Desenho do Processador

Luís Nogueira

luis@dei.isep.ipp.pt

Departamento Engenharia Informática

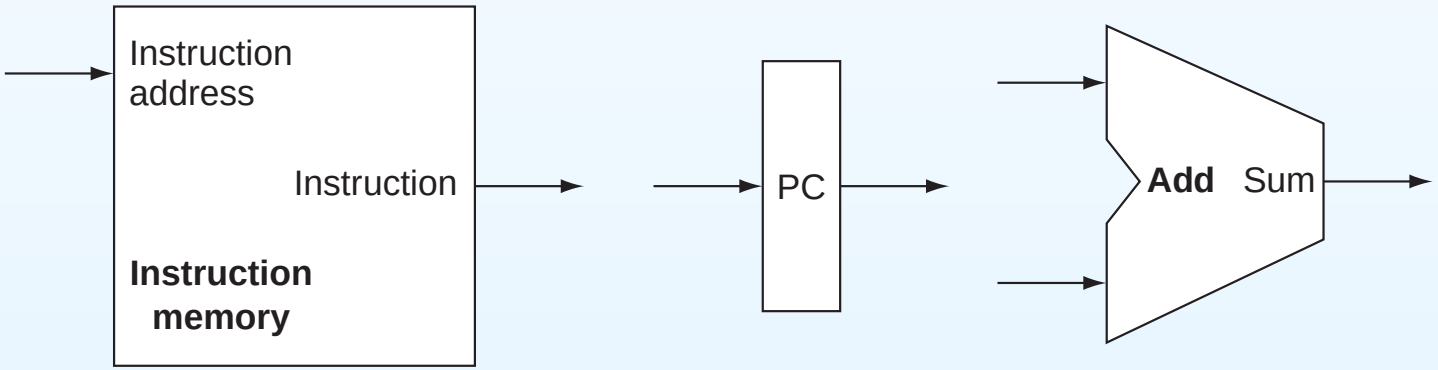
Instituto Superior de Engenharia do Porto

Introdução

- Definindo o seguinte subconjunto da ISA MIPS
  - Aritméticas e lógicas: `add, sub, and, or, slt`
  - Acesso à memória: `lw, sw`
  - Salto condicional e incondicional: `beq, bne, j`
- Desenhar um processador
  - Que componentes precisamos?
  - Como ligamos os componentes?
  - Como controlamos o fluxo de execução?
- Análise
  - ISA influencia a implementação
  - Estratégias seguidas influenciam ciclo de relógio

Componentes necessários

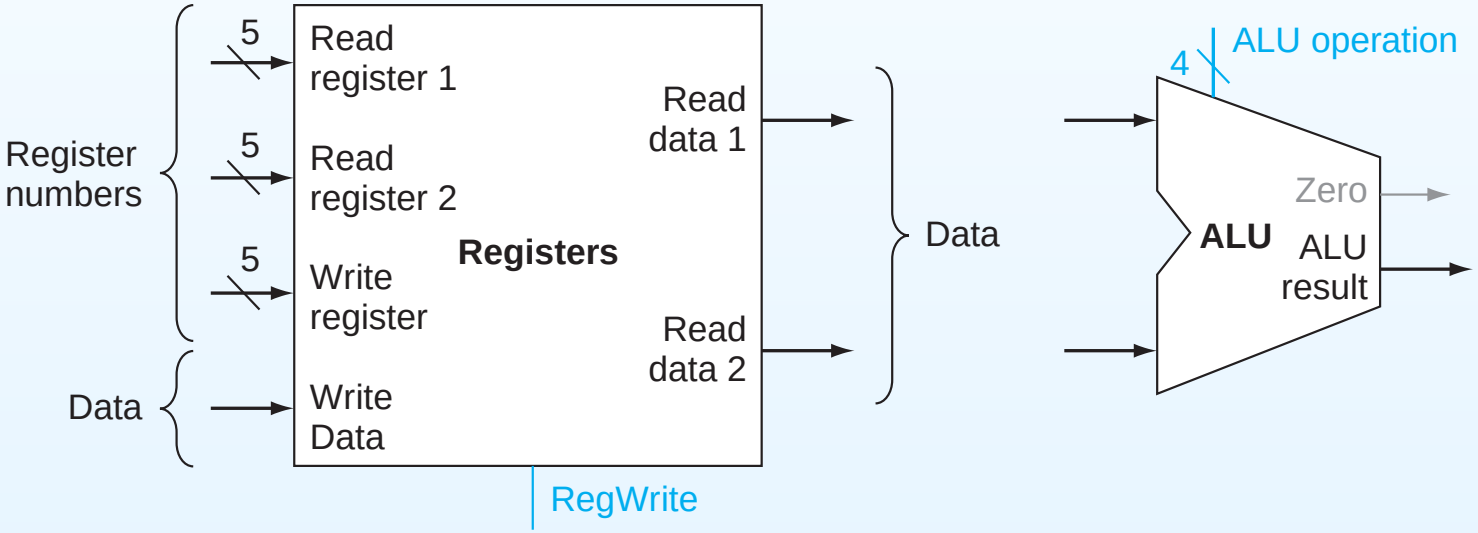
- Memória de instruções - contém programa a executar
- PC - endereço da próxima instrução
- Incrementador - avançar PC para próxima instrução



a. Instruction memory b. Program counter c. Adder

Componentes necessários

- Registos - armazenar operandos e resultados
- ALU - cálculo aritmético e lógico

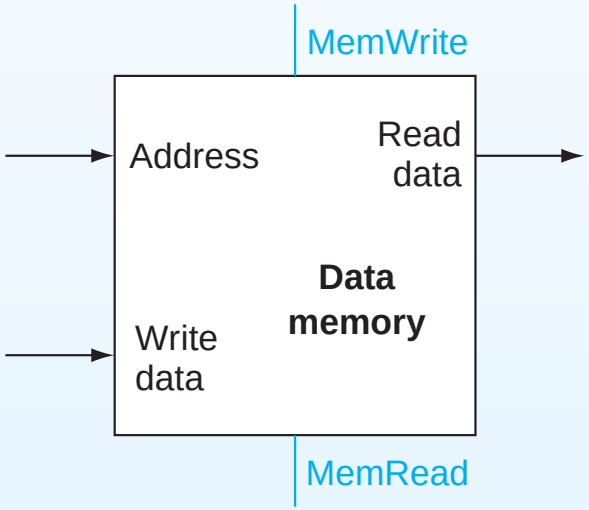


a. Registers

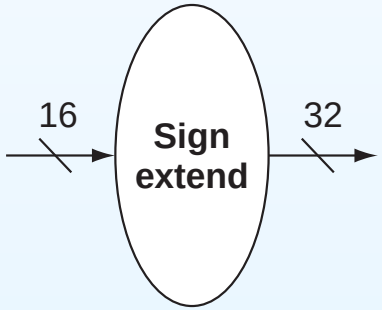
b. ALU

Componentes necessários

- Memória de dados - armazenar valores
- Conversor 16/32 bits - manipulação de endereços



a. Data memory unit



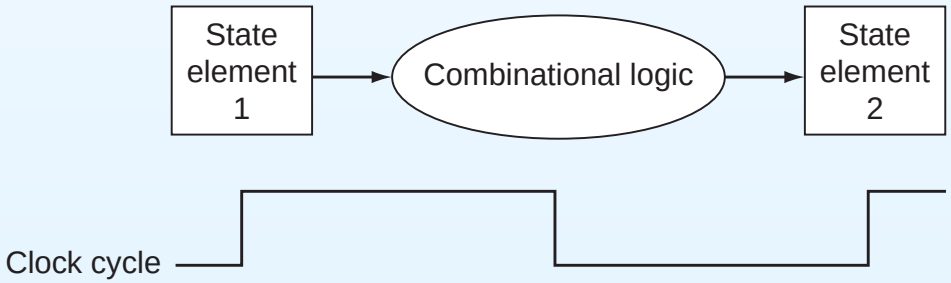
b. Sign-extension unit

Classificação dos componentes

- Elementos funcionais (lógica combinacional)
  - Dado o mesmo *input* produzem sempre o mesmo *output*
  - Não guardam informação internamente
  - `ALU, incrementador, conversor`
- Elementos com estado
  - Armazenam dados, instruções e resultados temporários
  - Contêm informação necessária à re-inicialização
  - `Registos, memórias, PC`

Sinal de relógio

- Sinal eléctrico com uma dada frequência (MHz)
- Decide momento de actualização dos elementos com estado
- Dois estados: (1) sinal activo, (0) sinal inactivo
- Sinal rectangular: mudança de estado nos lados verticais



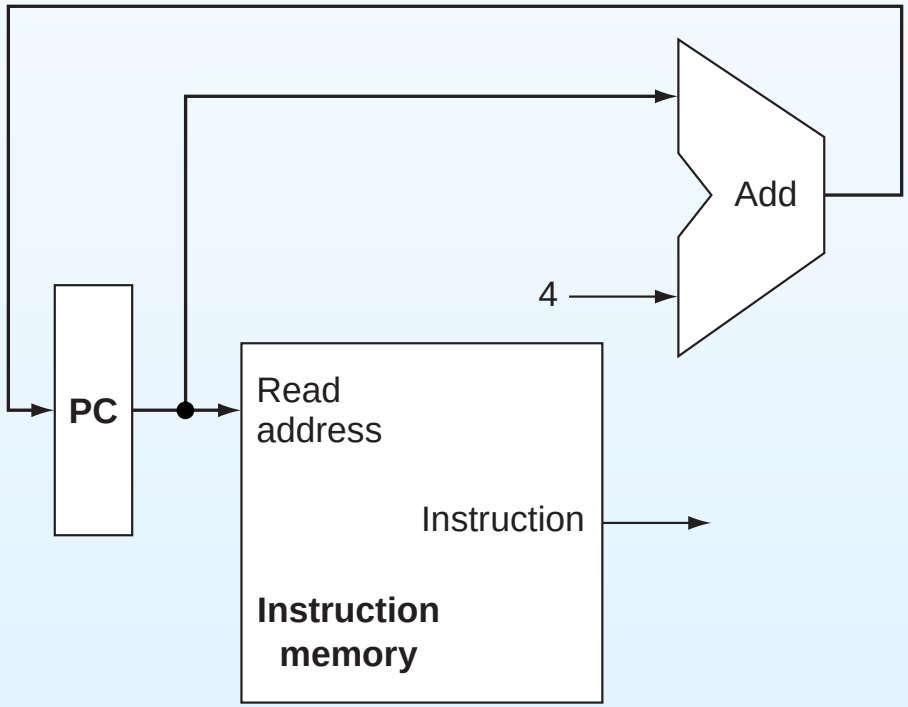
Ciclo de execução de uma instrução

1. Obter instrução da memória apontada por PC
2. Obter os operandos (endereço ou registos)
3. ALU executa operação ou calcula endereço
4. Instruções lw/sw acedem à memória
5. Resultado é armazenado
   (a) Dados da memória são escritos em registos
   (b) Resultado da ALU escrito em registo
   (c) Novo endereço escrito no PC

- Como interligar os componentes?

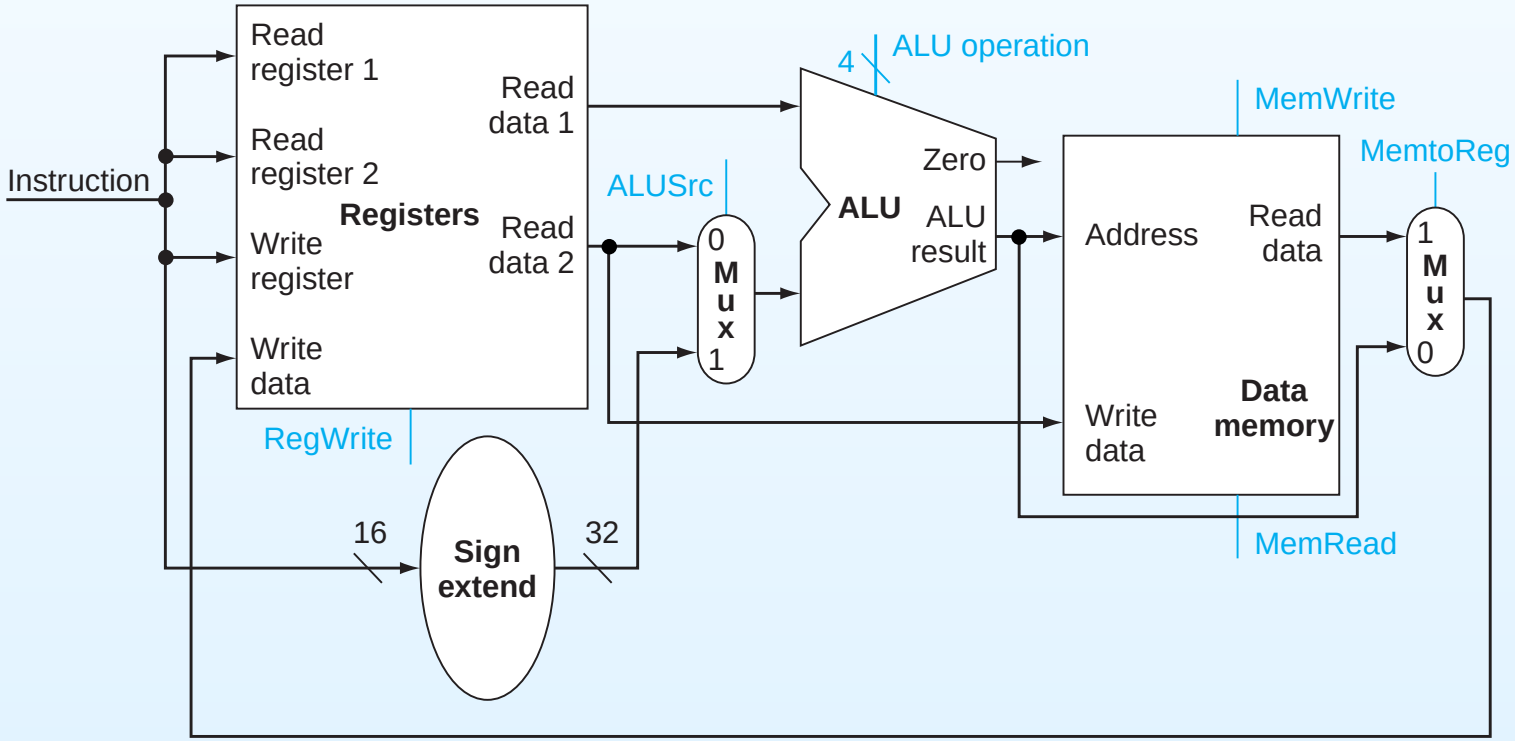
Obter instrução apontada por PC

- Aceder à memória de instruções
- Preparar a execução da instrução seguinte
  - Incrementar PC



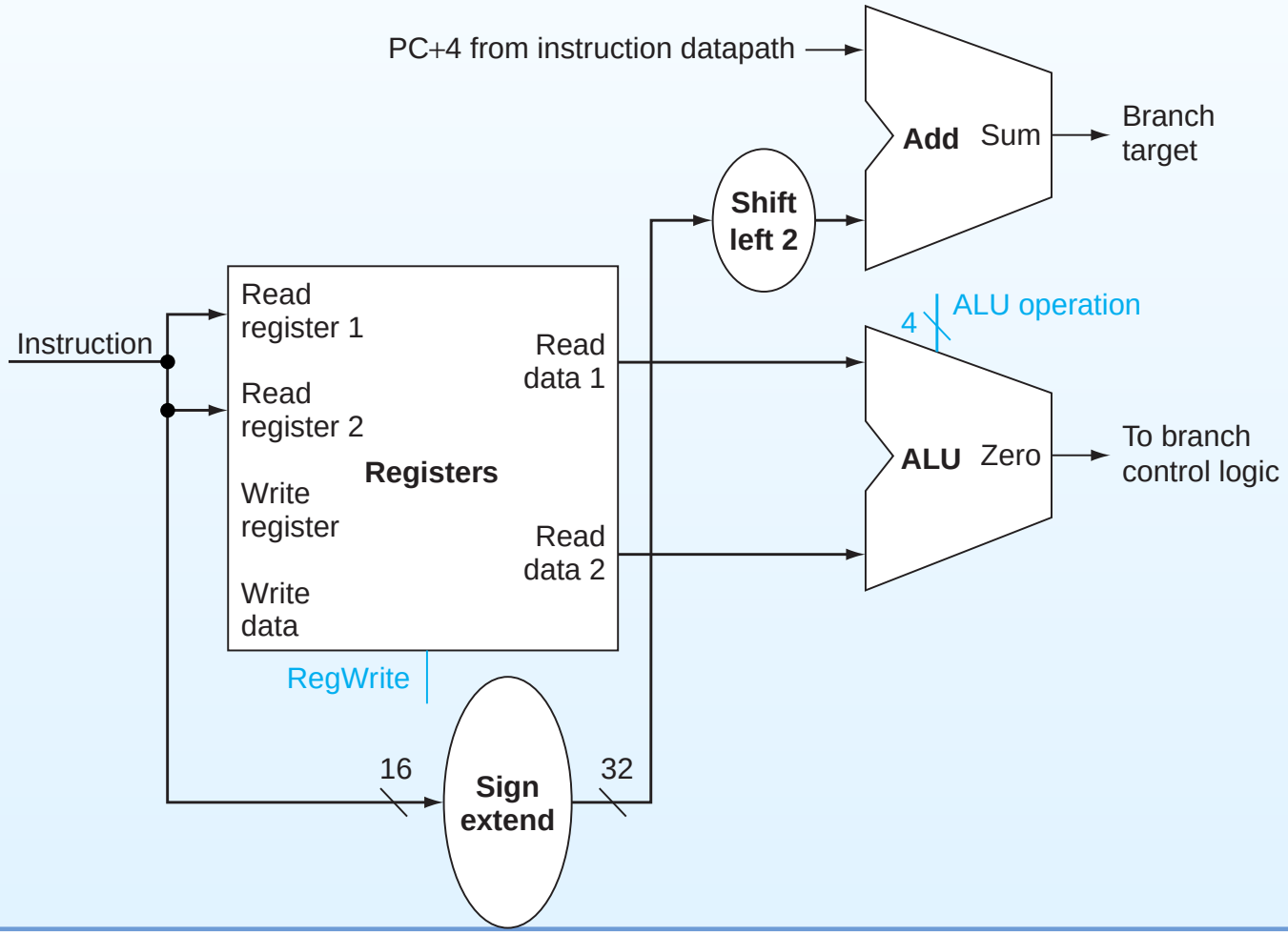
Instruções formato R e `lw/sw`

- Instruções aritméticas e de acesso à memória são similares
  - ALU calcula operação sobre registos ou endereços
  - Resultado armazenado em registo ou memória

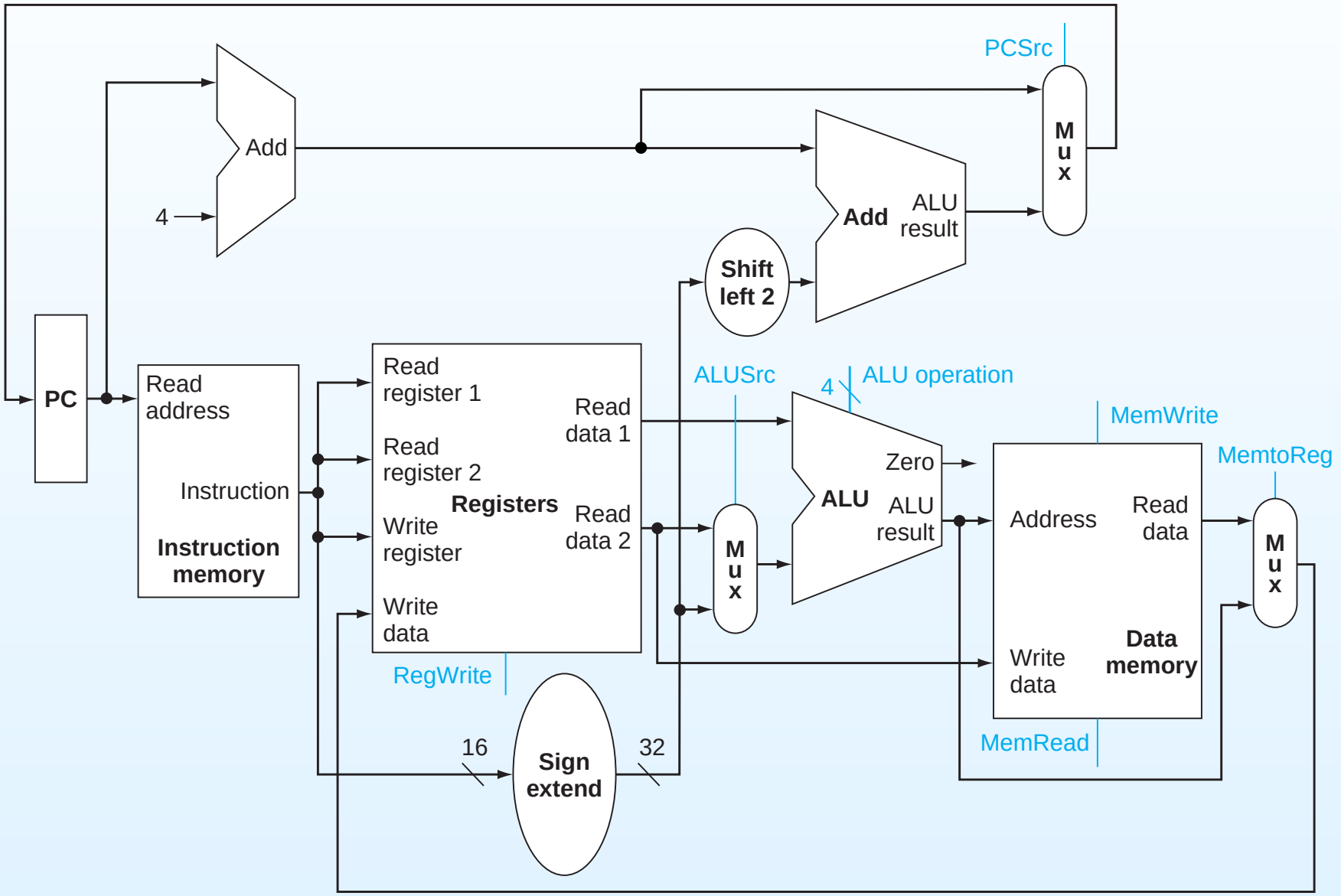


Instruções de salto condicional

- Comparar valor dos registos
- Calcular endereço destino



Arquitectura do processador



Controlo do fluxo de informação

- Necessário garantir que. . .
  - próxima instrução só começa quando actual termina
  - ALU executa operação correcta
  - durante escrita de registos não há leituras de registo
  - durante escrita na memória não há leituras da memória
  - PC avança para próxima instrução ou “salta”
  - numa instrução `lw/sw` ALU gera endereço de memória
  - após instrução aritmética o resultado é escrito num registo

Controlo do fluxo de informação

- Relógio
  - Sinal de controlo global que afecta todos os componentes
  - Determina ritmo a que a informação circula
  - Exemplo: 1 GHz equivale a um ciclo de 1 nano-segundo
- Sinais de controlo
  - Activam apenas uma parte do circuito
  - Dependem do tipo de instrução e da fase em que a instrução se encontra
  - São despoletados pelo opcode da instrução

Sinais de controlo

Formato R

opcode	rs	rt	rd	shamt	funct
31:26	25:21	20:16	15:11	10:6	5:0

Formato I

opcode	rs	rt	addr
31:26	25:21	20:16	15:0

- Registo destino
  - `rt` em `lw` nos bits 20:16
  - `rd` em instruções formato R nos bits 15:11
- Adicionar multiplexador para seleccionar campo da instrução

Sinais de controlo

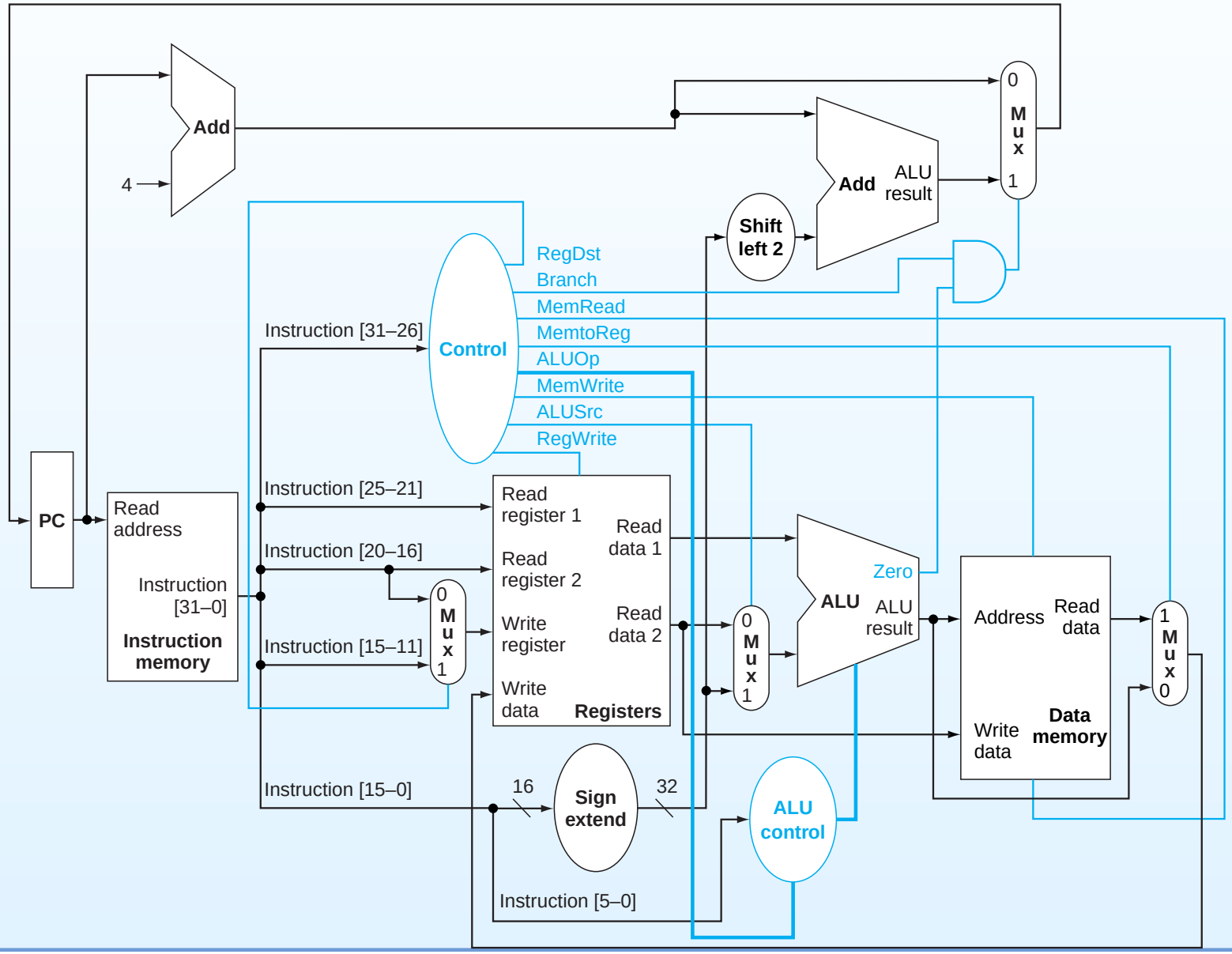
- `ALUOp` tipo de operação na ALU
- `RegDest` registo de destino
- `ALUSrc` distingue instruções formato R e I
- `PCSrc/Branch` determina se PC avança 4 bytes ou “salta”
- `MemRead` controla leitura da memória
- `MemWrite` controla escrita na memória
- `MemtoReg` selecciona resultado da ALU ou da memória
- `RegWrite` controla escrita nos registos

Sinais de controlo

Instrução	RegDest	ALUSrc	MemtoReg	RegWrite
Formato R	1	0	0	1
lw	0	1	1	1
sw	X	1	X	0
beq/bne	X	0	X	0

Instrução	MemRead	MemWrite	Branch	ALUOp
Formato R	0	0	0	10
lw	1	0	0	00
sw	0	1	0	00
beq/bne	0	0	1	01

Sinais de controlo



Controlo de fluxo `lw $1,offset($2)`

1. Instrução é obtida da memória de instruções e PC é incrementado
2. Instrução descodificada. Valor do registo `$2` é lido dos registos
   - `Branch=0`
3. ALU calcula soma do valor do registo e deslocamento convertido
   - `ALUSrc=1, ALUop=00`
4. Resultado da ALU usado como endereço na memória de dados
   - `MemRead=1, MemWrite=0`
5. Valor em endereço escrito no registo destino
   - `RegDest=0, RegWrite=1, MemtoReg=1`

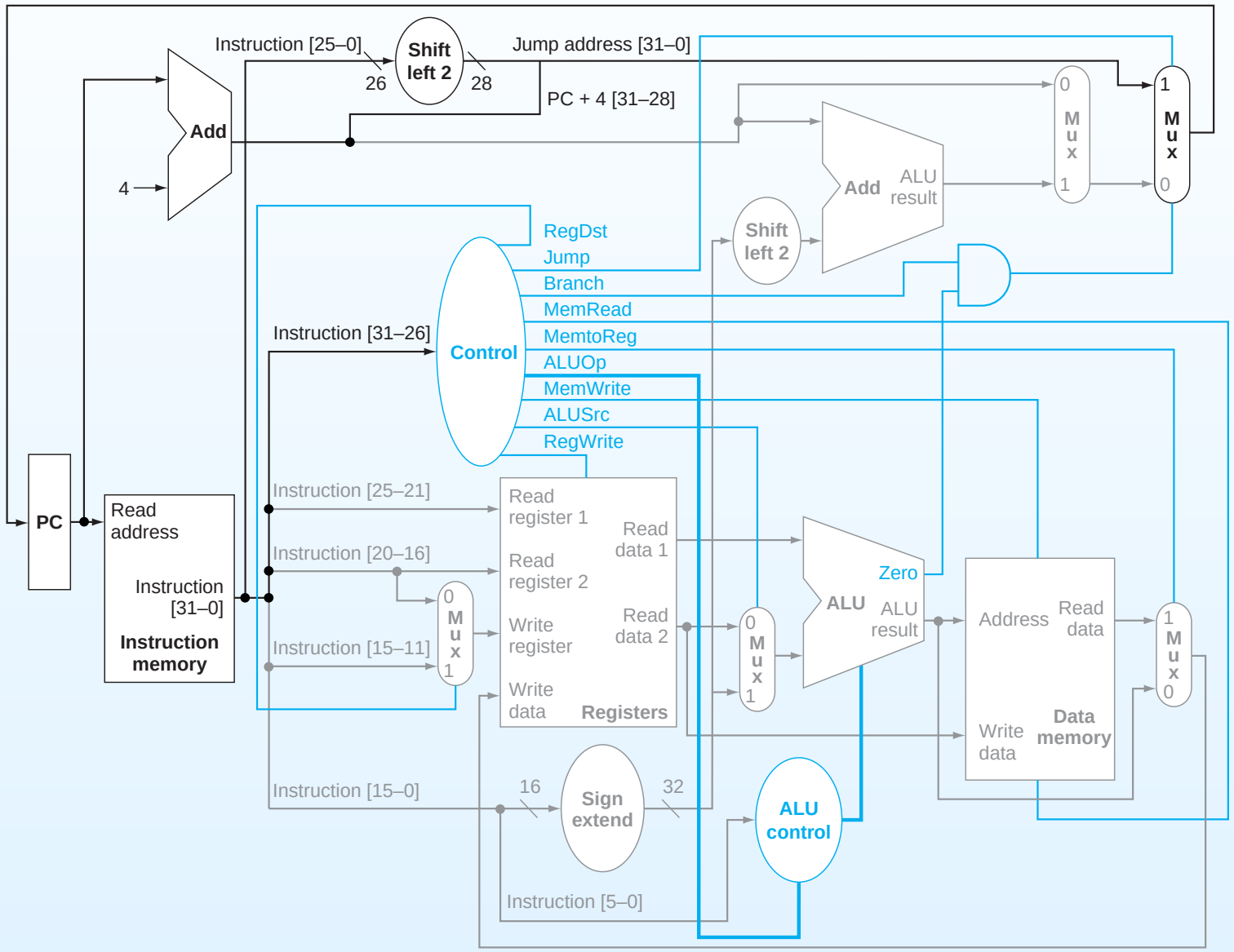
Saltos absolutos

Formato J

opcode	addr
31:26	25:0

- Endereço final
  - Conversor 26 para 28 bits
  - Últimos 4 bits são dependentes do PC
- Acrescentar ao processador
  - Sinal de controlo `Jump`
  - Multiplexador (salto condicional ou absoluto)

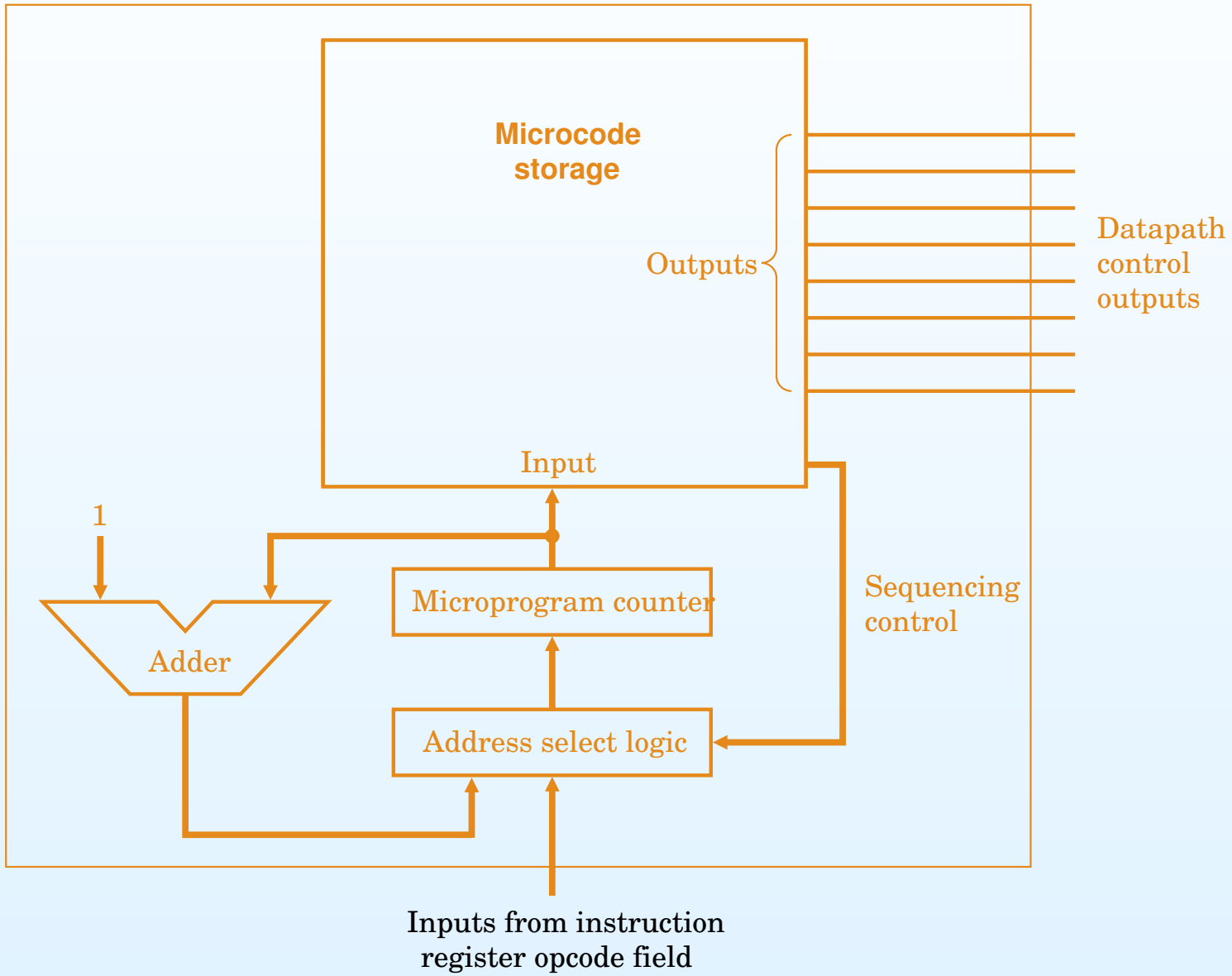
Saltos absolutos



Implementação do controlo

- Hardwired
  - Directamente nos circuitos
  - Possível se ISA tem poucas e simples instruções
  - Mais rápido
- Microcódigo
  - Em ROMs/PLAs com pequenos programas
  - Programas activam sinais de controlo em função do opcode
  - Exemplo: Intel IA-32

Controlador de microcódigo



Intel IA-32

- Instruções complexas que demoram vários ciclos de relógio
- Maior nº e complexidade de modos de endereçamento
- Adaptar fluxo de dados
  - Instruções variam no nº de ciclos de relógio
  - Reutilização de componentes numa instrução
- A partir do 80486 combinação de controlo hardwired e microcódigo
  - Hardwired para instruções simples
  - Microcódigo para instruções complexas

Intel IA-32

- Tradução de instruções complexas em micro-instruções
  - Semelhantes a instruções MIPS
- Controlo hardwired das micro-instruções
- Pentium II, Pentium III e Pentium Pro
  - Lê 3 instruções IA-32 simultaneamente
  - Usa PLAs para gerar até 6 micro-instruções
- Menor ciclo de relógio do Pentium 4 exigiu alterações
  - Se tradução origina até 3 micro-instruções$\rightarrow$trace cache
  - Maior nº de micro-instruções$\rightarrow$ ROM de micro-instruções (8000 instruções)