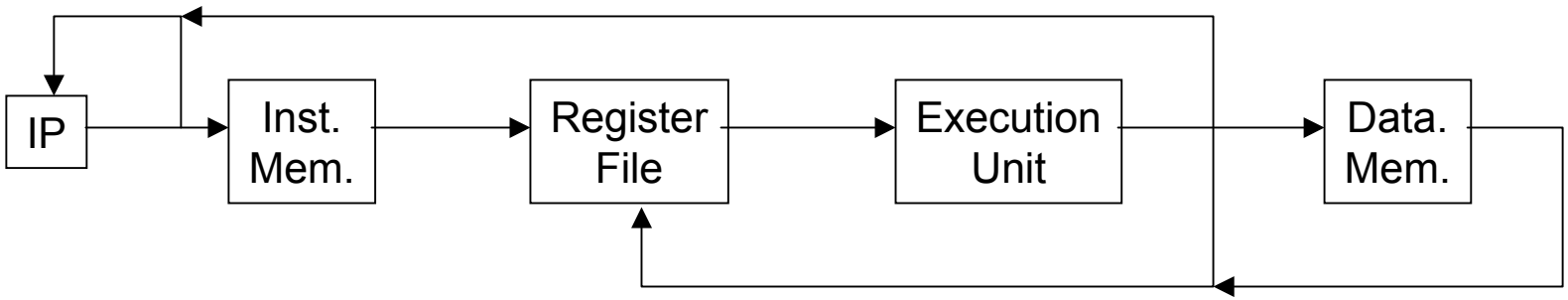
Organização ou MicroArquitectura

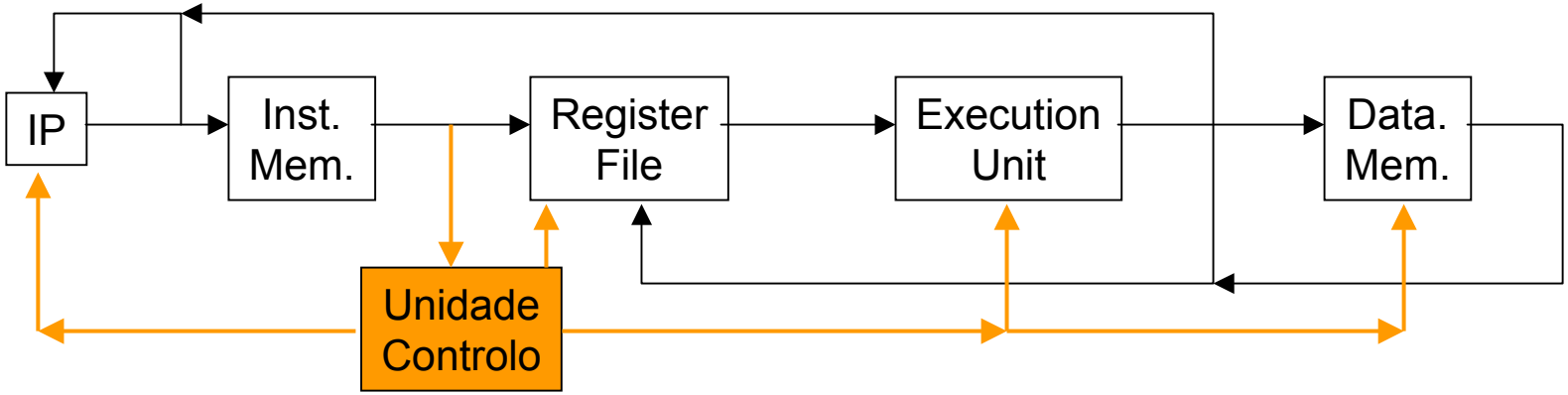
DataPath MIPS32

Datapath e *Controlpath*

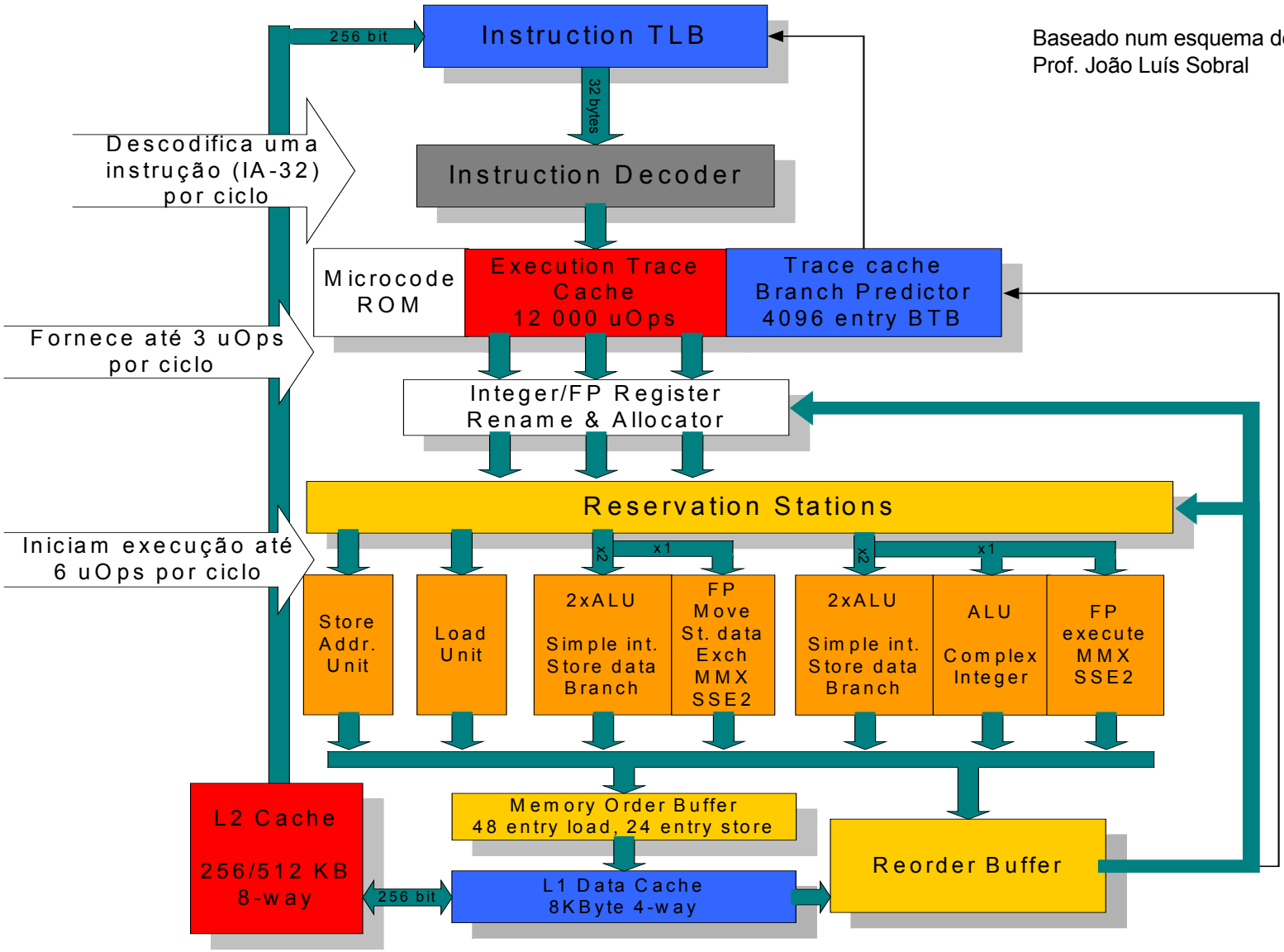
Datapath – circuito percorrido pelas instruções, endereços e resultados



Controlpath – circuito percorrido pelos sinais que controlam o *datapath*



Datapath Intel Pentium 4

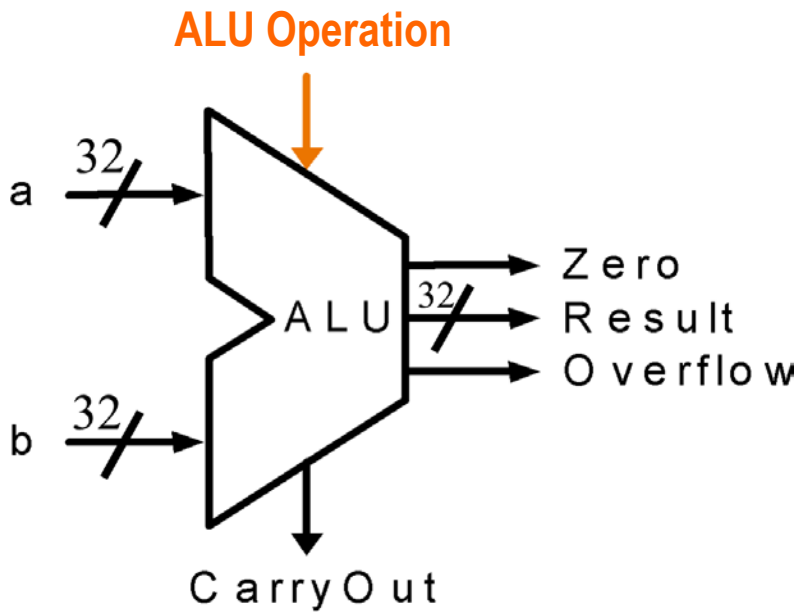


Baseado num esquema do Prof. João Luís Sobral

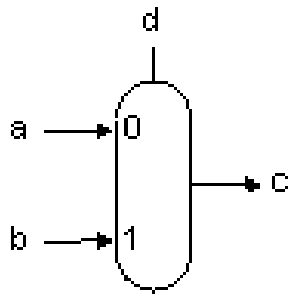
Elementos Combinatórios

As saídas destes elementos reagem **sempre** a variações nas entradas, após um certo atraso (*delay*) de propagação.

EXEMPLOS: um *multiplexer*, a ALU



se (d==0) c=a senão c=b



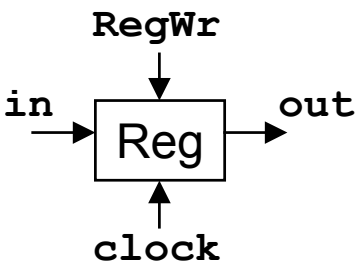
d	c
0	a
1	b

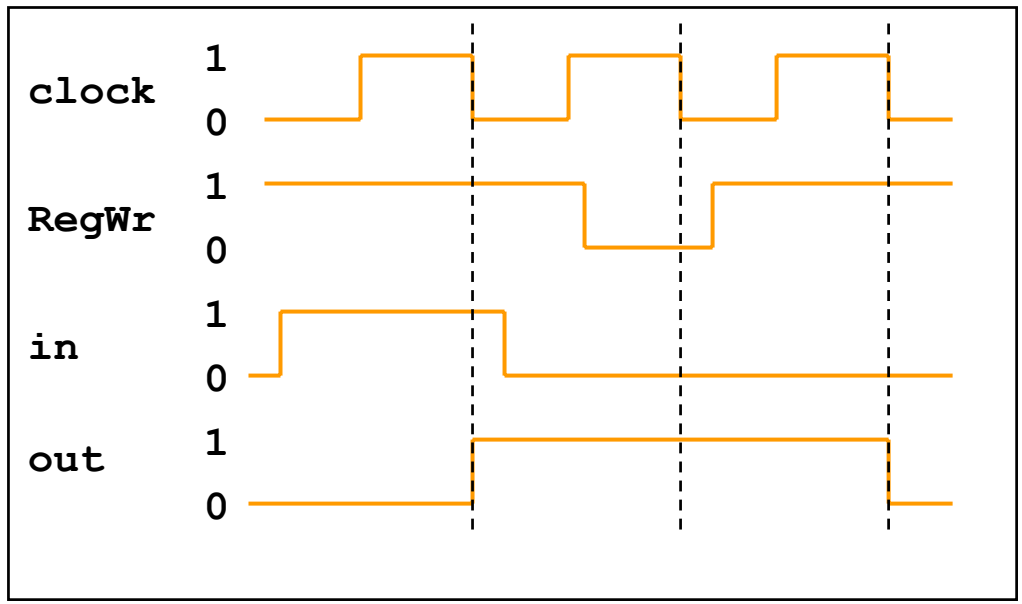
Elementos sequenciais

As saídas destes elementos **só** reagem a variações nas entradas quando se verifica um pulso do *clock*.
São estes elementos que memorizam dados, determinando o **estado** do CPU.

EXEMPLOS: os registos, a memória

Convenção de temporização – os elementos sequenciais só são escritos no pulso descendente do *clock*. Ignoram-se os atrasos.





Datapath MIPS32 – ciclo único

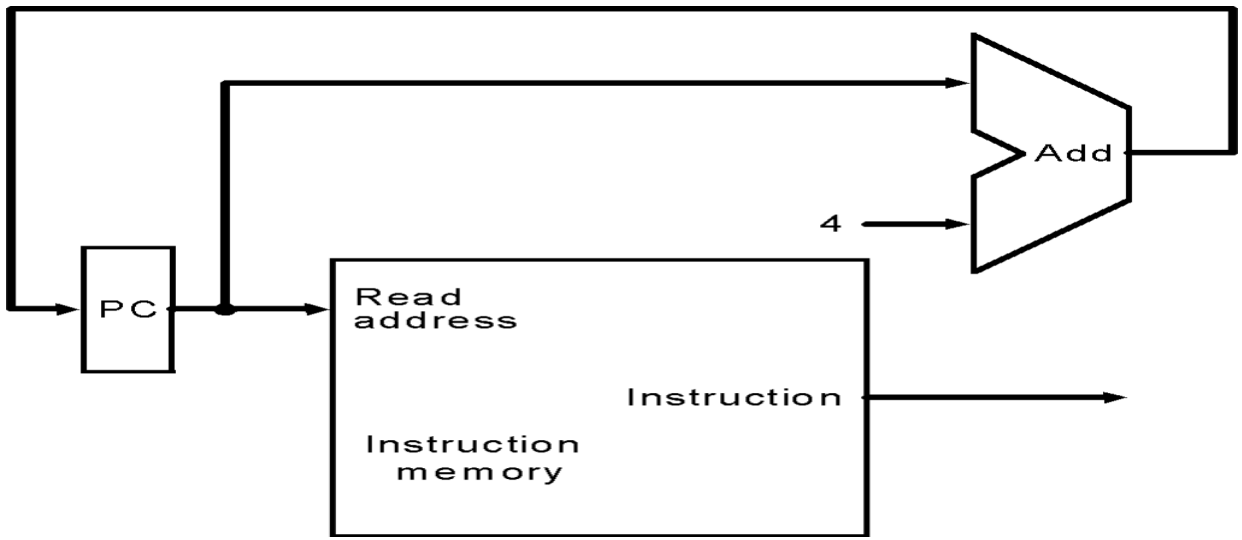
Implementação de ciclo único:

1. Todas as instruções demoram exactamente um único ciclo do relógio
2. O período do *clock* tem que ser igual à instrução mais demorada
3. Nenhum componente do *datapath* pode ser usado mais do que uma vez durante a mesma instrução

Sub conjunto de instruções do MIPS:

- **Acesso à memória –** `lw` e `sw`
- **Lógico-aritméticas –** `add`, `addi`, `sub`, `and`, `andi`, `or`, `ori`, `slt` e `slti`
- **Controlo de fluxo –** `beq`

Datapath : *fetch* da instrução



Todas as instruções têm que ser lidas do endereço apontado pelo PC.

O PC é incrementado 4 no fim do ciclo.

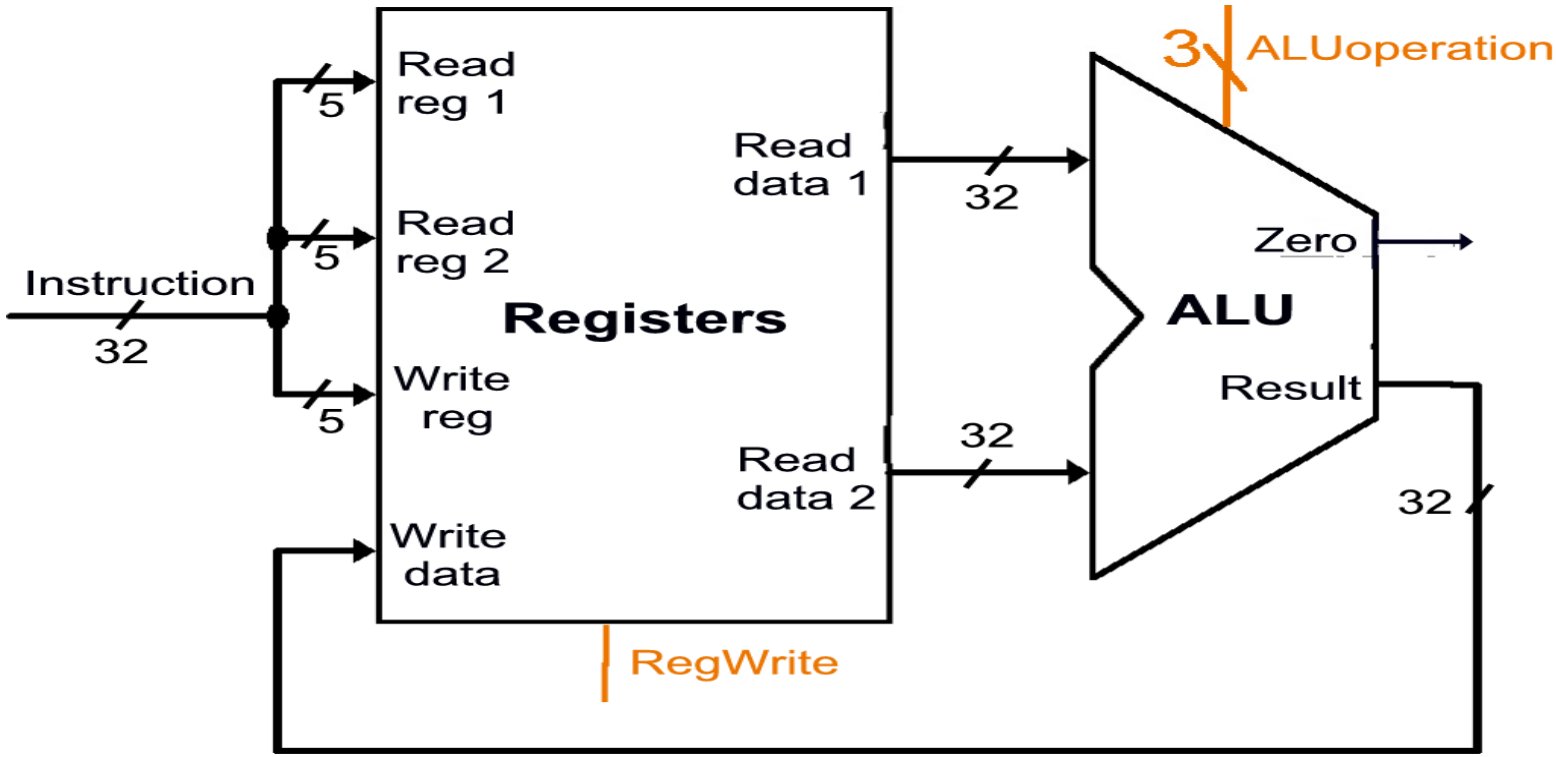
A mémória mantêm a instrução na saída até ao fim do ciclo.

Não são apresentados sinais de controlo da memória porque esta é só de leitura.

Esta ALU apenas realiza adições, não precisa de sinais de controlo.

A memória é dedicada a instruções, porque cada componente só pode ser usado uma vez em cada ciclo. A memória de dados será outro componente.

Datapath para instruções tipo R

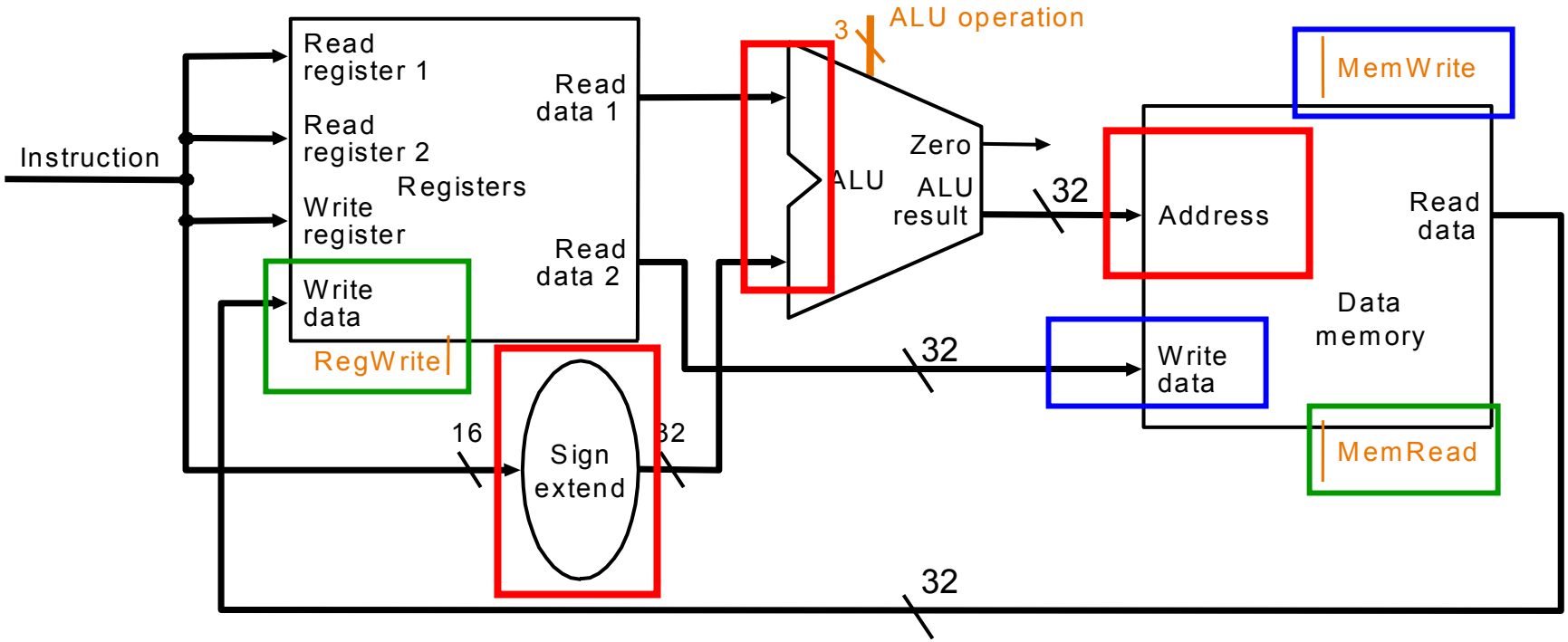


As instruções R lêem 2 registos (Rs, Rt) e escrevem num terceiro (Rd).

O campo Funct da instrução vai ser usado para gerar o sinal ALUoperation.

O sinal de controlo RegWrite indica se o resultado deve ser escrito num registo.

Datapath para `lw` e `sw`

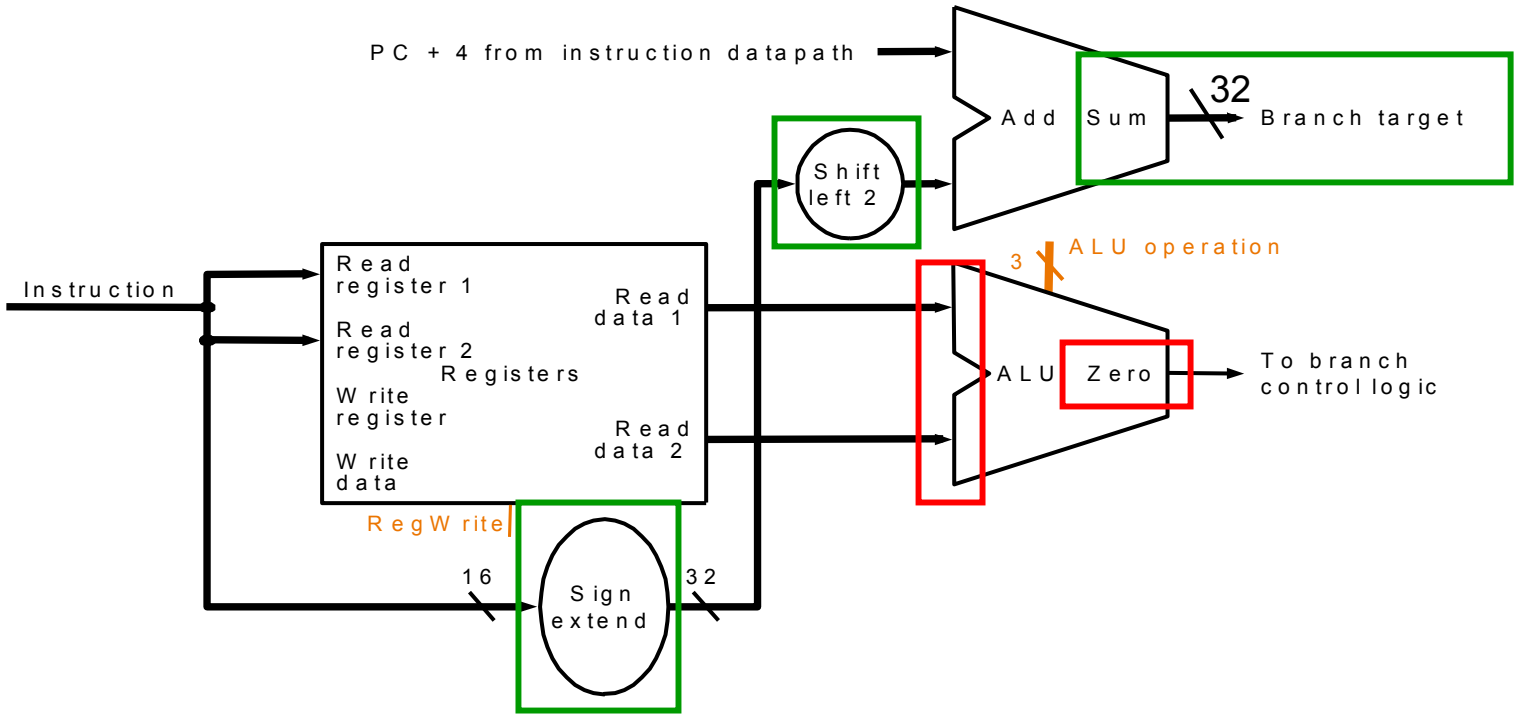


Cálculo do endereço - O *offset* imediato é estendido para 32 *bits* e somado ao registo base

lw – Os sinais MemRead e RegWrite são activados para que os dados sejam escritos no registo endereçado no campo Rt

sw – O sinal MemWrite é activado para que o valor do registo endereçado no campo Rt seja escrito na memória

Datapath para `beq`

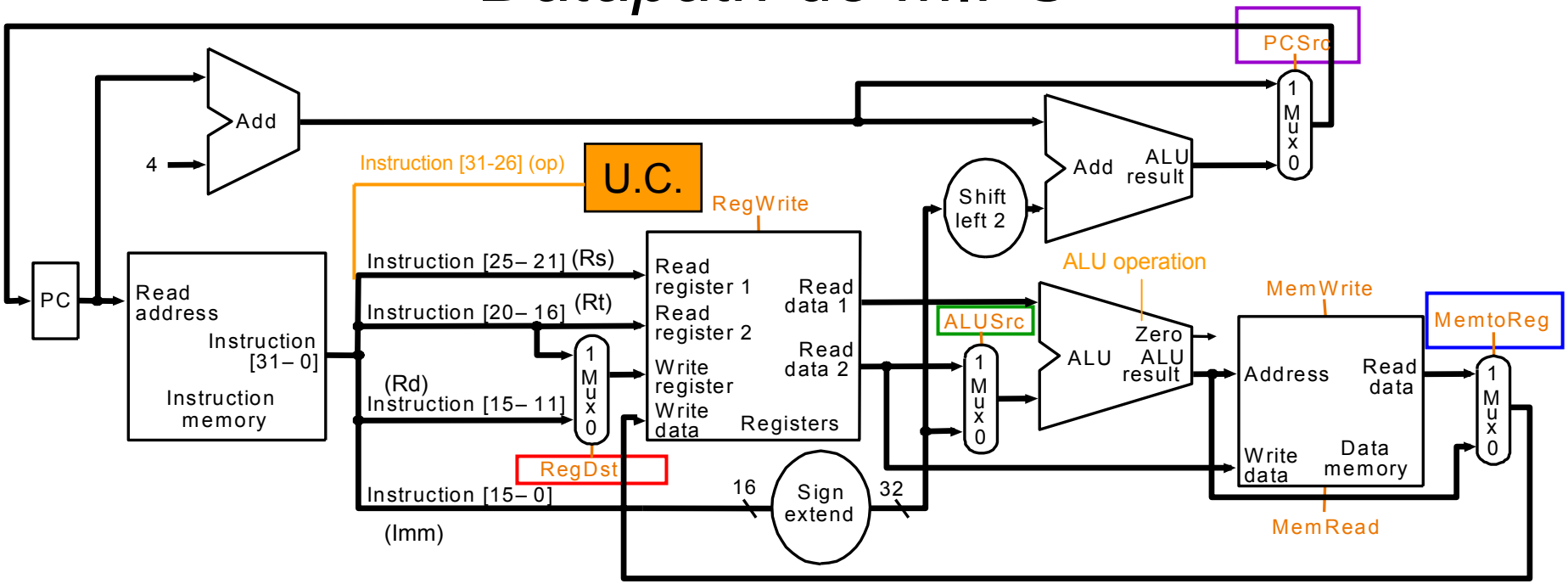


Os registos Rs e Rt são comparados activando a saída Zero da ALU

O *offset* do campo imm da instrução é estendido para 32 *bits*, multiplicado por 4 e somado ao PC

O PC só é carregado com este valor se a saída Zero da ALU estiver a 1

Datapath do MIPS



RegDst – O registo a escrever pode ser o Rd (tipo-R) ou o Rt (tipo-I)

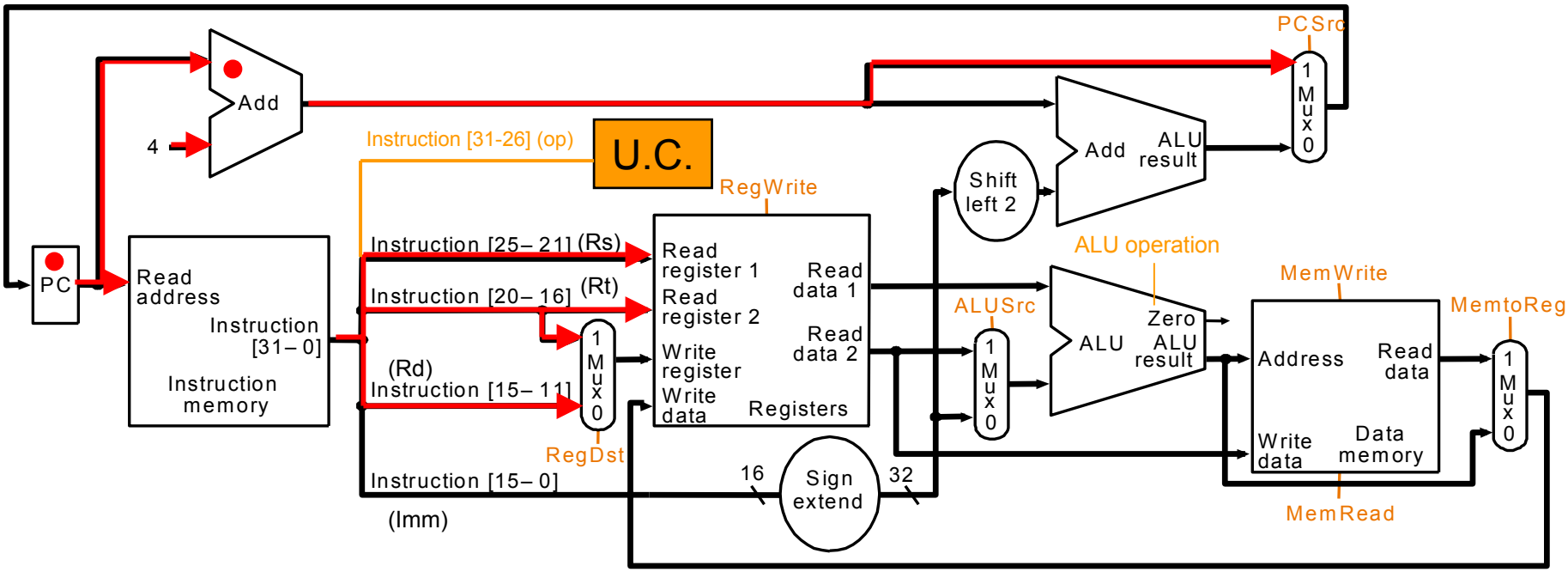
AluSrc – O operando pode ser um registo (tipo-R) ou uma constante (tipo-I)

MemtoReg – O valor a escrever num registo pode vir da ALU ou da memória (`lw`)

PCSrc – O valor a escrever no PC pode ser PC+4 ou o PC+4+offset de `beq`

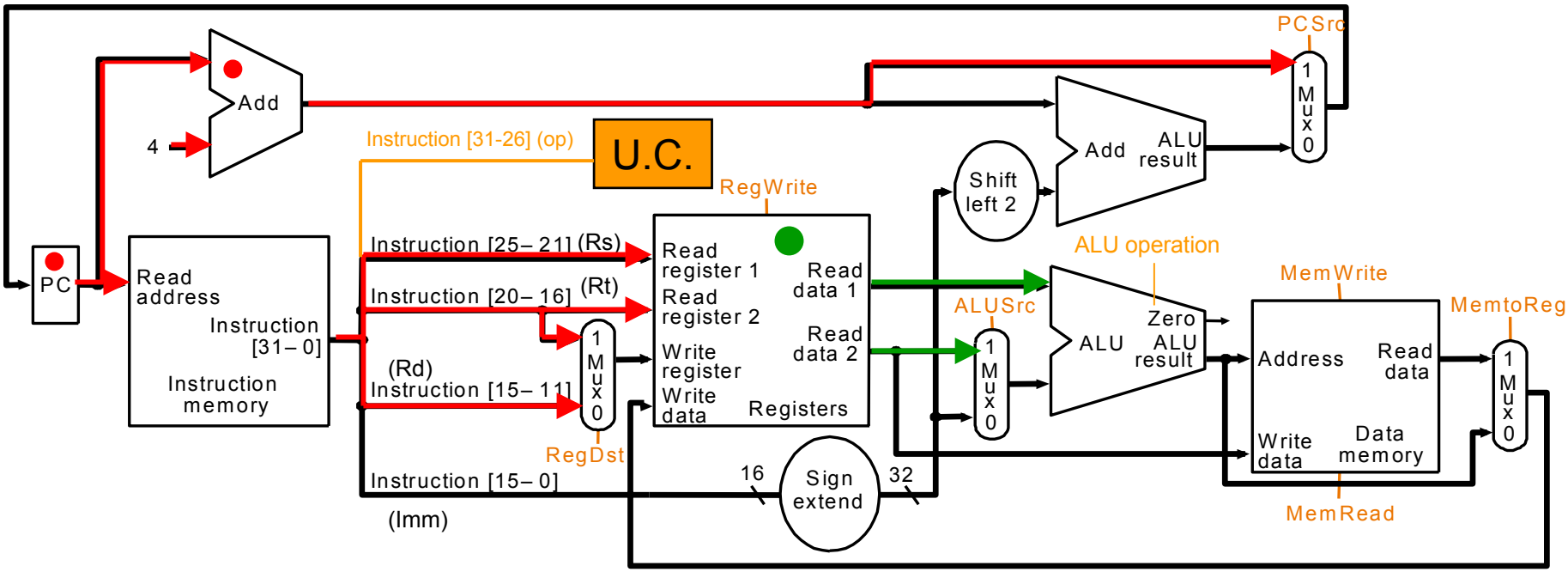
NOTA: Falta gerar os sinais de controlo, que entre outros, se baseiam no Opcode (Instruction[31-26]) e no Funct (Instruction[5-0])

Instrução do Tipo-R (*fetch*)



A instrução apontada pelo PC é lida e o valor do PC é incrementado de 4 (mas não é escrito no registo)

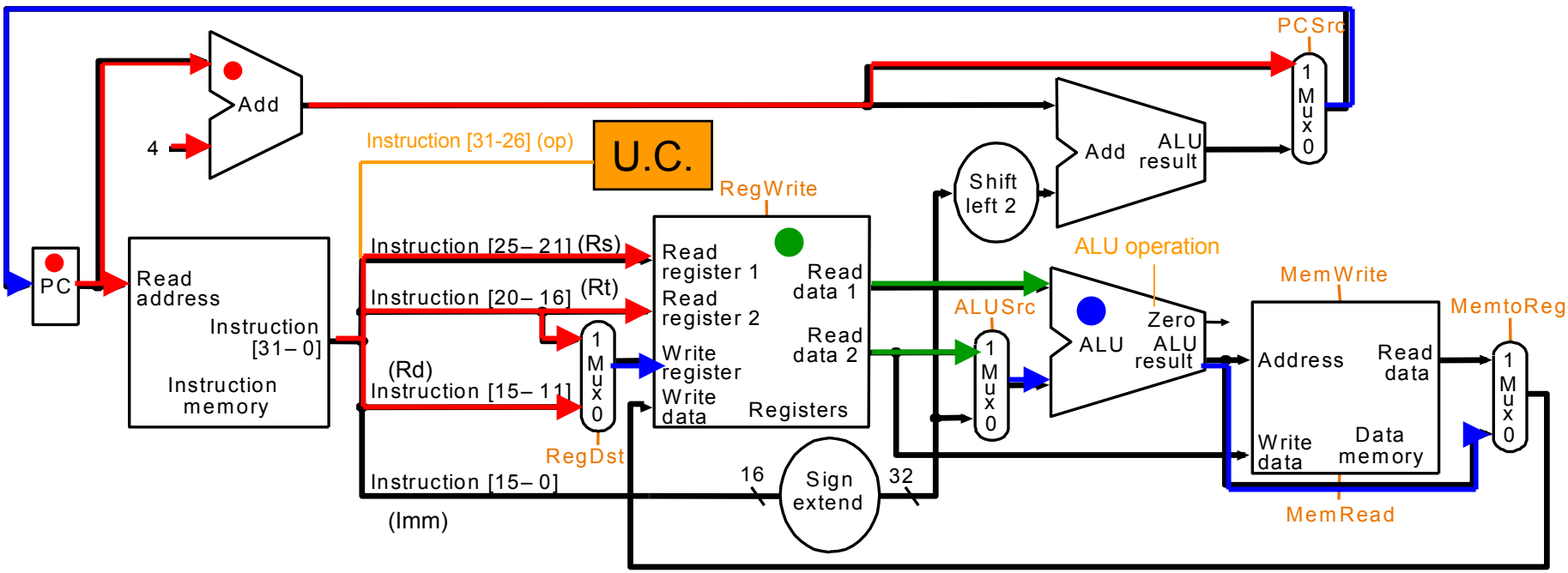
Instrução do Tipo-R (*decode/operand fetch*)



RegDst	0
RegWrite	1
ALUSrc	1
MemWrite	0
MemRead	0
MemtoReg	0
ALU operation	???
PCSrc	1

A unidade de controlo descodifica a instrução e gera os sinais de controlo.
(A unidade de controlo não aparece neste diagrama)
Os registos são lidos.

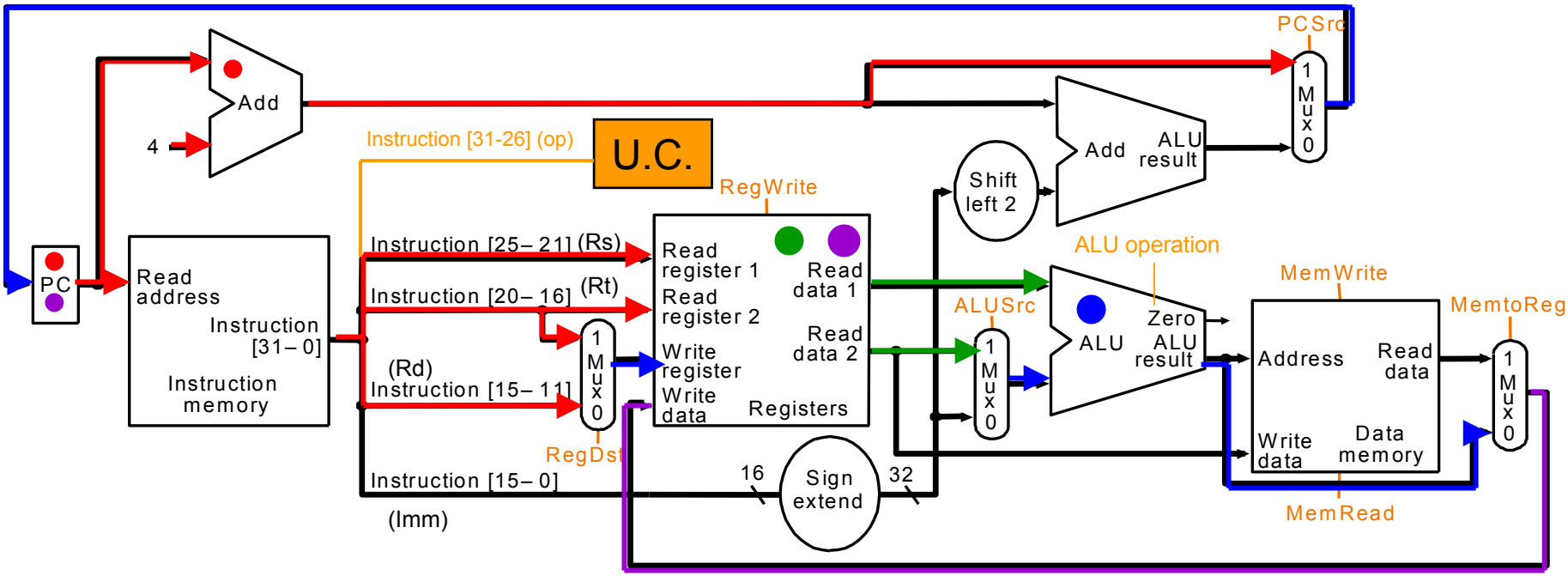
Instrução do Tipo-R (*execute*)



RegDst	0
RegWrite	1
ALUSrc	1
MemWrite	0
MemRead	0
MemtoReg	0
ALU operation	???
PCSrc	1

A ALU realiza a operação.

Instrução do Tipo-R (*writeback)*

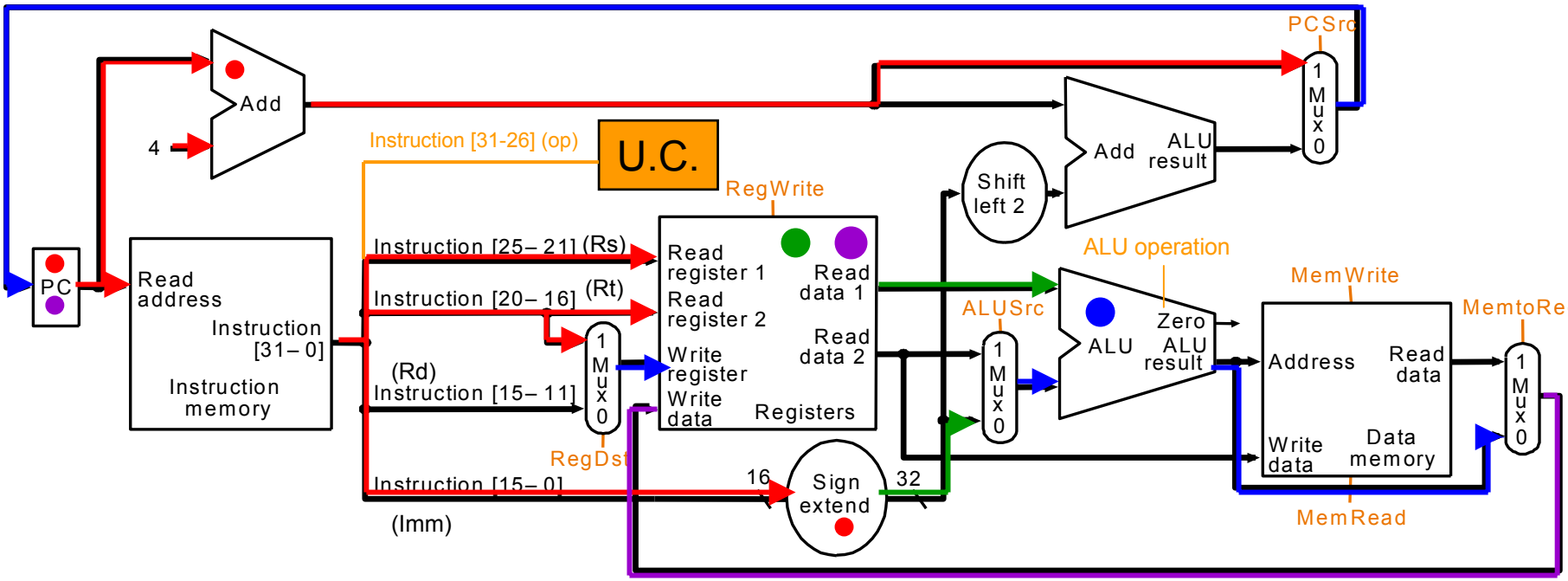


Apenas no fim do ciclo é que os registos são escritos (PC inclusive).

É possível distinguir 4 fases diferentes de execução, mas todas são realizadas no mesmo ciclo do *clock*.

RegDst	0
RegWrite	1
ALUSrc	1
MemWrite	0
MemRead	0
MemtoReg	0
ALU operation	???
PCSrc	1

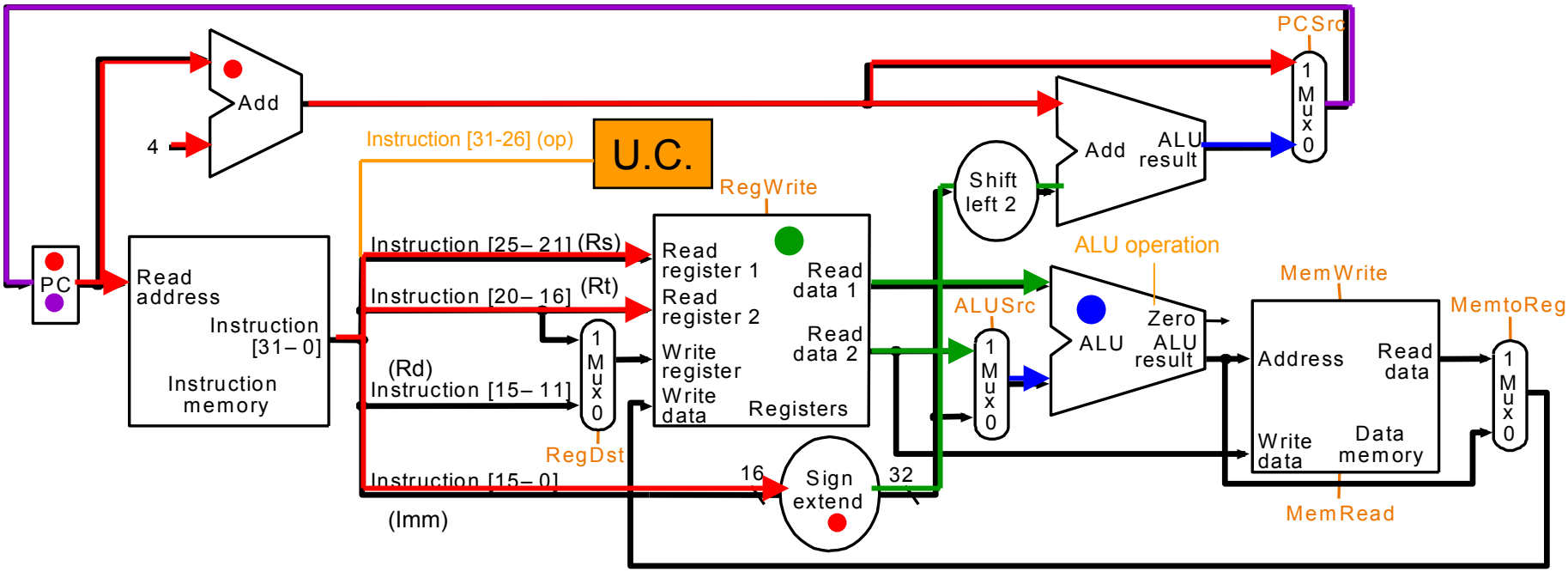
Instrução lógico-aritmética (Tipo I)



- *fetch*
- *decode/operand fetch*
- *execute*
- *writeback*

RegDst	1
RegWrite	1
ALUSrc	0
MemWrite	0
MemRead	0
MemtoReg	0
ALU operation	???
PCSrc	1

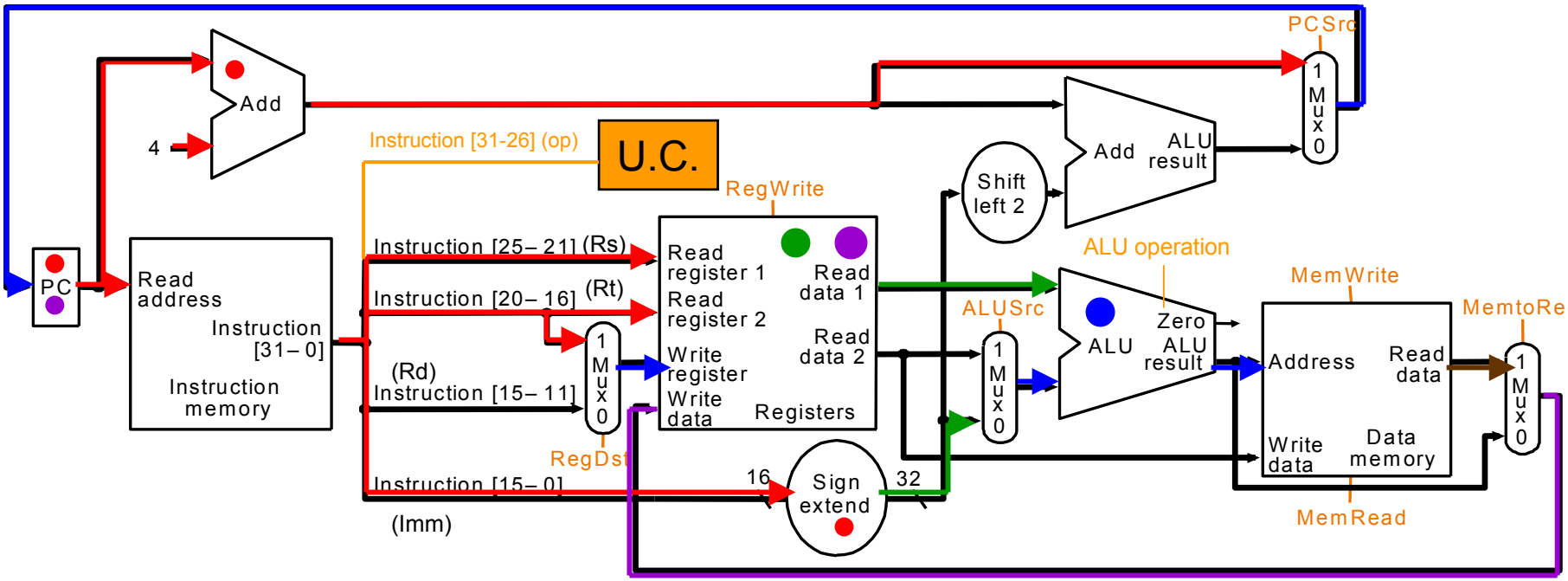
Instrução `beq`



- *fetch*
- *decode/operand fetch*
- *execute*
- *writeback*

RegDst	X
RegWrite	0
ALUSrc	1
MemWrite	0
MemRead	0
MemtoReg	X
ALU operation	110
PCSrc	Zero???

Instrução `lw`

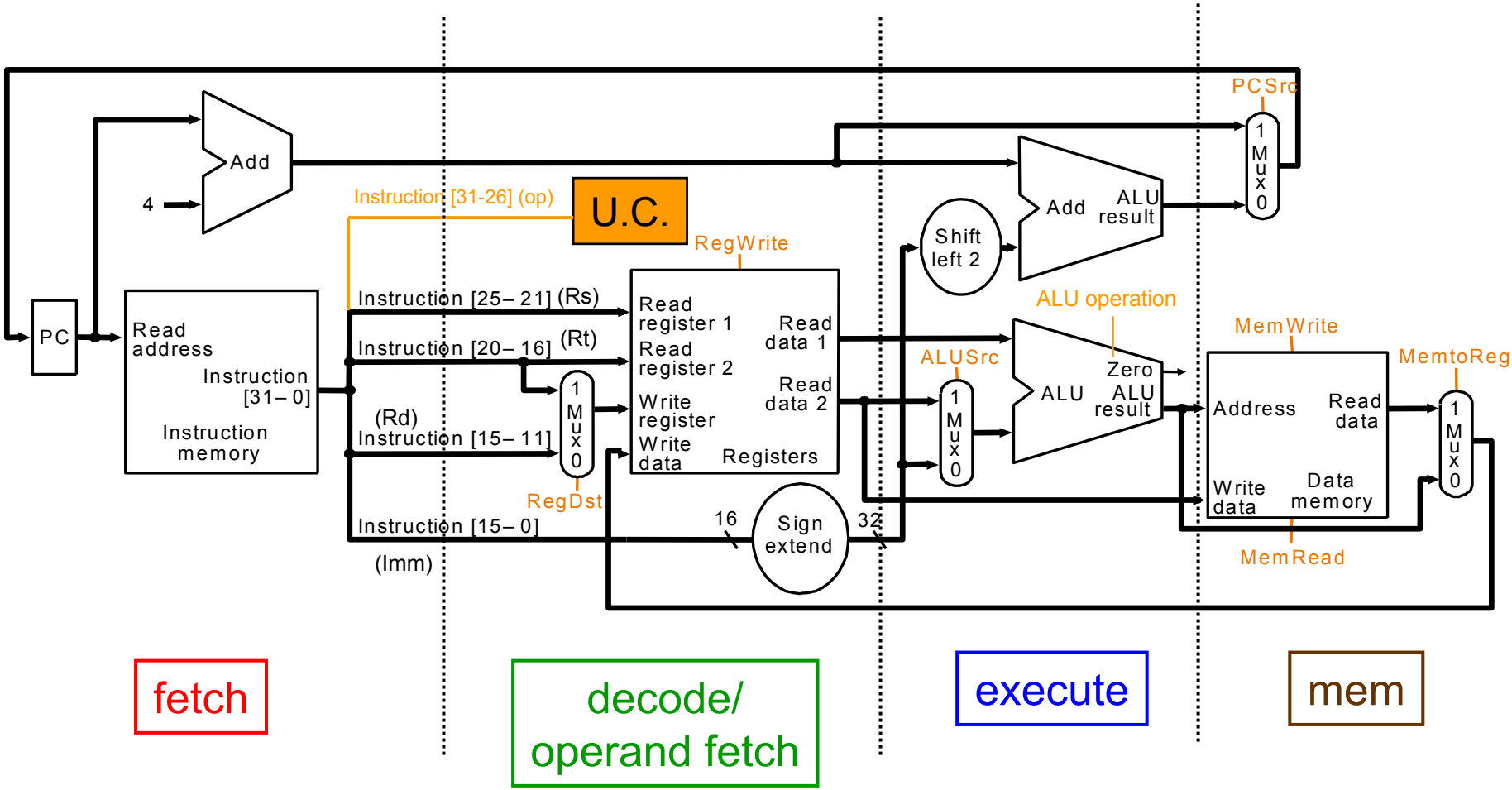


- *fetch*
- *decode/operand fetch*
- *execute*
- *mem*
- *writeback*

Exige 5 fases, todas executadas no mesmo ciclo

RegDst	1
RegWrite	1
ALUSrc	0
MemWrite	0
MemRead	1
MemtoReg	1
ALU operation	010
PCSrc	1

Datapath MIPS – 5 fases



writeback Ocorre no pulso descendente do relógio, resultando na escrita do PC e dos registos

Datapath ciclo único - Resumo

- Existem 5 fases de execução: *fetch, decode/operand fetch, execute, memory, writeback*
- Todos os sinais de controlo são gerados na fase de *decode*
- Todas as instruções demoram exactamente um ciclo a executar
- O ciclo tem que ser suficientemente longo para permitir a execução de todas as fases
- Apenas as instruções do tipo *load/store* necessitam da fase *memory*.

Corolários

1. **Todas as instruções levam tanto tempo como a instrução mais longa**
2. **Cada componente do CPU só pode ser usado uma vez por ciclo**

Sumário

Tema	H & P
Datapath single cycle	Sec. 5.1, 5.2, 5.3 (ignorar o controlo)