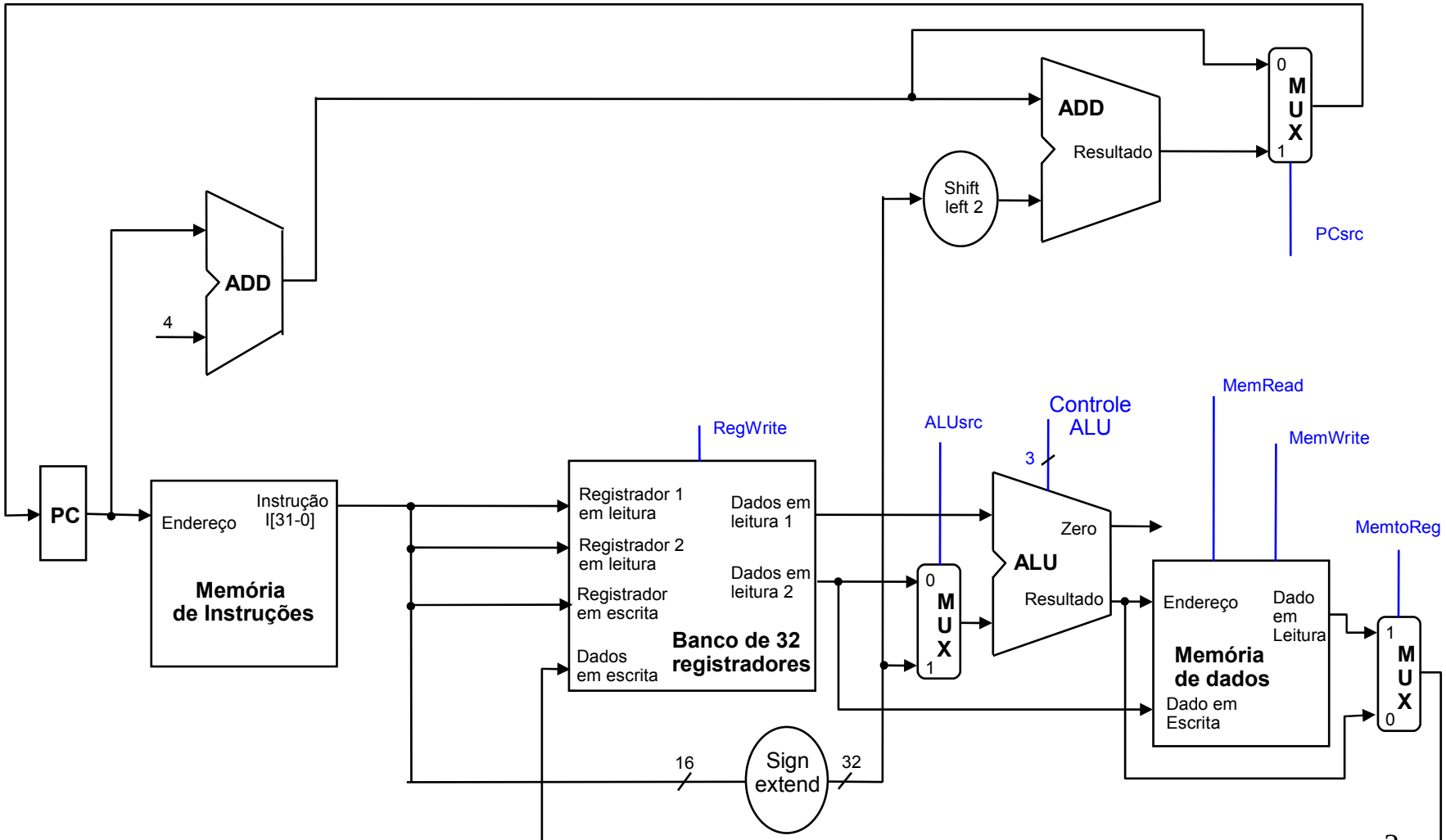
Universidade Federal do Rio Grande do Sul
Instituto de Informática

Organização de Computadores

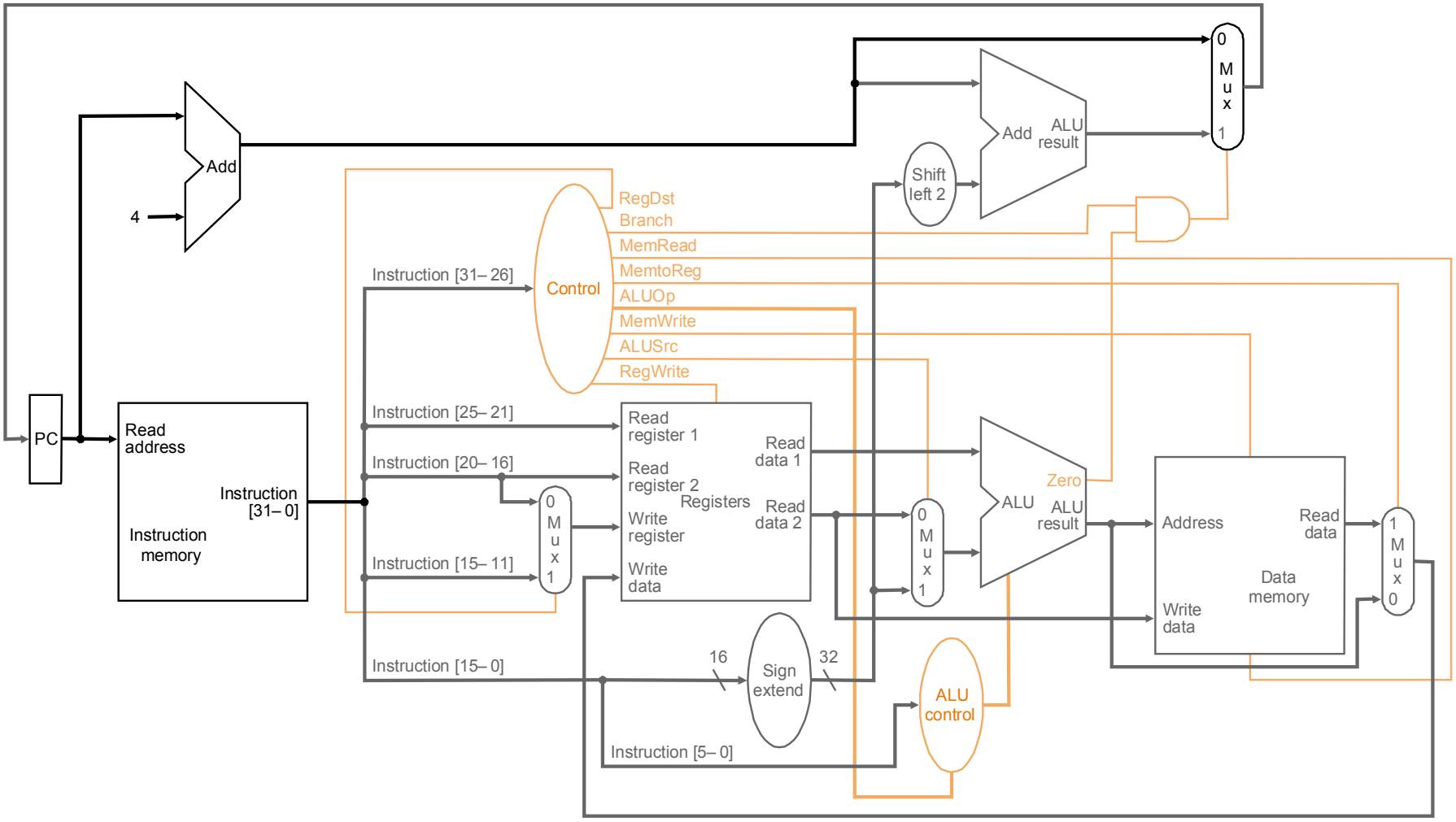
Aula 5

Bloco de controle mono-ciclo

Visão geral aula passada
Bloco Operacional Completo



Aula de hoje
Acrescentar o Controle



Adicionando Controle

- **CPU = Caminho de Dados + Controle**

- **Projeto Mono Ciclo :**
  - **Instruções levam exatamente um ciclo de relógio,**
  - **Caminhos de Dados usam somente um ciclo,**
  - **A atualização do estado de armazenamento é executada ao fim do ciclo,**

- **O que deve ser controlado ?**
  - **ALU (quais operações)**
  - **Multiplexadores**
  - **Elementos de Estado de armazenamento: Registradores, Memórias de Dados,**

Processador = Caminho de Dados + Controle

- **Projeto de Mono Ciclo: tudo deve acontecer em um ciclo de relógio**

  ⇒

  até a próxima descida do relógio,

- **O processador deve ter um BIG circuito Combinatório!!!!**

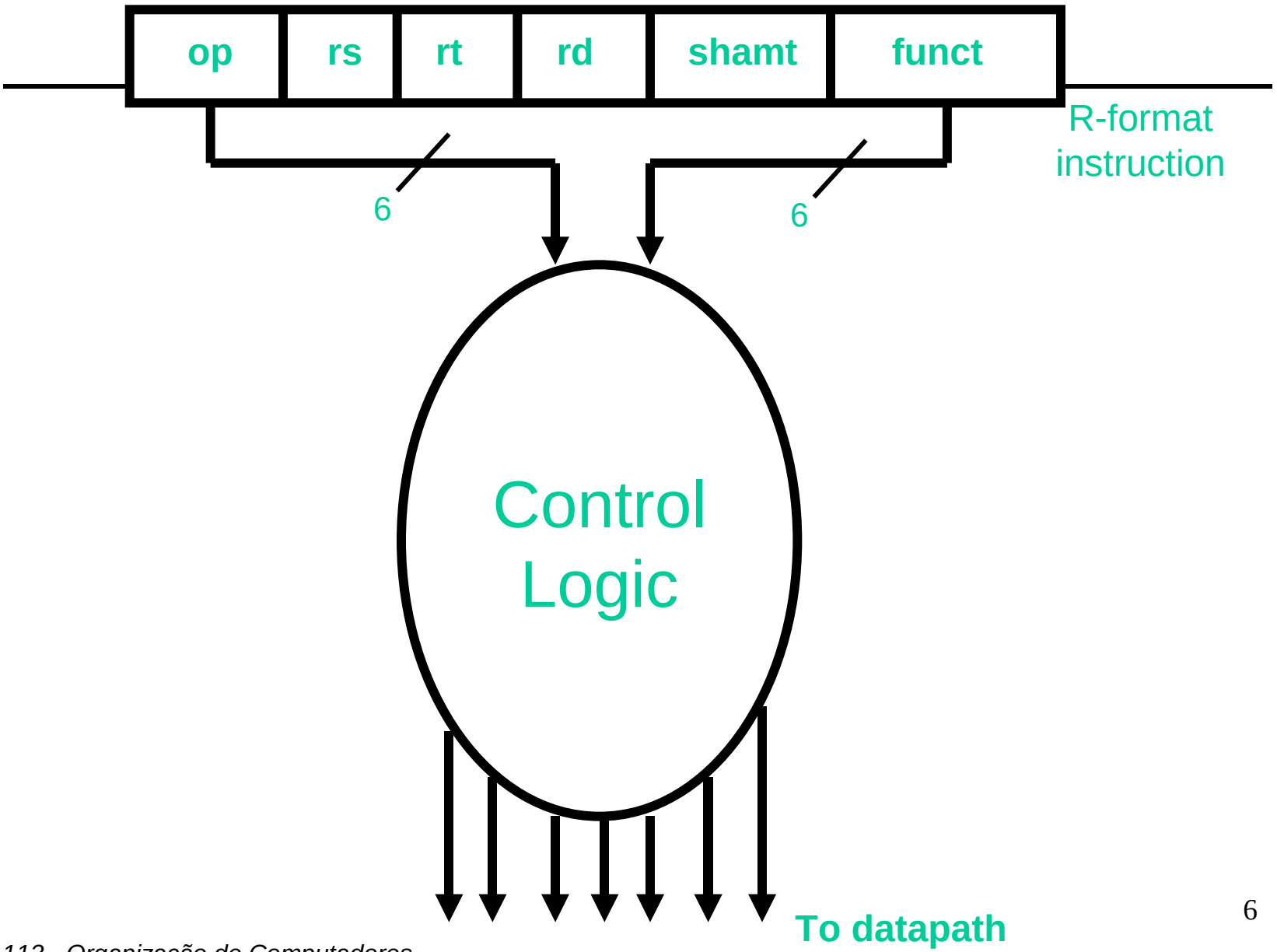
  ⇒

  controle é apenas um circuito combinacional,

- **As saidas são funções de entradas**
  - **Saidas? Pontos de controle do caminho de dados**
  - **Entradas? A instrução corrente!**

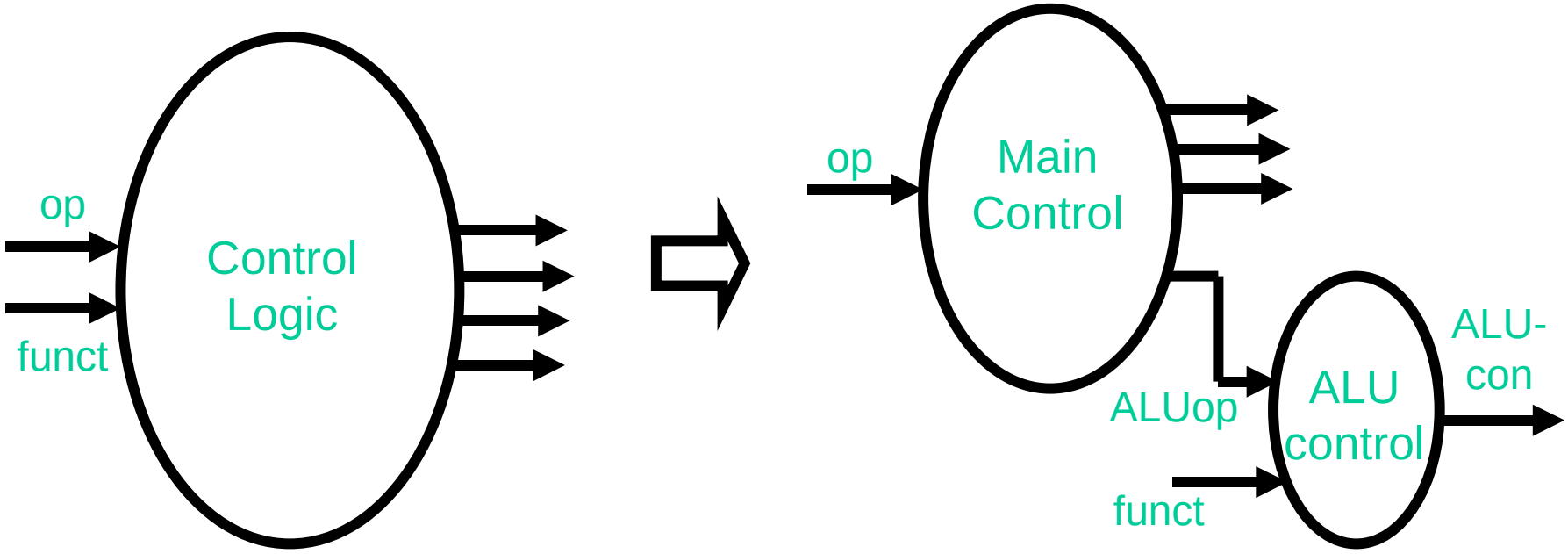
    (`opcode`, `funct` controlam tudo)

Definindo Controle



Definindo o Controle
controle funções ALU

- **Observem que o campo “funct” esta presente apenas no formato de instruções R - “funct” controla somente a ALU ,**
- **Para simplificar o controle, define-se a Main, uma controladora de ALU separada – o uso de múltiplos níveis também incrementará a velocidade – importante técnica de otimização**
- **Entradas ALUop serão definidas**



Bloco de controle mono-ciclo

1. **MIPS mono-ciclo: sinais de controle**

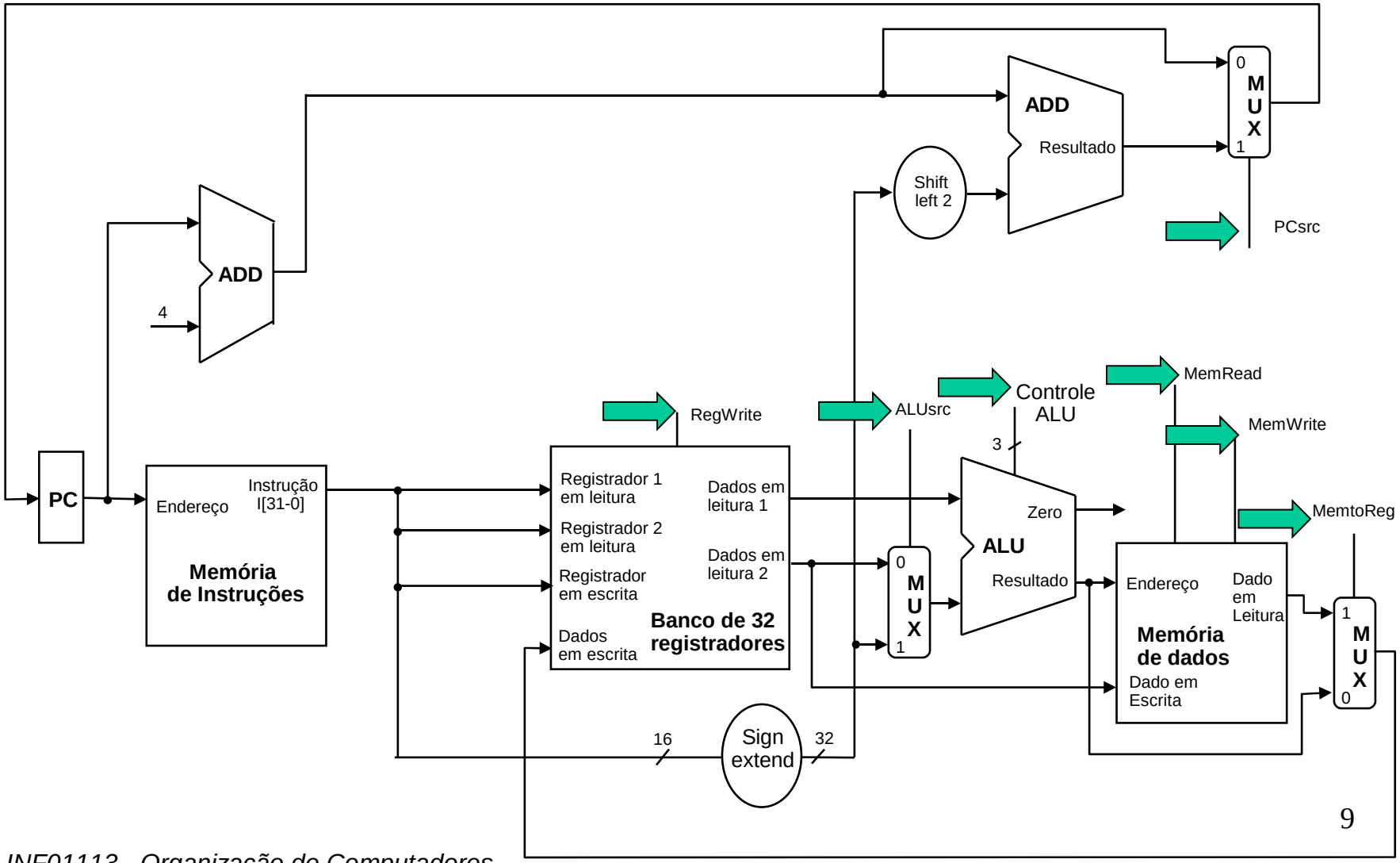
2. **Execução das instruções**
   - **Instruções aritméticas e lógicas (formato-R)**
   - **Instruções “load word”**
   - **Instruções “store word”**
   - **Instruções “branch-on-equal”**

3. **Sumário dos sinais de controle**

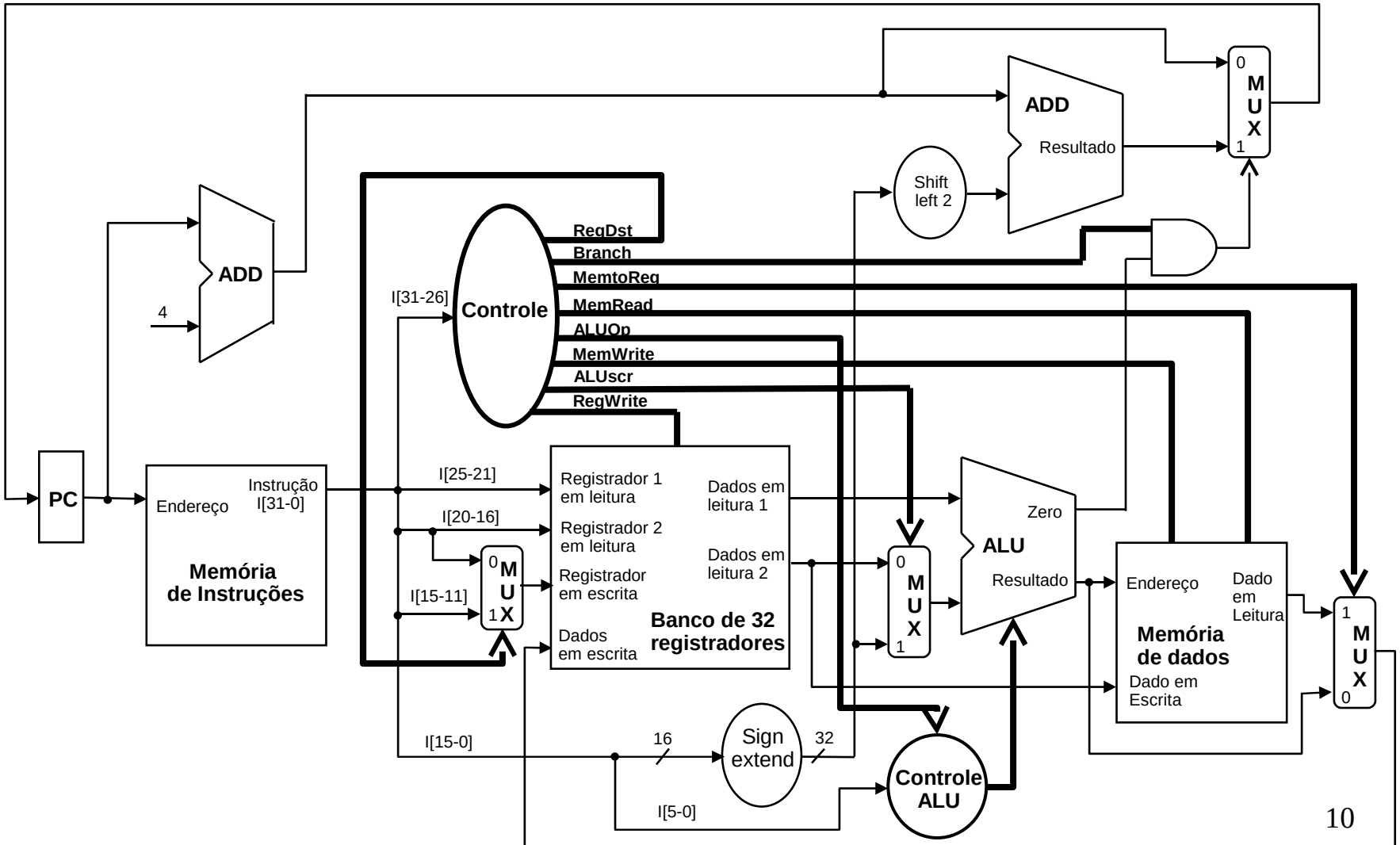
4. **Projeto lógico do controle da ULA**

5. **Projeto lógico do bloco de controle**

1. Sinais de Controle p/ MIPS Mono-Ciclo



MIPS mono-ciclo: sinais de controle



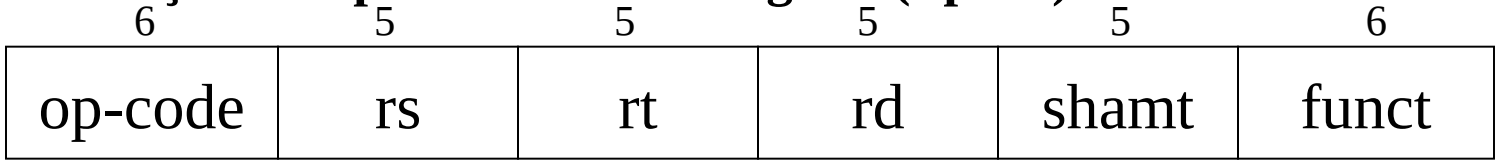
Efeito das 7 linhas de Controle

Nome do sinal	Efeito quando ativo	Efeito quando inativo
RegDst	O número do registrador-destino onde será escrito o resultado da operação (Reg a ser Escrito) vem do campo rt (bits 20–16).	O número do registrador-destino onde será escrito o resultado da operação (Reg a ser Escrito) vem do campo rt (bits 15–11).
EscReg	Nenhum	O registrador na entrada Reg a ser Escrito é escrito com o valor presente na entrada Dado de Escrita.
UALFonte	O segundo operando da UAL vem do segundo registrador do banco de registradores (Dado lido #2).	O segundo operando da UAL é o resultado da extensão de sinal dos 16 bits menos significativos da instrução.
FontePC	O PC é substituído pelo valor presente na saída do somador que calcula PC + 4.	O PC é substituído pelo valor presente na saída do somador que calcula o endereço-alvo do desvio condicional.
LerMem	Nenhum	O conteúdo da memória designado pela entrada de endereço é colocado na saída Dado Lido.
EscMem	Nenhum	O conteúdo da memória designado pela entrada de endereço é substituído pelo valor presente na entrada Dado a ser Escrito.
MemParaReg	O valor na entrada do registrador de escrita vem da UAL.	O valor na entrada do registrador de escrita vem da memória de dados.

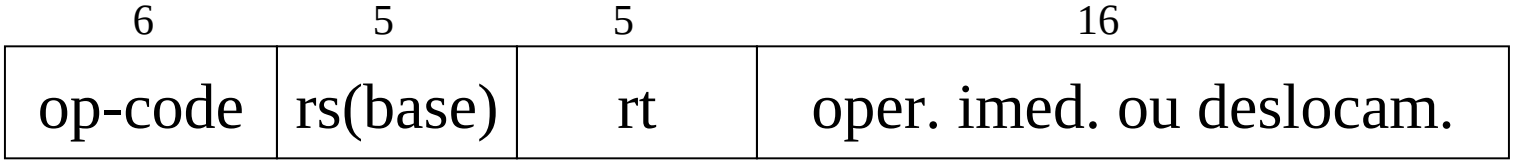
Fonte PC > Branch, EscReg > RegWrite, UALFonte > ALUscr

Classes de Instruções

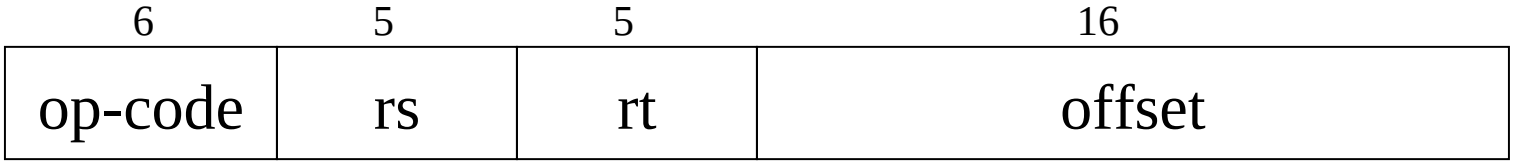
- **Instrução do tipo Aritméticas Lógicas (tipo R)**



- **Instrução do tipo Load Word ou Store Word**



- **Instrução do tipo Desvio Condicional**



2. Execução das instruções
Instruções aritméticas e lógicas (formato-R)

- **Quatro passos add $t1, $t2, $t3 :**

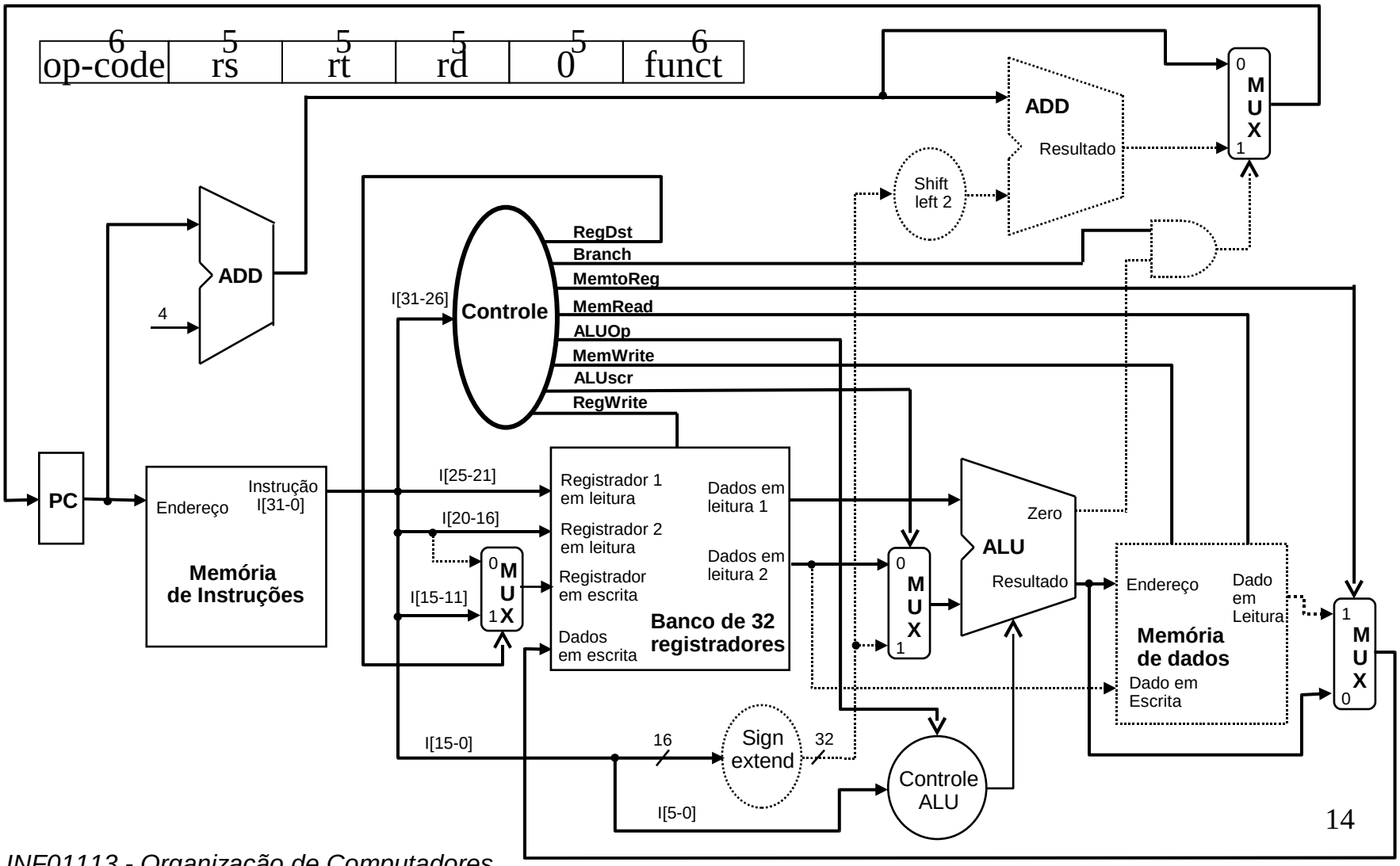
1. **Busca da instrução na memória de instruções e incrementa PC,**

3. **Dois registradores são lidos do banco de registradores**

5. **A UAL opera os dados lidos a partir do banco de registradores, usando o código da função vindo do funct,**

7. **O resultado da UAL é escrito no banco de registradores**

Instrução tipo R aritmética



Instrução do tipo Load Word

- **Cinco passos lw $t1, deslocamento($t2) :**

  2. **Busca da instrução na memória de instruções e incrementa PC,**

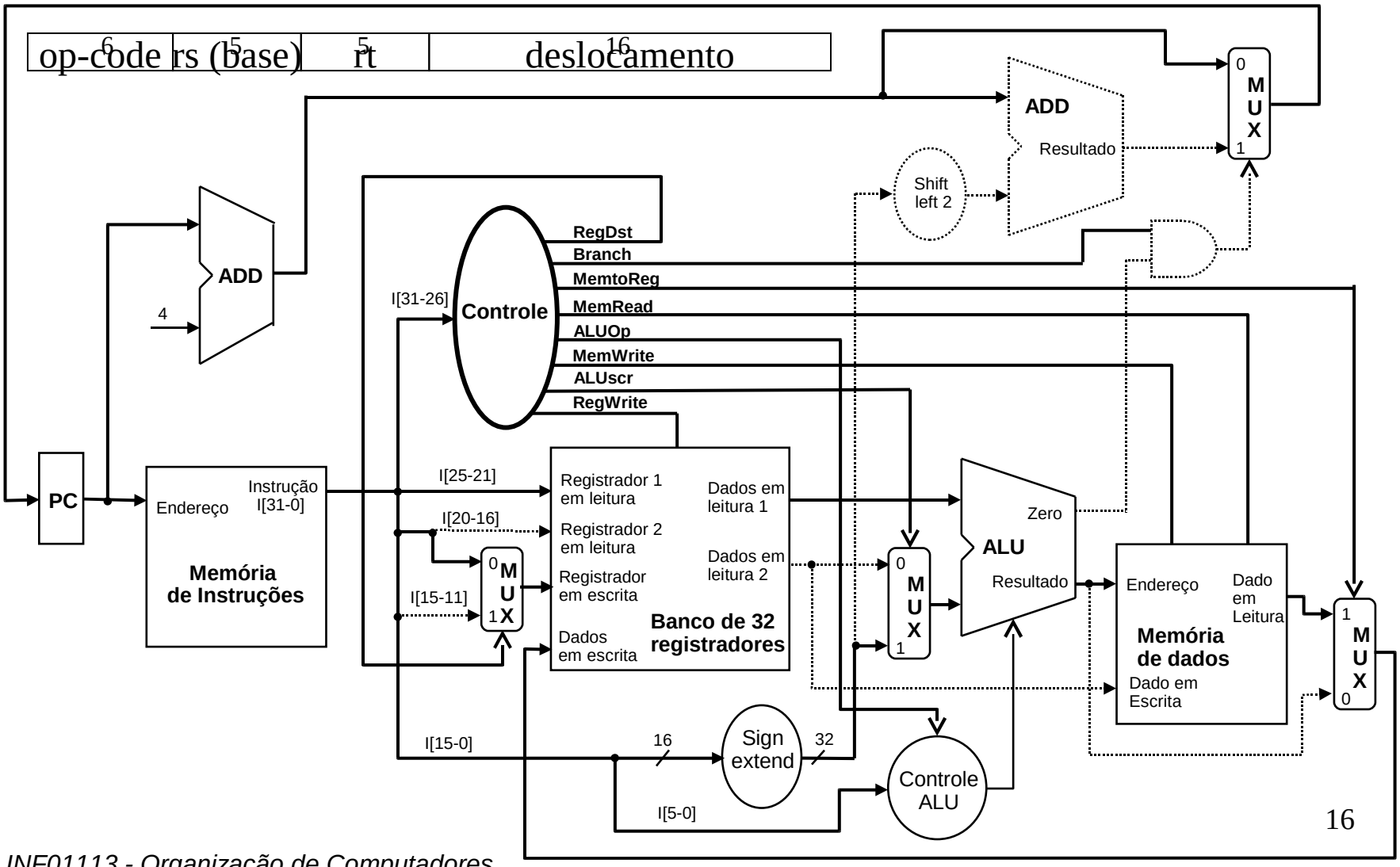
  4. **Leitura do conteúdo de um registrador ($t2) do banco de registradores,**

  6. **Cálculo da soma do valor lido do banco de registradores com o resultado da extensão do sinal dos 16 bits menos significativos da instrução (deslocamento),**

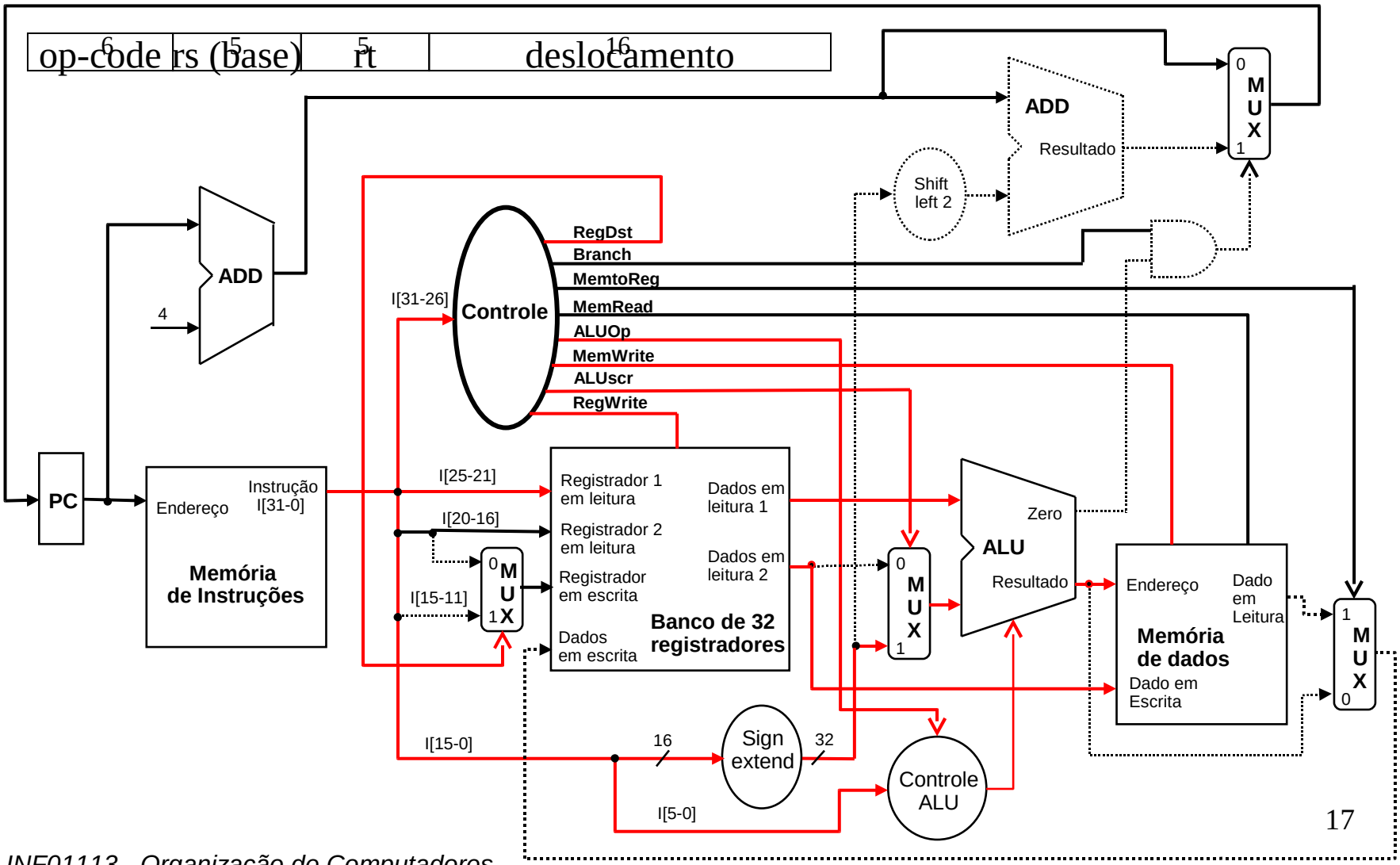
  8. **O resultado da soma é usado para endereçar a memória de dados,**

  10. **O dado vindo da unidade de memória é escrito no banco de registradores; o número do registrador-destino é dado pelos bits 20-16 ($t1).**

Instruções “load word”



Instruções “store word”



Instrução Branch beq

- **Quatro passos de beq $t1, $t2, deslocamento:**

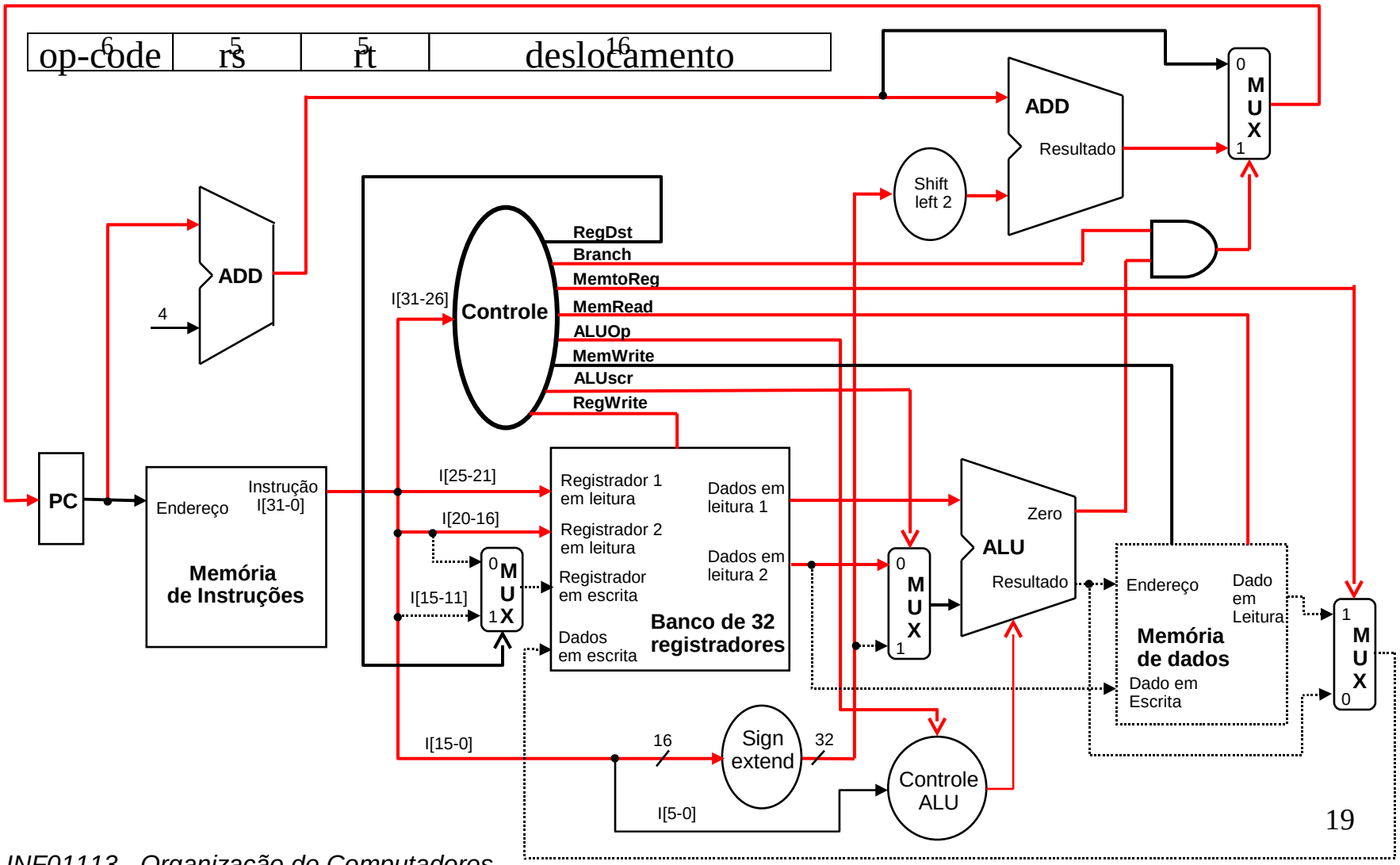
1. **Busca da instrução na memória de instruções e incrementa PC,**

3. **Leitura do conteúdo dos registradores $t1 e $t2 do banco de registradores,**

5. **Realização de uma subtração pela UAL, sobre os dois valores lidos do banco de registradores. O valor de PC +4 é somado ao resultado da extensão do sinal dos 16 bits menos significativos da instrução (deslocamento) deslocado de dois bits à esquerda. O resultado desta soma é o endereço-alvo do desvio,**

7. **A saída Resultado_Zero da UAL é usada para decidir se o PC deve ser atualizado com o valor de PC+4 ou com o valor do endereço-alvo do desvio condicional.**

Instruções “branch-on-equal”



3. Sumário dos sinais de controle

	RegDst	ALUSrc	Memto-Reg	Reg Write	Mem Read	Mem Write	Branch	ALUOp1	ALUOp2
tipo R	1	0	0	1	0	0	0	1	0
lw	0	1	1	1	1	0	0	0	0
sw	X	1	X	0	0	1	0	0	0
beq	X	0	X	0	0	0	1	0	1

4. Projeto lógico do controle da ALU

- **A ULA executa cinco funções aritméticas lógicas**

Entrada de controle da ULA	Função
000	AND
001	OR
010	SOMA
110	SUBTRAÇÃO
111	SET LESS THAN

- **Nas instruções de load word e store word é que a ULA calcula o endereço de memória**
- **Para a instrução de beq a ULA executa uma subtração.**

Controle da ALU

Bits de controle da ALU em função de ALUOp e dos códigos de função

op-code	ALUOp	operação	campo "function"	ação na ALU	controle da ALU
lw	00	load word	XXXXXX	add	010
sw	00	store word	XXXXXX	add	010
beq	01	branch equal	XXXXXX	subtract	110
R	10	add	1 0 0 0 0 0	add	010
R	10	subtract	1 0 0 0 1 0	subtract	110
R	10	and	1 0 0 1 0 0	and	000
R	10	or	1 0 0 1 0 1	or	001
R	10	set-on-less-than	1 0 1 0 1 0	set-on-less-than	111

Controle da ALU

Para obter a tabela-verdade dos bits de controle da ALU

ALUOp		Function						Operation	Operação
ALUOp1	ALUOp2	F5	F4	F3	F2	F1	F0		
0	0	X	X	X	X	X	X	010	LW,SW Soma
X	1	X	X	X	X	X	X	110	BEQ Sub
1	X	X	X	0	0	0	0	010	Soma
1	X	X	X	0	0	1	0	110	Subtr
1	X	X	X	0	1	0	0	000	And
1	X	X	X	0	1	0	1	001	Or
1	X	X	X	1	0	1	0	111	Set less than

Implementação com lógica aleatória ou PLA é trivial

5. Projeto lógico do bloco de controle

Op-codes

instrução	op-code decimal	op-code binário
R	0	0 0 0 0 0 0
lw	35	1 0 0 0 1 1
sw	43	1 0 1 0 1 1
beq	4	0 0 0 1 0 0

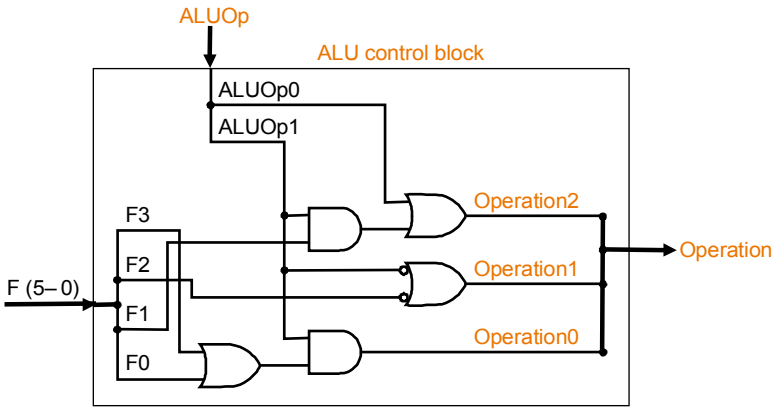
Implementação com lógica aleatória ou PLA é trivial

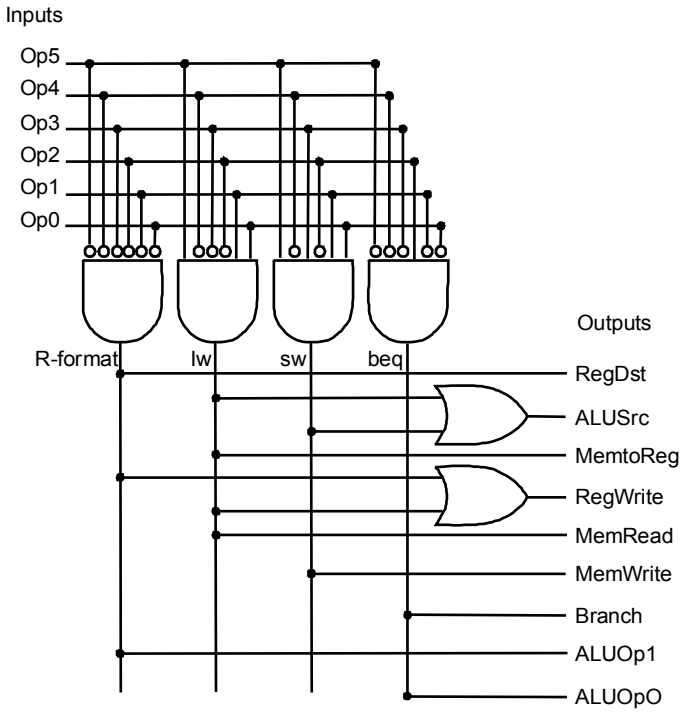
Tabela-verdade dos sinais de controle em função do op-code

		R	lw	sw	beq
Entradas	Op5	0	1	1	0
	Op4	0	0	0	0
	Op3	0	0	1	0
	Op2	0	0	0	1
	Op1	0	1	1	0
	Op0	0	1	1	0
Saídas	RegDst	1	0	X	X
	ALUSrc	0	1	1	0
	MemtoReg	0	1	X	X
	RegWrite	1	1	0	0
	MemRead	0	1	0	0
	MemWrite	0	0	1	0
	Branch	0	0	0	1
	ALUOp1	1	0	0	0
	ALUOp0	0	0	0	1

Controle

Lógica combinatória (tabela verdade).





add: op = 0d = 000000b
add: function = 32d = 100000b
lw: op = 35d = 100011b
lw: function = don't care = X

Ineficiência do Projeto Mono-Ciclo

- **Apesar de todas instruções serem executadas num ciclo de relógio e funcionar corretamente, é muito ineficiente,**
- **Uma das razões é que o ciclo do relógio será determinado pela instrução com tempo de execução maior. Provavelmente a de “load word” que emprega cinco unidades funcionais,**
- **A penalidade maior aparece na execução de uma instrução de ponto flutuante ou de instruções mais complexas.**
- **Além disto numa implementação mono-ciclo, cada unidade funcional só pode ser usada uma única vez em cada ciclo de relógio.**
- **Uma solução seria duplicar algumas unidades funcionais, mas isto aumentaria o custo de implementação.**

Implementação em um único ciclo

Classe	Memória de Instruções	Leitura Registrador	Operação UAL	Memória de Dados	Escrita no registrador	Total
Formato R	2	1	2 (16)	0	1	6 ns
Load Word	2	1	2	2	1	8 ns
Store Word	2	1	2	2		7 ns
Branch	2	1	2			5 ns
Jump	2					2 ns
Formato R sem e com ponto flutuante						

- **O ciclo de *clock* é definido em função da duração da instrução mais longa = 8 ns.**
- **Uma máquina com ciclo de *clock* variável: 2ns a 8ns.**
  - **Ciclo de *clock*: 8x24% + 7x12% + 6x44% + 5x18% + 2x2% = 6,3 ns**
- **Ganho de desempenho:**

  Tempo execução *clock* fixo / Tempo de execução *clock* variável = 8 / 6,3 = 1,27
- **No caso de operações em ponto flutuante podemos ter para a multiplicação:**

  2 + 1 + 16 + 1 = 20 ns
- **A mesma relação pode ser feita para operações em ponto flutuante e o ganho favorável ao *clock* variável pode ser ainda maior.**

Próxima aula ver solução com mais de um ciclo

FIM

Perguntas/Exercícios:

- **O controle atrasa o circuito? A freqüência máxima fica comprometida?**
- **Modifique o bloco operacional e de controle para incluir a instrução LUI. Houve impacto no ciclo de relógio?**
- **Modifique o bloco operacional e o controle para incluir a instrução que acessa a memória com pós-incremento:**
  - **lw $rs, M($rt)**
  - **addi $rt, $rt, 1**
- **Houve impacto no ciclo de relógio? Vale a pena fazer esta modificação no caso de ciclo único?**
- **É possível eliminar o sinal MemtoReg e substituí-lo por ALUSrc?**