UCP e pipelines

Aleardo Manacero Jr.

DCCE/UNESP

Grupo de Sistemas Paralelos e Distribuídos

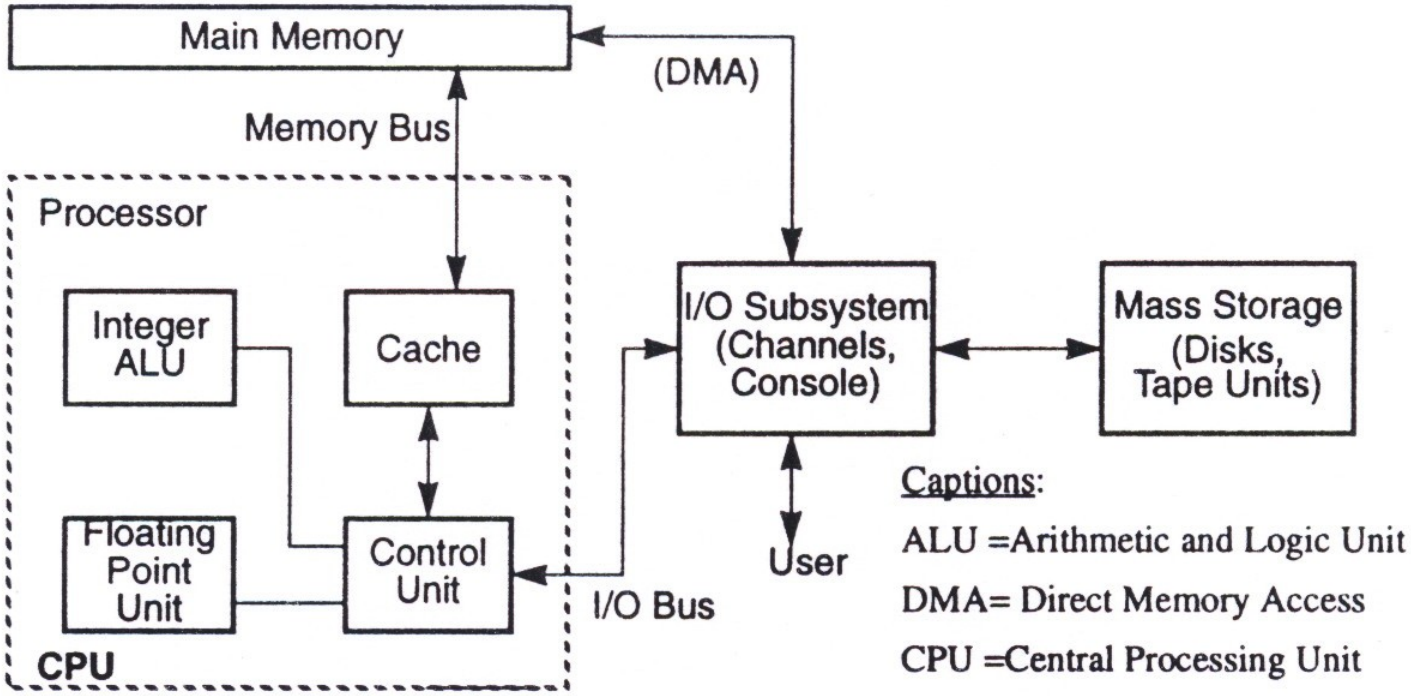
Conteúdo

- Introdução aos modelos arquiteturais da CPU
- Pipelines
  - Motivação
  - Estrutura
  - Princípios de projeto
  - Exemplos

Estrutura da CPU

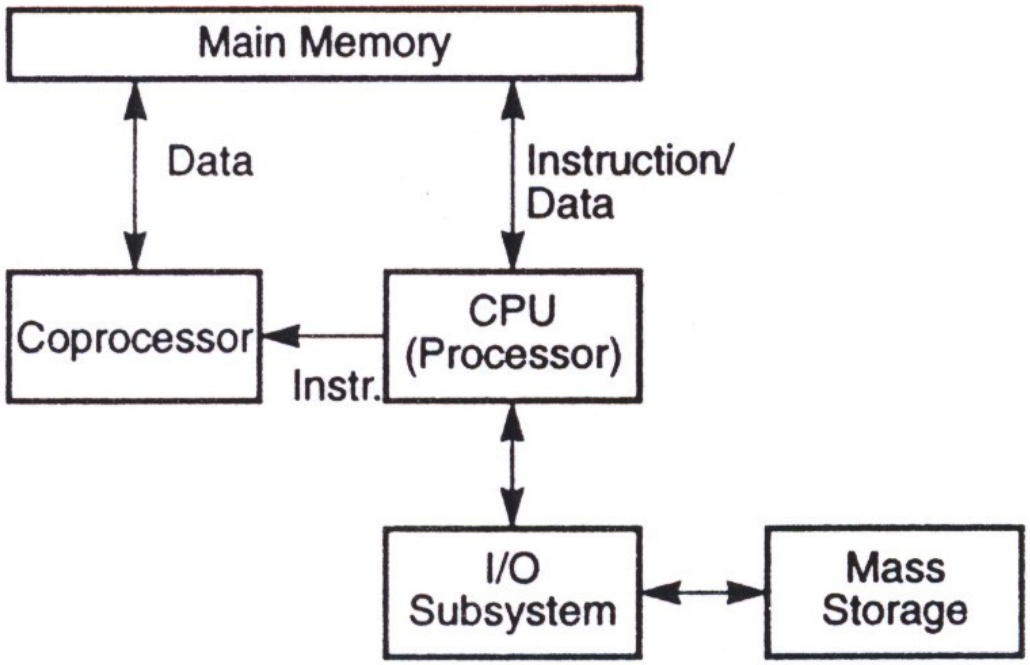
- Componentes básicos:
  - Registradores
  - Unidade de controle
  - Unidade lógico-aritmética

Modelos arquiteturais básicos



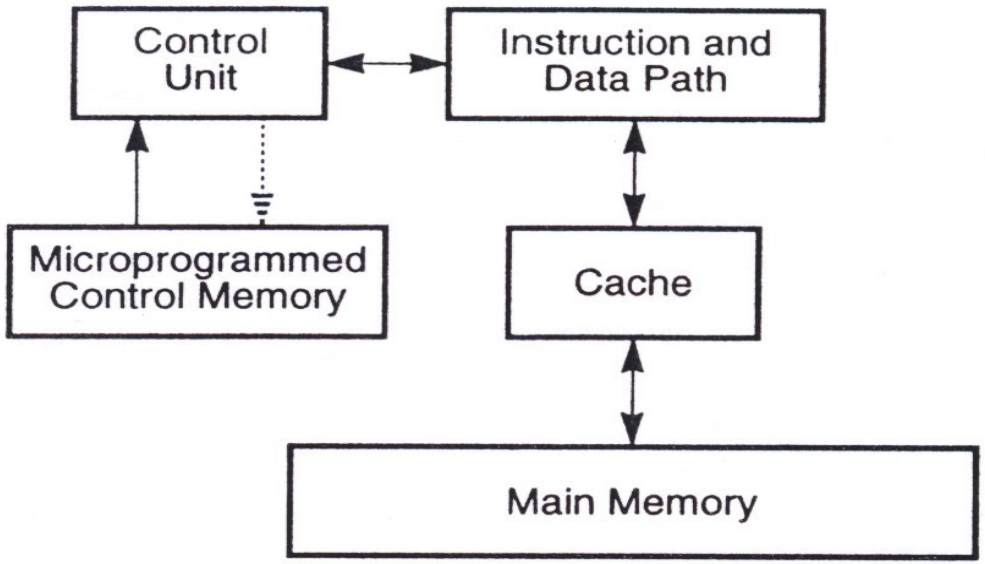
(a) CPU with built-in floating-point unit

Modelos arquiteturais básicos



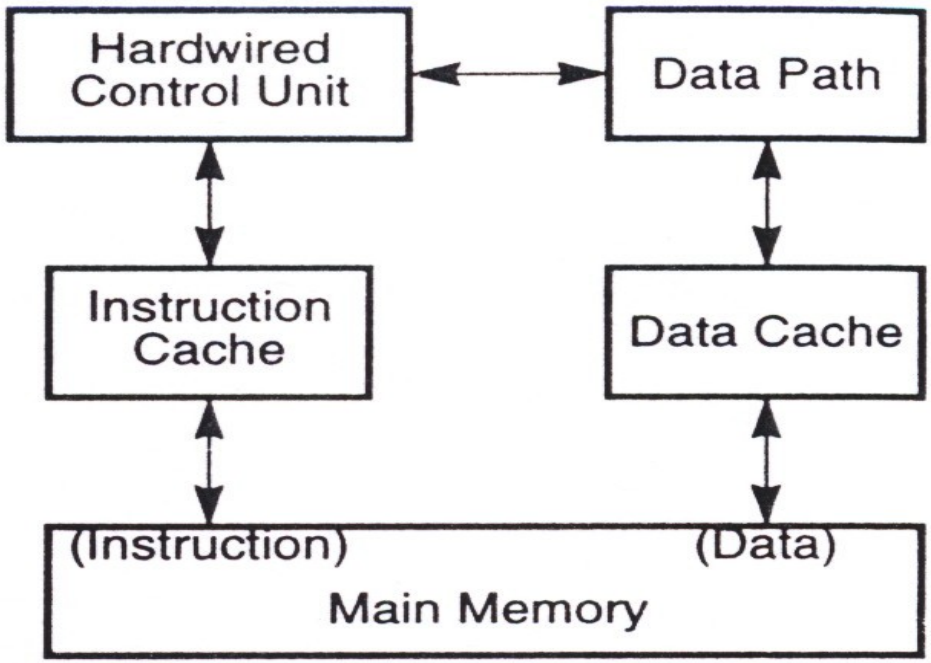
(b) CPU with an attached coprocessor

Modelo de arquitetura CISC



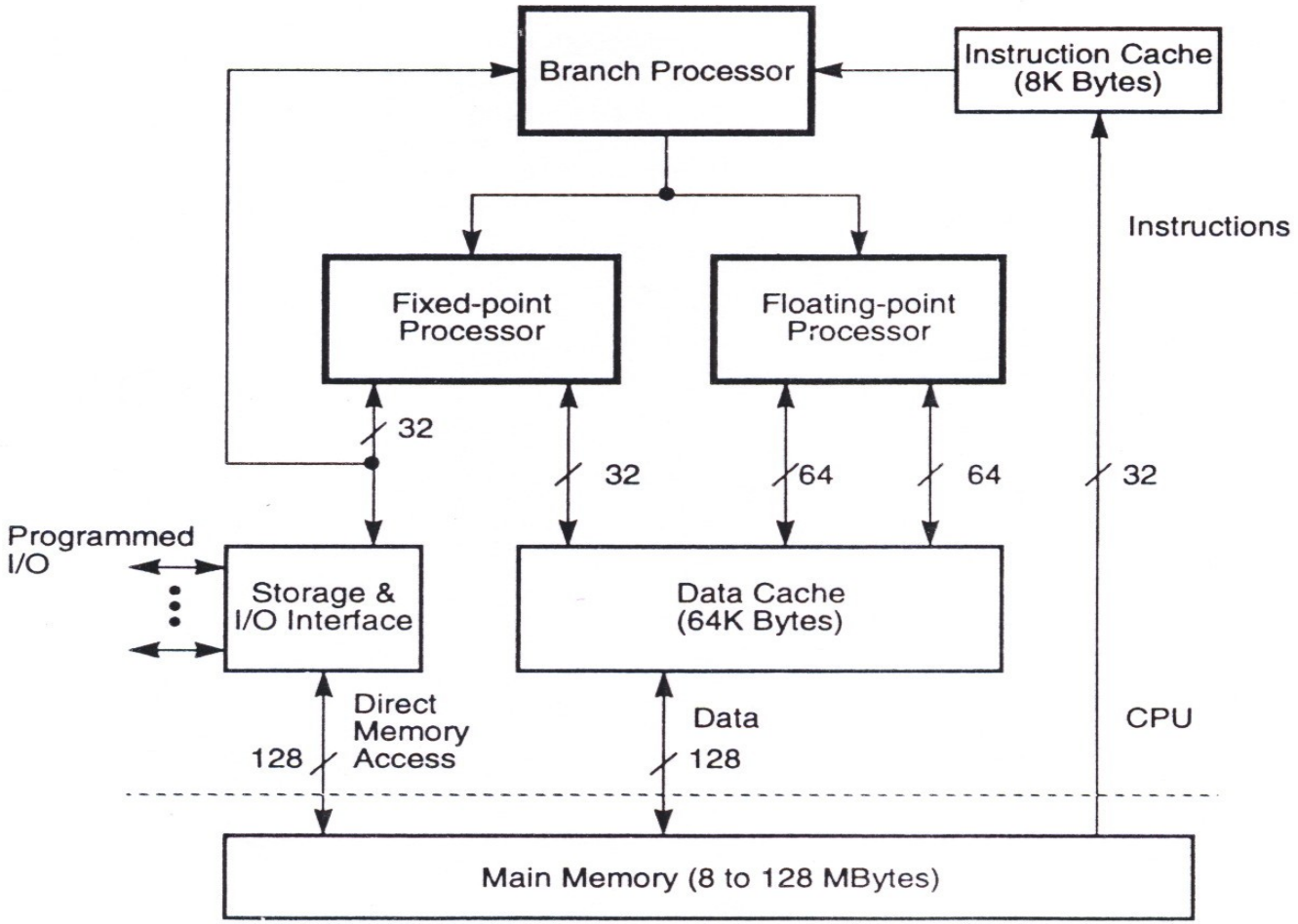
(a) The CISC architecture with microprogrammed control and unified cache

Modelo de arquitetura RISC



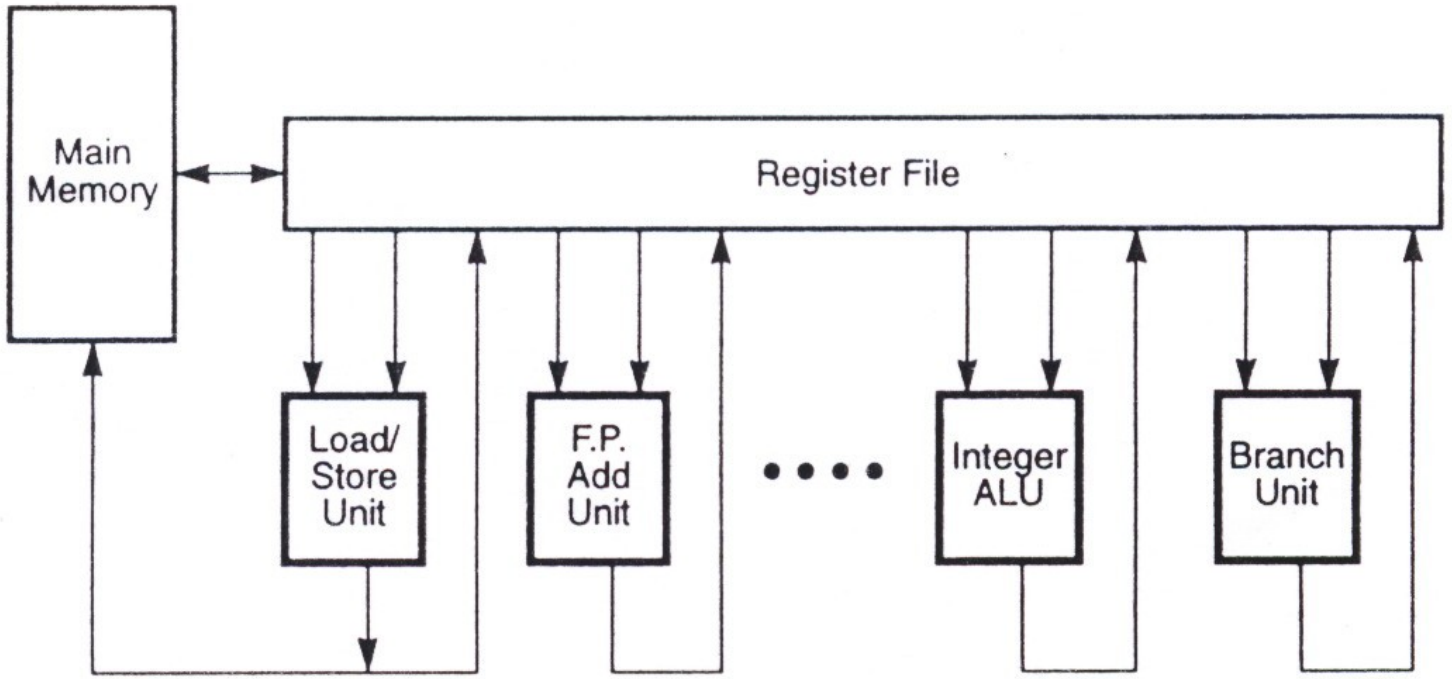
(b) The RISC architecture with hardwired control and split instruction cache and data cache.

Arquitetura POWER (32 bits)



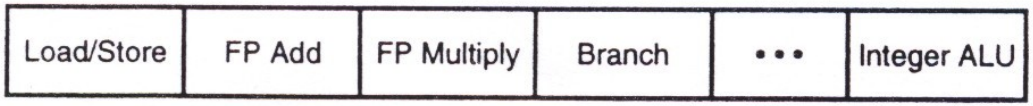
Arquitetura VLIW

- Modelo de processador

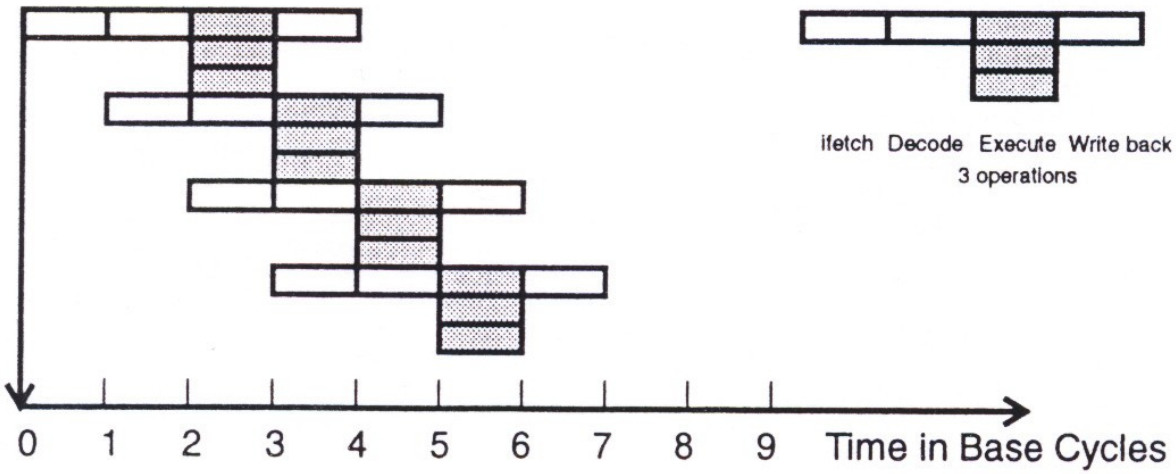


Arquitetura VLIW

- Formato de instrução e execução



(a) A typical VLIW processor and instruction format

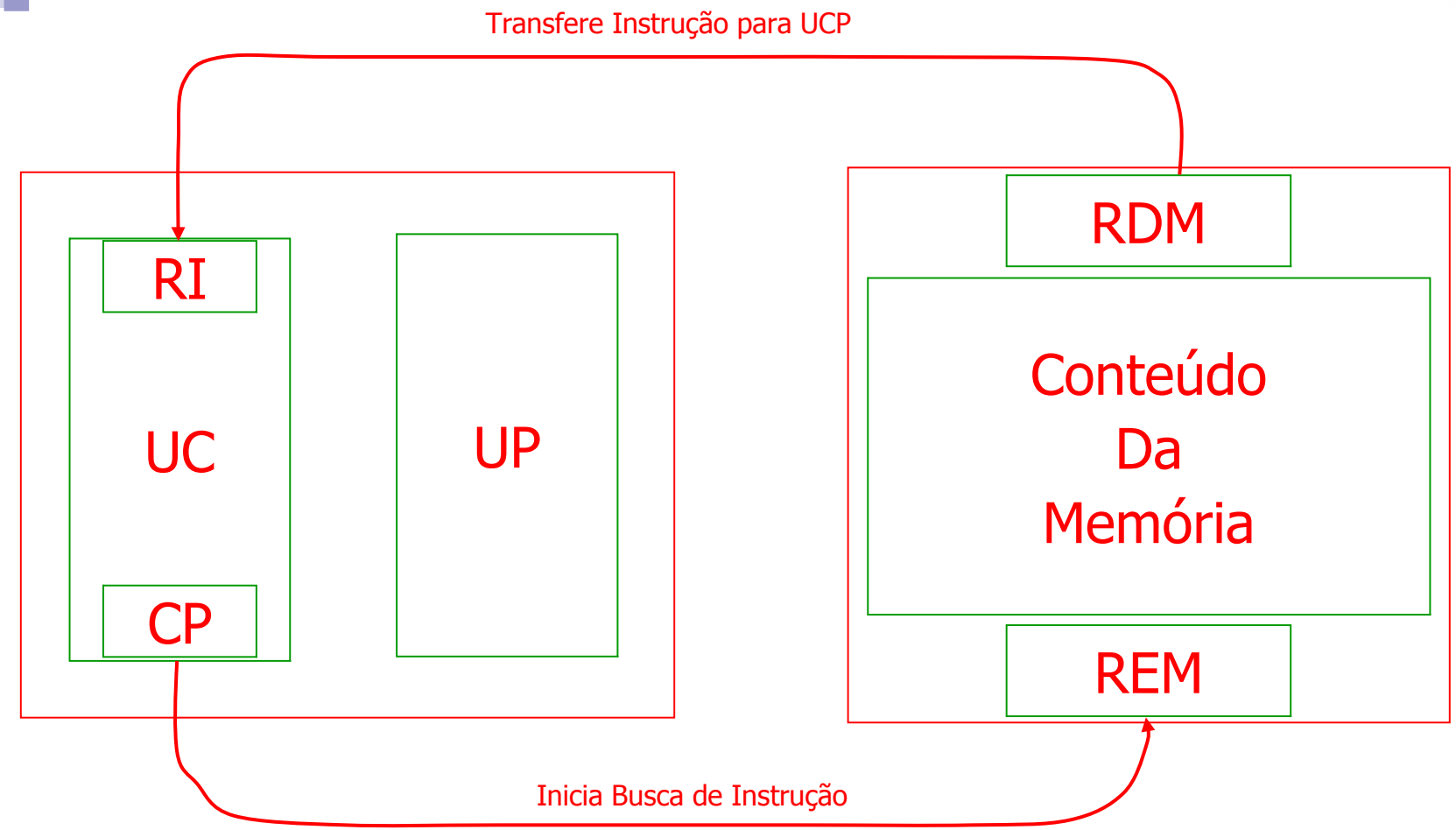


(b) VLIW execution with degree $m = 3$

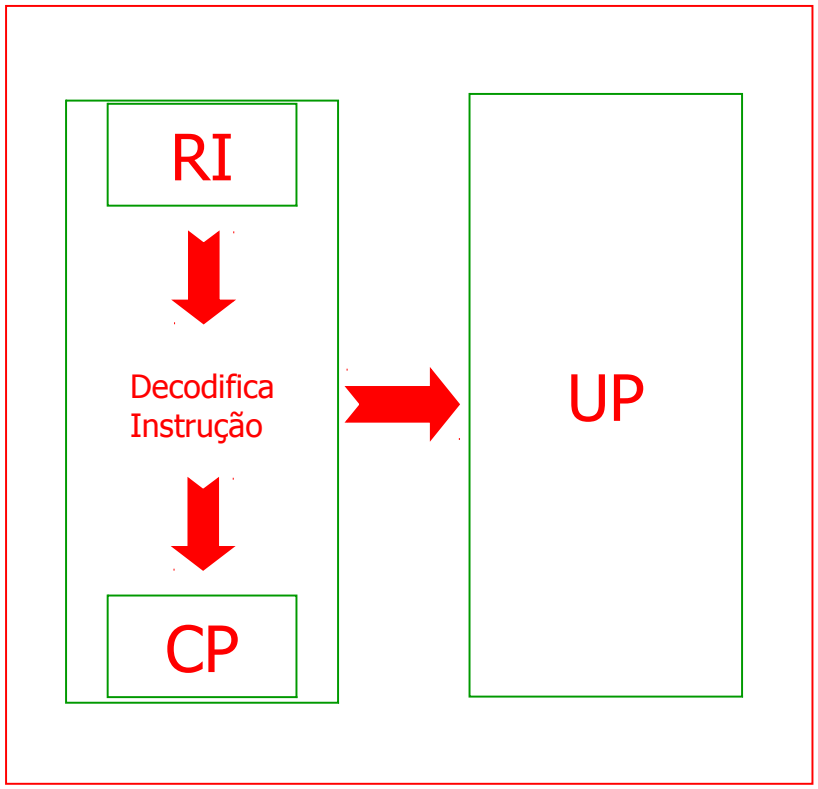
Estrutura da CPU

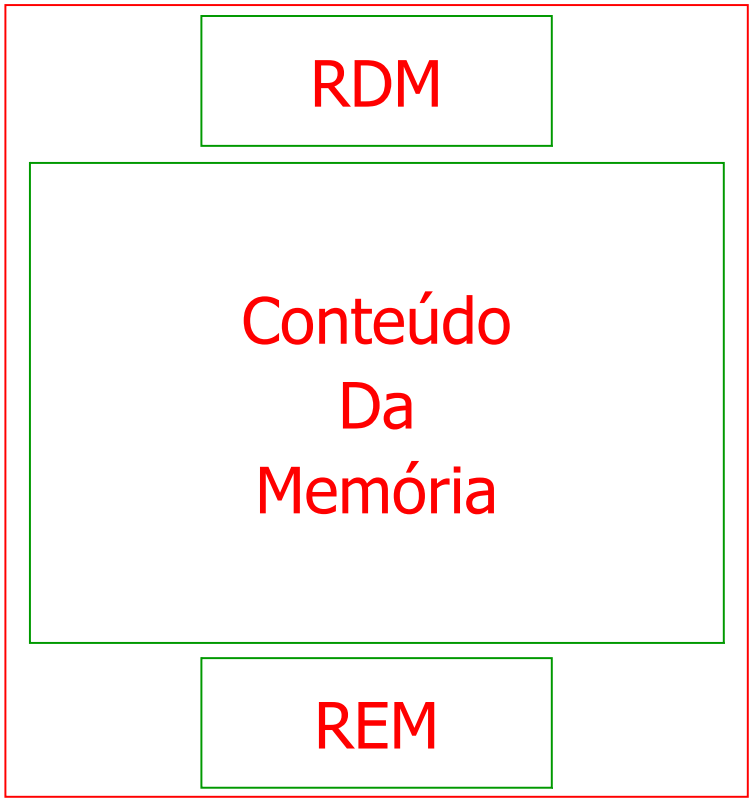
- Funcionamento
  - Instruções e dados são transferidos da memória para a CPU
  - CPU executa a tarefa
  - Resultados são retornados para a memória

Funcionamento da CPU

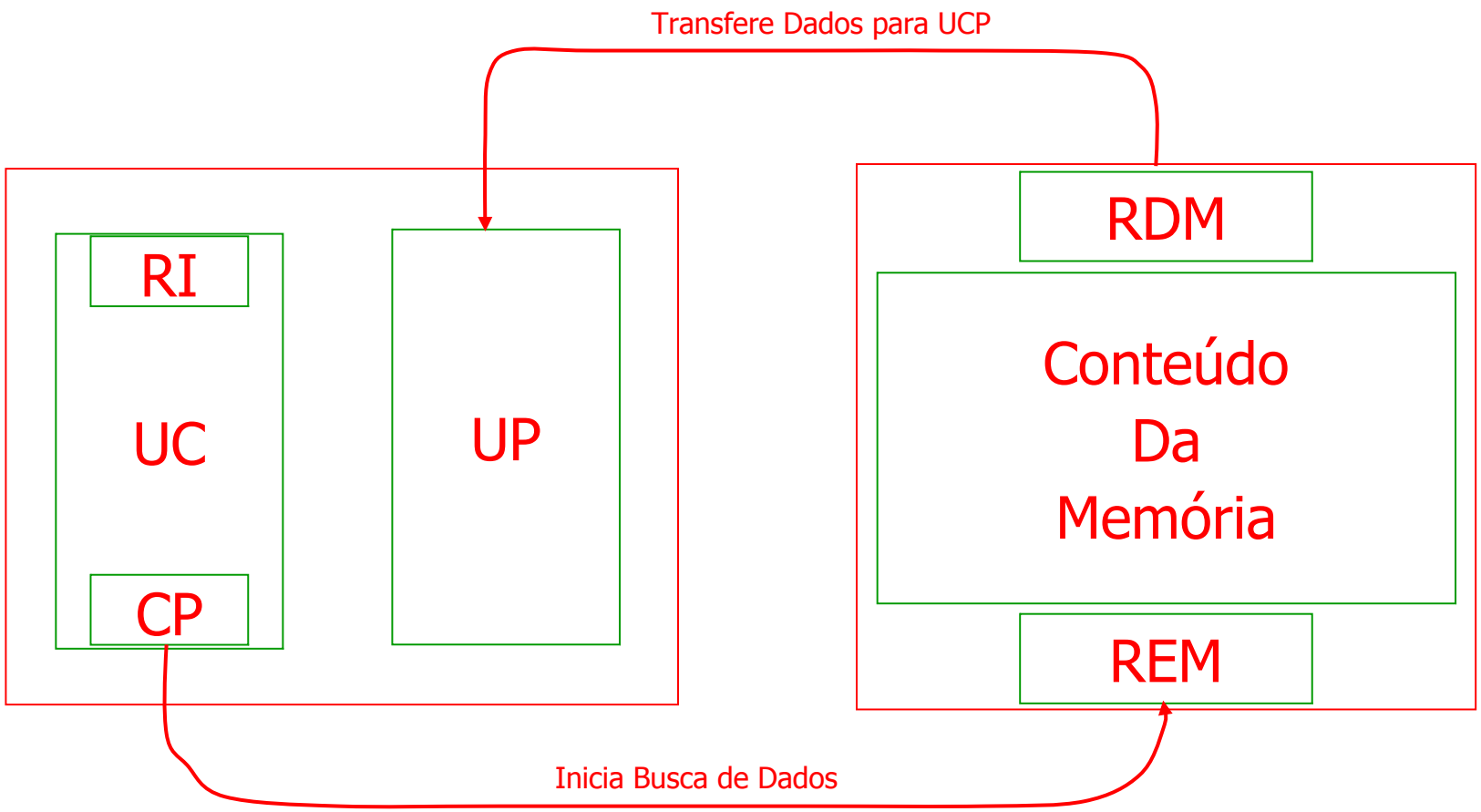


Funcionamento da CPU

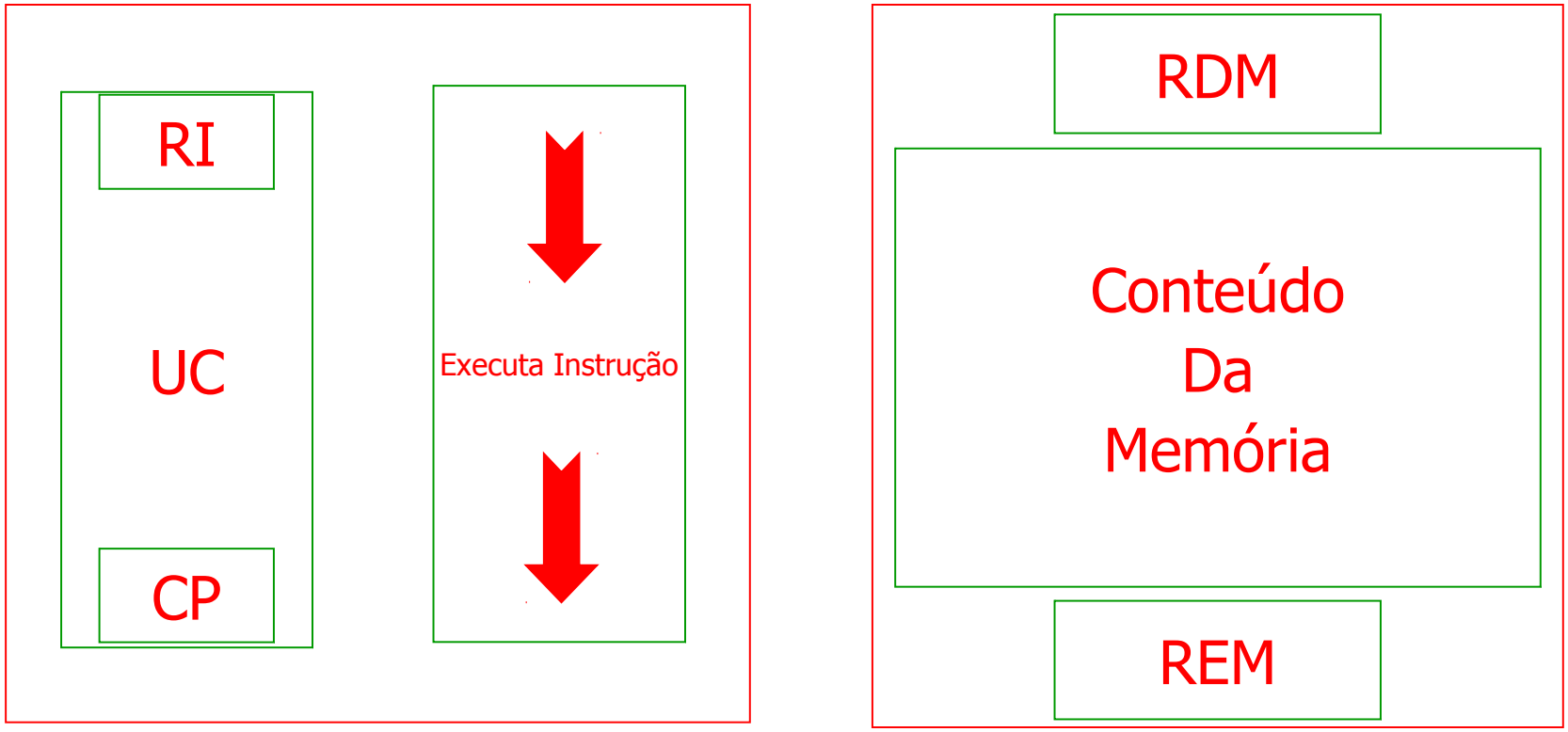




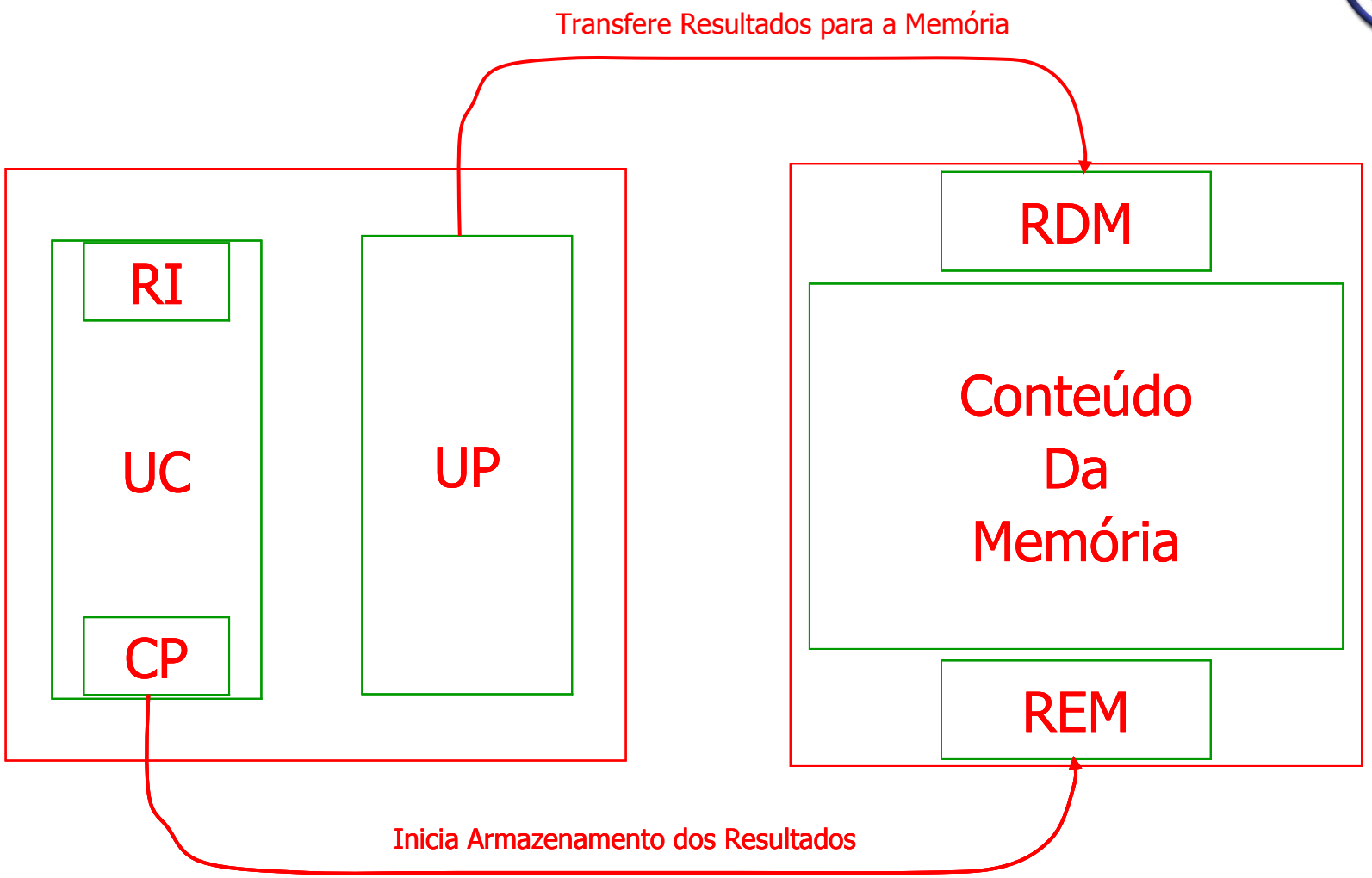
Funcionamento da CPU



Funcionamento da CPU



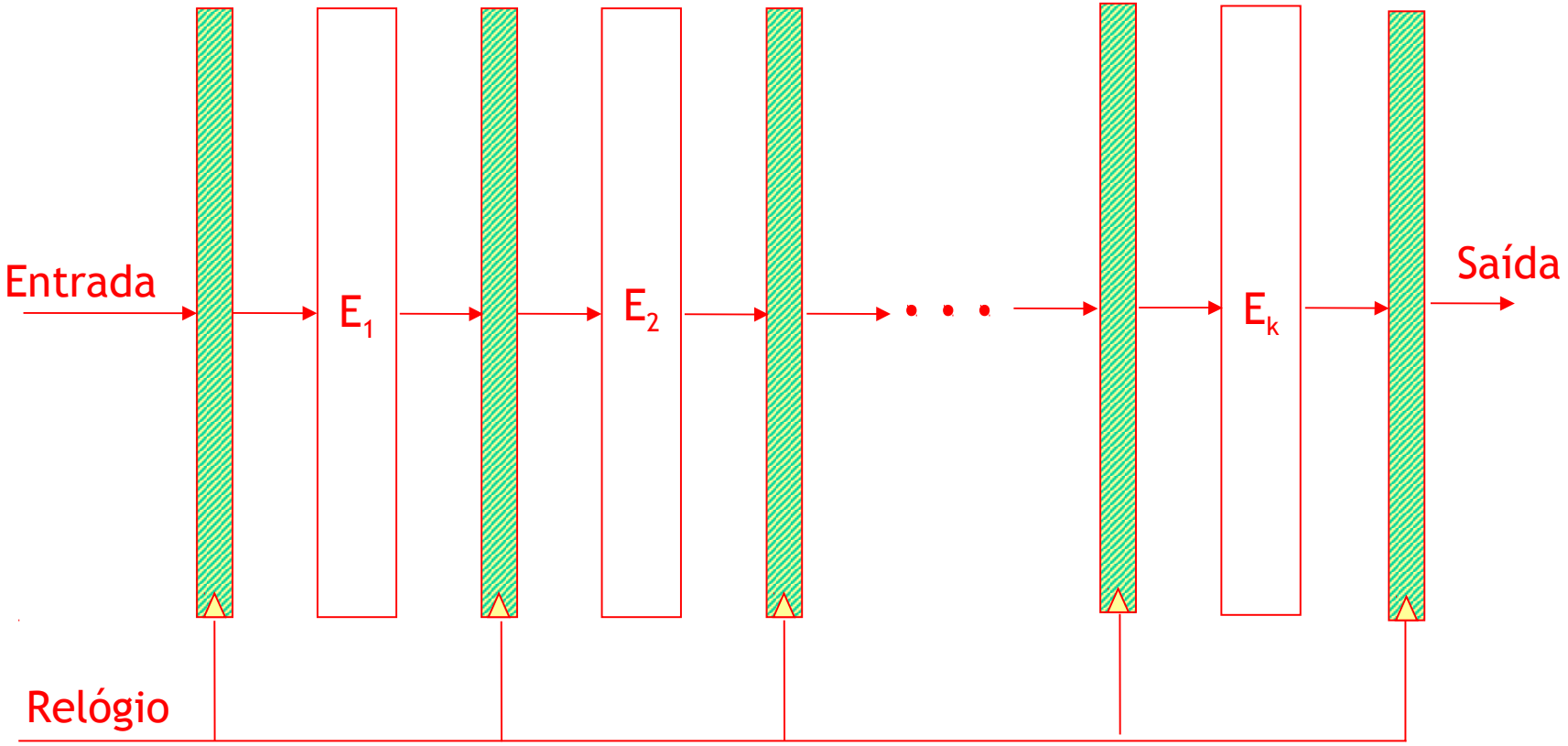
Funcionamento da CPU



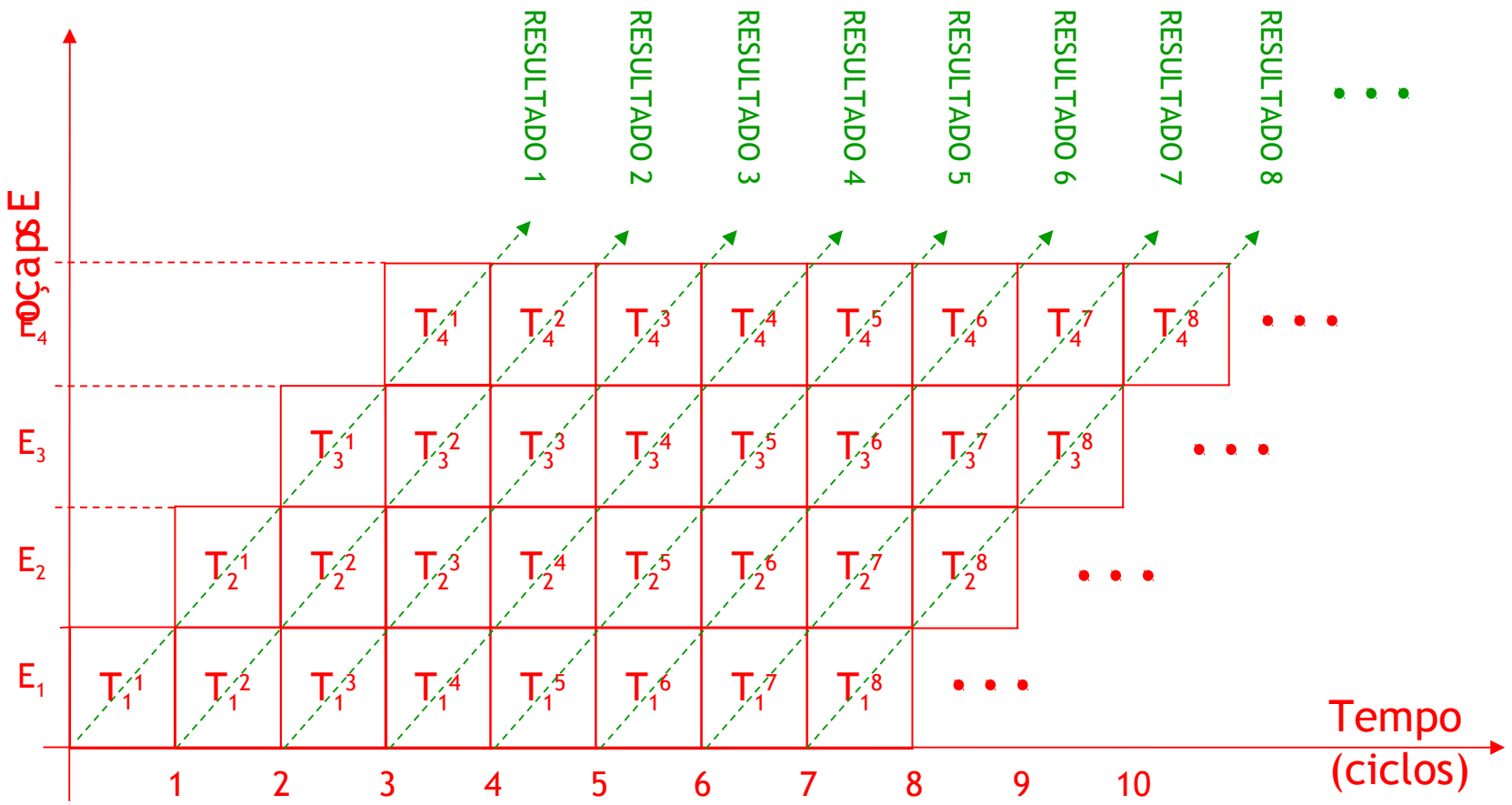
Funcionamento da CPU

- O problema na forma de funcionamento apresentada é o atraso inerente pela serialização das atividades
- Uma solução é realizar as operações como se estivessemos numa linha de produção
- Isso é feito através de pipelines, que permitem acelerar as fases serializadas da execução de instruções consecutivas

Princípio do pipeline



Princípio do pipeline



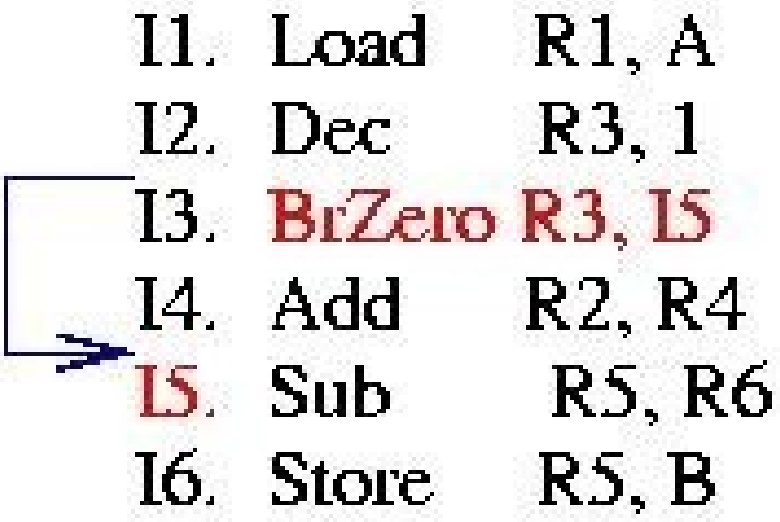
Problemas com pipelines

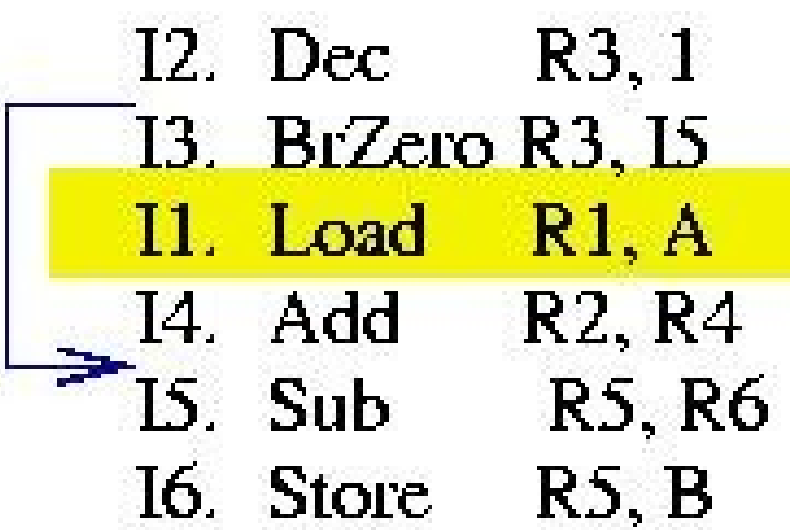
- A eficiência do pipeline depende de não se quebrar a sequência entre as instruções que são executadas
- O problema é que instruções de saltos tipicamente forçam essa quebra, o que implica que instruções já no pipeline terão que ser descartadas
- Isso é o que se chama de esvaziamento do pipeline

Tratamento do esvaziamento

- Delay slot
  - Insere instruções (ou altera a ordem de execução) de modo a permitir o aproveitamento das instruções que estão no pipeline quando da instrução de desvio
- Predição de caminho
  - Busca adivinhar qual o caminho que será seguido após uma instrução de salto
- Execução condicional
  - Executa os dois caminhos até que se saiba qual deve ser seguido

A técnica de *delay slot*





Dimensionamento de pipeline

- No projeto de um pipeline é importante identificar como a execução de cada instrução pode ser dividida de modo a aumentar a eficiência da execução.
- Assim, são parâmetros importantes:
  - Tipos de instruções
  - Número de estágios
  - Duração de cada estágio
  - Modelo de sincronismo entre estágios

Dimensionamento de pipeline

- Estágios básicos:
  - Busca de instrução
  - Decodificação de instrução
  - Busca de dados
  - Execução de instrução
- Esses estágios podem ser (e normalmente são) divididos em estágios ainda menores, a partir da identificação das microinstruções que efetivamente implementam cada operação

Dimensionamento de pipeline

- A busca de instrução envolve:
  - Interpretação do valor atual do contador de programas (PC)
  - Busca do conteúdo presente na posição de memória endereçada por PC
  - Transferência da instrução da memória para o registrador de instruções
  - Atualização do contador de programas

Dimensionamento de pipeline

- A decodificação de instrução envolve:
  - Identificação do código da instrução
  - Geração dos sinais de controle
  - Geração dos endereços de dados

Dimensionamento de pipeline

- A busca de dados segue um padrão semelhante ao da busca de instruções:
  - Interpretação dos endereços dos dados constantes do registrado de endereços
  - Identificação dos dados desejados
  - Transferência dos dados da memória para registradores da CPU
  - Atualização do contador de programas

Dimensionamento de pipeline

- Execução de instrução
  - Armazena no acumulador o resultado da aplicação da instrução sobre os dados
  - Transfere esse resultado do acumulador para algum registrador convencional

Dimensionamento de pipeline

- Exemplo de dimensionamento:

Fase	Micro-operação	Tempo (ns)
BI		
	REM←CP	3
	Decodifica REM	6
	RDM←M[REM]	10
	RI←RDM, inc(CP)	3
DI		
	Id. Cod. Oper.	12
	Gera Endereço	9
BD		
	REM←End.Dado	3
	Decodifica REM	6
	RDM←M[REM]	10
	Rx←RDM	3
EX		
	Ac←f{ULA}	22
	Ry←Ac	3
Total		90

Dimensionamento de pipeline

- Duração de cada estágio
  - Uma das técnicas é adotar relógio único para todos os estágios
  - Assim, num pipeline síncrono é possível fazer as instruções passarem de um estágio a outro simultaneamente
  - O problema aqui é definir uma quantização que seja eficiente para a máxima divisão das operações anteriormente descritas

Determinação do ciclo

BI → 22ns

DI → 12ns a 21ns

BD → até 22ns

Ex → 25ns

T_{ot} = 10ns (90ns ÷ 9 segmentos)

T_{min} = 10ns x 1,05 + 4ns = 14,5ns (o fator 1,05 e o ajuste de 4ns são parâmetros de projeto)

Determinação do ciclo

- Eficiência:
  - 90ns → 11MHz
  - 14,5ns → 70MHz (130,5ns por instrução)

Determinação do ciclo

T_{min} = 10ns x 1,05 + 4ns = 14,5ns

Fase	Micro-operação	Tempo (ns)	
BI			
	REM←CP	3	9ns
	Decodifica REM	6	
	RDM←M[REM]	10	10ns
	RI←RDM, inc(CP)	3	6ns
DI			
	Id. Cod. Oper.	12	9ns
	Gera Endereço	9	9ns
BD			
	REM←End.Dado	3	9ns
	Decodifica REM	6	
	RDM←M[REM]	10	10ns
	Rx←RDM	3	8ns
EX			
	Ac←f{ULA}	22	10ns
	Ry←Ac	3	10ns
Total		90	

Determinação do ciclo

Fase	Micro-operação	Tempo (ns)
BI		
	REM←CP	3
	Decodifica REM	6
	RDM←M[REM]	10
	RI←RDM, inc(CP)	3
DI		
	Id. Cod. Oper.	12
	Gera Endereço	9
BD		
	REM←End.Dado	3
	Decodifica REM	6
	RDM←M[REM]	10
	Rx←RDM	3
EX		
	Ac←f{ULA}	22
	Ry←Ac	3
Total		90

T_{min} = 15ns x 1,05 + 4ns = 19,75ns ≈ 20ns

(3+6+6)ns=15ns

(4+3+8)ns=15ns

(4+9+2)ns=15ns

(1+6+8)ns=15ns

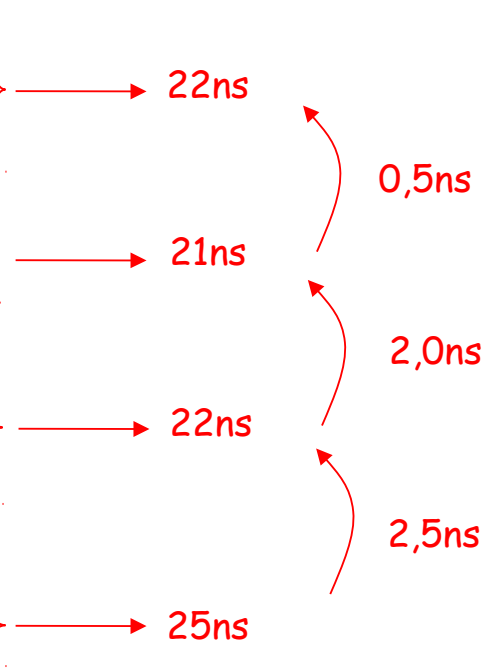
(2+3+10)ns=15ns

(12+3)ns = 15ns

Determinação do ciclo

Fase	Micro-operação	Tempo (ns)
BI		
	REM←CP	3
	Decodifica REM	6
	RDM←M[REM]	10
	RI←RDM, inc(CP)	3
DI		
	Id. Cod. Oper.	12
	Gera Endereço	9
BD		
	REM←End.Dado	3
	Decodifica REM	6
	RDM←M[REM]	10
	Rx←RDM	3
EX		
	Ac←f{ULA}	22
	Ry←Ac	3
Total		90

T_{min} = 22,5ns x 1,05 + 4ns = 27,625ns ≈ 28ns



Determinação do ciclo

Fase	Micro-operação	Tempo (ns)	
BI			
	REM←CP	3	
	Decodifica REM	6	
	RDM←M[REM]	10	
	RI←RDM, inc(CP)	3	→ 22ns
DI			
	Id. Cod. Oper.	12	
	Gera Endereço	9	→ 21ns
BD			
	REM←End.Dado	3	
	Decodifica REM	6	
	RDM←M[REM]	10	
	Rx←RDM	3	→ 22ns
EX			
	Ac←f{ULA}	22	
	R		
		3	→ 25ns
T			

$T_{min} = 25ns \times 1,05 + 4ns = 30,25ns$

Dependências de dados

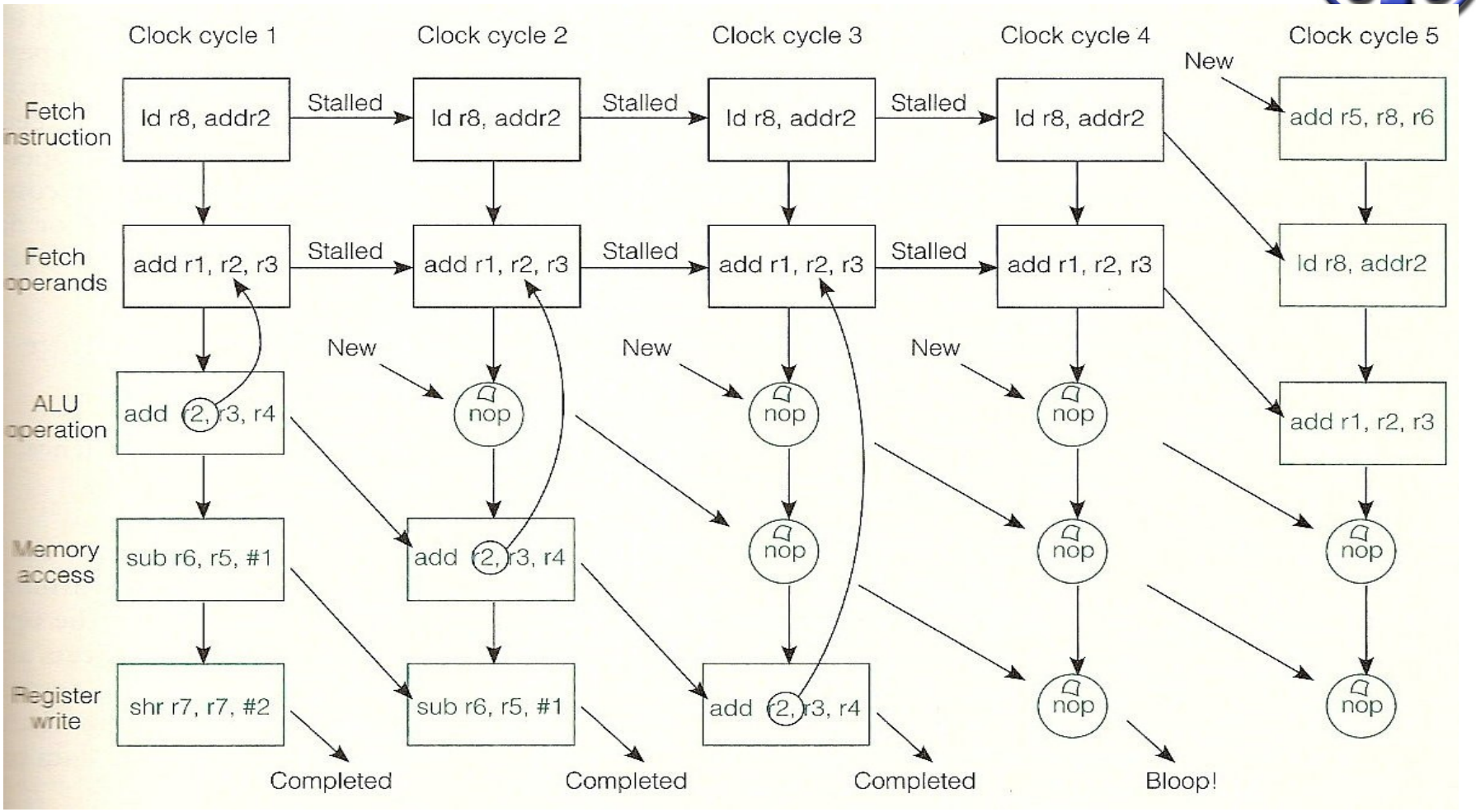
- *Hazards*
  - Instruções no *pipeline*
  - Resultados no *pipeline*

- Classificação
  - Dados
    - 100: add r0, r2, r4
    - 104: sub r3, r0, r1
  - Controle

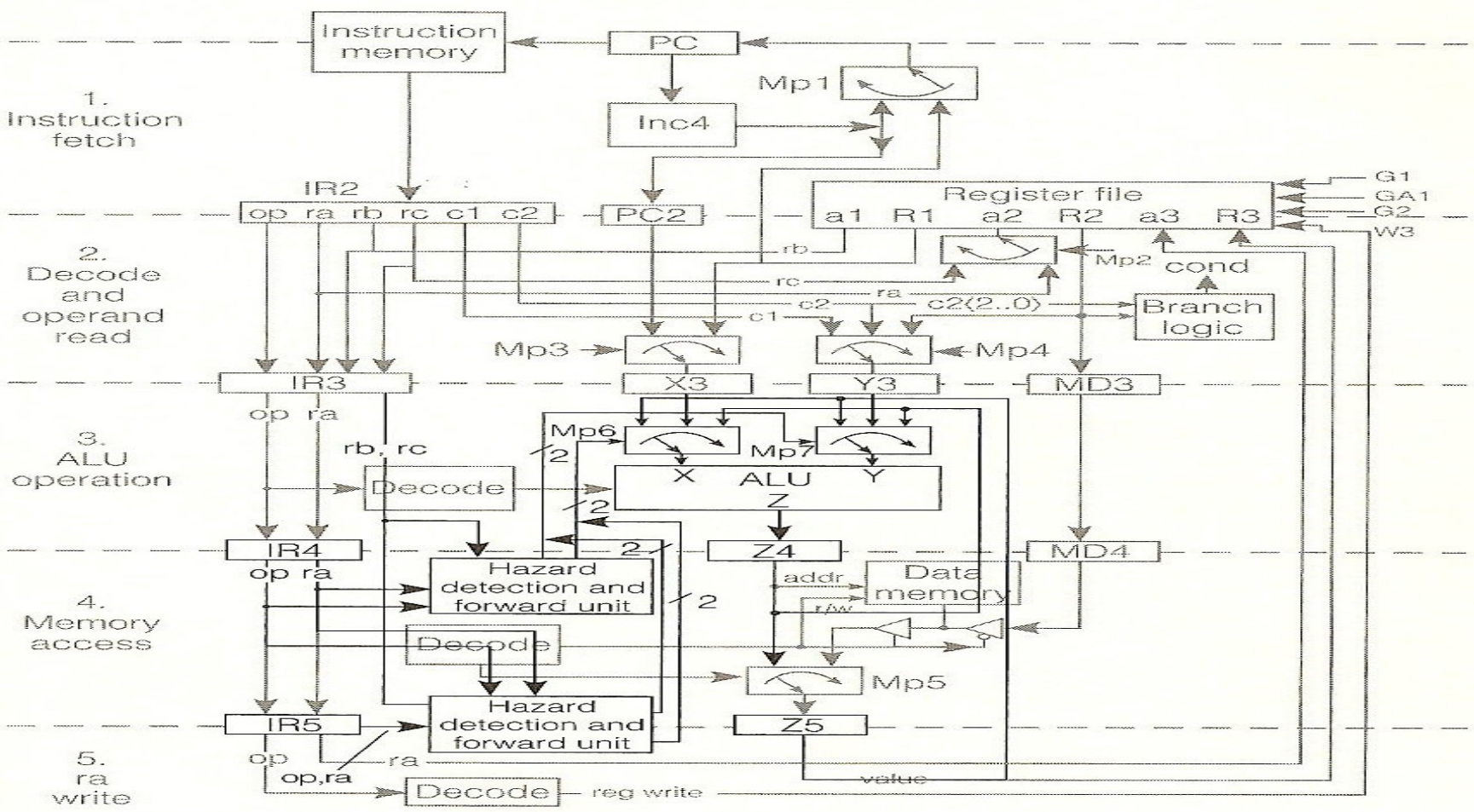
Dependências de dados

- *Hazards* de dados:
  - Leitura após escrita (RAW - *read after write hazard*)
  - Escrita após leitura (WAR - *write after read hazard*)
  - Escrita após escrita (WAW - *write after write hazard*)

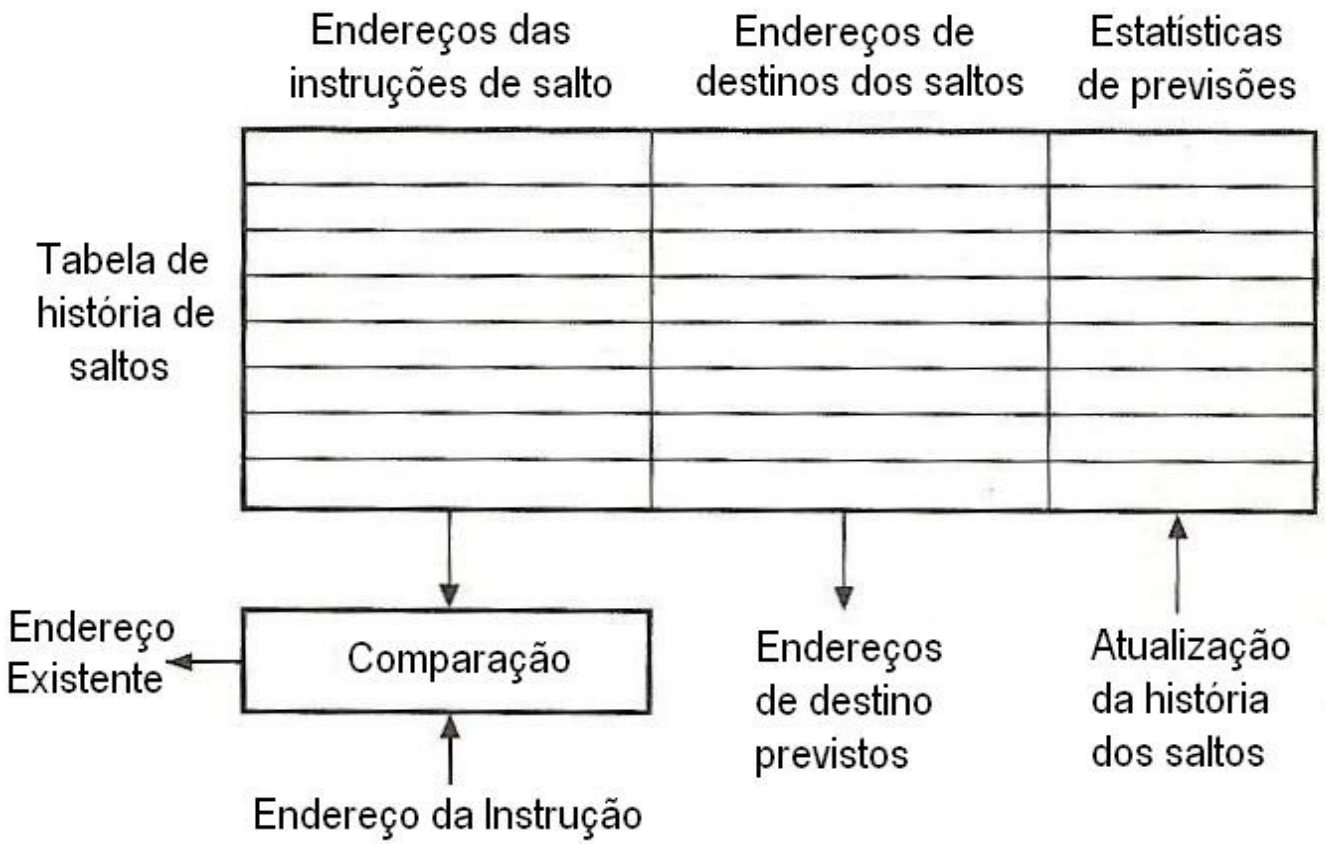
Criação de bolhas



Adiantamento de dados



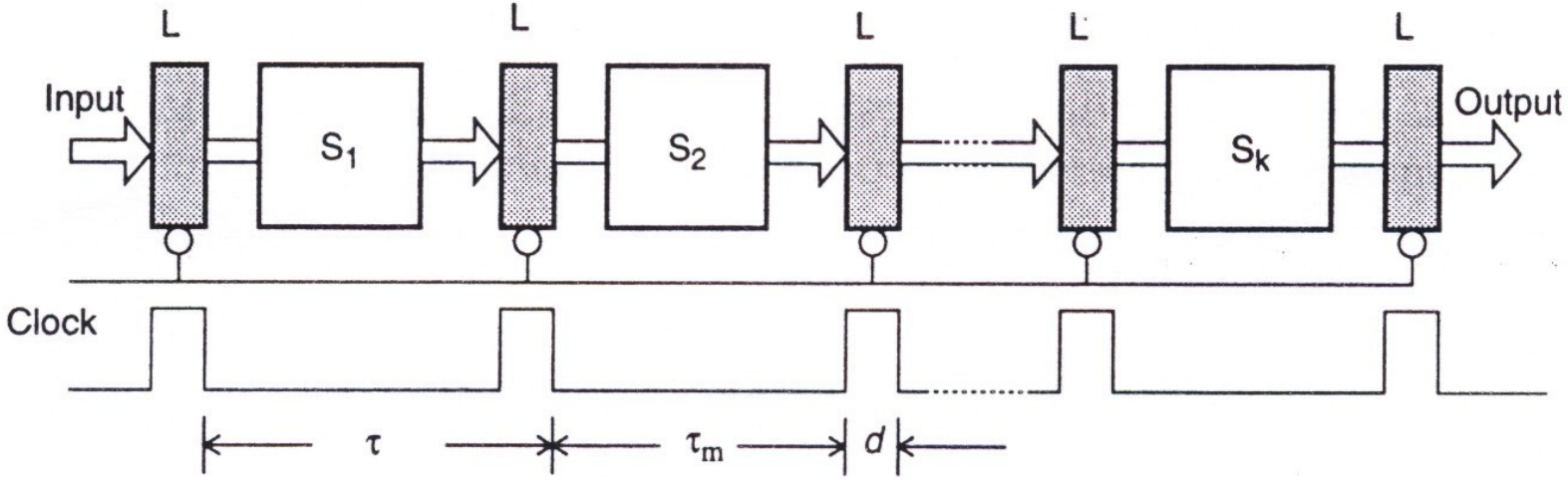
Previsão de saltos



Pipeline linear

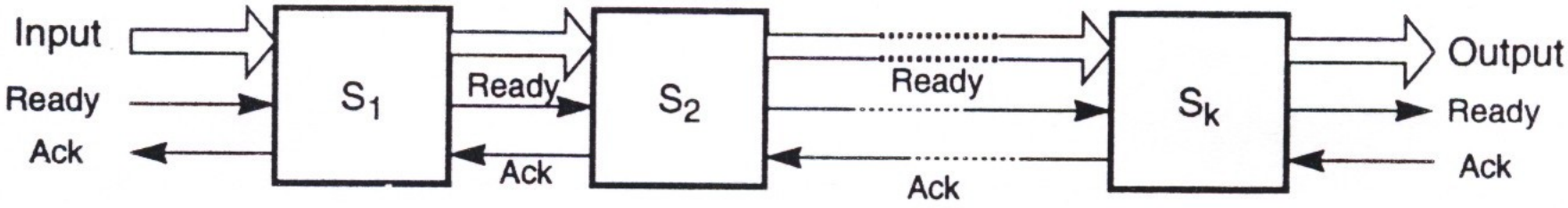
- É um pipeline em que o fluxo pelos estágios segue um padrão sequencial entre estágios
- É o pipeline típico que examinamos até agora
- Seu controle pode ser síncrono ou assíncrono

Pipeline linear síncrono



(b) A synchronous pipeline model

Pipeline linear assíncrono

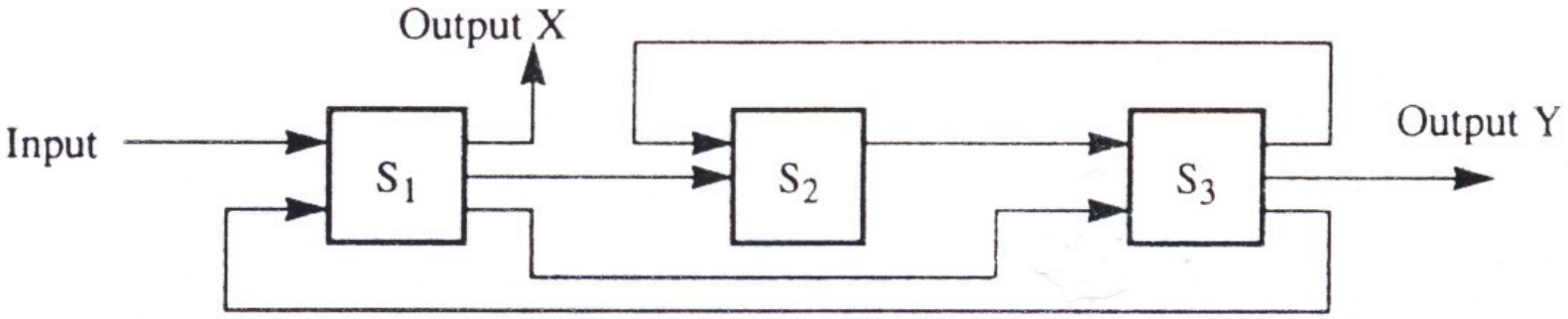


(a) An asynchronous pipeline model

Pipeline não-linear

- É um pipeline em que o fluxo pelos estágios pode sofrer grandes desvios ou até formar ciclos
- É típico de processadores CISC, pois instruções de formatos diferentes demandam caminhos de execução diferentes
- Seu controle é tipicamente assíncrono, demandando tabelas de reserva

Exemplo pipeline não-linear assíncrono



(a) A three-stage pipeline

Pipeline reconfigurável (não-linear)

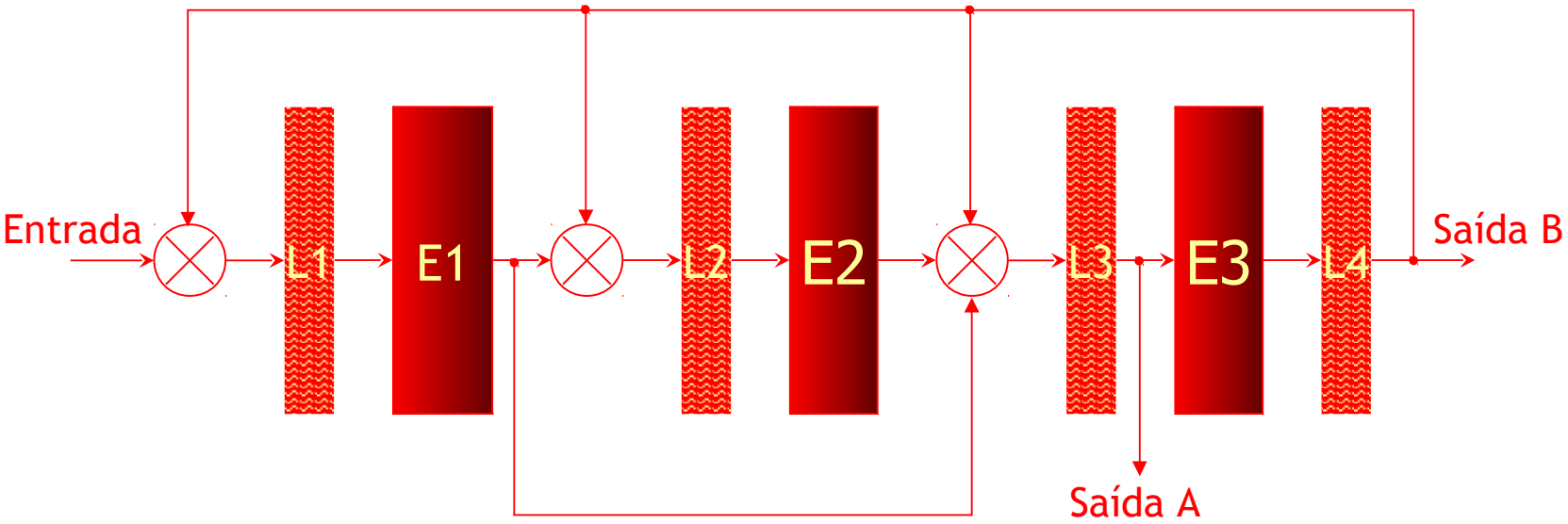


Tabela de reserva

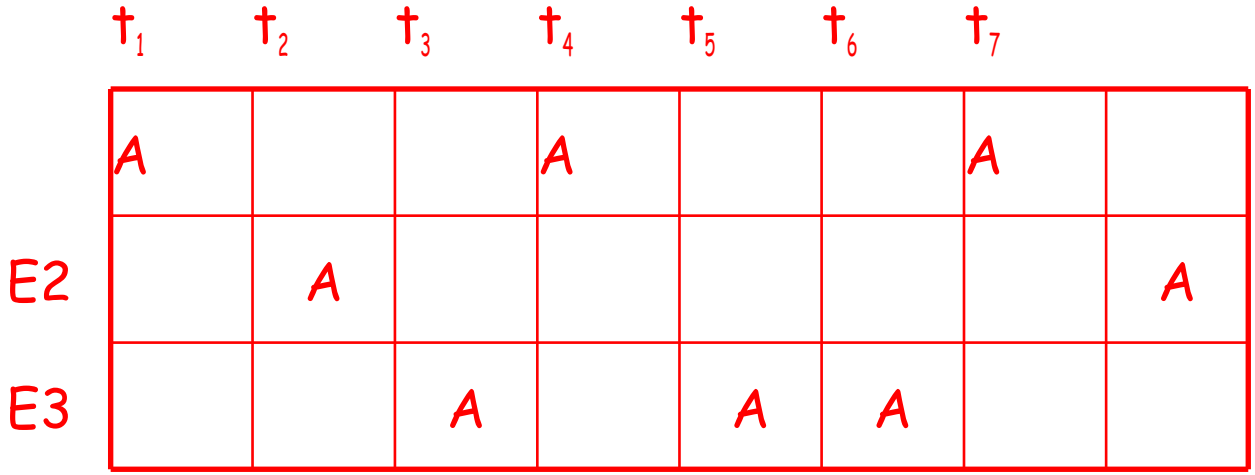
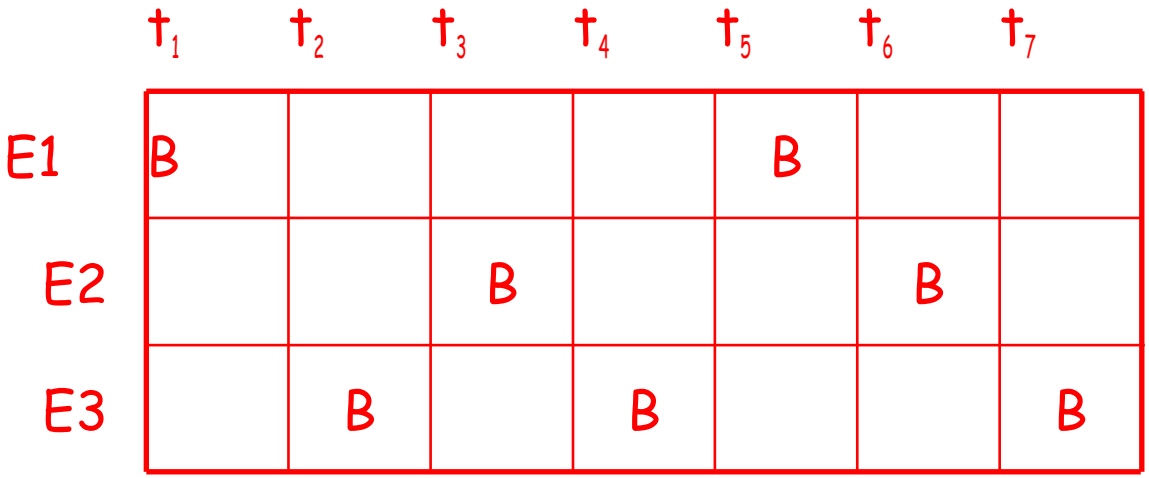


Tabela de reserva

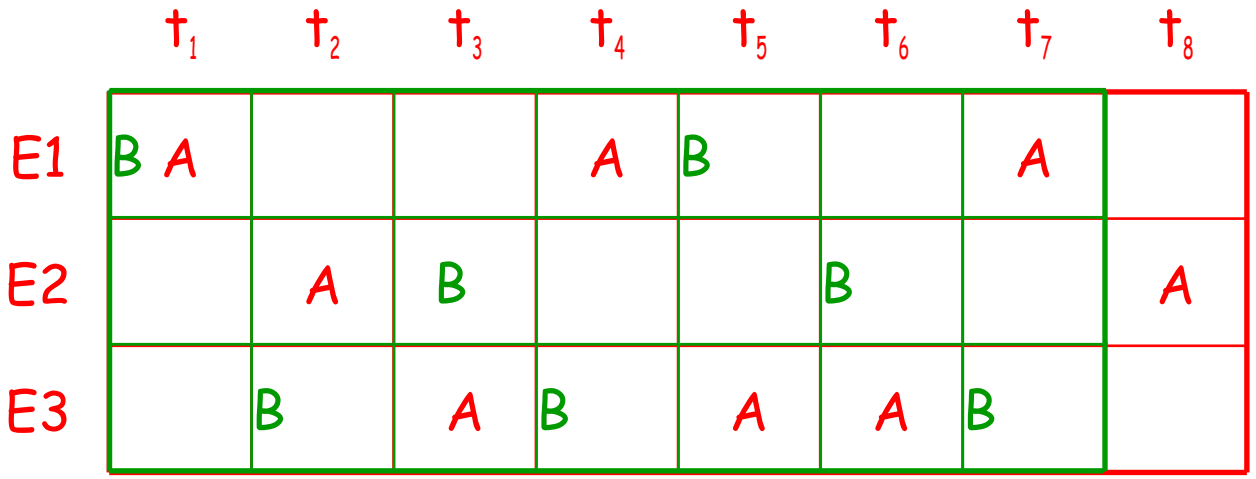


	t_1	t_2	t_3	t_4	t_5	t_6	t_7
E1	B				B		
E2			B			B	
E3		B		B			B

Vetor de colisões

- Quando é seguro iniciar uma nova função.
- Número de posições igual ao de períodos.
- $VC_{ij}(x)$
  - 1 → há colisão
  - 0 → não há colisão

Vetor de colisões



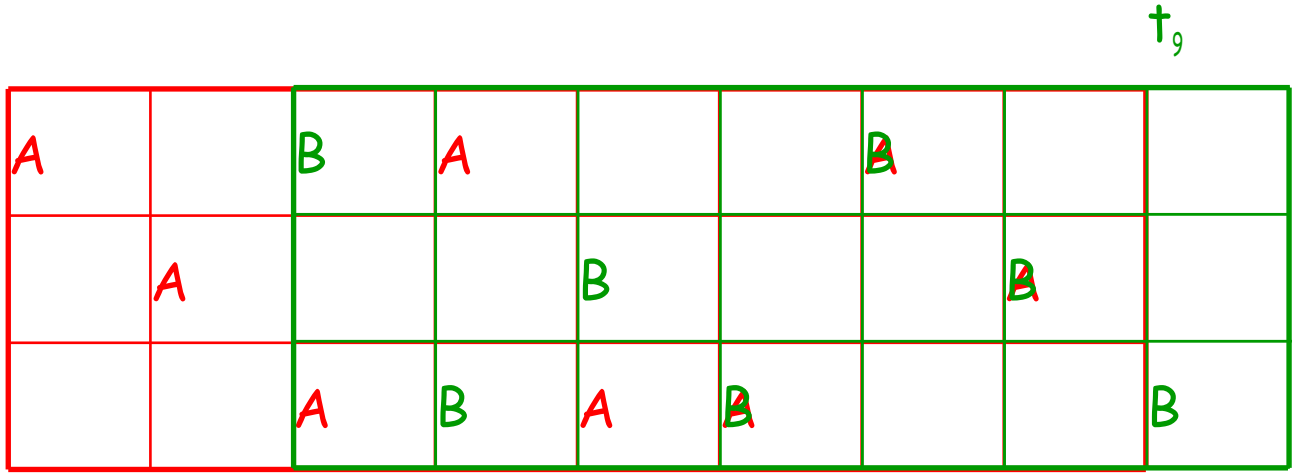
VC_{AB} = (1 _ _ _ _ _ _ _ _)

Vetor de colisões

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8
E1	A	B		A		B	A	
E2		A		B			B	A
E3			B A		B A	A		B

VC_{AB} = (1 1 _ _ _ _ _ _)

Vetor de colisões



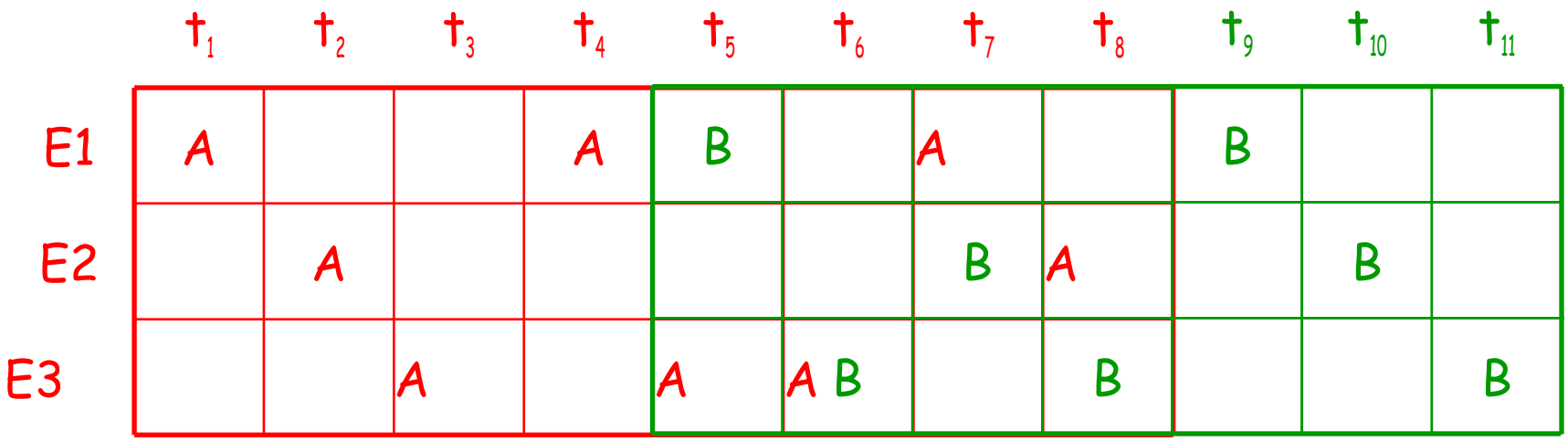
VC_{AB} = (1 1 1 _ _ _ _ _ _)

Vetor de colisões

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
E1	A			B			A	B		
E2		A				B		A	B	
E3			A		B	A	B			B

VC_{AB} = (1 1 1 1 _ _ _ _)

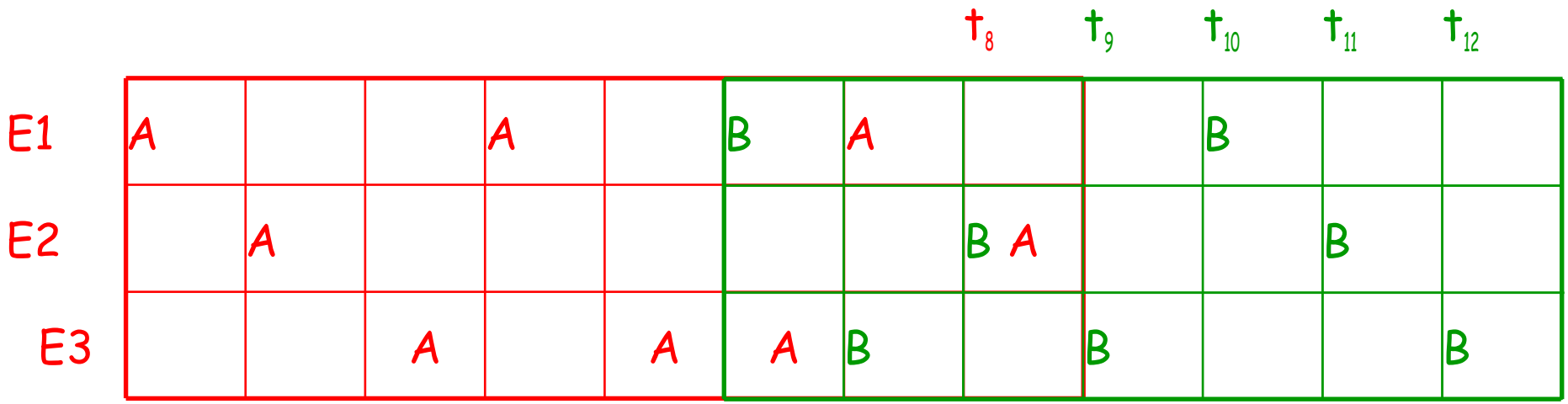
Vetor de colisões



	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}
E1	A			A	B		A		B		
E2		A					B	A		B	
E3			A		A	A B		B			B

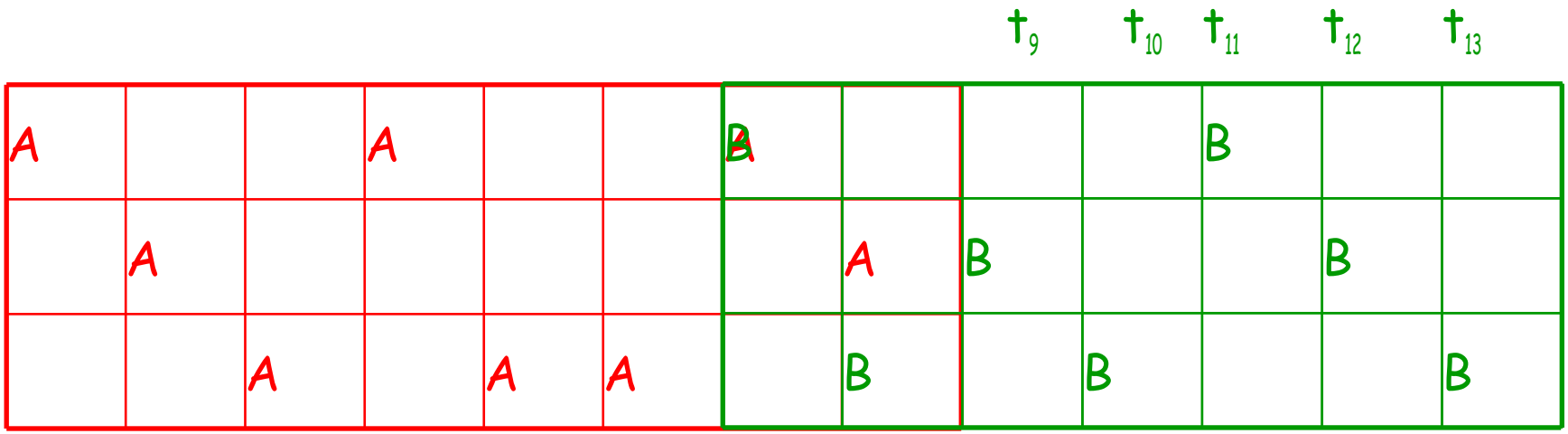
VC_{AB} = (1 1 1 1 1 _ _ _)

Vetor de colisões



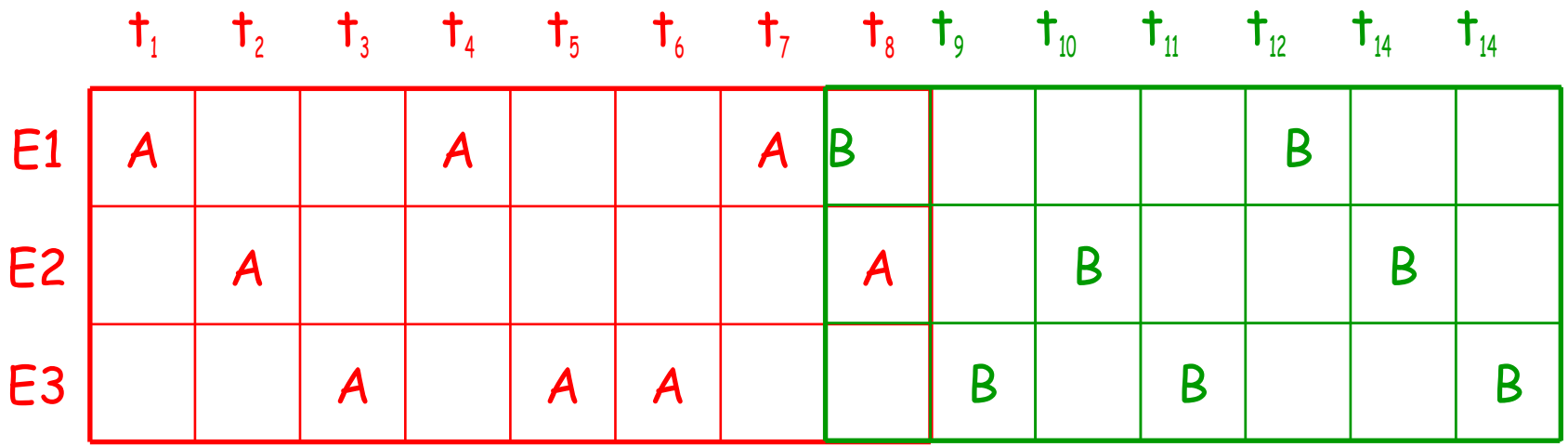
VC_{AB} = (1 1 1 1 1 1 _ _)

Vetor de colisões



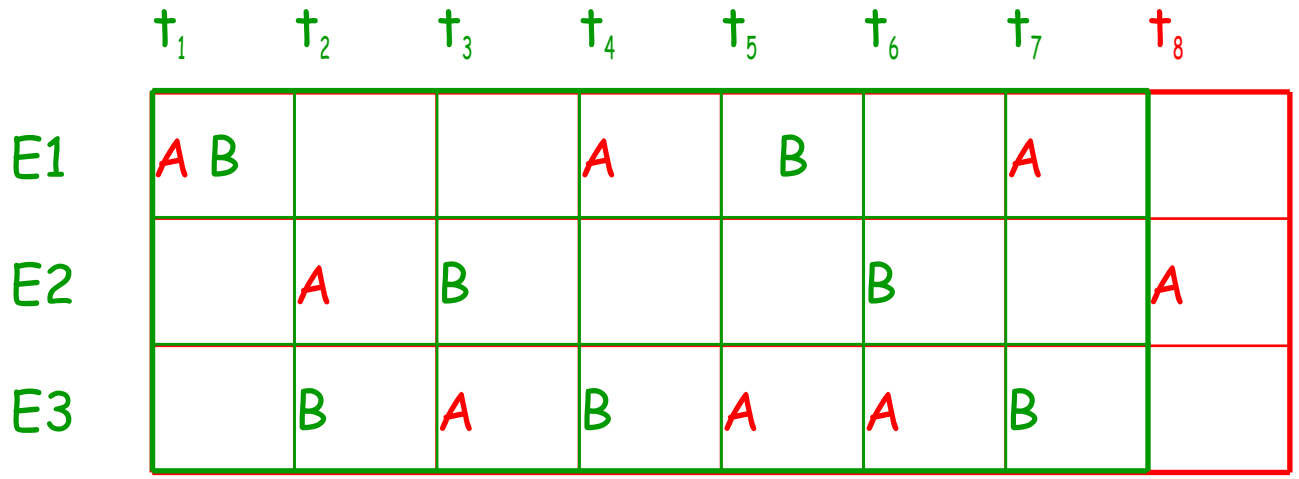
VC_{AB} = (1 1 1 1 1 1 1 _)

Vetor de colisões



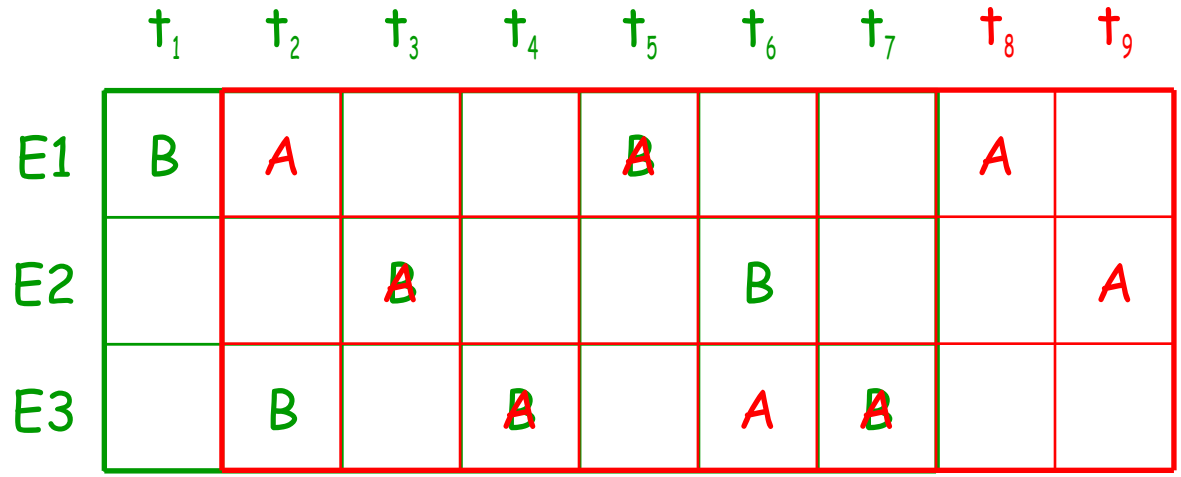
VC_{AB} = (1 1 1 1 1 1 1 0)

Vetor de colisões



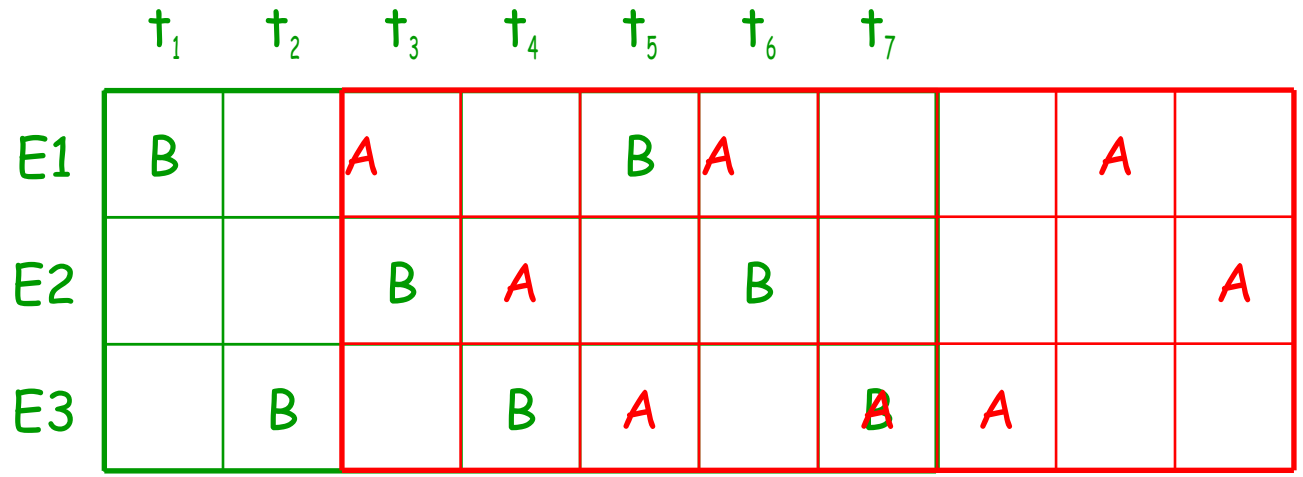
VC_{BA} = (1 _ _ _ _ _ _ _)

Vetor de colisões



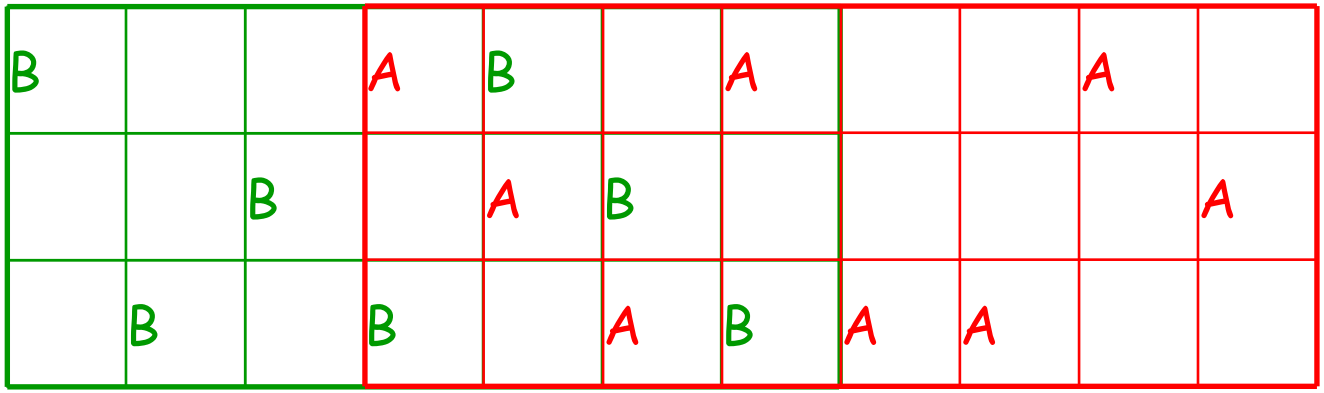
VC_{BA} = (1 1 _ _ _ _ _ _)

Vetor de colisões



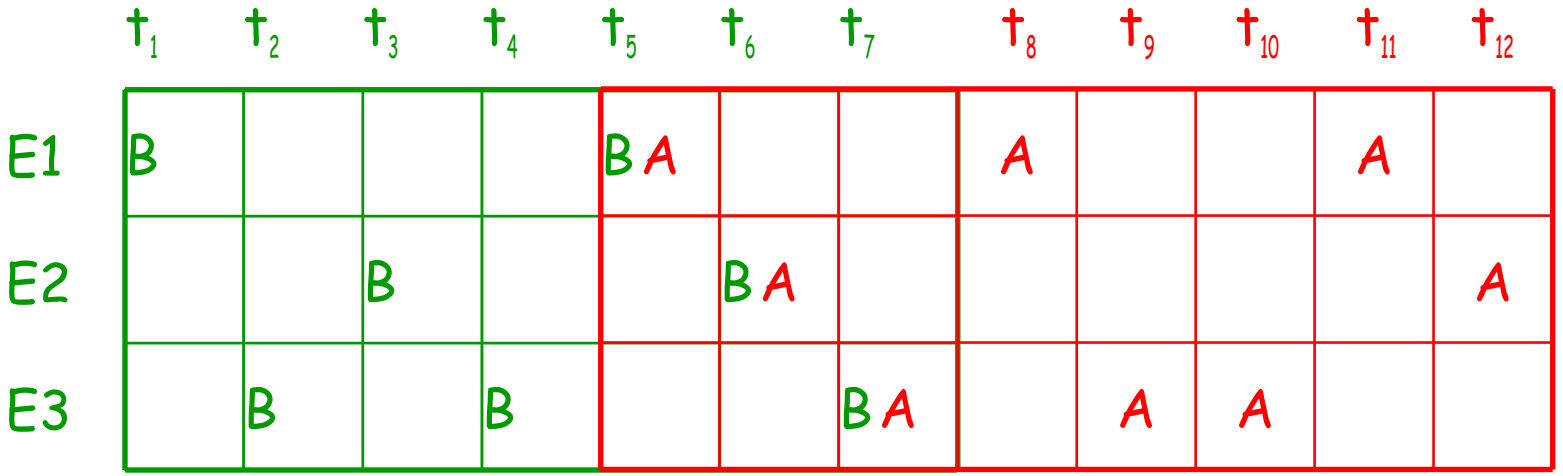
VC_{BA} = (1 1 1 _ _ _ _)

Vetor de colisões



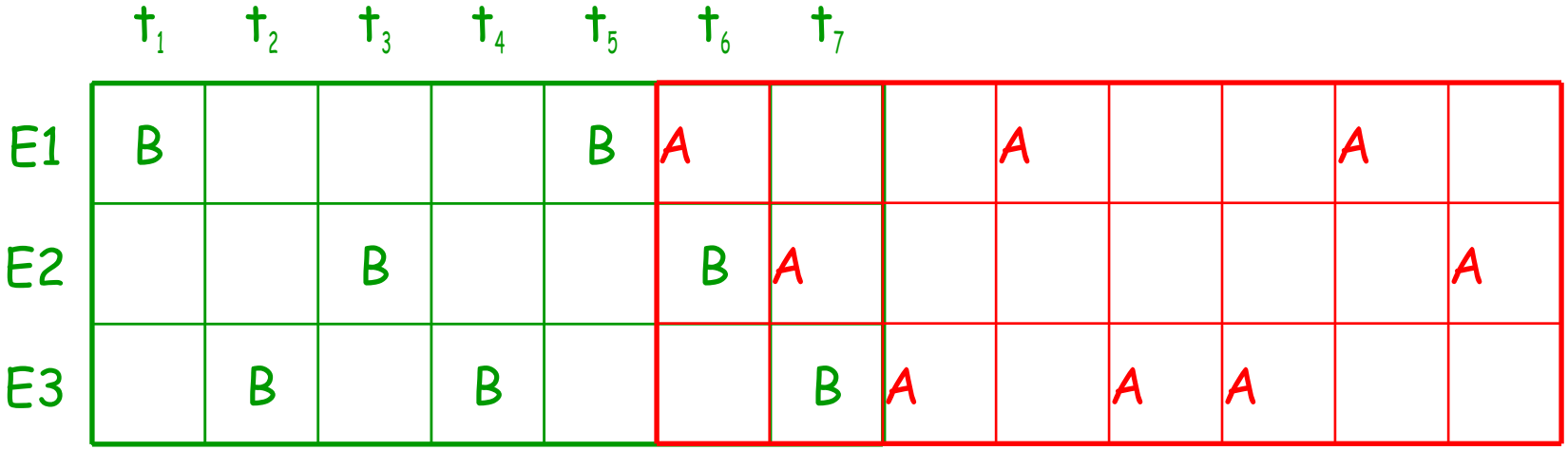
VC_{BA} = (1 1 1 0 _ _ _)

Vetor de colisões



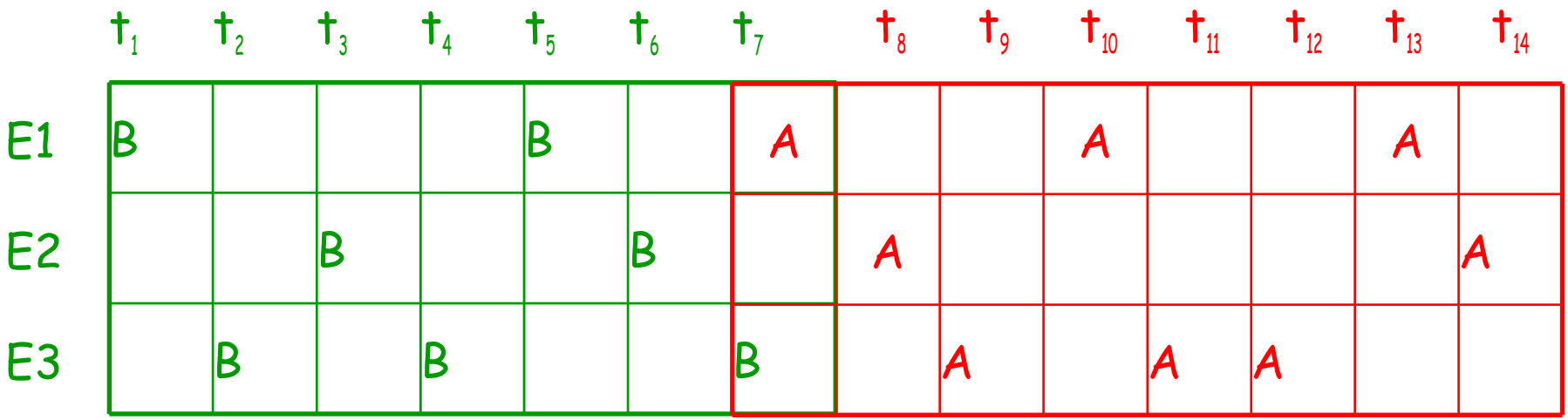
VC_{BA} = (1 1 1 0 1 _ _)

Vetor de colisões



VC_{BA} = (1 1 1 0 1 0 _)

Vetor de colisões



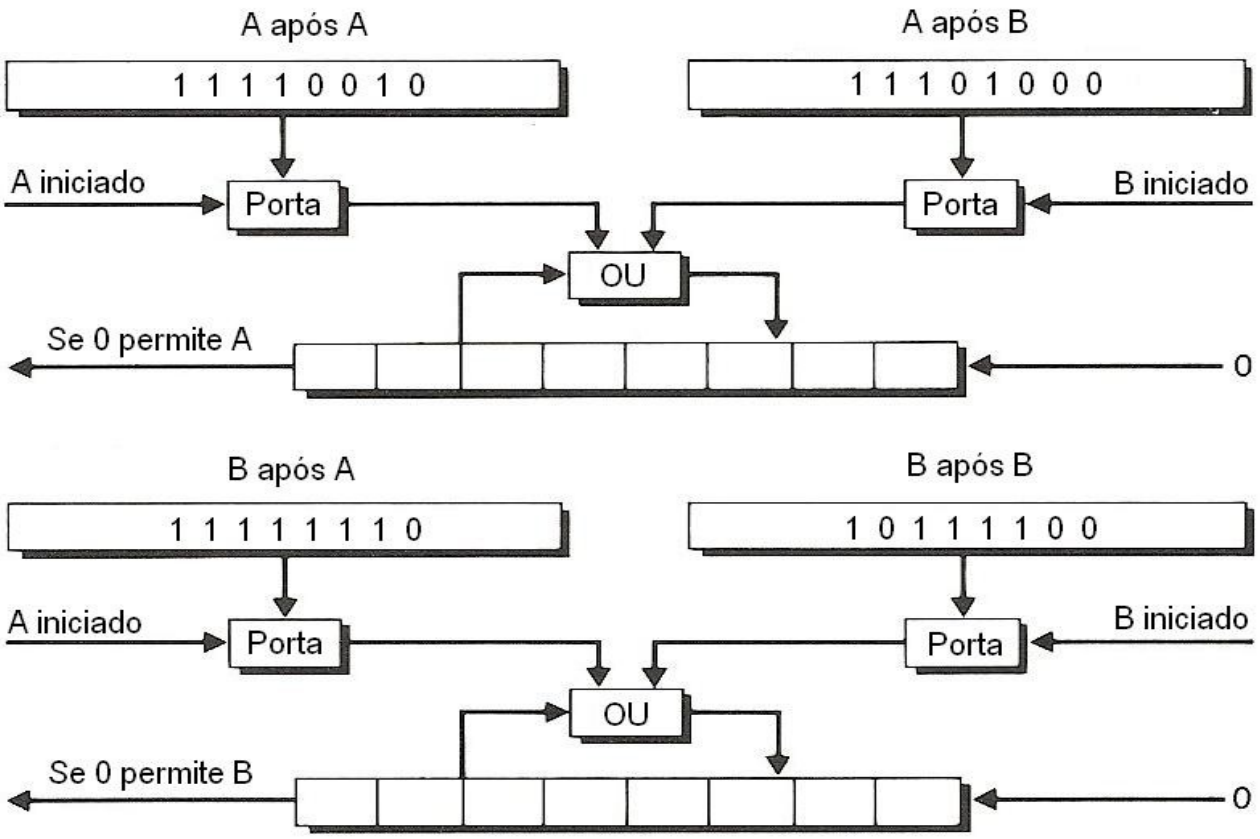
VC_{BA} = (1 1 1 0 1 0 0)

Vetor de colisões

- Como usar?

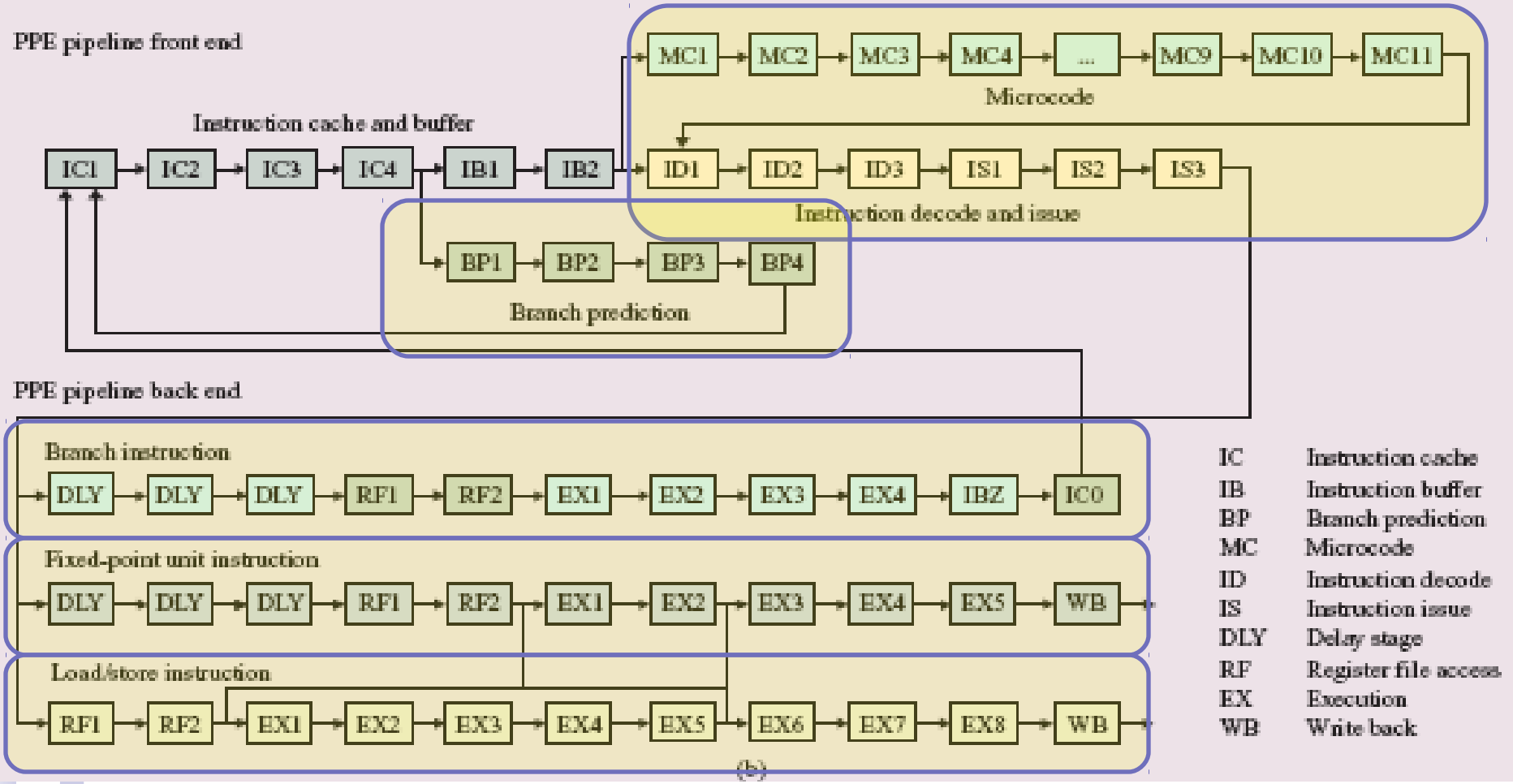
1	1	1	0	1	0	0	0

Vetor de colisões

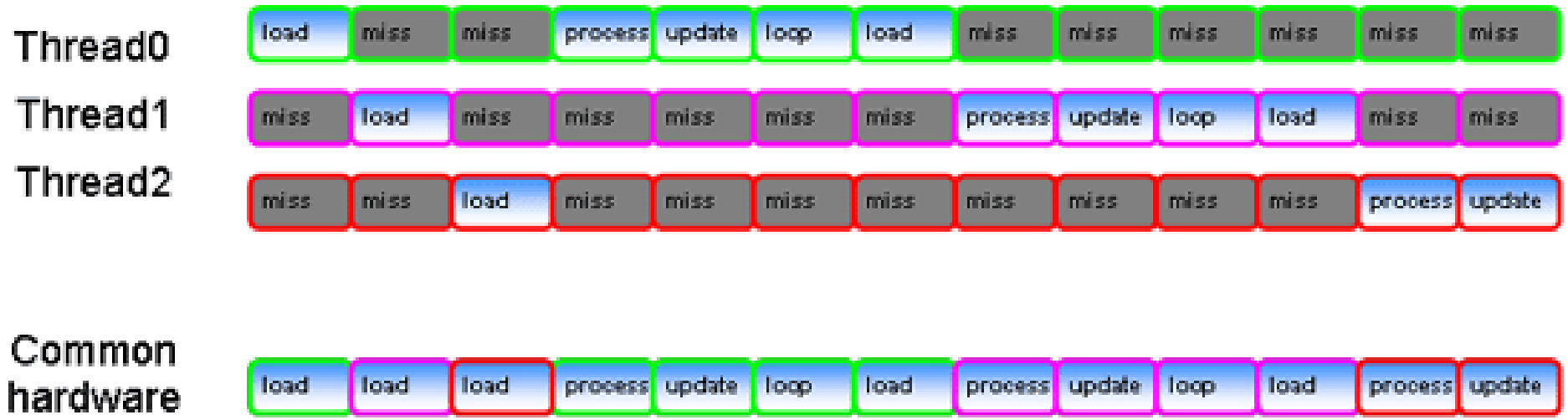


Exemplos de pipelines reais

Pipeline família Power



Pipeline superescalar (threads)



Pipeline processadores MIPS