

Fundamentos de Arquitetura de Computadores

Tiago Alves

Faculdade UnB Gama
Universidade de Brasília



Análise de Desempenho

Classe de instrução	Unidades funcionais usadas pela classe de instrução				
tipo R	Busca de instrução	Acesso a registrador	ALU	Acesso a registrador	
Load word	Busca de instrução	Acesso a registrador	ALU	Acesso à memória	Acesso a registrador
Store word	Busca de instrução	Acesso a registrador	ALU	Acesso à memória	
Branch	Busca de instrução	Acesso a registrador	ALU		
Jump	Busca de instrução				

Usando esses caminhos críticos, podemos calcular o tamanho exigido para cada classe de instrução:

Classe de instrução	Memória de instrução	Leitura de registrador	Operação ALU	Memória de dados	Leitura de registrador Escrita	Total
Tipo R	200	50	100	0	50	400ps
Load word	200	50	100	200	50	600ps
Store word	200	50	100	200		550ps
Branch	200	50	100	0		350ps
Jump	200					200ps

MIPS Multiciclo

Idéia: cada fase de execução de uma instrução dura um ciclo de relógio

Instruções podem ser executadas em um número diferente de ciclos

Exemplos:

- `lw` demora 5 ciclos
- `jump` demora 3 ciclos
- `add` demora 4 ciclos



MIPS Multiciclo

- Ciclo dimensionado de acordo com a etapa/módulo mais demorada
- Vamos reutilizar unidades funcionais:
 - ULA usada também para calcular endereço e incrementar o PC
 - Memória usada para armazenar instrução e dados
- A organização da parte operativa pode ser re-estruturada em função destas características
- Sinais de controle não serão determinados diretamente pela instrução
- Usaremos uma máquina de estado finito para controle



Unidade Operativa Multiciclo

Dividir as instruções em etapas, cada uma durando um ciclo:

- equilibrar a quantidade de trabalho a ser feito
- restringir cada ciclo para usar apenas uma vez cada unidade funcional

No final de um ciclo:

- armazenar os valores para uso em ciclos posteriores
- Logo, necessita introduzir registradores “internos” adicionais

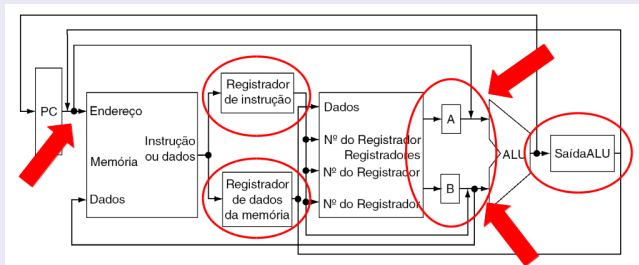


Unidade Operativa Multiciclo

Visão Abstrata:

- Uma única ULA
- Uma única Unidade de Memória
- Registradores após unidades funcionais

onde colocá-los? Unidade combinacional de 1 ciclo e onde necessitar ser salvo. No nosso caso:



Obs.:

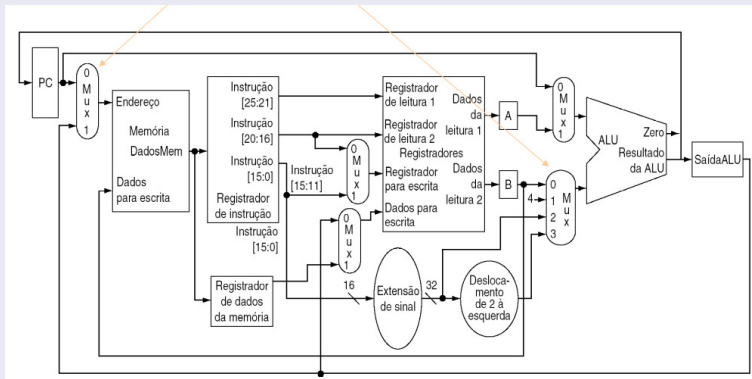
Dados utilizados pelas etapas subsequentes armazenados em registradores (elemento de estado invisível ao programador)

Dados utilizados pelas instruções subsequentes armazenados na Memória, Banco de Registradores e PC (visível ao programador)

Unidade Operativa Multiciclo

Detalhamento:

- Todos os registradores precisam manter o dado por 1 ciclo apenas (EXCEÇÃO IR).
Logo não necessitam de pino de controle de escrita
- **Incluir e expandir** Multiplexadores para poder **reusar** Unidades Funcionais.



Caminho de Dados Multiciclo

Definição de linhas de controle

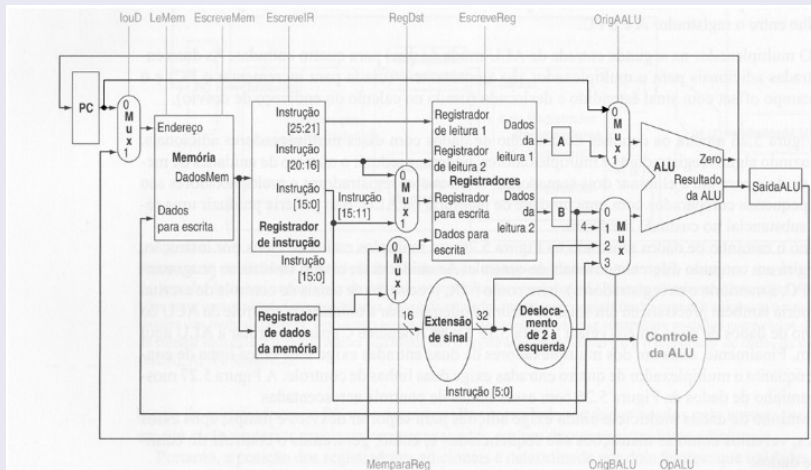
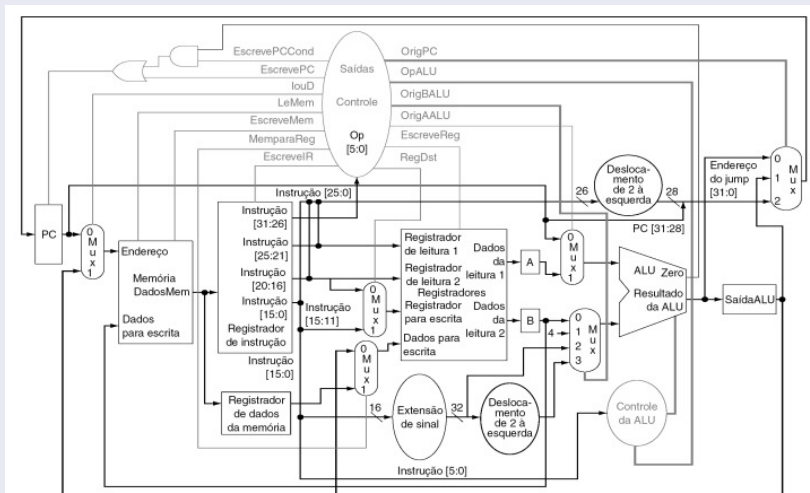


FIGURA 5.27 O caminho de dados multiciclo da Figura 5.26 com as linhas de controle indicadas.

Implementação MIPS III: MIPS Multiciclo

Caminho de Dados Multiciclo

Implementação do controle + controle do PC (PC+4, beq, jump)



Ações dos Sinais de Controle

Ações dos sinais de controle de 1 bit

Nome do sinal	Efeito quando inativo	Efeito quando ativo
RegDst	O número do registrador de destino do banco de registradores para a entrada Registrador para escrita vem do campo rt.	O número do registrador de destino do banco de registradores para a entrada Registrador para escrita vem do campo rd.
EscreveReg	Nenhum.	O registrador de uso geral selecionado pelo número na entrada Registrador para escrita é escrito com o valor da entrada Dados para escrita.
OrigAALU	O primeiro operando da ALU é o PC.	O primeiro operando da ALU vem do registrador A.
LeMem	Nenhum.	O conteúdo da memória no local especificado pela entrada Endereço é colocado na saída Dados da memória.
EscreveMem	Nenhum.	O conteúdo da memória no local especificado 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 de SaídaALU.	O valor enviado para a entrada Dados para escrita do banco de registradores vem do MDR.
louD	O PC é usado para fornecer o endereço para a unidade de memória.	SaídaALU é usado para fornecer o endereço para a unidade de memória.
IR Write	Nenhum.	A saída da memória é escrita no IR.
EscrevePC	Nenhum.	O PC é escrito; a origem é controlada por OrigPC.
EscrevePCCond	Nenhum.	O PC é escrito se a saída Zero da ALU também estiver ativa.



Ações dos Sinais de Controle

Ações dos sinais de controle de 2 bits

Nome do sinal	Valor (binário)	Efeito
OpALU	00	A ALU realiza uma operação de adição.
	01	A ALU realiza uma operação de subtração.
	10	O campo funct da instrução determina a operação da ALU.
OrigBALU	00	A segunda entrada para a ALU vem do registrador B.
	01	A segunda entrada da ALU é a constante 4.
	10	A segunda entrada da ALU são os 16 bits menos significativos com sinal estendido do IR.
	11	A segunda entrada da ALU são os 16 bits menos significativos com sinal estendido do IR deslocados de 2 bits para a esquerda.
OrigPC	00	A saída da ALU ($PC + 4$) é enviada ao PC para escrita.
	01	O conteúdo de SaídaALU (o endereço de destino do desvio) é enviado ao PC para escrita.
	10	O endereço de destino do jump ($IR[25:0]$) deslocado de 2 bits para a esquerda e concatenado com $PC + 4[31:28]$ é enviado ao PC para escrita.



Instrução Tipo-R

Poderia ser pensada na seguinte instrução em linguagem de descrição de hardware (HDL) Verilog:

```
Reg[Memory[PC][15:11]] <= Reg[Memory[PC][25:21]] op Reg[Memory[PC][20:16]];
```

Decompоста nas seguintes etapas usando “variáveis locais”:

```
IR <= Memory[PC];  
A <= Reg[IR[25:21]];  
B <= Reg[IR[20:16]];  
saidaALU <= A op B;  
Reg[IR[15:11]] = saidaALU;  
PC <= PC+4;
```



Controle Multiciclo: 5 Etapas

Considerando:

- 1 acesso à memória, 1 acesso ao banco de registradores ou 1 operação da ULA por ciclo. (determina o ciclo de clock).
- Projeto acionado por transição (valores dos regs não muda até então).

1: Etapa de Busca da Instrução

```
IR <= Memoria[PC];  
PC <= PC + 4;
```

Execução: Ler a Instrução da memória e incrementar PC:

- Ativar os sinais de controle LeMem e Escreve IR;
- Definir PC como origem do endereço: Colocar louD em 0;
- Incrementar PC+4: OrigAALU em 0 (PC para ALU), OrigBALU 01 (4 para ALU) e opALU 00 (soma).
- Armazenar o endereço de instrução incrementado em PC: origPC 00 e ativar EscrevePC.

Controle Multiciclo: 5 Etapas

2: Etapa de Decodificação da Instrução e busca de operandos

```
A <= Reg[IR[25:21]];
B <= Reg[IR[20:16]];
SaidaALU <= PC + (extensão-sinal(IR[15:0]) << 2);
```

Execução: Como não sabemos qual a instrução ainda, podemos fazer ações gerais que não causem problemas depois!

- Acessar o banco de registradores e ler rs e rt e armazenar em A e B.
- Cálculo prévio do endereço de desvio e coloca em SaidaALU: OrigAALU em 0 (PC para ALU), OrigBALU em 11 (offset extendido em ALU) e opALU 00 (soma).



Controle Multiciclo: 5 Etapas

3: Etapa de Execução, cálculo do endereço de memória ou conclusão de desvio

Referência à memória (lw,sw): $SaidaULA = A + extensão-sinal(IR[15-0])$

Execução: Calcular o endereço. OrigAALU em 1 (A na ALU), OrigBALU em 10 (saída ext-sinal na ALU), opALU 00 (soma)

Instrução Tipo-R: $SaidaULA \leq A \text{ op } B$

Execução: os valores de A e B estão prontos! OrigAALU em 1 (A na ALU), OrigBALU em 00 (B na ALU), opALU 10 (lógico-aritmética pelo campo funct)

beq: if (A==B) PC \leq SaidaALU

Execução: o endereço de desvio já está pronto.

- OrigAALU em 1 (A na ALU), OrigBALU em 00 (B na ALU), opALU 01 (subtrai)
- Ativar sinal EscrevePCCond
- Colocar OrigPC em 01 (valor do PC vem da ULA)

Controle Multiciclo: 5 Etapas

3: Etapa de Execução, cálculo do endereço de memória ou conclusão de desvio

Jump: $PC \leq \{PC[31:28], (IR[25:0]), 2b00\}$

Execução: PC é substituído pelo endereço de Jump

- Colocar OrigPC em 10
- Ativar EscrevePC



Controle Multiciclo: 5 Etapas

4: Etapa de acesso à memória ou conclusão de instrução Tipo-R

Referência à memória (lw, sw)::

$\overline{\text{MDR}} \leftarrow \text{Memoria}[\text{SaidaALU}];$ #para load (MDR \rightarrow Registrador de Dados da Memória) ou
 $\text{Memoria}[\text{saidaALU}] \leftarrow B$ # para store

Execução: Em qualquer dos casos o endereço já está na SaidaALU.

- loutD colocado em 1 (Endereço da ALU)
- LeMem ou EscreveMem deve ser ativado

Instrução Tipo-R: $\text{Reg}[\text{IR}[15:11]] \leftarrow \text{SaidaALU}$

Execução: Colocar o resultado no reg destino

- Colocar RegDST em 1 (usar rd como destino)
- Ativar EscreveReg
- Colocar MemparaReg em 0 (saída da ALU na entrada do banco de reg)

Controle Multiciclo: 5 Etapas

5: Etapa de conclusão da Leitura da Memória (lw)

`Reg[IR[20:16]] <= MDR;`

Execução: Escrever o dado do MDR no banco de reg

- Colocar MemparaReg em 1 (resultado da memória)
- Ativar o EscreveReg
- Colocar RegDst em 0 (campo rt é o reg destino)



Resumo do Controle Multiciclo

Etapa	Ação para instruções tipo R	Ação para instruções de acesso à memória	Ação para desvios	Ação para jumps
Busca da instrução	$IR \leftarrow \text{Memória}[PC]$ $PC \leftarrow PC + 4$			
Decodificação da instrução e busca dos registradores	$A \leftarrow \text{Reg}[IR[25:21]]$ $B \leftarrow \text{Reg}[IR[20:16]]$ $\text{SaídaALU} \leftarrow PC + (\text{estende-sinal}(IR[15:0]) \ll 2)$			
Execução, cálculo do endereço, conclusão do desvio/jump	$\text{SaídaALU} \leftarrow A \text{ op } B$	$\text{SaídaALU} \leftarrow A + \text{estende-sinal}(IR[15:0])$	if (A == B) $PC \leftarrow \text{SaídaALU}$	$PC \leftarrow \{PC[31:28], (IR[25:0], 2'b00)\}$
Acesso à memória ou conclusão de instrução tipo R	$\text{Reg}[IR[15:11]] \leftarrow \text{SaídaALU}$	Load: $\text{MDR} \leftarrow \text{Memória}[\text{SaídaALU}]$ ou Store: $\text{Memória}[\text{SaídaALU}] \leftarrow B$		
Conclusão da leitura da memória		Load: $\text{Reg}[IR[20:16]] \leftarrow \text{MDR}$		

FIGURA 5.30 Resumo das etapas realizadas para executar qualquer classe de instrução. As instruções levam de três a cinco etapas de execu-

Exemplo

Q: Considerando um workload de 25% loads, 10% stores, 11% branches, 2% jumps e 52% operações com ALU. Qual a CPI média implementada?

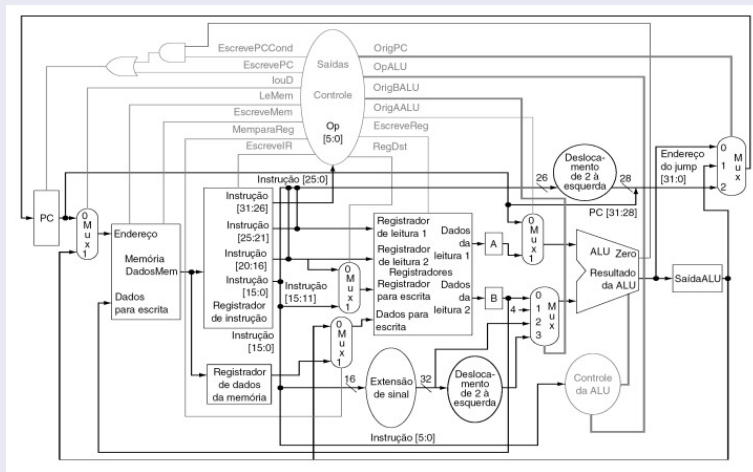
A: $CPI = 0.25 \times 5 + 0.1 \times 4 + 0.52 \times 4 + 0.11 \times 3 + 0.02 \times 3 = 4.12$

Lembrando que no pior caso: $CPI = 5$!

Implementação MIPS III: MIPS Multiciclo

Caminho de Dados Multiciclo: Revisitando!

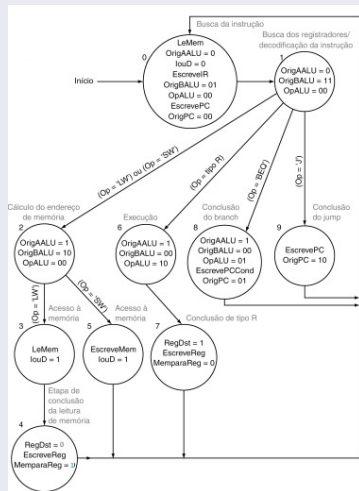
Implementação do controle + controle do PC (PC+4, beq, jump)



Caminho de Dados Multiciclo: Máquina de Estados Finitos

Diagrama de Estados:

- Cada nó do diagrama representa um estado
- A transição entre estados é indicada por arcos
- As condições de disparo de uma transição são associadas aos arcos
- Cada estado corresponde a um ciclo de relógio



Exceções e Interrupções

Exceções são mudanças no fluxo de execução devido a eventos inesperados gerados internamente ao processador.

- instrução inválida
- overflow em operações aritméticas
- ...

Interrupções são mudanças no fluxo devido a eventos externos, tipicamente entrada e saída.

- DMA
- acesso ao barramento
- ...



Exceções e Interrupções

A unidade de controle deve identificar a causa da exceção e armazená-la para posterior tratamento pelo sistema operacional

Além disso, deve informar o endereço onde ocorreu a exceção

MIPS utiliza dois registradores para isso:

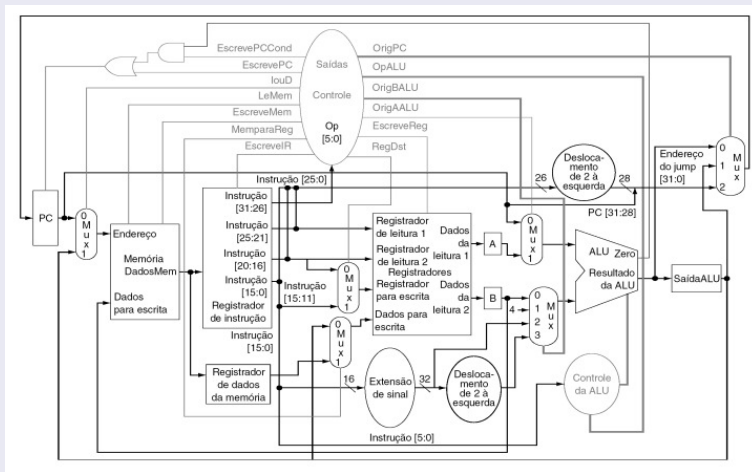
- EPC: endereço da instrução.
- Cause: indicação da causa da exceção.



Implementação MIPS III: MIPS Multiciclo

Caminho de Dados Multiciclo: Revisitando novamente!

Implementação do controle + controle do PC (PC+4, beq, jump)



Caminho de Dados Multiciclo com Controle de Exceção

