (R2)*
*011 -> 00010000 (R3)*
*100 -> 00001000 (R4)*
*101 -> 00000100 (R5)*
*110 -> 00000010 (R6)*
*111 -> 00000001 (R7)*

## 3.5 Extensor de sinal

**Função:** Extende o valor imediato de 10 bits para 16 bits.
**Implementação:** Concatena 6 bits zero com os 10 bits menos significativos da instrução: {6'b000000, instrucao[9:0]}

### 3.5 Registrador

**Função:** Implementa um registrador de 16 bits com habilitação.
**Sinais:**

**clock**: Clock do sistema
**enable**: Habilita escrita no registrador
**entrada[15:0]**: Dados de entrada
**saida[15:0]**: Dados de saída

---

# 4. Fluxo de Execução

## 4.1 Ciclo 1 - Decodificação

Neste primeiro ciclo, a instrução é decodificada através do módulo *decodificador,* extraindo os campos de Opcode, RX e RY. A unidade de controle prepara os sinais para os próximos ciclos, mas não executa nenhuma operação efetiva sobre os dados.

## 4.2 Ciclo 2 - Primeiro Operando

Para as instruções aritméticas e lógicas (ADD, SUB, NAN), o primeiro operando (registrador RX) é selecionado pelo multiplexador e carregado no registrador auxiliar A. Para a instrução OUT, o registrador RX é preparado para ser enviado ao barramento.

## 4.3 Ciclo 3 - Segundo Operando e Execução

- **ADD/SUB/NAN:** O segundo operando (RY), é selecionado, a ULA executa a operação correspondente e o resultado é armazenado em R.
- **REP:** O registrador RY é transferido através da ULA para o registrador R.
- **LDI:** O valor imediato, é transferido para o registrador R.
- **OUT:** O registrador RX é enviado para o barramento de saída.

### 4.4 Ciclo 4 - Writeback

Para instruções que modificam registradores (ADD, SUB, NAN, REP, LDI), o resultado armazenado no registrador R é escrito no registrador destino RX. A instrução OUT continua enviando dados para o barramento.

---

# 5. Testes Realizados

## 5.1 Teste 1 – Operação básica com imediato

**Objetivo:** Testar o carregamento de imediato e operação.
**Programa usado:**

Instrução 1: ldi r0, #28 -> 101_000_0000011100
Instrução 2: ldi r0, #10 -> 101_001_0000001010
Instrução 3: sub r0, r1 -> 001_000_001_0000000
Instrução 4: out r0 -> 100_000_0000000000

**Resultado esperado:**

R0 deve conter 28 após a primeira instrução.
R1 deve conter 1.º após a segunda instrução.
R0 deve conter 18 (28-10) após a terceira instrução.
Barramento deve mostrar 18 na quarta instrução.

**Status:** Passou, analisado na Figura 1.
**Observações:** Teste principal dado no testbench.

## 5.2 Teste 2 - Operação lógica "nan" e uso de registradores

**Objetivo:** Testar a instrução lógica *nan* entre dois registradores, transferência e saída via registrador intermediário.
**Programa usado:**

Instrução 1: ldi r2, #15 → 101_010_0000001111
Instrução 2: ldi r3, #7 → 101_011_0000000111

Instrução 3: nan r2, r3 → 010_010_011_0000000
Instrução 4: rep r5, r2 → 111_101_010_0000000
Instrução 5: out r5 → 100_101_0000000000

**Resultado esperado:**

R2 deve conter 15 após a primeira instrução.
R3 deve conter 7 após a segunda instrução.
R2 deve conter o resultado de ~(R2 & R3) = 1111 1111 1111 1000 = 65528 após a terceira instrução.
R5 deve copiar o valor de R2 (65528) na quarta instrução.
Barramento deve mostrar 65528 na quinta instrução.

**Status:** Passou, analisado na Figura 2.
**Observações:** O teste demonstra corretamente a operação *nan* com registradores de 16 bits e a manipulação do valor resultante até a saída.

## 5.3 Teste 3 - Operações aritméticas com registradores e imediato

**Objetivo:** estar instruções de soma e subtração entre registradores após carregamento de valores imediatos, com saída final.

**Programa usado:**

Instrução 1: ldi r1, #20 → 101_001_0000010100
Instrução 2: ldi r4, #8 → 101_100_0000001000
Instrução 3: add r1, r4 → 000_001_100_0000000
Instrução 4: ldi r6, #5 → 101_110_0000000101
Instrução 5: sub r1, r6 → 001_001_110_0000000
Instrução 6: rep r7, r1 → 111_111_001_0000000
Instrução 7: out r7 → 100_111_0000000000

**Resultado esperado:**

R1 deve conter 20 após a primeira instrução.
R4 deve conter 8 após a segunda instrução.
R1 deve conter 28 (20 + 8) após a terceira instrução.
R6 deve conter 5 após a quarta instrução.
R1 deve conter 23 (28 − 5) após a quinta instrução.
R7 deve copiar o valor de R1 (23) na sexta instrução.
Barramento deve mostrar **23** na sétima instrução.

**Status:** Passou, analisado na Figura 3.
**Observações:** Teste demonstra corretamente a execução de soma e subtração entre registradores com imediatos e o fluxo do valor até a saída.

---

# 6. Conclusão

O projeto atingiu seus objetivos principais, implementando com sucesso um processador multiciclo de 16 bits capaz de executar as instruções especificadas. A arquitetura modular facilitou o desenvolvimento e depuração, enquanto a implementação multiciclo proporcionou uma base sólida para futuras expansões. O processador demonstra compreensão adequada dos conceitos fundamentais de arquitetura de computadores.

---

# 7. Anexos

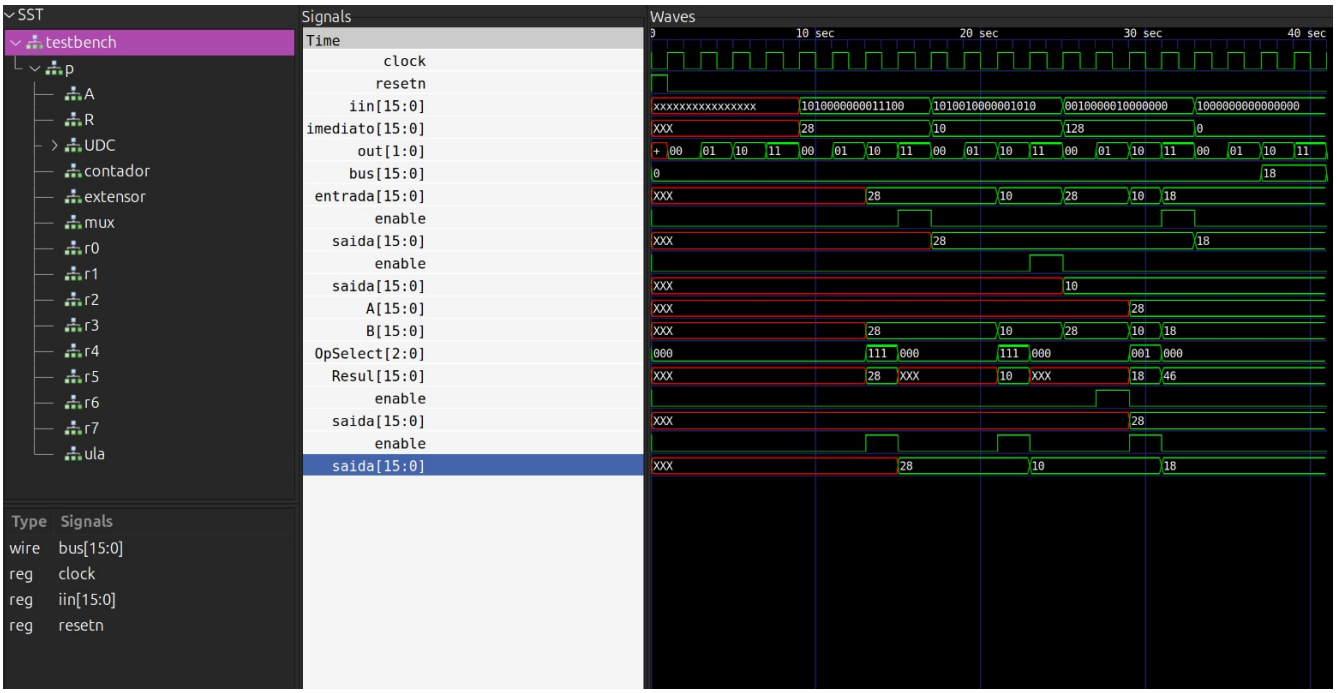

*Figura 1: Referência ao primeiro teste.*

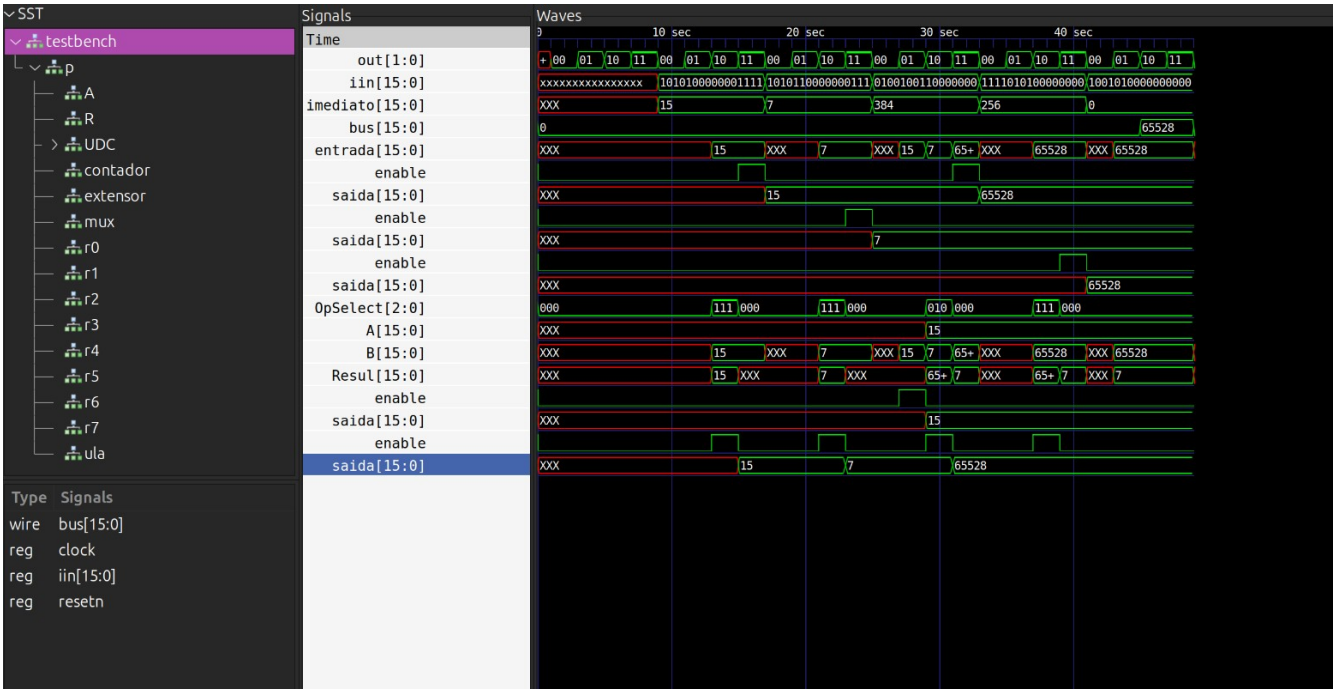

*Figura 2: Referência do segundo teste.*

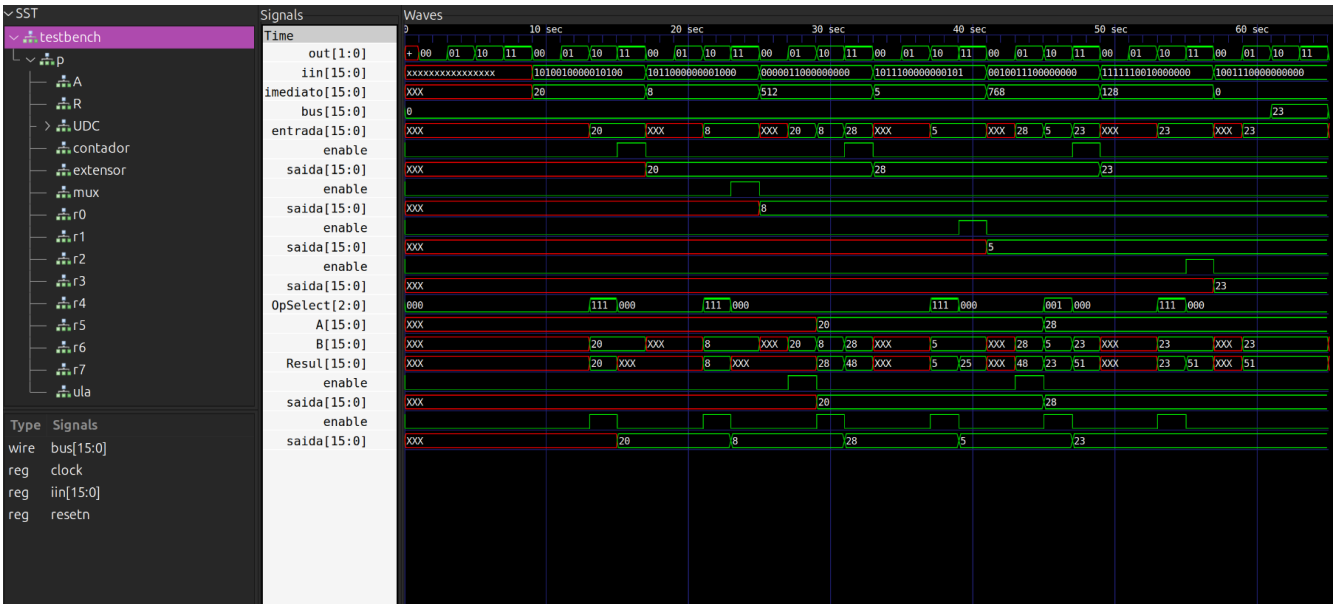


*Figura 3: Referência ao terceiro teste.*