DCC007 – Organização de Computadores II

Aula 2 – Arquitetura do Conjunto de Instruções - ISA

Prof. Omar Paranaiba Vilela Neto

ACM A.M Turing Award

PIONEERS OF MODERN COMPUTER ARCHITECTURE RECEIVE ACM A.M. TURING AWARD

Hennessy and Patterson's Foundational Contributions to Today's Microprocessors Helped Usher in Mobile and IoT Revolutions

ACM named John L. Hennessy, former President of Stanford University, and David A. Patterson, retired Professor of the University of California, Berkeley, recipients of the 2017 ACM A.M. Turing Award for pioneering a systematic, quantitative approach to the design and evaluation of computer architectures with enduring impact on the microprocessor industry. Hennessy and Patterson created a systematic and quantitative approach to designing faster, lower power, and reduced instruction set computer (RISC) microprocessors. Their approach led to lasting and repeatable principles that generations of architects have used for many projects in academia and industry. Today, 99% of the more than 16 billion micro-processors produced annually are RISC processors, and are found in nearly all smartphones, tablets, and the billions of embedded devices that comprise the Internet of Things (IoT).

Instruction Set Architecture

- Classificação do conjunto de instruções
- Observar como aplicações utilizam ISA
- Examinar ISA de processadores modernos
- Medições de utilização de ISA em computadores reais

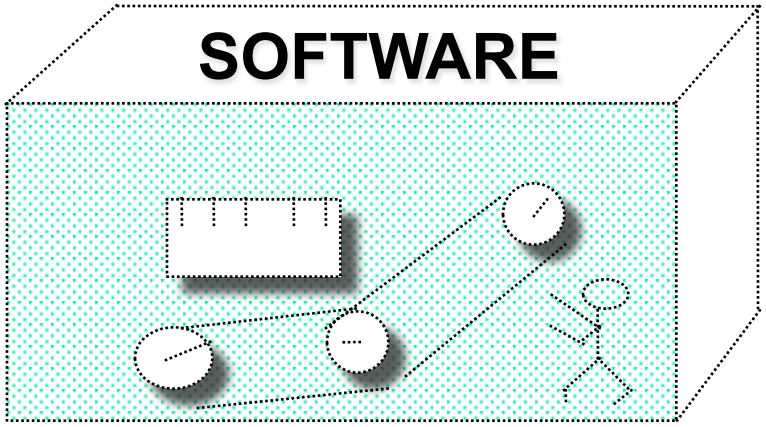
Instruction Set Architecture

- Anos 50s aos 60s: Arquitetura de computadores == aritmética de computadores
- Anos 70 a metade dos 80s: Arquitetura de computadores == projeto do conjunto de instruções, especialmente ISA apropriado para compiladores
- Anos 90s: Arquitetura de computadores == projeto de CPU, subsistema de memória, subsistema de I/O, multiprocessadores

Arquitetura de computadores

"os atributos de um sistema de computação na visão do programador, i.e., a estrutura conceitual e o comportamento funcional, …
em oposição à implementação física."

Amdahl, Blaaw, and Brooks, 1964



Avaliação de ISA e sua Organização

Conjunto de instruções: parte do processador visível ao programador ou pessoa que desenvolverá compiladores (o que você enxerga do hardware)

Evolução do Conjunto de Instruções

- Maiores avanços em arquitetura de computadores é associado a mudanças no projeto dos conjunto de instruções
  - Ex: Pilha vs GPR (Sistema 360)
- Decisões devem levar em conta:
  - tecnologia
  - organização de máquina
  - linguagens de programação
  - tecnologia de compiladores
  - sistemas operacionais
  - E eles influenciam...

Espaço de Projeto de uma ISA

Dimensões primárias

- **Número de operandos explícitos - (0, 1, 2, 3)**
- **Armazenamento de operandos - Onde além da memória?**
- **Endereço efetivo - Como endereço de memória é especificado?**
- **Tipo e tamanho dos operandos - byte, int, float, vector, . . .**
  Como especificá-lo?
- **Operações - add, sub, mul, . . .**
  Como especificá-lo?

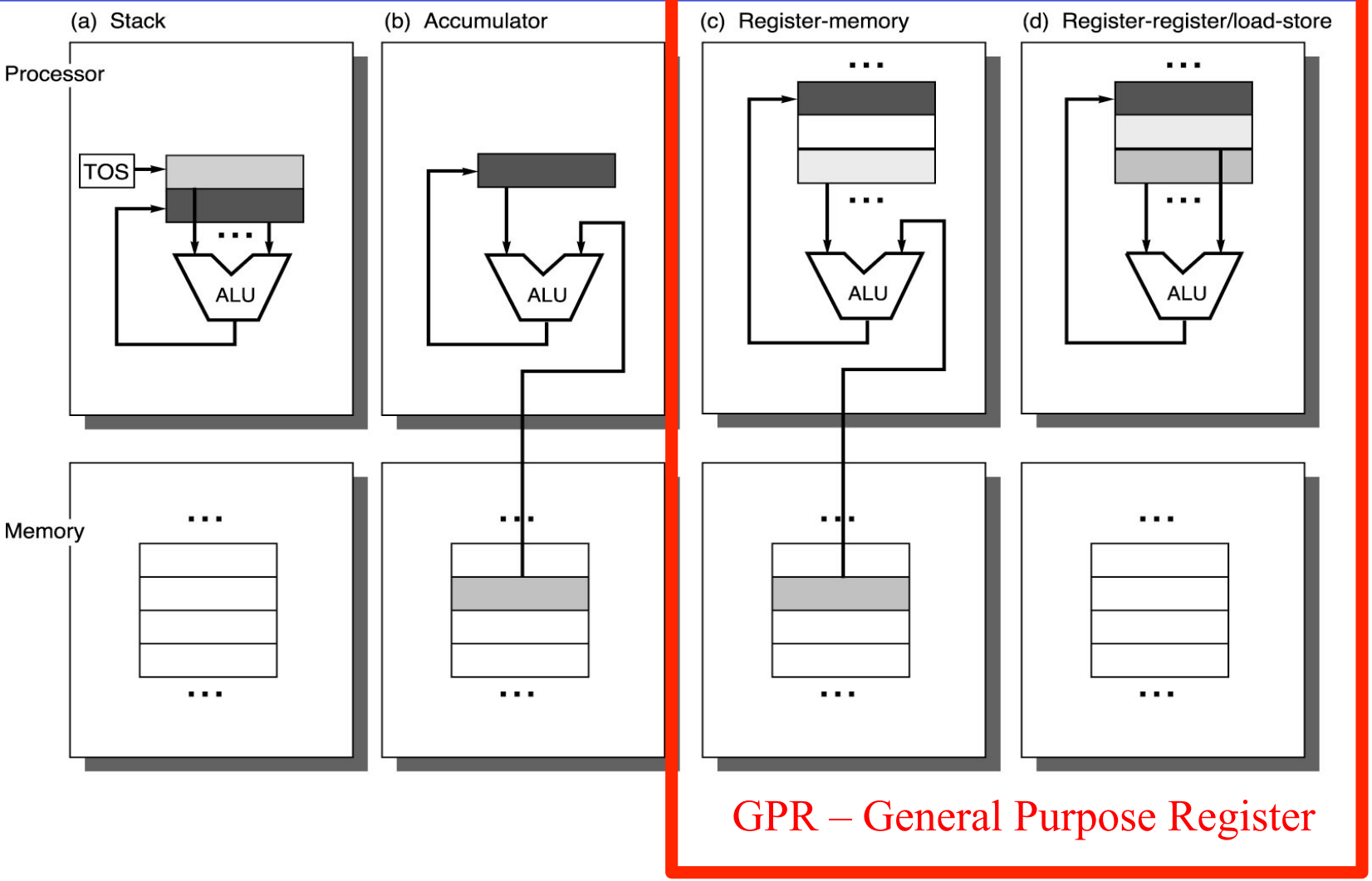
Outros aspectos

- **Sucessor** **Como é especificado?**
- **Condições** **Como são determinadas?**
- **Codificações** **Fixa ou variável? *Wide*?**
- **Paralelismo**

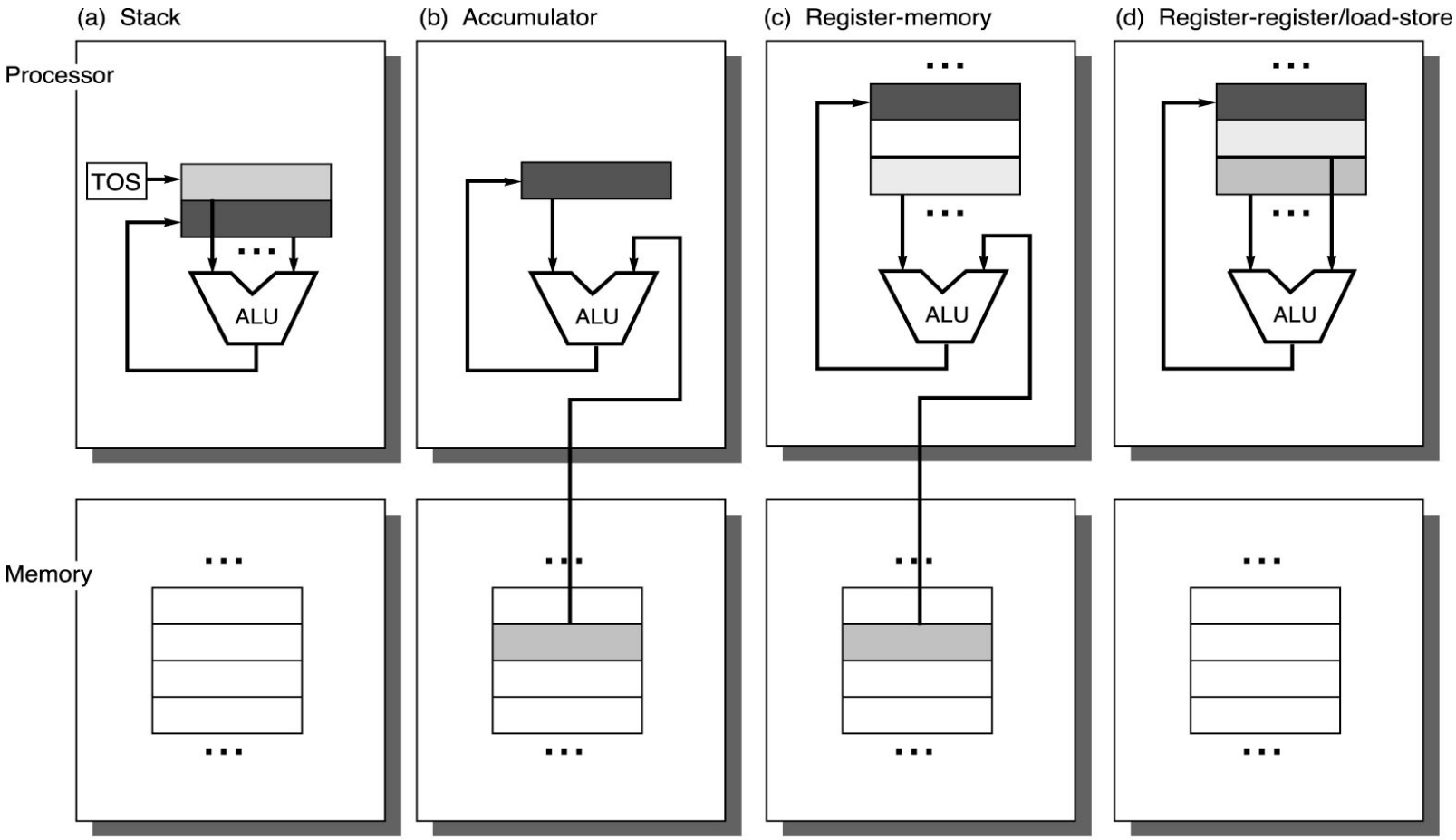
Métricas de um ISA

- Ortogonalidade
  - Nenhum registrador especial, poucas condições especiais, todos os modos de operando disponíveis com qualquer tipo de dado ou tipo de instrução
- Completude
  - Suporta gama variada de operações e aplicações
- Regularidade
  - Não devem haver overloadings para os diferentes campos da instrução
- Fácil entendimento
  - Recursos necessários devem ser facilmente determinados
- Facilidade de compilação (programação?)
- Facilidade de implementação
- Escalabilidade

Classes de ISAs



Classes de ISAs



# de Operandos explícitos	0	1	2 ou 3	2 ou 3

Classes de ISAs

Acumulador:

1 endereço add A acc ← acc + mem[A]

Pilha:

0 endereços add tos ← tos + próximo

General Purpose Register:

2 endereços add A B EA(A) ← EA(A) + EA(B)

3 endereços add A B C EA(A) ← EA(B) + EA(C)

0 endereços (Load/Store)

add Ra Rb Rc Ra ← Rb + Rc

load Ra Rb Ra ← mem[Rb]

store Ra Rb mem[Rb] ← Ra

Acumulador

- Operando é o registrador acumulador

  A = B + C

  - ld B
  - add C
  - sto A

- Atributos
  - Instruções compactas
  - Estado interno mínimo, projeto interno simples
  - Alto tráfego de memória, muitos loads e stores

- Exemplos:
  - IBM 7090, DEC PDP-8, MOS 6502, 8085

Máquinas de Pilha

- Conjunto de instrução:

  +, -, *, /, . . .

  push A, pop A

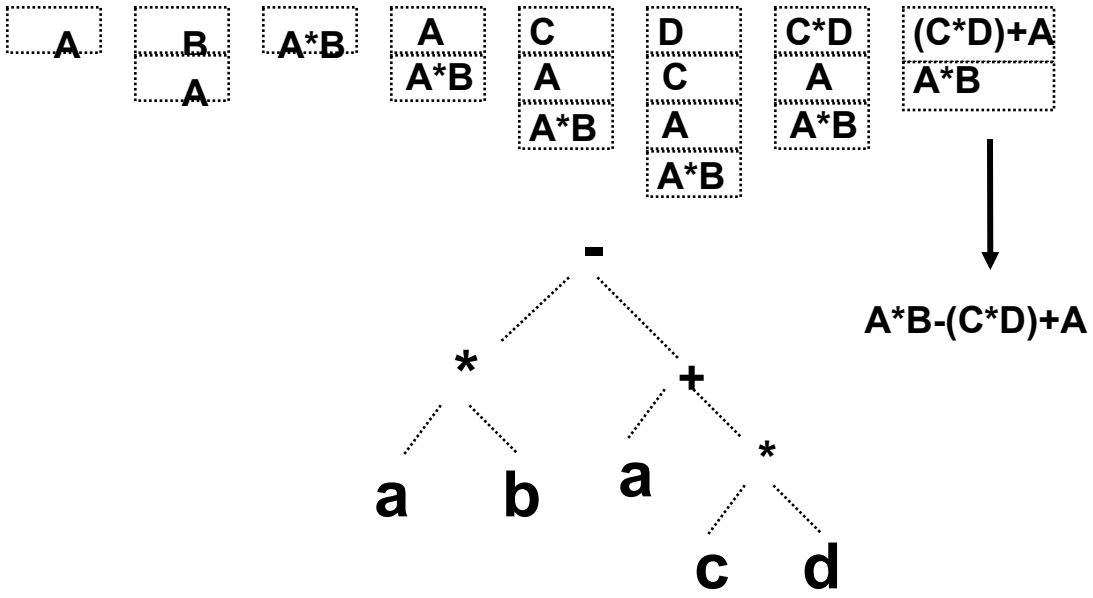
- Exemplo: a*b - (a+c*d)

push a
push b
*

push a
push c
Push d
*

+

-



Máquinas de Pilha

- Não utilizam registradores
- Não possuem operandos específicos para a ALU
- Vantagens:
  - Instruções compactas, endereço dos operandos implícito na pilha
  - Compilador é fácil de escrever
- Exemplos: Burroughs B5500/6500, HP 3000/70, Intel x87 co-processador de ponto flutuante, Java Virtual Machine

Máquinas de Pilha Desvantagens

- Desempenho é função da existência de diversos registradores rápidos e não da maneira de como eles são organizados
- Dados não aparecem no topo da pilha quando precisamos deles
  - Constantes, operandos repetidos
- Densidade de código é igual à densidade de máquinas com GPR
  - Registradores possuem campo de endereço com alta densidade
  - Mantenha resultados intermediários nos registradores para reusá-los
- Compilador é fácil de escrever, difícil de otimizar

Máquinas com Registradores de Uso Geral (GPR)

- Registradores considerados memória rápida bem próxima a ALU
- Opções:
  - 2 vs. 3 operandos:
    - 2 operandos R = R op y
    - 3 operandos R = x op y
  - Operações de ALU podem ou não acessar memória
    - RISC (L/S ou R/R) : não
    - CISC (R/M) : sim

Máquinas com Registradores de Uso Geral (GPR)

- Arquitetura dominante atualmente

- Vantagens:
  - Permite acesso mais rápido a valores temporários
  - Permite técnicas mais agressivas de compilação
  - Reduz tráfego à memória

- Desvantagens:
  - Instruções mais **longas**
  - Troca de contexto mais **lenta**

Máquinas GPR

- L/S ou R/R (0,3)
  - Tipicamente 3 operandos
  - Exemplos: CDC 6600, Cray 1, RISCs
  - Vantagens:
    - Codificação simples das instruções
    - Fácil de gerar código e *pipeline*
    - Ciclos/instrução são +/- iguais
  - Desvantagens:
    - IC mais alto
    - Desperdício de códigos

Máquinas GPR

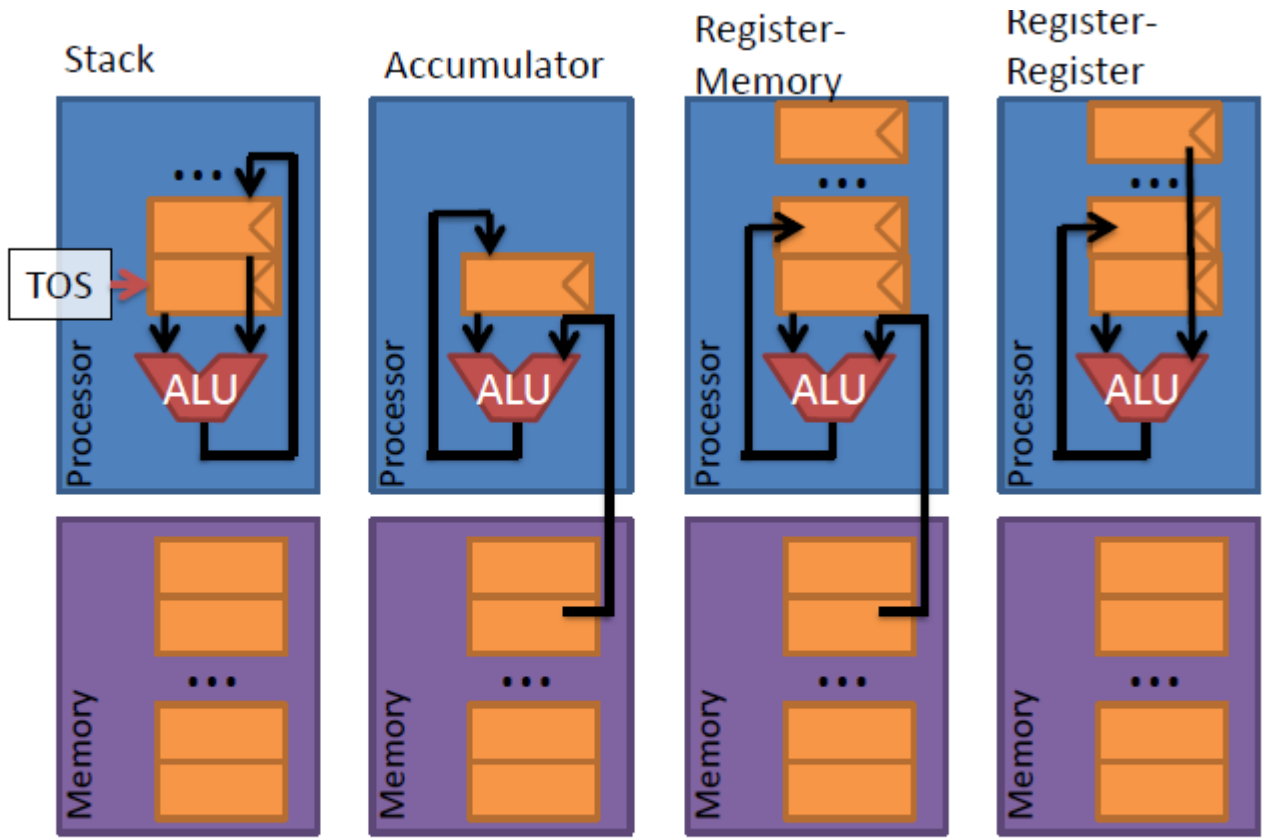
- R/M (1,2)
  - 1 operando de memória, 2 operandos
  - Exemplos: 80x86
  - Vantagens:
    - Acesso a dados sem carregar registradores
    - Boa densidade de código
  - Desvantagens:
    - Operandos não são equivalentes
    - Codificação de operandos limita número de registradores
    - Variância de CPI é mais alta
    - Mais difícil de *pipeline*

Máquinas GPR

- R/M (3,3)
  - 3 operandos de memória, 3 operandos
  - Exemplos: VAX
  - Vantagens:
    - Compactação e não utiliza registradores como temporários
  - Desvantagens:
    - Variação alta no tamanho da instrução
    - Gargalo na memória (1 instr, 4 referências)
    - Variância de CPI é a mais alta de todas
    - Muito mais difícil de *pipeline*

Classes de ISAs - Resumo

C = A + B

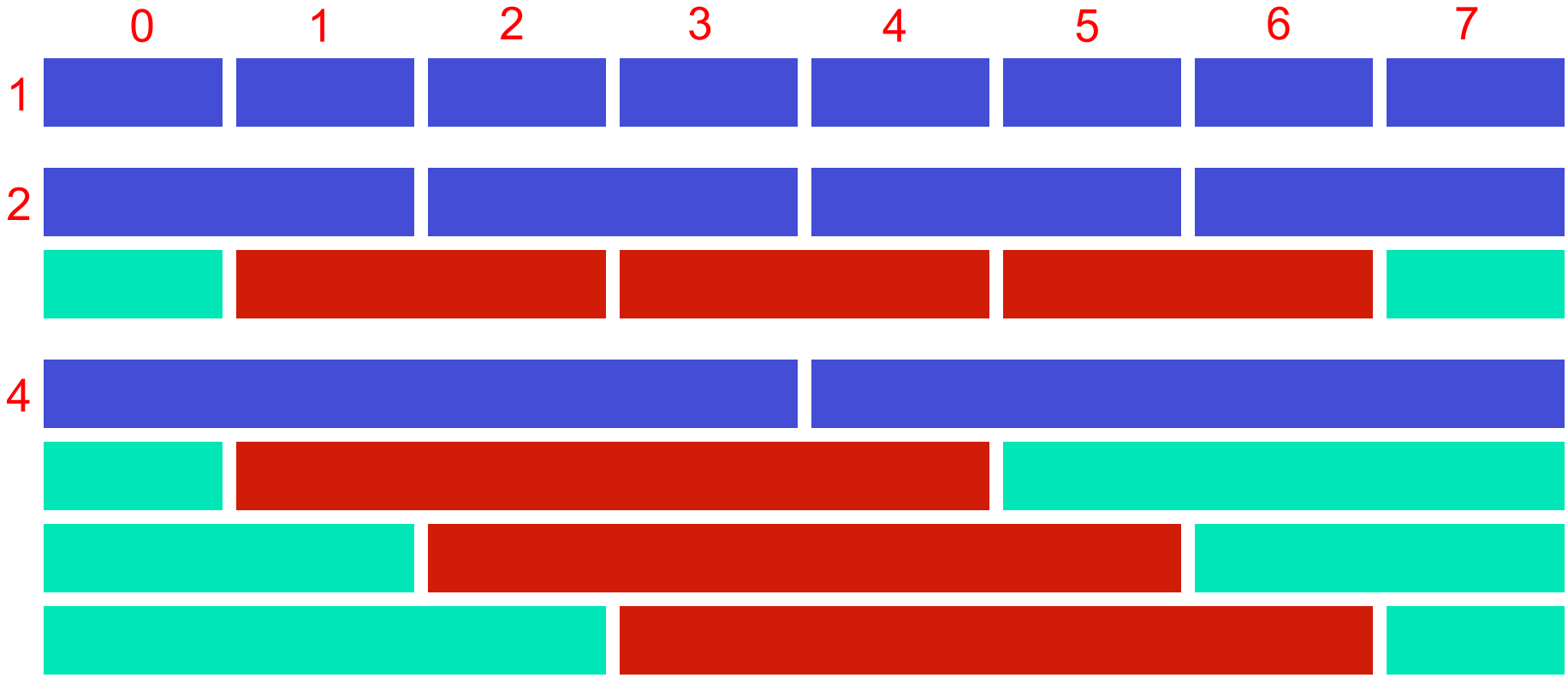


Endereçamento

Big Endian vs. Little Endian

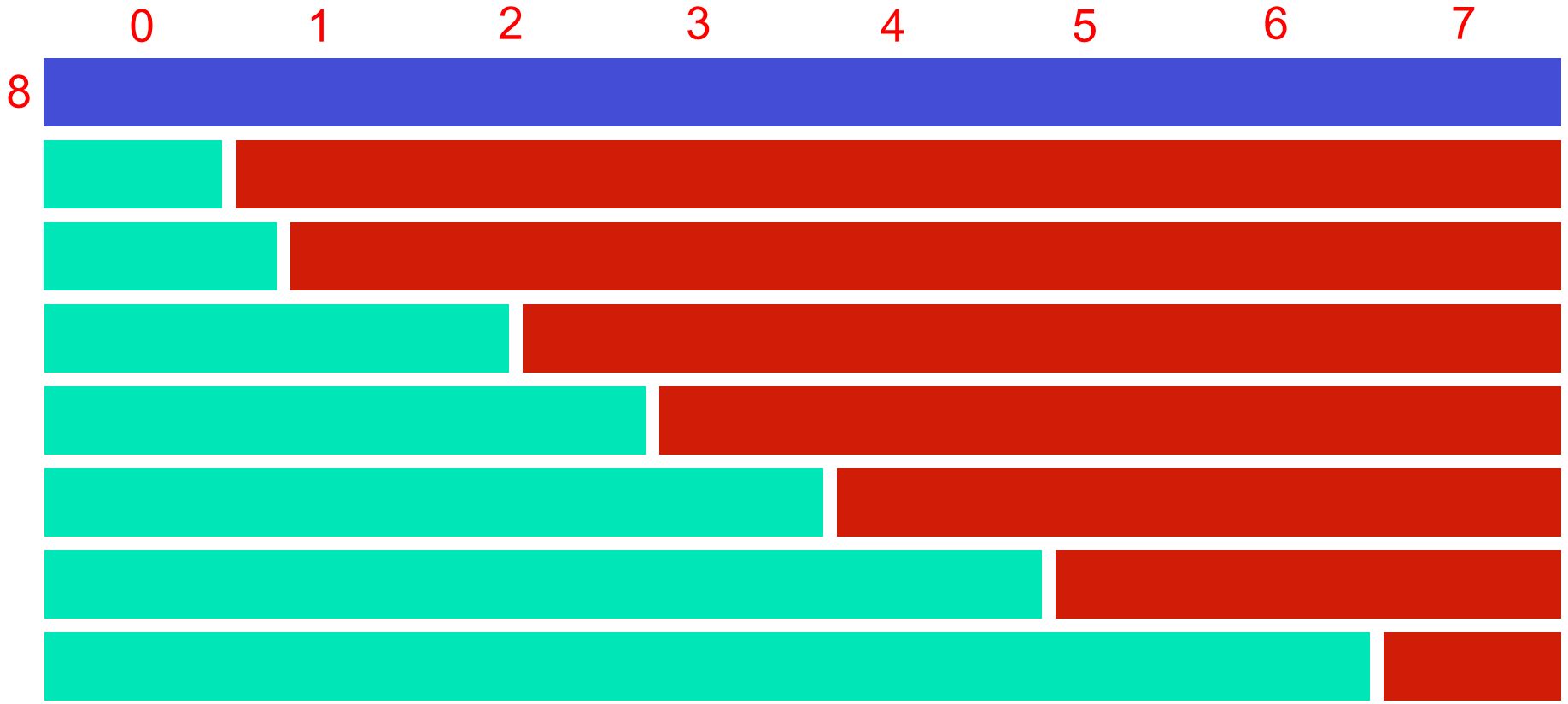
Endereçamento

Alinhamento



Endereçamento

Alinhamento



Modos de Endereçamento

Addressing Mode	Instruction	Function
Register	Add R4, R3, R2	Regs[R4] <- Regs[R3] + Regs[R2] **
Immediate	Add R4, R3, #5	Regs[R4] <- Regs[R3] + 5 **
Displacement	Add R4, R3, 100(R1)	Regs[R4] <- Regs[R3] + Mem[100 + Regs[R1]]
Register Indirect	Add R4, R3, (R1)	Regs[R4] <- Regs[R3] + Mem[Regs[R1]]
Absolute	Add R4, R3, (0x475)	Regs[R4] <- Regs[R3] + Mem[0x475]
Memory Indirect	Add R4, R3, @(R1)	Regs[R4] <- Regs[R3] + Mem[Mem[R1]]
PC relative	Add R4, R3, 100(PC)	Regs[R4] <- Regs[R3] + Mem[100 + PC]
Scaled	Add R4, R3, 100(R1)[R5]	Regs[R4] <- Regs[R3] + Mem[100 + Regs[R1] + Regs[R5] * 4]

** Não acessa a memória!

ISA do Mundo Real

Arch	Type	# Oper	# Mem	Data Size	# Regs	Addr Size	Use
Alpha	Reg-Reg	3	0	64-bit	32	64-bit	Workstation
ARM	Reg-Reg	3	0	32/64-bit	16	32/64-bit	Cell Phones, Embedded
MIPS	Reg-Reg	3	0	32/64-bit	32	32/64-bit	Workstation, Embedded
SPARC	Reg-Reg	3	0	32/64-bit	24-32	32/64-bit	Workstation
TI C6000	Reg-Reg	3	0	32-bit	32	32-bit	DSP
IBM 360	Reg-Mem	2	1	32-bit	16	24/31/64	Mainframe
x86	Reg-Mem	2	1	8/16/32/ 64-bit	4/8/24	16/32/64	Personal Computers
VAX	Mem-Mem	3	3	32-bit	16	32-bit	Minicomputer
Mot. 6800	Accum.	1	1/2	8-bit	0	16-bit	Microcontroler