> “Ao contrário”, continuou Tweedledee, “se foi assim, poderia ser; e se fosse assim, seria; mas como não é, então não é. Isso é Lógico” (Lewis Carroll, Through the Looking-glass: And what Alice Found There, 1875).

Controle da CPU Monociclo

Paulo Ricardo Lisboa de Almeida

DSBD

UFPR

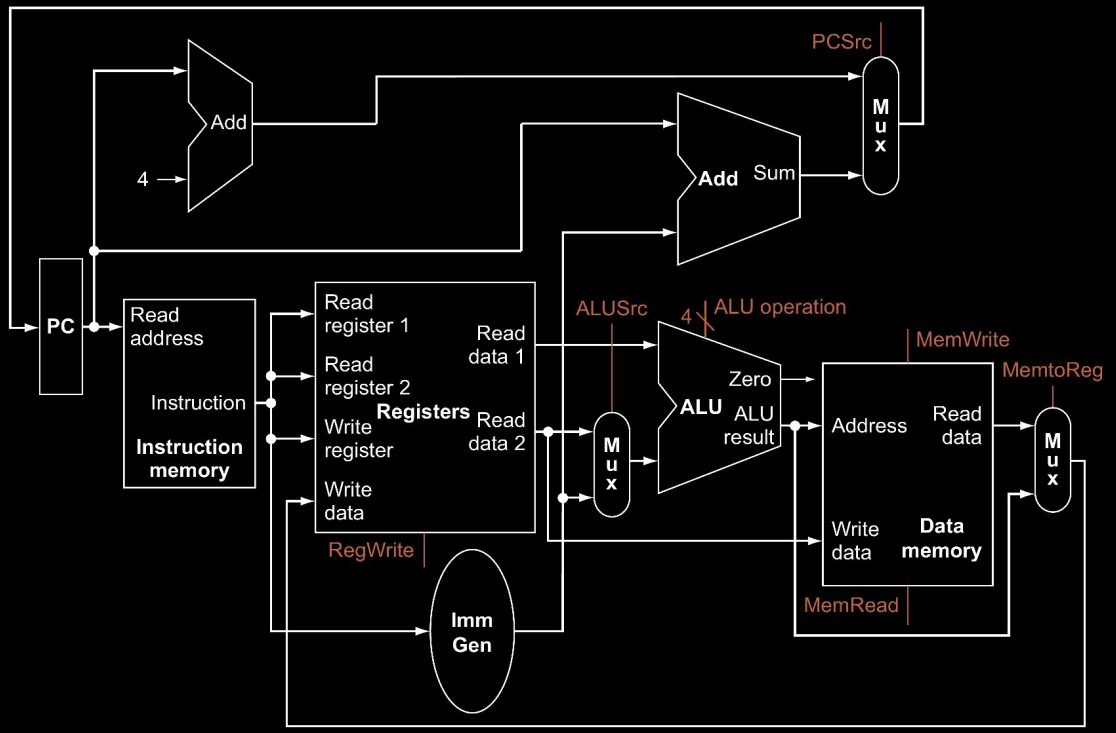
Recapitulando...

Diversos sinais de controle não ligados.

AluSrc, PCSrc, ALU Operation, ...

Determinam o comportamento de cada unidade.

Dependem diretamente das instruções.



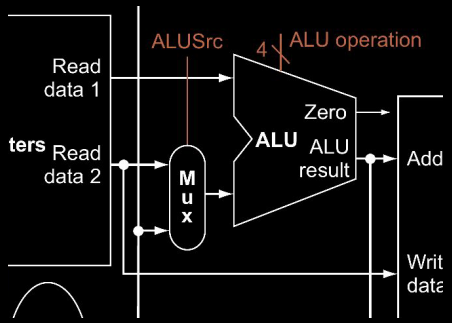
Controle da ALU

Começando com uma unidade de controle para a ALU.

Sinal ALU Operation de 4 bits.

Os sinais são:

Sinal	Função da ALU
0000	AND
0001	OR
0010	Soma
0110	Subtração



Controle da ALU

O sinal enviado para a ALU vai depender da instrução.

`lw` e `sw`?

`beq`?

Instruções do tipo-R?

Sinal	Função da ALU
0000	AND
0001	OR
0010	Soma
0110	Subtração

Controle da ALU

O sinal enviado para a ALU vai depender da instrução.

`lw` e `sw` precisam calcular o endereço através de uma adição (0010_2).

Um `beq` precisa realizar uma subtração (0110_2) para verificar se os valores são iguais.

Instruções do tipo-R têm a operação definida pelos campos funct7 (bits [31:25]) e funct3 (bits [14:12]).

Sinal	Função da ALU
0000	AND
0001	OR
0010	Soma
0110	Subtração

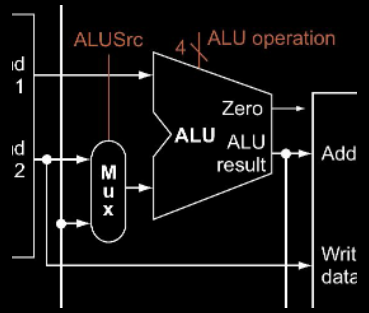
Controle da ALU

A **unidade de controle da ALU** vai receber como entrada um sinal de 2 bits, chamado **ALUOp**, que vai definir o tipo da instrução, e também vai receber o sinal dos campos funct7 e funct3.

ALUOp

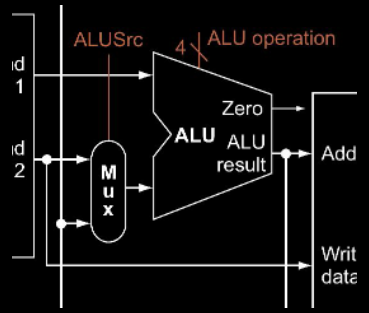
- 00_2 -› indica que a operação é um add (para loads e stores).
- 01_2 -› indica que a operação é um subtract (para beq).
- 10_2 -› indica que a operação vai ser definida pelo campo funct7 e funct3.

Controle da ALU – Tabela verdade



Opcode	ALUOp	Instrução	Func7	Func3	Operação da ALU	ALU Operation
lw	00	load word	XXX XXXX	XXX	Soma	0010
sw	00	store word	XXX XXXX	XXX	Soma	0010
beq	01	branch if equal	XXX XXXX	XXX	Subtração	0110
Tipo-R	10	add	000 0000	000	Soma	0010
Tipo-R	10	subtract	010 0010	000	Subtração	0110
Tipo-R	10	and	000 0000	111	and	0000
Tipo-R	10	or	000 0000	110	or	0001

Controle da ALU – Tabela verdade



Opcode	ALUOp	Instrução	Func7	Func3	Operação da ALU	ALU Operation
lw	00	load word	XXX XXXX	XXX	Soma	0010
sw	00	store word	XXX XXXX	XXX	Soma	0010
beq	01	branch if equal	XXX XXXX	XXX	Subtração	0110
Tipo-R	10	add	000 0000	000	Soma	0010
Tipo-R	10	subtract	010 0010	000	Subtração	0110
Tipo-R	10	and	000 0000	111	and	0000
Tipo-R	10	or	000 0000	110	or	0001

Entradas.

Saída. Para simplificar, a tabela contém somente as entradas para as quais nos importamos com a saída.

Controle da ALU

Dada a tabela verdade, podemos agora construir o Controle da ALU.

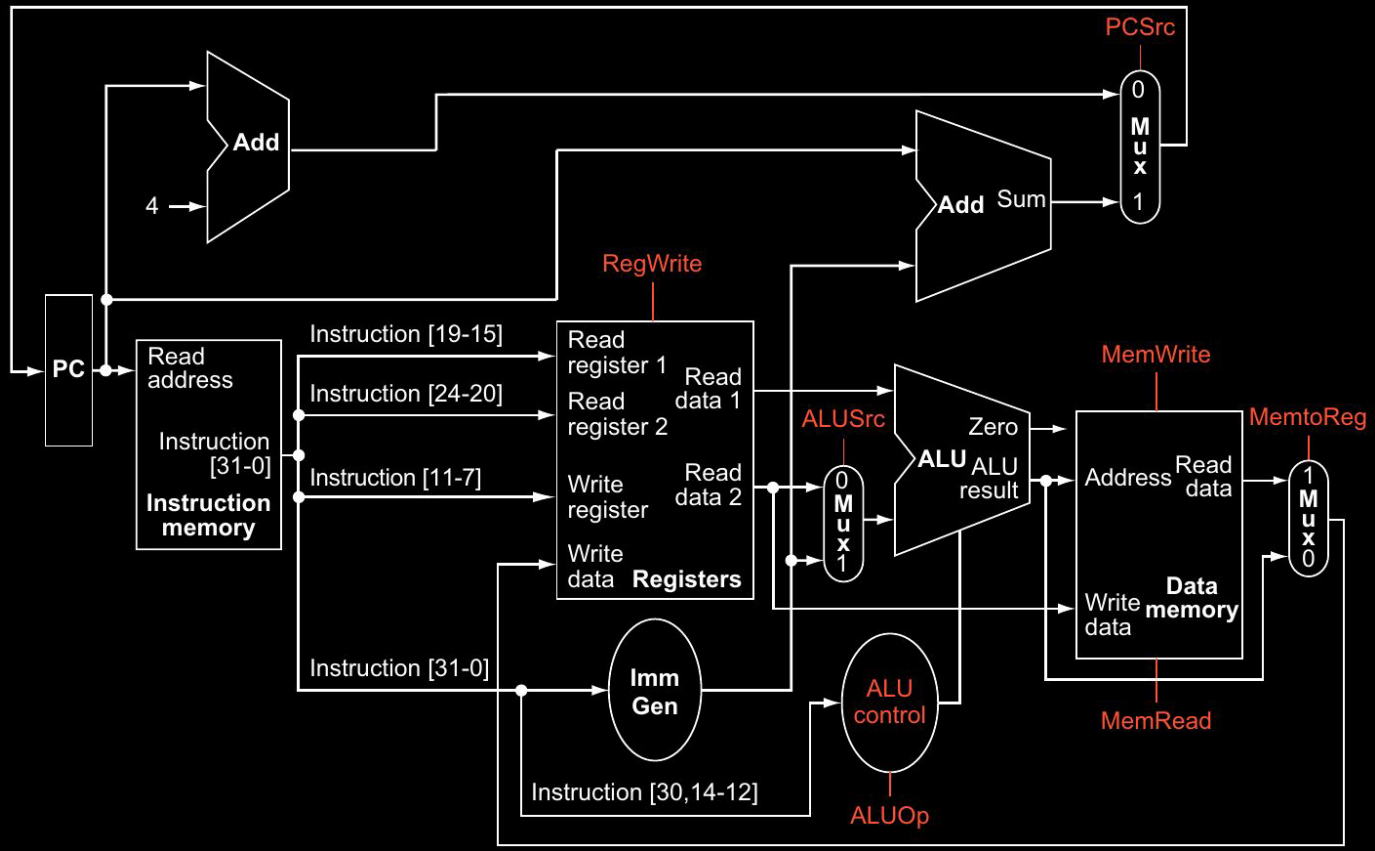
Definir a expressão Booleana para a ALU, simplificá-la, e implementá-la com portas lógicas.

Exemplo: utilizar soma dos produtos, e então simplificar a expressão lógica utilizando Álgebra de Boole.

Controle da ALU

O controle da ALU gera o sinal ALU Operation de 4 bits.

Depende de um sinal ALUOp de 2 bits.

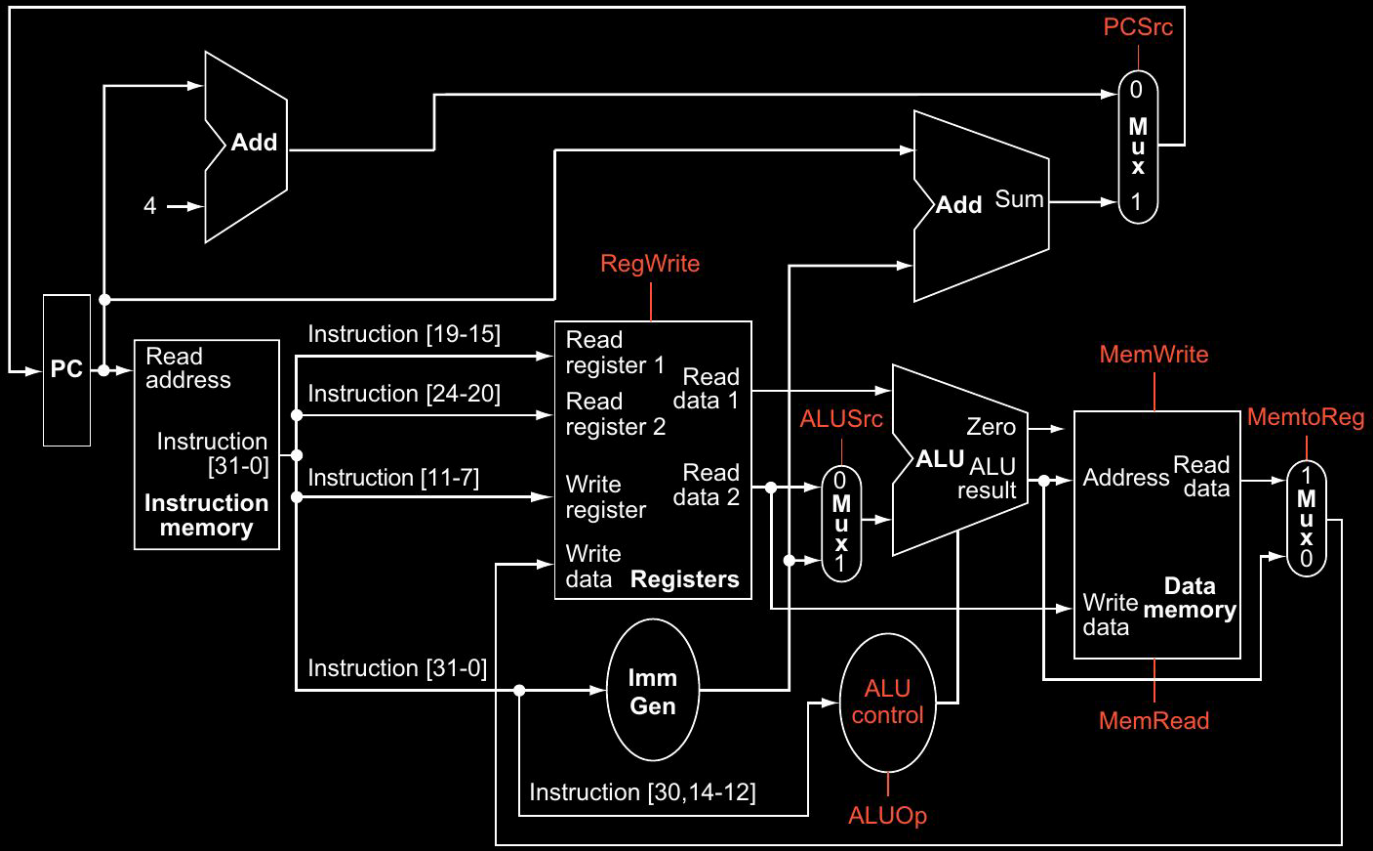


Controle da ALU

O controle da ALU gera o sinal ALU Operation de 4 bits.

Depende de um sinal ALUOp de 2 bits.

Parece que trocamos um problema por outro!



Controle da ALU

O sinal ALUOp vai ser gerado pela **unidade de controle principal**.

Múltiplos níveis de unidades de controle.

Controle da ALU

O sinal ALUOp vai ser gerado pela **unidade de controle principal**.

Múltiplos níveis de unidades de controle.

- Mais simples projetar.
- Possível redução no tamanho do circuito.
- Possível aumento de velocidade.

Unidades mais simples processam a informação mais rapidamente do que uma única unidade grande.

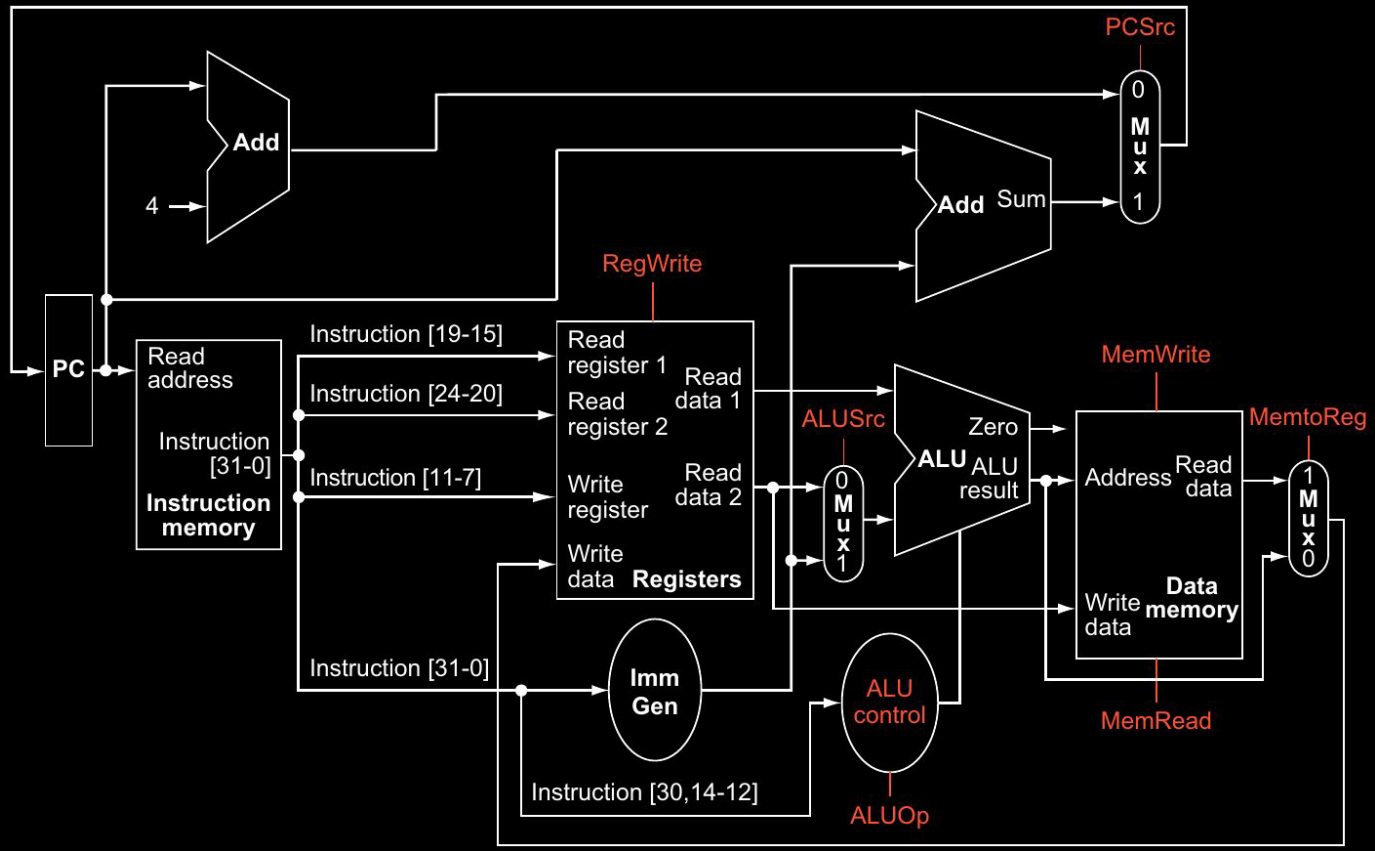
Especialmente útil quando adicionarmos um pipeline.

Redução do período do clock.

Unidade de Controle Principal

Resta criar a **unidade de controle principal**.

Vai cuidar das 6 linhas de controle de 1 bit, e do sinal de controle de dois bits ALUOp.



Faça você mesmo

Verifique os sinais no circuito, e escreva na tabela o que se espera quando cada um dos sinais é 0 ou 1.

Sinal	Efeito Esperado quando 0	Efeito Esperado quando 1
MemToReg	O valor a ser escrito no registrador destino vem da ALU.	O valor a ser escrito no registrador destino vem da memória de dados.
RegWrite		
ALUSrc		
PCSrc		
MemRead		
MemWrite		

Faça você mesmo

Verifique os sinais no circuito, e escreva na tabela o que se espera quando cada um dos sinais é 0 ou 1.

Sinal	Efeito Esperado quando 0	Efeito Esperado quando 1
MemToReg	O valor a ser escrito no registrador destino vem da ALU.	O valor a ser escrito no registrador destino vem da memória de dados.
RegWrite	Nada a fazer.	O registrador de destino é escrito com os dados de *write data*.
ALUSrc	O segundo operando da ALU vem da saída do registrador.	O segundo operando da ALU vem do *Imm Gen*.
PCSrc	PC = PC + 4.	PC = PC + deslocamento calculado pelo branch.
MemRead	Nada a fazer.	Memória é lida a partir do endereço de entrada.
MemWrite	Nada a fazer.	Memória é escrita a partir do endereço de entrada. Dado escrito é *write data*.

Unidade de Controle Principal

A Unidade de Controle Principal pode ter seus comportamentos definidos pelo opcode.

Instr.	Opcode							ALUSrc	MemToReg	RegWrite	MemRead	MemWrite	Branch	ALUOp1	ALUOp0
	Op6	Op5	Op4	Op3	Op2	Op1	Op0								
Tipo-R	0	1	1	0	0	1	1	0	0	1	0	0	0	1	0
lw	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0
SW	0	1	0	0	0	1	1	1	X	0	0	1	0	0	0
BEQ	1	1	0	0	0	1	1	0	X	0	0	0	1	0	1

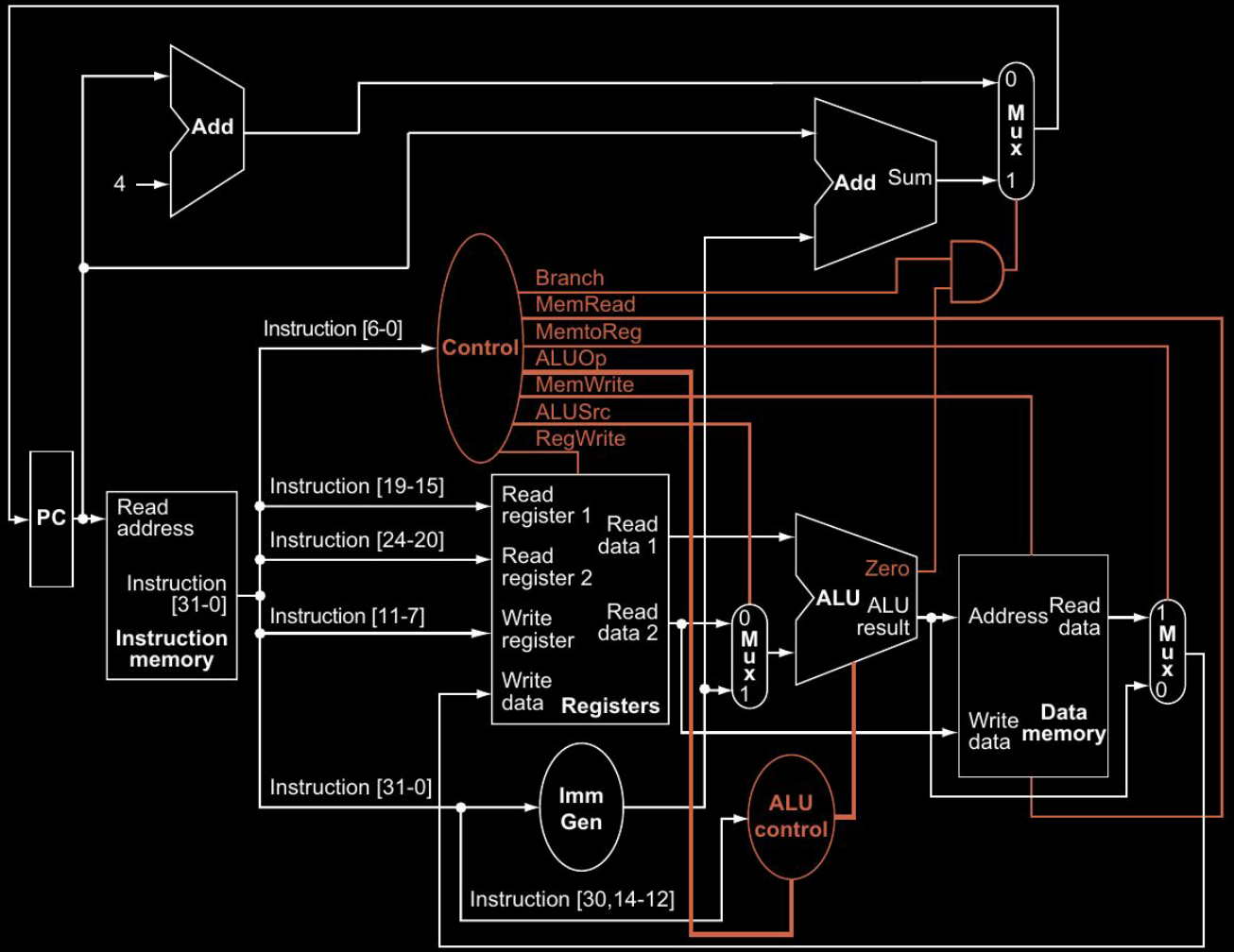
Unidade de Controle Principal

A Unidade de Controle Principal pode ter seus comportamentos definidos pelo opcode.

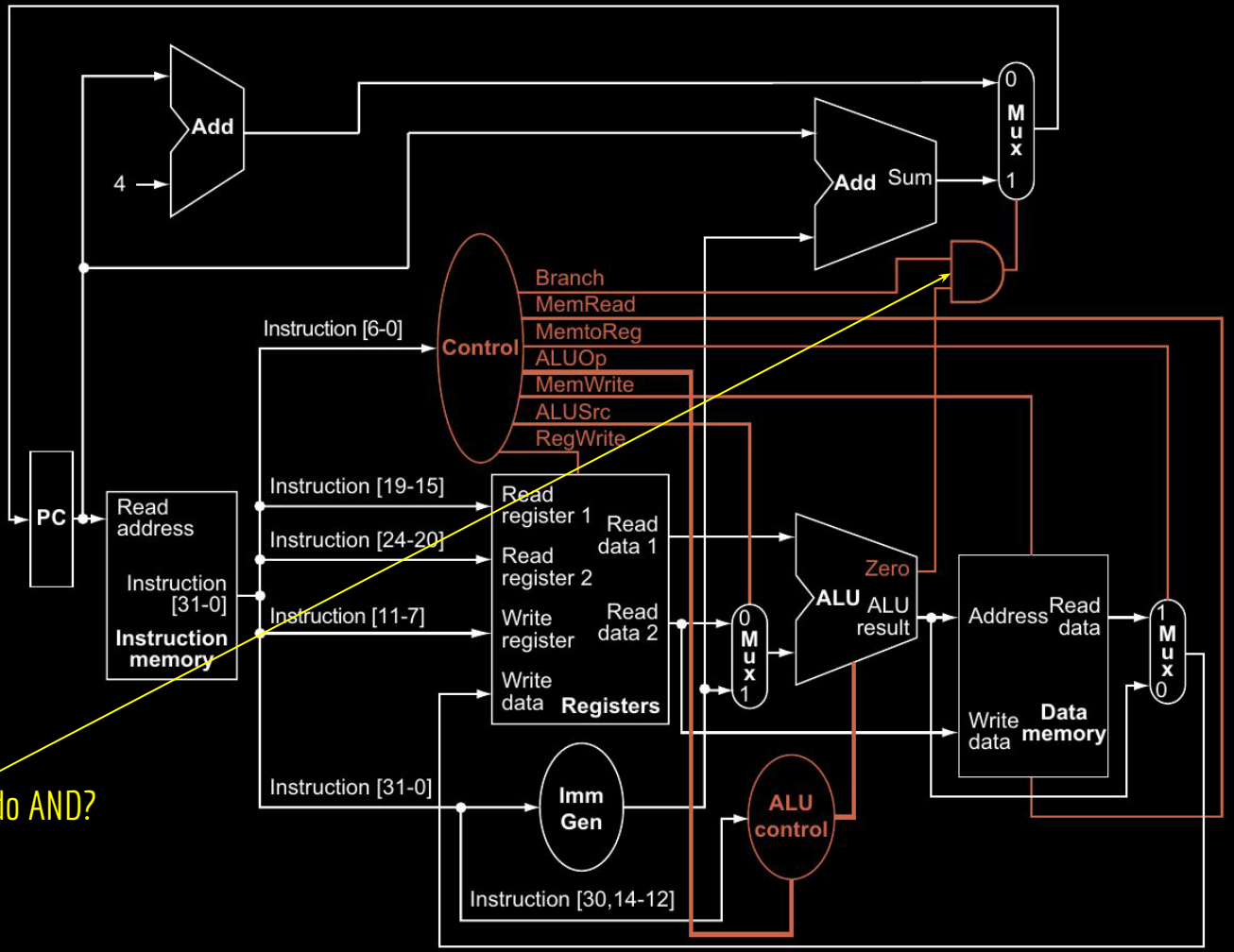
Instr.	Opcode							ALUSrc	MemToReg	RegWrite	MemRead	MemWrite	Branch	ALUOp1	ALUOp0
	Op6	Op5	Op4	Op3	Op2	Op1	Op0								
Tipo-R	0	1	1	0	0	1	1	0	0	1	0	0	0	1	0
lw	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0
SW	0	1	0	0	0	1	1	1	X	0	0	1	0	0	0
BEQ	1	1	0	0	0	1	1	0	X	0	0	0	1	0	1

Pesquise e verifique que esses realmente são os opcodes das instruções!

Circuito

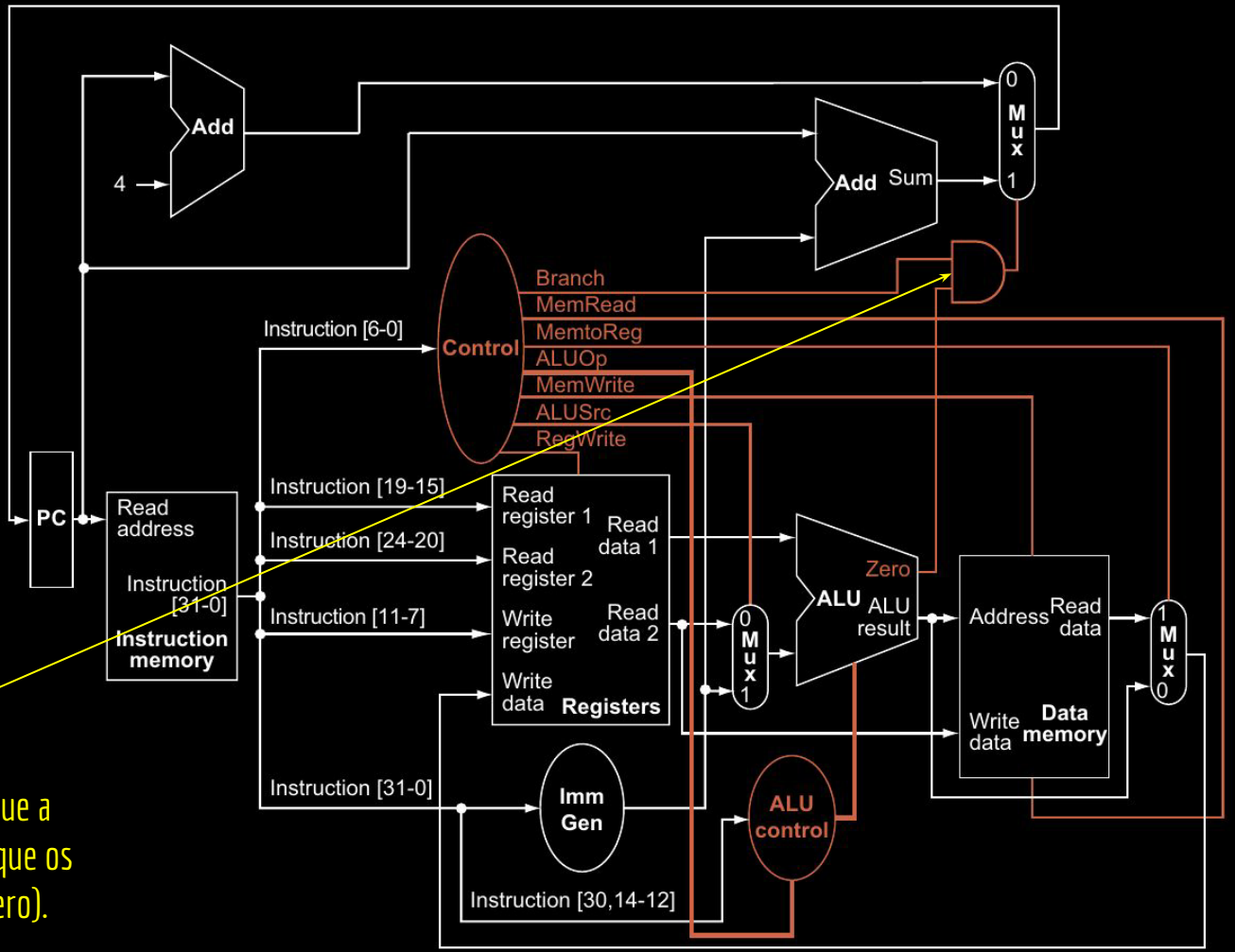

Add
4
Add
Sum
M
u
x
0
1
Branch
MemRead
MemtoReg
ALUOp
MemWrite
ALUSrc
RegWrite
Control
Instruction [6-0]
PC
Read address
Instruction [31-0]
Instruction memory
Instruction [19-15]
Instruction [24-20]
Instruction [11-7]
Read register 1
Read register 2
Write register
Write data
Read data 1
Read data 2
Registers
Zero
ALU
ALU result
Address
Read data
Write data
Data memory
Instruction [31-0]
Imm Gen
ALU control
Instruction [30,14-12]

Circuito



Qual o objetivo do AND?

Circuito



O desvio é tomado se o controle informa que a instrução é de branch, "E" a ALU informa que os operandos eram iguais (a subtração deu zero).

Controle Hardwired

Nossa unidade de controle é simples e pode ser do tipo ***hardwired***.

Definida com portas lógicas e de comportamento fixo.

Microcódigo

Em projetos complexos pode ser vantajoso criar unidades de controle programáveis.

- O programa na unidade de controle dita como os sinais de controle são gerados de acordo com as entradas.
- O programa é chamado de **microcódigo**.

Microcódigo

Em projetos complexos pode ser vantajoso criar unidades de controle programáveis.

- O programa na unidade de controle dita como os sinais de controle são gerados de acordo com as entradas.
- O programa é chamado de **microcódigo**.

É comum o uso para controlar pelo menos parte das CPUs atuais.

- Maior flexibilidade.

Podemos realizar correções no hardware *on-the-fly* (em pleno voo).

- Corrigir um erro de hardware atualizando seu software.

Microcódigo

CPUs modernas são complexas, e vulneráveis a erros de projeto que expõem falhas de segurança.

Exemplos de erros: Spectrem (2017), Meltdown (2017) e Foreshadow (2018).

A Intel e AMD “corrigiram” essas falhas atualizando o Microcódigo.

No Linux, abra um terminal e digite `dmesg | grep microcode` para verificar sua versão de microcódigo na CPU.

Microcódigos da Intel podem ser encontrados em https://github.com/intel/Intel-Linux-Processor-Microcode-Data-Files/blob/main/releasenote.md

Microcódigo

Muitas vezes o microcódigo é chamado de firmware.

> O programa que controla o fluxo interno do hardware.

Mais especificamente, **microcódigo é o firmware da CPU.**

Microcódigo

Muitas vezes o microcódigo é chamado de firmware.

> O programa que controla o fluxo interno do hardware.

Mais especificamente, **microcódigo é o firmware da CPU.**

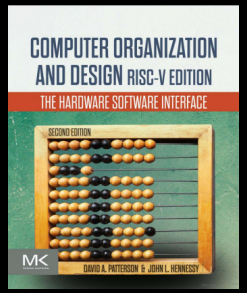
Veremos um pouco mais sobre o básico de microprograma em aulas futuras.

Exercícios

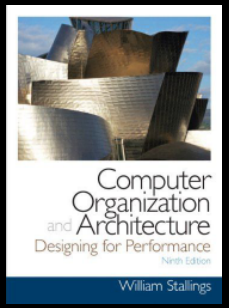
1. No livro base da disciplina é descrito em detalhes o caminho utilizado por cada tipo de instrução processador (tipo-R, lw/sw e branches). Leia esse trecho do capítulo.
2. Adicione instruções de Jump.
    a. Descreva as modificações no bloco operacional e de controle caso necessário.
    b. Indique o estado dos sinais de controle para essa nova instrução.
3. Considerando as instruções implementadas até o momento, qual a instrução que você considera que demora mais tempo para ser executada? Uma adição? Loads? Stores? ... Explique.
4. Adicione a instrução LUI no circuito. Descreva as modificações na unidade de controle, caso necessário. Indique o estado dos sinais de controle para essa nova instrução.

Referências

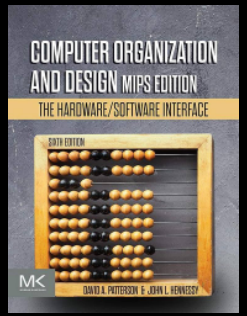
Patterson, Hennessy. Computer Organization and Design RISC-V Edition: The Hardware Software Interface. 2020.



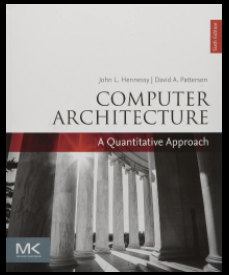
Stallings, W. Organização de Arquitetura de Computadores. 10a Ed. 2016.



Patterson, Hennessy. Computer Organization and Design MIPS Edition: The Hardware/Software Interface. 2020.



Hennessy, Patterson. Arquitetura de Computadores: uma abordagem quantitativa. 2019.



Licença