

Introdução à Arquitetura de Computadores

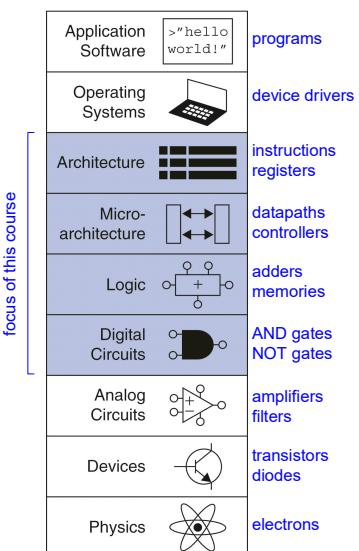
Bloco 4 O Modelo de Von Neumann

Componentes básicos de um sistema Computacional

Pedro M. Lavrador

Departamento de Electrónica, Telecomunicações e Informática
Universidade de Aveiro
plavrador@ua.pt

Onde Estamos?



Índice

- Conceitos fundamentais em Arquitectura de Computadores
 - Os elementos básicos de um computador
 - Modelo de Von Neumann
 - A noção de instrução e o ciclo básico de execução
- Arquitetura de Computadores
 - *Arquitetura do Conjunto de Instruções*
- Introdução à Arquitetura MIPS

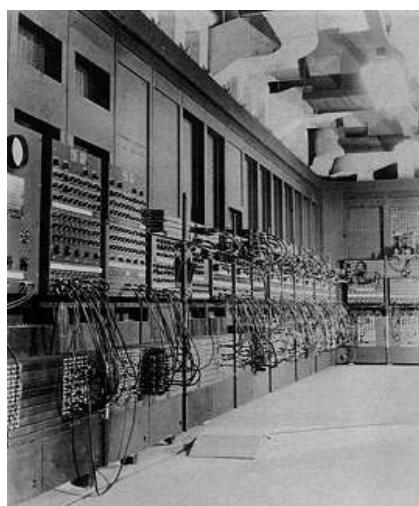
23/03/2023

PML – IAC - 2023

3

ENIAC: Primeiro Computador Eletrónico

- Electronic Numerical Integrator and Computer (1946)



23/03/2023

PML – IAC - 2023

4

ENIAC: Primeiro Computador Eletrónico

- Era utilizado para realizar cálculos balísticos. Podia fazer até 5000 operações por segundo.
- O problema maior era reconfigurá-lo para fazer uma operação diferente. Podia demorar dias a reprogramar.
- John Von Neumann propôs a armazenagem dos dados e do programa na mesma memória.
 - Deste modo os programas poderiam ser guardados e reaproveitados.

23/03/2023

PML – IAC - 2023

5

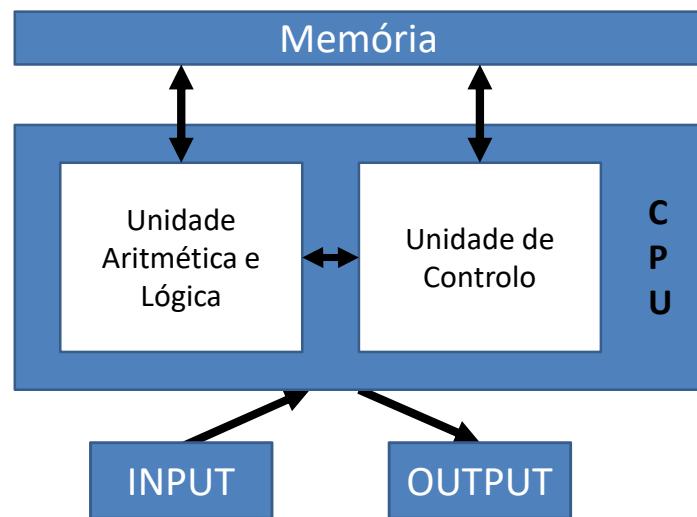
Mas... quais os blocos básicos que constituem uma arquitetura computacional genérica?

23/03/2023

PML – IAC - 2023

6

Modelo de Von Neumann



23/03/2023

PML – IAC - 2023

7

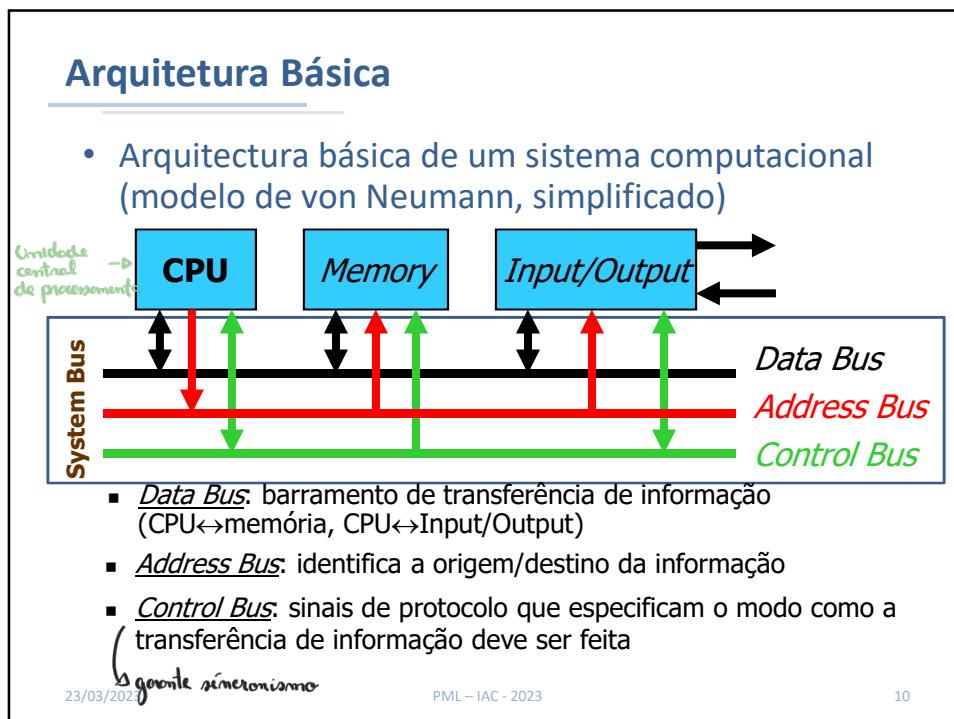
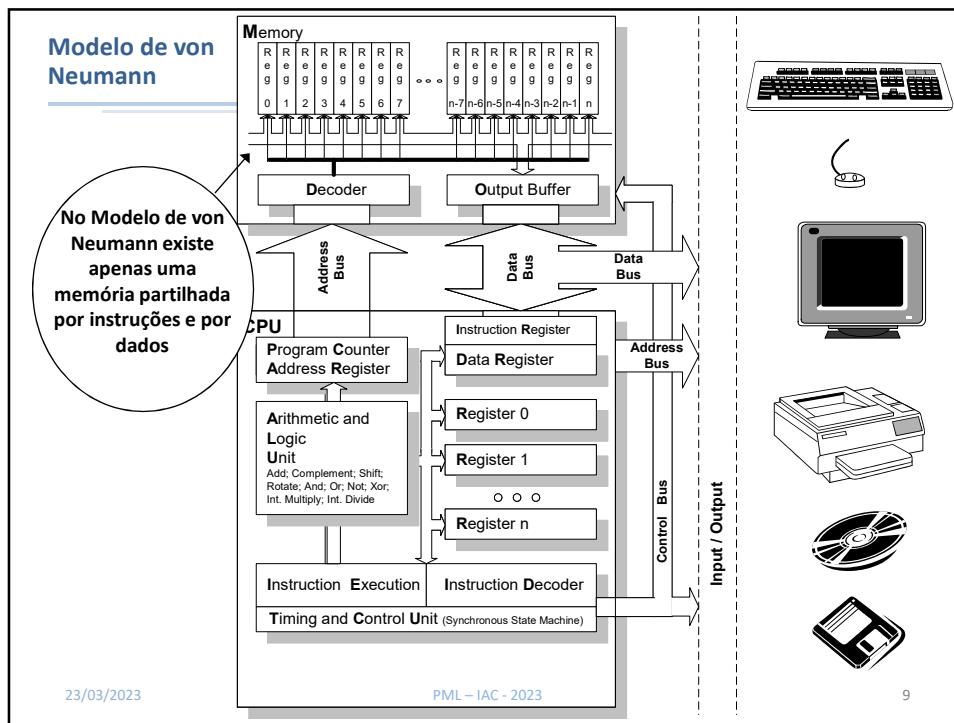
Modelo Básico de um Sistema Computacional

- As unidades fundamentais que constituem um computador são:
 - Unidades de entrada – permitem a recepção de informação vinda do exterior (dados, programas) e que é armazenada em memória
 - Unidades de saída – permitem o envio de resultados para o exterior
 - Memória – armazenamento de:
 - Programas
 - Dados para processamento (*dados de entrada/saída*)
 - Resultados
 - CPU – processamento da informação através da execução do programa armazenado em memória

23/03/2023

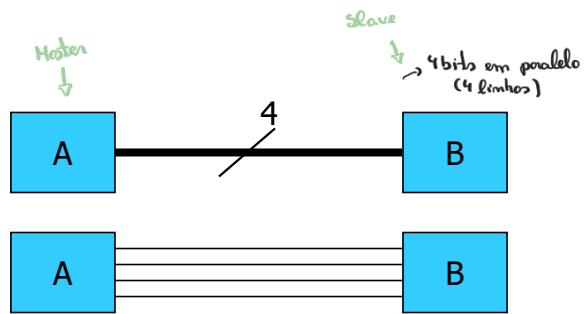
PML – IAC - 2023

8



Barramento

- Barramento (*bus*)= colecção de fios agrupados segundo uma dada função; cada fio transporta informação relativa a 1 bit. Ex. – barramento de 4 bits:



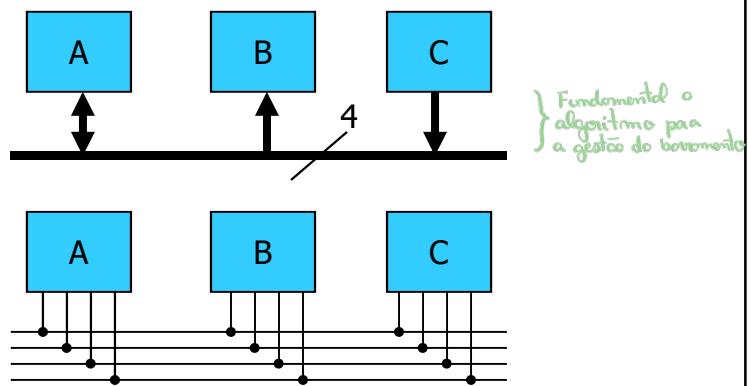
23/03/2023

PML – IAC - 2023

11

Barramento Partilhado

- Barramento partilhado (*shared bus*)= barramento que interliga diferentes blocos funcionais



23/03/2023

PML – IAC - 2023

12

Unidade de Processamento (CPU)

- A unidade central de processamento é organizada em várias unidades:
 - **Unidade de Controlo**
 - Responsável pela sequenciação da execução de cada instrução
 - **Unidade Aritmética e Lógica**
 - Executa as operações (somas, subtrações, operações lógicas...)
 - **Registros**
 - Armazenamento temporário de dados
 - Operandos e resultados de operações
 - Valores de Controlo (p.e. endereços especiais)
 - **Comprimento da Palavra** (p.e. processador de 32 bits)
 - Número de bits processados pela ALU; número de bits dos registos.

São a memória mais rápida

O processador funciona com m bits

23/03/2023

PML – IAC - 2023

13

Unidade Central de Processamento

- O CPU consiste, fundamentalmente, em duas secções:
 - **Secção de dados (datapath)**: elementos operativos:
 - Registos internos
 - Unidade Aritmética e Lógica (ALU) → calculadora aritmética + lógia
 - **Unidade de controlo**: responsável pela coordenação dos elementos do datapath, durante a execução de um programa
 - **Máquina de estados síncrona** (estado seguinte é função do estado atual e das entradas)
 - As entradas correspondem a informação retirada de cada uma das instruções lidas da memória

Máquina de Moore

controla os estados do sistema
dependendo das instruções
e do estado atual

23/03/2023

PML – IAC - 2023

14

Unidade de Controlo

- Controla a execução do programa:
 - Determina a operação a ser realizada no presente
 - Determina a próxima operação a ser realizada
- A unidade de controlo lê uma instrução da memória (*Instruction Fetch*) e interpreta a instrução (*Instruction Decode*) gerando os sinais de controlo que indicam à unidade de processamento o que fazer.
 - A tarefa a realizar pode ser completada num ciclo de relógio ou precisar de vários ciclos.
- **Instruction Register:** → Mesa de trabalho
 - Contem a Instrução que está a ser executada
- **Program Counter**
 - Contem o endereço da próxima instrução



23/03/2023

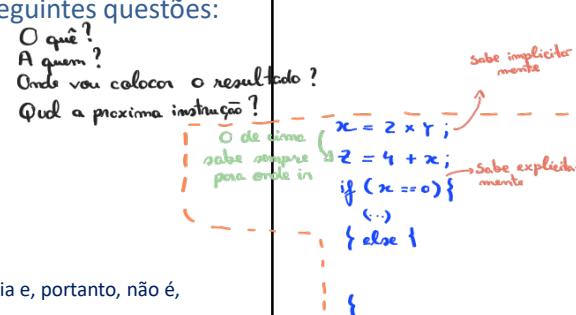
PML – IAC - 2023

15

A noção de “Instrução”

- Independentemente do tipo de CPU e da sua estrutura interna, qualquer instrução deve permitir responder às seguintes questões:
 - Qual a operação a realizar ?
 - Qual a localização dos operandos (se existirem) ?
 - reg. Internos / memória
 - Onde colocar o resultado ?
 - reg. Internos / memória
 - Qual a próxima instrução ?
 - em condições normais é a instrução seguinte na sequência e, portanto, não é normalmente, explicitamente mencionada
 - em instruções que alteram a sequência de execução a instrução deverá fornecer o endereço da próxima instrução a ser executada

↳ instrução tem indicações explícitos e implícitos



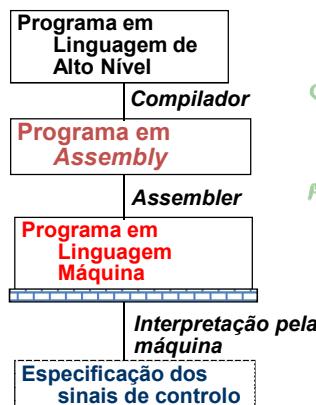
23/03/2023

PML – IAC - 2023

16

Como Representar as Instruções?

- Níveis de representação de instruções



temp = v[k];
 v[k] = v[k+1];
 v[k+1] = temp; } Programação alto nível
 lw \$15, 0(\$2)
 lw \$16, 4(\$2)
 sw \$16, 0(\$2)
 sw \$15, 4(\$2) } Assembly → Humano lê
 0000 1001 1100 0110 1010 1111 0101 1000
 1010 1111 0101 1000 0000 1001 1100 0110
 1100 0101 1010 1111 0101 1000 0000 1001
 0101 1000 0000 1001 1100 0110 1010 1111 } Código Máquina → Máquina lê
 ALUOP[0:3] <= InstReg[9:11] & MASK

23/03/2023

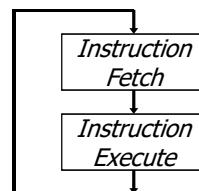
PML – IAC - 2023

17

Ciclo básico de execução de uma instrução

- 1º O processador acede à memória e lê a próxima instrução a ser executada.
- 2º O processador executa a Instrução
 - Acede aos operandos
 - Realiza a operação sobre eles
 - Guarda o resultado
- Chama-se a este processo o ciclo de leitura e execução.

• acede à memória
 • lê a instrução



23/03/2023

PML – IAC - 2023

18

Memória

CPU

Índice

- Conceitos fundamentais em Arquitectura de Computadores
 - Os elementos básicos de um computador
 - Modelo de Von Neumann
 - O ciclo básico de execução de uma instrução
- Arquitetura de Computadores
 - *Arquitetura do Conjunto de Instruções*
- Introdução à Arquitetura MIPS

23/03/2023

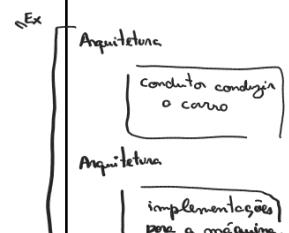
PML – IAC - 2023

19

Arquitetura de Computadores

- Arquitetura de Computadores =
 - Arquitetura do Conjunto de Instruções (ISA)
 - +
 - Organização da Máquina
- Conjunto (*Set*) de Instruções:
 - a coleção de todas as operações que o processador pode executar
- Que estrutura de processador se define para executar o Conjunto de Instruções?
- Microarquitetura:
 - A organização do processador, incluindo as principais unidades funcionais e respetivas ligações e controlo.
 - A uma arquitetura podem corresponder várias microarquitecturas diferentes.

1 Arquitetura Pode conter várias microarquiteturas



23/03/2023

PML – IAC - 2023

20

Arquitetura do Conjunto de Instruções

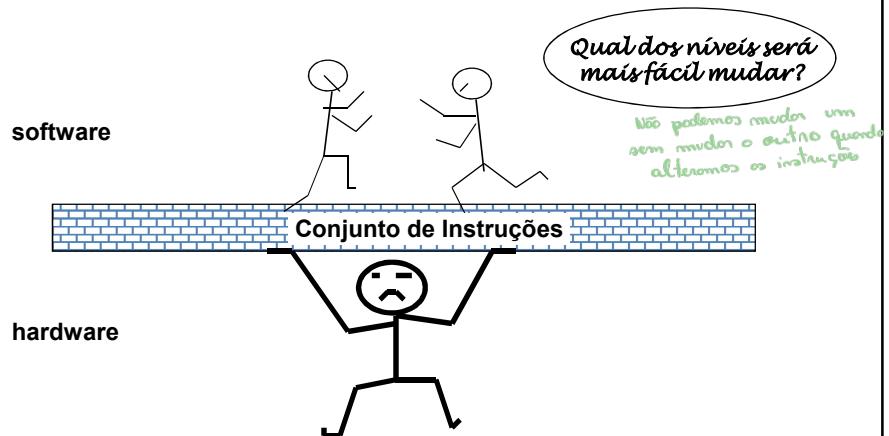
- Também designada por "modelo de programação":
 - Uma abstração que representa a interface entre o *hardware* e o nível mais básico de *software*
- Descreve tudo o que o programador necessita de saber para programar corretamente, em linguagem máquina, um determinado processador
- Descreve a funcionalidade, independentemente do hardware que a implementa.
 - A organização do fluxo de dados e da unidade de controlo são do nível dos Sistemas Digitais, enquanto a sua implementação é do nível da MicroElectrónica.

23/03/2023

PML – IAC - 2023

21

Conjunto de Instruções: um Interface Crítico



23/03/2023

PML – IAC - 2023

22

Uma Arquitetura Múltiplas Implementações

- Pode falar-se de "arquitetura" e "implementação de uma arquitetura"
 - (Ex. Processadores AMD compatíveis com Intel x86)

Ox processadores são implementados de diferente maneira mas oferecem a mesma arquitetura.
- A manutenção da arquitetura mantém a compatibilidade com o software mais antigo enquanto permite melhorias de performance.
 - Por exemplo, a arquitetura atual x86 ainda tem como base a original de 1978

(Mantêm os quatro registos de 16 bits)

• Agora têm os quatro registos de N bits

*• 32...
• 64...*

23/03/2023

PML – IAC - 2023

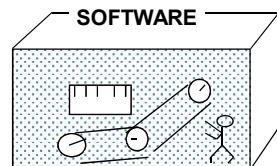
23



Arquitetura do Conjunto de Instruções

Aspetos a definir numa arquitetura:

- Quais as instruções suportadas
- Como organizar a memória (e os acessos)
- Quantos registos
 - Registros específicos ou gerais
- Tipos de dados e estruturas suportadas
- Modos de endereçamento e de acesso a dados e instruções
- Qual o formato das instruções
- Condições de Exceção «Ex: Divisão por "0"
- Atender às condições imprevistas



23/03/2023

PML – IAC - 2023

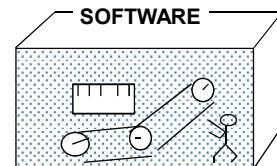
24

Arquitetura do Conjunto de Instruções

- Fatores a ter em conta no desenho de uma arquitetura:

- As aplicações a que se destina
- A linguagem de programação
- O sistema operativo
- As possibilidades tecnológicas
- A compatibilidade histórica

→ Tornadeira? Previsões balísticas?



- Objetivos de uma arquitetura:

- Implementação eficiente e simples em hardware
- Fácil de entender e programar
- Compiladores eficientes

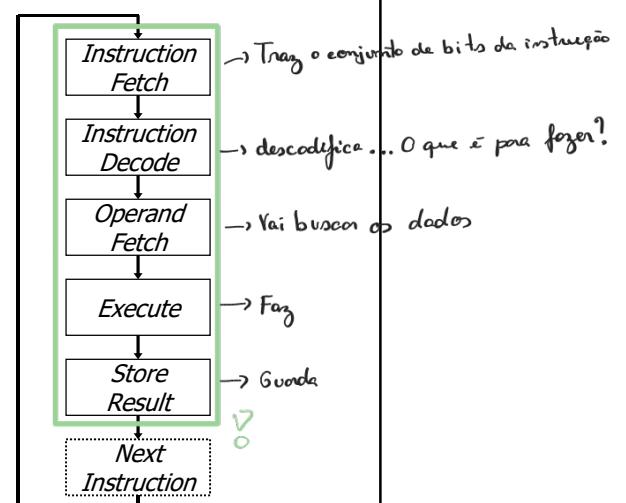
23/03/2023

PML – IAC - 2023

25

Ciclo básico de execução de uma instrução

Fetch-execute cycle



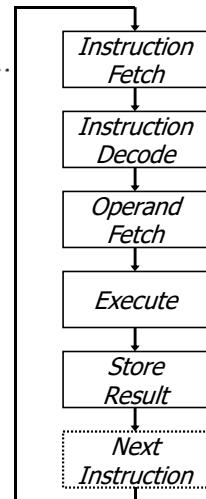
23/03/2023

PML – IAC - 2023

26

Arquitetura de Computadores

- Arquitetura do conjunto de instruções define:
 - Formato e codificação das instruções
 - como são descodificadas? → Quantos bits de operação, ...
 - Localização de operandos e resultados
 - onde?
 - quantos operandos explícitos?
 - como localizar?
 - quais podem residir na memória externa?
 - Tipo e dimensão dos dados
 - Operações
 - quais devem ser suportadas?
 - Instruções auxiliares
 - jumps, conditions, branches
 - fetch-decode-execute (implícito)!



23/03/2023

PML – IAC - 2023

27

Variáveis da Arquitetura de Computadores

- Intel* →
- Depende da aplicação*
- MIPS (32 bits)* →
- Formato das instruções → Como armam as instruções?
 - Tamanho variável
 - Código mais pequeno
 - maior flexibilidade
 - *Instruction fetch* em vários passos
 - Tamanho fixo
 - *Instruction fetch* e *decode* mais simples
 - Mais simples de implementar em *pipeline*
 - código maior
- *add op1 op2 destino*
- *jump Target*
- Tamanhos diferentes
- Não acontece, pois depende...
- O processador sabe facilmente antes da execução a ordem e quantos instruções temos

Dilema:

1 operação complexa	vs	Várias operações simples
---------------------	----	--------------------------

↳ Depende da aplicação simples

Variáveis da Arquitetura de Computadores

- Número de registos: muitos ou poucos?
 - Vantagens de um número pequeno de registos
 - Menos hardware
 - Acesso mais rápido
 - Menos bits para identificação do registo
 - Mudança de contexto mais rápida → várias de todos os variáveis guardadas no processador
 - Vantagens de um número elevado de registos
 - Menos acessos à memória → demandar mais a aceder, mas não tanto de ir à memória para ir buscar variáveis em outros programas
 - Variáveis em registos → demora mais a acceder, mas não tanto
 - Certos registos podem ter restrições de utilização

Mais uma vez ... depende ☺

unidade de memória básica → implementados dentro do próprio processador

Menos dificuldade em endereços

Mudar de programa A para programa B → mudar todo o que está no processador, buscar contexto da B → cover apl. B

Na instrução preciso de mais bits para dizer o destino pois os bits de registo são superiores

Mais bits de endereço

chegada de uma instrução, por exemplo

Guarda tudo o que está no processador, buscar contexto da B → cover apl. B

Acumulador é portátil

↓

Mais depois eu preciso de guardar resultado

↑ para guardar resultado

Instruções básicas para colocar os dados na pilha

add

Mais instruções

Variáveis da Arquitetura de Computadores

- Localização dos operandos
 - Acumulador
 - Resultado das operações é armazenado num registo especial designado de acumulador → destino está sempre no acumulador
 - Baseados em *Stack*
 - Operandos e resultado armazenados numa *stack* de registos → *metemos na pilha e depois sumamos*
 - Register-Memory*
 - Operandos residem em registos ou em memória
 - Load-store architecture*
 - Operandos residem em registos de uso geral.

Mais comum

23/03/2023

PML – IAC - 2023

30

15

Exemplos de ISAs (*Instruction Set Architecture*)

- Intel x86
 - Computadores Pessoais
 - Servidores
- MIPS → Vamos estudar?
 - Equipamentos de rede (CISCO)
 - Sistemas Embebidos (sensores, ...)
- IBM
 - Mainframes
- ARM
 - Sistemas Embebidos

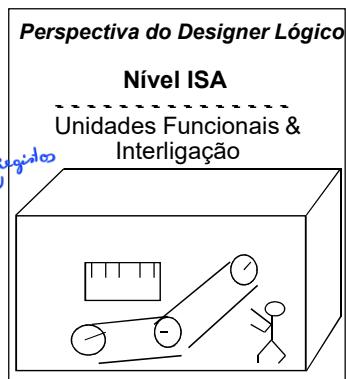
23/03/2023

PML – IAC - 2023

31

Organização da máquina

- Características operativas e de performance das principais unidades funcionais
 - (ALU, Registos, Shifters, Unidades Lógicas, ...)
- De que modo esses componentes são interligados → como pessoas os comunicam?
 - ALU → Registos
- Fluxo de informação entre componentes
- Lógica e meios através dos quais esse fluxo é controlado
- Coreografia das Unidades Funcionais para implementar a *Instruction Set Architecture*.



23/03/2023

PML – IAC - 2023

32

Resumindo...

- Arquitetura é a visão que o programador tem do computador
 - Define-se (basicamente) pelo conjunto de instruções e localização dos operandos
- Microarquitetura é o modo de implementar no hardware a arquitetura.

23/03/2023

PML – IAC - 2023

33

Resumindo...

- O conjunto de instruções
 - É o conjunto de todas as instruções que um processador pode implementar
- Diferentes processadores têm diferentes conjuntos de instruções *→ Pode dar origem a vários arquiteturas*
 - Mas muitos aspectos são comuns (pelo menos ao nível dos conceitos).

23/03/2023

PML – IAC - 2023

34

Índice

- Conceitos fundamentais em Arquitectura de Computadores
 - Os elementos básicos de um computador
 - Modelo de Von Neumann
 - O ciclo básico de execução de uma instrução
- Arquitetura de Computadores
 - *Arquitetura do Conjunto de Instruções*
- Introdução à Arquitetura MIPS

23/03/2023

PML – IAC - 2023

35

Arquitetura MIPS

- Foi desenvolvida nos anos 80, em Stanford, por John Hennessy.
 - É usada atualmente pela Cisco, Nintendo e em muitos sistemas de computação dedicada (controladores, etc.)
 - Em 2004 já tinham sido vendidos mais de 300 milhões de processadores MIPS.

→ Agora já existem outras versões
 - É uma arquitetura RISC (Reduced Instruction Set Computer)
- Intel & CISC ☺

23/03/2023

PML – IAC - 2023

36

Linguagem Assembly

- Instruções: são comandos em linguagem de computador
 - Linguagem *assembly* é formato das instruções para humano ler.
 $\text{add } \$t0, \$t1, \$t2$
memorética
 - Linguagem máquina é o formato das instruções para computador ler.
 $0x012a4020$
corresponde à memorética
e aos pogramas
- Depois de aprender uma arquitetura é fácil aprender outras por comparação.

23/03/2023

PML – IAC - 2023

37

Critérios de Seleção de um Conjunto de Instruções (ISA)

- A escolha de uma arquitetura para o conjunto de instruções deve garantir: Revisões:
 - A simplicidade da máquina que o implementa;
 - A clareza da sua aplicação aos problemas que realmente importam;
 - Uma solução tão rápida quanto possível dos problemas.
- **1º Princípio: A regularidade favorece a Simplicidade**
- **2º Princípio: Smaller is Faster**

23/03/2023

PML – IAC - 2023

38

As instruções da arquitetura MIPS

- Adição: $a = b + c;$
- A instrução *assembly* correspondente é:

add a, b, c

mmmc DST, OP1, OP2 → instrução em MIPS

- **add** é a mnemónica que indica a operação a realizar
- **b** e **c** são os operandos fonte
- **a** é o operando destino (quem guarda o resultado)

23/03/2023

PML – IAC - 2023

39

As instruções da arquitetura MIPS

- Subtração: $a = b - c;$
- A instrução *assembly* correspondente é similar à adição, apenas muda a mnemónica:

sub a, b, c

- **sub** é a mnemónica (indica a operação)
- **b** e **c** são os operandos fonte
- **a** é o operando destino

23/03/2023

PML – IAC - 2023

40

1º Princípio de Design

- Todas as instruções aritméticas seguem este padrão.
 - Dois operandos fonte e um destino
- **A simplicidade favorece a regularidade.**
 - O formato das instruções é sempre consistente
 - São mais simples de codificar e de processar em hardware

23/03/2023

PML – IAC - 2023

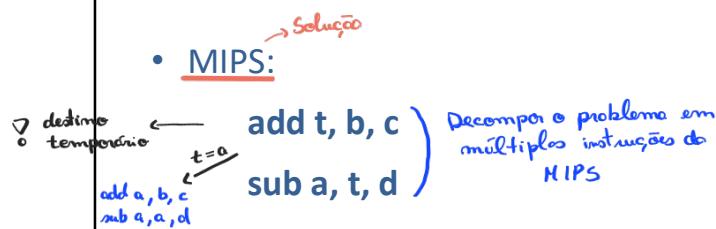
41

Múltiplas Instruções

- E como se resolve:

$$a = b + c - d;$$

- Operações mais complexas são codificadas em múltiplas instruções do MIPS.



23/03/2023

PML – IAC - 2023

42

2º Princípio de Design

- Tornar rápido o caso comum
 - O MIPS apenas tem instruções simples e frequentes
 - O hardware que implementa essas instruções simples é simples, pequeno e rápido
 - As operações mais complexas (menos comuns) são executadas usando múltiplas instruções simples
 - Exemplo: somar 2 operandos em memória.
- $lw t1, op1$
 $lw t2, op2$
 $add dst, t1,t2$
- O MIPS é um processador RISC
- Outras arquiteturas, como a família x86, segue uma filosofia CISC (Complex Instruction Set Computer)

23/03/2023

PML – IAC - 2023

43

Operандos

- Os operandos podem estar localizados:
 - Nos registos internos
 - Na memória (apenas nas instruções de acesso à memória)
 - lw e sw Apenas as operações load e store permitem operandos na memória
 - Na própria instrução (constantes)

 $add $t0, $t1, $t2$ $\# \$t0 = \$t1 + \$t2$ $addi $t0, $t1, imm$ $\# \$t0 = \$t1 + imm$ constante

23/03/2023

PML – IAC - 2023

44

Operandos: Registros

- O MIPS tem 32 registos de 32 bits.
 - Numerados de 0 a 31;
 - Com palavras de 32 bits
- O acesso a informação contida nos registos é mais rápido do que o acesso à memória.
 - Porque “Smaller is Faster”
 - É mais rápido aceder a 1 de 32 registos do que a 1 posição de entre milhões na memória.
- A arquitetura MIPS é de 32 bits porque opera com dados de 32 bits.

• Registos estão no processador
• Tempos de acesso, quantos mais bits de localização




23/03/2023

PML – IAC - 2023

45

Os registos do MIPS

- Registos:
 - São indicados com o símbolo \$
 - Por exemplo \$0, “o registo zero” ou “dólar zero”
- Alguns registos são usados com funções específicas:
 - \$0 guarda a constante zero
 - Os registos \$s0-\$s7 guardam variáveis (chamam-se saved registers)
 - Os registos \$t0-\$t9 guardam valores temporários.

* Aprendemos mais para a frente

23/03/2023

PML – IAC - 2023

46

Os Registros do MIPS

Name	Register Number	Usage
\$0	0	the constant value 0
\$at	1	assembler temporary
\$v0-\$v1	2-3	Function return values
\$a0-\$a3	4-7	Function arguments
\$t0-\$t7	8-15	temporaries
\$s0-\$s7	16-23	saved variables
\$t8-\$t9	24-25	more temporaries
\$k0-\$k1	26-27	OS temporaries
\$gp	28	global pointer
\$sp	29	stack pointer
\$fp	30	frame pointer
\$ra	31	Function return address

→ Era tão utilizada que tem um registro próprio \$0

23/03/2023

PML – IAC - 2023

47

A instrução add de novo

- Adição: $a = b + c;$
- $\$t0 \rightarrow a$
- $\$t1 \rightarrow b$
- $\$t2 \rightarrow c$

add \$t0, \$t1, \$t2

23/03/2023

PML – IAC - 2023

48

Operandos: Memória

- Um problema real não pode ser resolvido apenas com 32 registos de memória.

- Para armazenar mais dados usa-se a memória externa.
- A memória é grande, mas lenta...

Algunas variáveis vão para a memória e outras para os registos

Muitos ou menos utilizados

ex: Array vai para a memória

- Então...

- Procura-se que as variáveis usadas mais frequentemente sejam guardadas em registos.

- Para efetuar operações aritméticas com operandos em memória

- 1º carregam-se os dados em registos
- 2º faz-se a operação
- 3º Guarda-se o resultado

23/03/2023

PML – IAC - 2023

49

Operandos em Memória

- A memória é usada para armazenar dados

- Podem ser arrays, estruturas, etc.

Cada endereço de uma posição de memória é 1 byte → Mm, para aceder são precisos 4 bytes

- A memória no MIPS é byte addressable.

- É possível endereçar cada byte individualmente.

- Cada **palavra** (4 bytes) em memória tem que ser armazenada num endereço múltiplo de 4

- Alinhamento das palavras em memória

Alinhamento das palavras: Todos os endereços têm 1 byte

- Problema:

- Qual a ordem dos bytes? Isto é, o byte mais significativo coloca-se no endereço de memória mais alto ou mais baixo?

\$t0 → 0x 23 45 67 89

0, 1, 2 e 3
endereços

23/03/2023

PML – IAC - 2023



Temos de ir pelo endereço 4

56 múltiplos de 4

Ou começa pelo menos ou mais significativo para a posição 0 e depois 1, 2 e 3

Endereçamento de Palavras na Memória

- Alguns processadores usam representação Big-Endian, isto é, o fim do número está no endereço mais alto.
 - Ou seja o byte menos significativo do número está no endereço de memória mais alto.
- Outros usam representação Little-Endian, isto é, o fim do número está no endereço de memória mais baixo:
 - Ou seja o byte menos significativo do número está no endereço de memória mais baixo.
- Exemplo para o valor 0x23456789, guardado no endereço 0:



23/03/2023

PML – IAC - 2023

51

Instruções para acesso à memória

- Acesso à memoria para leitura: ***load***
- Mnemónica: **load word (lw)**
- Formato:
 - **lw \$t0, 8(\$t1)**
- Determinação do endereço:
 - Soma-se ao endereço base (o valor contido em **\$t1**) o valor **8**
 - Endereço = **\$t1+8**
- Resultado:
 - O registo **\$t0** fica com a palavra armazenada na memória no endereço **\$t1+8**.

23/03/2023

PML – IAC - 2023

*↳ Se for um endereço múltiplo de 4, 8
↳ Caso contrário da expo ou avise!*

Instruções para acesso à memória

- Exemplo ler a palavra do endereço 4 para o registo \$t2.

– `lw $t2, 4($0)`

- \$t2 fica com o valor 0xF2F1AC07

Address	Data	
:	:	:
12 - 0000000C	4 0 F 3 0 7 8 8	Word 3
8 - 00000008	0 1 E E 2 8 4 2	Word 2
4 - 00000004	F 2 F 1 A C 0 7	Word 1
0 - 00000000	A B C D E F 7 8	Word 0

ex: `lw $t0, 0($0)` # \$t0 = mem(0 + \$0)

23/03/2023

PML – IAC - 2023

53

Instruções para acesso à memória

- Acesso à memoria para escrita: **store**

- Mnemónica: store word (sw)

- Formato:

– `sw $t0, 4($0)`

- Determinação do endereço:

- Tal como no lw

- Resultado:

- O valor contido no registo \$t0 é escrito na memória na posição 4.

Exceção à Regra O destino vem depois do operando

ex: `sw $t0, 0($0)` # Mem(0 + \$0) = \$t0

23/03/2023

PML – IAC - 2023

54

Instruções para acesso à memória

- Já vimos que o MIPS é ***byte addressable***.
- Então deve ser possível fazer **load** e/ou **store** de apenas um byte.
- Isso é feito com as instruções **lb** e **sb**

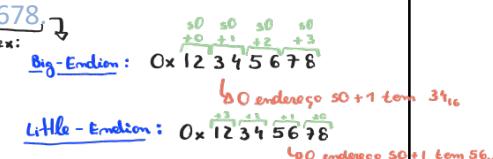
23/03/2023

PML – IAC - 2023

55

Big Endian, Little Endian?

- Numa máquina Big Endian, qual o resultado (em \$t1) das seguintes instruções:
 - Suponha que \$t0 contém 0x12345678
 - sw \$t0, 0(\$s0)
 - lb \$t1, 1(\$s0)
- \$t0 vai ser armazenado nos endereços a partir de \$s0
 - \$s0+0: 0x12 0b 0001 0010
 - \$s0+1: 0x34
 - \$s0+2: 0x56
 - \$s0+3: 0x78
- Então \$t1 assume o valor 0x34.



Problema do Endiânia:

Já está resolvida
atualmente

23/03/2023

PML – IAC - 2023

56

Operandos podem ser:

- Registros
- Endereços
- Immediato

Operandos: Constantes/Imediatos

- As instruções lw/sw usam constantes.
- São valores que estão imediatamente disponíveis na instrução
 - Codificados como um número representado em complemento para 2 com 16 bits.
- As instruções aritméticas e lógicas também podem usar imediatos.
- Por exemplo:
 - $a = a + 4;$ \rightarrow addi \$t0, \$t0, 4
 - Ib Immediato

23/03/2023

PML – IAC - 2023

57

Linguagem Máquina

- A representação binária das instruções.
 - Os computadores só entendem 0's e 1's.
- As instruções (no MIPS) são codificadas em palavras de 32 bits:
 - A simplicidade favorece a regularidade:
 - Palavras de 32 bits para dados e instruções.
- Existem 3 tipos de instruções possíveis:
 - Tipo R: para operandos em registros add / sub
 - Tipo I: para um operando imediato memi / lw / sw
 - Tipo J: para saltos

↳ Dentro de uma instrução: para a prox. operação terem bits reservados, pode ser de qualquer tipo R,I,J

23/03/2023

PML – IAC - 2023

58

↳ O computador codifica esses formatos antes de os executar