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 16 & 17

Pressupostos para a construção de um *Datapath* genérico para uma arquitectura tipo MIPS

Análise dos blocos constituintes necessários à execução de cada tipo de instruções básicas:

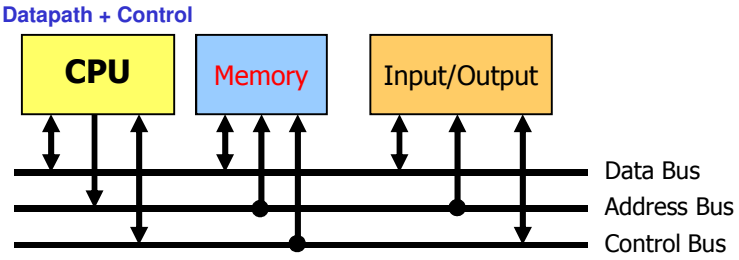
- Tipo R
- *Load* e *store*
- Salto condicional

Montagem de um *datapath* completo para execução de instruções *single cycle*

Arquitectura de Computadores I 2007/08

Relembremos a arquitectura básica de um sistema computacional

Modelo de von Neumann



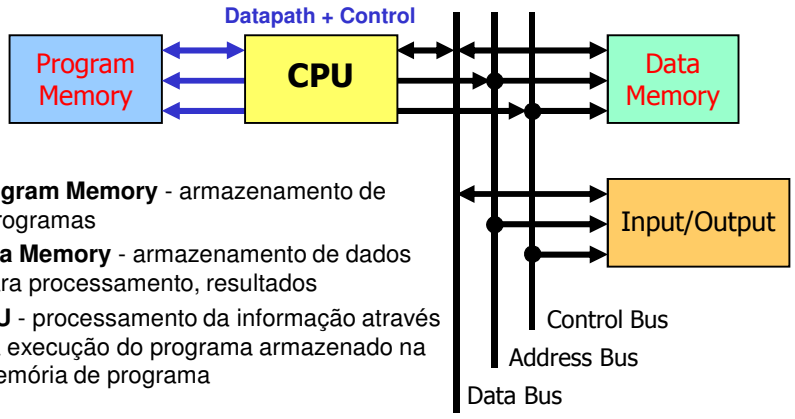
Memory – armazenamento de: programas, dados para processamento, resultados

CPU – processamento da informação através da execução do programa armazenado em memória

Arquitectura de Computadores I 2007/08

Arquitectura básica de um sistema computacional

Modelo de Harvard



Program Memory - armazenamento de programas

Data Memory - armazenamento de dados para processamento, resultados

CPU - processamento da informação através da execução do programa armazenado na memória de programa

Implementação de um *Datapath*

- Nestes slides faz-se uma abordagem à implementação de um *datapath* capaz de interpretar e executar o seguinte subconjunto de instruções do MIPS:
  - As instruções aritméticas e lógicas (à excepção das instruções de multiplicação, divisão e *shift*)
  - Instruções de acesso à memória: *load word* (`"lw"`) e *store word* (`"sw"`)
  - A instrução *branch if equal* (`"beq"`) e salto incondicional (`"j"`)
- Como iremos ver, independentemente da quantidade e tipo de instruções suportadas por uma dada arquitectura, uma parte importante do trabalho realizado pelo CPU e da infra-estrutura necessária para executar essas instruções é comum a praticamente todas elas.

Arquitectura de Computadores I 2007/08

Implementação de um *Datapath*

- No caso particular do MIPS, para cada instrução que compõe o *set* de instruções, as duas primeiras operações necessárias à sua realização são sempre as mesmas:
  1. Usar o conteúdo do registo *Program Counter* (PC) para indicar o endereço da memória do qual vai ser lida a próxima instrução e efectuar essa leitura;
  2. Ler um ou mais registos internos, usando para isso os números obtidos nos respectivos campos da instrução. Nas instruções de transferência memória→registo apenas um registo é necessário. Em todas as outras são sempre lidos dois registos.
- Depois destes destas operações genéricas, realizam-se as acções específicas para completar a execução da instrução em causa.

Implementação de um *Datapath*

- As acções específicas necessárias para executar as instruções de cada uma das três classes de instruções descritas anteriormente são, em grande parte, semelhantes, independentemente da instrução exacta em causa. Por exemplo, todas as classes de instruções, à excepção do *jump*, utilizam a ALU depois da leitura dos registos:
  - as instruções aritméticas e lógicas para a execução da instrução
  - as acções de acesso à memória usam a ALU para calcular o endereço de memória
  - a instrução de *branch* para efectuar a subtracção que permite determinar se os operandos são iguais ou diferentes
- Depois de utilizar a ALU, as acções que completam as várias classes de instruções diferem: as instruções ***lw*** e ***sw*** acedem à memória para leitura e escrita, respectivamente; as instruções aritméticas armazenam o resultado da ALU num registo; a instrução de *branch* pode ter que alterar o endereço onde se encontra a próxima instrução (o conteúdo do registo PC).

Arquitectura de Computadores I 2007/08

Implementação de um *Datapath*

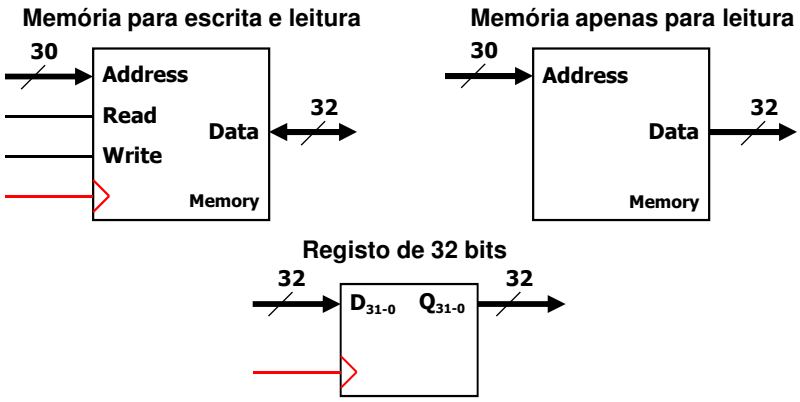
- As unidades funcionais que constituem o *datapath* do MIPS são de dois tipos:
  - Elementos combinatórios (por exemplo a ALU)
  - Elementos de estado, isto é, que têm capacidade de armazenamento (por exemplo os registos internos, a memória)
- Um elemento de estado possui, pelo menos, duas entradas:
  - Os dados a serem armazenados
  - O relógio, que determina o instante em que os dados são armazenados
- A saída de um elemento de estado disponibiliza a informação armazenada na última transição positiva do relógio
- Um elemento de estado pode ser lido em qualquer momento

Implementação de um *Datapath*

- Para além do sinal de relógio, um elemento de estado pode ainda ter sinais de controlo adicionais:
  - Um sinal de leitura (***read***), que permite ou não que a informação armazenada seja disponibilizada na saída
  - Um sinal de escrita (***write***) que autoriza (quando activo) a escrita de informação na próxima transição activa do relógio
- Se algum destes sinais não estiver explicitamente presente, isso significa que a operação respectiva é sempre realizada
- Havendo um sinal de relógio comum, e por uma questão de simplificação dos diagramas, o sinal de relógio pode não ser explicitamente representado

Implementação de um *Datapath*

Exemplos de representação gráfica de elementos de estado



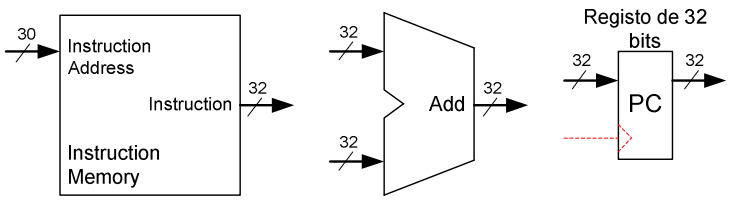
Implementação de um *Datapath - Instruction Fetch*

- O processo de acesso à memória para leitura da próxima instrução é genericamente designado por ***Instruction Fetch***
- Por uma questão de simplificar a organização da informação, as instruções que compõem um programa são armazenadas sequencialmente na memória (se a instrução ***n*** se encontra armazenada no endereço ***k***, então a instrução ***n+1*** encontra-se armazenada no endereço ***k+x***, em que ***x*** é a dimensão da instrução ***n,*** medida em bytes).
- No MIPS, a dimensão das instruções é fixa e igual a 4 bytes
- O processo de *Instruction Fetch* deverá assim, uma vez concluído, deixar o conteúdo do PC pronto para endereçar a próxima instrução.
- No caso do MIPS, tal corresponderá a adicionar a constante 4 ao actual valor do PC.

Implementação de um *Datapath - Instruction Fetch*

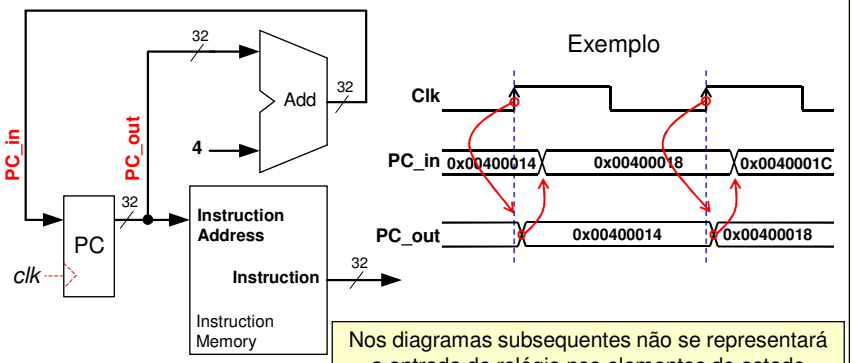
Os elementos operativos (combinatórios e/ou de memória) necessários à implementação de uma operação de *Instruction Fetch* serão portanto:

- A memória (de código)
- O *Program Counter* (um registo de 32 bits)
- Um somador



Implementação de um *Datapath - Instruction Fetch*

A parte do *Datapath* necessário à execução de um *Instruction Fetch* tomará assim a seguinte configuração:



Nos diagramas subsequentes não se representará a entrada de relógio nos elementos de estado

Implementação de um *Datapath*

Que outros elementos operativos básicos serão necessários para suportar a execução dos vários tipos de instruções? Consideremos, numa primeira abordagem os seguintes tipos:

- Instruções aritméticas e lógicas (Tipo R)
- Instruções de leitura e escrita da memória (Tipo I)
- Instruções de salto condicional (Tipo I)

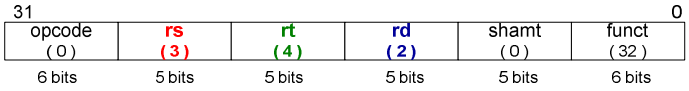
Relembremos que, na análise que se segue, não se explicita a Unidade de Controlo. Esta unidade é responsável pela geração dos sinais que determinam a ordem pela qual os dispositivos de armazenamento são actuados para efeitos de leitura e escrita.

Implementação de um *Datapath* (*Instruções tipo R*)

Operações realizadas por uma instrução do tipo R

- *Instruction Fetch* (leitura da instrução, incremento do PC)
- Leitura dos registos operandos (registos especificados nos campos "rs" e "rt" da instrução)
- Realização da operação na ALU (especificada no campo "funct")
- Escrita do resultado no registo destino (especificado no campo "rd")

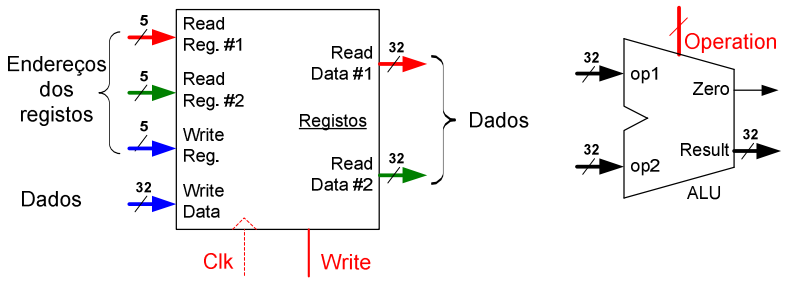
Exemplo: **add $2, $3, $4**



Implementação de um *Datapath* (*Instruções tipo R*)

Os elementos necessários à execução das instruções aritméticas e lógicas (tipo R) são:

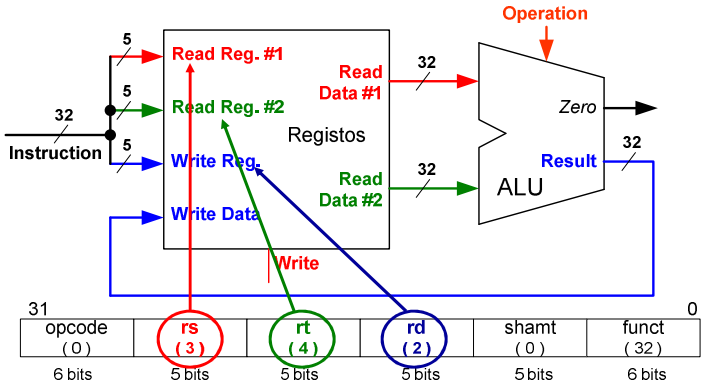
- Uma ALU de 32 bits
- Um conjunto de registos internos (***File Register*** com 32 registos)

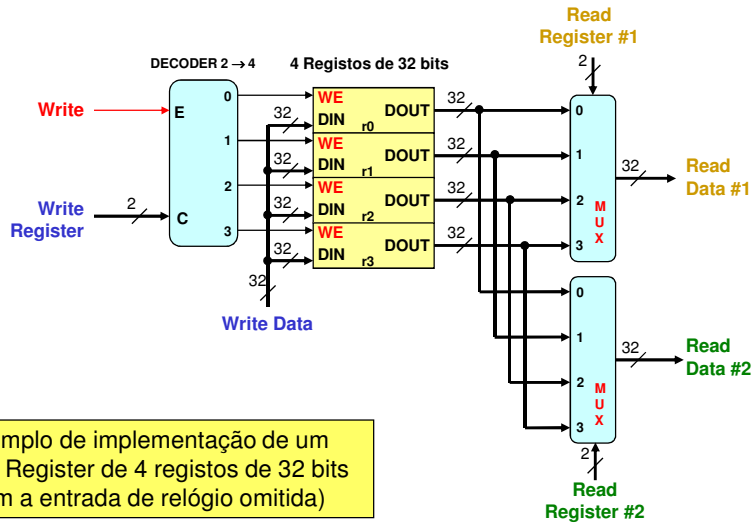


Implementação de um *Datapath* (*Instruções tipo R*)

A configuração respectiva será:

Exemplo: **add** $2, $3, $4





Exemplo de implementação de um File Register de 4 registos de 32 bits (com a entrada de relógio omitida)

Implementação de um *Datapath* (*Instrução SW*)

Operações realizadas por uma instrução "sw"

- *Instruction Fetch* (leitura da instrução e incremento do PC)
- Leitura dos registos que contêm o endereço base e o valor a transferir (reg. especificados nos campos “rs” e “rt”da instrução)
- Cálculo, na ALU, do endereço de acesso (soma entre o conteúdo do registo “rs” e o *offset* especificado na instrução)
- Escrita na memória

Exemplo: **sw $2, 0x24($4)**

opcode (43)	rs (4)	rt (2)	offset (0x24)

Implementação de um *Datapath* (*Instrução LW*)

Operações realizadas por uma instrução “lw"

- *Instruction Fetch* (leitura da instrução e incremento do PC)
- Leitura do registo que contem o endereço base (registo especificado no campo “rs” da instrução)
- Cálculo, na ALU, do endereço de acesso (soma entre o conteúdo do registo “rs” e o *offset* especificado na instrução)
- Leitura da memória
- Escrita do valor lido da memória no registo destino (especificado no campo “rt” da instrução)

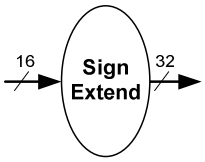
Exemplo: **lw $4, 0x2F($15)**

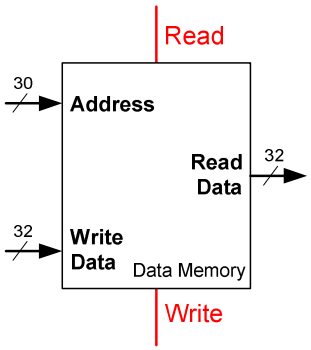
opcode (35)	rs (15)	rt (4)	offset (0x2F)

Implementação de um *Datapath* (*Instruções lw e sw*)

Os elementos necessários à execução das instruções de transferência de informação entre registos e memória (*load* e *store*) são, para além da ALU e do *File Register*:

- A memória externa (de dados)
- Um extensor de sinal





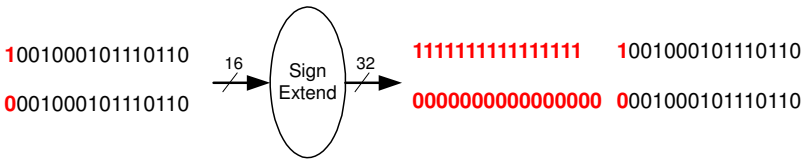
O extensor de sinal replica o bit 15 da instrução para os 16 mais significativos criando uma constante com 32 bits em complemento para 2

Implementação de um *Datapath* (*Instruções lw e sw*)

Os elementos necessários à execução das instruções de transferência de informação entre registos e memória (*load* e *store*) são, para além da ALU e do *File Register*:

- A memória externa (de dados)
- Um extensor de sinal

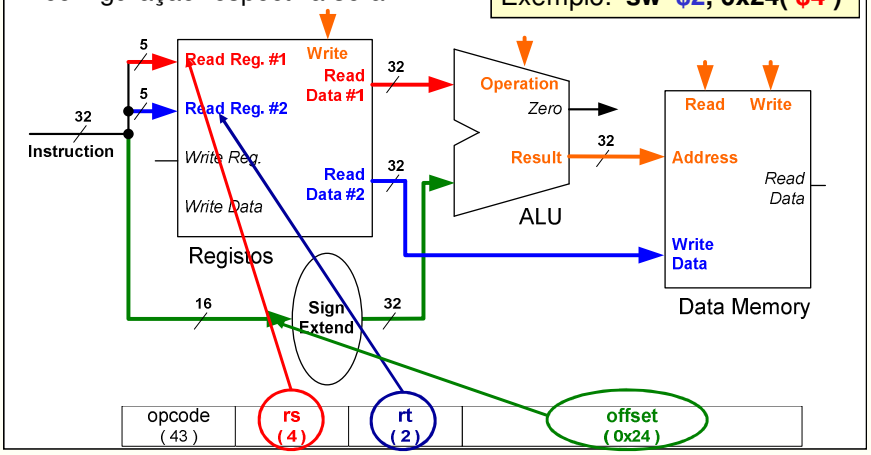
Exemplo:



Implementação de um *Datapath* (*Instruções lw e sw*)

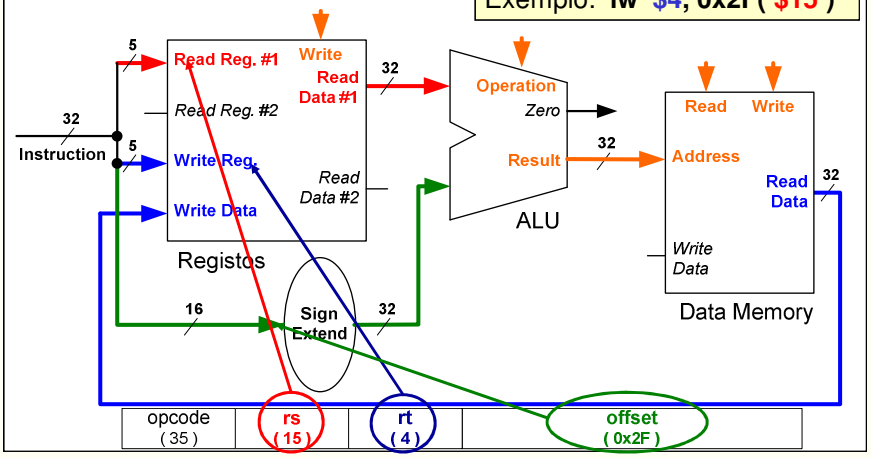
A configuração respectiva será:

Exemplo: **sw $2, 0x24($4)**



Implementação de um *Datapath* (*Instruções lw e sw*)

Exemplo: **lw $4, 0x2F($15)**



Implementação de um *Datapath* (*Instruções tipo branch*)

Operações executadas por uma instrução do tipo *branch*:

- *Instruction Fetch* (leitura da instrução, incremento do PC)
- Leitura dos registos a comparar (nºs regs. nos campos "rs" e "rt")
- Comparação dos valores dos registos (realização de uma operação de subtracção na ALU)
- Cálculo do endereço alvo da instrução *branch*

  PC + 4 + (*instruction_offset* * 4)
- Alteração do valor do registo PC (se a condição do *branch* for verdadeira)

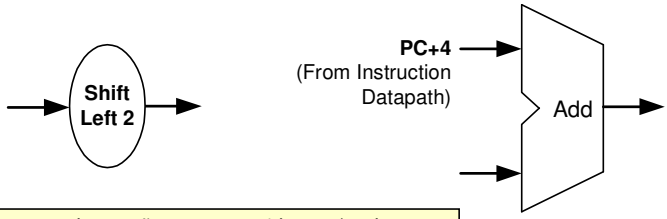
Ex.: **beq $2, $3, 0x20**

opcode (4)	rs (2)	rt (3)	instruction_offset (0x20)

Implementação de um *Datapath* (*Instruções tipo branch*)

Finalmente, os elementos necessários à execução das instruções de salto condicional implicam a inclusão dos seguintes elementos:

- *2 bit left shifter*
- Um somador com base no valor de PC+4

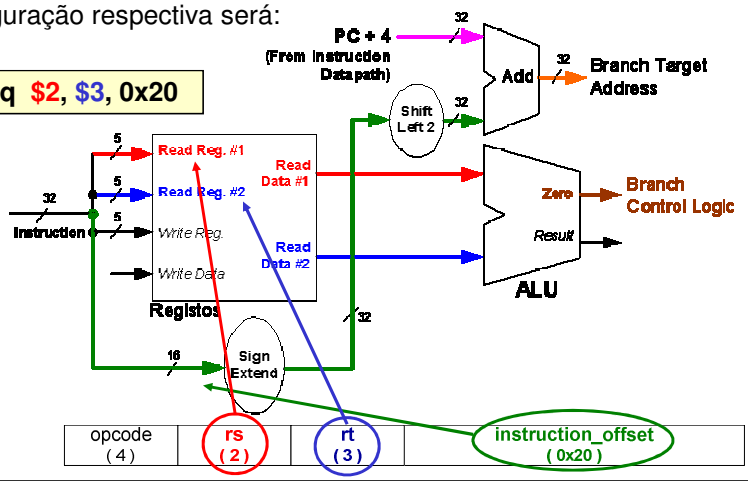


Uma vez que as instruções ocupam 4 bytes (endereços múltiplos de 4), o *left shifter* permite poupar 2 bits no campo de *instruction_offset* da instrução.

Implementação de um *Datapath* (*Instruções tipo branch*)

A configuração respectiva será:

Ex.: **beq $2, $3, 0x20**



Implementação de um *Datapath*

- Tendo identificado, separadamente, os blocos básicos constituintes do *Datapath* necessários à execução dos vários tipos de instruções do MIPS (dos quais se omitiram desta discussão, por enquanto, as instruções do tipo J), coloca-se a seguinte questão:
  - **Como juntar e interligar os diversos blocos, por forma a servir todas as instruções?**
- A resposta a esta pergunta passa pela identificação dos blocos que podem ser partilhados pelos vários tipos de instruções e pelo desenvolvimento de uma estratégia que permita que os mesmos possam ser "configurados" para cada caso.

Implementação de um *Datapath*

Relembremos o formato dos três tipos de instruções!

R-type

31					0
opcode (0)	**rs**	**rt**	**rd**	shamt	funct
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

LW & SW

31			0
opcode (35 ou 43)	**rs**	**rt**	offset
6 bits	5 bits	5 bits	16 bits

Branch

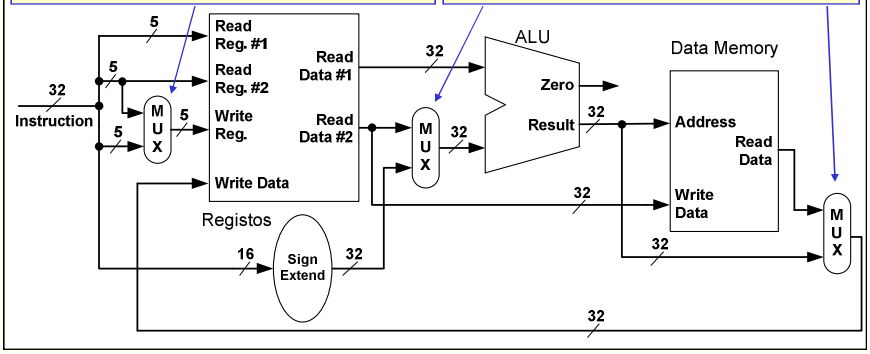
31			0
opcode (4)	**rs**	**rt**	instruction_offset
6 bits	5 bits	5 bits	16 bits

Implementação de um *Datapath* (1º passo)

A combinação das instruções de transferência registo / memória / registo com as instruções do tipo R pode ser conseguida da seguinte forma

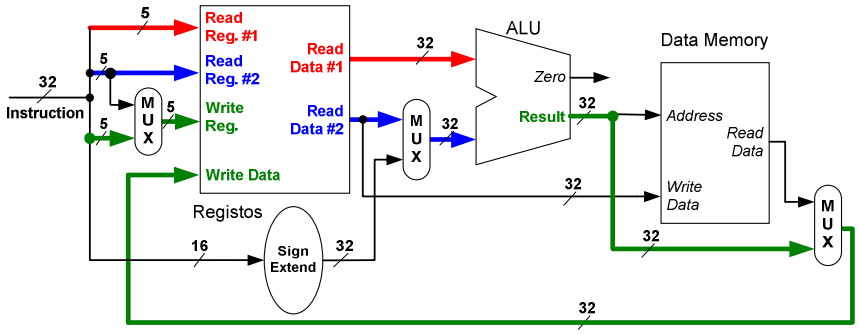
Escolha do registo destino (3º campo nas instruções tipo R, 2º na instrução LW)

Note-se a utilização de multiplexers para adequar os recursos às necessidades



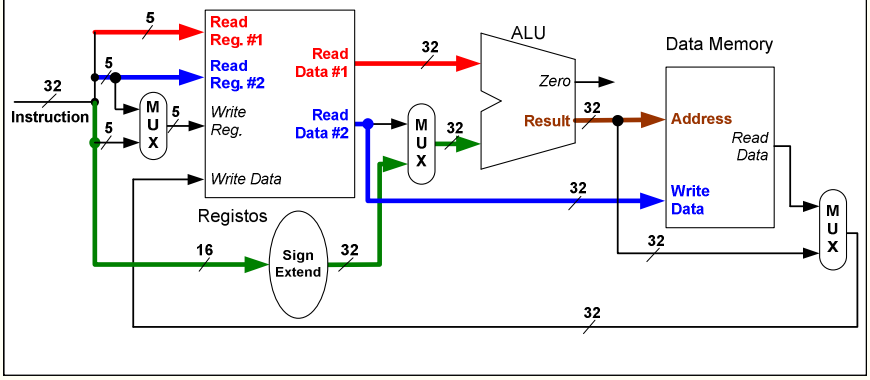
Implementação de um *Datapath* (1º passo)

A instrução do tipo R executada sobre um *datapath* misto.



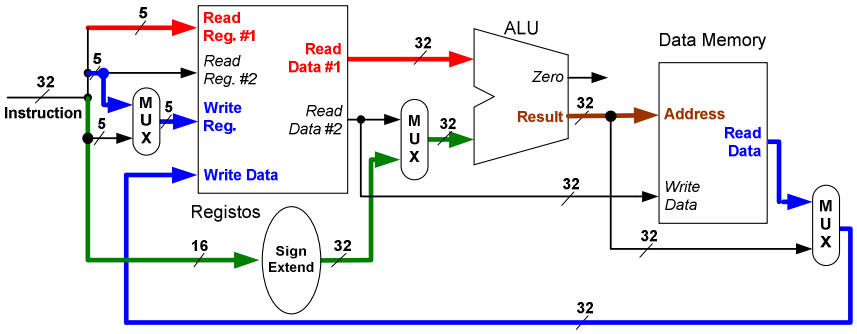
Implementação de um *Datapath* (1º passo)

A instrução SW (*store word*) executada sobre um *datapath* misto.



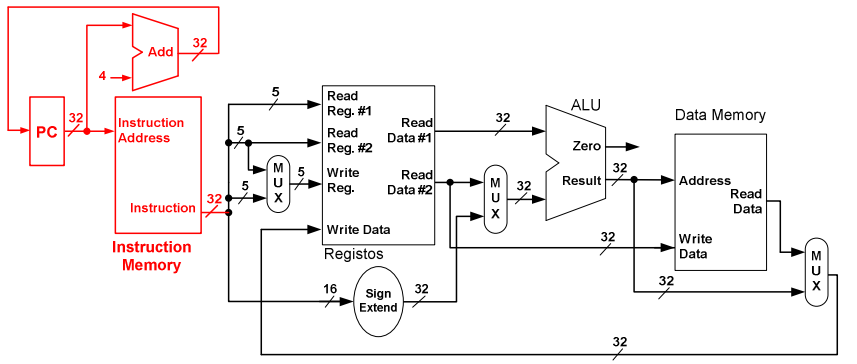
Implementação de um *Datapath* (1º passo)

A instrução LW (*load word*) executada sobre um *datapath* misto.



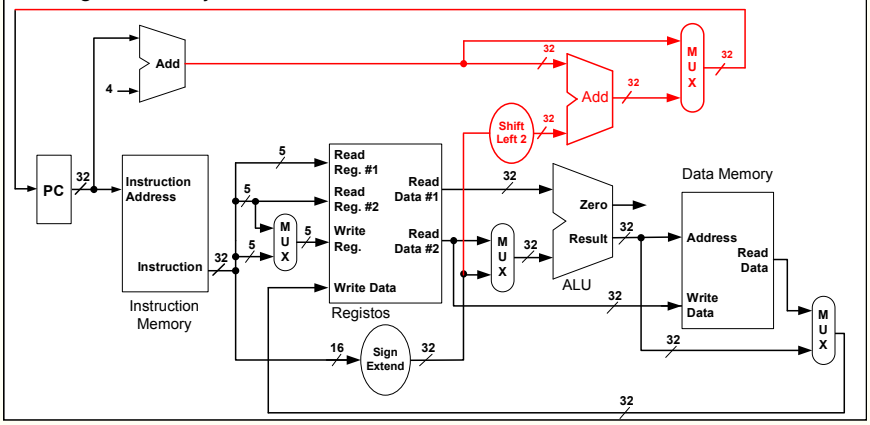
Implementação de um *Datapath* (2º passo)

O bloco de *Instruction Fetch* pode ser acrescentado sem grandes complicações



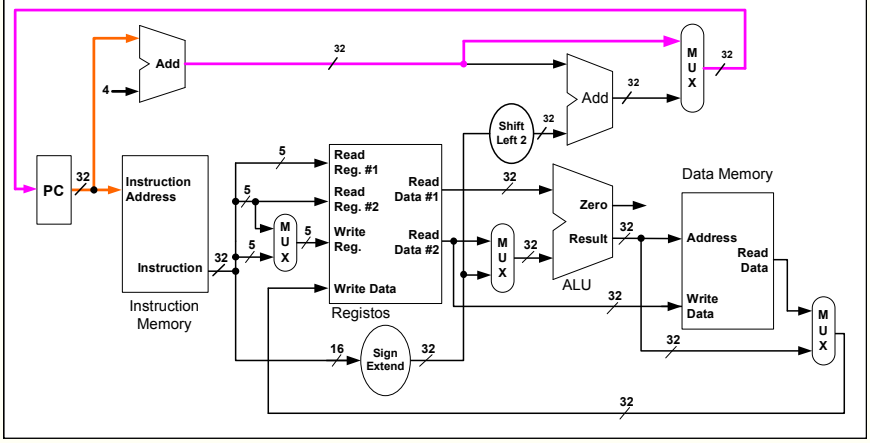
Implementação de um *Datapath* (3º passo)

Finalmente, as instruções de salto condicional, implicam algumas alterações à imagem de conjunto



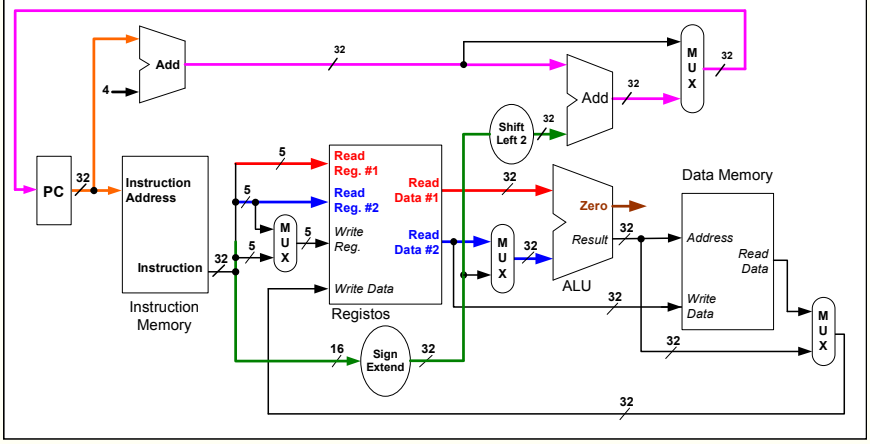
Implementação de um *Datapath* (3º passo)

Datapath misto (*Instruction fetch*)



Implementação de um *Datapath* (3º passo)

Datapath misto (Instrução *Branch if Equal*)



Implementação de um *Datapath*

Datapath misto com identificação dos sinais de controlo