



MC 613

IC/Unicamp

Prof Guido Araújo

Prof Mario Côrtes

Conceitos: Via de Dados e Processadores

m1ps: meu primeiro proc simples
(minúsculo MIPS)



Tópicos

- Objetivos e especificações
- Diagrama de bloco
- Módulos e componentes
- ISA – Instruction Set Architecture – Conjunto de Instruções
- Execução das instruções: ciclo a ciclo
- Unidade de controle e FSM
- Montador
- HW de suporte para depuração (telas)
- Extensões no ISA



Objetivos do m1ps

- Primeira exposição à organização de processadores
- Conceitos principais, sem ser exaustivo
- Simples, mas não mínimo (pouca complexidade)
- Modular, intelegível, intuitivo, apreensível
- (quase)Completo: possível de implementar códigos básicos
- Extensível
- Uso de algumas estruturas iguais ou próximas ao MIPS

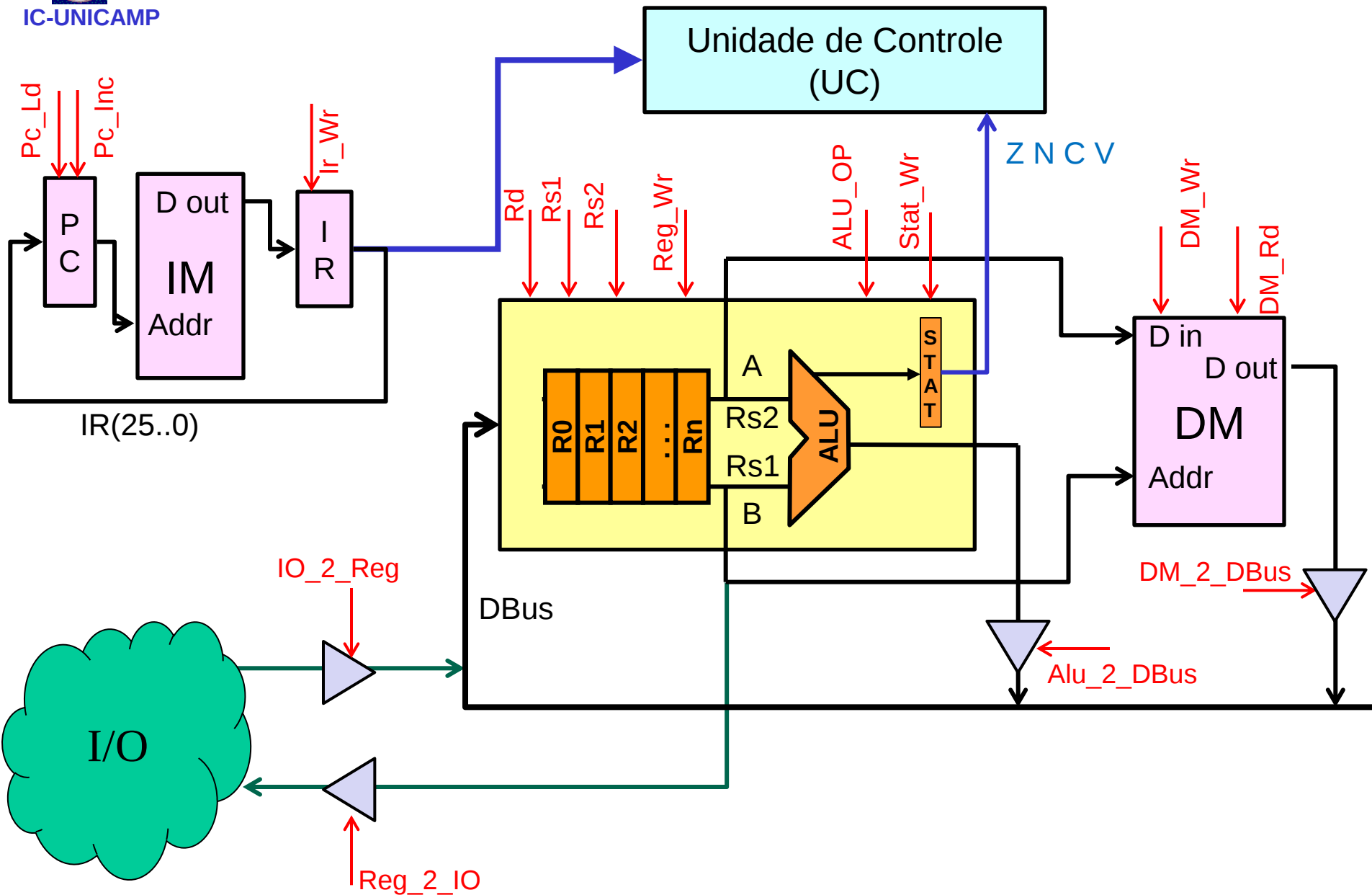


Specs de implementação

- Dados e instruções de 32 bits
- Endereço de dados e instruções: palavras
- Banco de registradores = MIPS
- ALU: quase igual à do MIPS
- Registrador de status/condição (Z, C, N, V)
- Desvio condicional (status) e incondicional com endereço imediato completo (simplicidade)
- Formatos de instrução iguais ao MIPS



Diagrama de blocos

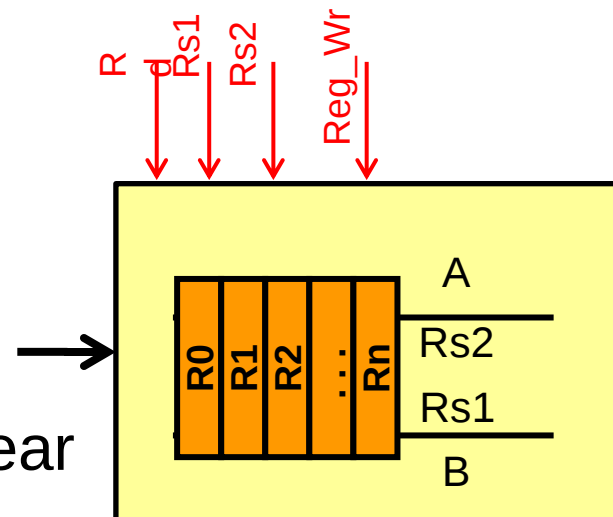


Módulos: banco de registradores



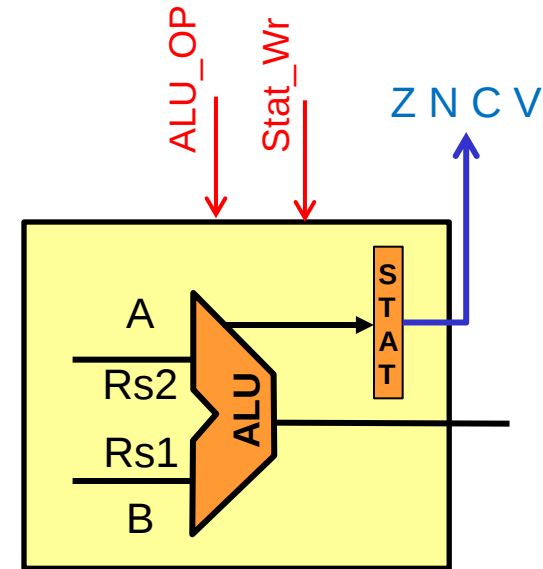
IC-UNICAMP

- Parte da via de dados
- 32 registradores de 32 bits
 - ATENÇÃO: R0 = 0
 - permite pseudo instruções move e clear
- Dados
 - Entrada: Barramento Dbus (32 bits)
 - Saídas: Barramentos A e B (32 bits)
- Controle
 - Rs1 e Rs2 (5bits): selecionam registradores → saídas A e B
 - Rd (5bits): seleciona registrador a ser escrito
 - Reg_Wr (1b): controle de escrita



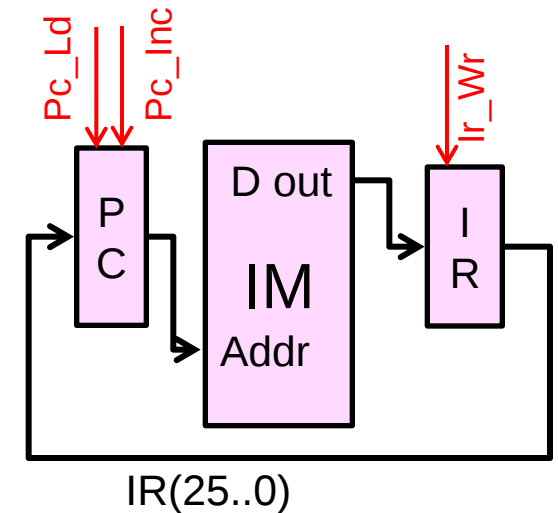
Módulos: ALU

- Parte da via de dados
- Operações lógicas e aritméticas de operandos de 32 bits: add, sub, and, or
- Dados
 - Entradas: Barramentos A e B (32 bits)
 - Saídas: ALU_out (32 bits)
- Controle
 - AluOp: define operação da ALU (ver conj de instruções)
 - Z,C,V,N: bits de status da operação



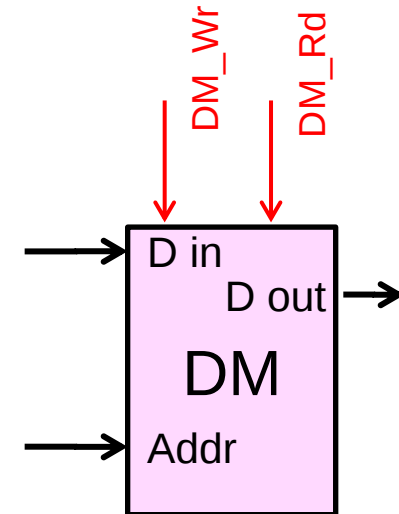
Módulos: sistema de memória de instruções (IM)

- IM (somente leitura): 2^{26} linhas de 32b de largura
 - Dados: leitura da instrução (32 bits) $\rightarrow$ IR
 - Endereço: PC (26 bits)
 - Controle: leitura sempre
- PC
 - Dados: incremento ou carga paralela
 - Controle:
 - PC-Inc: $PC \leftarrow PC + 1$
 - PC-Ld: $PC \leftarrow \text{Target Address}$
 - carga paralela de endereço de desvio (26 bits)
- Observações:
 - cuidado com a temporização
 - limitar tamanho na implementação DE1 ($< 2^{26}$ linhas)



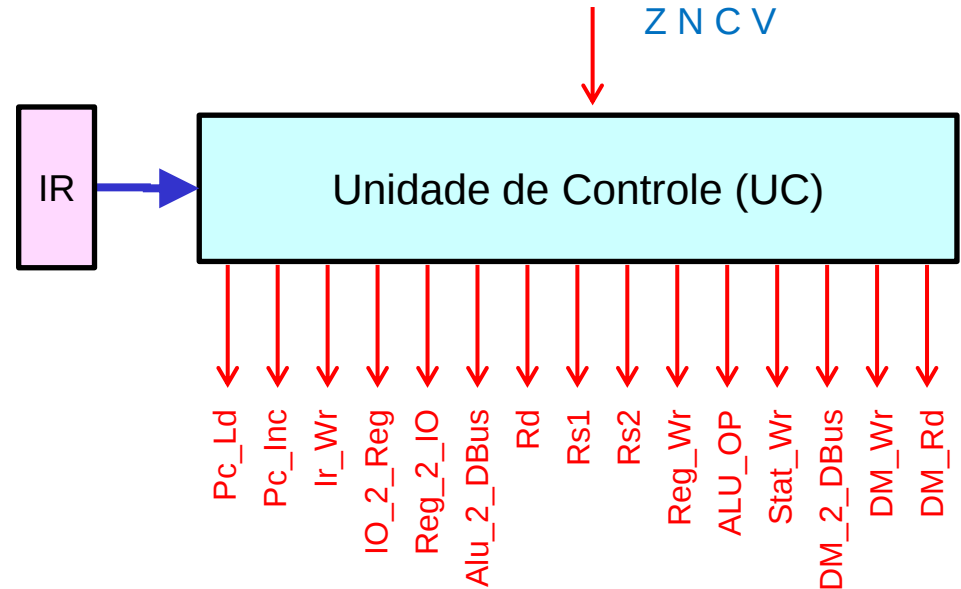
Módulos: sistema de memória de dados (DM)

- DM: 2^{32} linhas de 32b de largura
 - sem Regs dedicados para dados e endereço
- Dados:
 - saída Dout (32b) é um dos sinais a acionar o DBus
 - entrada Din (32b): saída A do banco de registradores (definido por Rs2)
- Endereço: vem de Rs1
 - entrada Addr (32b): saída B do banco de registradores (definido por Rs1)
- Controles (1b): DM_Wr e DM_Rd
- Observações:
 - cuidado com a temporização
 - limitar tamanho na implementação DE1 ($< 2^{32}$ linhas)



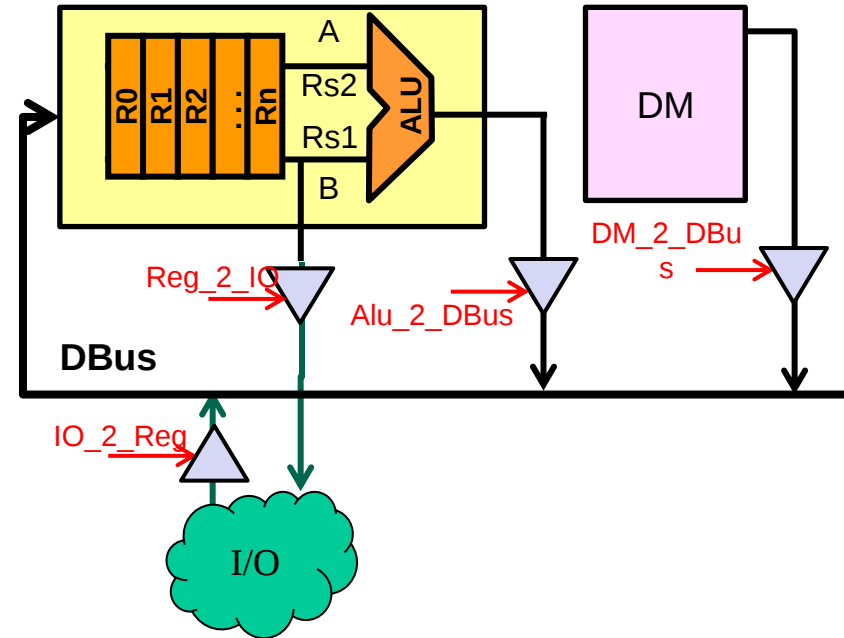
Módulos: Unidade de controle

- Interfaces
 - Entradas: IR e Status
 - Saídas: 15 sinais de controle
- Estrutura
 - Principal: FSM
 - Apoio:
 - lógica para controle de desvio
 - lógica para controle da ALU



Módulos: DBus e I/O

- I/O
 - In: I/O → Reg, via DBus
 - Out: Reg → I/O
- DBus
 - 32 bits
 - Acionam o barramento:
 - DM, saída da ALU, I/O in
 - Leem do barramento
 - Banco d registradores





ISA: código de máquina

Instruções Formato R, lógicas e aritméticas

	Op(31..26)	Rd(25..21)	Rs1(20..16)	Rs2(15..11)	Unused(10..0)	Obs
add Rd, Rs1, Rs2	001 000				-	$Rd \leftarrow Rs1 + Rs2$
sub Rd, Rs1, Rs2	001 001				-	$Rd \leftarrow Rs1 - Rs2$
and Rd, Rs1, Rs2	001 010				-	$Rd \leftarrow Rs1 \text{ And } Rs2$
or Rd, Rs1, Rs2	001 011				-	$Rd \leftarrow Rs1 \text{ Or } Rs2$
	6	5	5	5	5	

Instruções Formato R, transf de dados

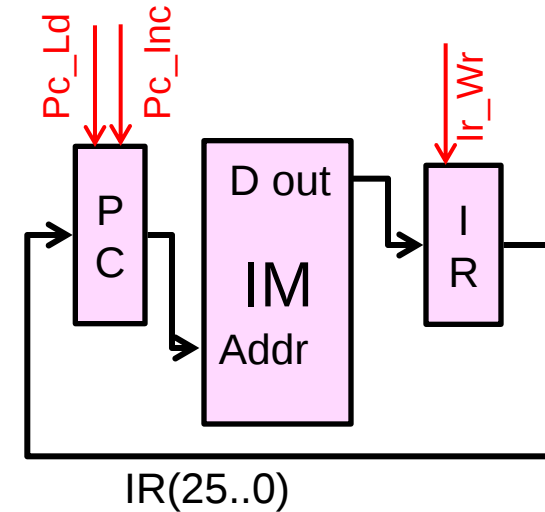
	Op(31..26)	Rd(25..21)	Rs1(20..16)	Rs2(15..11)	Unused(10..0)	Obs
lw Rd, Rs1	000 111			-	-	$Rd \leftarrow DM(Rs1)$
sw Rs1, Rs2	010 111	-			-	$DM(Rs1) \leftarrow Rs2$
in Rd	100 000		-	-	-	$Rd \leftarrow IO \text{ data in}$
out Rs1	110 000	-		-	-	$IO \text{ data out} \leftarrow Rs1$
	6	5	5	5	5	

Instruções Formato J, desvio

	Op(31..26)	Addr(25..0)	Obs
J addr	111 111		$Pc \leftarrow \text{addr}$
BrZ	111 000		" if Z=1
BrN	111 001		" if N=1
BrV	111 010		" if V=1
BrC	111 011		" if C=1
BrnZ	111 100		" if Z=0
BrnN	111 101		" if N=0
BrnV	111 110		" if V=0

Ciclos de execução: Fetch e Decode

- Ciclo 1: Fetch (busca de instrução)
 - IR_Ld: saída da IM escrita em IR
 - saída de IM mostra continuamente conteúdo da posição apontada por PC
 - PC_Inc: atualiza PC
 - a ser usado na próxima instrução
 - pode ser sobre-escrito se instrução = desvio

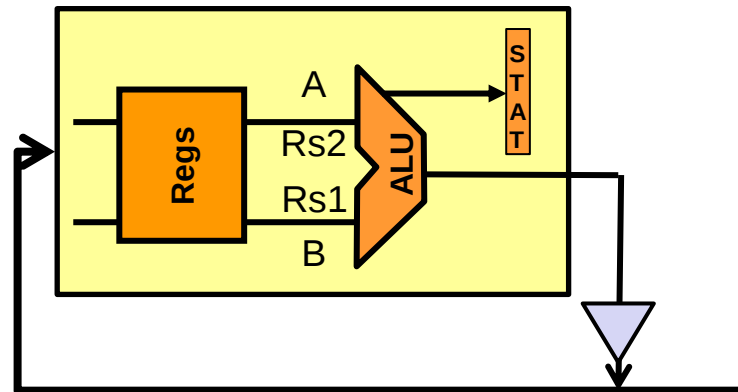


- Ciclo 2: Decodificação
 - Um ciclo para a unidade de controle decodificar a instrução e gerar os sinais de controle



Ciclo 3 de execução: arit/lógicas

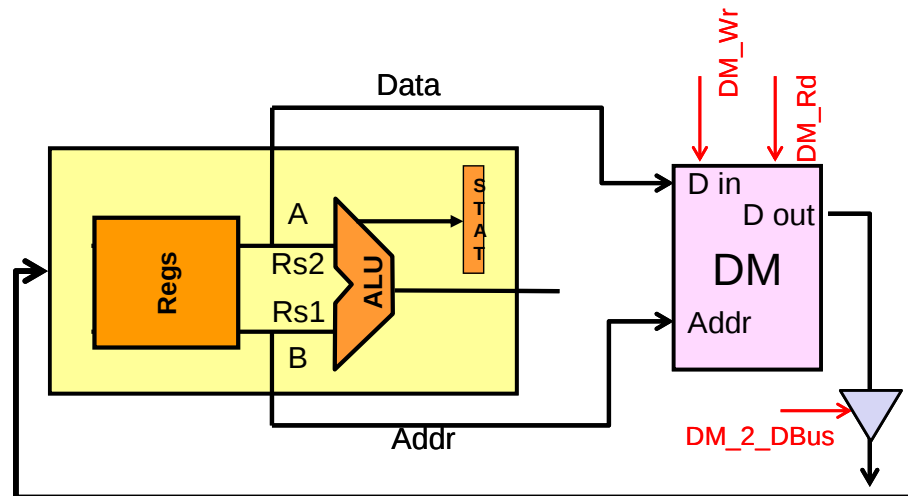
- Configura ALU
 - função: ALU_OP
 - operandos de entrada: Rs1, Rs2
 - registrador de destino: Rd
- Aciona saída do barramento ALU_2_DBus
 - ALU_2_DBus
- Ao final do ciclo (borda do próx. clock), escrita
 - No registrador de destino: Reg_Wr
 - No registrador de status: Stat_Wr
- Pode ser realizada em um ciclo ou 2
 - caminho crítico: seleciona operandos, envia p ALU, realiza operação (32bits), aciona barramento, escreve em Rd





Ciclo 3 de execução: lw e sw

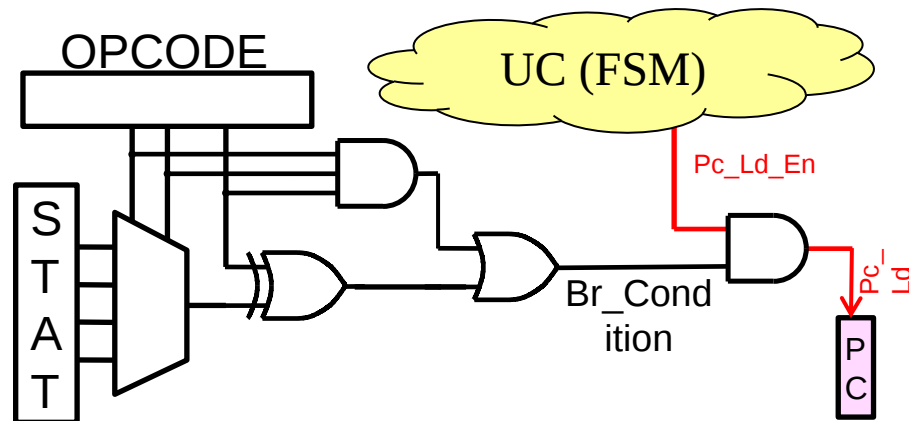
- DM no ciclo 2, definidos
 - Endereço: Rs1
 - Dados para escrita: Rs2
 - Registrador destino: Rd
- SW
 - $DM(Rs1) \leftarrow Rs2$
 - ao final do ciclo: DM_Wr
- lw
 - $Rd \leftarrow DM(Rs1)$
 - $DM_2_DBus = 1$
 - ao final do ciclo: Reg_Wr
- Pode ser realizada em um ciclo ou 2
 - caminho crítico: seleciona dados e endereço, envia p DM, operação de leitura ou escrita, aciona barramento, escreve em Rd





Ciclo 3 de execução: desvio

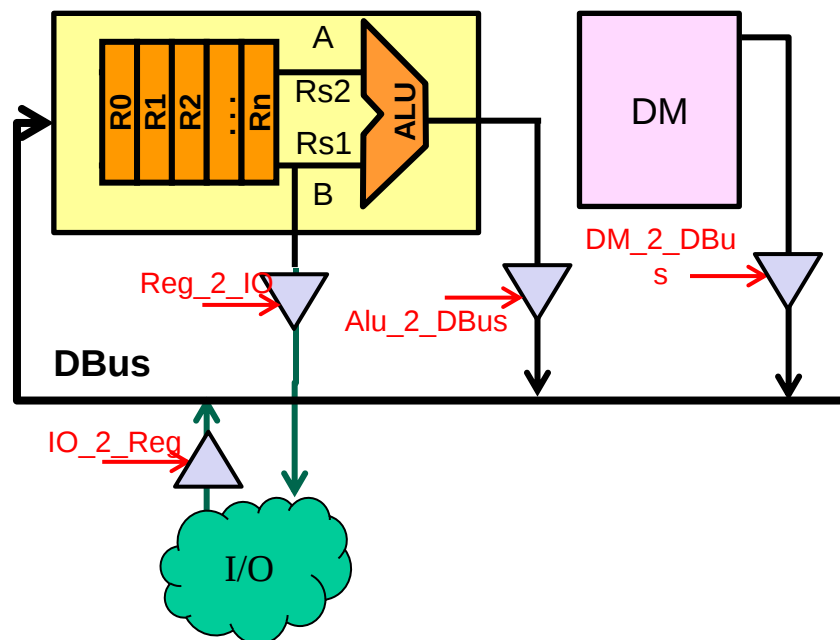
- Uma única ação da UC
 - ver 
 - sinalizar momento da carga de PC → Pc_Ld_En
- Desvio condicional
 - opcode comparado com condição em STAT (última operação aritmética)
- Desvio incondicional
 - decodifica OpCode e gera condição, independente de STAT





Ciclo 3 de execução: I/O

- In Rd
 - $Rd \leftarrow I/O$
 - controles: IO_2_Reg
 - ao final do clock: Reg_Wr
- Out Rs1
 - $I/O \leftarrow Rs1$
 - controles: Reg_2_IO
- Pode ser realizada em um ciclo ou 2
 - caminho crítico: seleciona Rd, ativa leitura do barramento, escreve em Rd



Controle de Desvio: Possibilidades

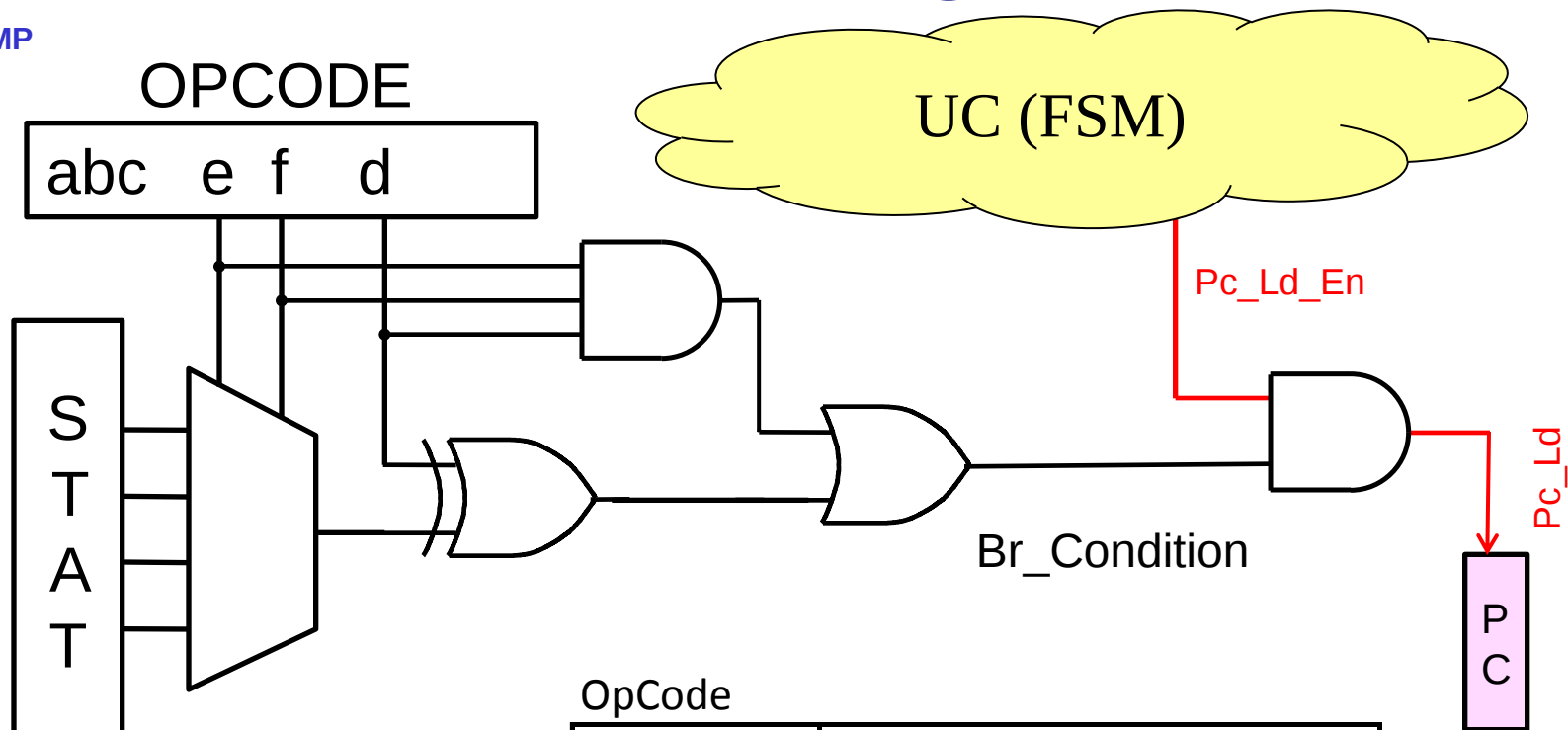


IC-UNICAMP

- Um estado para cada instrução de desvio
 - Teste é específico para o estado
 - Haverá tantos estados específicos quanto instruções de desvio
 - Complica a máquina de estados desnecessariamente
- Alternativa
 - Um único estado na FSM de controle sinaliza instrução de desvio
 - Hardware especializado para controlar desvios (Target Address Controller – TrgtAdrCtl)
 - Situado entre a FSM, PC e Stat
 - Menos HW e mais flexibilidade



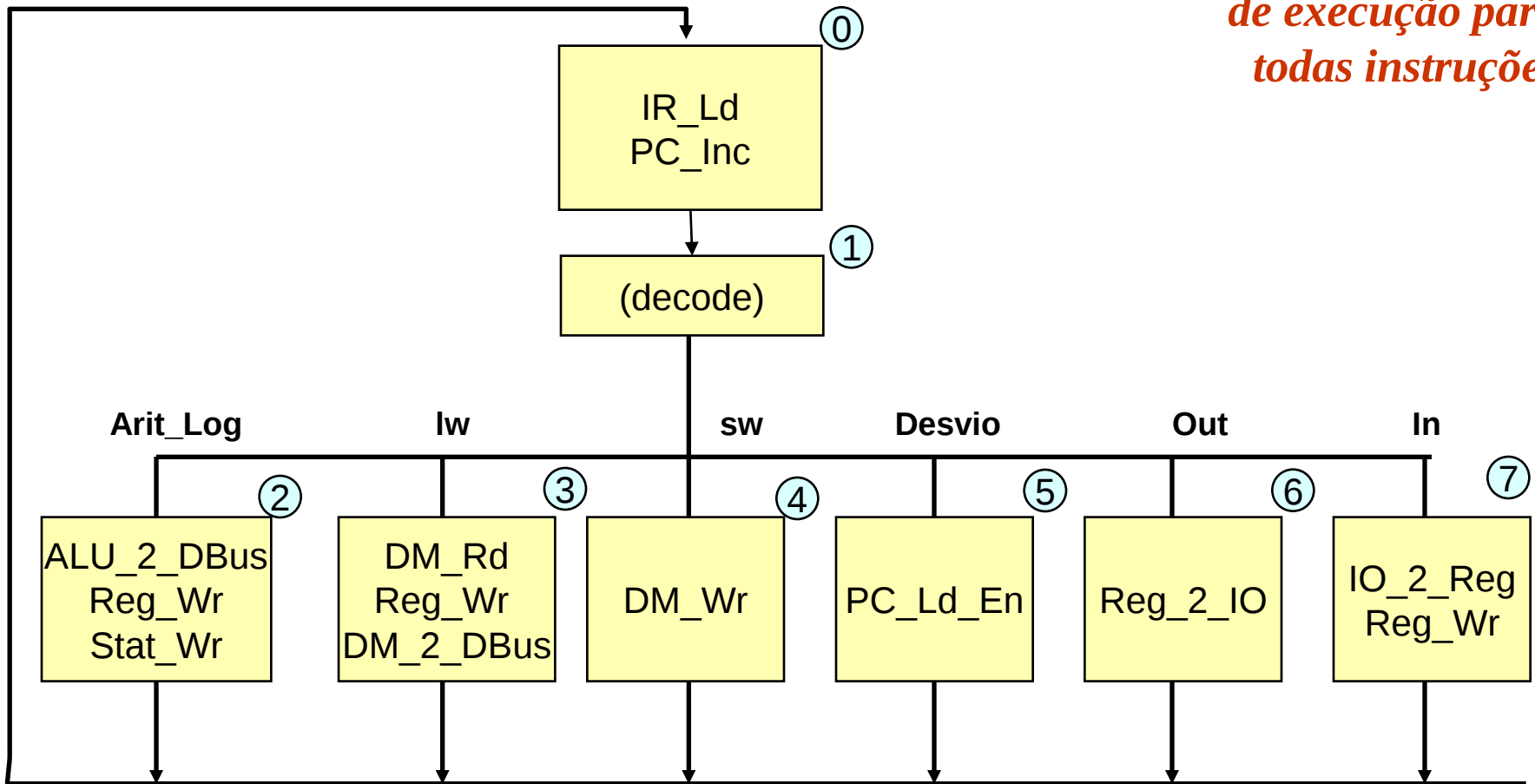
Controle de Desvio: Target Address Ctl



OpCode		
J addr	111 111	$Pc \leftarrow addr$
BrZ	111 000	$Pc \leftarrow addr$ if Z=1
BrN	111 001	$Pc \leftarrow addr$ if N=1
BrV	111 010	$Pc \leftarrow addr$ if V=1
BrC	111 011	$Pc \leftarrow addr$ if C=1
BrnZ	111 100	$Pc \leftarrow addr$ if Z=0
BrnN	111 101	$Pc \leftarrow addr$ if N=0
BrnV	111 110	$Pc \leftarrow addr$ if V=0

Possível fluxo de controle

*Assumindo 1 ciclo
de execução para
todas instruções*

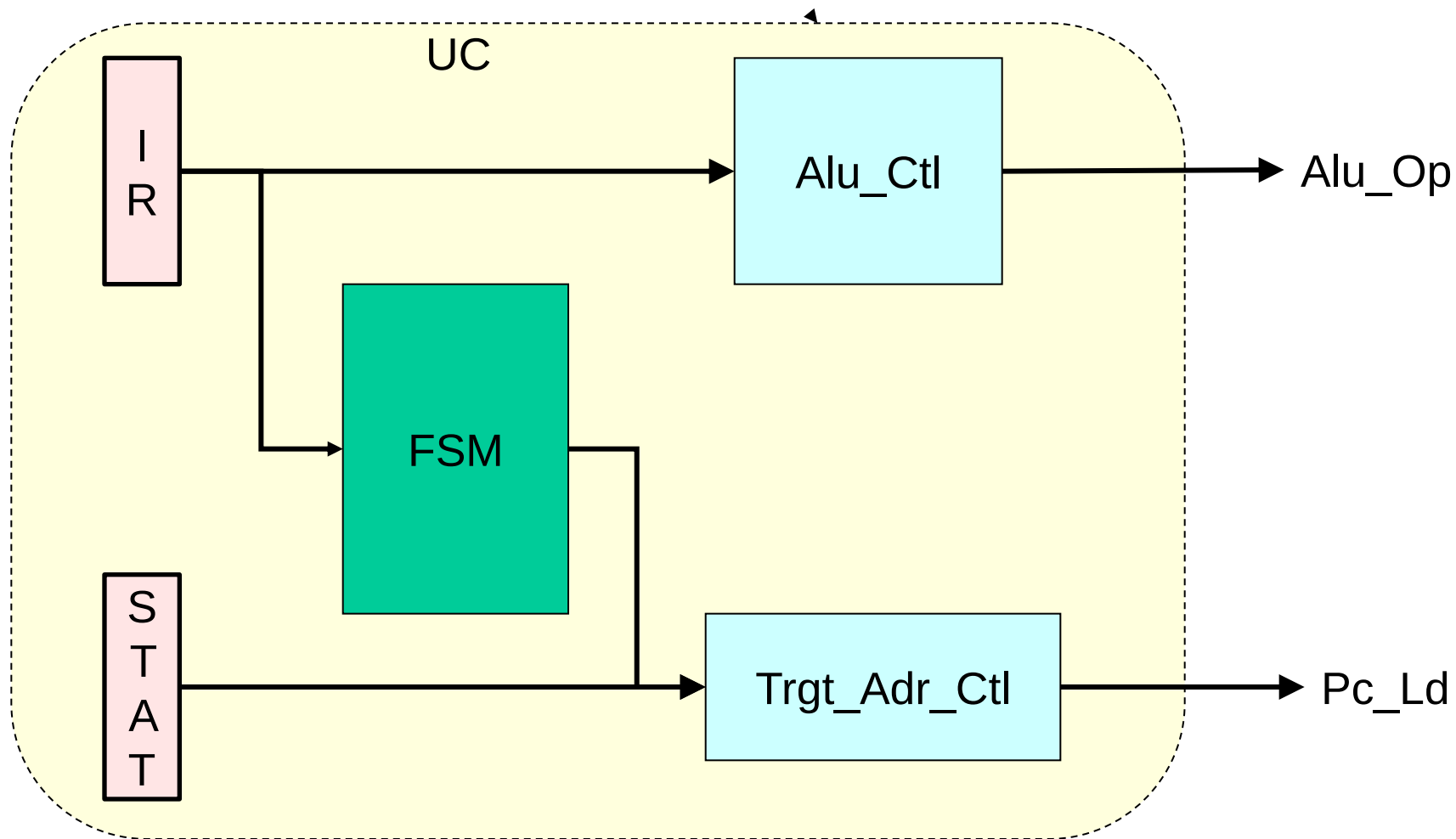


Detalhes da UC



IC-UNICAMP

- FSM + controladores especializados: ALU e PC



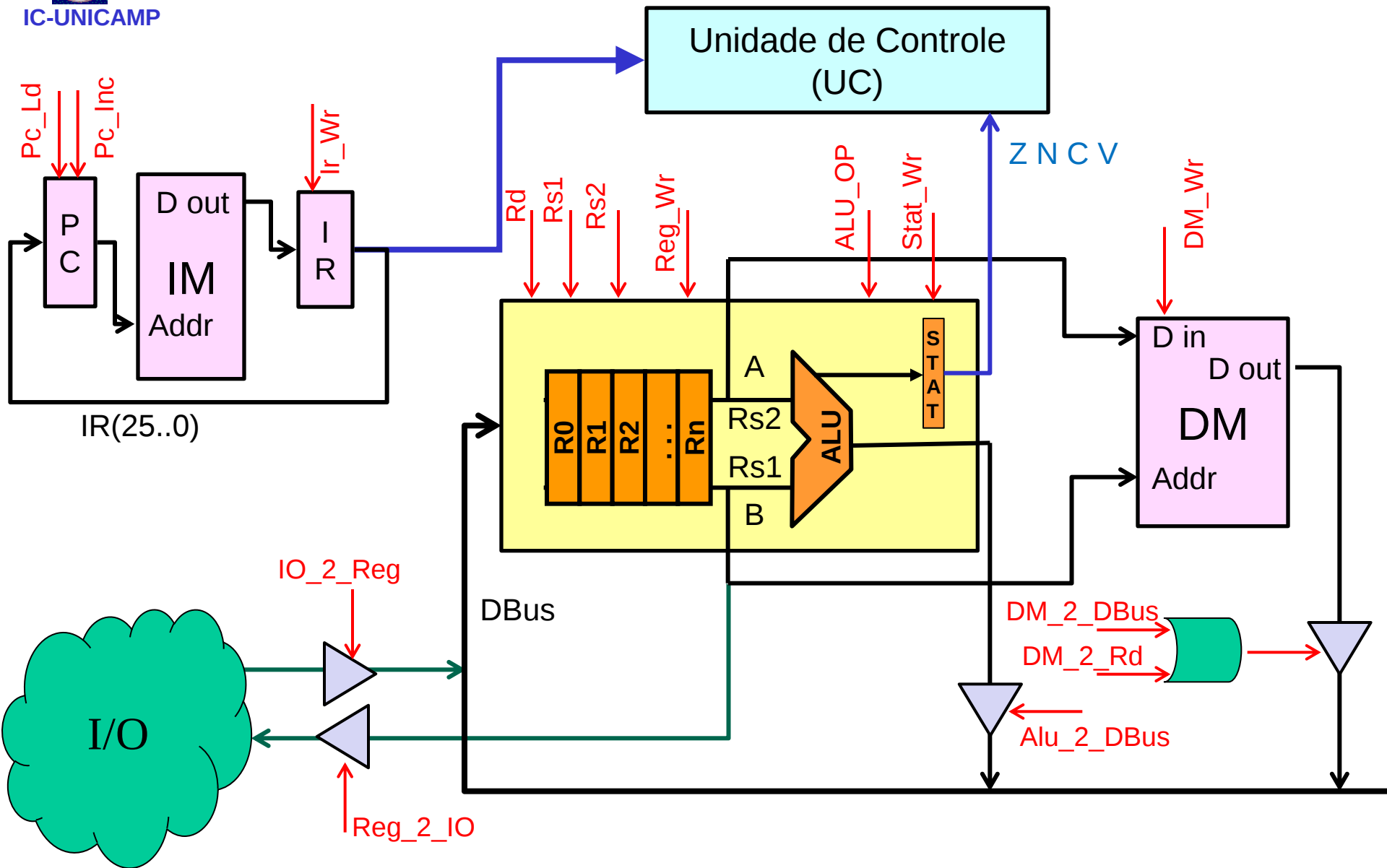


Convenções de timing

- Controles:
 - síncronos com clock (mudança na borda de subida + Δ)
- Escrita em registradores:
 - na próxima borda de subida (sensível à borda)
- Escrita na Data Memory
 - depende da implementação da memória



Diagrama de blocos





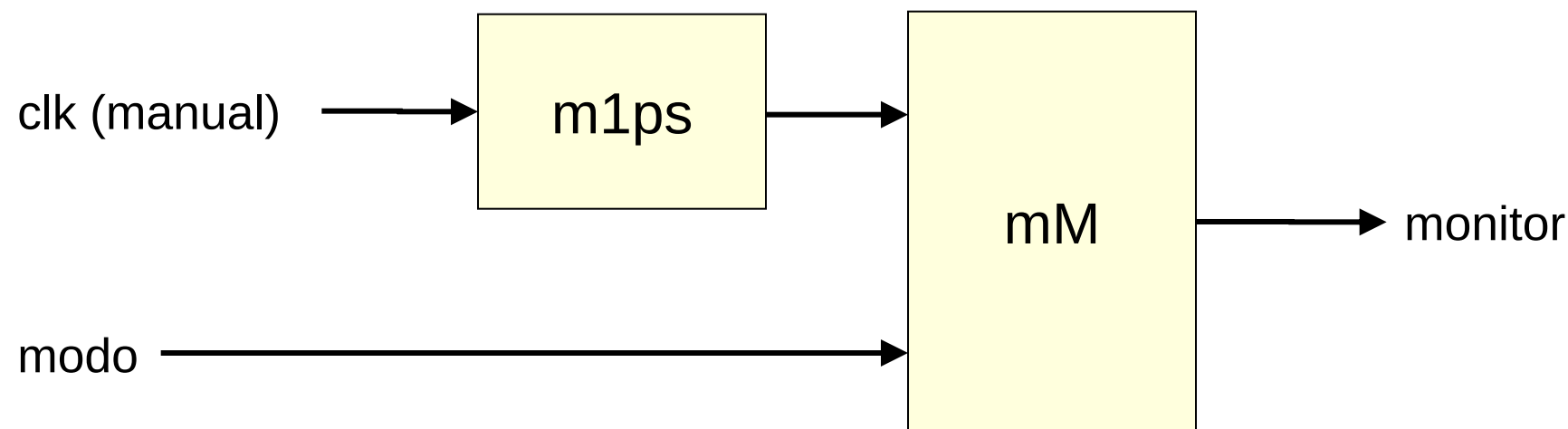
Montador

- Disponível montador
 - linguagem assembly → código binário
- Instruções a parte
- Acesso
 - <http://mc613.caiohoffman.org/>
 - escolher arquivo fonte (arquivo .m1ps)
 - copiar código objeto para .mif



Monitoramento da execução na DE1

- m1ps Monitor (mM)





Telas geradas pelo mM: passo a passo

- Saídas de dados e controle do m1ps
- Atualizada a cada clock: scroll automatico
- Valores dos regs: devem ser derivados pelo mM

m1ps Monitor

– Shadow Register Bank

--> Geral

Pc: 0x76543210 Rs1: 0x76543210
Ir: 0x76543210 Rs2: 0x76543210
DBus: 0x76543210 Stat: 0x3210

--> Sinais de controle

Pc_Ld: 1b	Stat_Wr: 1b	Alu_2_DBus: 1b
Pc_Inc: 1b	DM_Wr: 1b	Reg_2_IO: 1b
Ir_Wr: 1b	DM_Rd: 1b	IO_2_Reg: 1b
Alu_Op: 1b	DM_2_DBus: 1b	Reg_Wr: 1b

--> Registradores

R0: 0x76543210	R8: 0x76543210	R16: 0x76543210	R24: 0x76543210
R1: 0x76543210	R9: 0x76543210	R17: 0x76543210	R25: 0x76543210
R2: 0x76543210	R10: 0x76543210	R18: 0x76543210	R26: 0x76543210
R3: 0x76543210	R11: 0x76543210	R19: 0x76543210	R27: 0x76543210
R4: 0x76543210	R12: 0x76543210	R20: 0x76543210	R28: 0x76543210
R5: 0x76543210	R13: 0x76543210	R21: 0x76543210	R29: 0x76543210
R6: 0x76543210	R14: 0x76543210	R22: 0x76543210	R30: 0x76543210
R7: 0x76543210	R15: 0x76543210	R23: 0x76543210	R31: 0x76543210

Tela principal: SW(0) = 0; Memoria: SW(0) = 1;



Telas geradas pelo mM: final

- Conteúdo da memória de dados (total)
- Deve ser derivado pelo mM: Shadow DM
 - Exemplo da tela no próximo slide.



--> Memoria

	00	01	02	03	04	05	06	07
00	76543210	76543210	76543210	76543210	76543210	76543210	76543210	76543210
01	76543210	76543210	76543210	76543210	76543210	76543210	76543210	76543210
02	76543210	76543210	76543210	76543210	76543210	76543210	76543210	76543210
03	76543210	76543210	76543210	76543210	76543210	76543210	76543210	76543210
04	76543210	76543210	76543210	76543210	76543210	76543210	76543210	76543210
05	76543210	76543210	76543210	76543210	76543210	76543210	76543210	76543210
06	76543210	76543210	76543210	76543210	76543210	76543210	76543210	76543210
07	76543210	76543210	76543210	76543210	76543210	76543210	76543210	76543210
08	76543210	76543210	76543210	76543210	76543210	76543210	76543210	76543210
09	76543210	76543210	76543210	76543210	76543210	76543210	76543210	76543210
0A	76543210	76543210	76543210	76543210	76543210	76543210	76543210	76543210
0B	76543210	76543210	76543210	76543210	76543210	76543210	76543210	76543210
0C	76543210	76543210	76543210	76543210	76543210	76543210	76543210	76543210
0D	76543210	76543210	76543210	76543210	76543210	76543210	76543210	76543210
0E	76543210	76543210	76543210	76543210	76543210	76543210	76543210	76543210
0F	76543210	76543210	76543210	76543210	76543210	76543210	76543210	76543210
10	76543210	76543210	76543210	76543210	76543210	76543210	76543210	76543210
11	76543210	76543210	76543210	76543210	76543210	76543210	76543210	76543210
12	76543210	76543210	76543210	76543210	76543210	76543210	76543210	76543210
13	76543210	76543210	76543210	76543210	76543210	76543210	76543210	76543210
14	76543210	76543210	76543210	76543210	76543210	76543210	76543210	76543210
15	76543210	76543210	76543210	76543210	76543210	76543210	76543210	76543210
16	76543210	76543210	76543210	76543210	76543210	76543210	76543210	76543210
17	76543210	76543210	76543210	76543210	76543210	76543210	76543210	76543210
18	76543210	76543210	76543210	76543210	76543210	76543210	76543210	76543210
19	76543210	76543210	76543210	76543210	76543210	76543210	76543210	76543210
1A	76543210	76543210	76543210	76543210	76543210	76543210	76543210	76543210
1B	76543210	76543210	76543210	76543210	76543210	76543210	76543210	76543210
1C	76543210	76543210	76543210	76543210	76543210	76543210	76543210	76543210
1E	76543210	76543210	76543210	76543210	76543210	76543210	76543210	76543210
1F	76543210	76543210	76543210	76543210	76543210	76543210	76543210	76543210

Tela principal: SW(0) = 0; Memoria: SW(0) = 1;

Addr = linha*8 + coluna



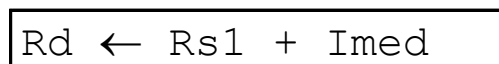
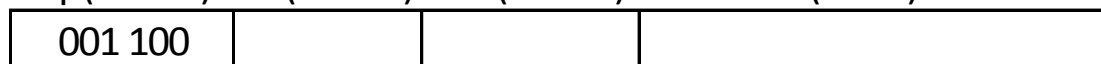
Extensões do ISA: addi

- addi Rd, Rs1, imediato
- Alterações
 - Via de dados: caminho de IR(15..0) até a entrada da ALU, com extensão de sinal
 - UC: inclusão de novo ramo na fase de execução da máquina de estado
- Necessidade de novo formato I
 - campo para imediato

Instruções Formato I, contendo campo imediato: addi Rd, Rs1, Imed

Op(31..26) Rd(25..21) Rs1(20..16) Imediato(15..0)

Obs



6

5

5

16