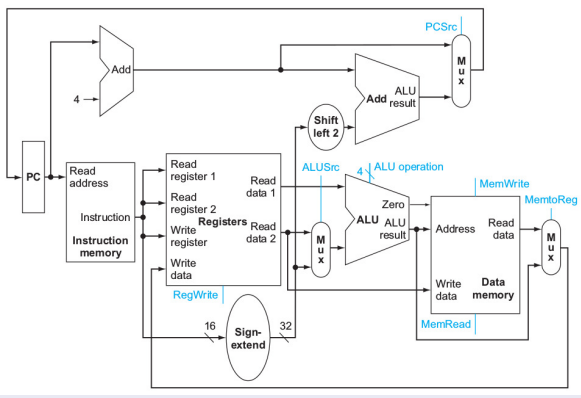
Fundamentos de Arquitetura de Computadores

Tiago Alves

Faculdade UnB Gama
Universidade de Brasília

Unidade Operativa: versão melhorada



MIPS Uniciclo

Foram desenvolvidas, na verdade, três tipos de unidades operativas:

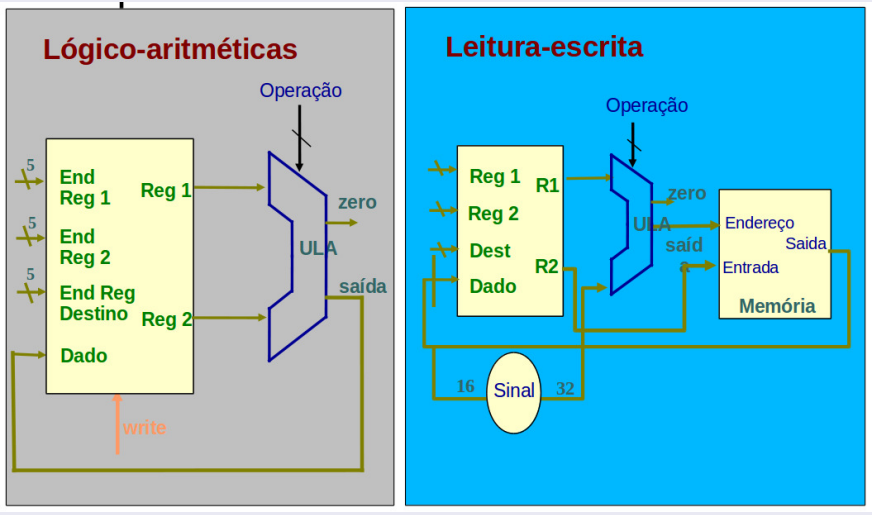
- uma para instruções no formato R (`add`, `sub`, etc.)
- uma para instruções de acesso a memória `lw` e `sw` - formato I
- uma para instruções condicionais (`beq`) - formato J

Na fase de projeto, às vezes precisamos replicar recursos

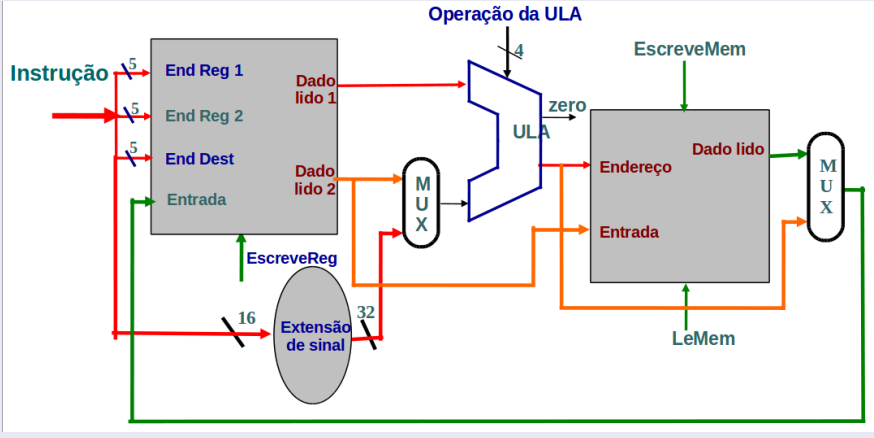
A via de dados mais simples deve-se propor executar as instruções num único período do clock

Isto quer dizer que nenhum dos recursos pode ser usado mais de uma vez por instrução (por quê?)

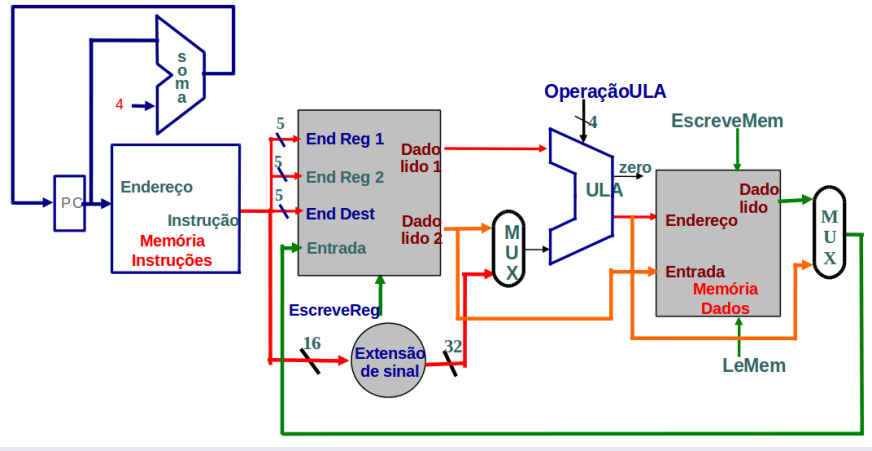
Instruções Lógico/Aritméticas e lw/sw



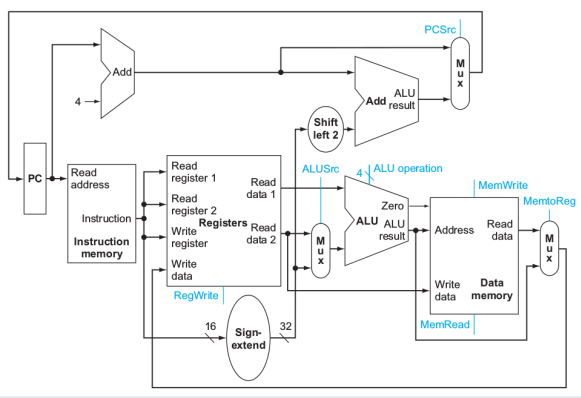
Combinando as unidades estruturais



Adicionando a unidade de Busca/Fetch



Unidade Operativa: versão melhorada II



MIPS Uniciclo

Cada operação requer a utilização adequada dos recursos do MIPS

Ex: `add $t0, $s1, $s2`

- é necessário que os endereços dos operandos $s1 e $s2 sejam enviados ao banco de registradores
- Da mesma maneira, o endereço do registrador destino ($t0) deverá ser informando ao banco de registradores
- Uma vez que o banco de registradores disponibilize os valores de $s1 e $s2, estes deverão ser encaminhados à ULA
- Posteriormente, o resultado deverá ser escrito no banco de registradores (registrador $t0)

Unidade Operativa: versão melhorada

As operações da ULA são controladas por um código de 4 bits:

ALU control lines	Function
0000	AND
0001	OR
0010	add
0110	subtract
0111	set on less than
1100	NOR

- As instruções `lw` e `sw` utilizam a operação de soma da ULA
- As instruções do Tipo R utilizam uma das 5 operações
- A instrução `beq` utiliza a subtração da ULA
- `Nor` não é usada nesta implementação.

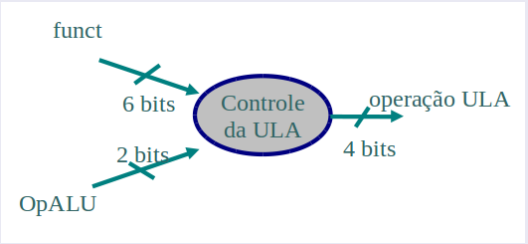
Identificação da Operação

O campo `funct`, de 6 bits (no formato R) indica que tipo de operação aritmética/lógica será executada:

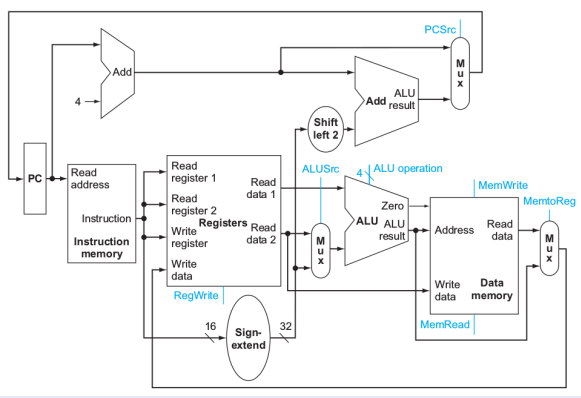
op	rs	rt	rd	shamt	funct
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

Vamos definir 2 bits para identificar cada uma das categorias de instruções (opALU):

- 00: acesso à memória (`lw, sw`) [`add`]
- 01: desvio (`beq, bne`) [`sub`]
- 10: lógico-aritméticas ([`add, sub, and, or, slt`])



Lógica de Controle da ULA



Circuito de Controle da ULA

Projeto do Circuito de Controle da ULA

Instruction opcode	ALUOp	Instruction operation	Funct field	Desired ALU action	ALU control input
LW	00	load word	XXXXXX	add	0010
SW	00	store word	XXXXXX	add	0010
Branch equal	01	branch equal	XXXXXX	subtract	0110
R-type	10	add	100000	add	0010
R-type	10	subtract	100010	subtract	0110
R-type	10	AND	100100	AND	0000
R-type	10	OR	100101	OR	0001
R-type	10	set on less than	101010	set on less than	0111

Circuito de Controle da ULA

Com base nas definições anteriores podemos montar a Tabela Verdade do circuito que gera os 4 bits de controle da ULA

ALUOp		Funct field						
ALUOp1	ALUOp0	F5	F4	F3	F2	F1	F0	Operation
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

Unidade de Controle Uniciclo

A unidade de controle deve, a partir do código da instrução, fornecer os sinais que realizam as instruções na unidade operativa

Entrada da Unidade de Controle

As informações necessárias a execução de uma instrução são retiradas da própria instrução

- O opcode (código de operação) sempre está nos bits `[31-26]=Op[5:0]`
- Os 2 registradores a serem lidos (rs e rt) sempre estão nas posições `[25-21]` e `[20-16]` (tipo-R, `beq` e `sw`)
- O registrador base (rs) para as instruções `lw` e `sw` sempre está especificado nas posições `[25-21]`
- Os 16 bits de deslocamento para as instruções `beq`, `lw` e `sw` estão sempre nas posições `[15-0]`
- O registrador destino está em uma das duas posições:

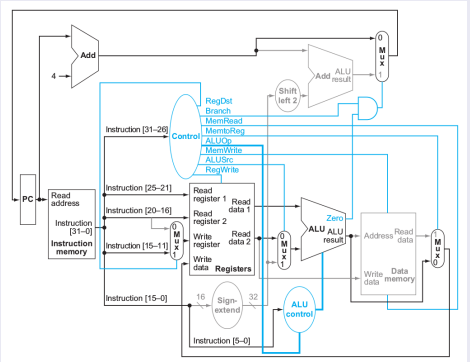
  `[20-16]` para `lw` (registrador rt)

  `[15-11]` para instruções aritméticas/lógicas (registrador rd)

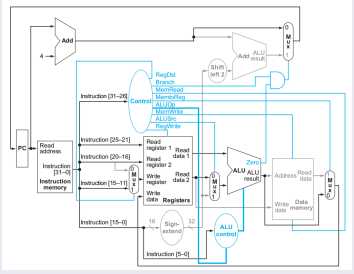
Sinais de Controle

A unidade de controle deve prover:

- sinais para os multiplexadores
- sinais de leitura e escrita para as memórias
- seleção da operação da ULA
- controle do novo endereço a ser carregado no PC, para instruções de salto.



Sinais de Controle

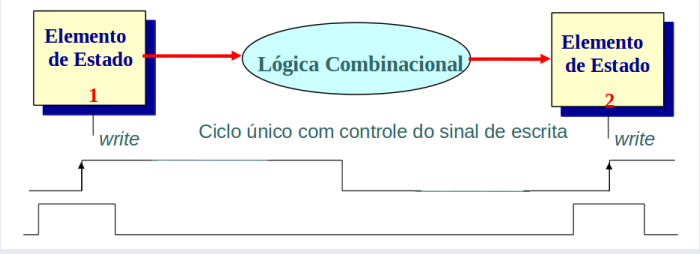


Instruction	RegDst	ALUSrc	Memto-Reg	Reg-Write	Mem-Read	Mem-Write	Branch	ALUOp1	ALUOp0
R-format	1	0	0	1	0	0	0	1	0
lw	0	1	1	1	1	0	0	0	0
sw	X	1	X	0	0	1	0	0	0
beq	X	0	X	0	0	0	1	0	1

Temporização

Todas as tarefas executadas em 1 período do clock

Elementos de estado alteram seu valor apenas na borda de subida do clock



Quais são os elementos de estado? Quando são escritos?

Problemas com MIPS Uniciclo

Período do relógio determinado pelo caminho mais longo:

- instrução `lw`:

  leitura da instrução

  leitura do registrador de base, extensão de sinal

  cálculo do endereço

  leitura do dado da memória

  escrita em registrador

Qual o impacto dessa estratégia se houvesse operações com precisão dupla?

Casos mais graves:

- Viola o princípio de tornar o caso comum rápido ("filosofia" de projeto)
- Unidades funcionais duplicadas (maior espaço em chip!)

Análise de Desempenho Uniciclo

Supondo os seguintes tempos de execução das unidades operativas:

- Acesso a memória: 200 ps
- ULA e somadores: 100 ps
- Acesso ao banco de registradores (leitura ou escrita): 50 ps
- Multiplexadores, Unidade de Controle, acesso ao PC, Unidade de Extensão de Sinal e fios considerados sem atraso (Impossível!)

Qual o impacto dessa estratégia se houvesse operações com precisão dupla?

Qual das seguintes implementações seria mais rápida e quanto?

1. Implementação onde toda instrução é feita em 1 ciclo de clock de duração fixa.
2. Implementação onde toda instrução é feita em 1 ciclo de clock de duração somente o necessário (exemplo ideal ? CNTP!)

Para comparação consideremos um workload composto de:

25% loads, 10% stores, 45% ALU, 15% desvios condicionais, 5% jumps.

Análise de Desempenho Uniciclo

Tempo Execução = Contagem de Instruções x CPI x Período de clock

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 Escrita registrador	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

Tempo médio:

Uniciclo fixo: 600ps

Uniciclo variável: 600x0.25+550x0.1+400x0.45+350x0.15+200x0.05=447.5ps

Fator de desempenho: n=600/447.5 = 1.34