Arquitectura de Computadores I 2007/08

1º Semestre de 2007/2008

Bernardo Cunha, José Luís Azevedo, Arnaldo Oliveira

Arquitectura de Computadores I 2007/08

Aula 18

Desenho da unidade de controlo da ALU

A unidade de controlo principal

Exemplos de funcionamento do *datapath* c/ unidade de controlo

Suporte da instrução "Jump"

Desenho da unidade de controlo da ALU

A diversidade de acções suportada pela ALU e o facto de a mesma ter de ser controlada por entidades diversas justifica a existência de uma unidade de controlo específica para este elemento operativo.

As instruções básicas que fazem uso da ALU são:

- *Load e store* – para calcular o endereço da memória externa
- *Branch if equal* – para calcular a condição de salto condicional
- Aritméticas e lógicas – para efectuar a operação seleccionada pelo campo *funct* da instrução

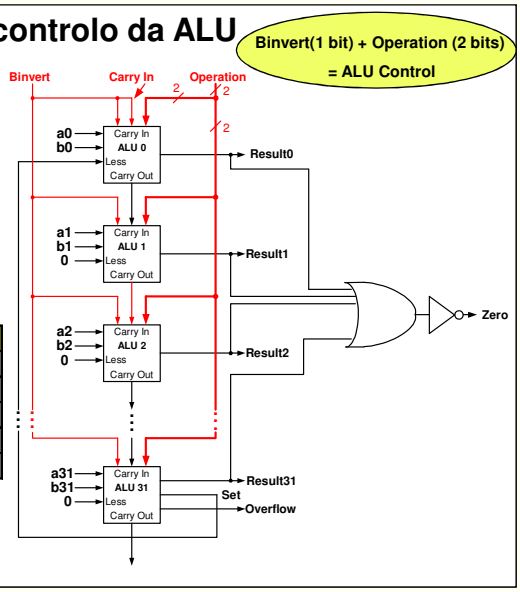
Uma vez que a selecção da operação efectuada pelas instruções aritméticas e lógicas provém da própria instrução, faz sentido que exista um outro nível de controlo que determine a operação da ALU em função do tipo de instrução.

Desenho da unidade de controlo da ALU

Os sinais de controlo directo da ALU podem ser reduzidos a três, uma vez que os sinais ***Binvert*** e ***Carry In*** podem ser combinados num só.

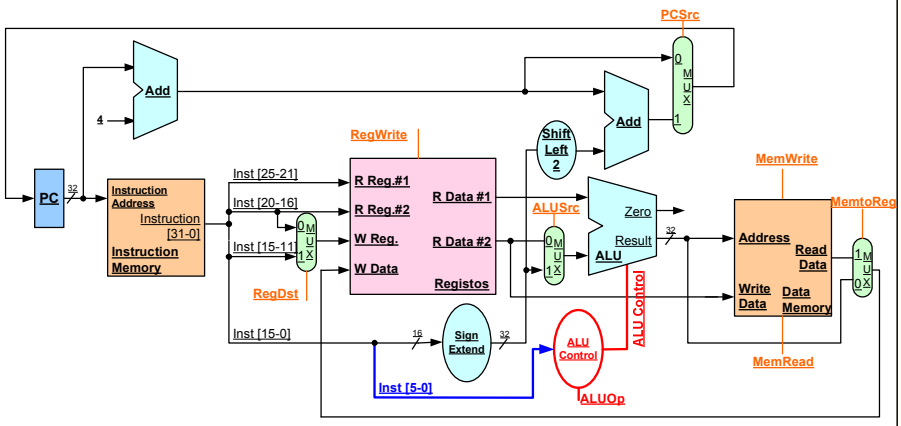
ALU Control	Function
0 0 0	And
0 0 1	Or
0 1 0	Add
1 1 0	Subtract
1 1 1	Set-on-less-than

Bit "Binvert"



Desenho da unidade de controlo da ALU

O *datapath* com a unidade de controlo da ALU seria assim:



Desenho da unidade de controlo da ALU

A relação entre o tipo de instruções, o campo "funct", a operação efectuada pela ALU e os sinais de controlo da mesma podem assim ser resumidos pela seguinte tabela:

OpCode	funct	Operation	ALU Action	ALUOp	ALU Control
100011 ("lw")	xxxxxx	load word	**add**	**00**	010
101011 ("sw")	xxxxxx	store word	**add**	**00**	010
000100 ("beq")	xxxxxx	branch equal	**subtract**	**01**	110
000000 (R-Type)	100000	add	**add**	**10**	010
000000 (R-Type)	100010	subtract	**subtract**	**10**	110
000000 (R-Type)	100100	AND	**and**	**10**	000
000000 (R-Type)	100101	OR	**or**	**10**	001
000000 (R-Type)	101010	set-on-less-than	**set-on-less-than**	**10**	111

Desenho da unidade de controlo da ALU

A unidade de controlo pode assim ser sintetizada a partir de uma tabela de verdade que identifique o valor dos três sinais de controlo em função dos 6 bits do campo "*funct*" e dos dois bits do *ALUOp*

F5	F4	F3	F2	F1	F0	ALUOp1	ALUOp2	ALU Control
X	X	X	X	X	X	0	0	0 1 0
X	X	X	X	X	X	0	1	1 1 0
X	X	0	0	0	0	1	X	0 1 0
X	X	0	0	1	0	1	X	1 1 0
X	X	0	1	0	0	1	X	0 0 0
X	X	0	1	0	1	1	X	0 0 1
X	X	1	0	1	0	1	X	1 1 1

ALUControl0 = ALUOp1 . (F3 + F0)

ALUControl1 = ALUOp1\ + F2\

ALUControl2 = ALUOp2 + ALUOp1 . F1

Desenho da unidade de controlo principal

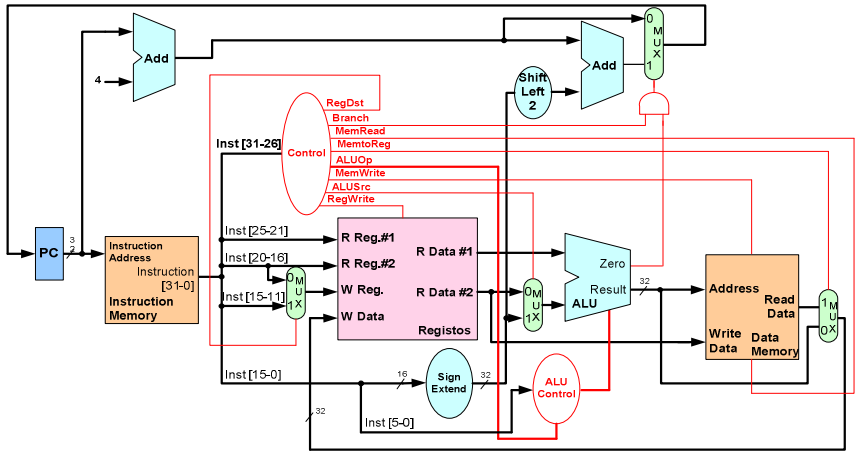
A síntese da unidade de controlo principal do nosso CPU simplificado apoia-se na observação de um conjunto de factos que decorrem da forma como são codificadas as instruções do MIPS:

- O campo ***op*** (*Operation code*) está situado nos bits 31-26 de todas as instruções
- Os dois registos que devem ser lidos (nas instruções em que tal se aplica), surgem sempre nos bits 25-21(rs) e 20-16(rt).
- Nas instruções load/store, o registo base de endereçamento está sempre nos bits 25-21(rs)
- As constantes e/ou *offsets* (nos casos em que se refere a um endereço) surgem sempre nos bits 15-0 da instrução
- O registo destino (quando se aplique) pode aparecer em dois campos: nos bits 20-16 (l*oad*), ou nos bits 15-11(aritméticas e lógicas

Desenho da unidade de controlo principal

- Alguns dos elementos de estado presentes no *datapath* são acedidos sincronamente em todos os ciclos de relógio. É o caso do PC e da memória de instruções. Nestes casos não há necessidade de explicitar um sinal de controlo
- Outros, porém, podem ser lidos ou escritos casuisticamente, dependendo da instrução que estiver a ser executada, embora sempre de forma síncrona. Para estes últimos é necessário explicitar os respectivos sinais de controlo.
- No *datapath*, por outro lado, existem dispositivos combinatórios que fazem o agulhamento da informação (multiplexers). Para estes é igualmente necessário definir os respectivos sinais de controlo.

O aspecto do nosso *datapath*, agora associado ao respectivo elemento de controlo será:



Desenho da unidade de controlo principal

Teremos assim de especificar um total de oito sinais de controlo, para além do ALUop que já antes definíramos. São eles:

Sinal	Efeito quando não activo	Efeito quando activo
MemRead	Nenhum	O conteúdo da memória de dados no endereço indicado é apresentado à saída
MemWrite	Nenhum	O conteúdo do registo de memória de dados cujo endereço é fornecido é substituido pelo valor apresentado à entrada
ALUSrc	O segundo operando da ALU provém da segunda saída do *File Register*	O segundo operando da ALU provém dos 16 bits menos significativos da instr. após expansão do sinal
RegDst	O endereço do registo destino provém do campo **rt**	O endereço do registo destino provém do campo **rd**
RegWrite	Nenhum	O registo indicado no endereço de escrita é alterado pelo valor presente na entrada de dados
MemtoReg	O valor apresentado para escrita no registo destino provém da ALU	O valor apresentado na entrada de dados dos registo internos provém da memória externa
PCSrc	O PC é substituido pelo seu valor actual mais 4	O PC é substituido pelo resultado do somador que calcula o endereço target do *branch* condicional
Branch	Nenhum	Indica que a intrução é um branch condicional

Desenho da unidade de controlo principal

A tabela de verdade respectiva em função do tipo de instrução, será:

Instrução	Opcode	RegDst	ALU Src	Memto Reg	Reg Write	Mem Read	Mem WRite	Branch	ALUOp1	ALUOp2
R - Format	000000	1	0	0	1	0	0	0	1	0
lw	010011	0	1	1	1	1	0	0	0	0
sw	011011	X	1	X	0	0	1	0	0	0
beq	000100	X	0	X	0	0	0	1	0	1

Note-se que, tal como já aconteceu com a unidade de controlo da ALU, também a unidade de controlo principal é meramente combinatória.

Análise do funcionamento do *datapath*. Exemplos.

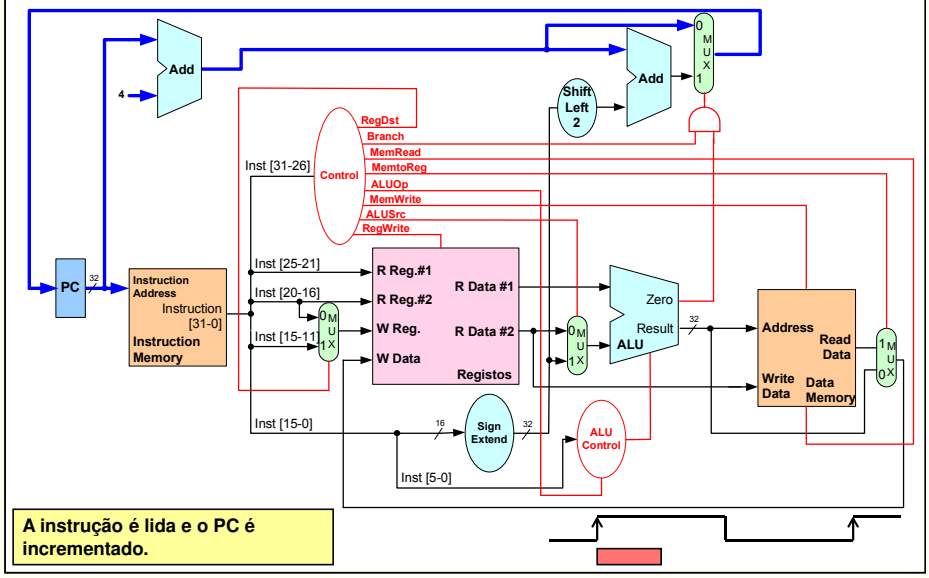
Embora a execução de qualquer uma das instruções suportadas ocorra no intervalo de tempo correspondente a um único ciclo de relógio, poderemos, para simplificar a análise, admitir que a utilização dos vários elementos operativos é sequencial e decorre ao longo do seguinte conjunto de operações:

- *Fetch* de uma instrução e cálculo do endereço da próxima instrução.
- Leitura de dois registos do *File Register*.
- A ALU opera sobre dois valores (a fonte dos valores a operar depende do tipo de instrução que estiver a ser executada).
- O resultado da operação efectuada na ALU é escrita no *File Register* (**R-Type**) ou na memória (***sw***), ou é efectuada uma leitura da memória de dados (***lw***).
- O valor lido da memória de dados é escrito no *File Register*.

Exemplo 1

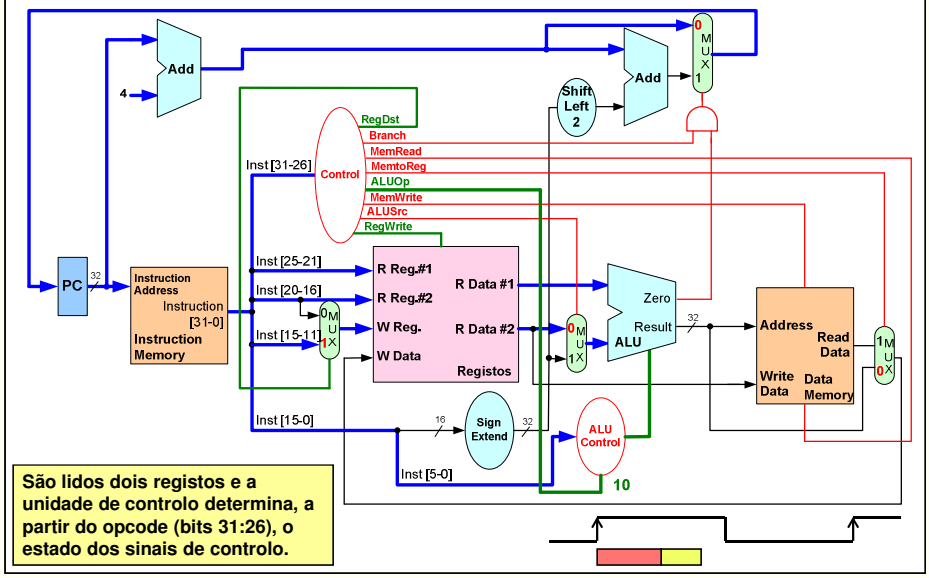
Funcionamento do datapath nas instruções do tipo R

Arquitectura de Computadores I 2006/07



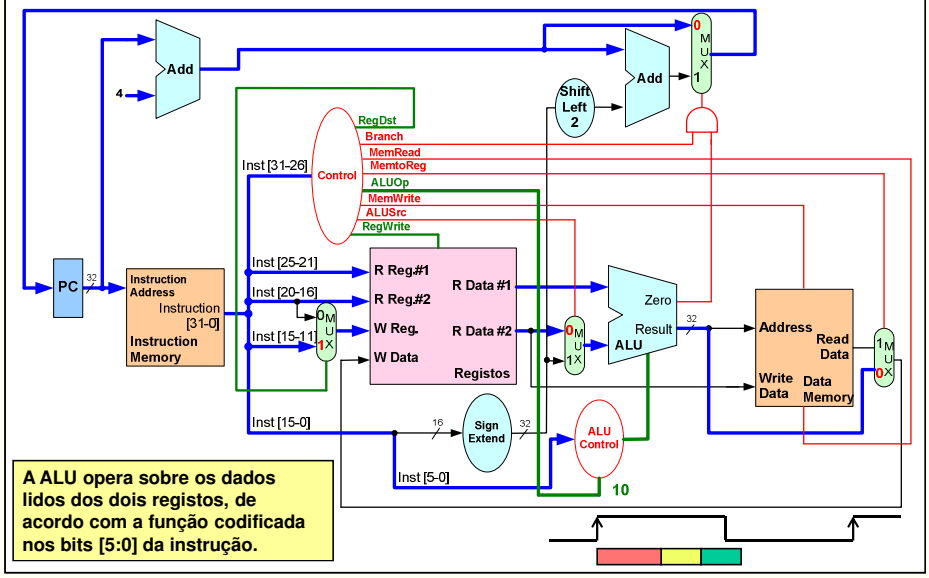
A instrução é lida e o PC é incrementado.

Arquitectura de Computadores I 2006/07



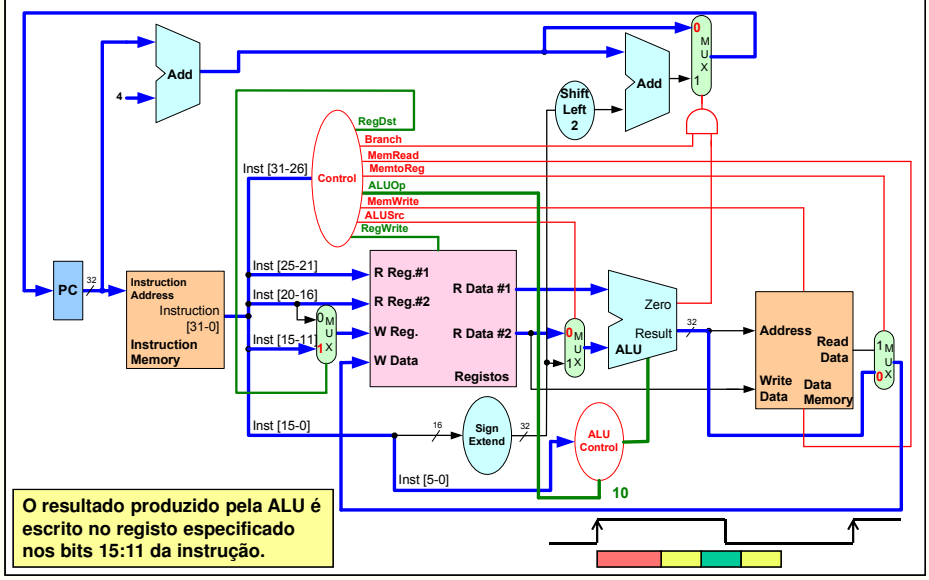
São lidos dois registos e a unidade de controlo determina, a partir do opcode (bits 31:26), o estado dos sinais de controlo.

Arquitectura de Computadores I 2006/07



A ALU opera sobre os dados lidos dos dois registos, de acordo com a função codificada nos bits [5:0] da instrução.

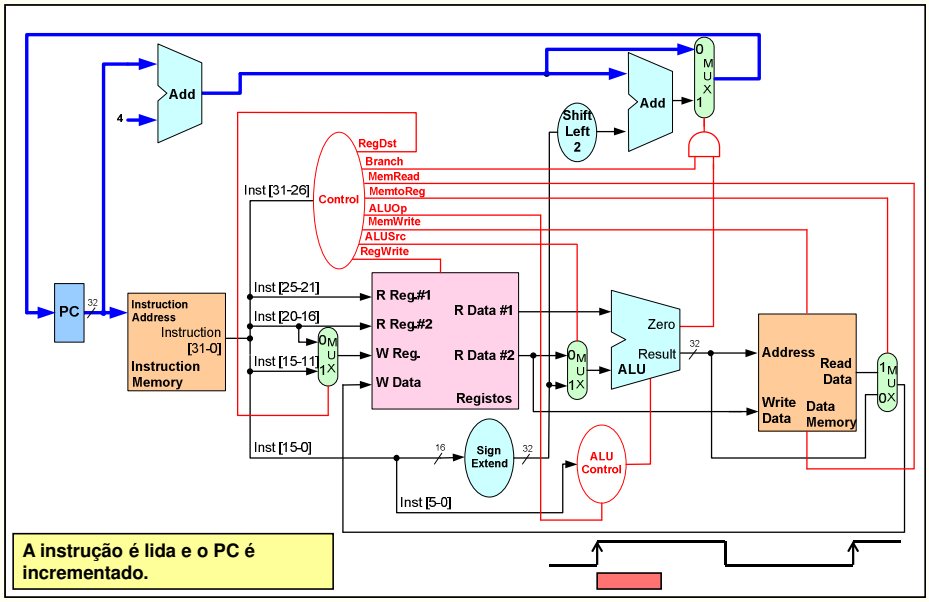
Arquitectura de Computadores I 2006/07



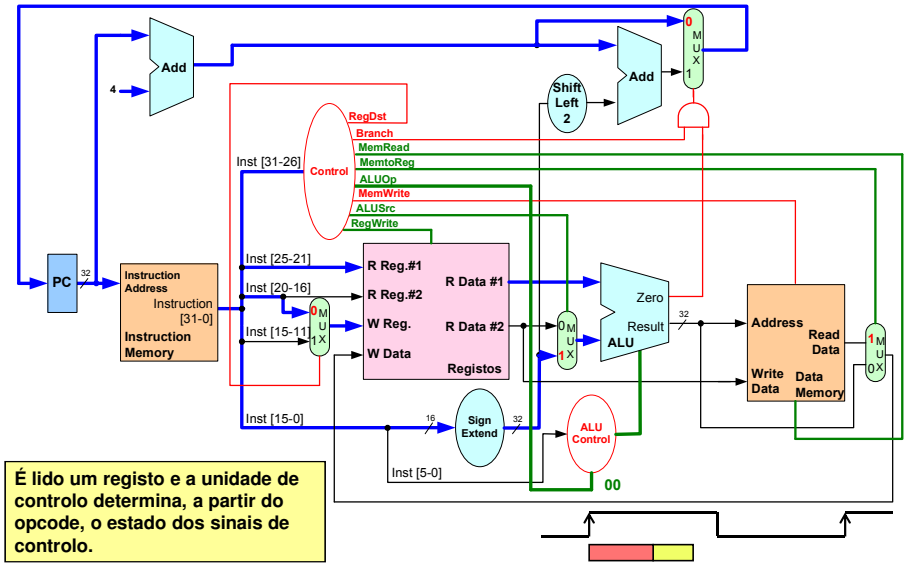
O resultado produzido pela ALU é escrito no registo especificado nos bits 15:11 da instrução.

Exemplo 2

Funcionamento do datapath na instrução *load word* ("lw")

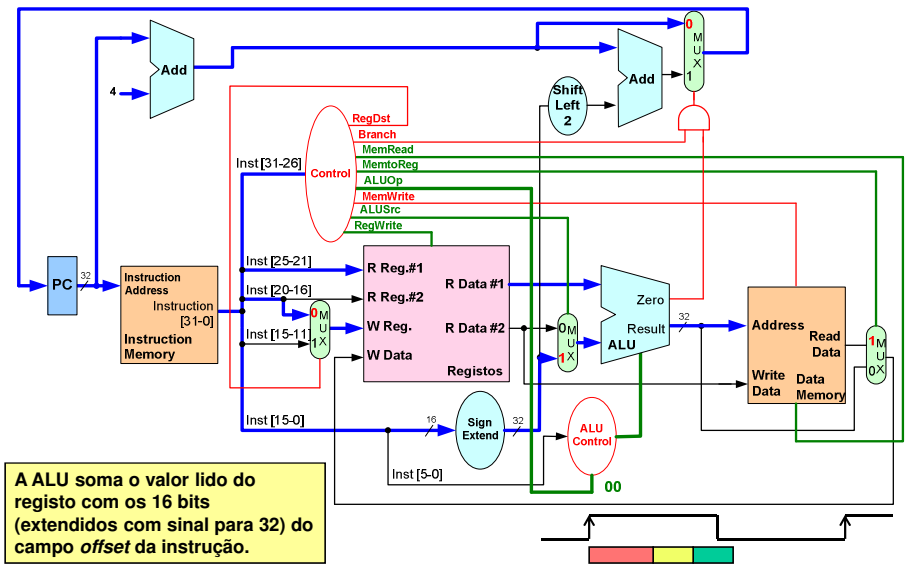


Arquitectura de Computadores I 2006/07



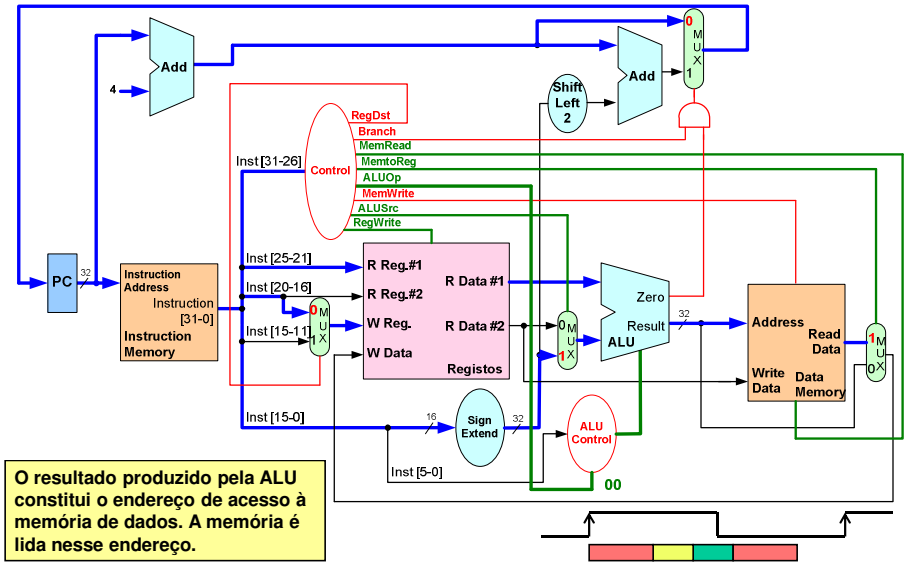
É lido um registo e a unidade de controlo determina, a partir do opcode, o estado dos sinais de controlo.

Arquitectura de Computadores I 2006/07



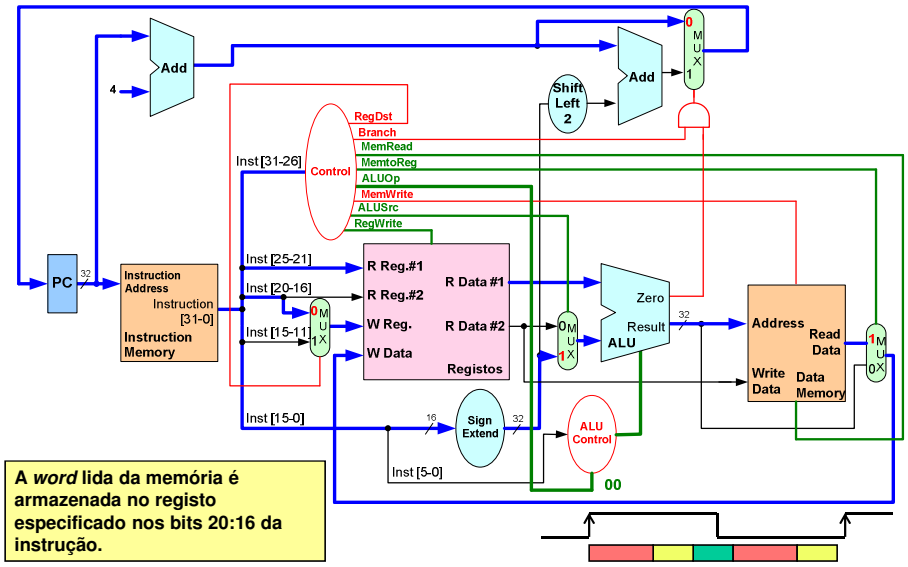
A ALU soma o valor lido do registo com os 16 bits (extendidos com sinal para 32) do campo *offset* da instrução.

Arquitectura de Computadores I 2006/07



O resultado produzido pela ALU constitui o endereço de acesso à memória de dados. A memória é lida nesse endereço.

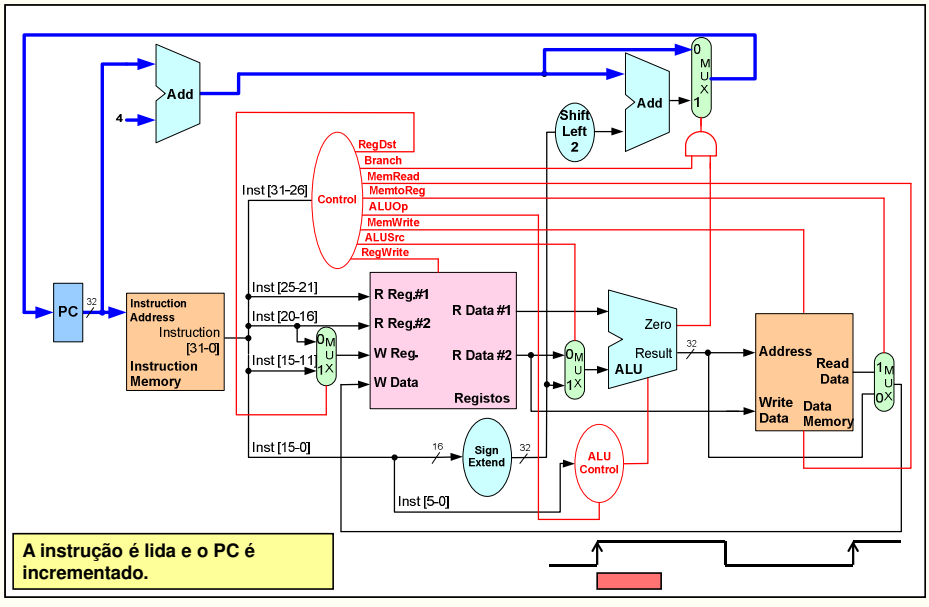
Arquitectura de Computadores I 2006/07



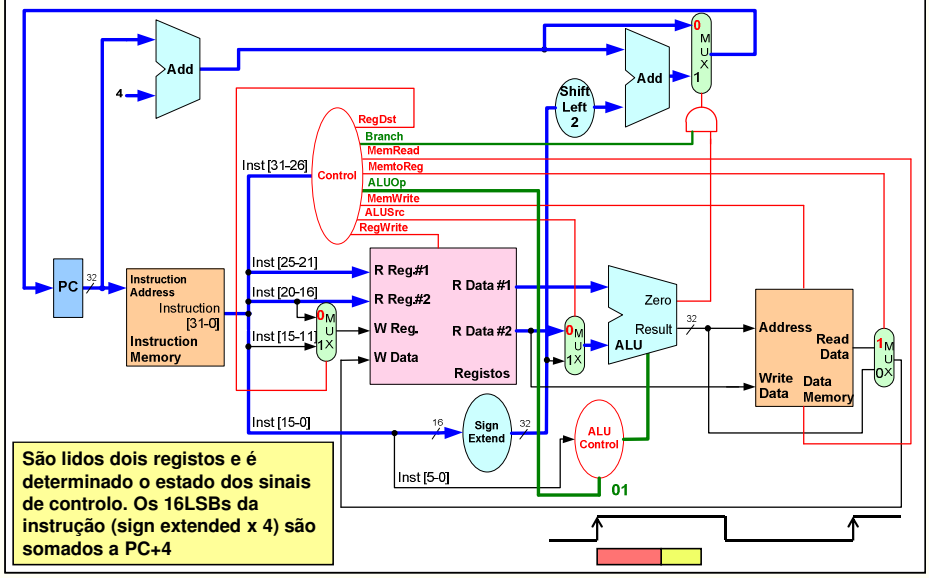
A *word* lida da memória é armazenada no registo especificado nos bits 20:16 da instrução.

Exemplo 3

Funcionamento do datapath na instrução *branch if equal* ("beq")

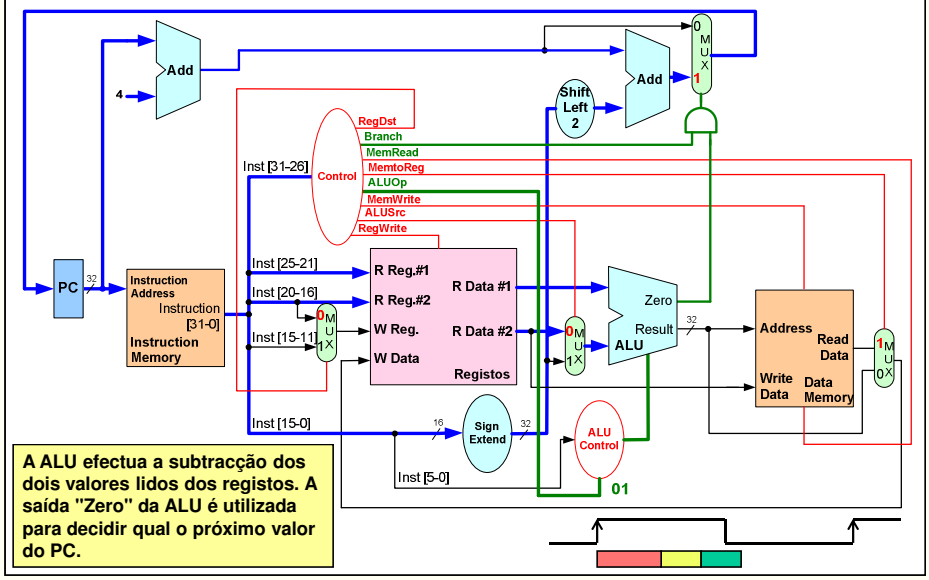


Arquitectura de Computadores I 2006/07



São lidos dois registos e é determinado o estado dos sinais de controlo. Os 16LSBs da instrução (sign extended x 4) são somados a PC+4

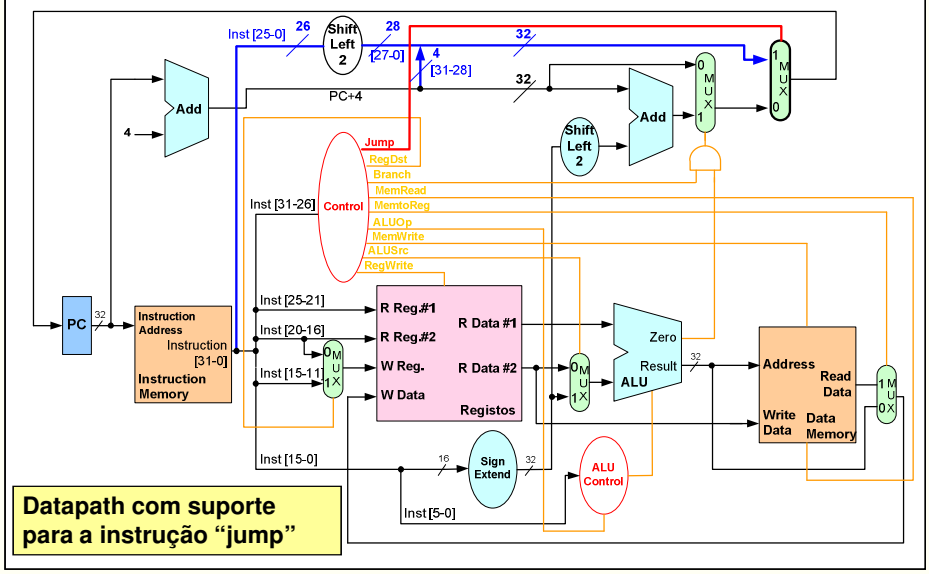
Arquitectura de Computadores I 2006/07



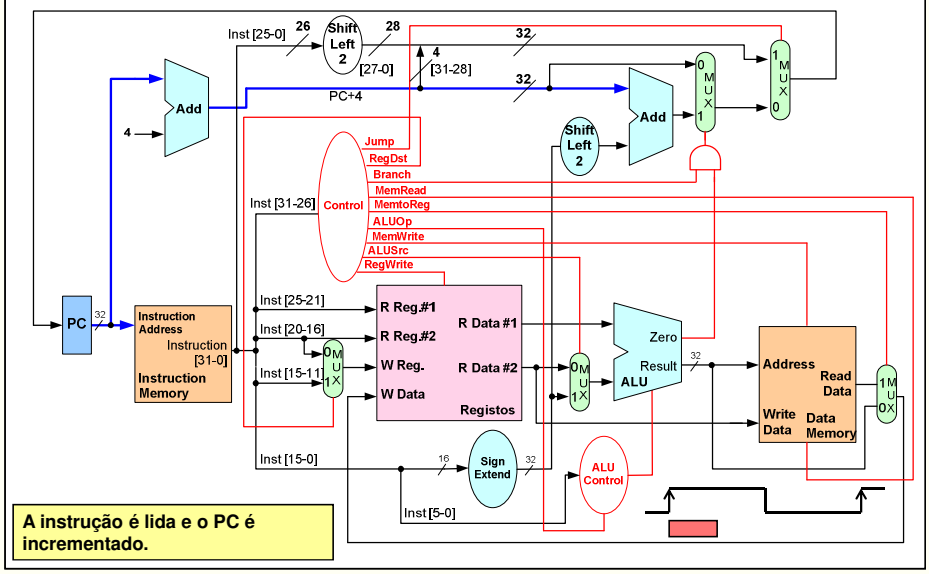
A ALU efectua a subtracção dos dois valores lidos dos registos. A saída "Zero" da ALU é utilizada para decidir qual o próximo valor do PC.

Datapath com suporte para a instrução "*jump*"

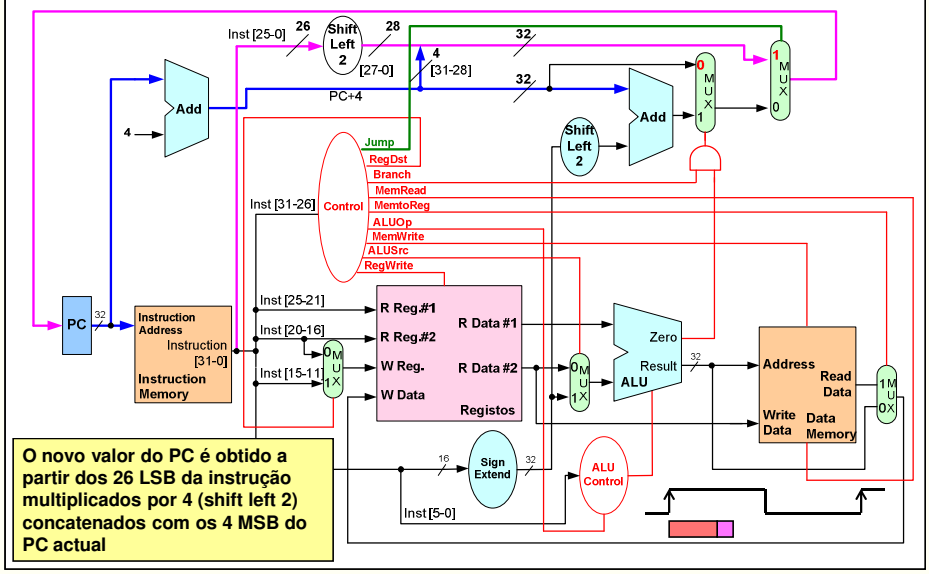
- A instrução "*jump*" corresponde a um caso particular de codificação (instruções do tipo J). Nestas instruções existem apenas dois campos: o campo *op* (bits 31-26) e o campo de endereço (bits 25-0)
- Nas instruções de "*jump*", o endereço *target* obtém-se pela concatenação dos bits 31-28 do PC+4 com os bits do campo de endereço da instrução multiplicados por 4.
- Será necessário acrescentar ainda um bit de saída à unidade de controlo para seleccionar a fonte de informação disponibilizada à entrada do PC
- O *datapath* simplificado, com suporte para instruções do tipo "*jump*" ficará assim com a seguinte configuração:



Arquitectura de Computadores I 2006/07



Arquitectura de Computadores I 2006/07



Arquitectura de Computadores I

2006/07