RISC-V

Implementação Monociclo

GEX 612 - Organização de Computadores

Prof. Luciano L. Caimi
lcaimi@uffs.edu.br

Roteiro

Introdução

Busca da instrução

Instruções aritméticas

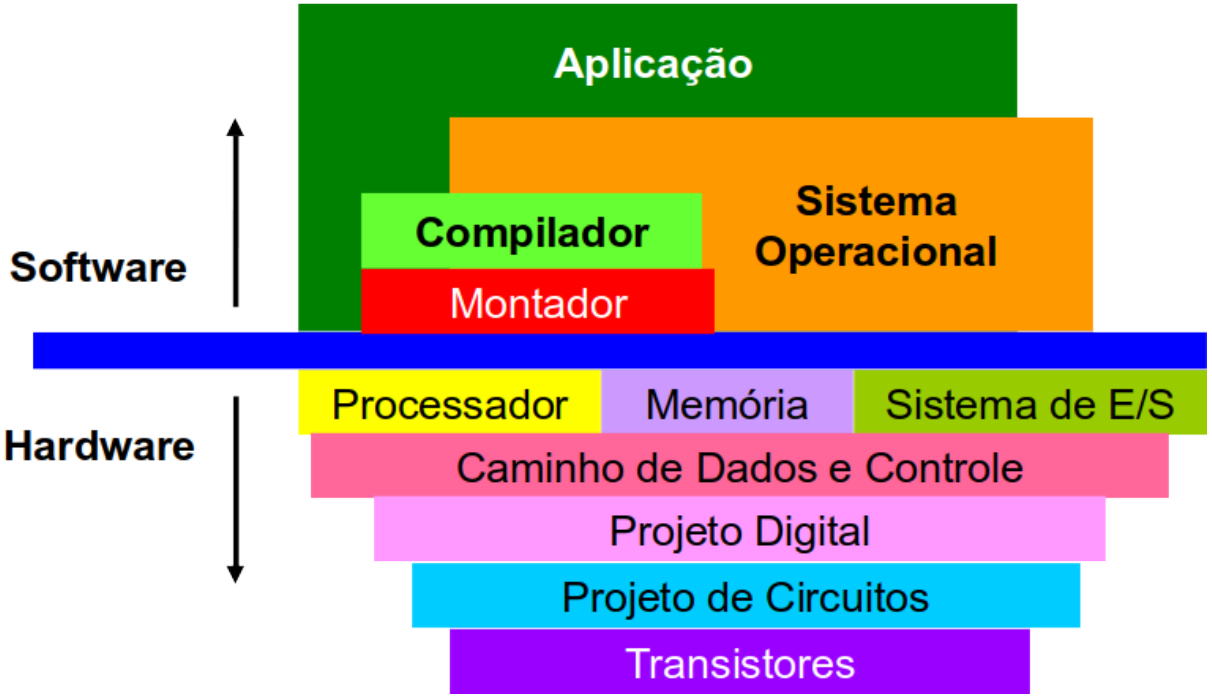
Instruções de acesso à memória

Instruções de Desvio

Combinando instruções

Bloco operativo completo

Introdução: Arquitetura Multinível



Introdução: RISC-V - ISA

Formato das Instruções:

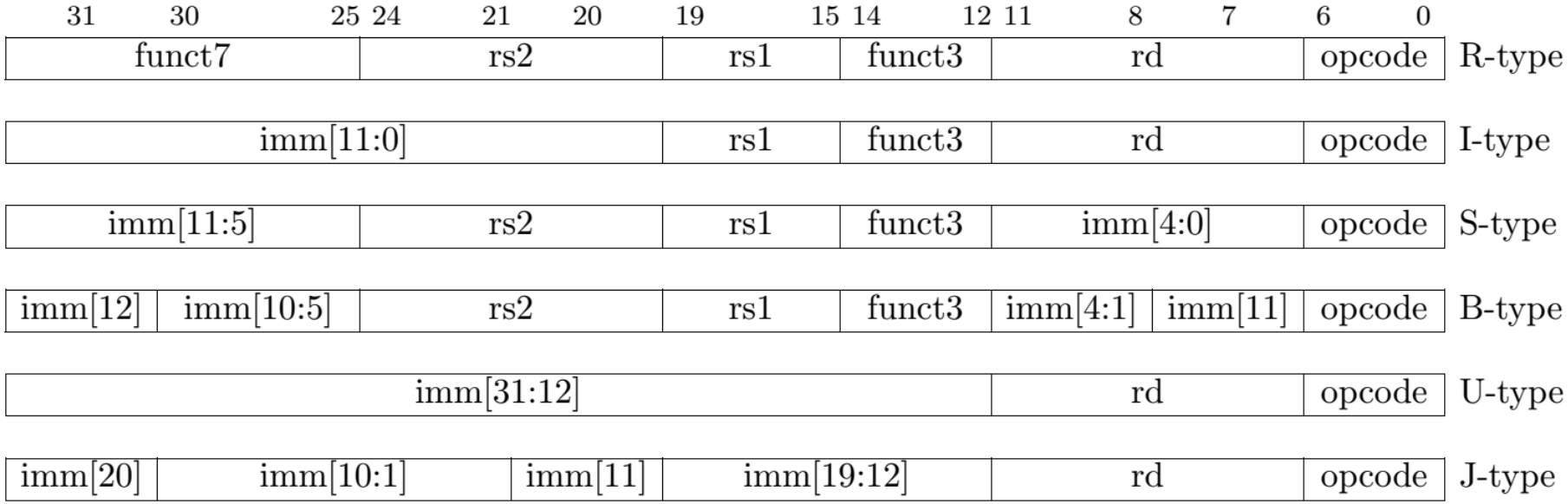
32-bit RISC-V Instruction Formats

Instruction Formats	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Register/register	funct7							rs2					rs1					funct3			rd					opcode						
Immediate	imm[11:0]												rs1					funct3			rd					opcode						
Upper Immediate	imm[31:12]																				rd					opcode						
Store	imm[11:5]							rs2					rs1					funct3			imm[4:0]					opcode						
Branch	[12]	imm[10:5]						rs2					rs1					funct3			imm[4:1]				[11]	opcode						
Jump	[20]	imm[10:1]										[11]	imm[19:12]								rd					opcode						

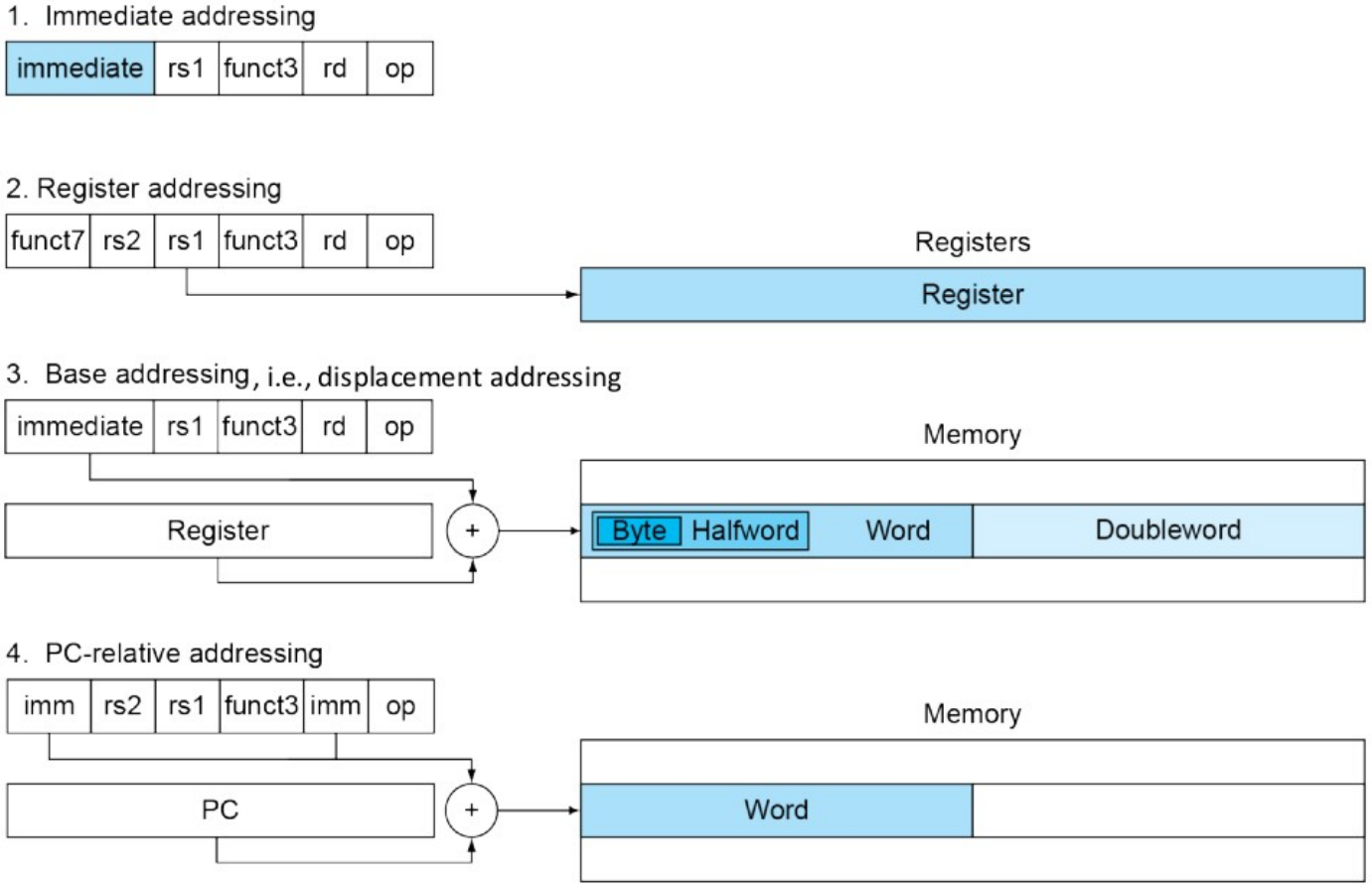
- **opcode (7 bit):** partially specifies which of the 6 types of *instruction formats*
- **funct7 + funct3 (10 bit):** combined with **opcode**, these two fields describe what operation to perform
- **rs1 (5 bit):** specifies register containing first operand
- **rs2 (5 bit):** specifies second register operand
- **rd (5 bit)::** Destination register specifies register which will receive result of computation

Introdução: RISC-V - ISA

Formato das Instruções:



Introdução: RISC-V - modos de endereçamento



Introdução

- **ISA simplificado contendo somente:**
  - Instruções de referência a memória: lw, sw
  - Instruções lógico-aritméticas: add, sub, and, slt
  - Instruções de controle de fluxo: beq, j
- **Implementação do Ciclo de Instrução Básico**

  1) Busca de instruções da memória
  2) Decodifica a instrução a partir do OpCode
  3) Acessa o Banco de Registradores (BR) para ler os dados
  4) Executa a instrução
  5) Armazena o Resultado (na memória ou no BR)

Introdução: RISC-V - ISA

Instruções RV32I

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
csr		rs1	001	rd	1110011	CSRRW
csr		rs1	010	rd	1110011	CSRRS
csr		rs1	011	rd	1110011	CSRRC
csr		zimm	101	rd	1110011	CSRRWI
csr		zimm	110	rd	1110011	CSRRSI
csr		zimm	111	rd	1110011	CSRRCI

Introdução

- Todas as instruções usam a ALU após a leitura dos registradores

  Porque? Referência a memória!

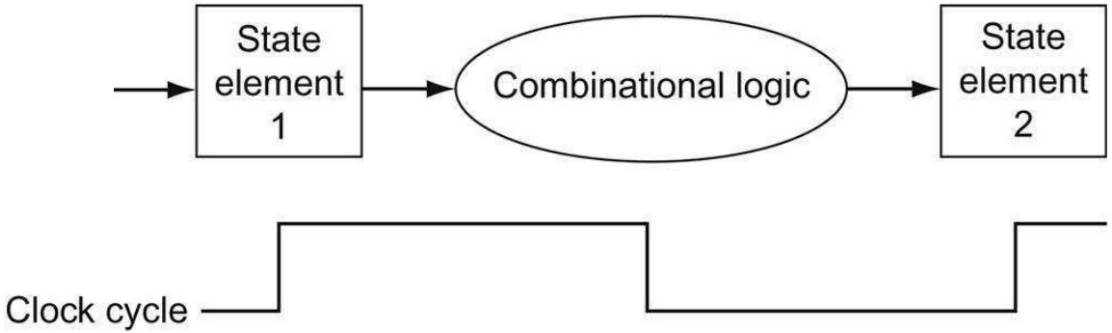
  Aritmética!

  Controle de fluxo!

- Contador de Programa (PC) para endereçar instruções a serem executadas

Introdução

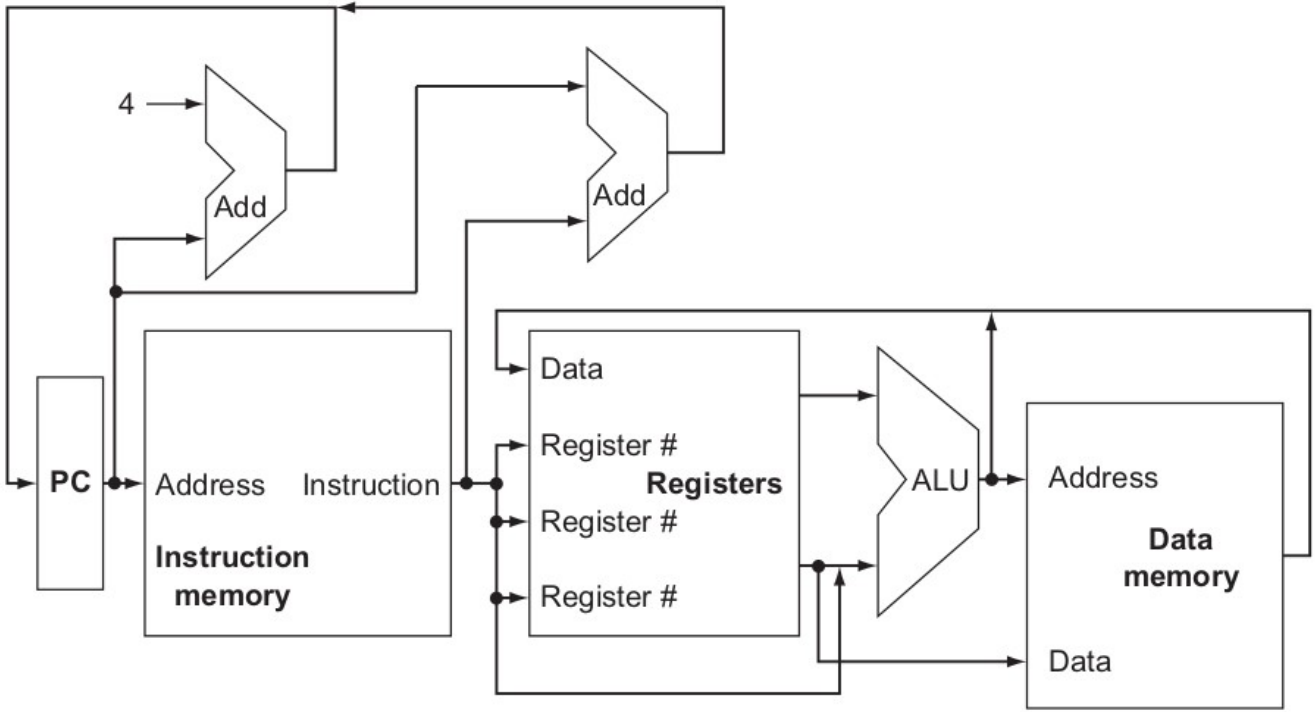
- Elementos de armazenamentos gatilhados na borda de subida do clock



- Valor é armazenado no final do ciclo de clock anterior e lido no início do clock seguinte
- Saída é igual ao valor armazenado no elemento (não é necessário permissão para ler o valor)

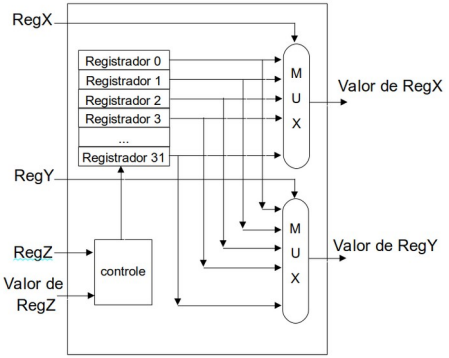
Introdução

- Visão Geral

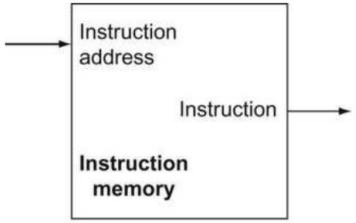


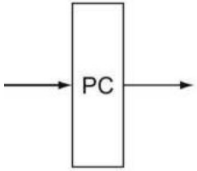
Introdução

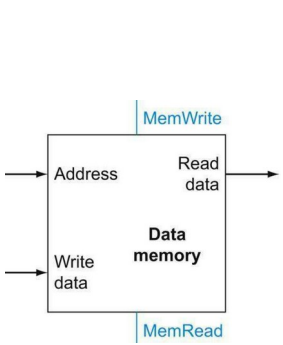
- **Banco de Registradores:**

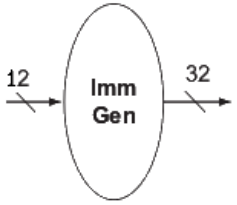


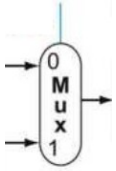
- **Demais Unidades:**

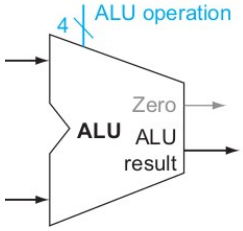


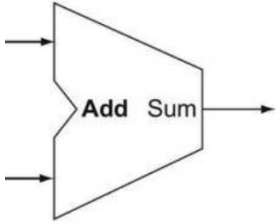


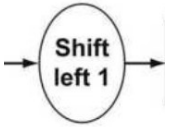






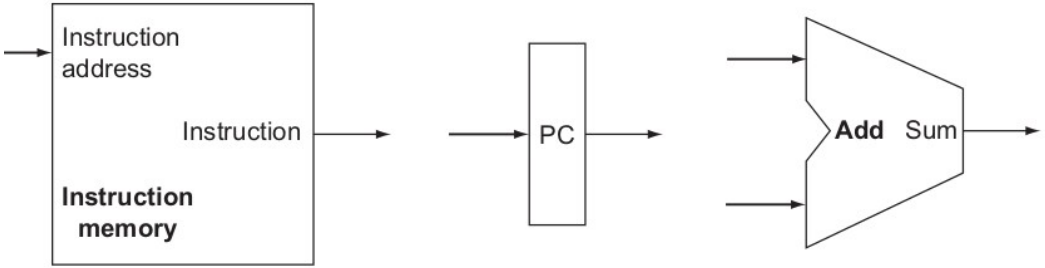






Busca da Instrução

- **Três elementos são necessários para executar uma busca de instrução**
  - a memória onde estão armazenadas as instruções
  - o contador de programa (PC) para armazenar o endereço da instrução
  - um somador é necessário para calcular o endereço da próxima instrução

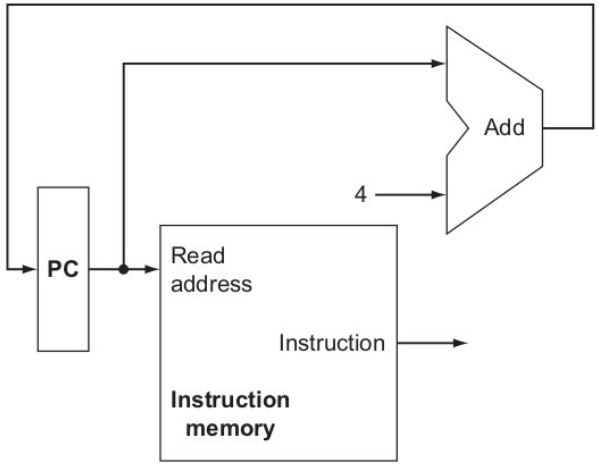


Busca da Instrução

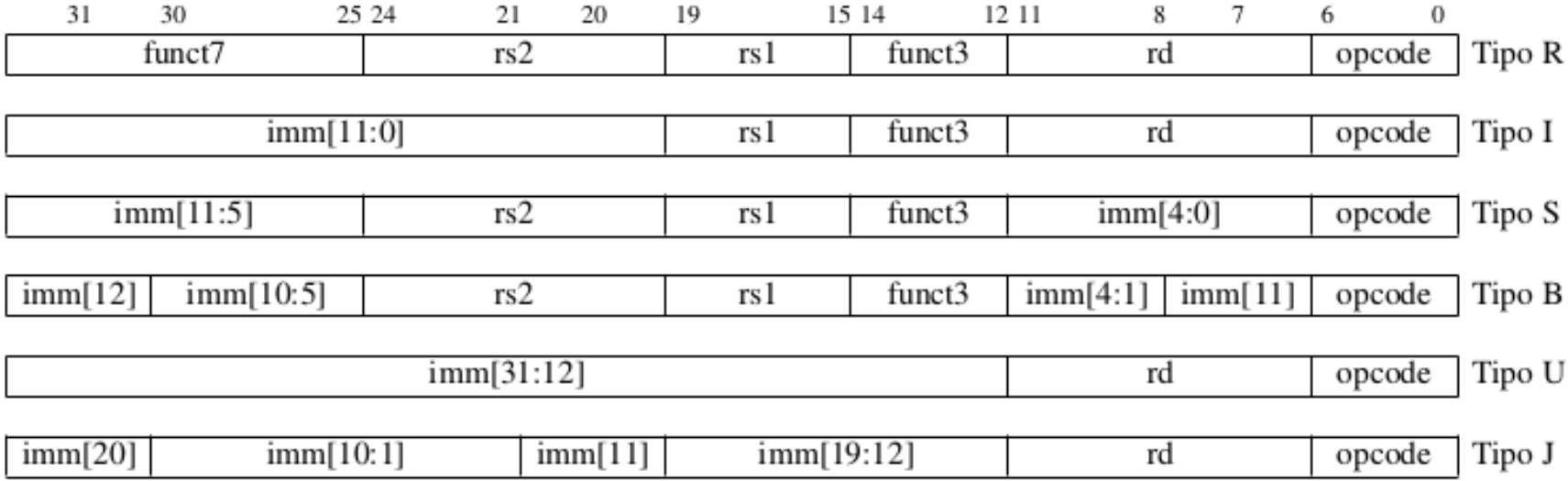
- **Lê Instrução na memória de programa e atualiza PC**
  - O contador de programa (PC) contém o endereço da instrução a ser executada
  - O endereço da próxima instrução é obtido pela soma de 4 posições ao contador de programa (PC)

instrucao ← [PC];

PC ← PC + 4



- Formatos das instruções do RISC-V



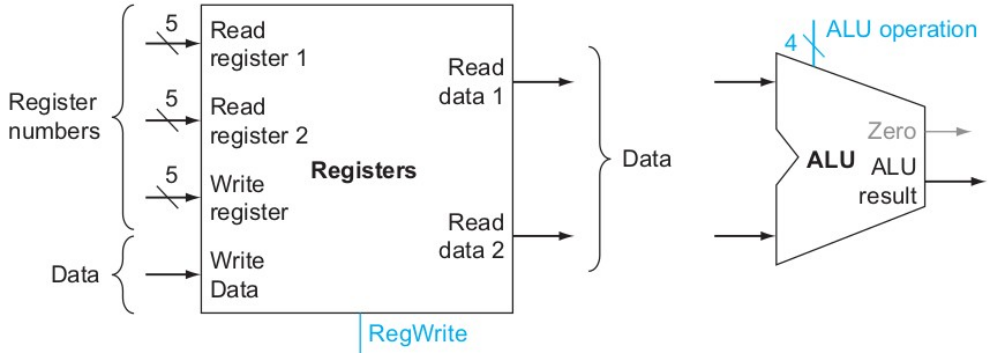
Instruções de formato R

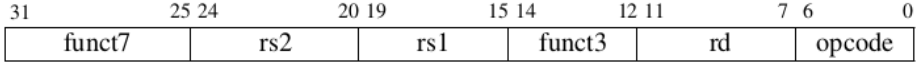
Instruction Formats	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Register/register	funct7							rs2					rs1					funct3			rd					opcode							
0000000	rs2							rs1					000					rd			0110011											ADD
0100000	rs2							rs1					000					rd			0110011											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

<MNE> rd, rs1, rs2 # reg[rd] ← reg[rs1] MNE reg[rs2]

Instruções de formato R

- **Dois elementos são necessários para executar instruções de formato R (R-format)**
  - o Banco de registradores para ler os operandos e armazenar o resultado da instrução
  - a Unidade Lógica/Aritmética (ALU) que será utilizada para executar as instruções



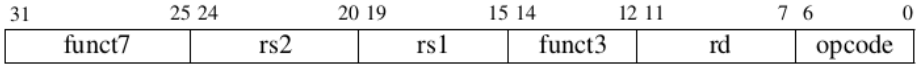


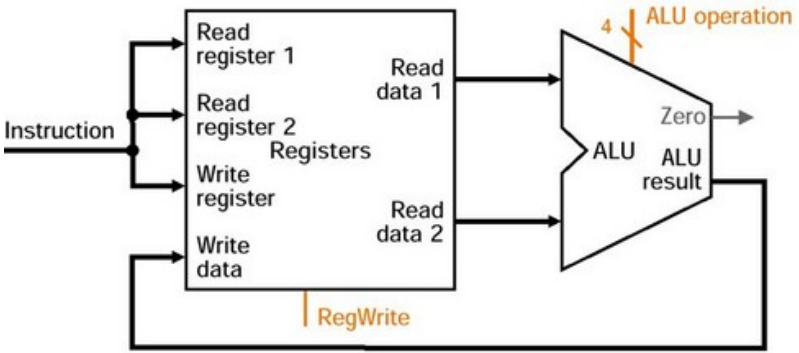
Tipo R

Instruções de formato R

- **Caminho de dados do formato R (R-format)**
  - A instrução contém o endereço de três registradores
  - Dois registradores são lidos e seus valores vão para a ULA
  - O resultado da operação na ULA é armazenado em um terceiro registrador
  - O controle da ULA determina a operação que será realizada (a partir do código da instrução - Opcode)

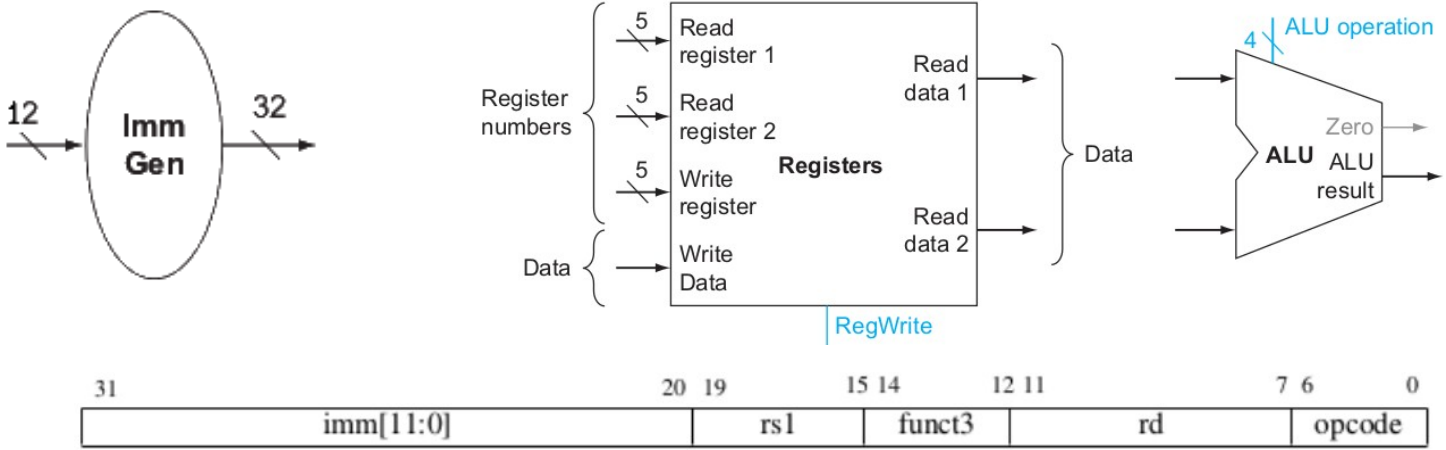
rd ← rs1 operation rs2





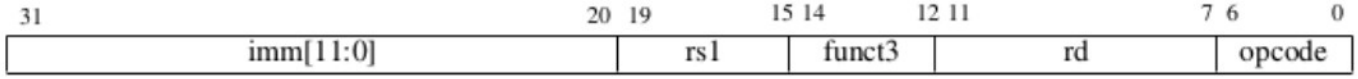
Instruções de formato I

- **Caminho de dados das instruções de formato I utilizam:**
  - módulo de extensão de sinal (valor imediato presente na instrução)
  - banco de registradores (registrador de origem e destino
  - ALU (cálculo da instrução)

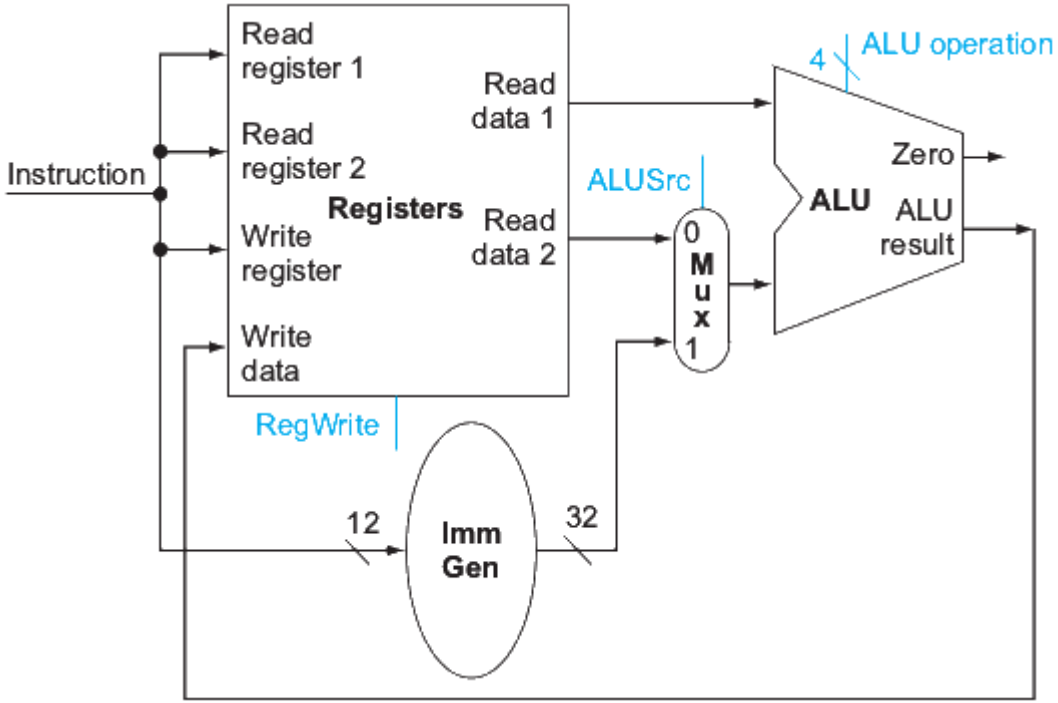


Instruções de formato I

- **Caminho de dados para R-Format + I-Format**
  - MUX inserido na 2ª entrada da ULA

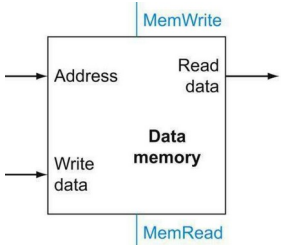


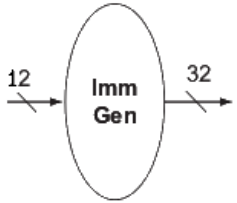
rd ← rs1 operation ImmGen(imm)

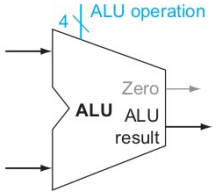


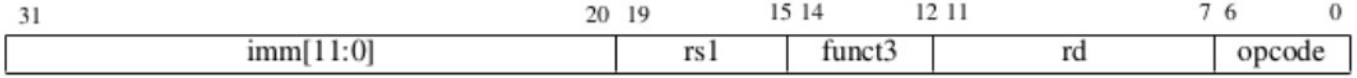
Instruções de acesso a memória

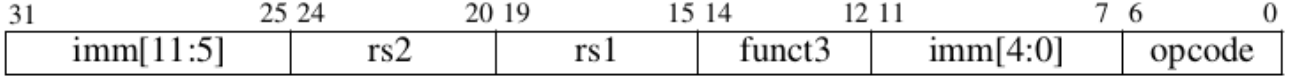
- **Caminho de dados das instruções de acesso a memória utilizam:**
  - memória de dados (onde o dado é lido ou escrito)
  - módulo de extensão de sinal (valor imediato presente na instrução)
  - banco de registradores (registrador apontador e registrador origem (SW) ou destino (LW)
  - ALU (cálculo do endereço de acesso)





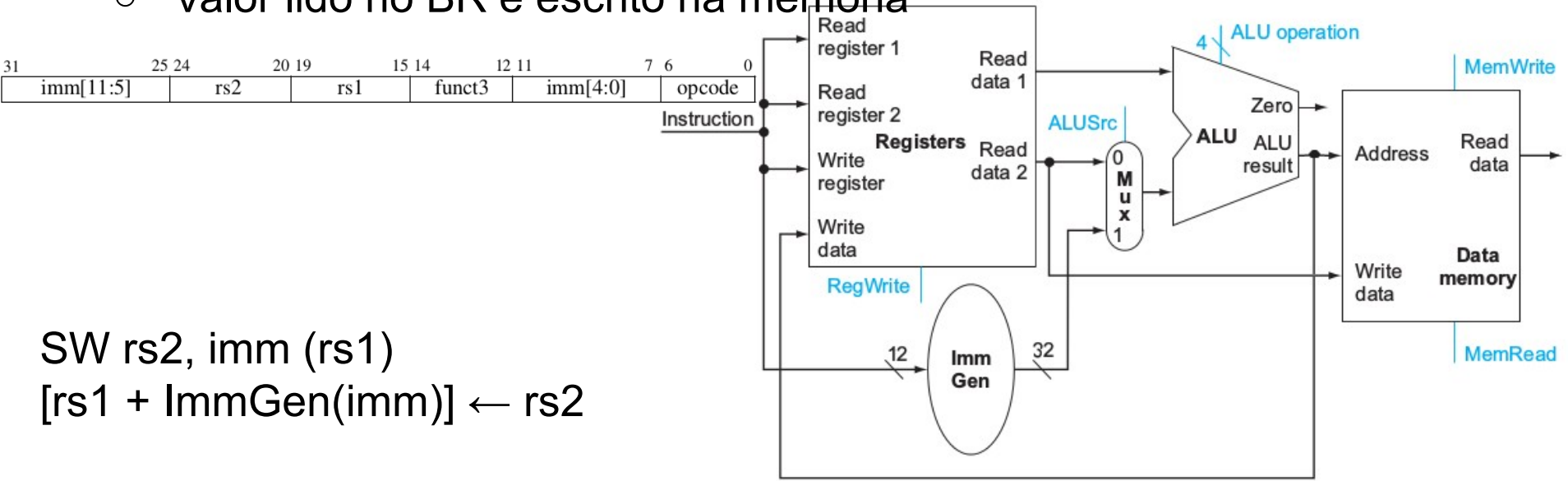






Instruções de acesso a memória: SW

- **Caminho de dados para R-Format + I-Format + SW**
  - Endereço de acesso é dado pela soma do registrador base (rs1) com a extensão de sinal (saída ImmGen)
  - Valor lido no BR é escrito na memória



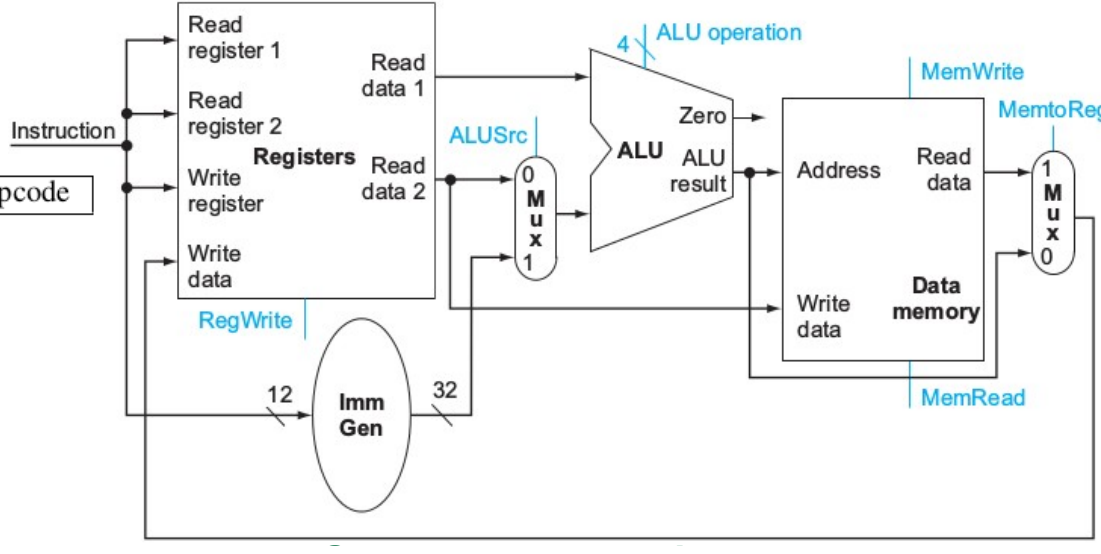
SW rs2, imm (rs1)
[rs1 + ImmGen(imm)] ← rs2

Instruções de acesso a memória: LW

- **Caminho de dados para R-Format + I-Format + SW + LW**
  - Endereço de acesso é dado pela soma do registrador base (rs1) com o deslocamento (saída ImmGen)
  - Valor lido na memória é escrito no BR conforme rd
  - MUX inserido na entrada *write data*

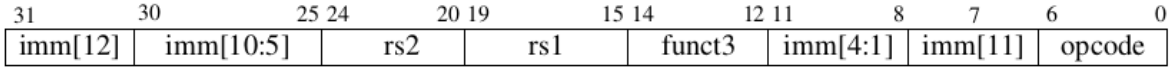
imm[11:0]	rs1	funct3	rd	opcode

LW rd, imm (rs1)
rd ← [rs1 + ImmGen(imm)]



Instruções de desvio B-Format : BEQ

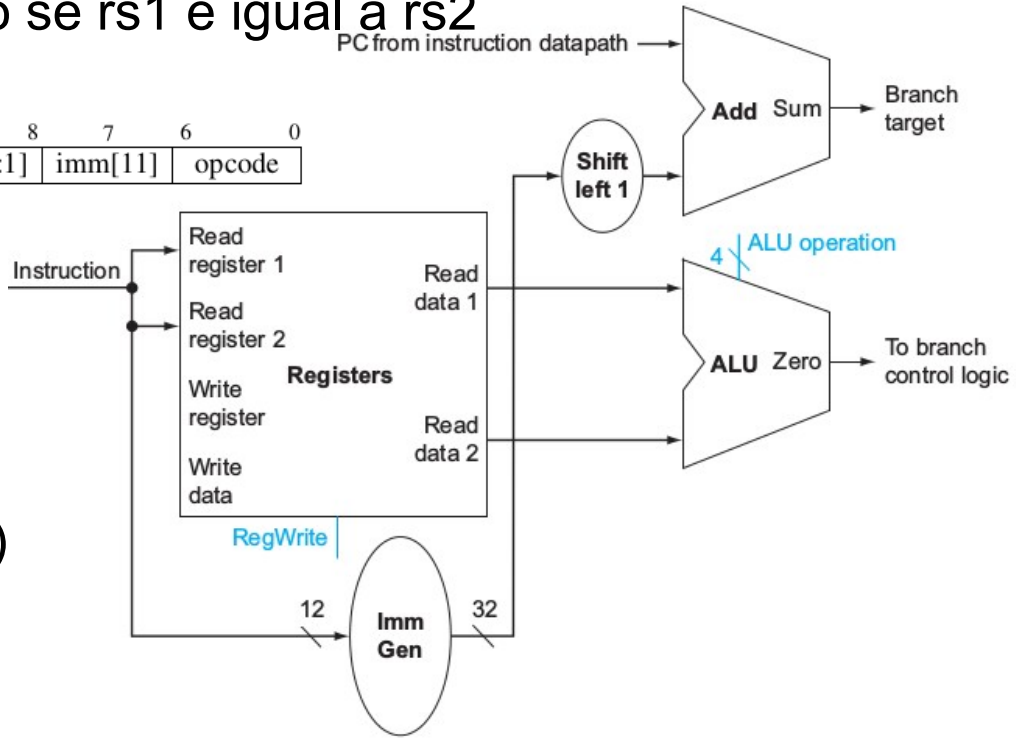
- **Caminho de dados para Branch (beq)**
  - Novo PC é calculado pela soma de PC atual com imm (deslocado)
  - PC calculado só é transferido se rs1 é igual a rs2



BEQ rs1, rs2, imm

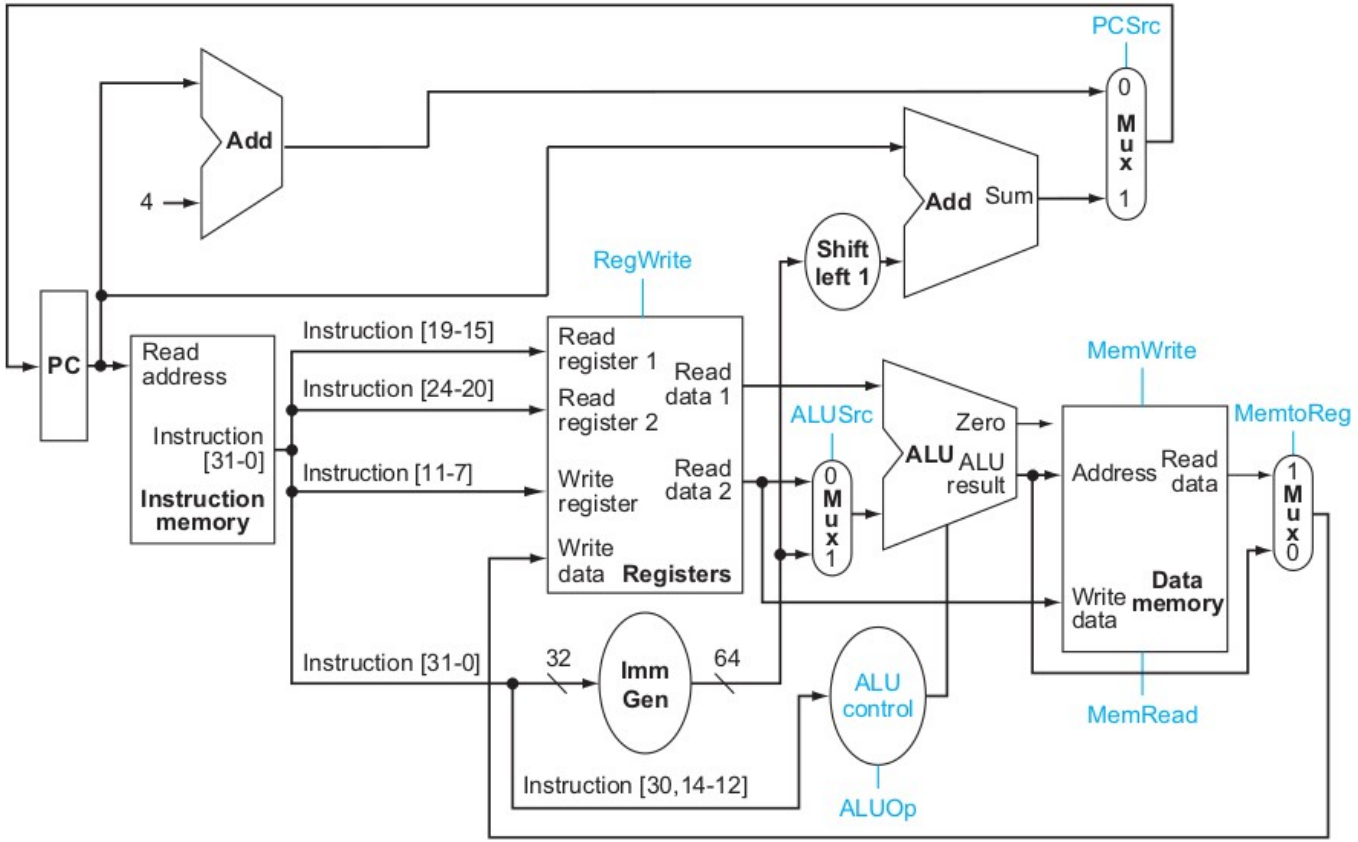
if (rs1 = rs2)

PC ← PC + (ImmGen(imm) << 1)



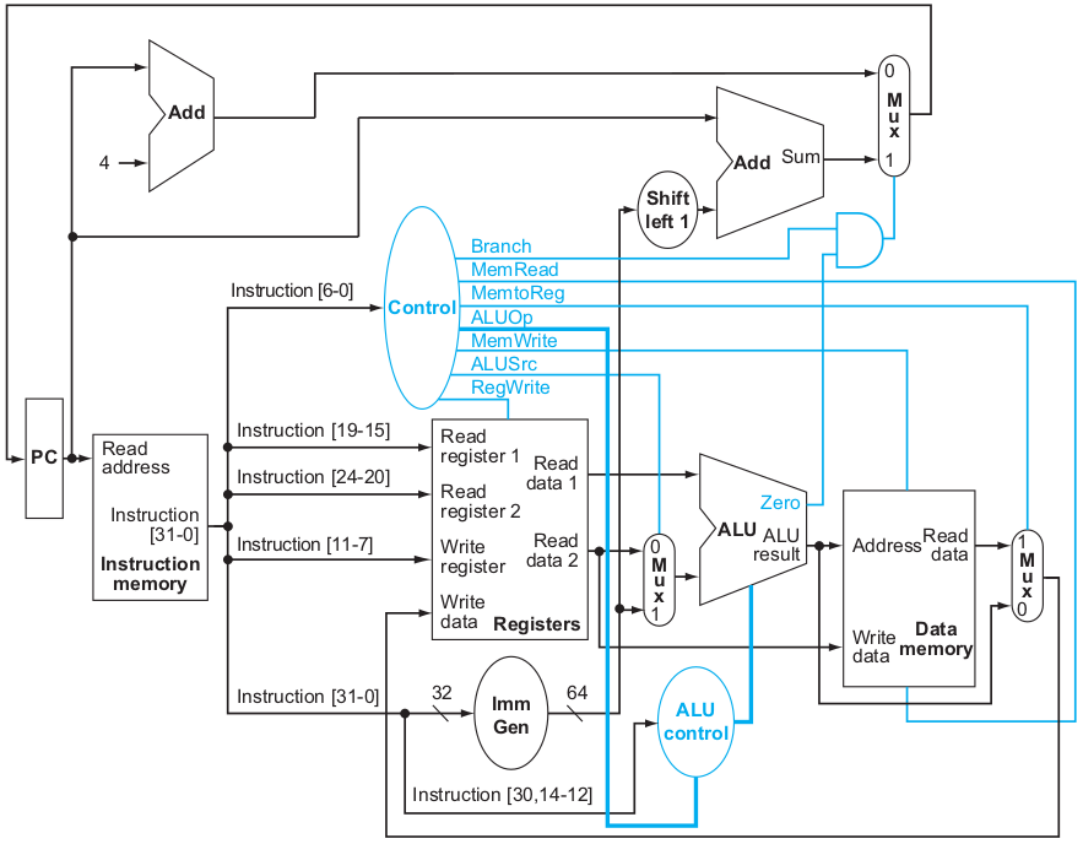
Bloco operativo monociclo completo

- **Implementação monociclo básica**



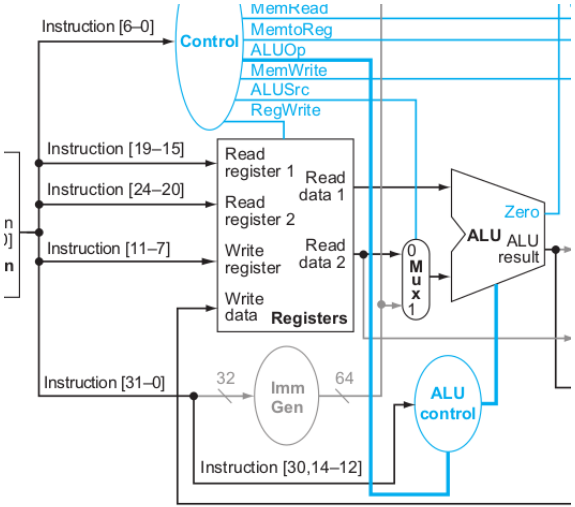
Bloco operativo monociclo completo

- **Implementação monociclo básica**



Controle monociclo

- **Controle da ULA**

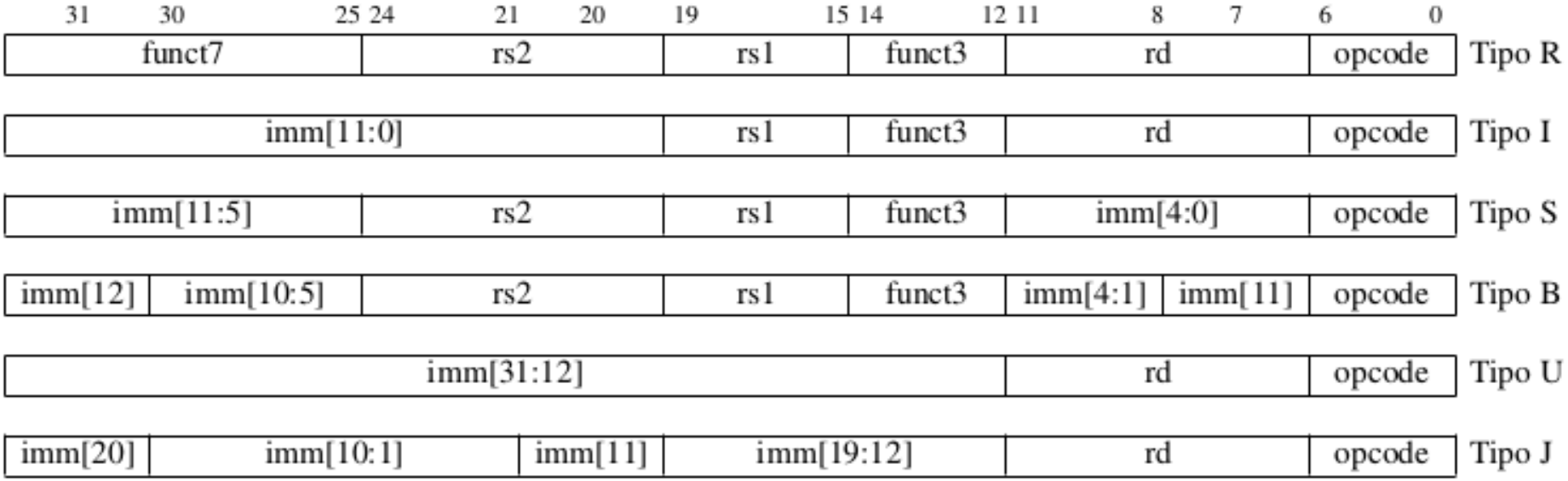


ALU control lines	Function
0000	AND
0001	OR
0010	add
0110	subtract

Instruction opcode	ALUOp	Operation	Funct7 field	Funct3 field	Desired ALU action	ALU control input
ld	00	load doubleword	XXXXXXX	XXX	add	0010
sd	00	store doubleword	XXXXXXX	XXX	add	0010
beq	01	branch if equal	XXXXXXX	XXX	subtract	0110
R-type	10	add	0000000	000	add	0010
R-type	10	sub	0100000	000	subtract	0110
R-type	10	and	0000000	111	AND	0000
R-type	10	or	0000000	110	OR	0001

Controle monociclo

- **Formatos das instruções do RISC-V**



Controle monociclo

- **Unidade de controle**

Instruction	ALUSrc	Memto-Reg	Reg-Write	Mem-Read	Mem-Write	Branch	ALUOp1	ALUOp0
R-format								
ld								
sd								
beq								

Controle monociclo

- **Unidade de controle**

Instruction	ALUSrc	Memto-Reg	Reg-Write	Mem-Read	Mem-Write	Branch	ALUOp1	ALUOp0
R-format	0	0	1	0	0	0	1	0
ld	1	1	1	1	0	0	0	0
sd	1	X	0	0	1	0	0	0
beq	0	X	0	0	0	1	0	1