

## Construindo a CPU: Parte 1

Yuri Kaszubowski Lopes

UDESC

YKL (UDESC)

Construindo a CPU: Parte 1

1 / 20

Anotações

## Implementação básica

- Vamos construir um processador MIPS básico, que implementa o seguinte
  - ▶ Instruções de referência a memória `sw` e `lw`
  - ▶ Instruções lógicas e aritméticas tipo-R: `add`, `sub`, `and`, `or` e `slt`
  - ▶ Instruções lógicas e aritméticas tipo-I: `addi`, `andi` e `ori`
  - ▶ Instruções de desvio `beq` e `j`
- Seguiremos a Implementação descrita no livro base da disciplina
  - ▶ D. Patterson; J. Henessy. **Organização e Projeto de Computadores: Interface Hardware/Software**. 5a Edição. Elsevier Brasil, 2017.

YKL (UDESC)

Construindo a CPU: Parte 1

2 / 20

Anotações

## O formato das instruções

- Até o momento aprendemos os formatos básicos de instruções MIPS:
  - ▶ Tipo R
  - ▶ Tipo I
  - ▶ Tipo J
- Devemos considerar:
  - ▶ Tamanho das instruções (sempre 32 bits no MIPS)
  - ▶ Tamanho do opcode
  - ▶ Outros campos conforme o tipo
- Veremos que isso impacta diretamente no hardware
  - ▶ O conjunto de instruções e a forma com que o hardware é implementado são diretamente relacionados

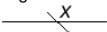

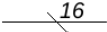
YKL (UDESC)

Construindo a CPU: Parte 1

3 / 20

Anotações

## Convenções Iniciais

- O tamanho da palavra no MIPS é de 32 bits
  - ▶ Assumiremos que as linhas de conexão no nosso projeto são na verdade um barramento de 32 bits
    - ★ Como se tivéssemos 32 fios em paralelo
  - ▶ Caso o barramento tenha uma largura diferente de 32, isso será indicado da seguinte forma (onde x indica a largura do barramento):  
  
A horizontal line with an 'x' above it and a diagonal slash.
  - ▶ Exemplos:
    - ★ Um barramento de 32 bits:  
  
A horizontal line with '32' above it and a diagonal slash.
    - ★ Um barramento de 16 bits:  
  
A horizontal line with '16' above it and a diagonal slash.
  - ▶ Para sinais de controle (em azul) um único fio para um único bit

### Anotações

---

---

---

---

---

---

---

---

## O Caminho de Dados

- Vamos analisar os principais componentes que precisamos para executar um programa
  - ▶ Uma memória, para **armazenar as instruções** do nosso programa
    - ★ Entrada: O endereço da instrução desejada
    - ★ Saída: A instrução no endereço apontado
  - ▶ Um registrador que vai armazenar o endereço da próxima instrução a ser executada: Registrador PC no MIPS
    - ★ Entrada: O próximo endereço a ser armazenado no registrador
    - ★ Saída: O endereço corrente

### Anotações

---

---

---

---

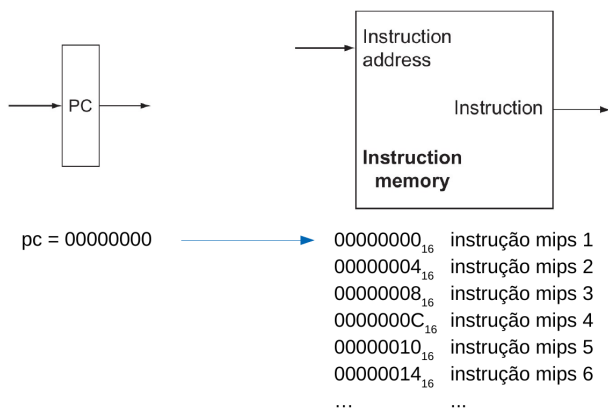
---

---

---

---

## O Caminho de Dados



### Anotações

---

---

---

---

---

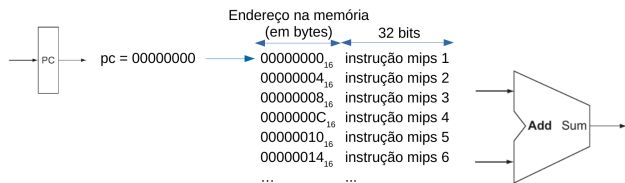
---

---

---

## Incrementando PC

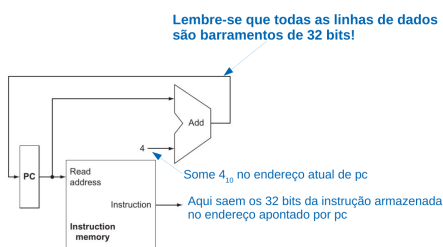
- Caso não tenhamos desvios, após a execução da instrução atual, devemos executar a próxima instrução
  - ▶ O que devemos fazer com o PC? Como?
  - ▶ Precisamos adicionar 4 no valor atual do pc
    - \* Memória endereçada em bytes, logo "saltamos" 4 bytes = 32 bits
  - ▶ Vamos precisar de um Somador
    - \* Podemos, por exemplo, utilizar o circuito somador estudado em SID (ineficiente, mas funciona)



Símbolo geral utilizado para uma Unidade Lógica Aritmética (ULA ou ALU). Nesse caso está explicitado que a unidade deve realizar somas.

### Anotações

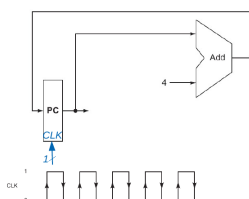
## PC e Memória de Instrução



### Anotações

## Sincronização

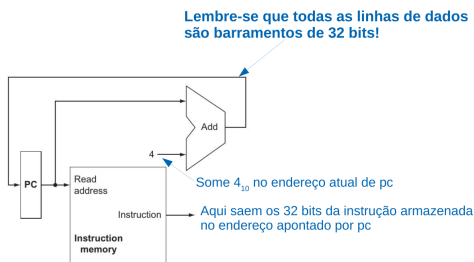
- Sinais de clock são adicionados nos elementos de estado (sequenciais)
  - ▶ e.g., O registrador pc só vai ler a entrada na transição de clock
- Enquanto não há transição, o valor antigo de PC é enviado para a saída, e lido pelo adder
- Nos nossos circuitos, o **sinal de clock vai ser omitido para simplificar o raciocínio**
  - ▶ Mas sempre assuma que existe sincronismo, para que o sinal anterior não seja "atropelado" pelo próximo



### Anotações

## PC e Memória de Instrução

- O “loop” principal está pronto
  - ▶ Sempre enviamos a instrução para execução, e incrementamos pc em 4 para apontar para a próxima instrução
  - ▶ O que é feito com a instrução agora depende do seu formato
    - ★ Vamos começar com instruções básicas do tipo-R
  - ▶ Como são as instruções do tipo-R?



### Anotações

---

---

---

---

---

---

---

---

## Instruções do tipo-R

- Exemplo:  

```
1 add $a0, $a1, $a2
```
- Leem dois registradores, executam uma operação aritmética em uma ALU (soma, subtração, deslocamento,...), e armazenam o resultado em um terceiro registrador

### Anotações

---

---

---

---

---

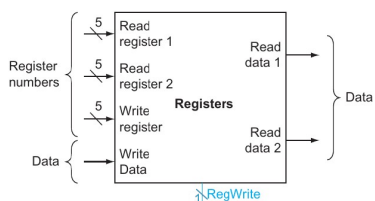
---

---

---

## Banco de Registradores

- Precisamos de uma estrutura denominada banco de registradores
  - ▶ Vai conter todos os 32 registradores do MIPS
  - ▶ Entradas:
    - ★ Endereço do registrador de leitura 1
    - ★ Endereço do registrador de leitura 2
    - ★ Endereço do registrador de escrita
    - ★ Dados a serem escritos no registrador de escrita
  - ▶ Saídas:
    - ★ Dados do registrador de leitura 1
    - ★ Dados do registrador de leitura 2
  - ▶ Qual o tamanho de cada um dos barramentos de entrada e saída no banco de registradores?



### Anotações

---

---

---

---

---

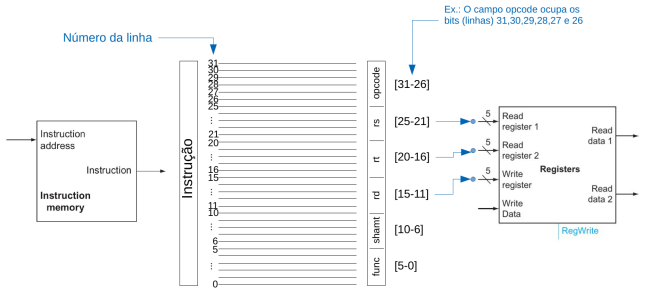
---

---

---

Fonte dos bits

- Como definimos, uma linha sem marcação tem largura de 32 bits
- Então na verdade são 32 linhas (barramento), endereçadas de 0 a 31
- Conforme o livro, vamos seguir uma abordagem little-endian, onde a instrução fica “ao contrário” no endereçamento



Anotações

---

---

---

---

---

---

---

---

ALU

- Além de precisar dos registradores, as instruções precisam executar a operação com esses registradores
  - ▶ Soma, subtração, deslocamento, lógicas, ...
- Vamos utilizar uma ALU geral para isso
  - ▶ Arithmetic Logic Unit
  - ▶ Entradas:
    - \* Dois valores de 32 bits
    - \* Uma entrada ALUop de 4 bits, que define qual a operação deve ser realizada com os valores
  - ▶ Saídas:
    - \* Uma saída de 32 bits com o resultado da operação
    - \* Uma saída de 1 bit indicando se o resultado da operação foi zero ( Essa saída simplificará a construção do nosso circuito em breve)

Anotações

---

---

---

---

---

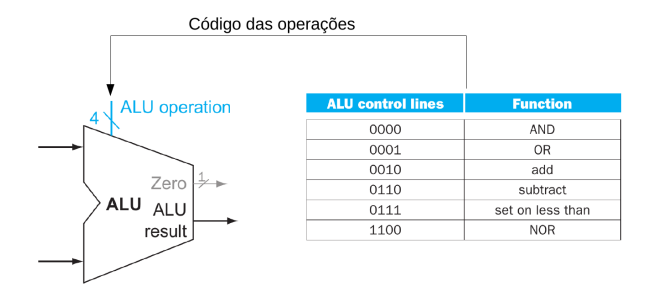
---

---

---

ALU

- Dentro da ALU podemos ter diversos circuitos especialistas
  - ▶ Somadores
  - ▶ Operadores Lógicos
- Qual desses circuitos será ativado depende do ALUOp



Anotações

---

---

---

---

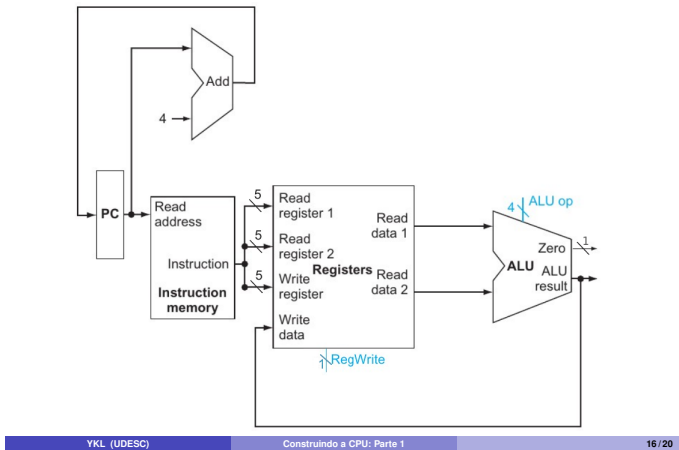
---

---

---

---

## Ligando os Componentes



### Anotações

---

---

---

---

---

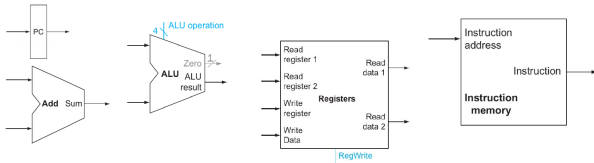
---

---

---

## Exercícios

- Sem olhar os slides anteriores, utilizando os seguintes componentes, monte novamente o processador capaz de executar instruções do tipo R básicas. Marque no circuito a largura de cada barramento. Caso o barramento utilize somente algumas linhas de outro (e.g., número do registrador de entrada), indique quais as linhas que são utilizadas por ele através da notação [n-m]



### Anotações

---

---

---

---

---

---

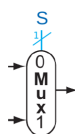
---

---

- Os 4 bits que informam a ALU sobre qual a operação que deve ser executada podem vir de onde? Não precisa ligar no circuito, mas dê suas ideias sobre como poderíamos definir isso.

## Exercícios

- A seguir é dado um multiplexador, com duas entradas de 32 bits e uma saída de 32 bits. A entrada superior é enviada para a saída se o bit S=0, caso contrário, a entrada inferior é enviada para a saída. Compare as instruções do tipo R com instruções de load/store (l.w, s.w) (tipo I). Quais registradores (fonte 1, fonte 2 e destino) podem mudar de acordo com o tipo da instrução? Adicione o multiplexador no seu circuito para resolver esse problema. No momento não precisa se preocupar em onde ligar a entrada S.



### Anotações

---

---

---

---

---

---

---

---

Referências

- D. Patterson; J. Henessy. **Organização e Projeto de Computadores: Interface Hardware/Software**. 5a Edição. Elsevier Brasil, 2017.
- Andrew S. Tanenbaum. **Organização estruturada de computadores**. 5. ed. São Paulo: Pearson, 2007.
- Harris, D. and Harris, S. **Digital Design and Computer Architecture**. 2a ed. 2012.
- [courses.missouristate.edu/KenVollmar/mars/](https://courses.missouristate.edu/KenVollmar/mars/)

Anotações

---

---

---

---

---

---

---

---

Anotações

---

---

---

---

---

---

---

---

Anotações

---

---

---

---

---

---

---

---