[Aula 08] Caminho de dados e controle 2 – MIPS Monociclo

Prof. João F. Mari
joaof.mari@ufv.br

Roteiro

- Introdução
- MIPS Monociclo
- Acrescentando o controle
- O controle da ULA
- A unidade de controle principal
- Caminho de dados com todos os multiplexadores e linhas de controle
- Descrição dos sinais de controle
- Execução de instruções do tipo R
- Execução de instruções Load/Store
- Execução de instruções BEQ
- Implementando jumps
- Implementação e ciclo único

MIPS Monociclo

- Projeto de ciclo único: a busca, decodificação e execução das instruções ocorre em um único ciclo de clock
  - Nenhum recurso do caminho de dados pode ser usados mais de uma vez por instrução
    - Aqueles que necessitam ser utilizados mais de uma vez devem ser replicados
    - EX: memória de instruções e de dados separados, mais de um somador.
  - Multiplexadores são necessários na entrada dos componentes compartilhados para realizar a seleção
  - Sinais de escrita para controlar a gravação no banco de registradores e na memória de dados
- O tempo de ciclo é determinado pelo tamanho do caminho mais longo (caminho com maior tempo de execução).

Acrescentando o controle

- O controle de ALU
  - Dependendo da instrução, uma das operações abaixo deverá ser executada
    - Aritméticas e lógicas (and, or, sub, add, slt)
    - Load/store (add para cálculo do endereço)
    - BEQ (subtração)

Entrada de controle da ULA	Função
0000	AND
0001	OR
0010	Soma
0110	Subtração
0111	Set on less than
1100	NOR

Acrescentando o controle

- O controle da ALU
  - A entrada do controle da ALU de 4 bits é gerada por uma pequena unidade de controle com duas entradas:
    - O campo funct da instrução
    - Um campo controle de 2 bits - OpALU.
  - OpALU indica:
    - 00 : add para load/stores
    - 01: sub para beq
    - 10: determinada pela operação do campo funct

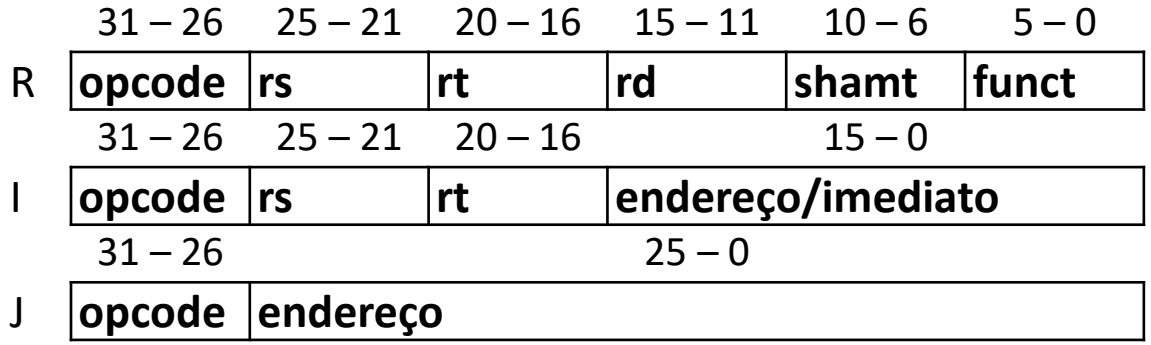
O controle da ULA

- OpALU: 2 bits de controle
  - Juntamente com o campo funct, definem a função.
- Entradas OpALU1 e OpALU2:
  - 00: load e store (soma endereços)
  - 01: beq
  - 10: a função é determinada pelo campo de função

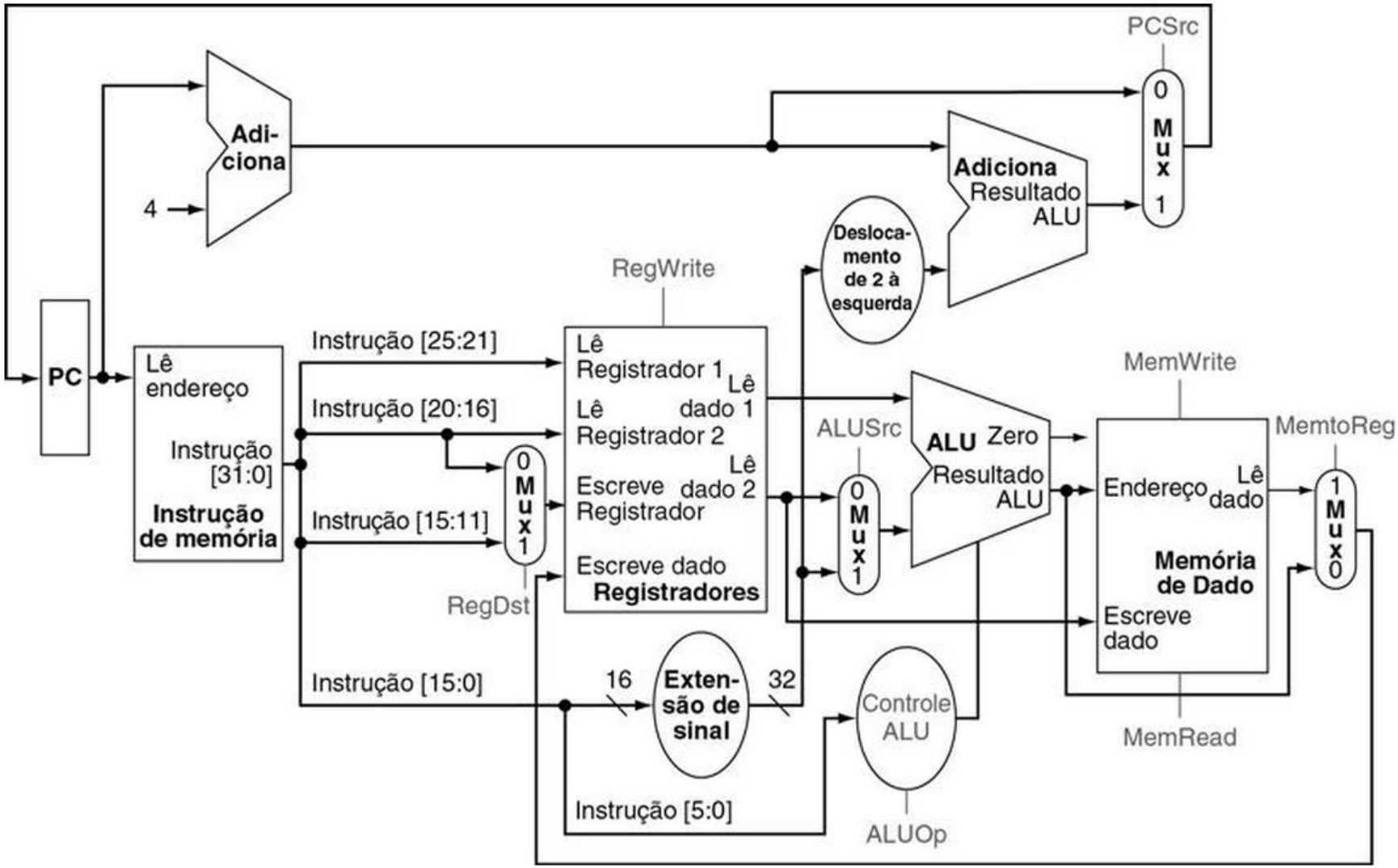
Opcode da instrução	OpALU	Operação da instrução	Campo funct	Ação da ALU desejada	Entrada do controle da ALU
LW	00	load store	XXXXXX	add	0010
SW	00	store word	XXXXXX	add	0010
Branch equal	01	branch equal	XXXXXX	subtract	0110
tipo R	10	add	100000	add	0010
tipo R	10	subtract	100010	subtract	0110
tipo R	10	AND	100100	AND	0000
tipo R	10	OR	100101	OR	0001
tipo R	10	set on less than	101010	set on less than	0111

A unidade de controle principal

- Formatos de instrução considerados
  - Opcode: bits 31-26 (6 bits).
  - Registradores a serem lidos (rs e rt): bits 25-21 e 20-16 (5 bits cada)
    - Instruções tipo R, BEQ e store word
  - Registrador-base para instruções de load e store (rs): bits 25-21 (5 bits)
  - Constante de 16 bits: bits 15-0 (16 bits)
    - Deslocamento do BEQ e offset do lw e sw
  - Registrador destino:
    - Para lw (rt): bits 20-16 (5 bits).
    - Para tipo R (rd): bits 15-12 (5 bits)
    - É necessário um multiplexador antes do banco de registradores.



Caminho de dados com todos os multiplexadores e linhas de controle.



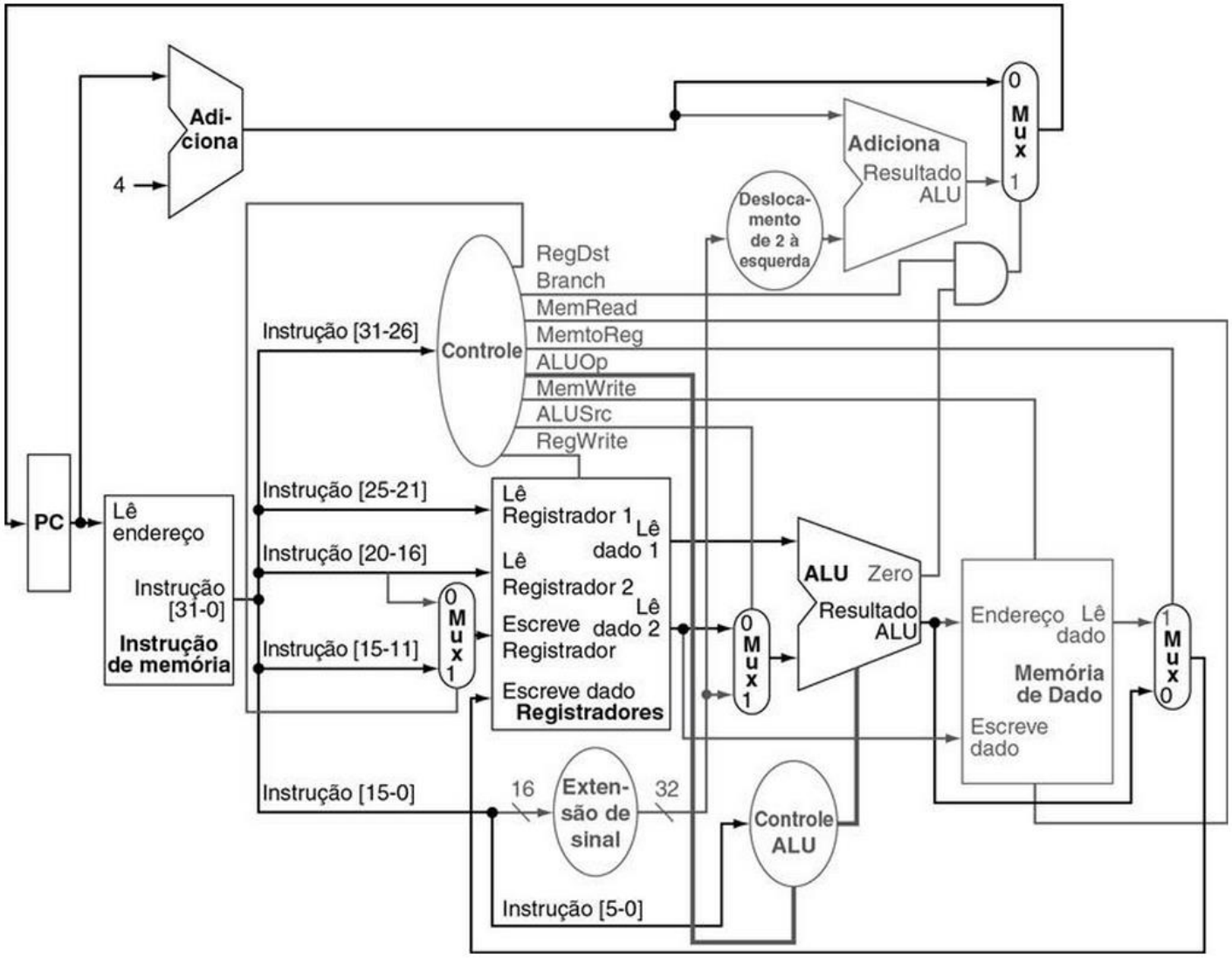
Efeito dos sinais da unidade de controle

Nome do sinal	Efeito quando inativo	Efeito quando ativo
RegDst	O número do registrador destino para a entrada Registrador para escrita vem do campo rt (bits 20:16)	O número do registrador destino para a entrada 'Registrador para escrita' vem do campo rd (bits 15:11)
EscreveReg	Nenhum.	O registrador na entrada 'Registrador para escrita' é escrito como o valor da entrada 'Dados para escrita'.
OrigALU	O segundo operando da ULA vem da segunda saída do banco de registradores (Dados de leitura 2).	O segundo operando da ULA consiste nos 16 bits mais baixos da instrução com sinal estendido.
OrigPC	O PC é substituído pela saída do somador que calcula o valor de PC + 4.	O PC é substituído pela saída do somador que calcula o destino do desvio.
LerMem	Nenhum.	O conteúdo da memória de dados designado pela entrada Endereço é colocado na saída 'Dados da leitura'.
EscreveMem	Nenhum.	O conteúdo da memória de dados designado pela entrada Endereço é substituído pelo valor na entrada 'Dados para escrita'.
MemParaReg	O valor enviado para a entrada 'Dados para escrita' do banco de registradores vem da ULA.	O valor enviado a entrada 'Dados para escrita' do banco de registradores vem da memória de dados.

Execução de instruções R

1. Busca da instrução na memória de instruções e incremento do PC
2. Dois registradores, $t0 e $t1, são lidos do banco de registradores. A unidade de controle coloca valores nas linhas de controle
3. A UAL opera sobre os dados lidos do banco de registradores, usando o código da função (bits 5-0) para gerar a função da UAL
4. O resultado da UAL é escrito no banco de registradores usando-se os bits 15-11 da instrução para selecionar o registrador-destino ($t1)

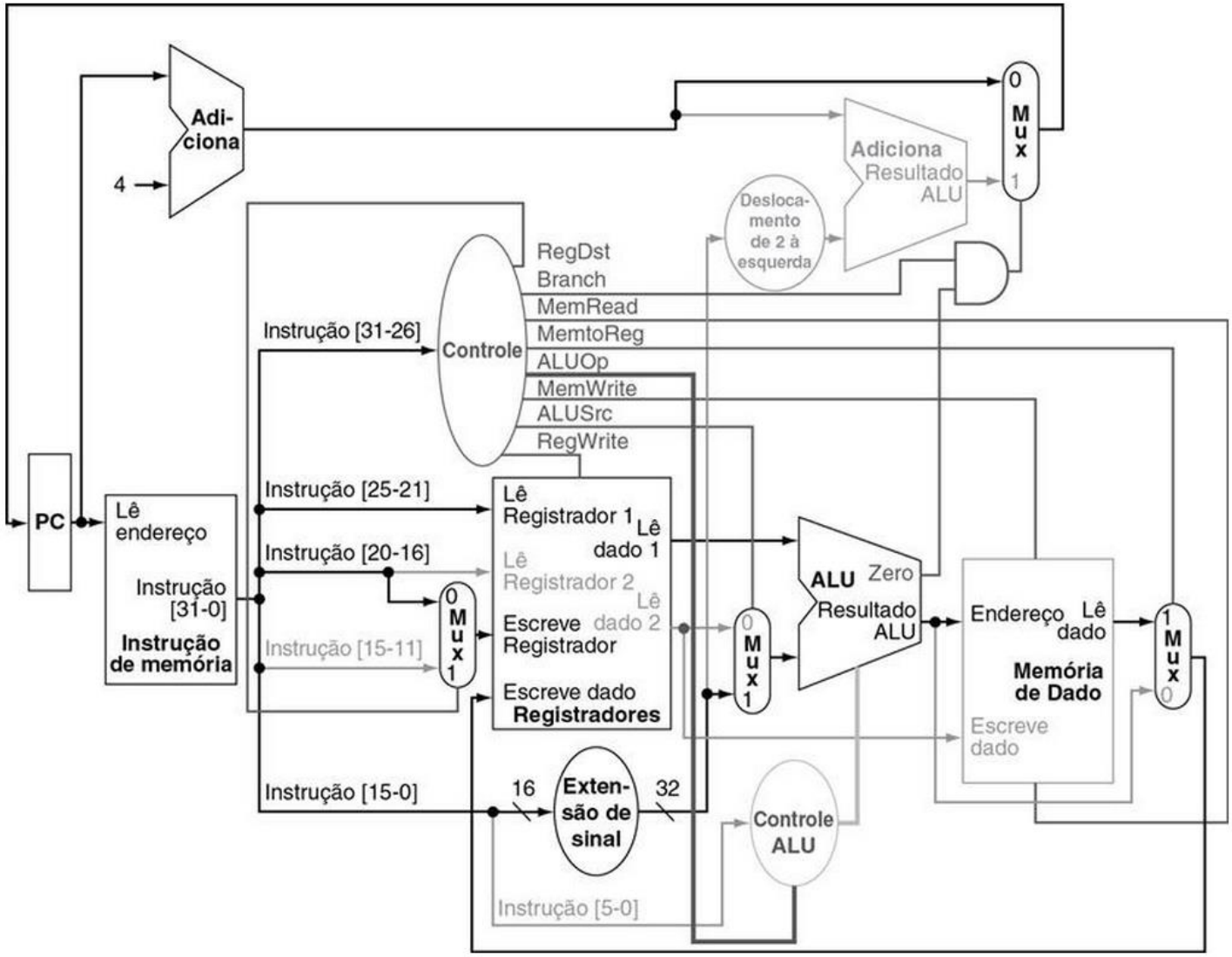
[EX] add $t1, $t2, $t3



Execução de instruções LW e SW

1. Busca da instrução na memória de instruções e incremento do PC
2. Leitura do conteúdo de um registrador ($t2) do banco de registradores
3. Cálculo da soma do valor lido do banco de registradores com o resultado da extensão do sinal de 16 bits menos significativos da instrução (deslocamento)
4. O resultado da soma é usado para endereçar a memória de dados
5. O dado vindo da unidade de memória é escrito no banco de registradores;
   - O número do registrador-destino é dado pelos bits 20-16 da instrução ($t1)

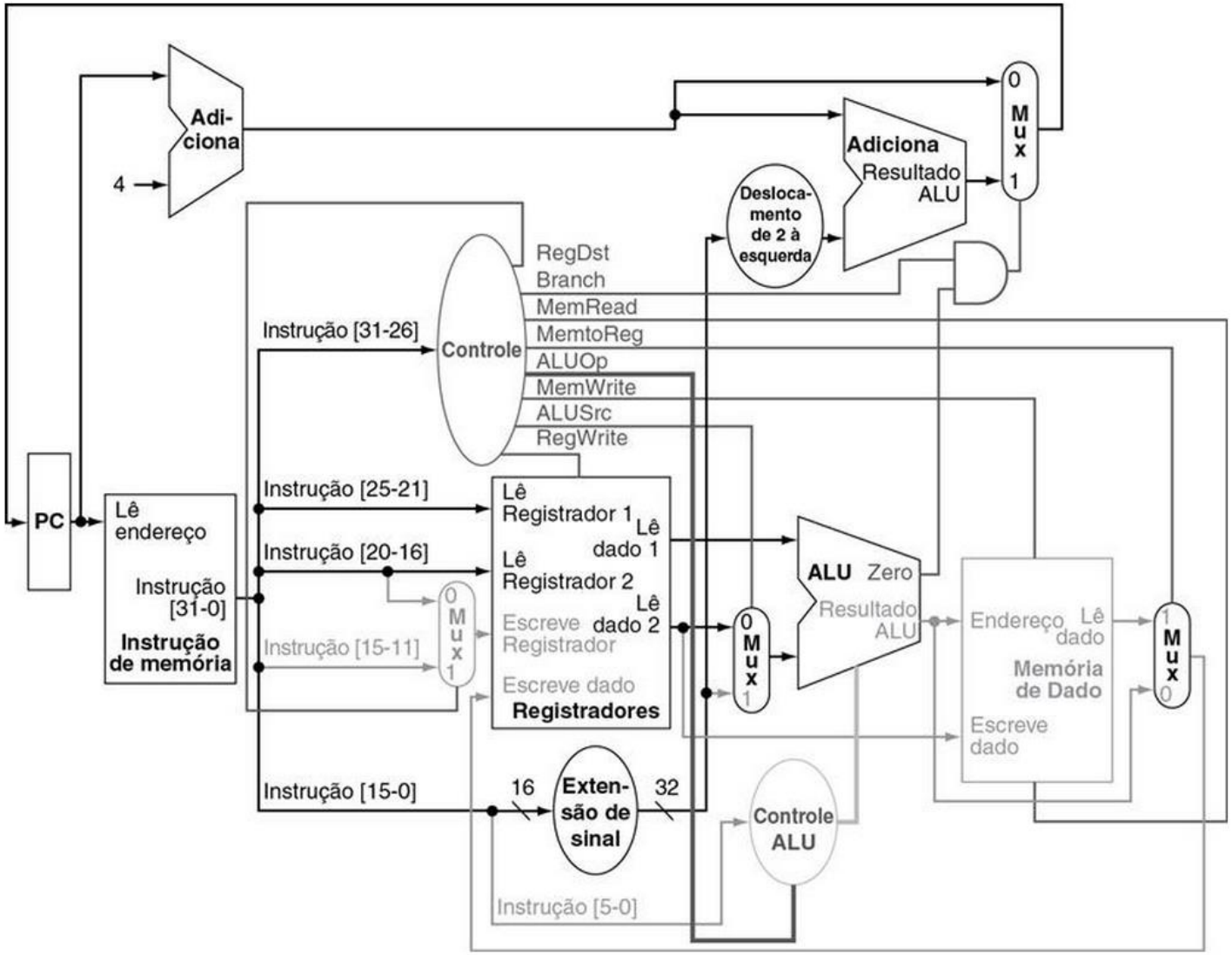
[EX] lw $t1, offset($t2)



Execução de instruções BEQ

1. Busca da instrução na memória de instruções e incremento do PC
2. Leitura do conteúdo dos registradores $t1 e $t2
3. Realização de uma subtração pela ALU sobre os dois valores lidos do bando de registradores.
   - O valor de PC+4 é somado ao resultado da extensão do sinal dos 16 bits menos significativos da instrução (deslocamento) deslocado de dois bits à esquerda.
   - O resultado dessa soma é o endereço de destino do desvio
4. A saída Zero da ALU é usada para decidir se o PC deve ser atualizado com o valor de PC+4 ou com o valor do endereço de destino do desvio condicional

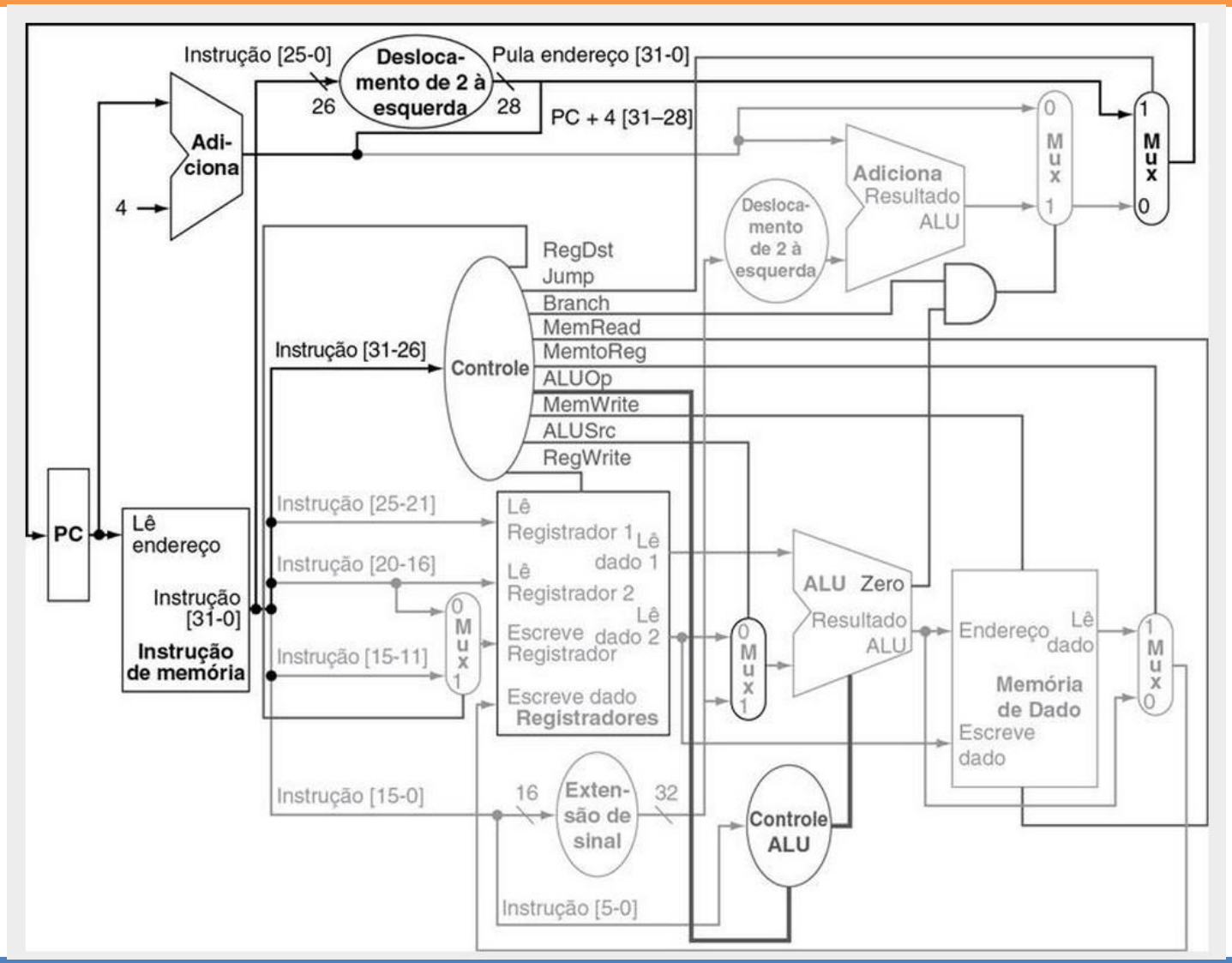
[EX] beq $t1, $t2, offset



Implementando jumps

- A instrução jump se parece um pouco com uma instrução branch,
  - Mas calcula o PC de destino de maneira diferente e não é condicional
- Os 2 bits menos significativos são sempre 00.
  - Deslocamento de dois bits a esquerda (mult. por 4).
  - Os próximos 26 bits mais significativos vem do campo de 26 bits imediato da instrução
- Assim, pode-se implementar um jump armazenando no PC a concatenação dos:
  - 4 bits superiores do PC atual + 4 (esses são bits 31:28 do endereço da instrução imediatamente seguinte)
  - O campo de 26 bits imediato da instrução jump
  - Os bits 00

Implementando jumps



Implementação de ciclo único

- O projeto de ciclo único é ineficiente
  - O clock precisa ter a mesma duração para cada instrução .
    - O CPI será sempre 1!
- O tempo de ciclo é determinado pelo caminho mais longo
  - Esse caminho corresponde a instrução lw que usa cinco unidades funcionais em série:
    - Memória de instruções
    - Banco de registradores
    - ALU
    - Memória de dados
    - Banco de registradores
  - Embora o CPI seja 1, o desempenho geral de uma implementação de ciclo único provavelmente não será bom
    - Outras instruções poderiam ser um ciclo de clock mais simples.

Implementação de ciclo único

- Suponha os seguintes tempos de operação
  - Unidade de memória: 200 ps
  - ALU e somadores: 100 ps
  - Banco de registradores: 50 ps

- Qual das seguintes implementações seria mais rápida e por quanto?
  - Considerando que os multiplexadores, unidade de controle, acessos ao PC, unidade de extensão do sinal e os fios não possuem atraso.
    - A) Toda instrução opera em 1 ciclo de clock de duração fixa
    - B) Toda instrução é executada em 1 ciclo de clock. Considerando um ciclo de clock com duração variável que, para cada instrução, tem apenas a duração necessária.
      - Método não prático.

Implementação de ciclo único

- **Solução:**
  - Considerando o mix de instruções:
    - 25% loads, 10% stores, 45% ALU, 15% desvios, 5% jumps
  - Tempo de execução = Contagem instruções x CPI x Tempo de ciclo
  - Tempo de execução = Contagem instruções x 1 x Tempo de ciclo
  - Tempo de execução = Contagem instruções x Tempo de ciclo

- Caminhos críticos:

Classe de instrução	**Unidades funcionais usadas**				
Tipo R	Busca de instrução	Acesso a registrador	ULA	Acesso a registrador	
Load word	Busca de instrução	Acesso a registrador	ULA	Acesso a memória	Acesso a registrador
Store word	Busca de instrução	Acesso a registrador	ULA	Acesso a memória	
Branch	Busca de instrução	Acesso a registrador	ULA		
Jump	Busca de instrução				

Implementação de ciclo único

- Tempo exigido para cada classe de instruções:

Classe de instrução	Memória de instrução	Leitura de registrador	Operação ULA	Memória de Dados	Escrita de Registrador	Tempo total (ps)
Tipo R	200	50	100	200	50	400
Load word	200	50	100	200	50	600
Store word	200	50	100	200		550
Branch	200	50	100			350
Jump	200					200

- O tempo do ciclo de clock é determinado pela instrução mais longa:
  - Nesse caso, 600ps para ser executada:

Implementação de ciclo único

- Ciclo único
  - Como precisamos considerar que o ciclo de clock é igual ao atraso de pior caso para todas instruções
    - Não é possível utilizar técnicas de implementação que reduzem o atraso do caso comum:
      - Violação dos princípios de projeto!
  - Na implementação de ciclo único cada unidade funcional precisa ser duplicada
    - Eleva o custo de implementação
      - O projeto de ciclo único é ineficiente no desempenho e no custo
  - Solução: utilizar uma implementação multiciclo
    - Dessa forma, as instruções podem utilizar mais de um ciclo de clock para serem executadas, dependendo de sua complexidade.

Bibliografia

1. PATTERSON, D.A; HENNESSY, J.L. **Organização e Projeto de Computadores: A Interface Hardware/Software**. 3a. Ed. Elsevier, 2005.
   - Capítulo ***4.***

2. Notas de aula do prof. Luciano J. Senger:
   - http://www.ljsenger.net/classroom.html

3. Notas de aula da Profa. Mary Jane Irwin
   - CSE 331 Computer Organization and Design
   - http://www.cse.psu.edu/research/mdl/mji/mjicourses

FIM

- FIM:
  - **Aula 08** – Caminho de dados e controle 2 – MIPS Monociclo
- Próxima aula:
  - **Aula 09** – Caminho de dados e controle 3 – MIPS Multiciclo