O Processador: Caminho de Dados e Unidade de Controle

Ivanildo Miranda
Octávio Augusto Deiroz

Caminho de Dados e Unidade de Controle

- **Datapath -> Caminho que os dados e instruções percorre, de acordo com os sinais gerados pela unidade de controle**
- **Controle -> Componente do processador que comanda o datapath, memória e dispositivos de E/S de acordo com as instruções de um programa**
- **Independente da classe da instrução, as duas primeiras etapas para sua execução são as mesmas:**
- **Enviar o PC para a memória e buscar a instrução**
- **Ler um ou dois registradores (usando o campo da instrução, para selecionar os registradores a serem lidos)**

Caminho de Dados e Unidade de Controle

- Os passos seguintes dependem da classe da instrução (referência à memória, lógica-aritmética e desvios) e estes passos são bastantes semelhantes e independem do opcode
- Por exemplo, todas as instruções, independente da classe utilizam a ULA após a leitura de um registrador. Para uma instrução de referência à memória, utiliza para cálculo do endereço, lógica-aritmética para execução e desvios para comparação
- Após a utilização da ULA, os passos são diferentes para as diferentes classes.

Caminho de Dados e Unidade de Controle

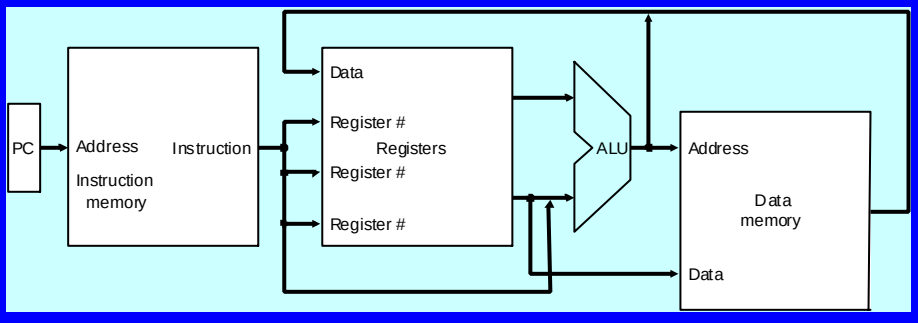


Figura 5.1 - Implementação do MIPS – visão em alto nível

Caminho de Dados e Unidade de Controle

Revisão/Convenções adotadas

- **Sinal lógico alto – *asserted***
- **Sinal que pode ser logicamente alto - *assert***
- **Elementos combinacionais . Exemplo: ULA**
- **Elementos de estado è Exemplo: Registradores e Memória**
- **Sinal de Clock è usado para determinar quando se pode escrever em um elemento de estado.**

A leitura pode ser a qualquer momento

- **Metodologia de sincronização : sincroniza o elemento de estado para a permissão de leitura e de escrita. Porque é necessário ?**

Caminho de Dados e Unidade de Controle

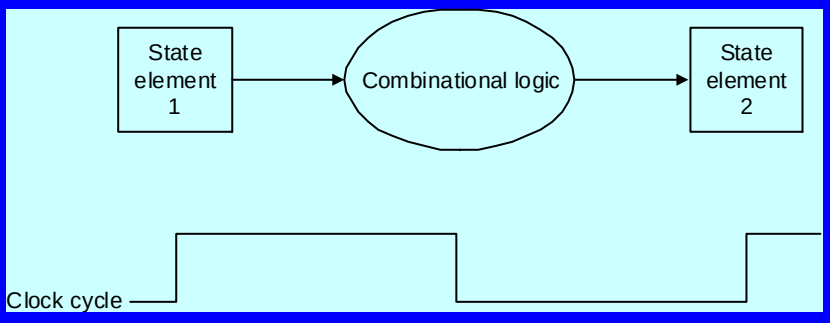


Figura 5.2 – Lógica combinacional, elemento de estados e clock (sensível à subida).

Caminho de Dados e Unidade de Controle

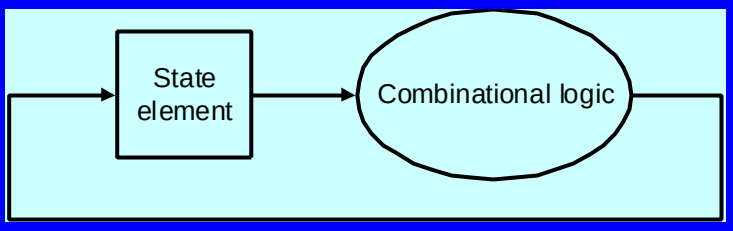


Figura 5.3 – A metodologia edge-triggered permite a um elemento de estado ler e escrever no mesmo período de clock.

Caminho de Dados e Unidade de Controle
Datapath

OBS.: Primeiro implementaremos um Datapath utilizando apenas um clock com ciclo grande.
Cada instrução começa a ser executada em uma transição e acaba na próxima transição do clock.
Na prática isto não é factível, pois temos instruções de diferentes classes e portanto de diferentes números de ciclos de clock.

Para construir um Datapath:

- **Um lugar para armazenar as instruções do programa**
- **Memória de instruções**
- **Um lugar para armazenar o endereço da instrução a ser lida na memória Program Counter – PC**
- **Somador para incrementar o PC para que ele aponte para a próxima instrução**

Caminho de Dados e Unidade de Controle
Datapath

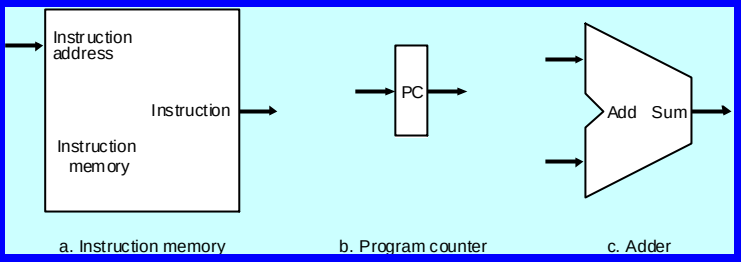
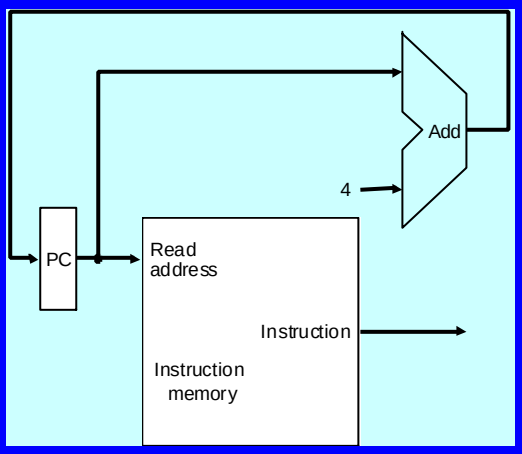


Figura 5.4 – Elementos necessários a armazenar e acessar informações mais um somador para calcular o endereço do próximo estado.

Caminho de Dados e Unidade de Controle
Datapath

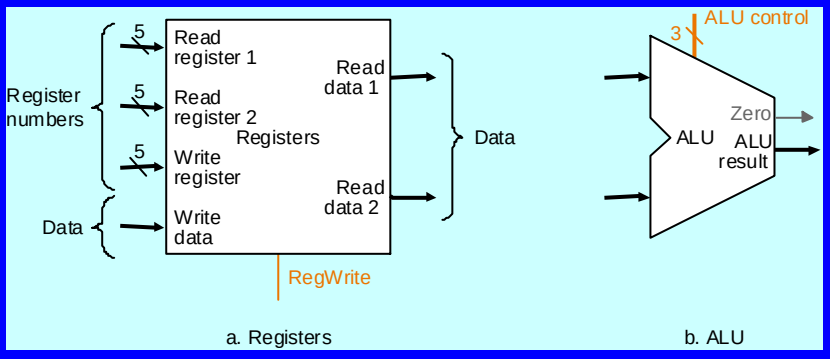


Caminho de Dados e Unidade de Controle
Datapath

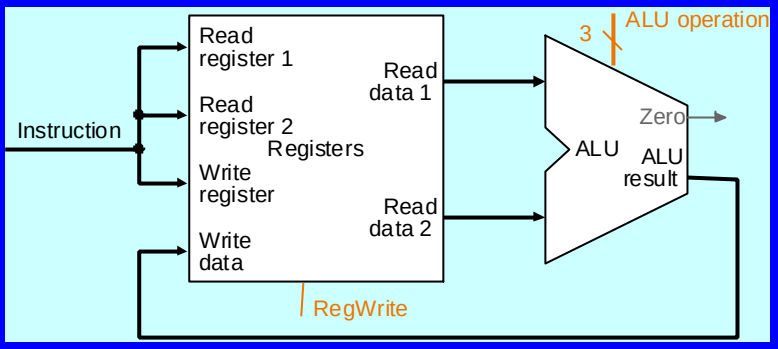
Instruções R-type

- Instruções aritméticas e lógicas è add, sub, slt
- Estrutura de registradores (32) chamada de register file que é um conjunto de registradores que podem ser acessados (lidos ou escritos) especificando seu número.
- Nele se encontra o registrador de estado da máquina
- Instruções de formato R tem 3 operandos registradores (add $t1,$t2,$t3)
- necessidade de ler 2 dados do register file e escrever um dado nele, para cada instrução
- Para ler um dado do register file é preciso de uma entrada (número do do registrador) e uma saída (o dado lido)
- Para escrever um dado no register file, são necessárias duas entradas: o número do registrador e o dado a ser escrito
- Para escrever è sinal de controle (RegWrite)
- A ULA é controlada por um sinal de controle (ALU control)

Caminho de Dados e Unidade de Controle



Caminho de Dados e Unidade de Controle

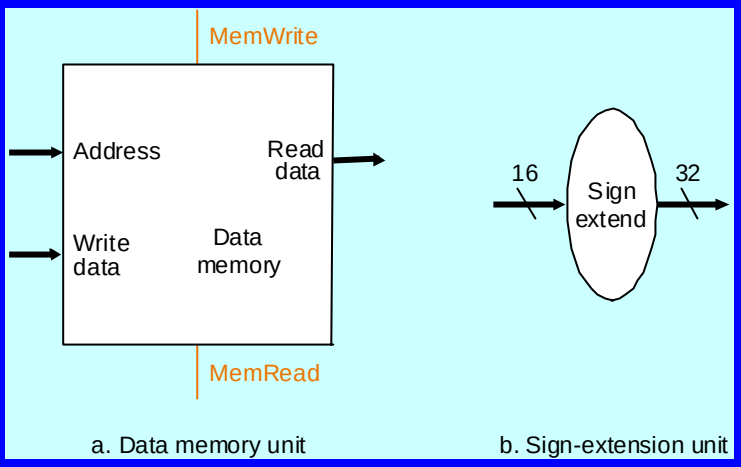


Caminho de Dados e Unidade de Controle

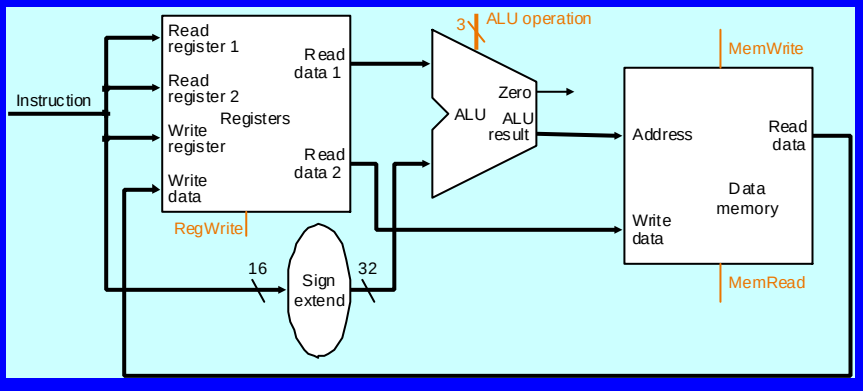
Instruções LOAD e STORE

- **lw $t1,offset_value($t2) e sw $t1,offset_value,($t2)**
- **Endereço de memória = value_offset + $t2**
- **Value_offset èoffset sinalizado de 16 bits**
- **É preciso de um register file e uma ULA**
- **Unidade que transforme valor de 16 bits sinalizado em um valor de 32 bits**
- **Unidade de memória de dados com controle de leitura (MemRead) e escrita (MemWrite)**

Caminho de Dados e Unidade de Controle



Caminho de Dados e Unidade de Controle

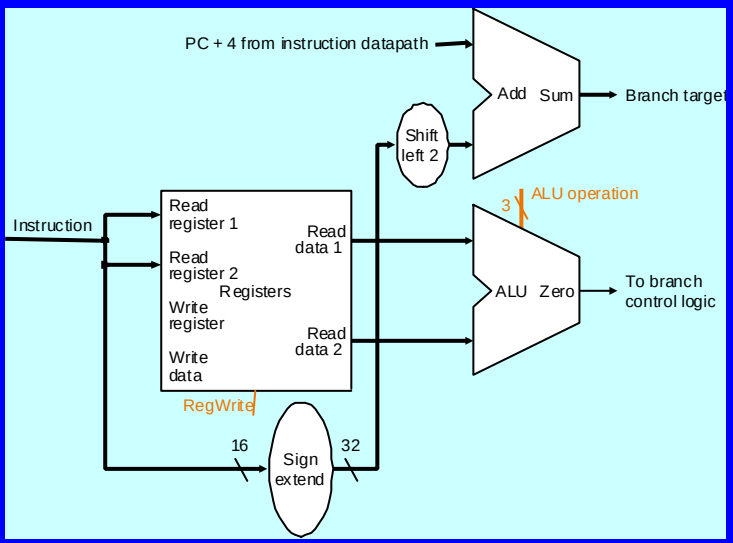


Caminho de Dados e Unidade de Controle

Instrução beq

- beq $t1,$t2,offset è2 registradores que são comparados e um offset de 16 bits usado para calcular o endereço relativo, alvo do branch
- A base para o cálculo do endereço alvo de branch é o endereço da próxima instrução em relação à instrução branch
- O campo offset é deslocado de 2 bits para aumentar o alcance do desvio (multiplicado por 4)
- Além do cálculo do endereço do branch, deve-se determinar qual endereço será escolhido, o do branch (taken) ou o armazenado em PC (not taken) dependendo do resultado da comparação
- OBS.: Instrução jump èos 28 bits menos significativos de PC são substituídos pelos 26 bits do imediato, deslocado de 2 bits (X 4).

Caminho de Dados e Unidade de Controle



Caminho de Dados e Unidade de Controle

Datapath Geral – Instruções de memória + Instruções do tipo R

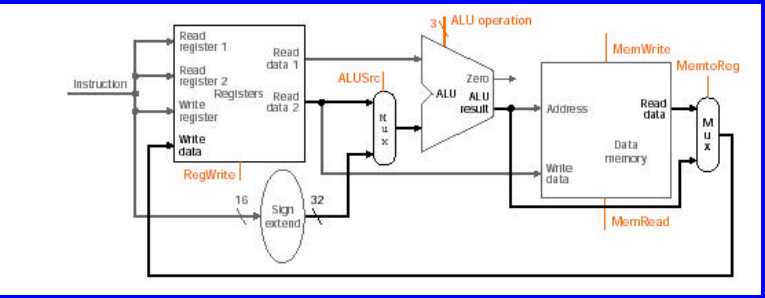
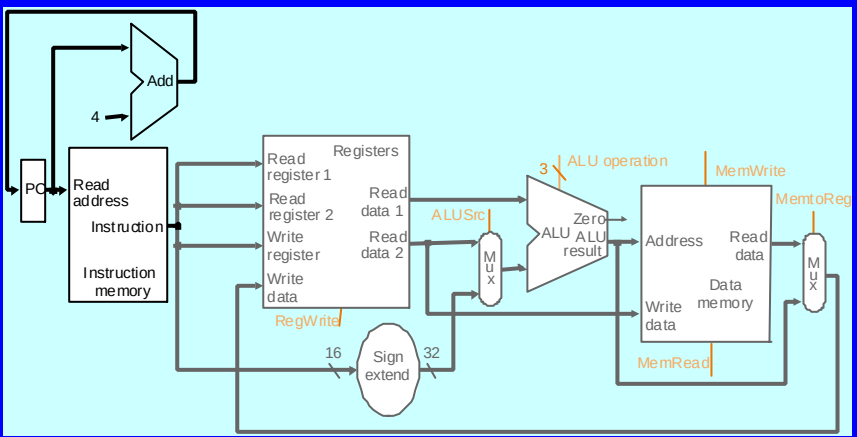


Fig. 5.11

Caminho de Dados e Unidade de Controle

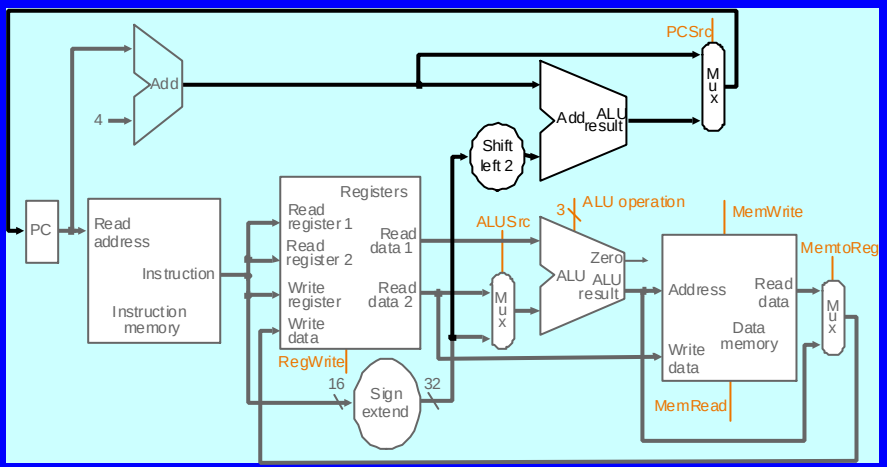


Caminho de Dados e Unidade de Controle

- Controle da ULA

Sinal de Controle da ULA	Função
000	AND
001	OR
010	ADD
110	SUB
111	SLT

Caminho de Dados e Unidade de Controle

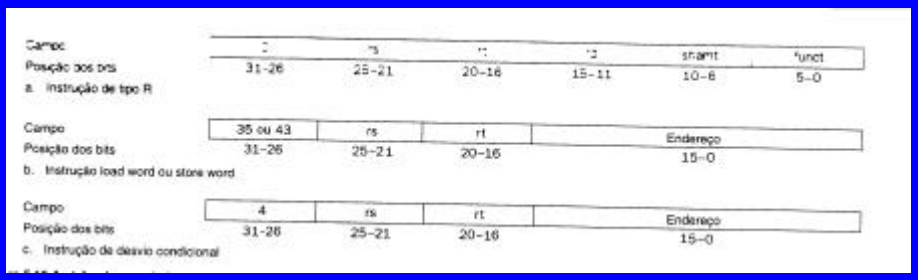


Caminho de Dados e Unidade de Controle

Código de operação da instrução	UALOp	Operação da instrução	Campo de Função	Operação desejada da UAL	Entrada de controle da UAL
LW	00	load word	XXXXXX	soma	010
SW	00	store word	XXXXXX	soma	010
Branch equal	01	branch equal	XXXXXX	subtração	110
Tipo R	10	add	100000	soma	010
Tipo R	10	subtract	100010	subtração	110
Tipo R	10	AND	100100	and	000
Tipo R	10	OR	100101	or	001
Tipo R	10	set less than	101010	set less than	111

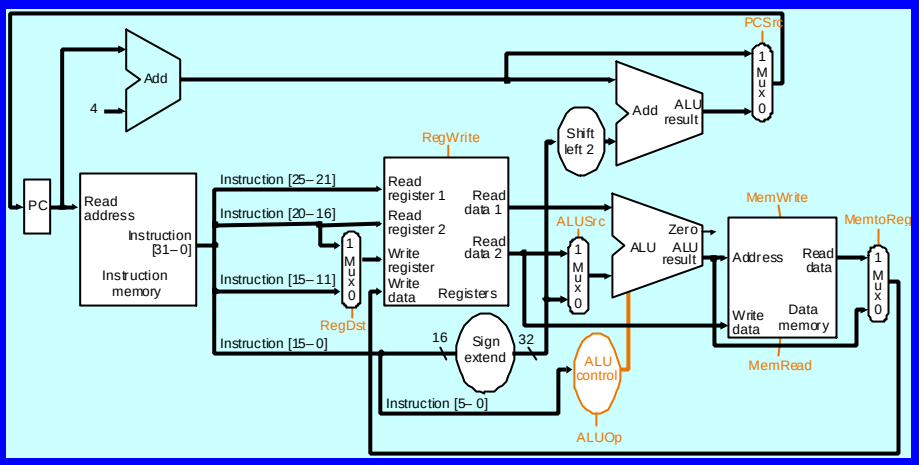
UALOp		Campo de função						Operação da UAL
UALOp1	UALOp2	F5	F4	F3	F2	F1	F0	
0	0	X	X	X	X	X	X	010
X	1	X	X	X	X	X	X	110
1	X	X	X	0	0	0	0	010
1	X	X	X	0	0	1	0	110
1	X	X	X	0	1	0	0	000
1	X	X	X	0	1	0	1	001
1	X	X	X	1	0	1	0	111

Caminho de Dados e Unidade de Controle



- **O campo opcode -> bits 31-26**
- **Os dois registradores a serem lidos : rs e rt -> bits 25-21 e 20-16**
- **O registrador base (load e store) : bits 15-0**
- **O valor a ser guardado no PC que pode vir do cálculo de um endereço de salto ou simplesmente do PC + 4.**
- **O registrador destino (a ser escrito), que deverá ser selecionado dentre 2 opções, o que requer um multiplexador: para um load -> bits 20-16 (rt)**

Caminho de Dados e Unidade de Controle

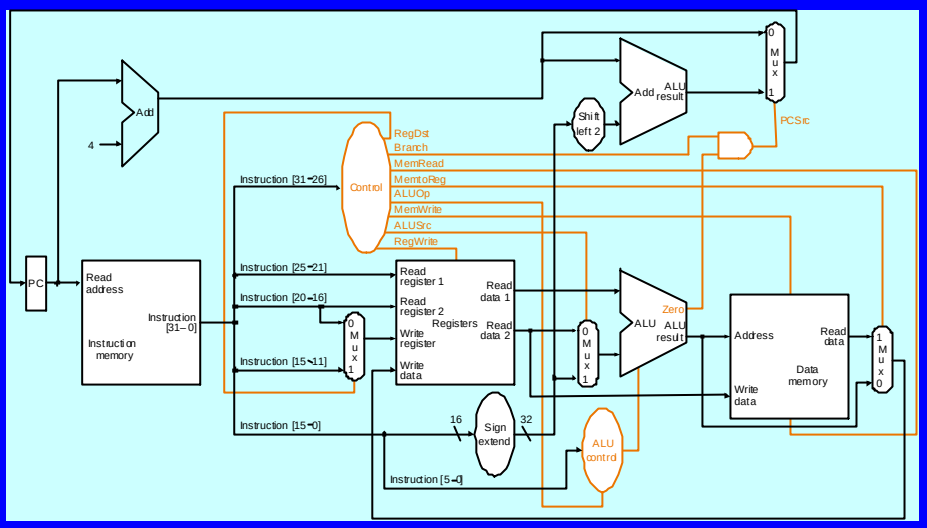


Caminho de Dados e Unidade de Controle

- Sinais de controle – Tabela da figura 5.18

Signal name	Effect when deasserted	Effect when asserted
RegDst	The register destination number for the Write register comes from the rt field (bits 20–16).	The register destination number for the Write register comes from the rd field (bits 15–11).
RegWrite	None	The register on the Write register input is written with the value on the Write data input.
ALUSrc	The second ALU operand comes from the second register file output (Read data 2).	The second ALU operand is the sign-extended, lower 16 bits of the instruction.
PCSrc	The PC is replaced by the output of the adder that computes the value of PC + 4.	The PC is replaced by the output of the adder that computes the branch target.
MemRead	None	Data memory contents designated by the address input are put on the Read data output.
MemWrite	None	Data memory contents designated by the address input are replaced by the value on the Write data input.
MemtoReg	The value fed to the register Write data input comes from the ALU.	The value fed to the register Write data input comes from the data memory.

Caminho de Dados e Unidade de Controle

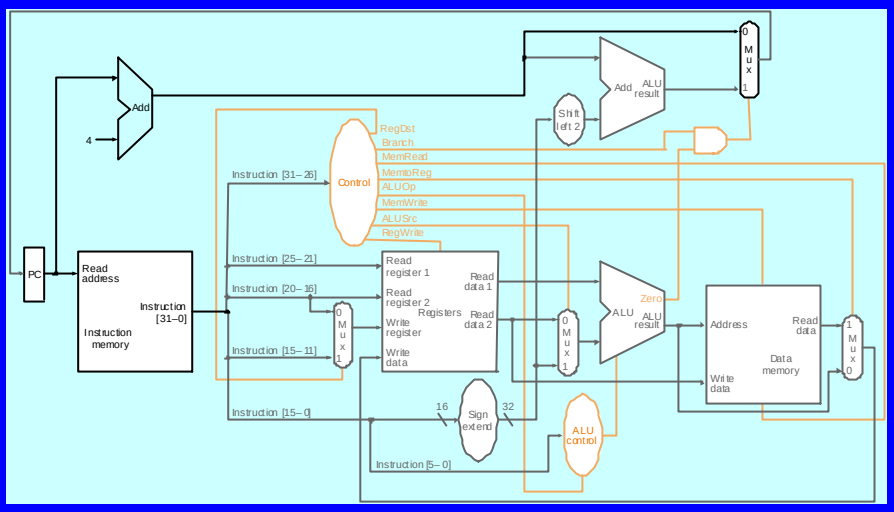


Caminho de Dados e Unidade de Controle

Instrução	RegDst	UALFonte	MemParaReg	EscReg	LerMem	EscMem	DvC	UALOp1	UALOp0
Formato R	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

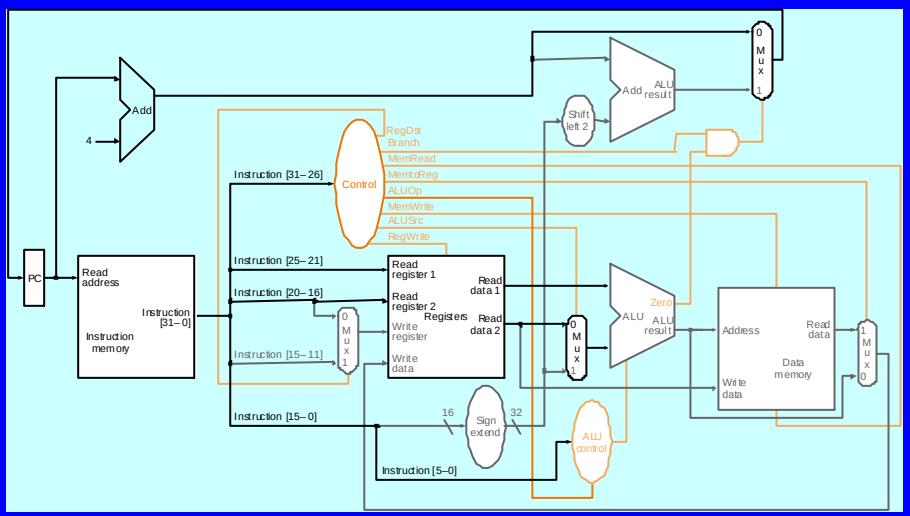
- Operação do Datapath
- instrução R-type -> add $t1,$t2,$t3
- fetch da instrução
- leitura de $t2 e $t3
- operação da ULA com os dados lidos
- resultado da ULA escrito em $t1

Caminho de Dados e Unidade de Controle



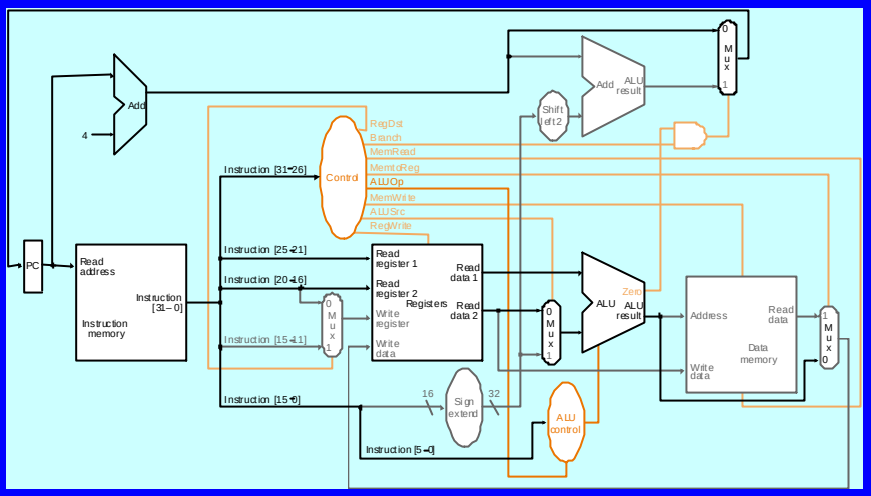
Caminho de Dados e Unidade de Controle

Fig. 5.22 Leitura dos Registradores e Decodificação – Instruções do tipo R



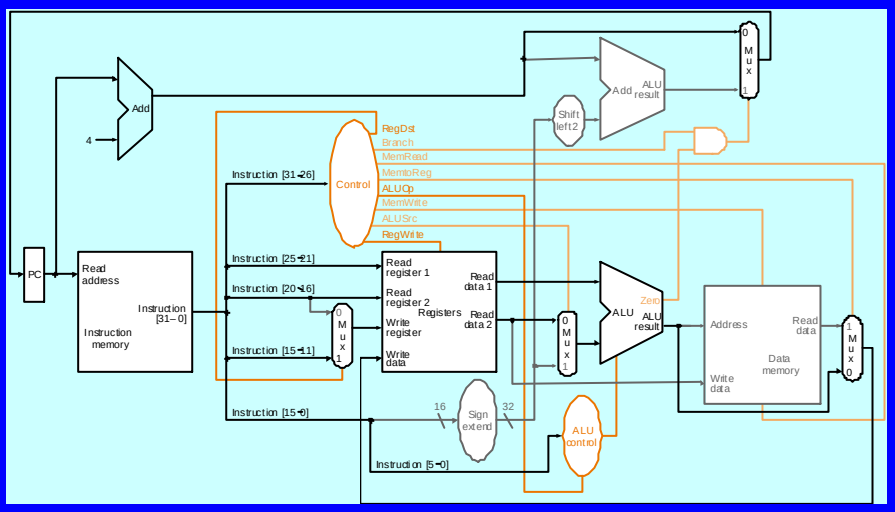
Caminho de Dados e Unidade de Controle

Fig. 5.23 Operação da ULA – Instrução tipo R



Caminho de Dados e Unidade de Controle

Fig. 5.24 Escrita no registrador – Instrução tipo R



Caminho de Dados e Unidade de Controle

Instrução load word

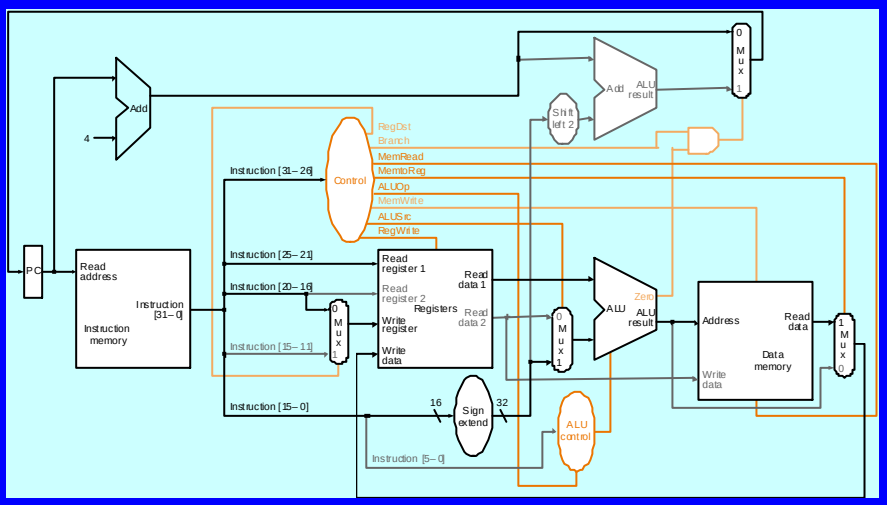
lw $t1, offset($t2)

Instruction Fetch

- **$t2 é lido**
- **ULA calcula a soma do valor lido e o imediato de 16 bits**
- **O resultado é usado como endereço da memória de dados**
- **O dado da memória de dados é escrito no register file**

Caminho de Dados e Unidade de Controle

Fig. 5.25 Operação de lw com um esquema simples de controle



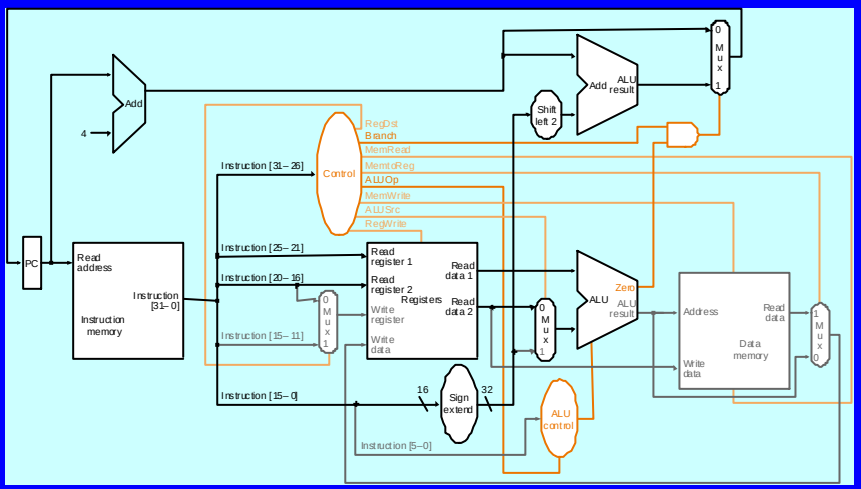
Caminho de Dados e Unidade de Controle

Instrução de branch

- **beq $t1,$t2,offset**
- **Fetch da instrução**
- **$t1 e $t2 são lidos**
- **ULA faz subtração dos valores lidos. PC+4 é adicionado ao imediato de 16 bits, deslocado de 2 bits**
  resultado é o endereço do desvio
- **A saída Zero é usada para decidir qual endereço será armazenado em PC**

Caminho de Dados e Unidade de Controle

Fig. 5.26 – Datapath para a instrução beq



Caminho de Dados e Unidade de Controle

Entrada ou saída	Nome do sinal	Formato R	lw	sw	beq
Entradas	Op5	0	1	1	0
	Op4	0	0	0	0
	Op3	0	0	1	0
	Op2	0	0	0	1
	Op1	0	1	1	0
	Op0	0	1	1	0
Saídas	RegDst	1	0	X	X
	UALFonte	0	1	1	0
	MemParaReg	0	1	X	X
	EscReg	1	1	0	0
	LerMem	0	1	0	0
	EscMem	0	0	1	0
	DvC	0	0	0	1
	UALOp1	1	0	0	0
	UALOp0	0	0	0	1

Caminho de Dados e Unidade de Controle

Instrução de JUMP

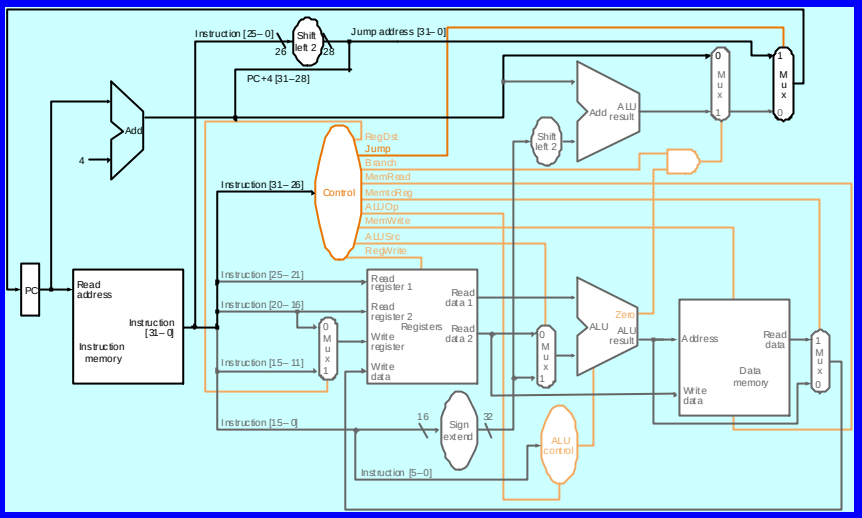
2	endereço

31 - 26 25 - 0

PC é formado pelos 4 bits mais significativos + 26 bits do imediato + 00 (deslocamento de 2)

Caminho de Dados e Unidade de Controle

Fig 5.29 Datapath para instrução de JUMP



Caminho de Dados e Unidade de Controle

- Implementação em um ciclo de clock não usada
- Funciona corretamente mas não é eficiente
- Para single-cycle o ciclo do clock deve ter o mesmo comprimento para todas as instruções CPI = 1 para isto, o ciclo de clock é determinado pelo maior caminho no datapath da máquina (instrução de load que usa 5 unidades funcionais em série: instruction memory, register file, ULA, data memory e register file)

Exemplo.
Sejam os seguintes tempos de operação:
- unidade de memória : 2 ns
- ULA e somadores: 2ns
- register file (read ou write) : 1 ns
- unidade de controle, multiplexadores, acesso ao PC, circuito para extensão do sinal e linhas não tem atraso, quais das seguintes implementações seria mais rápida e quanto ?

Caminho de Dados e Unidade de Controle

1. Uma implementação na qual cada instrução opera em um ciclo de clock de tamanho fixo
2. Uma implementação onde cada instrução é executada usando clock de tamanho de ciclo variável (exatamente do tamanho necessário para a execução da respectiva instrução è também não utilizado na prática)

Para comparar a performance, assuma que o seguinte conjunto de instruções : 24% de loads, 12% de store, 44% instruções tipo R, 18% de branches e 2% de jumps.

Solução:
tempo de execução CPU = número de instruções X CPI X período do clock
CPI = 1 è tempo de execução CPU = número de instruções X período do clock

Temos que encontrar o período do clock para as duas implementações, pois o número de instruções e a CPI são iguais para ambas implementações.

Caminho de Dados e Unidade de Controle

O caminho crítico para cada classe de instrução é:

Classe da Instrução	Unidades funcionais envolvidas				
R-TYPE	Inst. fetch	Reg. access	ALU	Reg. access	
LW	Inst. fetch	Reg. access	ALU	Mem access	Reg access
SW	Inst. fetch	Reg. access	ALU	Mem access	
BRANCH	Inst. fetch	Reg. access	ALU		
JUMP	Inst. fetch				

Usando o caminho crítico, podemos computar o comprimento do ciclo de clock necessário para cada classe de instrução:

Classe da Instrução	Memória Instrs.	Leitura Regs.	ALU oper.	Memória Dados	Escrita Regs.	Total
R-TYPE	2	1	2	0	1	6 ns
LW	2	1	2	2	1	8 ns
SW	2	1	2	2		7 ns
BRANCH	2	1	2			5 ns
JUMP	2					2 ns

Caminho de Dados e Unidade de Controle

O período do clock para a máquina com um único clock é determinado pela maior instrução 8ns
A máquina com clock variável, terá seus períodos de clock variando entre 2ns e 8ns. O clcock médio será:

Tempo de clock da CPU = 8 X 24% + 7 X 12% + 6 X 44% + 5 X 18 % + 2 X 2% = 6.3ns
CPU perfomancevc / CPU perfomancesc =
= tempo de execução CPUsc / tempo de execução CPUvc =
= IC X período de clock da CPUsc / IC X período de clock da CPUvc =
= período de clock da CPUsc / período de clock da CPUvc =
= 8 / 6.3 = 1.27
Exemplo
- FP unit 8ns para add e 16ns para mul
- todos os loads tem o mesmo tempo e são 33% das instruções
- todos os stores tem mesmo tempo e são 21% das instruções
- instruções tipo R são 27 % das instruções
- Branches são 5 % e jumps 2% das instruções
- FP add e sub tem o mesmo tempo e juntos tem 7% das instruções
- FP mul e div tem o mesmo tempo e &5 das instruções

Caminho de Dados e Unidade de Controle

Solução
CPU perfomancevc / CPU perfomancesc =
= tempo de execução CPUsc / tempo de execução CPUvc
Para a máquina de único clock è período do clock = ciclo da instrução FP mul (mais longa) è 2+1+16+1 = 20ns
Para a máquina de clock variável:
período do clock CPU = 8 X 31% + 7 X 21% + 6 X 27% + 5 X 5% = 7.0 ns
Portanto:
CPU perfomancevc / CPU perfomancesc =
= tempo de execução CPUsc / tempo de execução CPUvc = 20/7 = 2.9