



Universidade Presbiteriana  
**Mackenzie**

Faculdade de Computação e Informática

Disciplina: Organização de Computadores

# Conjunto de Instruções

(Instruction Set Architecture - ISA)

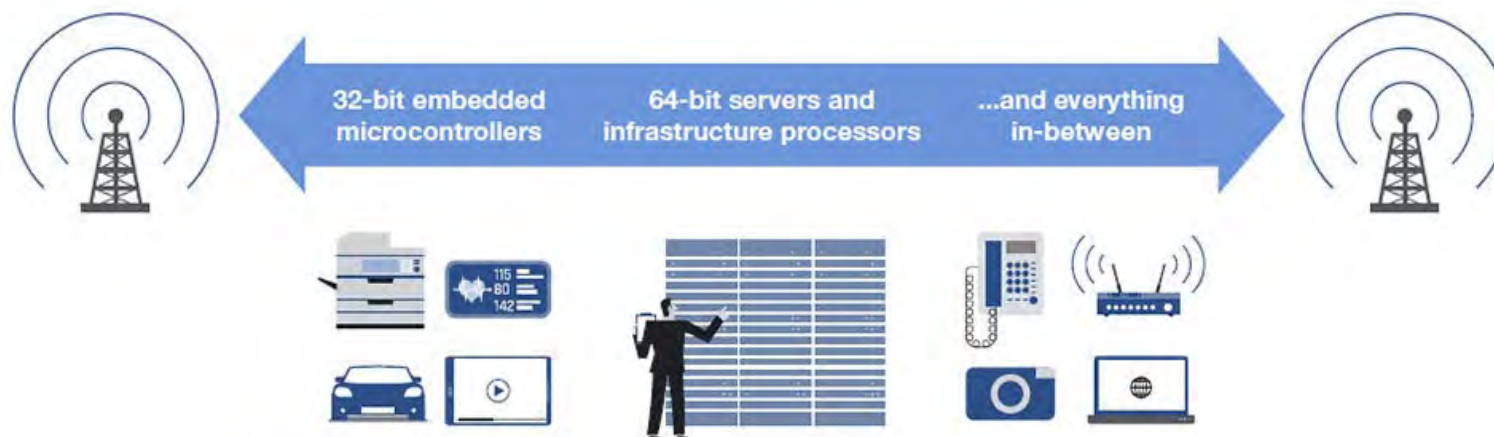
## MIPS

Dr. Jean M. Laine e Dr. Wilian F. Costa

# MIPS



- A empresa *MIPS Computer Systems* foi fundada em 1984 por um grupo de pesquisadores da Universidade de Stanford e o foco era os microprocessadores com Arquitetura RISC<sup>1</sup>
- Hoje ela é chamada de MIPS Technologies, Inc  
[www.mips.com](http://www.mips.com)

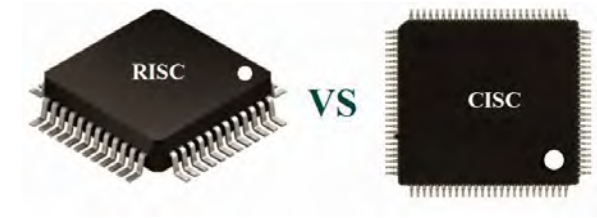



1 – Arquitetura com um conjunto reduzido de instruções

# Instruções

- Os computadores, notebooks e celulares atuais são dispositivos de uso geral. Eles precisam ter capacidade de processar diversos tipos de instruções e dados diferentes, que vão desde um simples editor de texto até um vídeo 3D.
- É diferente de um controle remoto de TV, de uma geladeira, de um Microondas, que são projetados para um fim muito específico.
- Algumas instruções que o **microprocessador** de uso geral pode executar são:
  - operações aritméticas e lógicas;
  - operações relacionais;
  - operações de ponto flutuante;
  - transferência de dados;
  - desvios condicionais;
  - desvios incondicionais;
  - operações de controle.

# Tamanho da Instrução



- É o tamanho, em **bits**, de uma instrução.
- Os processadores x86 (**CISC**) costumam **não** ter instruções de tamanho fixo.
  - uma operação aritmética pode ter um tamanho diferente de uma operação de transferência, por exemplo. Isso não é muito interessante!
  - uma instrução gasta mais que um ciclo de clock! 
- Assim, em um projeto de ISA, o tamanho da instrução, em bits, pode ser **FIXO** ou **VARIÁVEL**.
- A arquitetura MIPS atual tem tamanho fixo de 64 bits e é chamada de MIPS 64 Bits. É uma arquitetura **RISC**. Podemos trabalhar com MIPS 32 bits também.

# Formato da Instrução

- Em geral, uma instrução tem:
  - um campo reservado para o **código de operação**, chamado de **OPCODE**
  - campos reservados para os **operandos** envolvidos na operação
- O OPCODE indica qual é a operação que deverá ser executada, exemplo:  
**ADD \$s1, \$s2, \$3.**
- ADD é o *mnemônico* que identifica o código da operação, neste exemplo, uma adição com dois operandos.
- Os operandos estão representados pelos registradores de uso geral \$s2, \$s3.
- O resultado da operação de soma será armazenado no registrador \$s1 (destino).

# Registradores de uso geral no MIPS 32 bits

- Sempre que escrevemos instruções em linguagem de montagem manipulamos registradores para armazenar variáveis, endereços, resultados de operações, contadores, etc
- No MIPS, em geral, as variáveis que armazenam valores usam os registradores de nome **t** e as variáveis que contêm os operandos da instrução usam os registradores de nome **s**

## Exemplo em C:

`a = b + c;`

## Exemplo em MIPS:

`ADD $t0, $s0, $s1`

- **ADD** é o mnemônico para ADIÇÃO, **\$s0** é o valor que está armazenado em **b**, **\$s1** o valor da variável **c** e **\$t0** é a variável **a**.

Nome do Registrador	Número	Binário	Uso
\$zero	0	000 000	constante zero
\$at	1	000 001	reservado para o montador
\$v0	2	000 010	avaliação de expressão e resultados de uma função
\$v1	3	000 011	avaliação de expressão e resultados de uma função
\$a0	4	000 100	argumento 1 (passam argumentos para as rotinas)
\$a1	5	000 101	argumento 2
\$a2	6	000 110	argumento 3
\$a3	7	000 111	argumento 4
\$t0	8	001 000	temporário (valores que não precisam ser preservados entre chamadas)
\$t1	9	001 001	temporário
\$t2	10	001 010	temporário
\$t3	11	001 011	temporário
\$t4	12	001 100	temporário
\$t5	13	001 101	temporário
\$t6	14	001 110	temporário
\$t7	15	001 111	temporário

Registradores de uso  
geral do MIPS 32 bits

\$s0	16	010 000	temporário salvo (valores de longa duração e devem ser preservados entre chamadas)
\$s1	17	010 001	temporário salvo
\$s2	18	010 010	temporário salvo
\$s3	19	010 011	temporário salvo
\$s4	20	010 100	temporário salvo
\$s5	21	010 101	temporário salvo
\$s6	22	010 110	temporário salvo
\$s7	23	010 111	temporário salvo
\$t8	24	011 000	temporário
\$t9	25	011 001	temporário
\$k0	26	011 010	reservado para o Kernel do sistema operacional
\$k1	27	011 011	reservado para o Kernel do sistema operacional
\$gp	28	011 100	ponteiro para área global
\$sp	29	011 101	stack pointer (aponta para o último local da pilha)
\$fp	30	011 110	frame pointer (aponta para a primeira palavra do frame de pilha do procedimento)
\$ra	31	011 111	endereço de retorno de uma chamada de procedimento



# Alguns OPCODES

OPCODE	SIGNIFICADO	DECIMAL	BINÁRIO
ADD	soma	32	100 000
SUB	Subtração	34	100 010
OR	Ou lógico	36	100 100
AND	And lógico	37	100 101
MULT	Multiplicação	24	011 000
DIV	Divisão	26	011 010
BNE	Branch if Not Equal	4	000 100
BEQ	Branch if Equal	5	000 101
J	Jump to Address	2	000 010
JR	Jump to Address in Register	8	001 000
LW	Load Word	35	100 011
SW	Store Word	43	101 011

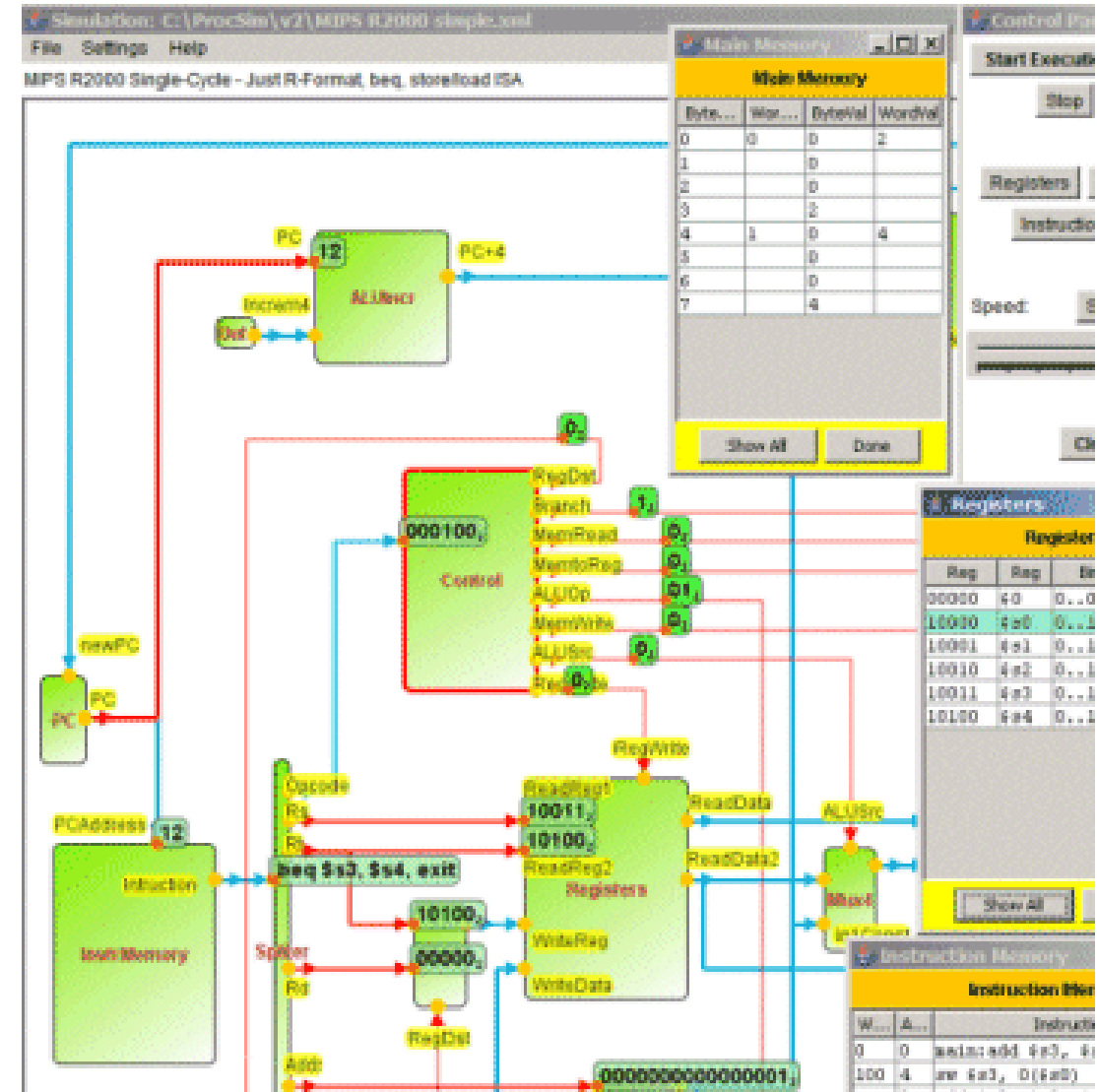
# Exemplos de Codificação

## 1. ADD \$S3, \$S1, \$S2

- ADD = 100000
- \$S3 = 10011
- \$S1 = 10001
- \$S2 = 10010

## 2. SUB \$S5, \$S1, \$S2

- SUB = 100010
- \$S5 = 10101
- \$S1 = 10001
- \$S2 = 10010



# Convertendo para Linguagem de Máquina

- Cada registrador tem um número, conforme apresentado na Tabela 1. A Linguagem de Máquina consiste em trocar o NOME DO REGISTRADOR pelo seu NÚMERO. A instrução ficará da seguinte forma:

**ADD \$8, \$16, \$17**

- O próximo passo é fazer a REPRESENTAÇÃO da instrução. No MIPS existem três FORMATOS de instruções, veremos um agora, que é o **formato do tipo R** (registrador).

# Instrução do Tipo R

<b>op</b>	<b>rs</b>	<b>rt</b>	<b>rd</b>	<b>shamt</b>	<b>funct</b>
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits
opcode ou código da operação	registrador do primeiro operando fonte	registrador do segundo operando fonte	registrador do operando destino	deslocamento	função de operação ou código de função

# Campos

- **OP** = código de operação que será realizada. O último campo seleciona uma operação secundária, por exemplo, a operação principal é a aritmética, mas a secundária é uma soma. Nem todas as operações têm dois códigos (op-funct) e nesse caso o campo **FUNCT** é setado com o valor zero.
- **RS** e **RT** são destinados aos operandos fontes, em ordem, da esquerda para a direita, conforme aparecem na operação.
- **RD** é destinado para o armazenamento do resultado da operação, (destino).
- **Shamt** é um campo usado para setar uma quantidade de deslocamento. Não será usado por enquanto, por isso será setado com o valor zero.

# Código Binário

- Instruções de formato tipo R sempre terão o **opcode** igual a zero e **funct** será correspondente à instrução específica. A instrução está representada no seu formato específico, agora fica bem mais fácil pra chegar no **código de máquina**, conforme abaixo:

ADD \$t0, \$s0, \$s1

op	rs	rt	rd	shamt	funct
000000	10000	10001	01000	00000	100000

- O código binário **00000010000100010100000000100000** é a instrução

**a = b + c**

em MIPS 32 bits.

# Exercícios

1. Escreva o código binário das instruções contidas no código exemplo do MIPS R2000 Just R-Format (Simulador):
  - a) ADD \$S3, \$S1, \$S1
  - b) ADD \$S4, \$S1, \$S0
  - c) SUB \$S5, \$S1, \$S2
  - d) SUB \$S6, \$S2, \$S1
  - e) ADD \$S7, \$S2, \$S1
  
2. Indique quais opcodes o simulador Procsim implementa e apresente o respectivo código binário de cada instrução.