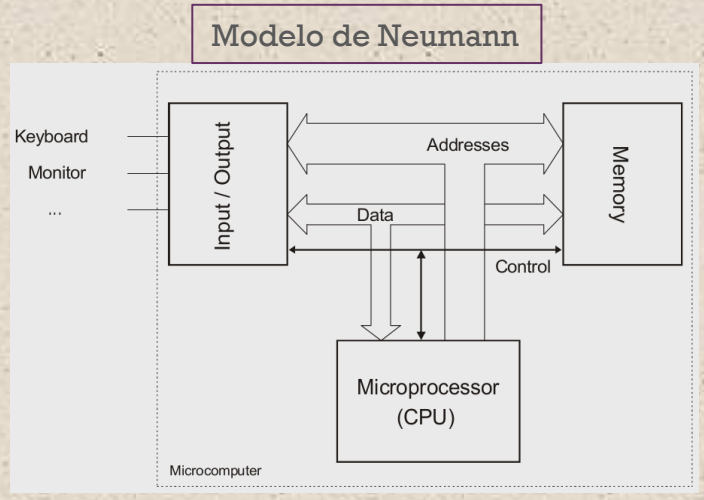
Capítulo 12
Conjuntos de instruções

William Stallings
Arquitetura e Organização de Computadores
10ª Edição

Tradução e Adaptação:
Profa. Thaína A. A. Tosta

Revisão

- Organização
  - Componentes do computador: (1) processador, (2) memória, (3) caminhos de dados, (4) dispositivos de entrada e (5) dispositivos de saída;
- Histórico do computador
  - Rápido crescimento
  - Chegamos ao limite de frequência...
    - Quente!!!
- Lei de Moore
  - Continuamos dobrando número de transistores a cada 18 meses.



CPU

A partir da aula de hoje, examinaremos o funcionamento e implementação da CPU:

- **Arquitetura**;
- Organização.

Questão de arquitetura: se um computador terá uma instrução de multiplicação.

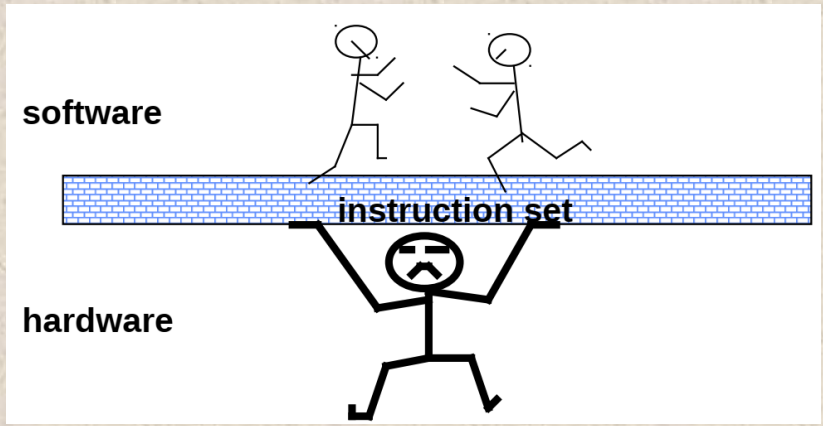
Questão de organização: se essa instrução será implementada por uma unidade de multiplicação especial ou por um mecanismo que faça repetidas adições.

O que é um conjunto de instruções?

- Conjunto de instruções compreendido pela CPU, assim como instruções em linguagens de programação;
- As instruções são conhecidas como código de máquina, que é o programa em binário;
- Esse código é comumente representado por códigos em assembly, para facilitar nossa leitura.

Conjunto de instruções

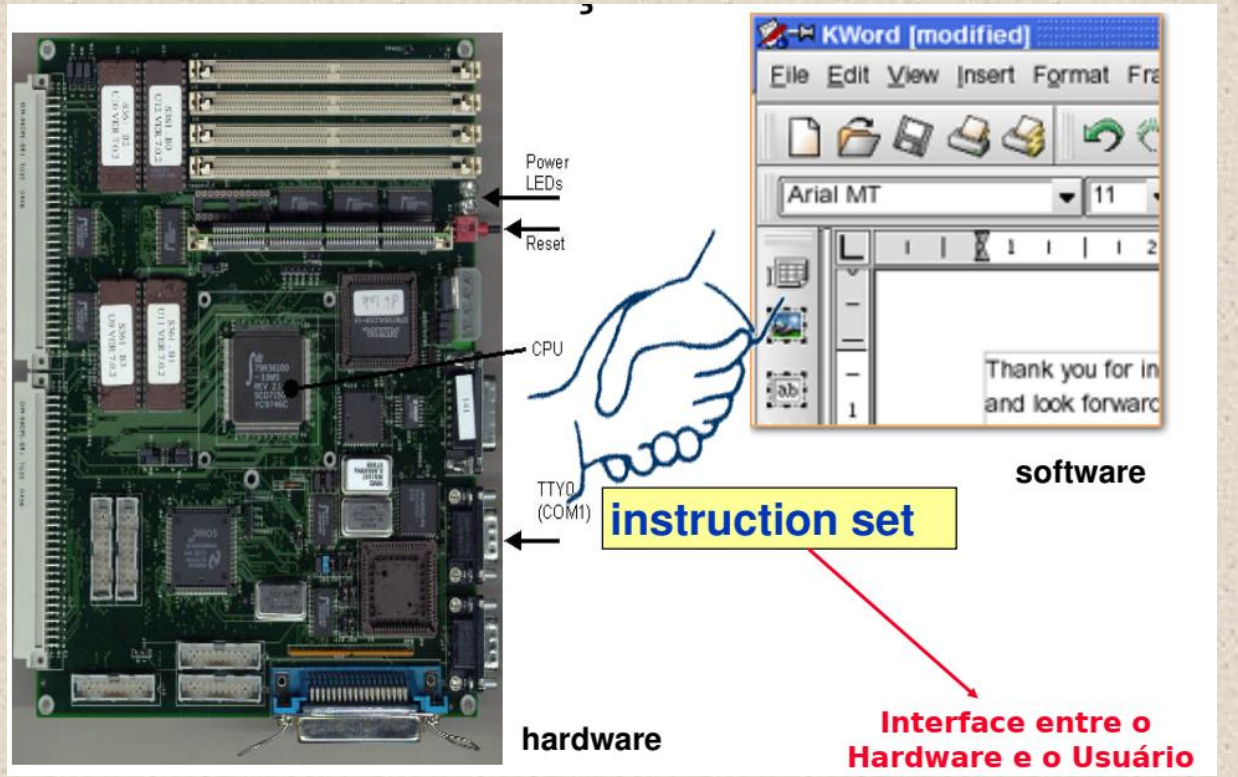
O ISA (do inglês, *Instruction Set Architecture*) é a porção da máquina visível ao programador (nível de montagem) ou aos projetistas de compiladores;



1. Quais as vantagens e desvantagens das diversas alternativas de ISA?
2. Como as linguagens e compiladores afetam (ou são afetados pelo) o ISA?
3. Arquitetura MIPS como exemplo de arquitetura RISC.

Conjunto de instruções

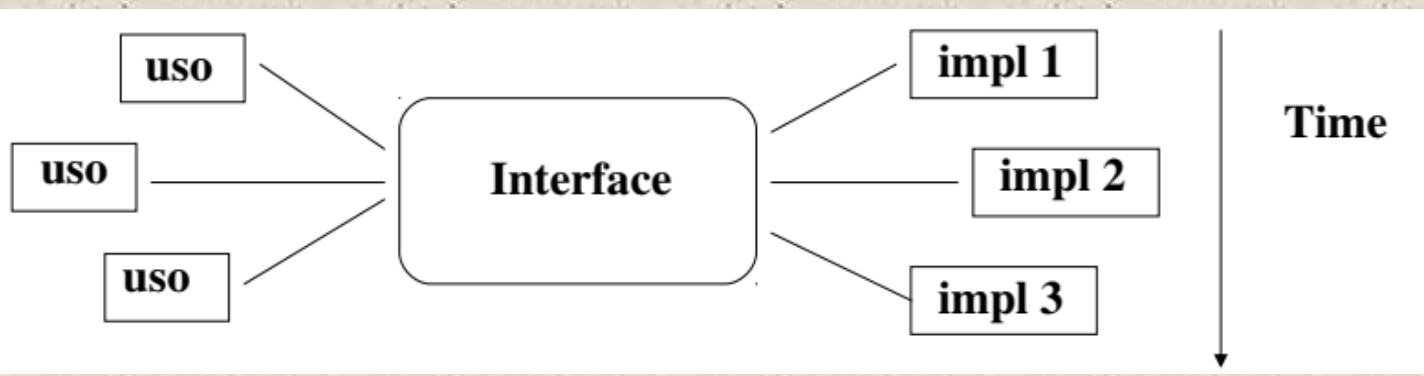
Um limite onde quem projeta um computador e quem o programa podem ver a mesma máquina é o conjunto de instruções de máquina.



Introdução - Interface

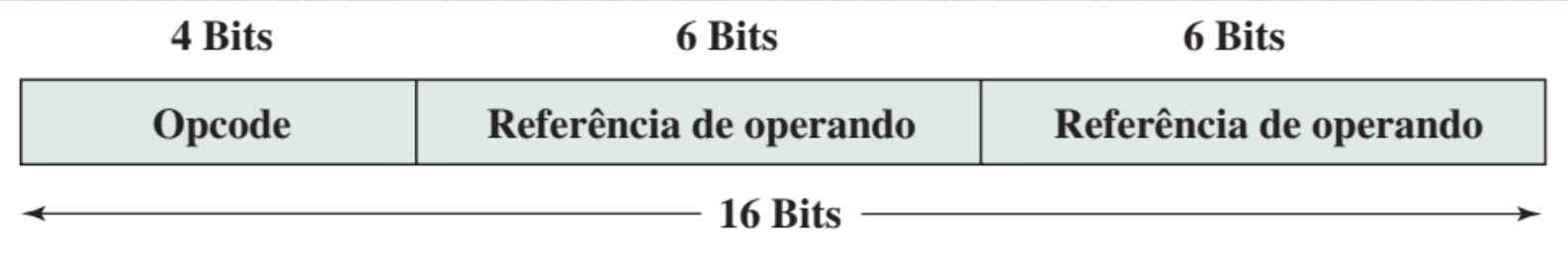
Uma boa interface:

- Sobrevive a várias implementações (portabilidade, capacidade, escalabilidade, …);
- Usada de diferentes formas por diferentes programas (generalidade);
- Provê funcionalidades para os níveis superiores;
- Permite uma implementação eficiente no nível inferior (hw).



Representação da instrução

- Dentro do computador, cada instrução é representada por uma sequência de bits;
- Na maioria dos conjuntos de instruções, mais de um formato é utilizado;
- Durante a execução da instrução, uma instrução é lida para um registrador de instrução (IR) no processador;



- Instrução em bits (para o hardware);
- Opcode: código identificador da instrução;
- Referência de operando: identifica qual é ou como/onde encontrar o operando.

Representação da instrução

- Os *opcodes* são representados por abreviações, chamadas *mnemônicos*;
- Exemplos incluem:
  - ADD Adição
  - SUB Subtração
  - MULT Multiplicação
  - DIV Divisão
  - LOAD Carrega dados da memória
  - STORE Armazena dados na memória
- Operandos também são representados simbolicamente: ADD R, Y, onde R é o conteúdo do registrador e Y endereço de memória;
- Cada *opcode* tem uma representação binária fixa
  - Quem programa é quem especifica a localização de cada operando simbólico.

Representação da instrução

- Em código de máquina, cada instrução tem um padrão único de bits;
  - 0010010001010101O
- Para entendimento humano (para quem programa), uma representação simbólica é usada pelo código assembly;
  - ADD, SUB, LOAD
- Operandos podem ser representados dessa forma:
  - ADD A,B.

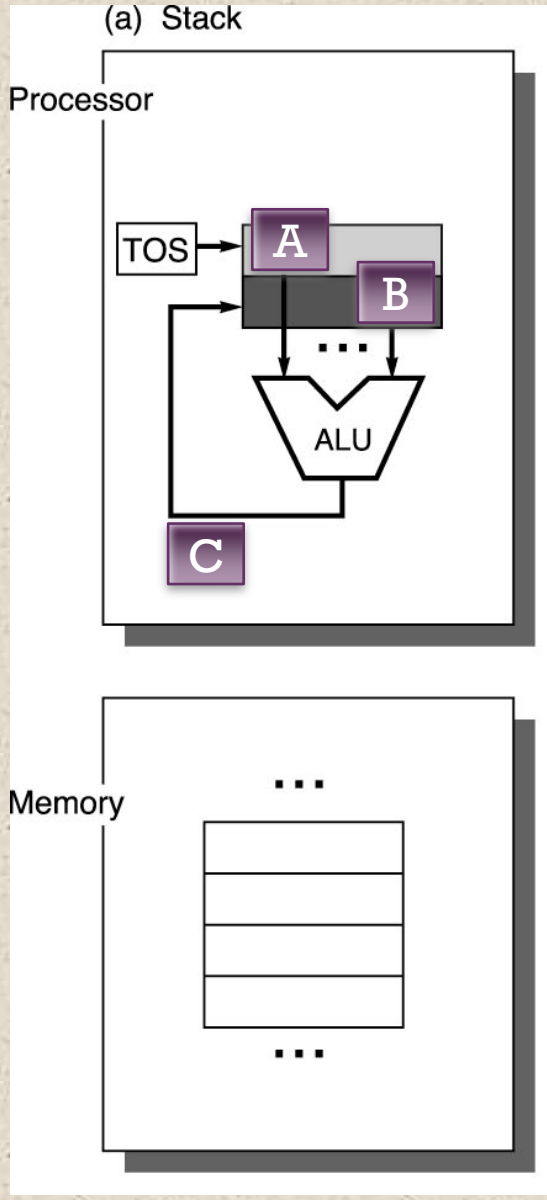
Classes básicas de ISA

Pilha:

- A memória é uma pilha;
- Calculadoras polonesas

C = A+B

Push A
Push B
Add
Pop C



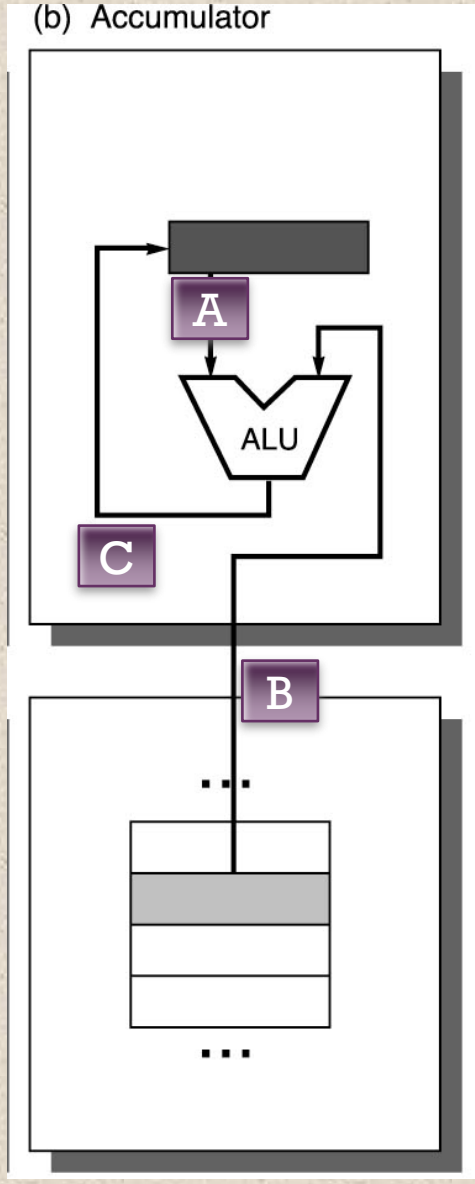
Classes básicas de ISA

Acumulador:

- Um registrador para todas as operações;
- Dados saem desse registrador e ele também recebe o resultado.

C = A+B

Load A
Add B
Store C



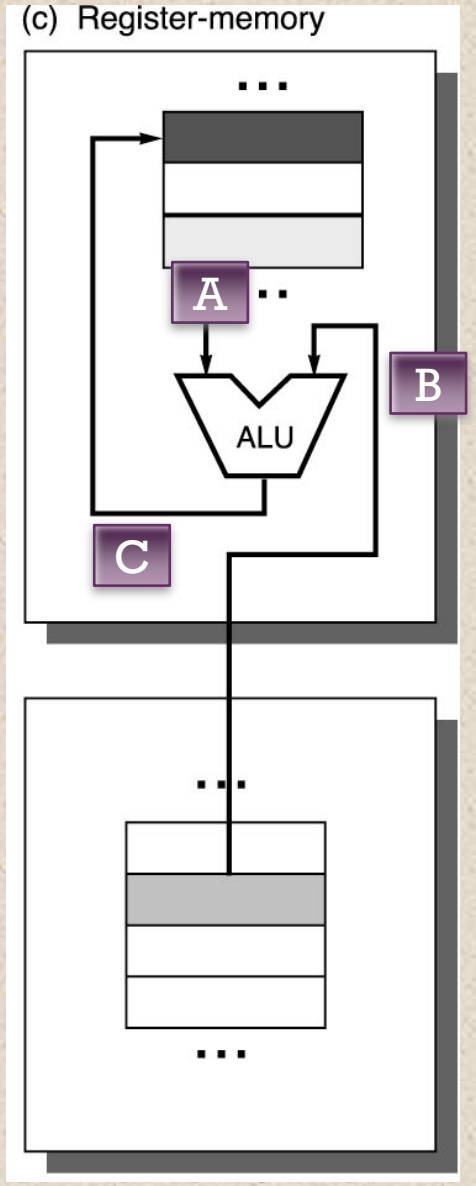
Classes básicas de ISA

Registrador (registrador-memória)

- As operações usam um registrador e um endereço de memória;
- Conjunto de registradores no processador.

C = A+B

Load R1, A
Add R1, B
Store C, R1

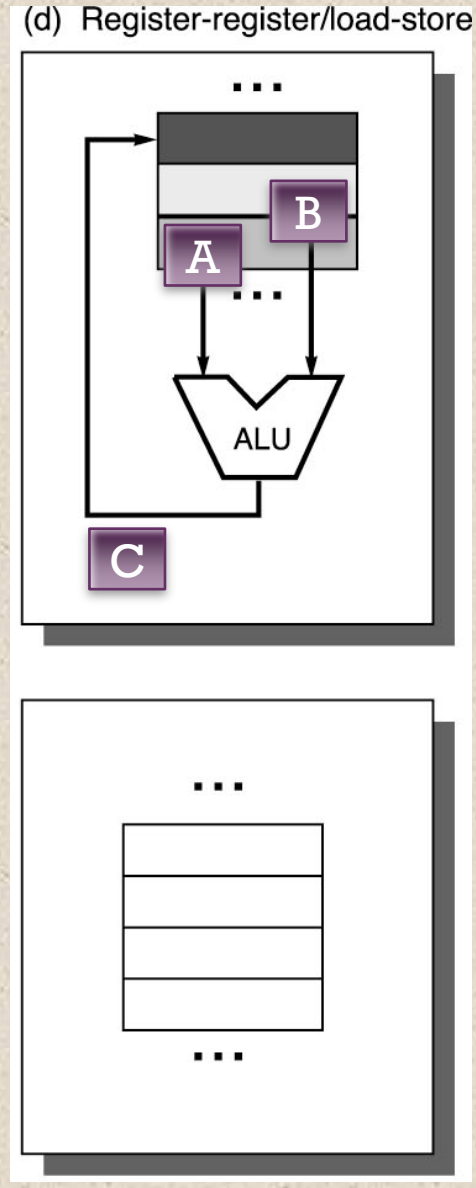


Classes básicas de ISA

Registrador (*load-store*)

- As operações não usam endereços de memória;
- ↑ instruções.

C = A+B
Load R1, A Load R2, B Add R3, R1, R2 Store C, R3



Classes básicas de ISA

Código nas diferentes classes de endereçamento para:

C = A + B

Stack	Accumulator	Register (Register-memory)	Register (load-store)
Push A	Load A	Load R1, A	Load R1, A
Push B	Add B	Add R1, B	Load R2, B
Add	Store C	Store C, R1	Add R3, R1, R2
Pop C			Store C, R3

Variações pela qtde de operandos

Análise das classes básicas de ISA

Pilha (*stack*)

- Vantagens:
  - Forma simples para avaliação de expressões (notação polonesa);
  - Instruções curtas, que podem levar a uma boa densidade de código com economia de memória.
- Desvantagens:
  - A pilha não pode ser acessada randomicamente;
  - Esta limitação torna mais complicada a geração de código eficiente;
  - A implementação eficiente também é mais difícil, porque a pilha torna-se um gargalo.

Análise das classes básicas de ISA

Acumulador

- Vantagens:
  - Minimiza o número de estados internos da máquina, com organização simples da máquina;
  - Instruções curtas.
- Desvantagens:
  - Uma vez que o acumulador é um temporário, o tráfego na memória é alto.

Análise das classes básicas de ISA

Registrador

- Vantagens:
  - Formato mais geral para geração de código.
- Desvantagens:
  - Todos os operandos devem ser nomeados, instruções mais longas.

Análise das classes básicas de ISA

Enquanto as máquinas mais antigas usavam o estilo "*stack*" ou "*accumulator*", arquiteturas modernas (projetadas nos últimos 10-20 anos) usam "*general purpose register*"

- Registradores são mais rápidos;
- O uso de registradores é mais fácil para os compiladores;
- Registradores podem ser utilizados mais efetivamente como forma de armazenamento.

Histórico de máquinas *vs* arquiteturas

Machine	Number of general-purpose registers	Architectural style	Year
EDSAC	1	Accumulator	1949
IBM 701	1	Accumulator	1953
CDC 6600	8	Load-store	1963
IBM 360	16	Register-memory	1964
DEC PDP-8	1	Accumulator	1965
DEC PDP-11	8	Register-memory	1970
Intel 8008	1	Accumulator	1972
Motorola 6800	2	Accumulator	1974
DEC VAX	16	Register-memory, memory-memory	1977
Intel 8086	1	Extended accumulator	1978
Motorola 68000	16	Register-memory	1980
Intel 80386	8	Register-memory	1985
MIPS	32	Load-store	1985
HP PA-RISC	32	Load-store	1986
SPARC	32	Load-store	1987
PowerPC	32	Load-store	1992
DEC Alpha	32	Load-store	1992

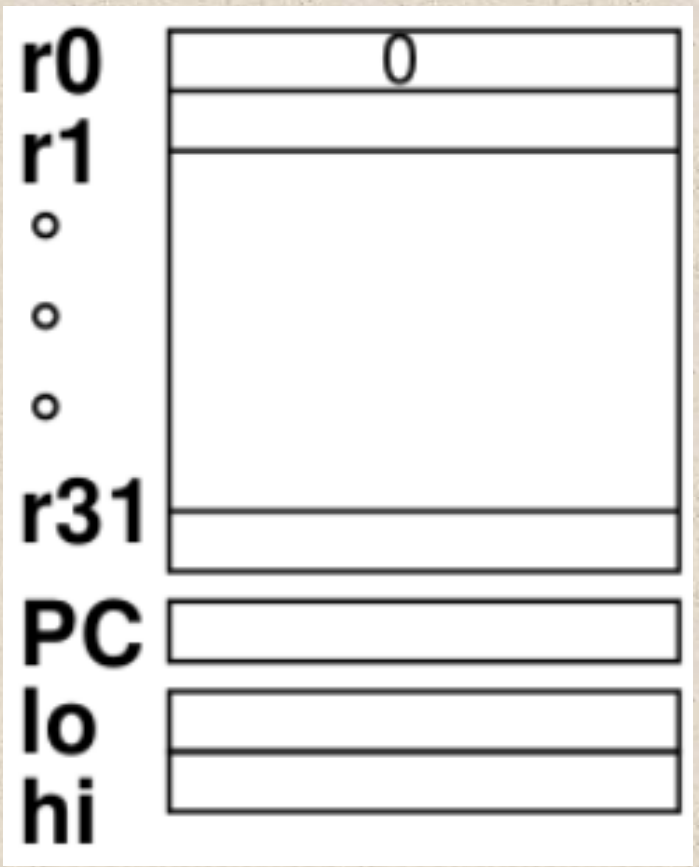
Registradores no Intel 80x86

GPR0	EAX	Accumulator
GPR1	ECX	Count register, string, loop
GPR2	EDX	Data Register; multiply, divide
GPR3	EBX	Base Address Register
GPR4	ESP	Stack Pointer
GPR5	EBP	Base Pointer - for base of stack seg.
GPR6	ESI	Index Register
GPR7	EDI	Index Register
	CS	Code Segment Pointer
	SS	Stack Segment Pointer
	DS	Data Segment Pointer
	ES	Extra Data Segment Pointer
	FS	Data Seg. 2
	GS	Data Seg. 3
PC	EIP	Instruction Counter
	Eflags	Condition Codes

Registradores no MIPS (~ARM)

Armazenamento programável

- ISA *load-store*;
- 2^{32} x bytes de memória;
- 31 x 32-bits GPRs (R0 = 0);
- HI, LO, PC.



Decisões para desenvolver arquiteturas

- Repertório de operações:
  - Quantas operações? (qtde de bits *vs* qtde de instruções);
  - O que as instruções podem fazer?
  - Quão complexas serão as instruções?
- Tipos de dados (32 bits e 64 bits);
- Formatos de instruções:
  - Tamanho do campo de *opcode*;
  - Número de endereços (Dois ou três endereços? Resultados implícitos?).

Decisões para desenvolver arquiteturas

- Registradores
  - Número de registradores na CPU disponíveis;
  - Quais operações podem ser executadas em quais registradores (registradores de propósito geral ou específicos a operações)?
- Modos de endereçamento;
- RISC *vs* CISC.

Tipos de operandos

- Endereços
- Números
  - Inteiro, ponto flutuante
- Caracteres
  - ASCII
- Dados lógicos
  - Bits ou flags
- Há diferença entre números e caracteres? Operações possíveis e memória usada.

Tipos específicos de dados

- *General* - conteúdo binário arbitrário;
- *Integer* - valor binário único;
- *Ordinal* - inteiro sem sinal;
- *Unpacked BCD* - 1 dígito por byte;
- *Packed BCD* - 2 dígitos BCD por byte;
- *Near Pointer* - 32 bits de offset no segmento;
- *Bit field*;
- *Byte String*;
- *Floating Point*.

Tipos de operações

- Transferência de dados;
- Operações aritméticas;
- Operações lógicas;
- Conversões;
- E/S (I/O);
- Controle do sistema;
- Transferência de controle.

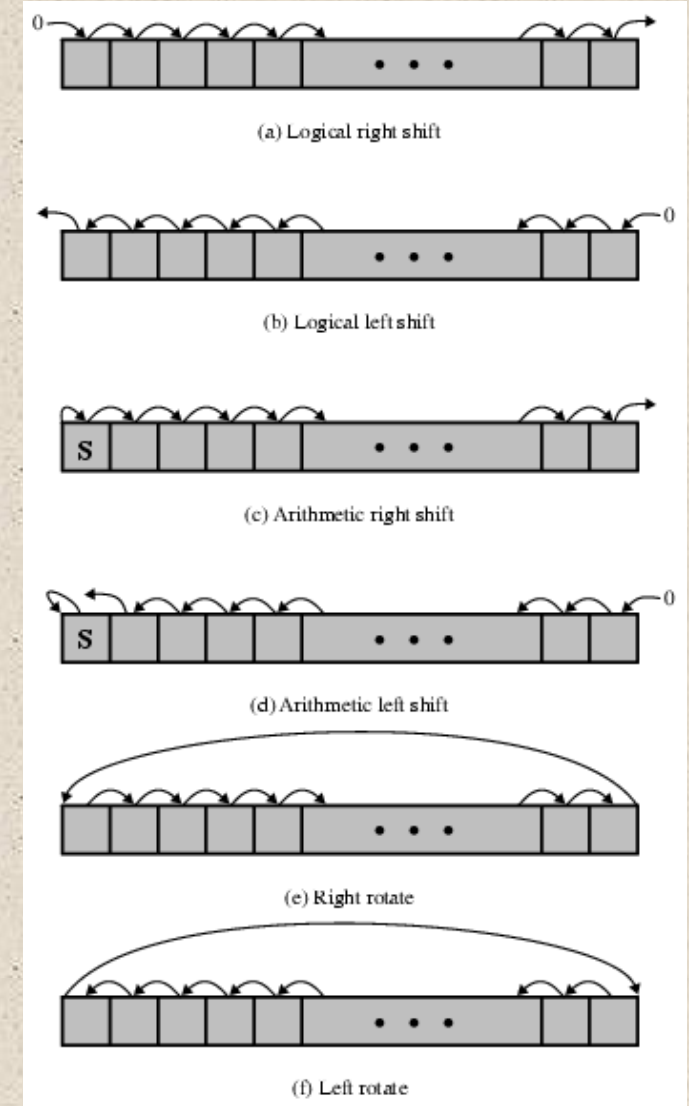
Tipos de operações - Transferência de dados

- Deve especificar:
  - Fonte;
  - Destino;
  - Quantidade de dados;
- Pode ser diferentes instruções para diferentes transferências
  - IBM 370;
- Ou uma instrução com endereços diferentes
  - VAX.

Tipos de operações – Operações aritméticas

- Soma, subtração, multiplicação e divisão;
- Mesma instrução para inteiros com e sem sinal?
- Mesma instrução para pontos flutuantes?
- Pode incluir:
  - Incremento (a++);
  - Decremento (a--);
  - Negativo (-a).

Tipos de operações – Operações lógicas (*shift* e rotação)

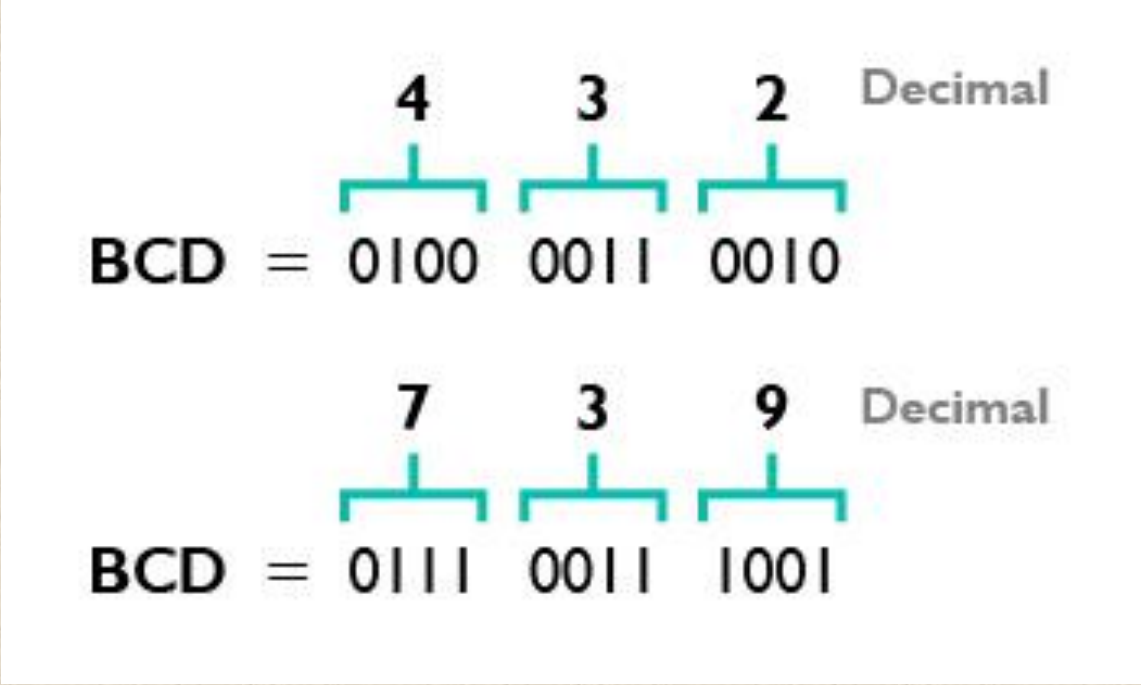


Tipos de operações – Operações lógicas

- Operações bit a bit;
- AND, OR, NOT.

Tipos de operações – Conversões

Binário para BCD (Em hardware? Em software?)



https://materialpublic.imd.ufrn.br/curso/disciplina/1/17/1/13

Tipos de operações – E/S

- Podem ser instruções específicas (interrupções);
- Pode ser feito por instruções de movimento de dados (memória mapeada);
- Pode ser feito por um controlador separado (DMA, do inglês *Direct Access Memory*).

Tipos de operações – Controle do sistema

- Instruções privilegiadas;
- CPU precisa estar em um estado específico
  - Modo kernel;
- Para uso do sistema operacional.

Tipos de operações – Transferência de controle

- Desvio (*branch*)
  - Desvio para x se o resultado for zero;
- *Skip*
  - Incremente e pule se for zero;
- Chamadas de subrotinas
  - Chamadas de interrupção.

Operações em um ISA

Rank	80x86 instruction	Integer average (% total executed)
1	load	22%
2	conditional branch	20%
3	compare	16%
4	store	12%
5	add	8%
6	and	6%
7	sub	5%
8	move register-register	4%
9	call	1%
10	return	1%
Total		96%

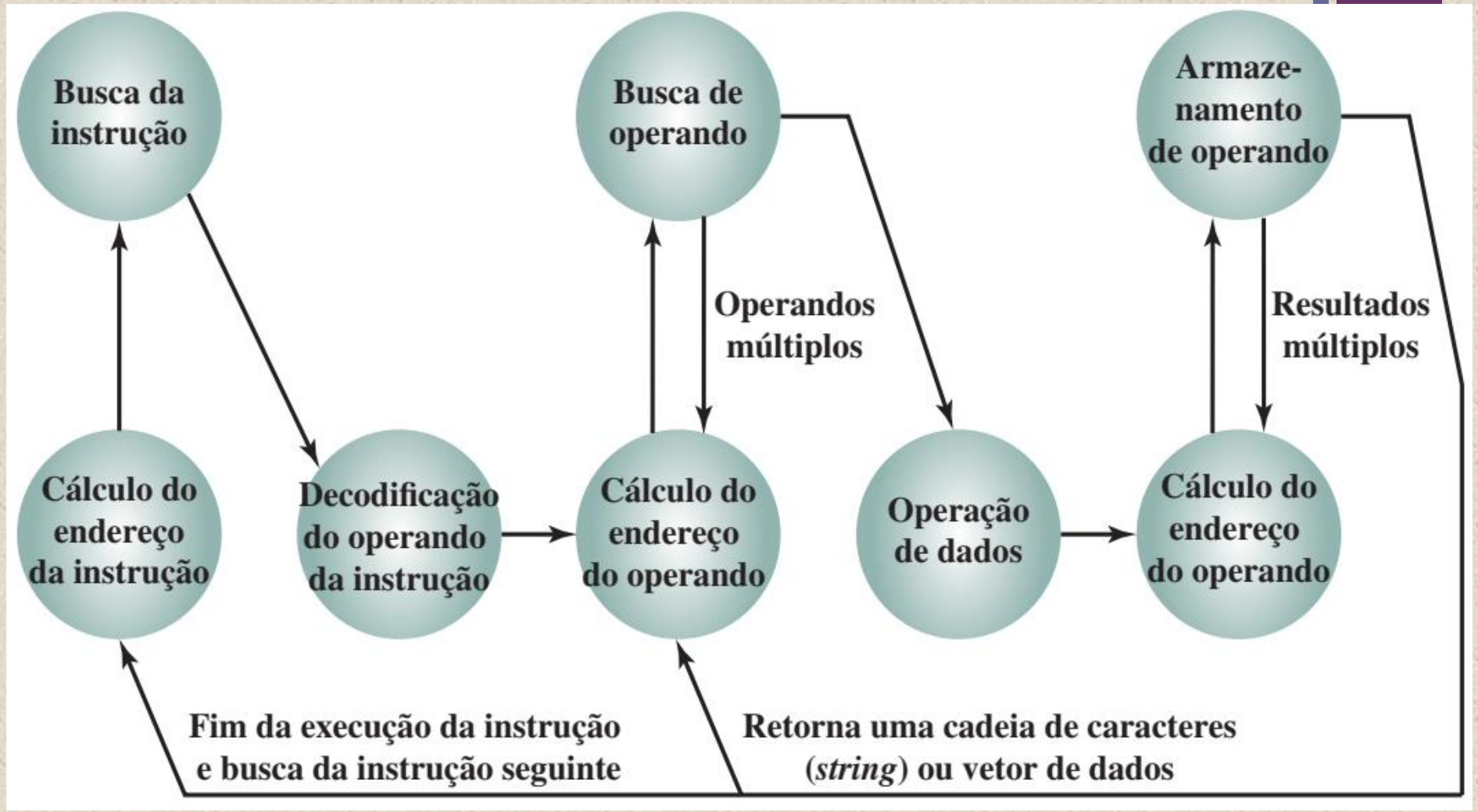
Vale a pena usar um grande conjunto de instruções sabendo da % de seu uso?

Evolução dos ISAs

- As maiores vantagens em uma arquitetura, em geral, estão associadas com as mudanças do ISA;
  - Pilha *vs* registradores de propósito geral
- Decisões de projeto que devem ser levadas em consideração:
  - tecnologia (quantas portas lógicas? quantos transistores?);
  - organização (o que será implementado?);
  - linguagens de programação;
  - tecnologia em compiladores (compiladores serão beneficiados na tradução de códigos?);
  - sistemas operacionais.

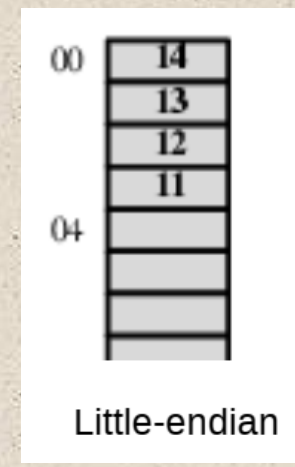
Diagrama de execução de instruções

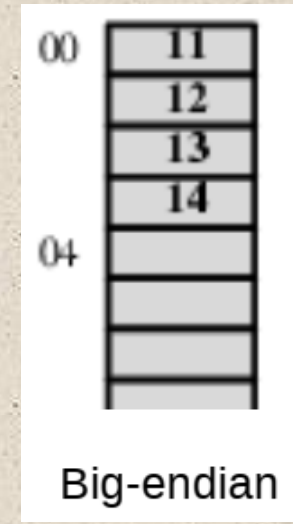
Troca entre o processador e a memória ou um módulo de E/S



Modos de endereçamento (interpretando endereços de memória)

- Qual objeto é acessado em função do endereço e qual o seu tamanho?
  - Objetos endereçados a byte – um endereço refere-se ao número de bytes contados do início da memória
- Considerando uma palavra de 4 bytes
  - Little-endian – o byte cujo endereço é xx00 é o byte menos significativo da palavra.
  - Big-endian – o byte cujo endereço é xx00 é o mais significativo da palavra.
- Alinhamento – o dado deve ser alinhado em fronteiras iguais a seu tamanho.
  - endereço % sizeof (tipo de dado) == 0;
  - bytes podem ser alinhados em qualquer endereço;
  - inteiros de 4 bytes são alinhados em endereços múltiplos de 4.





Existe padrão? Qual padrão?

- Pentium (80x86), VAX são little-endian;
- IBM 370, Motorola 680x0 (Mac), e a maioria dos RISC são big-endian;
- Internet é big-endian;
- Por isso são necessários programas para fazer essa conversão.

Capítulo 12

Conjuntos de instruções

Agradeço a Prof. Dr. Fábio A. M. Cappabianco, Dave Patterson e Paulo Centoducatte pelos materiais disponibilizados.

William Stallings
Arquitetura e Organização de Computadores
10ª Edição

Tradução e Adaptação:
Profa. Thaína A. A. Tosta
tosta.thaina@unifesp.br