"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

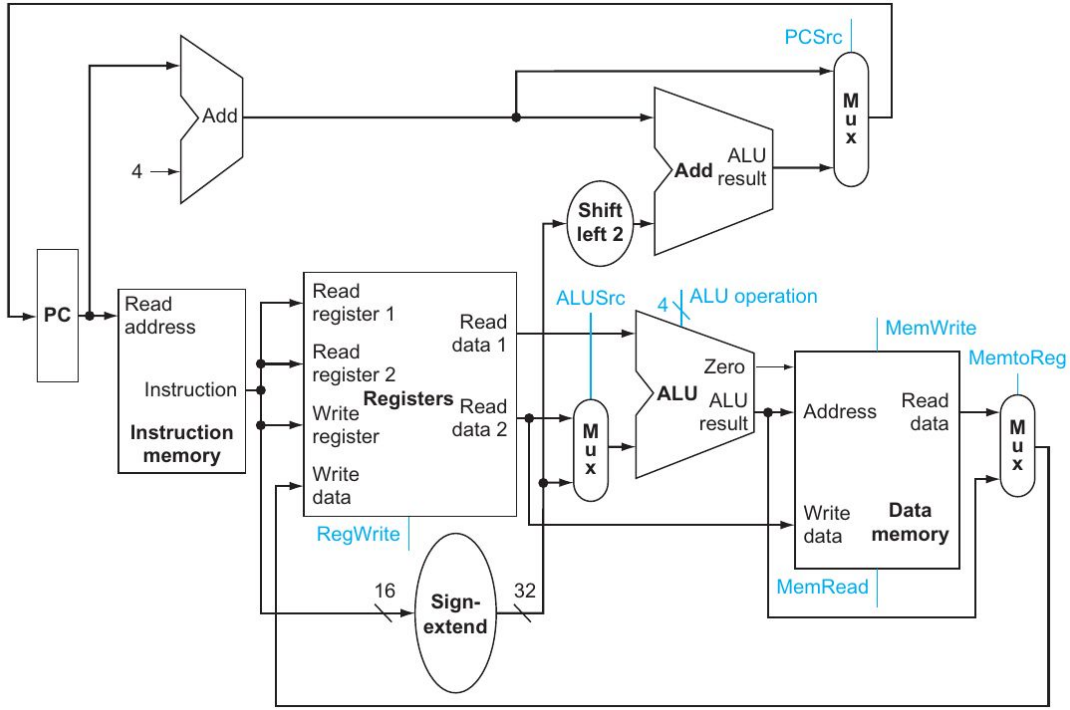
DSBD

UFPR
UNIVERSIDADE FEDERAL DO PARANÁ

Recapitulando...

Diversos sinais de controle não ligados.

- AluSrc, PCSrc, AluOperation, ...
- Determinam o comportamento de cada unidade.
- Dependem diretamente das instruções.



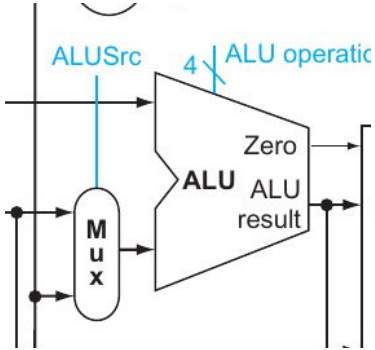
Controle da ALU

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
0111	1 se reg1 < reg2, 0 caso contrário (SLT)
1100	NOR



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 pelo campo *func* de 6 bits.

Sinal	Função da ALU
0000	AND
0001	OR
0010	Soma
0110	Subtração
0111	1 se reg1 < reg2, 0 caso contrário (SLT)
1100	NOR

Controle da ALU

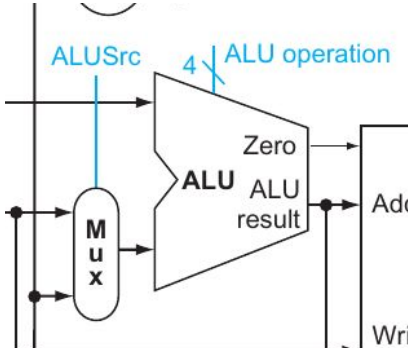
A unidade de controle da ALU vai receber como entrada um sinal de 2 bits, chamado ALUOp, que vai definir o tipo da instrução, e também vai receber o sinal do campo funct.

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 func.

Controle da ALU – Tabela verdade

Opcode	ALUOp	Instrução	Func	Operação da ALU	ALU Operation
lw	00	load word	XX XXXX	Soma	0010
sw	00	store word	XX XXXX	Soma	0010
beq	01	branch if equal	XX XXXX	Subtração	0110
Tipo-R	10	add	10 0000	Soma	0010
Tipo-R	10	subtract	10 0010	Subtração	0110
Tipo-R	10	and	10 0100	and	0000
Tipo-R	10	or	10 0101	or	0001
Tipo-R	10	slt	10 1010	slt	0111

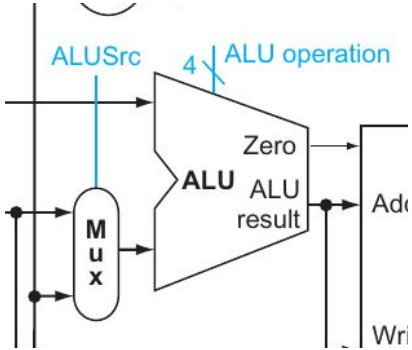


Controle da ALU – Tabela verdade

Opcode	ALUOp	Instrução	Func	Operação da ALU	ALU Operation
lw	00	load word	XX XXXX	Soma	0010
sw	00	store word	XX XXXX	Soma	0010
beq	01	branch if equal	XX XXXX	Subtração	0110
Tipo-R	10	add	10 0000	Soma	0010
Tipo-R	10	subtract	10 0010	Subtração	0110
Tipo-R	10	and	10 0100	and	0000
Tipo-R	10	or	10 0101	or	0001
Tipo-R	10	slt	10 1010	slt	0111

Entradas.

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



Controle da ALU – Tabela verdade

ALUOp		Funct						ALU Operation
ALUOp1	ALUOp2	F5	F4	F3	F2	F1	F0	
0	0	X	X	X	X	X	X	0010
X	1	X	X	X	X	X	X	0110
1	X	X	X	0	0	0	0	0010
1	X	X	X	0	0	1	0	0110
1	X	X	X	0	1	0	0	0000
1	X	X	X	0	1	0	1	0001
1	X	X	X	1	0	1	0	0111

Controle da ALU – Tabela verdade

ALUOp		Funct						ALU Operation
ALUOp1	ALUOp2	F5	F4	F3	F2	F1	F0	
0	0	X	X	X	X	X	X	0010
X	1	X	X	X	X	X	X	0110
1	X	X	X	0	0	0	0	0010
1	X	X	X	0	0	1	0	0110
1	X	X	X	0	1	0	0	0000
1	X	X	X	0	1	0	1	0001
1	X	X	X	1	0	1	0	0111

É fácil concluir que os dois bits do funct não são relevantes. No entanto, em nossos circuitos todos os campos de funct serão enviados para o controle da ALU, pois uma implementação com mais instruções do MIPS32 precisa desses campos.

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

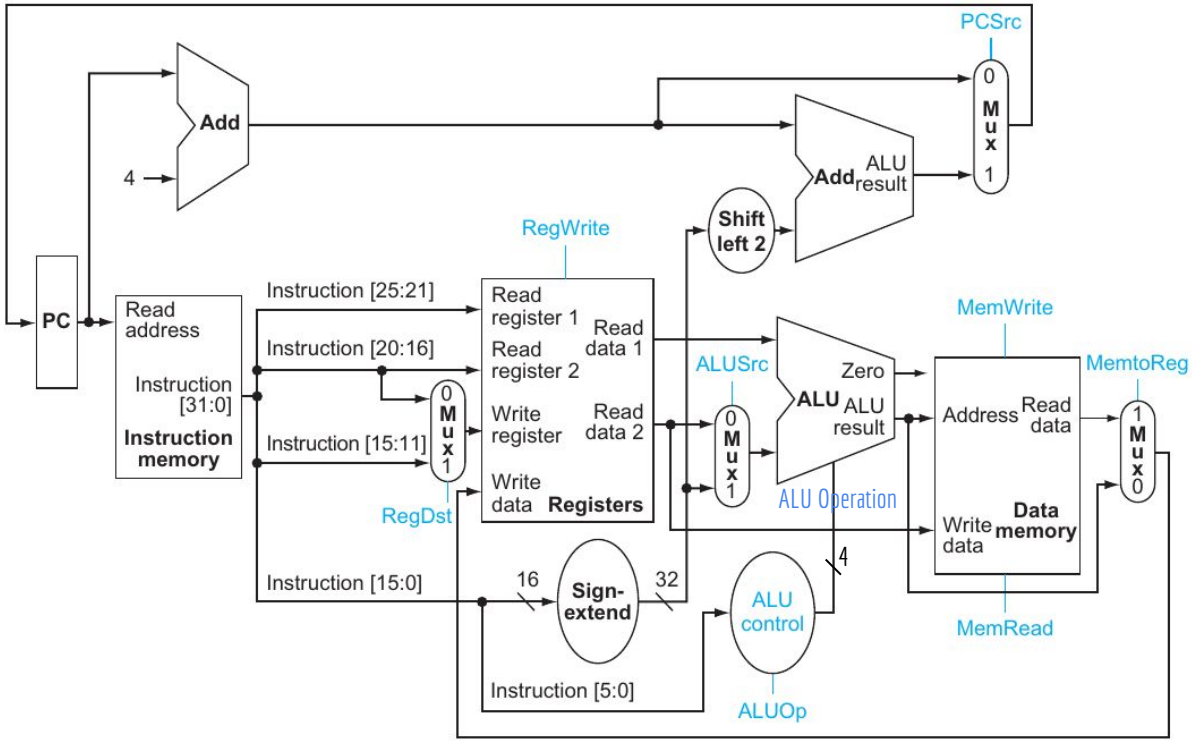
Por Álgebra de Boole ou,

Mapas de Karnaugh.

Controle da ALU

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

Depende de um sinal ALUOp de 2 bits.

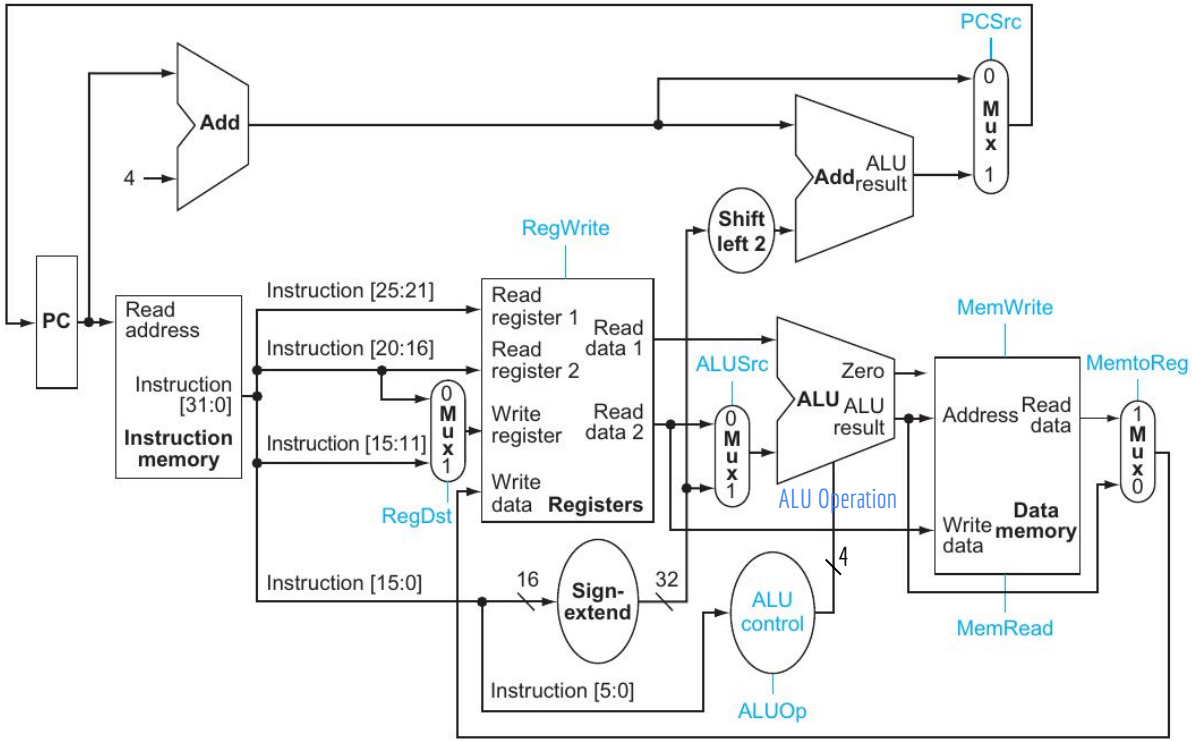


Controle da ALU

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.

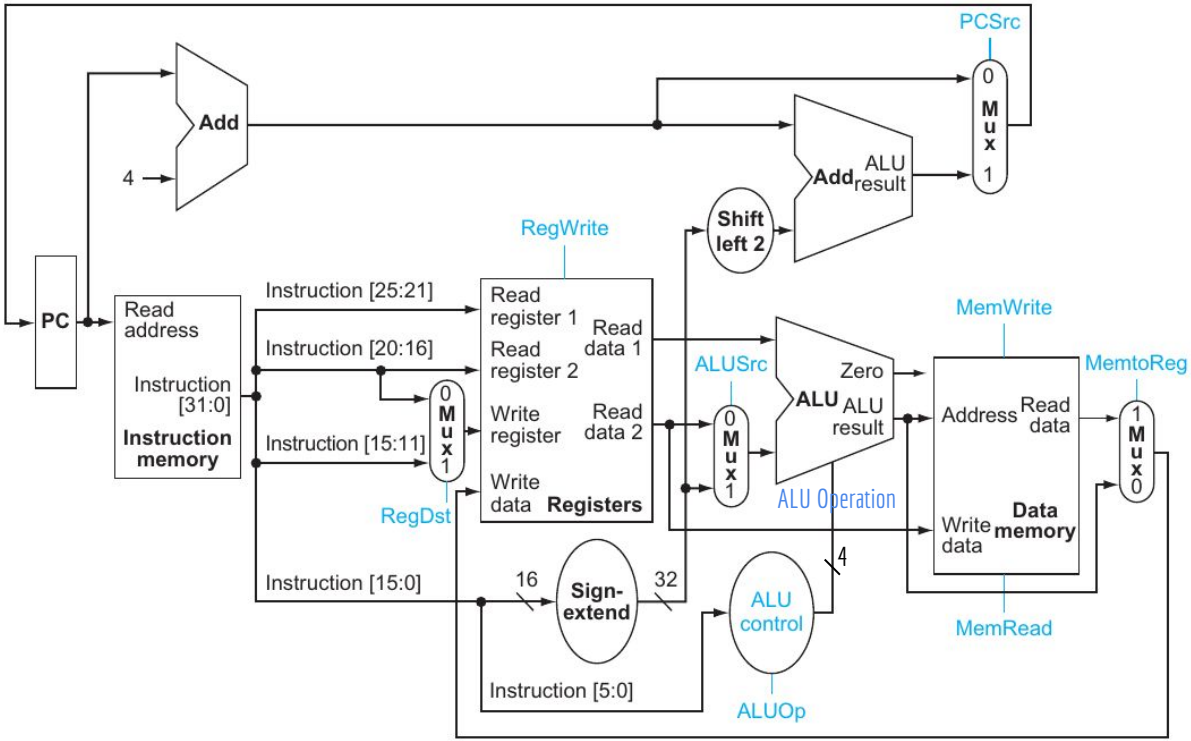
- Mais simples projetar.
- Possível redução no tamanho do circuito.
- Possível aumento de velocidade.
  - Unidades mais simples processam a informação mais rapidamente do que uma única unidade grande.

    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 7 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
RegDst	O núm. do reg. de destino deve vir do campo rt (bits [20:16]).	O núm. do reg. de destino deve vir do campo rd (bits [15:11]).
RegWrite		
ALUSrc		
PCSrc		
MemRead		
MemWrite		
MemToReg		

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
RegDst	O núm. do reg. de destino deve vir do campo rt (bits [20:16]).	O núm. do reg. de destino deve vir do campo rd (bits [15:11]).
RegWrite	Nada acontece.	Os dados enviados ao banco de registradores são escritos no endereço do registrador de escrita.
ALUSrc	O segundo operando da ALU vem do segundo registrador fonte lido do banco de registradores.	O segundo operando da ALU vem do campo imediato da instrução (16 bits estendidos para 32).
PCSrc	PC recebe PC+4.	PC recebe o endereço de desvio calculado.
MemRead	Nada acontece.	O conteúdo da memória no endereço especificado é enviado para a saída de leitura da memória.
MemWrite	Nada acontece.	O conteúdo da memória no endereço especificado é substituído pelo dado na entrada “WriteData” da memória.
MemToReg	O valor a ser escrito no registrador de destino deve vir da ALU.	O valor a ser escrito no registrador de destino deve vir da Memória.

Tabela verdade da Unidade de Controle

	Opcode						RegDst	ALUSrc	MemToReg	RegWrite	MemRead	MemWrite	Branch	ALUOp1	ALUOp0
	Op5	Op4	Op3	Op2	Op1	Op0									
Tipo-R	0	0	0	0	0	0	1	0	0	1	0	0	0	1	0
LW	1	0	0	0	1	1	0	1	1	1	1	0	0	0	0
SW	1	0	1	0	1	1	X	1	X	0	0	1	0	0	0
BEQ	0	0	0	1	0	0	X	0	X	0	0	0	1	0	1

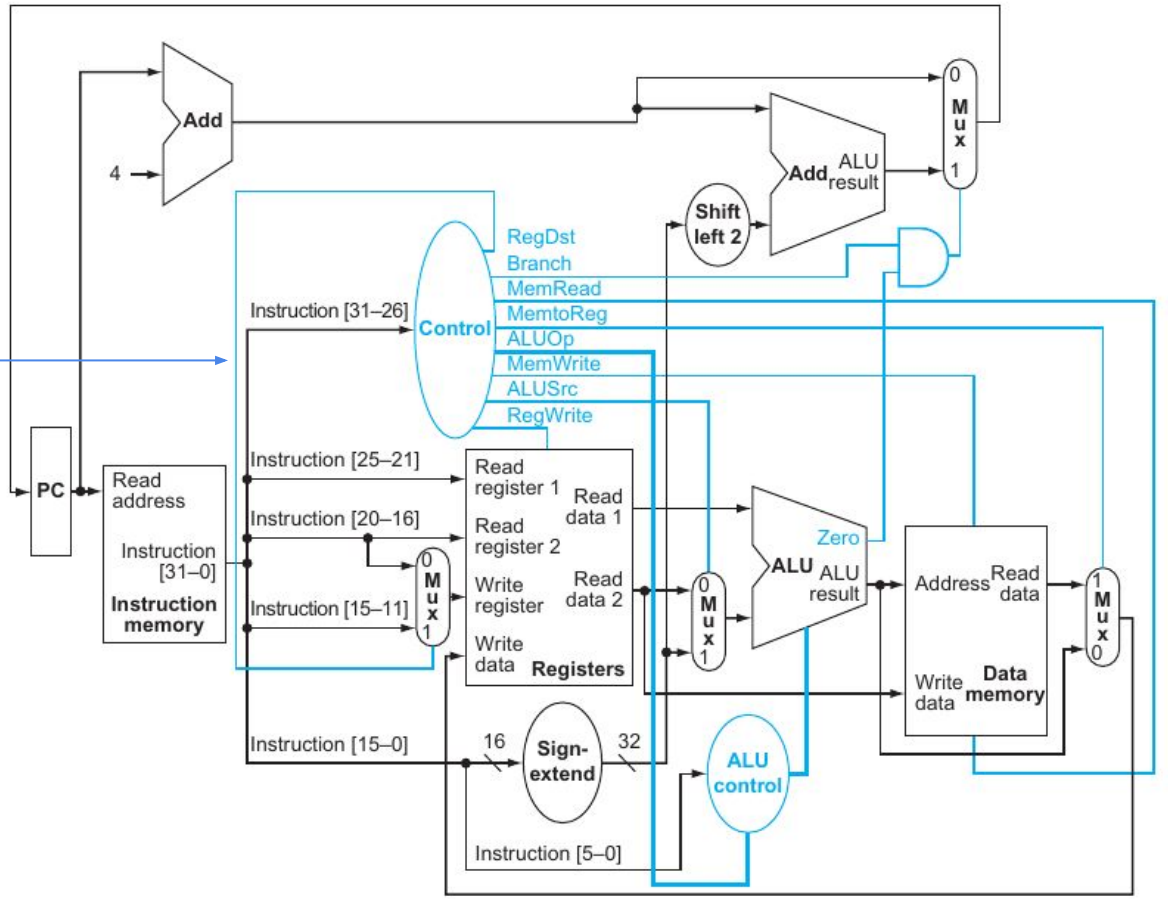
Tabela verdade da Unidade de Controle

	Opcode						RegDst	ALUSrc	MemToReg	RegWrite	MemRead	MemWrite	Branch	ALUOp1	ALUOp0
	Op5	Op4	Op3	Op2	Op1	Op0									
Tipo-R	0	0	0	0	0	0	1	0	0	1	0	0	0	1	0
LW	1	0	0	0	1	1	0	1	1	1	1	0	0	0	0
SW	1	0	1	0	1	1	X	1	X	0	0	1	0	0	0
BEQ	0	0	0	1	0	0	X	0	X	0	0	0	1	0	1

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

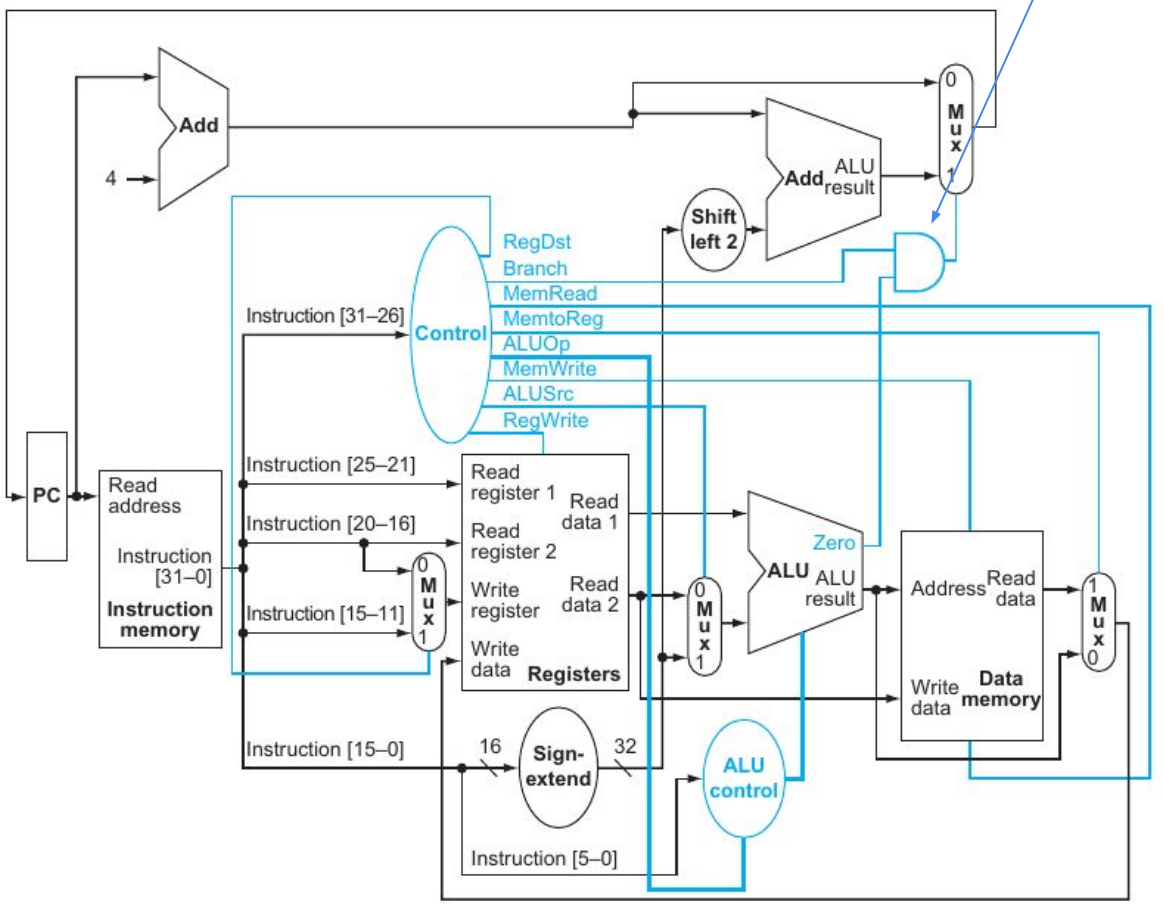
Circuito

Linhas “finas” representam sinais de 1 bit.



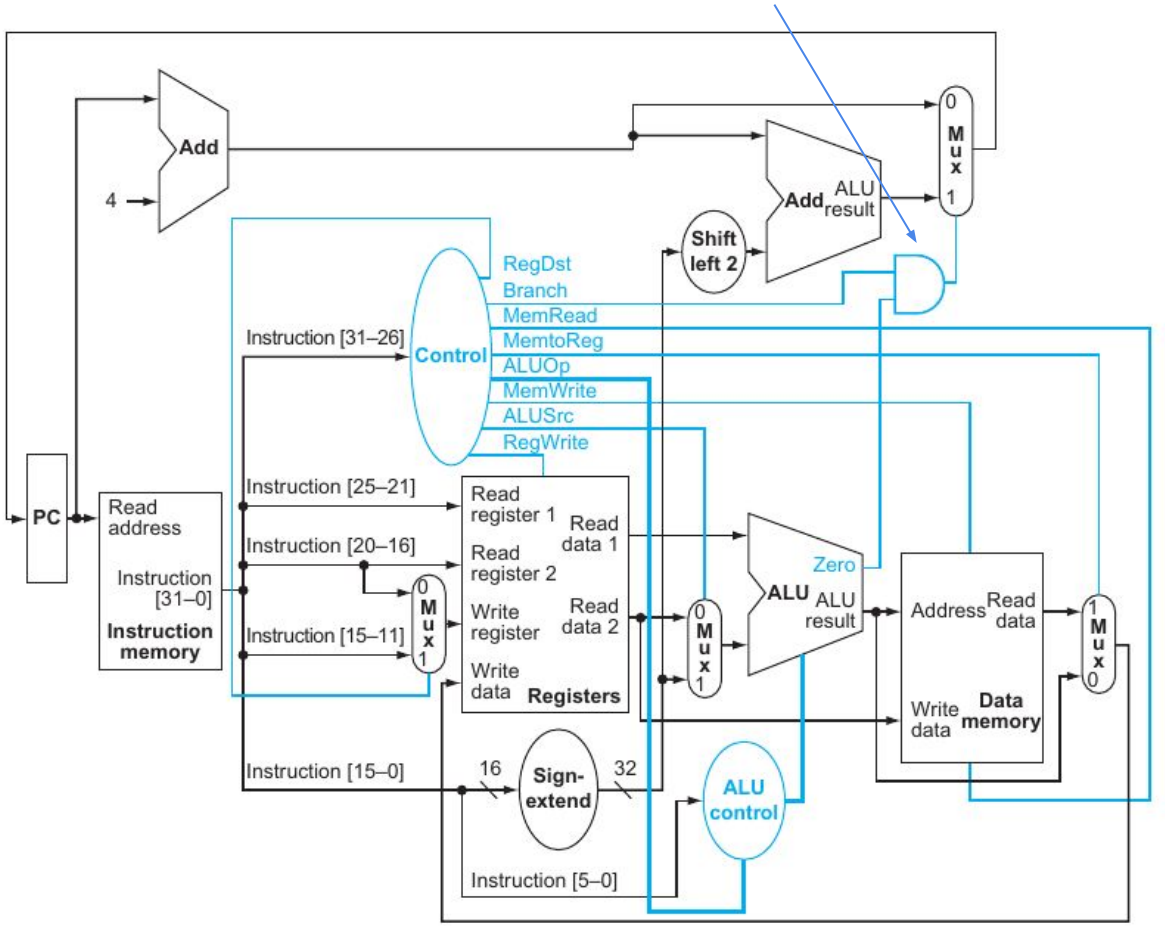
Circuito

Qual o objetivo desse 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**.

Microprograma

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 **microprograma**.

Microprograma

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 **microprograma**.

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.

Microprograma

Podemos realizar correções no hardware “on-the-fly” (em pleno voo)

Corrigir um erro de hardware atualizando seu software.

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

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

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

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

Microprograma

Muitas vezes o microprograma é chamado de **firmware.**

O programa que controla o fluxo interno do hardware.

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

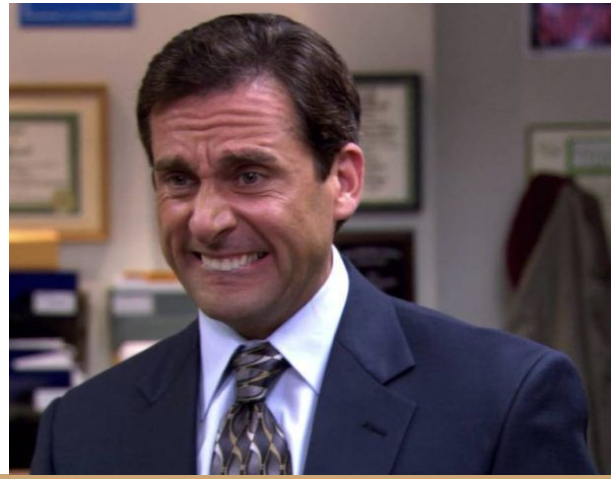
Microprograma

Necessário criar uma “mini CPU” dentro da CPU, que vai gerar os sinais de controle.

A unidade de controle microcontrolada precisa ser muito rápida (para gerar os sinais de controle a tempo).

Vai ter seu próprio conjunto de instruções.

Micro-instruções.



Microprograma

Necessário criar uma “mini CPU” dentro da CPU, que vai gerar os sinais de controle.

A unidade de controle microcontrolada precisa ser muito rápida (para gerar os sinais de controle a tempo).

Vai ter seu próprio conjunto de instruções.

Micro-instruções.

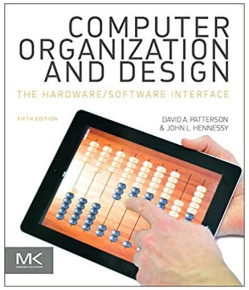
Veremos um pouco mais sobre o **básico** de microcódigo em aulas futuras.

Exercícios

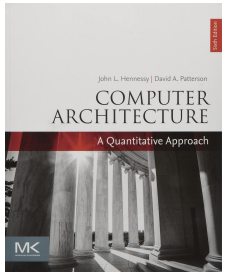
1. No livro base da disciplina é descrito em detalhes o caminho utilizado por cada tipo de instrução processador (tipo-R, lw/sw e branches). Leia esse trecho do capítulo, que foi disponibilizado no Moodle.
2. Adicione a instrução de Jump.
    a. Descreva as modificações no bloco operacional e de controle 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.
    a. Descreva as modificações no bloco operacional e de controle caso necessário.
    b. Indique o estado dos sinais de controle para essa nova instrução.

Referências

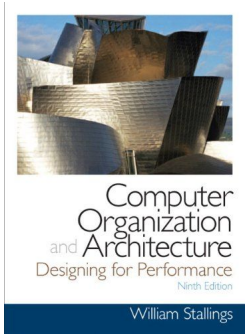
Patterson, Hennessy . Arquitetura e Organização de Computadores: A interface hardware/software. 2014.



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



Stallings, W. Organização de Arquitetura de Computadores. 2016.



Licença