Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Unidade Acadêmica de Sistemas e Computação
Curso de Bacharelado em Ciência da Computação

Organização e Arquitetura de Computadores

(Processador – Parte I – A)

Profa. Joseana Macêdo Fechine Régis de Araújo
joseana@computacao.ufcg.edu.br

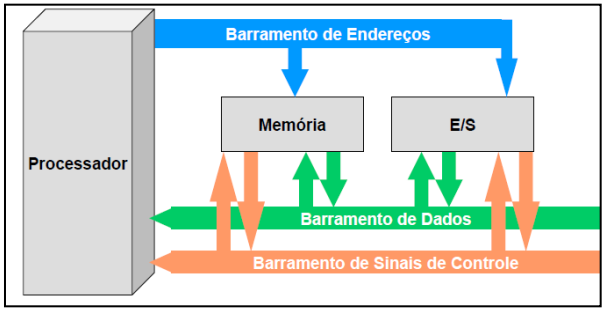
Carga Horária: 60 horas

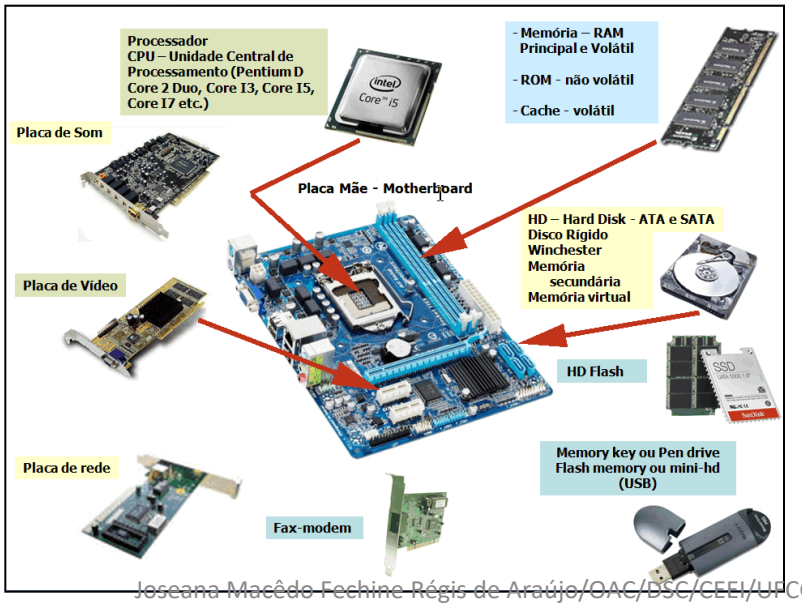
Tópicos

- **Organização e Arquitetura Básicas de Computadores**
  - Conceitos Básicos (Processadores)

Organização e Arquitetura Básicas de Computadores

- **Composição Básica de um Computador Digital**





Organização e Arquitetura Básicas de Computadores

- **Composição básica de um Computador digital**
  - **Processador**
  - Memória
    - **Memória Principal**
    - **Memória Secundária**
  - Dispositivos de entrada e saída interligados

Organização e Arquitetura Básicas de Computadores

UCP (CPU) - Componentes Fundamentais

- Unidade de Controle
- Unidade Lógica e Aritmética
- Registradores
- Sistemas de Comunicação (Barramentos)

Organização e Arquitetura Básicas de Computadores

UC - Unidade de Controle

- **Funções**: busca, interpretação e controle de execução das instruções e o controle dos demais componentes do computador.
- Envia ordens de cálculo para a ULA, que indica os valores a processar e os coloca nos registradores para esse efeito.
- A partir da UC a informação é transferida para as outras partes que constituem o computador, como a memória, os sistemas de E/S, etc..

Organização e Arquitetura Básicas de Computadores

ULA - Unidade Lógica e Aritmética

- **Função**: a execução efetiva das instruções.
- Circuitos lógicos e componentes eletrônicos simples que, integrados, realizam as operações aritméticas e lógicas (soma, subtração, multiplicação, divisão, AND, OR, XOR, complemento, deslocamento, incremento e decremento).
- Processadores modernos utilizam mais de uma ULA.

Organização e Arquitetura Básicas de Computadores

Registradores

- **Função**: armazenamento de dados e resultados que serão usados pela ULA.
- Servem de memória auxiliar básica para a ULA.
- Classificação (atual): registradores de uso geral e registradores de uso específico.
- Em geral, os registradores de dados da UCP têm uma largura (quantidade de bits que podem armazenar) igual ao tamanho estabelecido pelo fabricante para a **palavra do referido processador**.
- A quantidade e o emprego dos registradores variam bastante entre modelos de UCP.

Organização e Arquitetura Básicas de Computadores

Registradores

- Como estão dentro do processador, podem ser lidos e escritos a uma velocidade bastante alta.
- Exemplos:
  - *Program Counter* (PC): armazena o endereço da próxima instrução
  - *Registrador de Instruções* (IR): armazena instrução que está sendo executada.
  - Registradores de uso geral, registradores de segmentos, registrador FLAGS (PSW - *Program Status Word*), ...

Mais informações:
http://www.numaboa.com.br/informatica/oiciliS/assembler/referencias/arquitetura.php

Organização e Arquitetura Básicas de Computadores

Barramentos

- Conjunto de fios paralelos que permite a transmissão de dados, endereços, sinais de controle e instruções
- Tipos: barramentos internos e externos ao processador

Organização e Arquitetura Básicas de Computadores

CPU - Elementos importantes

- Caminho de dados
- Ciclo de Busca-Decodificação-Execução de Instrução
- Execução de Instruções

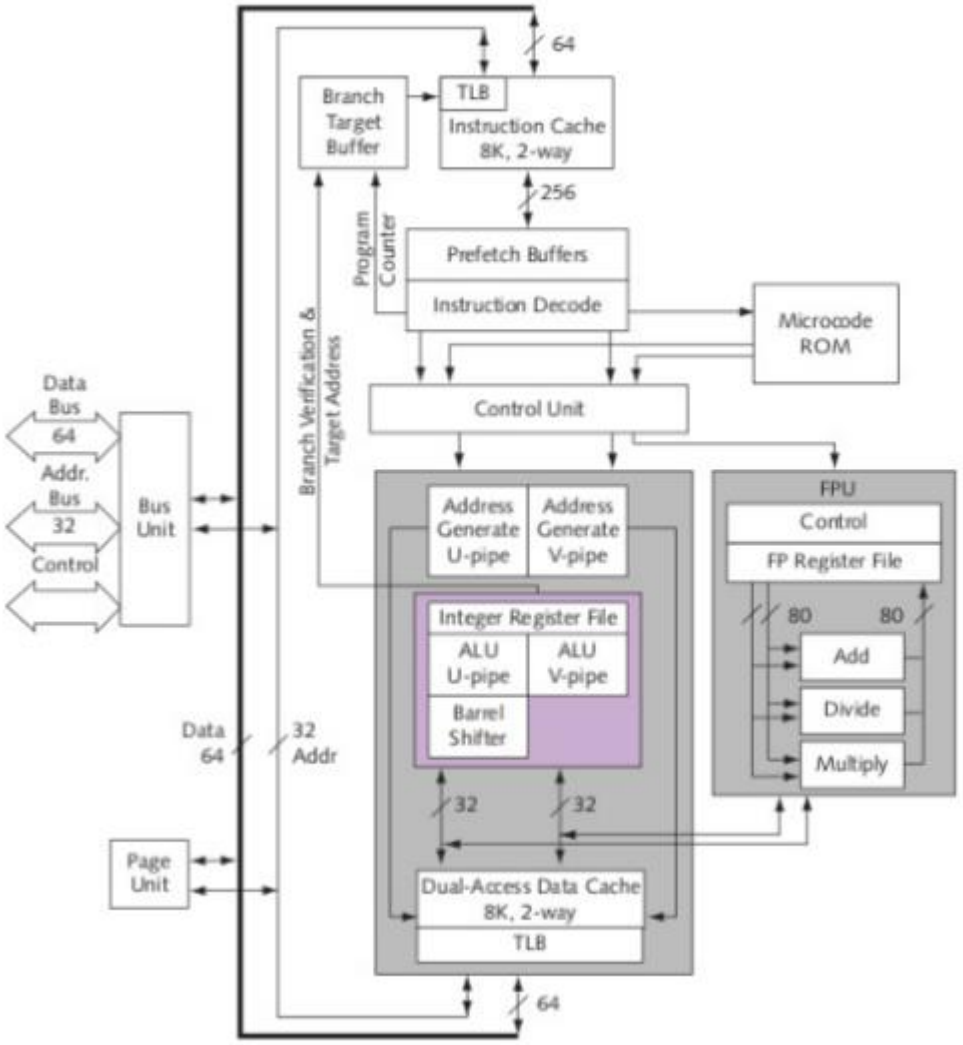
Organização e Arquitetura Básicas de Computadores

Arquitetura RISC x CISC

- CISC (*Complex Instruction Set Computing*, Computador com um Conjunto Complexo de Instruções): suporta mais instruções. A execução dessas fica mais lenta.
- RISC (*Reduced Instruction Set Computing*, Computador com um Conjunto Reduzido de Instruções): suporta menos instruções. A execução dessas fica mais rápida.
- O x86 atualmente é mais CISC do que RISC.
- Exemplos de Microprocessadores com Arquitetura RISC: SPARC, MIPS e PowerPC e processadores dos videogames.

Organização e Arquitetura Básicas de Computadores

Arquitetura de um Computador com Processador Intel

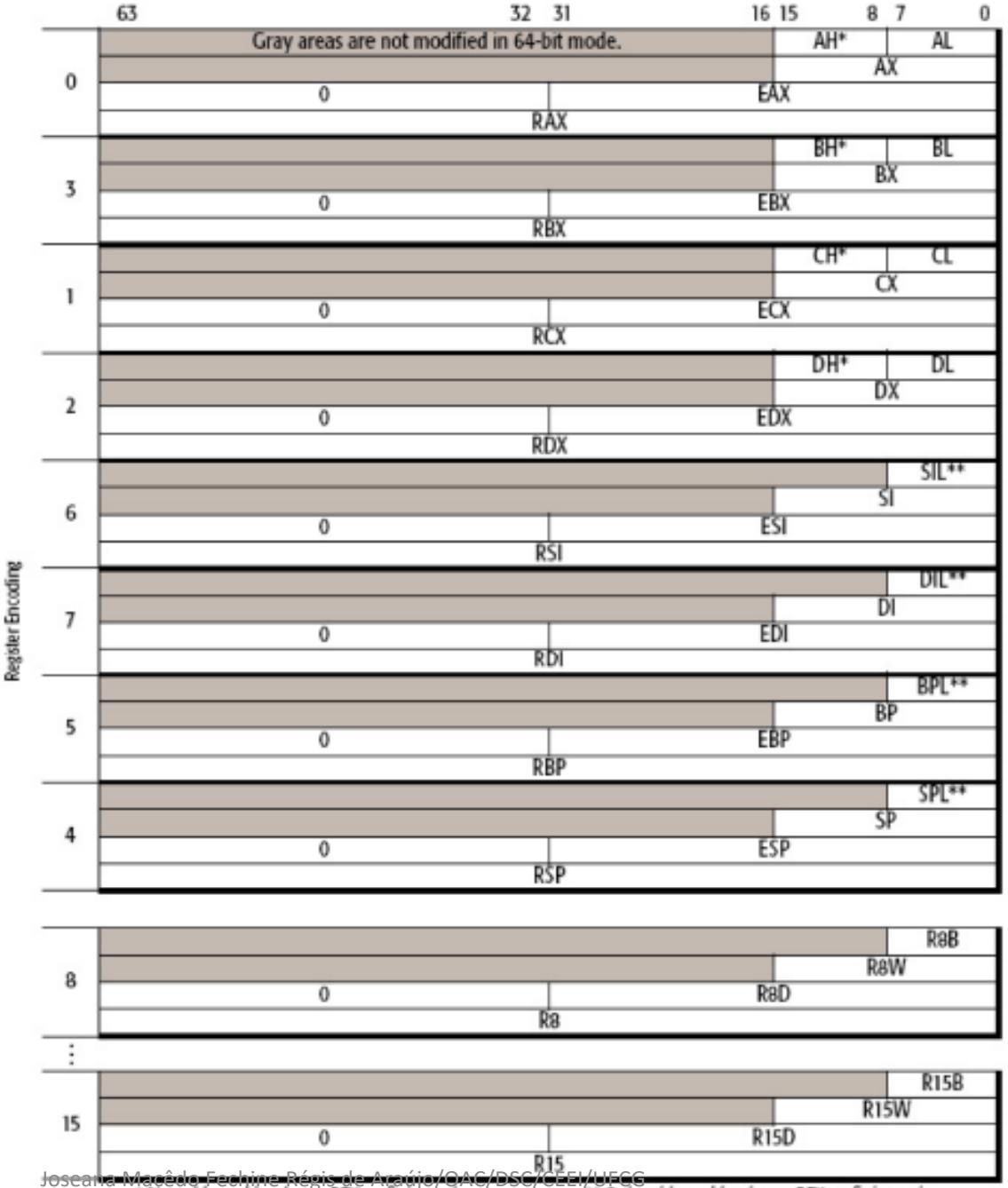


Organização e Arquitetura Básicas de Computadores

Arquitetura IA-64

- Máquina completa de 64 bits
- 128 registradores de uso geral de 64 bits.
- 128 registradores de ponto flutuante
- 64 registradores de bits (predicação)
- Janelas de registradores com tamanho variável
- Vários outros registradores

Registradores (Intel)



Organização e Arquitetura Básicas de Computadores

Registradores (Processadores de 64 bits)

- Os registradores do x86 foram estendidos para 64 bits e receberam o prefixo “R”.
- Registrador de 64 bits correspondente ao EAX -> RAX.
- Foram acrescentados mais 8 registradores de uso geral: R8-R15.
- Foram acrescentados 8 registradores XMM: XMM8-XMM15
- O apontador de instrução, EIP, foi ampliado também, e agora se chama RIP.
- Registrador de FLAGS aumentou, embora não foram acrescentadas novas flags.

Organização e Arquitetura Básicas de Computadores

Arquitetura IA-64

Tipos de instruções típicas:

- LOAD, STORE - realizam o movimento de dados e instruções entre memória e registradores
- MOVE - realizam cópia de valores entre registros
- ADD, SUB, MULT,... - realizam operações aritméticas
- AND, OR, XOR, ... - realizam operações lógicas
- EQ, NEQ, LEQ,... - realizam operações de comparação
- GOTO - operação de desvio

Organização e Arquitetura Básicas de Computadores

Arquitetura de um Computador Simples

- **Composição**
  - Caminho de Dados (*Datapath*)
  - Unidade de Controle para controlar as operações do *Datapath*
- **Especificação de um *Datapath***
  - Um conjunto de registradores
  - As micro-operações
    - ULA e Shifter
  - Uma interface de controle

Organização e Arquitetura Básicas de Computadores

Arquitetura de um Computador Simples

- **Caminho de dados**
  - Parte constituída dos registradores, ULA e barramentos.
  - Os registradores alimentam as duas entradas (A e B) da ULA.
  - A saída da ULA é conectada a um dos registradores.

Organização e Arquitetura Básicas de Computadores

Arquitetura de um Computador Simples

- **Caminho de dados**
  - **Importante:** A velocidade do ciclo do caminho de dados determina, em última análise, a velocidade do computador.
  - **Observação:** “Palavras” – são as unidades de dados movidas entre a memória e os registradores. A referência a uma palavra deve ser feita por meio de um número inteiro.

Organização e Arquitetura Básicas de Computadores

Arquitetura de um Computador Simples

- Independente da classe da instrução, as duas primeiras etapas para sua execução são as mesmas:
  - Enviar o PC para a memória e buscar a instrução;
  - Ler um ou dois registradores (usando o campo da instrução, para selecionar os registradores a serem lidos).
- Após a utilização da ULA, os passos são diferentes para as diferentes classes.

Organização e Arquitetura Básicas de Computadores

Computador Simples - Exemplo

MIPS

(***Microprocessor without Interlocking Pipeline Stages***)

- Arquitetura tipo RISC.
- Versões de 32 e 64 bits.
- Quase 100 milhões de processadores MIPS fabricados em 2009.
- Usada pela NEC, Nintendo, Cisco, Silicon Graphics, Sony, impressoras HP e Fuji, etc.

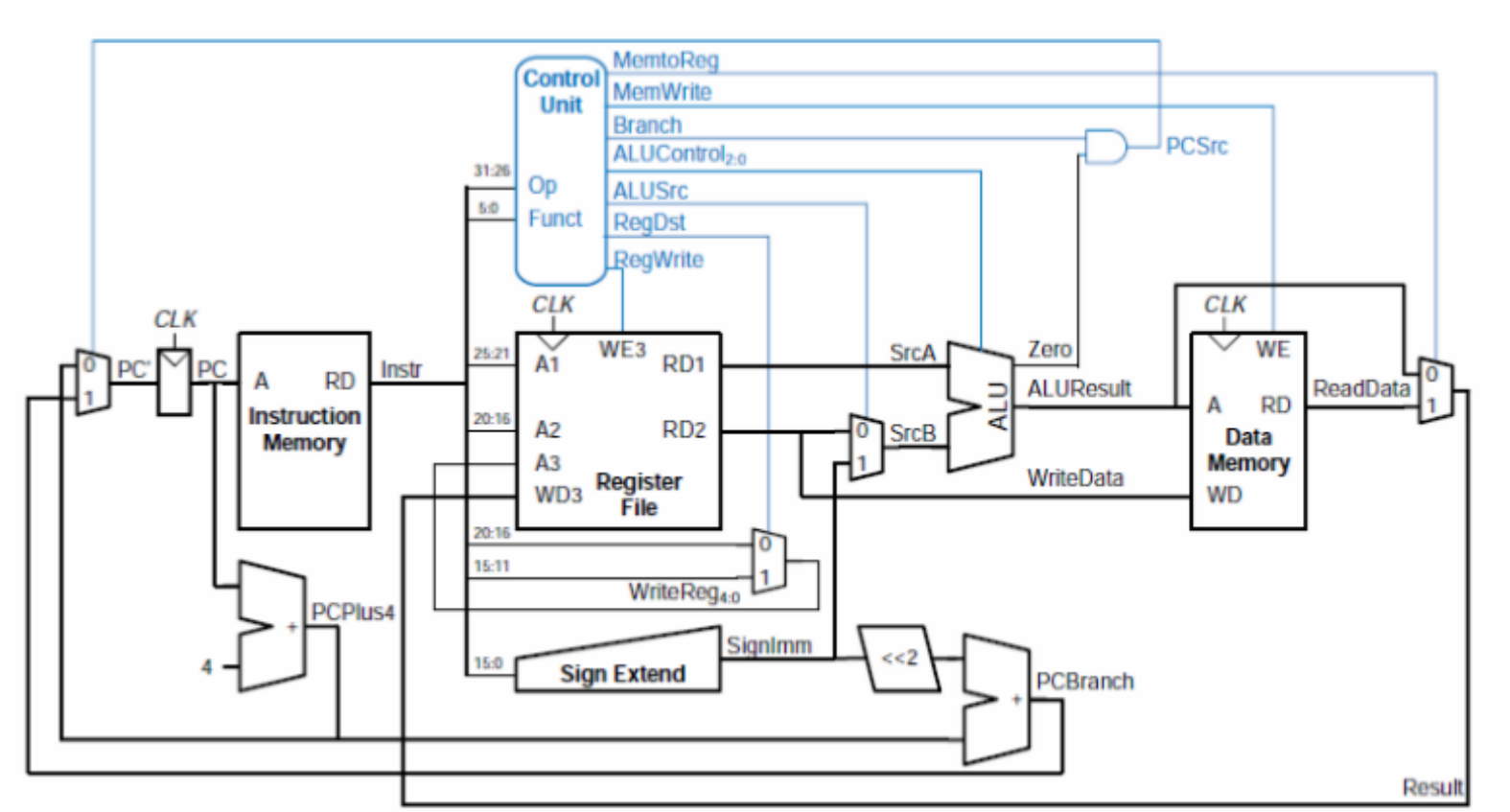
Organização e Arquitetura Básicas de Computadores

MIPS - Componentes Básicos

- Barramento
- Unidade de Controle
- Banco de Registradores
- Unidade Lógica e Aritmética (ULA)
- Contador de Programa (PC)
- Memória
- Registrador de instruções (IR)

Organização e Arquitetura Básicas de Computadores

MIPS – Composição Básica



Organização e Arquitetura Básicas de Computadores

Computador - Exemplo

RISC-V

- Arquitetura tipo RISC.
- *Open Source*, disponível gratuitamente para o meio acadêmico e indústria.
- Versões de 32 e 64 bits (ou 128 bits).
- Suporte para Inteiro e Ponto Flutuante, Padrão IEEE 754.

Mais informações em https://riscv.org/

Organização e Arquitetura Básicas de Computadores

Computador - Exemplo

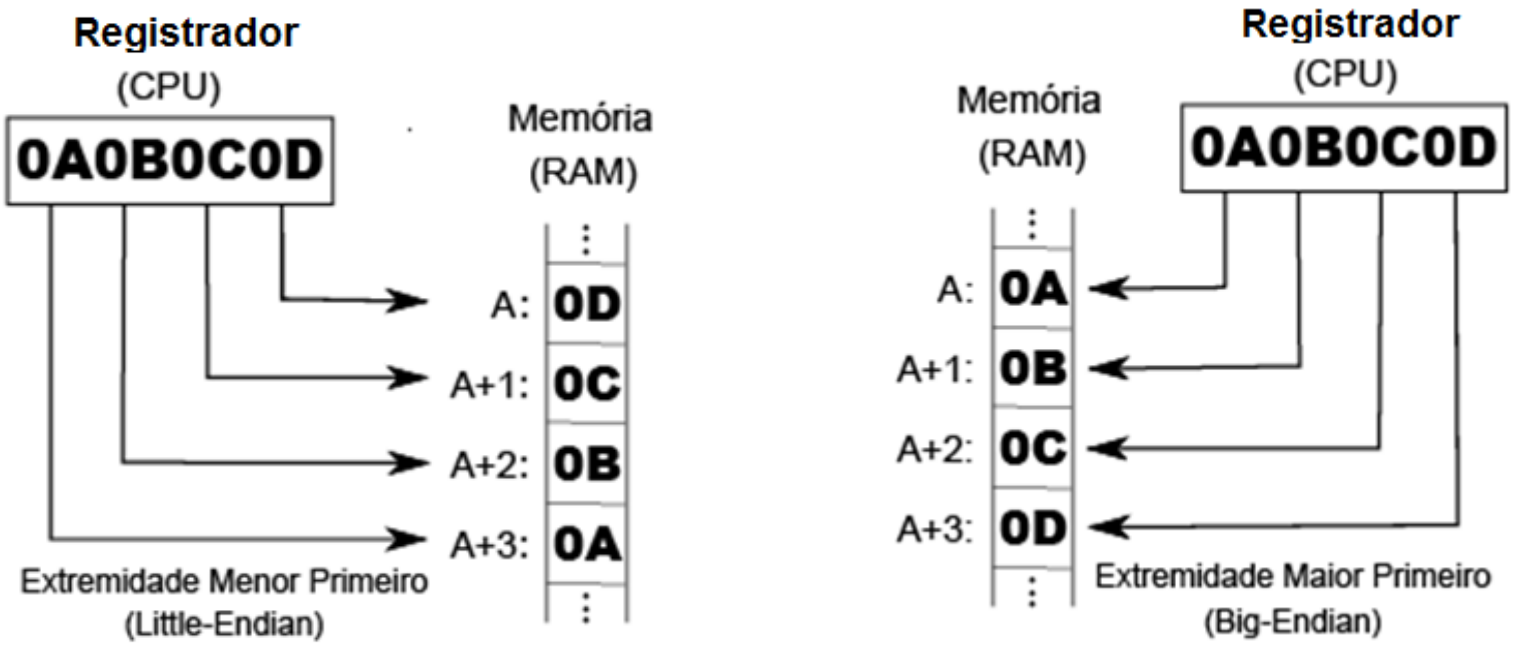
RISC-V

- Para Arquitetura de 32 bits, possui 32 registradores de propósito geral (**x1-x31**) e um Registrador **pc**.
- Armazenamento na Memória: forma Alinhada e *Little-Endian*.
- Possui 6 formatos de Instrução e 47 instruções.

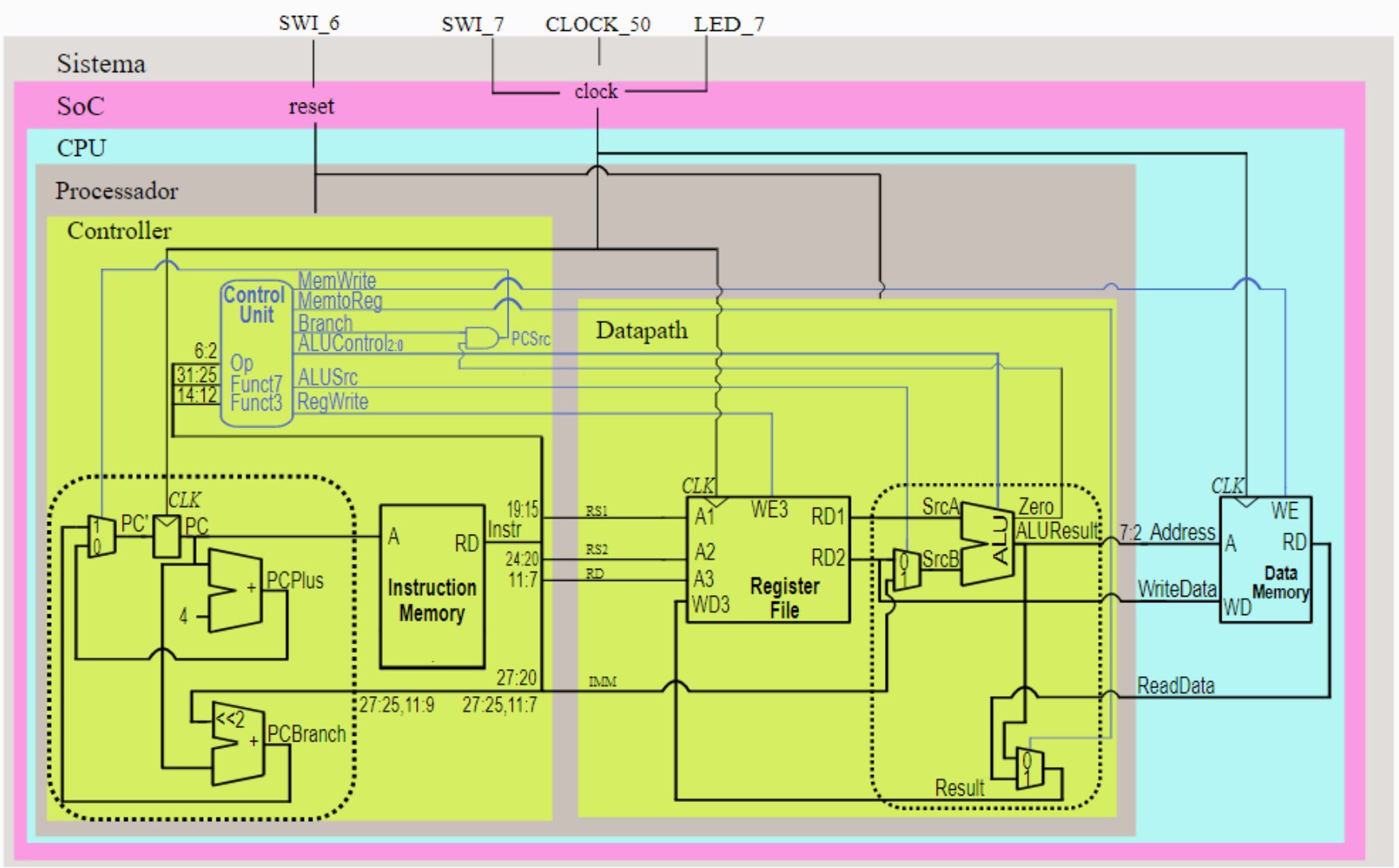
Organização e Arquitetura Básicas de Computadores

Computador - Exemplo

RISC-V



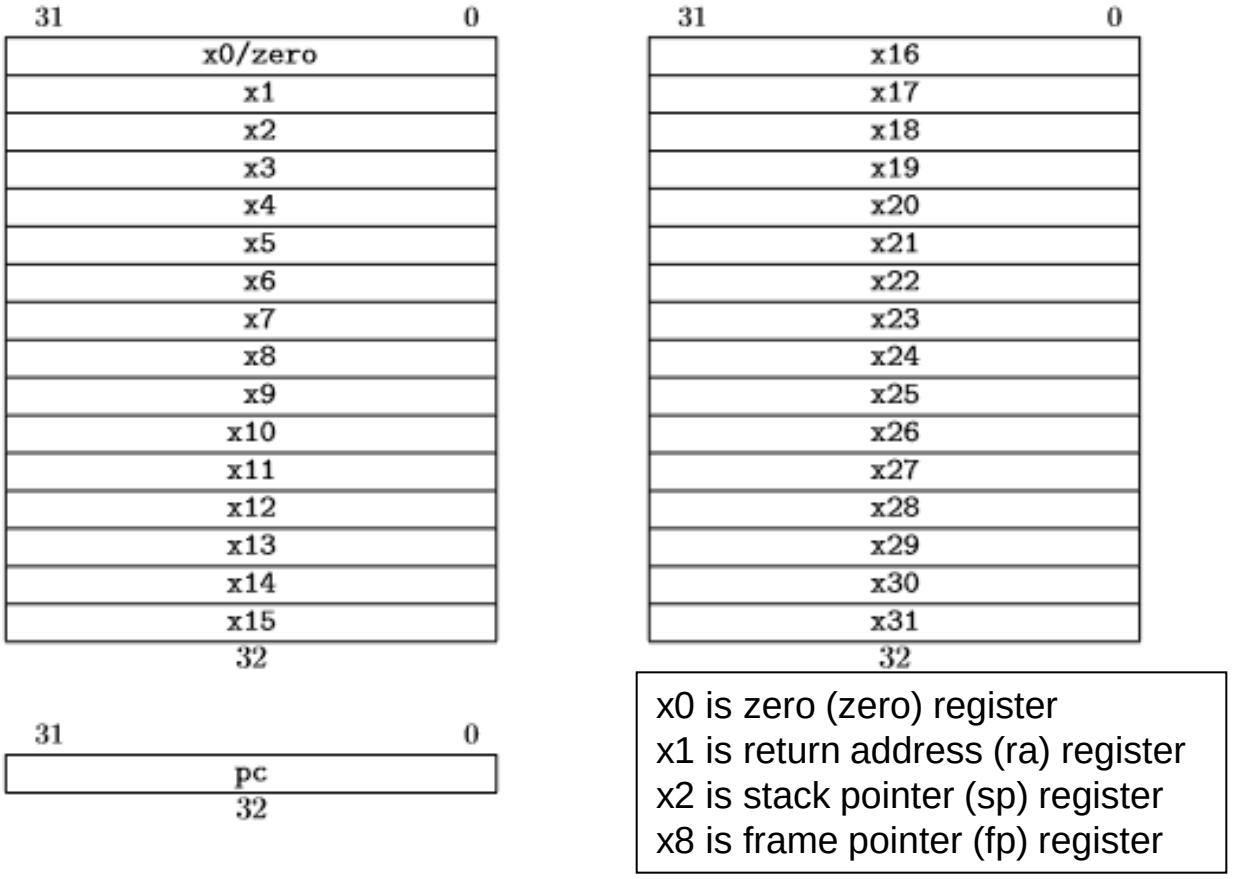
Organização e Arquitetura Básicas de Computadores



Organização e Arquitetura Básicas de Computadores

Arquitetura RISC-V

RV32I: Registradores de Propósito Geral



Organização e Arquitetura Básicas de Computadores

Arquitetura RISC-V

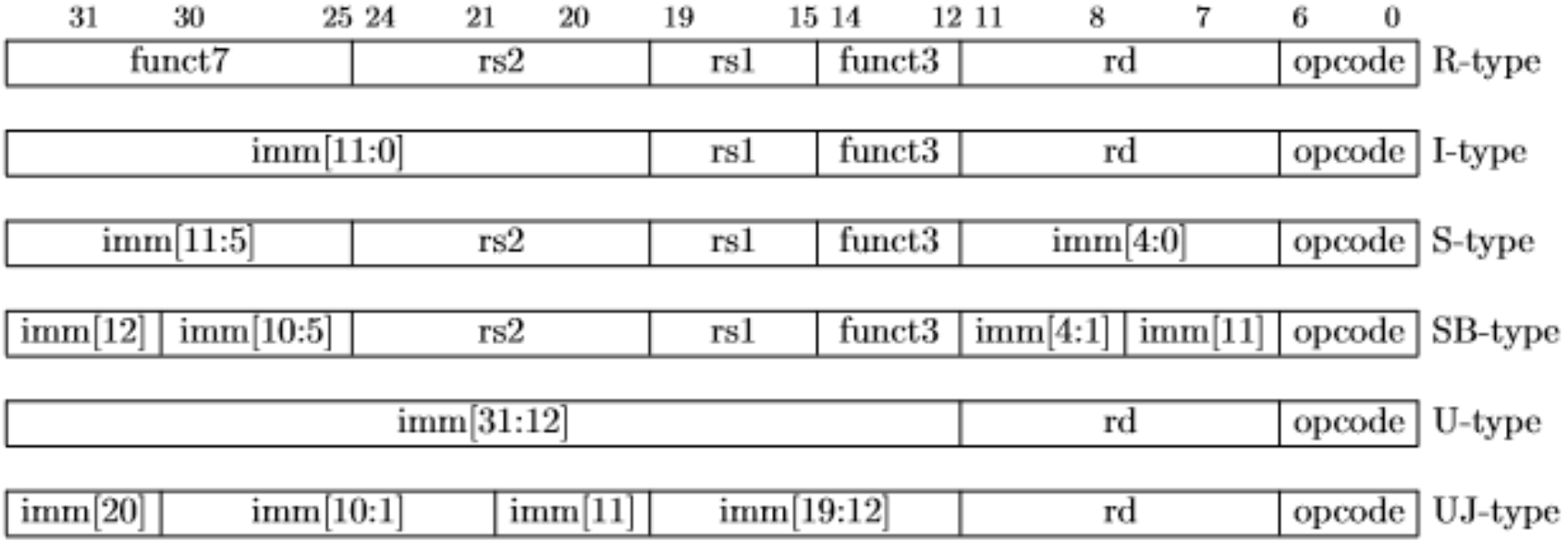
RV32I: Registradores

Register	ABI Name	Description	Saver
x0	zero	Hard-wired zero	—
x1	ra	Return address	Caller
x2	sp	Stack pointer	Callee
x3	gp	Global pointer	—
x4	tp	Thread pointer	—
x5–7	t0–2	Temporaries	Caller
x8	s0/fp	Saved register/frame pointer	Callee
x9	s1	Saved register	Callee
x10–11	a0–1	Function arguments/return values	Caller
x12–17	a2–7	Function arguments	Caller
x18–27	s2–11	Saved registers	Callee
x28–31	t3–6	Temporaries	Caller

Organização e Arquitetura Básicas de Computadores

Arquitetura RISC-V

RV32I: Formatos das Instruções



Organização e Arquitetura Básicas de Computadores

Arquitetura RISC-V

RV32I: Exemplos de Instruções

Instruction	Format	Meaning
`add rd, rs1, rs2`	R	Add registers
`sub rd, rs1, rs2`	R	Subtract registers
`sll rd, rs1, rs2`	R	Shift left logical by register
`srl rd, rs1, rs2`	R	Shift right logical by register
`sra rd, rs1, rs2`	R	Shift right arithmetic by register
`and rd, rs1, rs2`	R	Bitwise AND with register
`or rd, rs1, rs2`	R	Bitwise OR with register
`xor rd, rs1, rs2`	R	Bitwise XOR with register
`slt rd, rs1, rs2`	R	Set if less than register, 2's complement
`sltu rd, rs1, rs2`	R	Set if less than register, unsigned
`addi rd, rs1, imm[11:0]`	I	Add immediate
`slli rd, rs1, shamt[4:0]`	I	Shift left logical by immediate
`srli rd, rs1, shamt[4:0]`	I	Shift right logical by immediate
`srai rd, rs1, shamt[4:0]`	I	Shift right arithmetic by immediate
`andi rd, rs1, imm[11:0]`	I	Bitwise AND with immediate
`ori rd, rs1, imm[11:0]`	I	Bitwise OR with immediate
`xori rd, rs1, imm[11:0]`	I	Bitwise XOR with immediate
`slti rd, rs1, imm[11:0]`	I	Set if less than immediate, 2's complement
`sltiu rd, rs1, imm[11:0]`	I	Set if less than immediate, unsigned
`lui rd, imm[31:12]`	U	Load upper immediate
`auipc rd, imm[31:12]`	U	Add upper immediate to pc

Arquitetura RISC-V

RV32I Base Instruction Set

imm[31:12]				rd	0110111	LUI
imm[31:12]				rd	0010111	AUIPC
imm[20|10:1|11|19:12]				rd	1101111	JAL
imm[11:0]		rs1	000	rd	1100111	JALR
imm[12|10:5]	rs2	rs1	000	imm[4:1|11]	1100011	BEQ
imm[12|10:5]	rs2	rs1	001	imm[4:1|11]	1100011	BNE
imm[12|10:5]	rs2	rs1	100	imm[4:1|11]	1100011	BLT
imm[12|10:5]	rs2	rs1	101	imm[4:1|11]	1100011	BGE
imm[12|10:5]	rs2	rs1	110	imm[4:1|11]	1100011	BLTU
imm[12|10:5]	rs2	rs1	111	imm[4:1|11]	1100011	BGEU
imm[11:0]		rs1	000	rd	0000011	LB
imm[11:0]		rs1	001	rd	0000011	LH
imm[11:0]		rs1	010	rd	0000011	LW
imm[11:0]		rs1	100	rd	0000011	LBU
imm[11:0]		rs1	101	rd	0000011	LHU
imm[11:5]	rs2	rs1	000	imm[4:0]	0100011	SB
imm[11:5]	rs2	rs1	001	imm[4:0]	0100011	SH
imm[11:5]	rs2	rs1	010	imm[4:0]	0100011	SW
imm[11:0]		rs1	000	rd	0010011	ADDI
imm[11:0]		rs1	010	rd	0010011	SLTI
imm[11:0]		rs1	011	rd	0010011	SLTIU
imm[11:0]		rs1	100	rd	0010011	XORI
imm[11:0]		rs1	110	rd	0010011	ORI
imm[11:0]		rs1	111	rd	0010011	ANDI
0000000	shamt	rs1	001	rd	0010011	SLLI
0000000	shamt	rs1	101	rd	0010011	SRLI
0100000	shamt	rs1	101	rd	0010011	SRAI
0000000	rs2	rs1	000	rd	0110011	ADD
0100000	rs2	rs1	000	rd	0110011	SUB
0000000	rs2	rs1	001	rd	0110011	SLL
0000000	rs2	rs1	010	rd	0110011	SLT
0000000	rs2	rs1	011	rd	0110011	SLTU
0000000	rs2	rs1	100	rd	0110011	XOR
0000000	rs2	rs1	101	rd	0110011	SRL
0100000	rs2	rs1	101	rd	0110011	SRA
0000000	rs2	rs1	110	rd	0110011	OR
0000000	rs2	rs1	111	rd	0110011	AND
0000 | pred	succ	00000	000	00000	0001111	FENCE
0000 | 0000	0000	00000	001	00000	0001111	FENCE.I
000000000000		00000	000	00000	1110011	ECALL
000000000001		00000	000	00000	1110011	EBREAK

Organização e Arquitetura Básicas de Computadores

Arquitetura RISC-V

RV32I: Exemplos de Execução de Instruções

pc	Instrução	Assembly	Registradores
00000128	004d0d13	addi x26, x26, 4	x26=001002d8, x26:001002d4
0000012C	00000513	addi x10, x0, 0	x10=00000000
00000130	00100593	addi x11, x0, 1	x11=00000001
00000134	00bd0d33	add x26, x26, x11	x26=001002d9, x26:001002d8, x11:00000001

Organização e Arquitetura Básicas de Computadores

Arquitetura RISC-V

RV32I: Exemplos de Execução de Instruções

```
/* x1 = 1 */
li x1, 1
/* x2 = 2 */
li x2, 2
/* x3 = x1 + x2 */
add x3, x1, x2
/* x2 = *x1 */
ld x2, (x1) /*
*x1 = x0 */
sd x0, (x1)
```

Organização e Arquitetura Básicas de Computadores

Arquitetura RISC-V

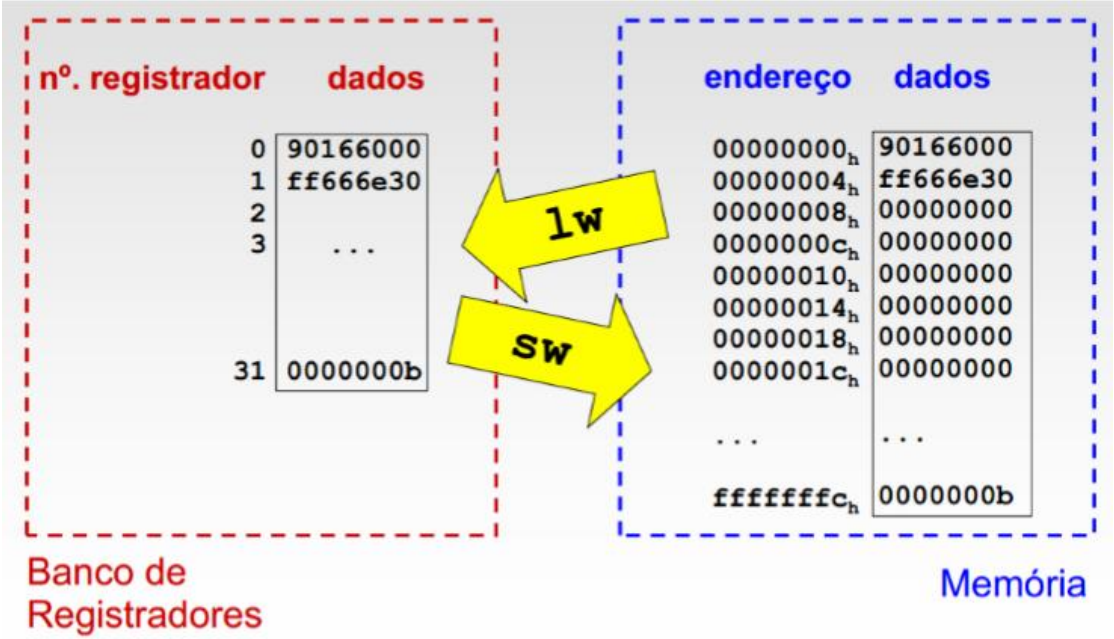
RV32I: Exemplos de Execução de Instruções

```
/* branch if x1 == x2 */
beq x1, x2, loop
/* call */
call func /* jal func */
/* return */
ret /* jr ra */
```

Organização e Arquitetura Básicas de Computadores

Arquitetura RISC-V

RV32I: Instruções (Acesso à Memória)



Copiar dados de → para	Instrução
Memória → Registrador	load word (lw)
Registrador → Memória	store word (sw)

Organização e Arquitetura Básicas de Computadores

Arquitetura RISC-V

RV32I: Instruções (Acesso à Memória)

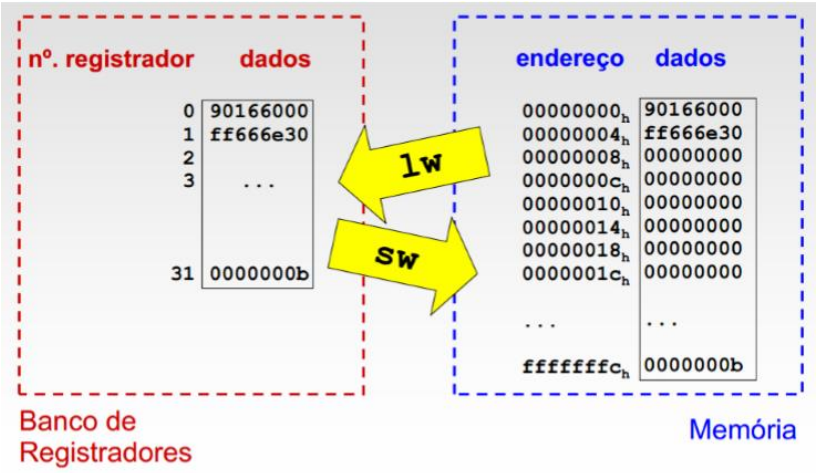
Instruction	Format	Meaning
`lb rd, imm[11:0](rs1)`	I	Load byte, signed
`lbu rd, imm[11:0](rs1)`	I	Load byte, unsigned
`lh rd, imm[11:0](rs1)`	I	Load half-word, signed
`lhu rd, imm[11:0](rs1)`	I	Load half-word, unsigned
`lw rd, imm[11:0](rs1)`	I	Load word
`sb rs2, imm[11:0](rs1)`	S	Store byte
`sh rs2, imm[11:0](rs1)`	S	Store half-word
`sw rs2, imm[11:0](rs1)`	S	Store word
`fence pred, succ`	I	Memory ordering fence
`fence.i`	I	Instruction memory ordering fence

Organização e Arquitetura Básicas de Computadores

Arquitetura RISC-V

RV32I: Exemplos de Execução de Instruções

pc	Instrução	Assembly	Registradores
00000120	000d2d83	lw x27,0(x26)	x27:001002d4 x26:00000004 x27=ff666e30
00000124	000d2023	sw x0,0(x26)	x26:001002d4 PA:001002d4 (0)



Organização e Arquitetura Básicas de Computadores

Arquitetura RISC-V

- **Instruções para tomada de decisão**
  - Alteram o fluxo de controle do programa.
  - Alteram a “próxima” instrução a ser executada.
- **Instruções de controle**
  - Salto condicional.
  - Salto incondicional.

Organização e Arquitetura Básicas de Computadores

Arquitetura RISC-V

RV32I: Instruções (Controle)

Instruction	Format	Meaning
`beq rs1, rs2, imm[12:1]`	SB	Branch if equal
`bne rs1, rs2, imm[12:1]`	SB	Branch if not equal
`blt rs1, rs2, imm[12:1]`	SB	Branch if less than, 2's complement
`bltu rs1, rs2, imm[12:1]`	SB	Branch if less than, unsigned
`bge rs1, rs2, imm[12:1]`	SB	Branch if greater or equal, 2's complement
`bgeu rs1, rs2, imm[12:1]`	SB	Branch if greater or equal, unsigned
`jal rd, imm[20:1]`	UJ	Jump and link
`jalr rd, rs1, imm[11:0]`	I	Jump and link register

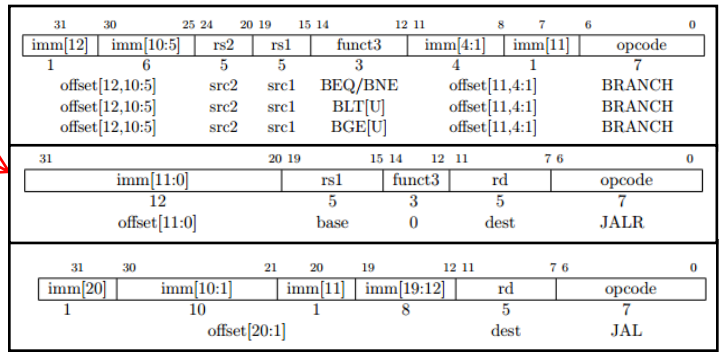
Organização e Arquitetura Básicas de Computadores

Arquitetura RISC-V

RV32I: Exemplos de Execução de Instruções

pc	Instrução	Assembly	Registradores
00000120	ffaddce3	bge x27, x26, -8	x27:001002d4 x26:001002d4
00000124	ffad8ce3	beq x27, x26, -8	x27:001002d4 x26:001002d4
00000128	008d0de7	jalr x27, x26, 8	x27:001002d4 x26:001002d4

```
Cálculo do valor "imm" -8:
-8[31:0] = 11111111111111111111111111111000
-8[11:0] =                      1111111111000
-8[12:1] =                      1111111111100
```



Organização e Arquitetura Básicas de Computadores

Importante

- Instruções podem ter tamanhos diversos (complica o projeto, mas proporciona economia de memória) ou

- Instruções podem ser todas de tamanhos iguais (simplifica o projeto, mas desperdiça espaço. **Por que?**)

Organização e Arquitetura Básicas de Computadores

Importante

- O tamanho ideal de uma instrução deve considerar, além do preço da memória, o tempo de decodificação e de execução de uma instrução.
- Como os processadores modernos são capazes de executar várias instruções no mesmo ciclo de clock, torna-se imperativo um mecanismo de busca de várias instruções em cada ciclo de clock (memórias cache).
- Quando uma instrução tem endereços, o tamanho do endereço deve ser compatível com o tamanho máximo da memória do computador. Mas, memórias maiores requerem endereços mais longos resultando em instruções maiores.