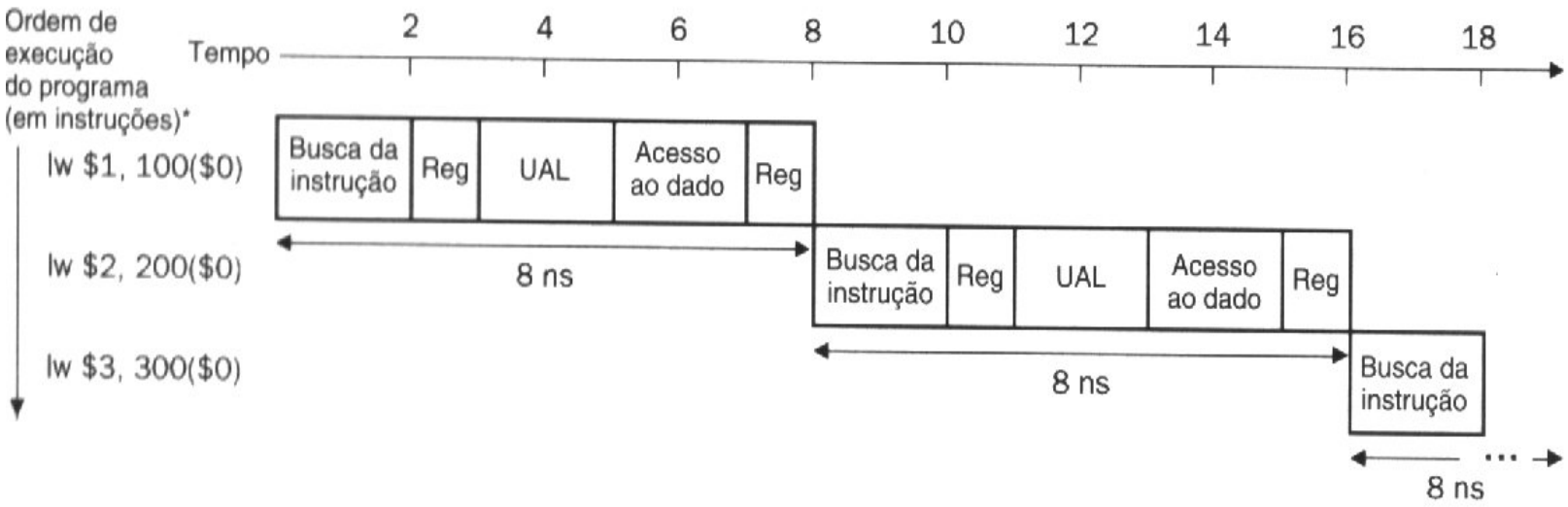
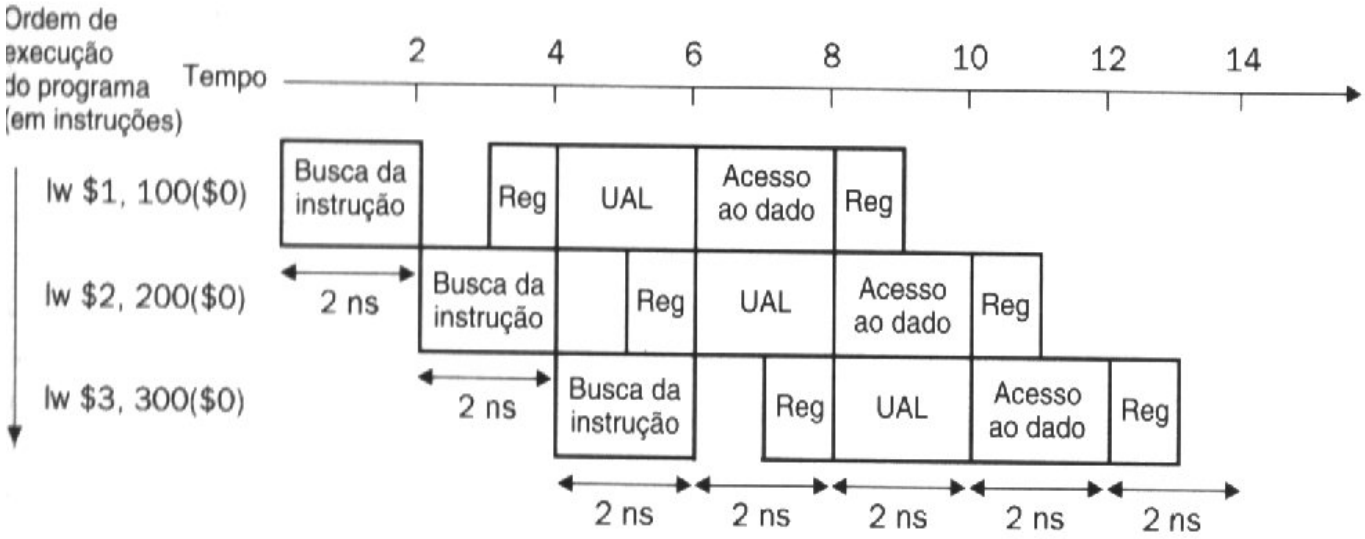
Exemplo de Pipeline de Instruções

Classe da Instrução	Busca da Instrução	Leitura Operando	Operação da ULA	Acesso à Memória	Escrita do Resultado	Total
Load Word (lw)	2 ns	1 ns	2 ns	2 ns	1 ns	8 ns
StoreWord (sw)	2 ns	1 ns	2 ns	2 ns		7 ns
Aritméticas (add, sub, and)	2 ns	1 ns	2 ns		1ns	6 ns
Branch (beqz)	2 ns	1 ns	2 ns			5 ns


Ordem de execução do programa (em instruções)*
Tempo
2
4
6
8
10
12
14
16
18
lw $1, 100($0)
lw $2, 200($0)
lw $3, 300($0)
Busca da instrução
Reg
UAL
Acesso ao dado
Reg
8 ns
8 ns
8 ns


Ordem de execução do programa (em instruções)
Tempo
2
4
6
8
10
12
14
lw $1, 100($0)
lw $2, 200($0)
lw $3, 300($0)
Busca da instrução
Reg
UAL
Acesso ao dado
Reg
2 ns
2 ns
2 ns
2 ns
2 ns
2 ns
2 ns

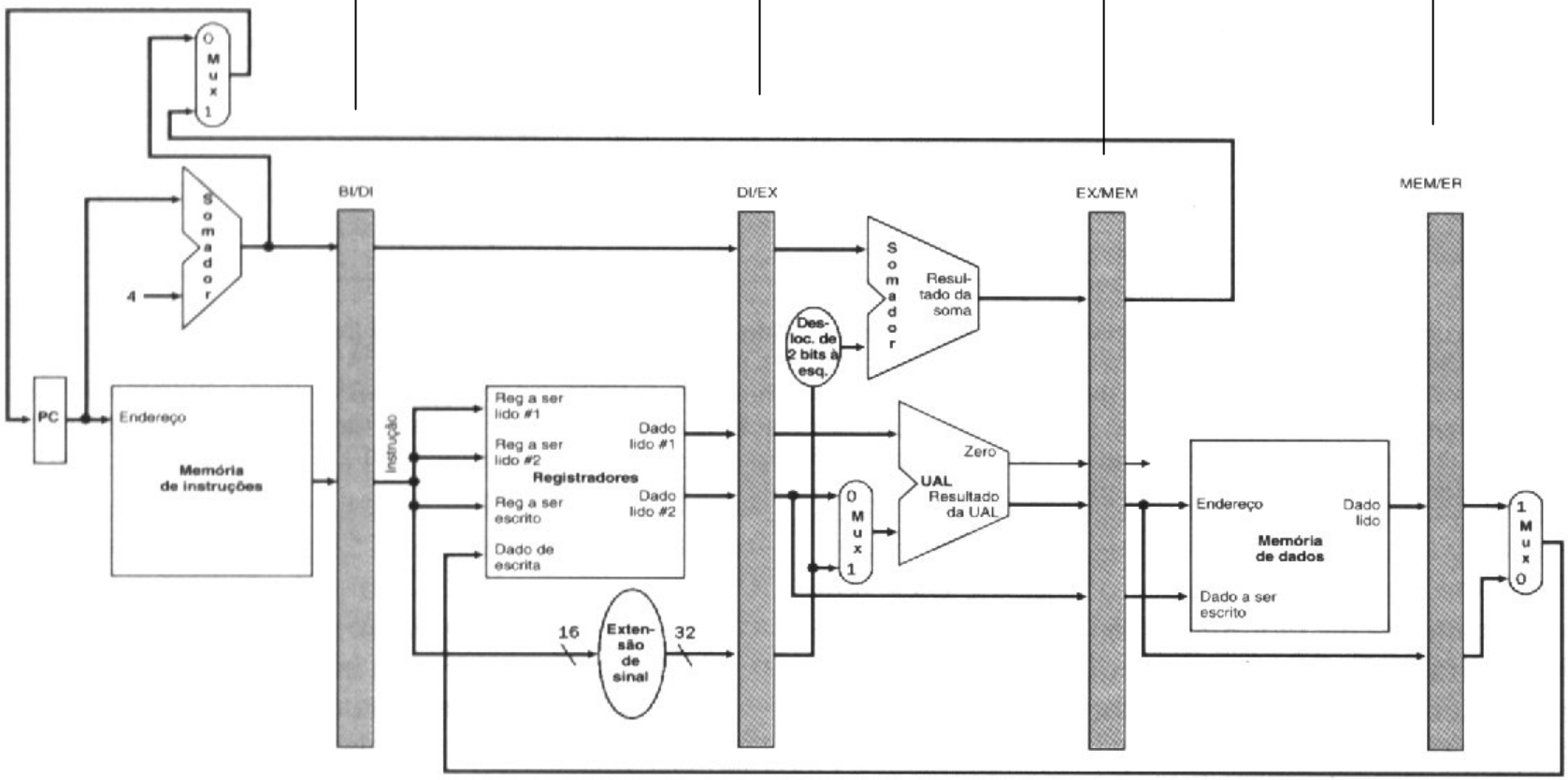
Características dos Pipelines de Instrução

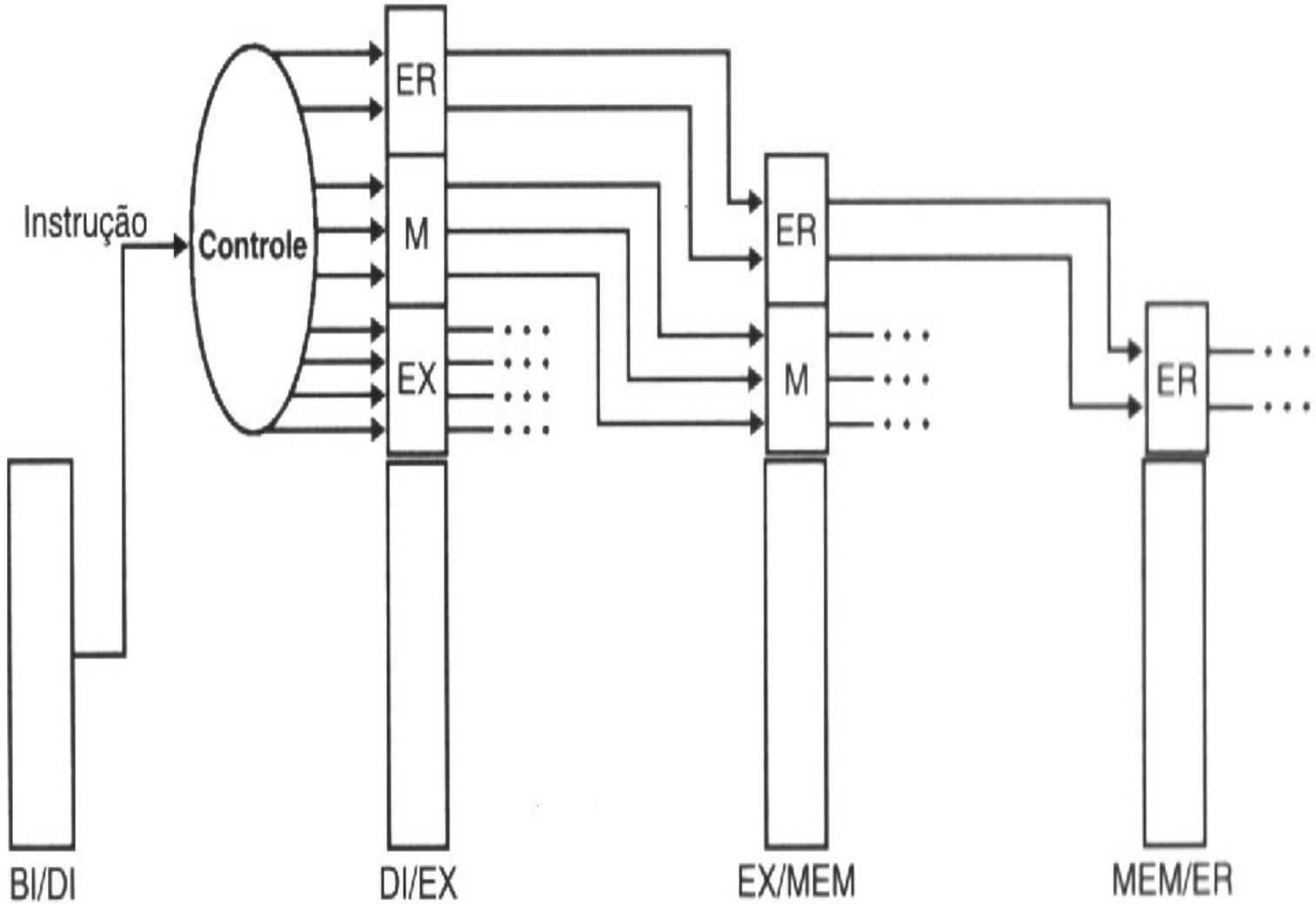
- O tempo do ciclo do relógio do processador deve ser igual ao tempo de execução do estágio mais lento do "pipeline".
- Deve-se procurar dividir a execução da instrução em estágios com o mesmo tempo de execução.
- O pipeline deve ser mantido sempre "cheio" para que o desempenho máximo seja alcançado.
- Cada instrução, individualmente, continua levando o mesmo tempo para ser executada.

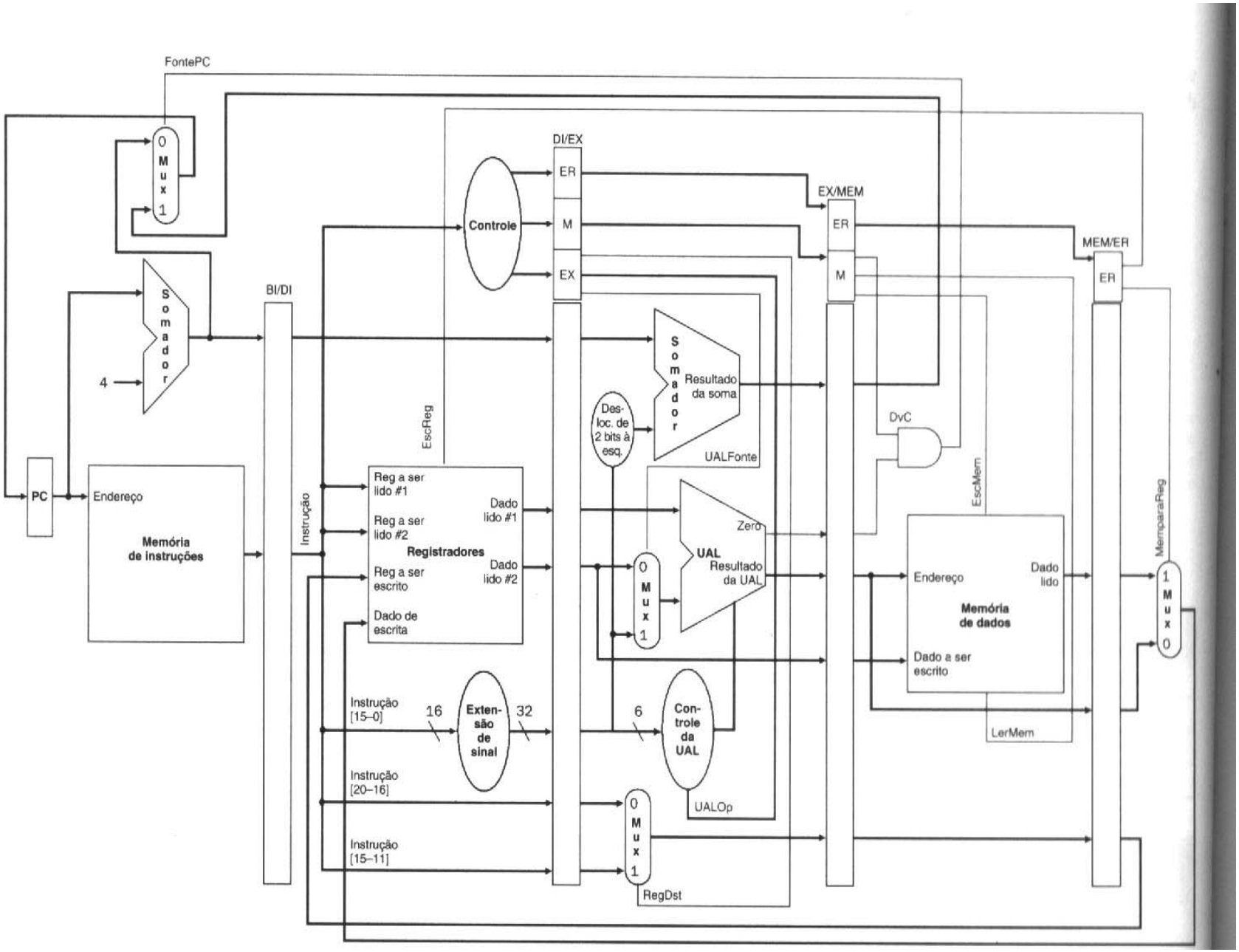
Divisão da Execução da Instrução em 5 estágios:

- Busca da instrução na memória (B)
- Leitura dos registradores e decodificação da instrução (D)
- Execução da instrução / cálculo do endereço de memória (E)
- Acesso a um operando na memória (M)
- Escrita do resultado em um registrador (W)

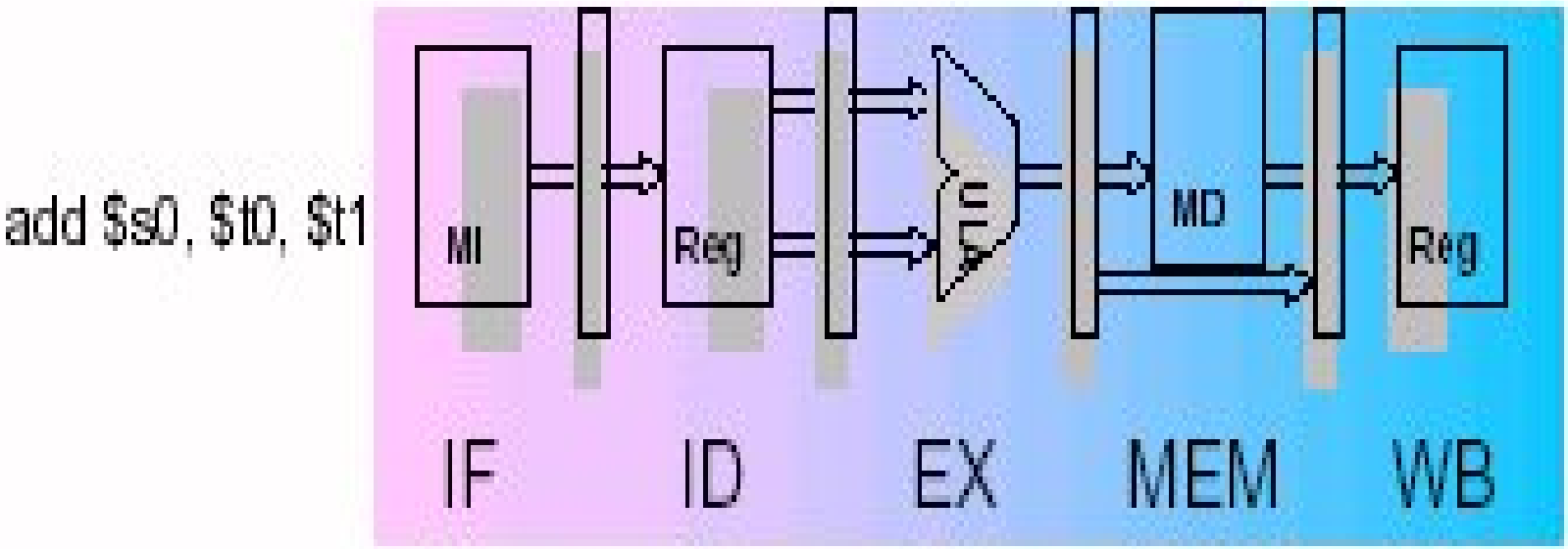
Busca de instrução	decodificação leitura de resgistrador	execução cálculo de end.	acesso a memória	escrita em Regist.

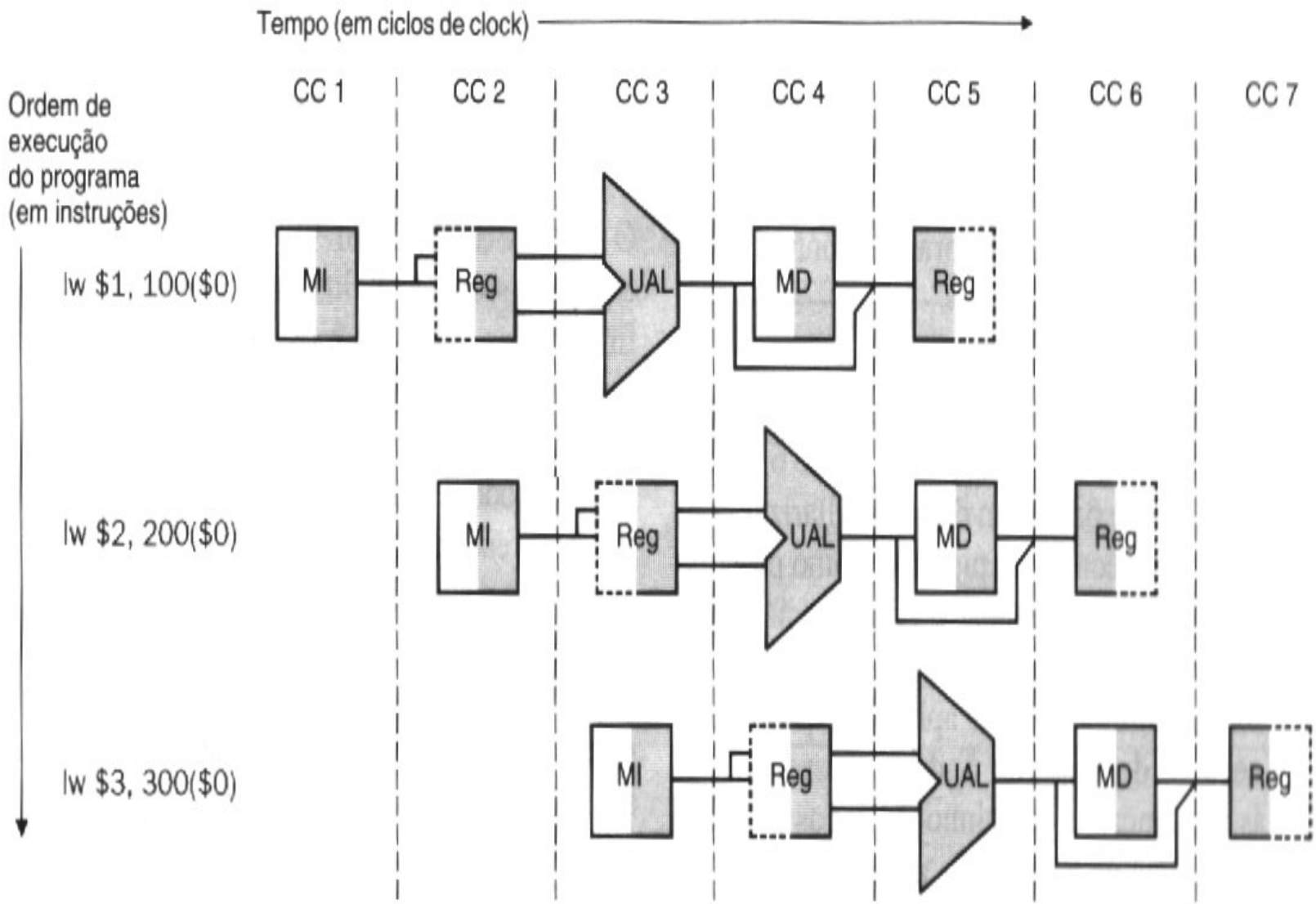



Instrução
Controle
ER
M
EX
ER
M
ER
BI/DI
DI/EX
EX/MEM
MEM/ER


FontePC
0
M
u
x
1
Somador
4
BI/DI
Controle
DI/EX
ER
M
EX
EX/MEM
ER
M
MEM/ER
ER
Somador
Resultado da soma
UALFonte
Des-loc. de 2 bits à esq.
DvC
EscReg
PC
Endereço
Memória de instruções
Instrução
Reg a ser lido #1
Reg a ser lido #2
Registradores
Reg a ser escrito
Dado de escrita
Dado lido #1
Dado lido #2
0
M
u
x
1
Zero
UAL
Resultado da UAL
EscMem
Endereço
Dado lido
Memória de dados
Dado a ser escrito
LerMem
MemparaReg
1
M
u
x
0
Instrução [15–0]
16
Extensão de sinal
32
6
Controle da UAL
UALOp
Instrução [20–16]
Instrução [15–11]
0
M
u
x
1
RegDst

Simplificação do diagrama




Tempo (em ciclos de clock)
CC 1
CC 2
CC 3
CC 4
CC 5
CC 6
CC 7
Ordem de execução do programa (em instruções)
lw $1, 100($0)
lw $2, 200($0)
lw $3, 300($0)
MI
Reg
UAL
MD
Reg
MI
Reg
UAL
MD
Reg
MI
Reg
UAL
MD
Reg

Desempenho. O que pode dar errado.

Um processador com *pipeline* pode pausar por várias razões, incluindo atrasos na leitura da memória, conjunto de instruções mal projetado, dependência entre instruções, ...

Latência de instruções

Instruções que necessitam ler e escrever no mesmo registro. Por exemplo "ADD 5 ao registro 3", deve ler o registro 3, adicionar 5 ao valor lido e escrever o resultado de volta ao mesmo registro (que pode estar ocupado pela operação de leitura anterior causando uma pausa no processador até o registro ficar disponível).

Dependência de uma única origem como o registro de código de condição. Se uma instrução altera alguns bits do registro de estado e a seguinte tenta lê-los, ela deve aguardar até que a anterior termine.

Problemas no Uso de Pipelines

- Existem situações em que a próxima instrução não pode ser executada no ciclo seguinte do relógio:
  - Conflitos Estruturais
    - Pode haver acessos simultâneos à memória feitos por 2 ou mais estágios.
  - Dependências de Dados
    - As instruções dependem de resultados de instruções anteriores, ainda não completadas.
  - Dependências de Controle
    - A próxima instrução não está no endereço subseqüente ao da instrução anterior.

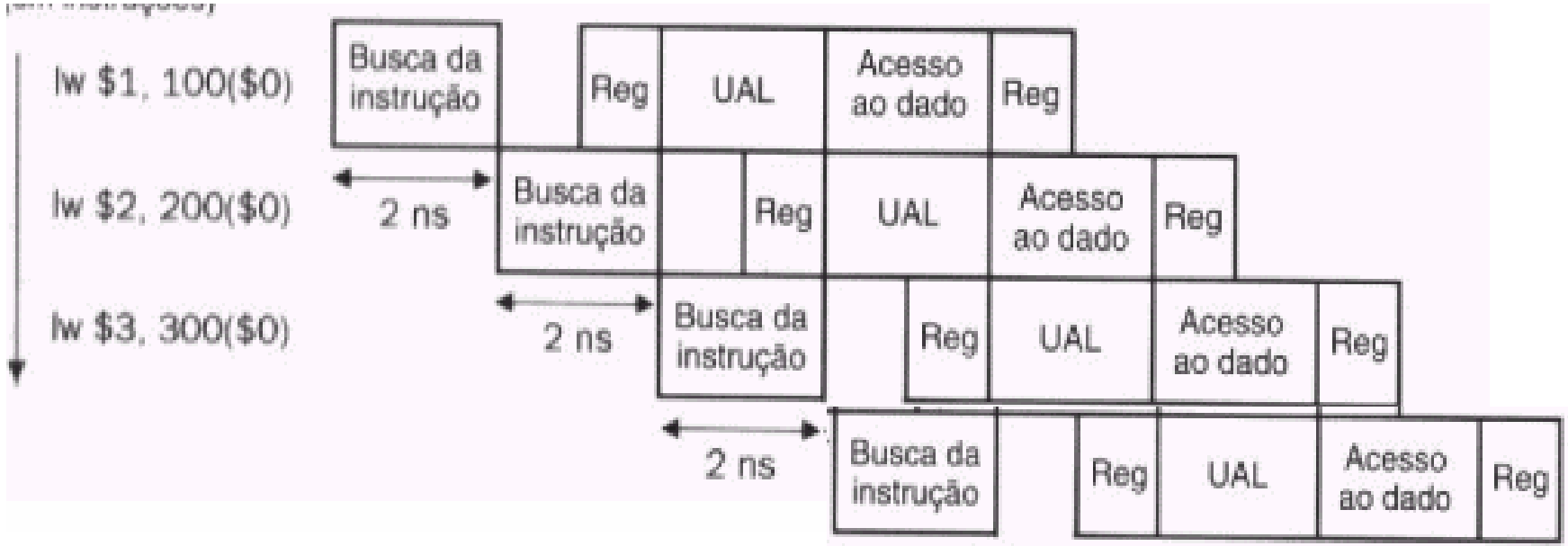
Stall (bubble)

Muitas vezes algumas instruções consomem mais de um *clock* para completar um estágio.

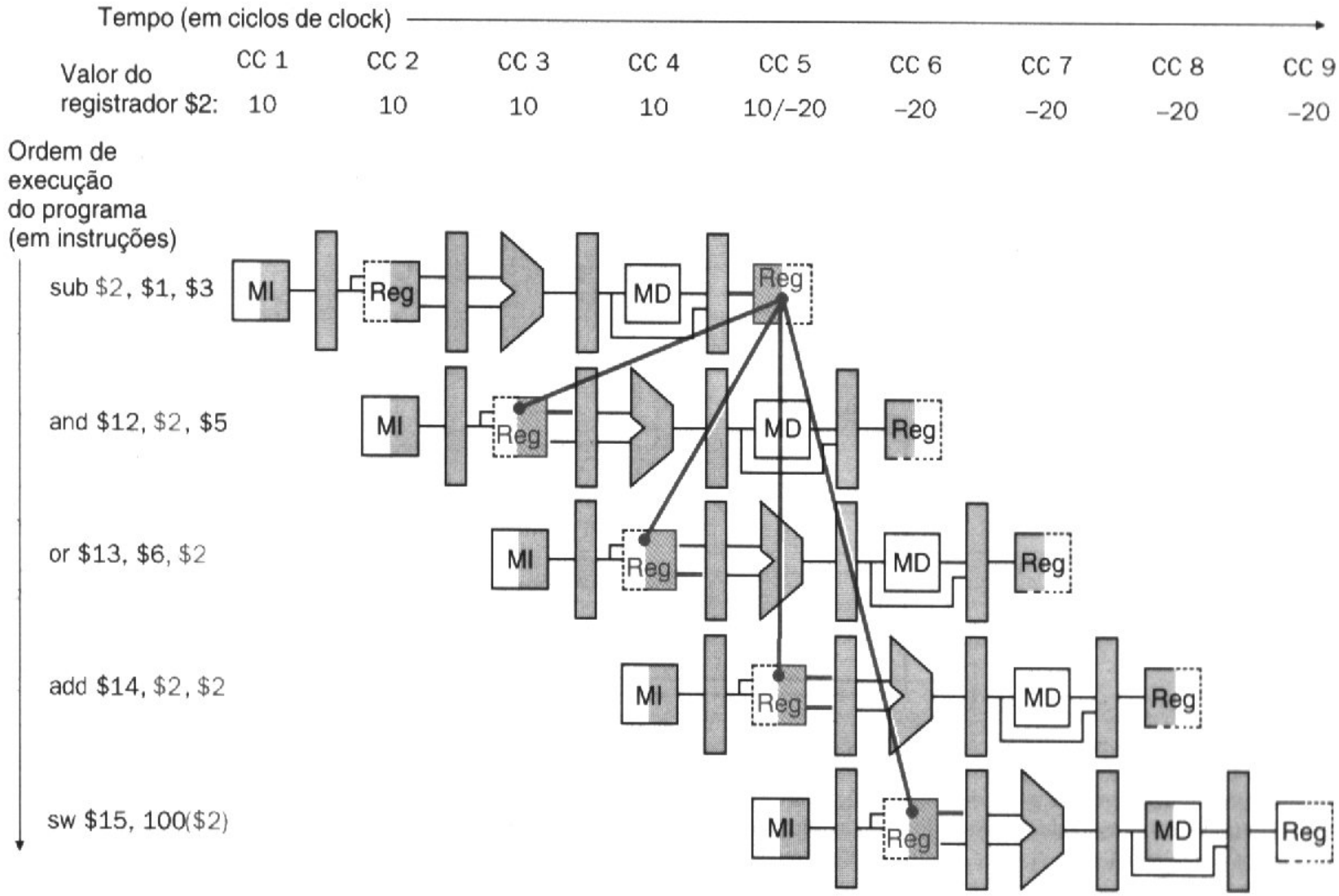
Quando isto acontece, o processador deve introduzir uma pausa e não aceitar novas instruções até que a instrução lenta tenha movido para o próximo estágio.

Conflito Estrutural:

A arquitetura não suporta a combinação de instruções que se quer executar no mesmo ciclo.



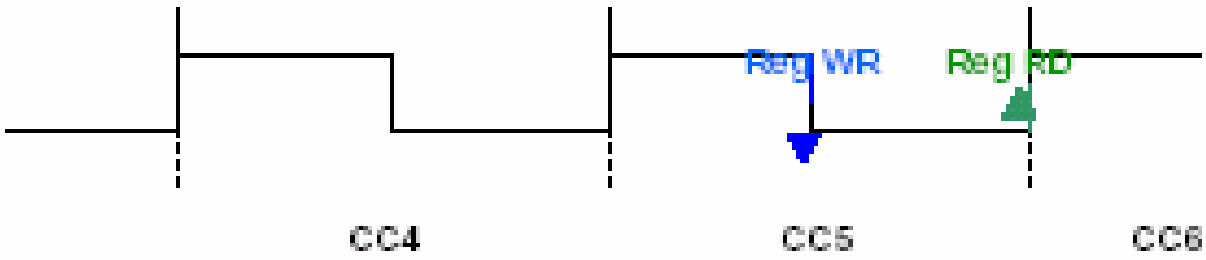
Dependência de dados:



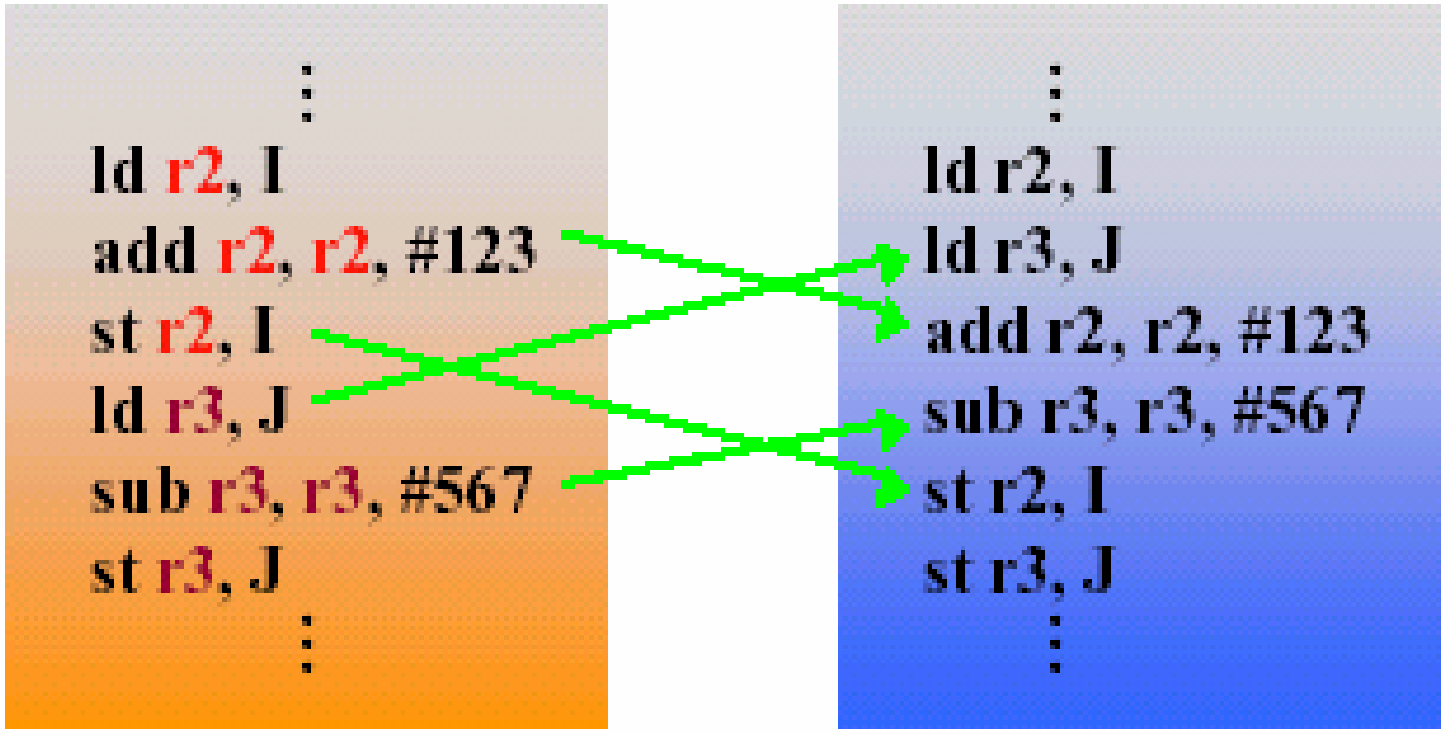
Erro devido à dependência de dados

```
sub   $2,  $1, $3
and   $12, $2, $5
or    $13, $6, $2
add   $14, $2, $2
sw    $15, 100($2)
```

- *and* e *or* lêem resultado errado (velho 10)
- store está claramente à direita (depois no tempo) e lê resultado certo (-20)
- hazard no add pode ser evitado se a escrita no banco de registradores for feita (em CC5) na metade do ciclo (borda de descida)



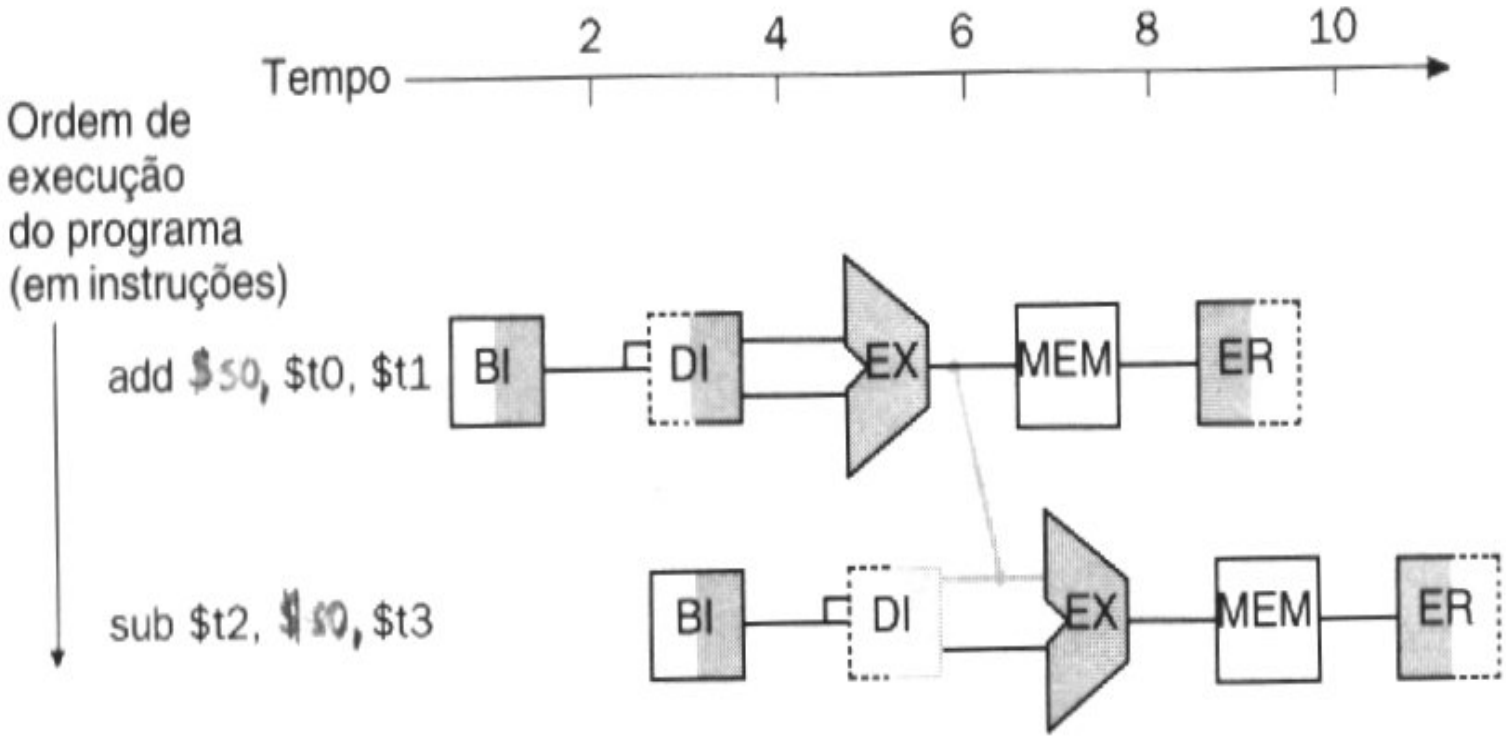
Solução por *software*: escalonamento de instruções



Um rearranjo das instruções no programa, chamado escalonamento de instruções, pode resolver esta pendência.

Uma solução para (Data Harzad) dependência de dados:

Adiantamento do resultado(forwarding)

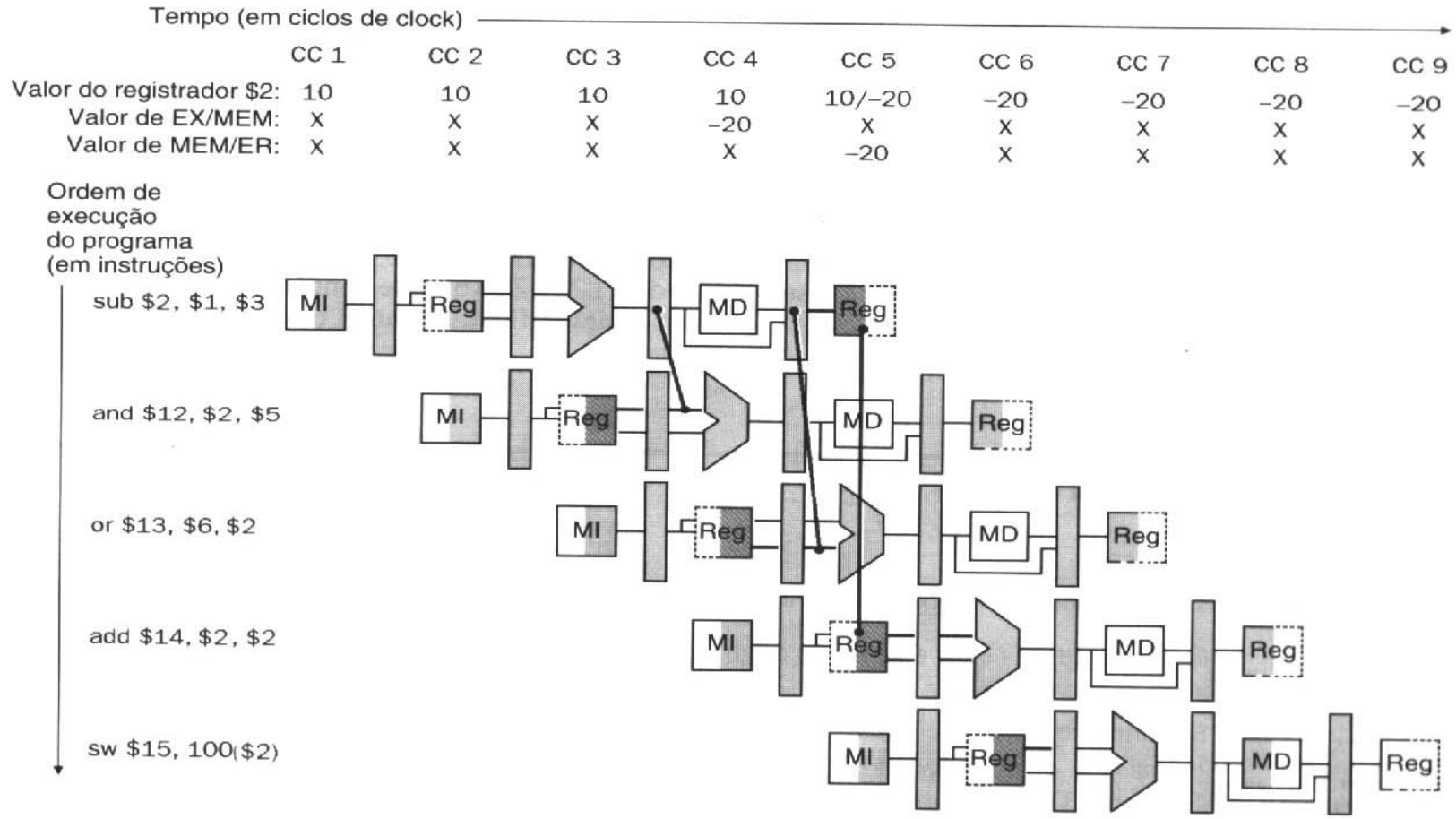


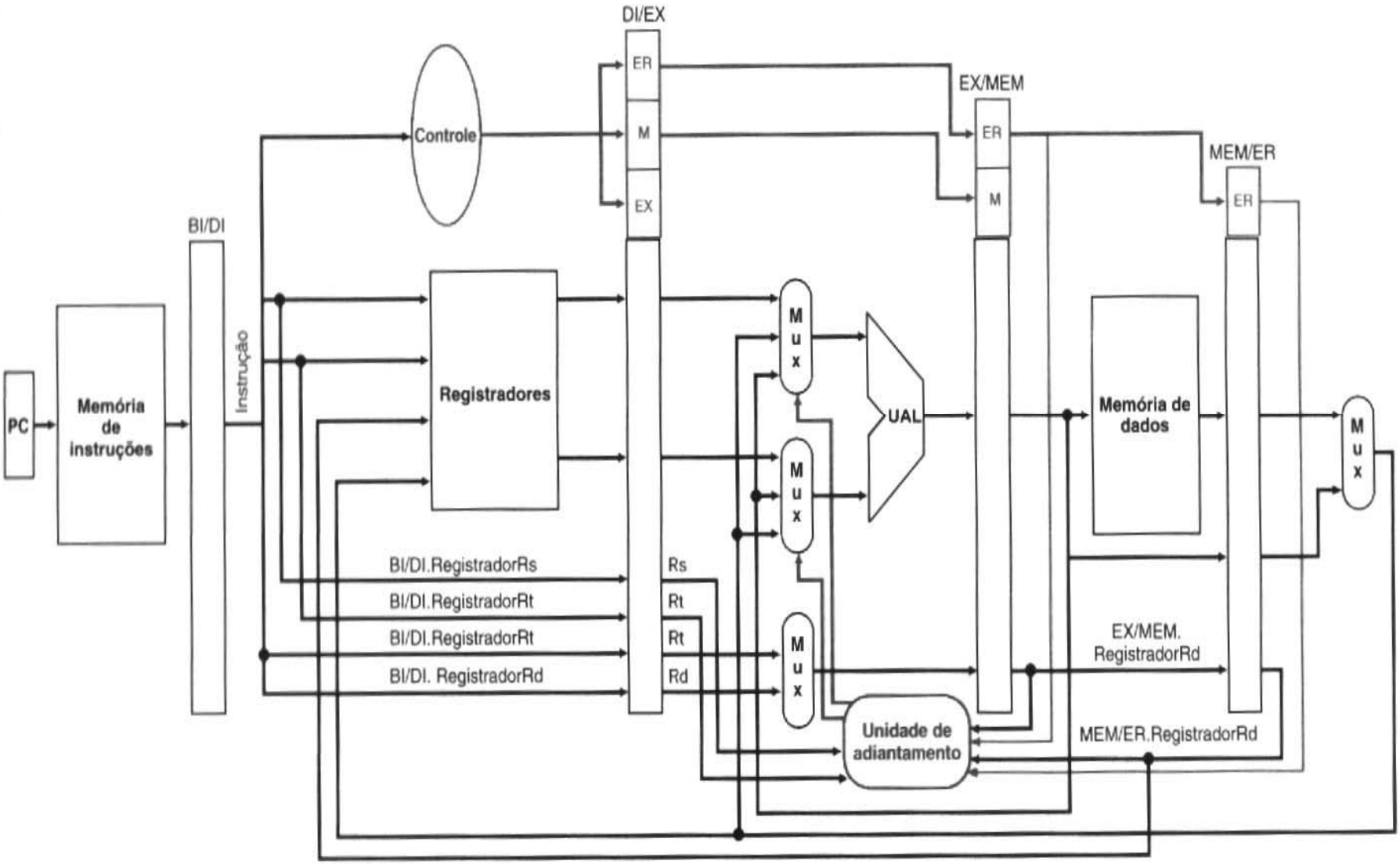
Adiantamento de resultados:

Utilizar valores temporários. Não esperar pelos valores a serem escritos.

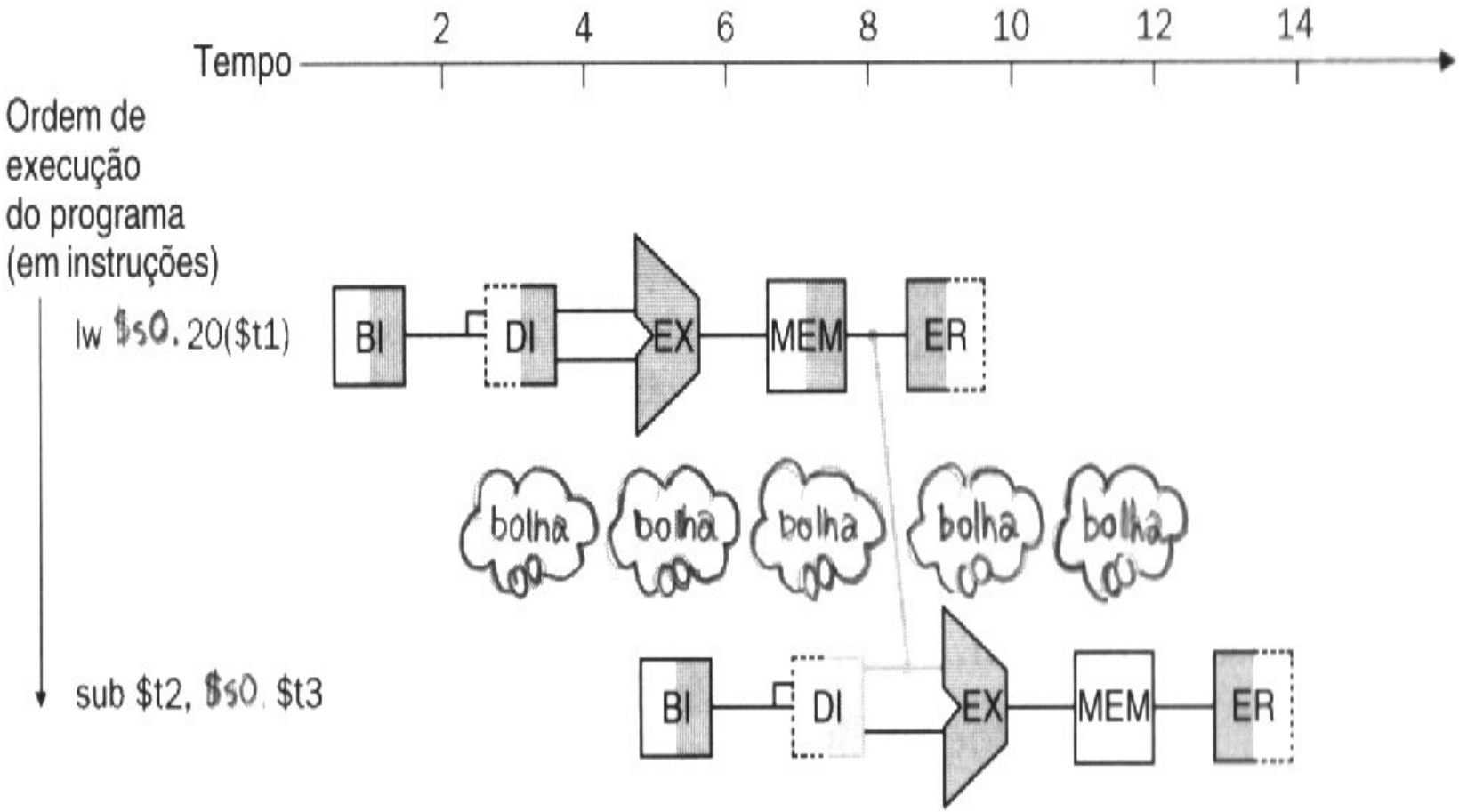
• Avançar conjunto de registradores para suportar leitura\escrita no mesmo registrador.

•Avançar o resultado da ALU.




DI/EX
EX/MEM
MEM/ER
ER
M
EX
Controle
BI/DI
Instrução
PC
Memória de instruções
Registradores
UAL
Memória de dados
Mux
BI/DI.RegistradorRs
BI/DI.RegistradorRt
BI/DI.RegistradorRt
BI/DI. RegistradorRd
Rs
Rt
Rt
Rd
EX/MEM. RegistradorRd
MEM/ER.RegistradorRd
Unidade de adiantamento

Adiantamento(forwarding): não resolve todos os problemas

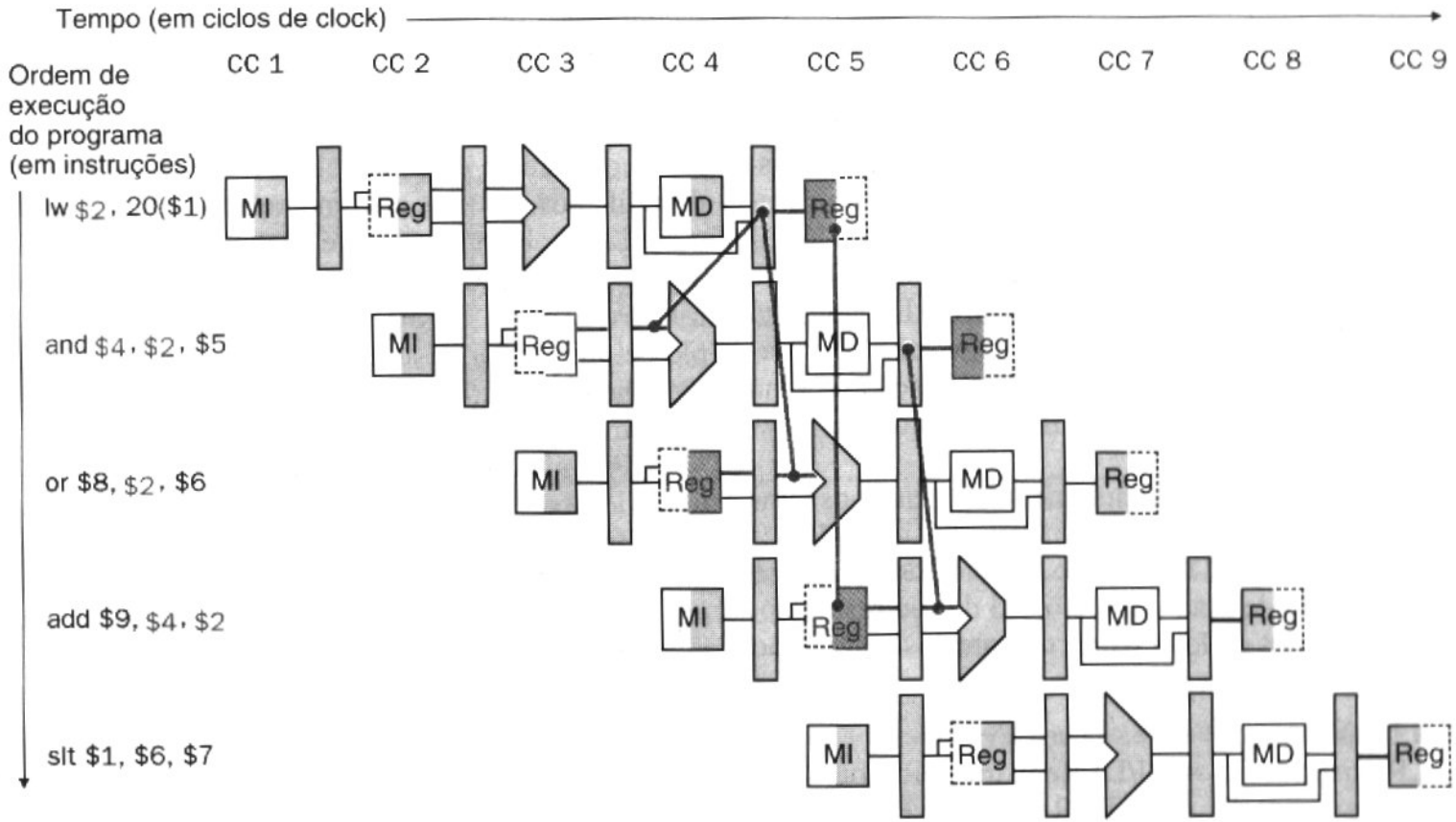


Adiantamento do resultado(forwarding)

Load pode causar um atraso:

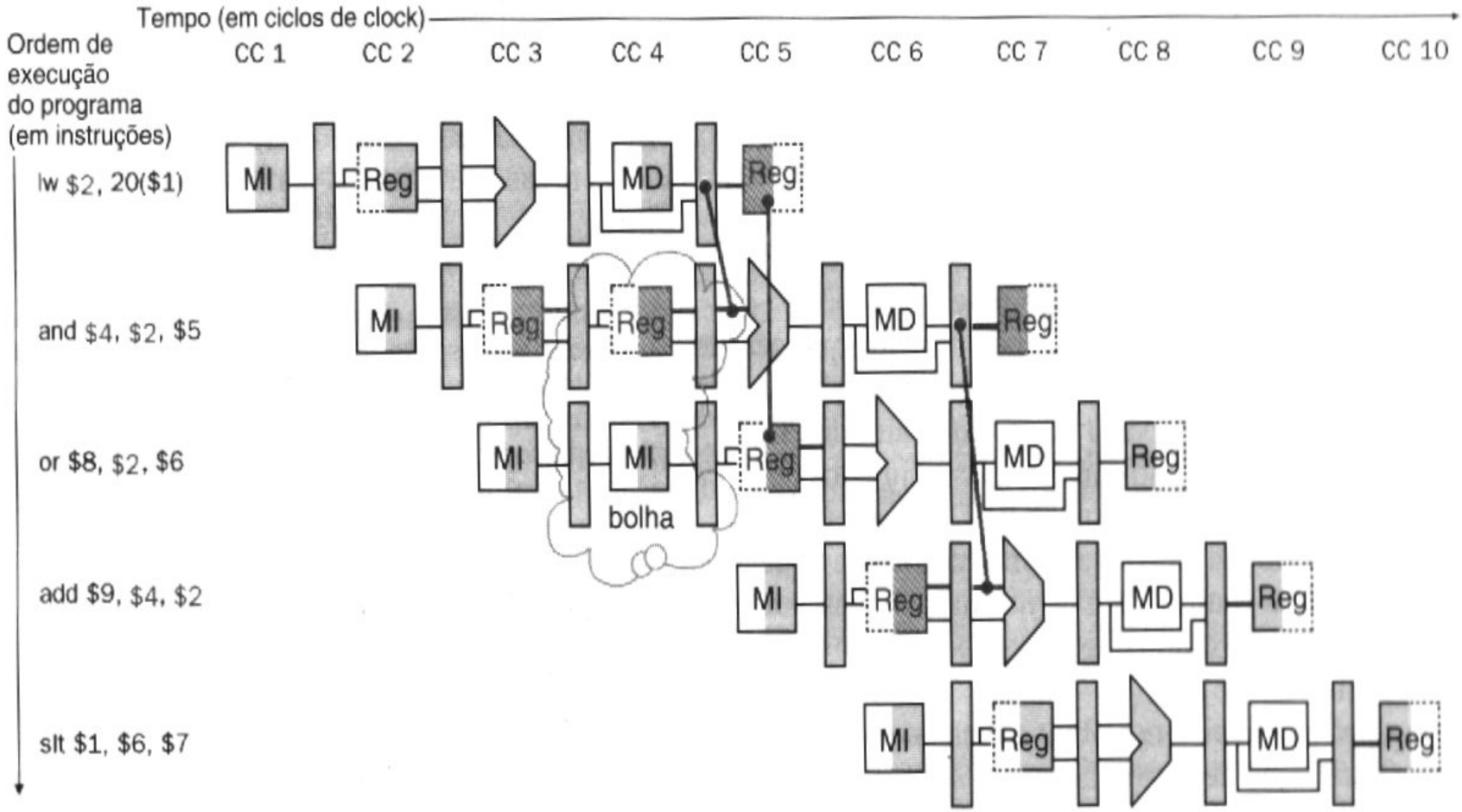
-Uma instrução tenta ler um registrador após um load que escreve no mesmo registrador.

E´ preciso inserir uma unidade de deteção deste conflito para atrasar a entrada da instr. seguinte ao load.



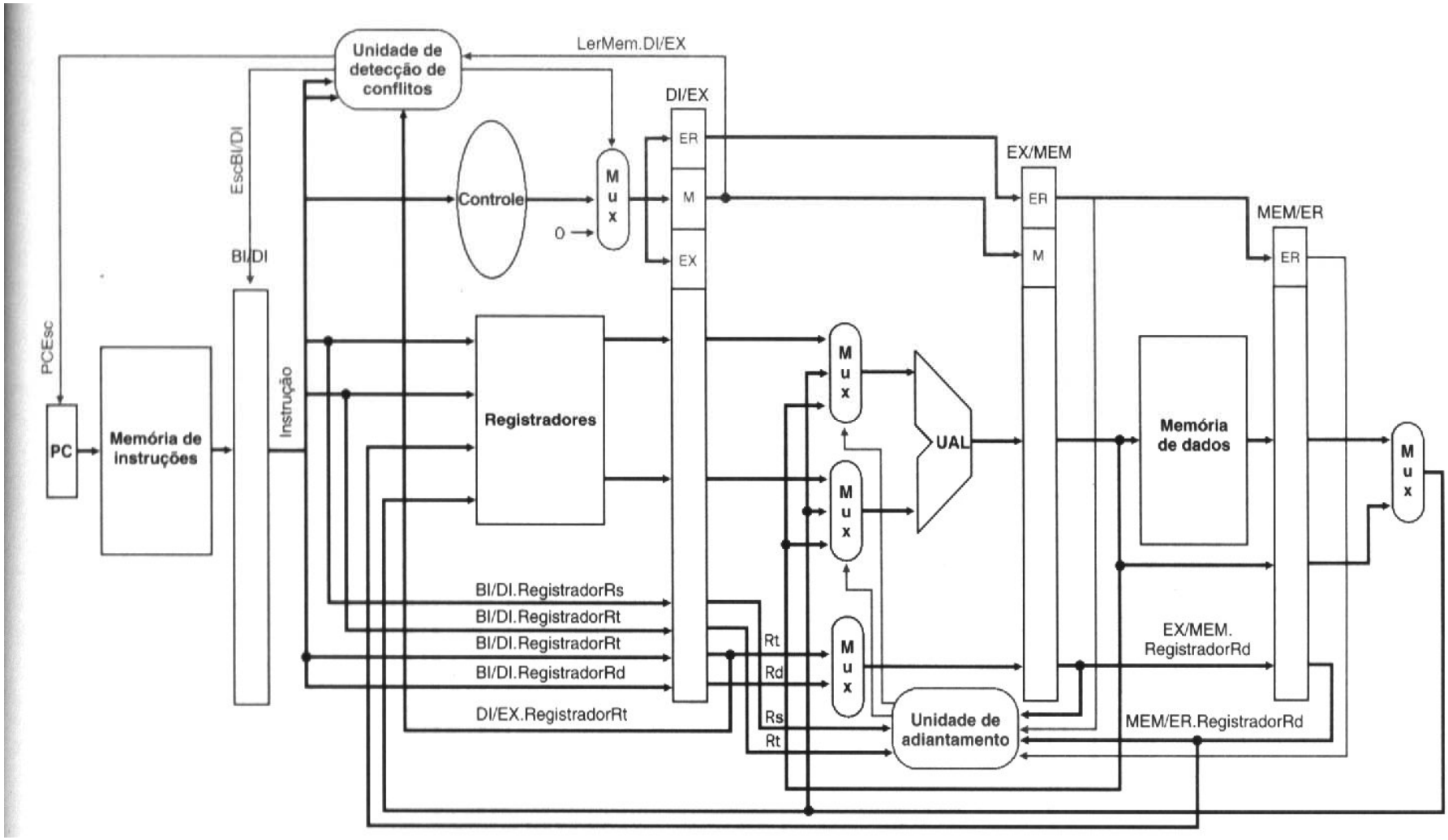
Solução parada do pipeline (Stalling)

Deve-se atrasar o pipeline mantendo uma instrução no mesmo estágio.

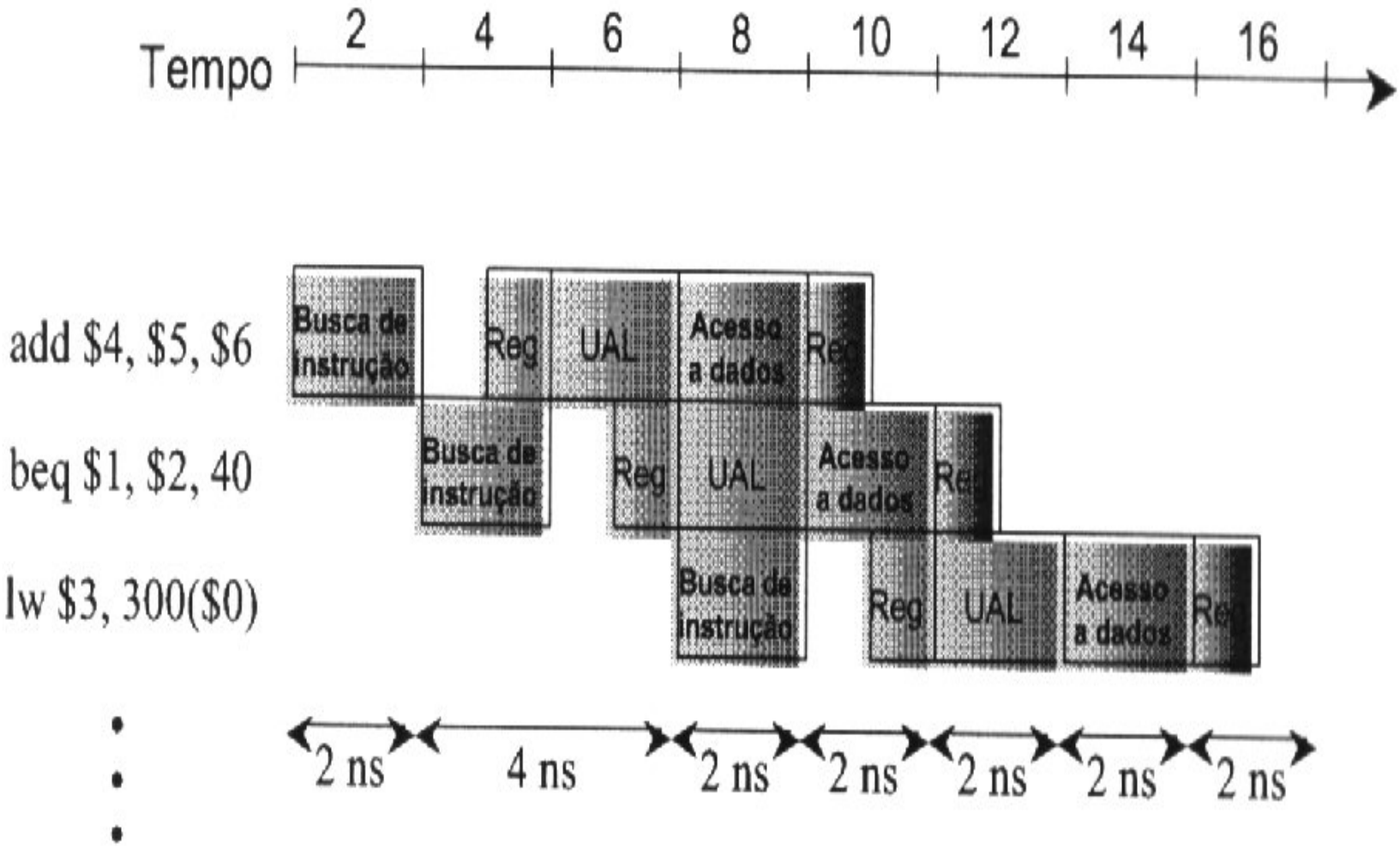


Unidade de detecção de conflitos:

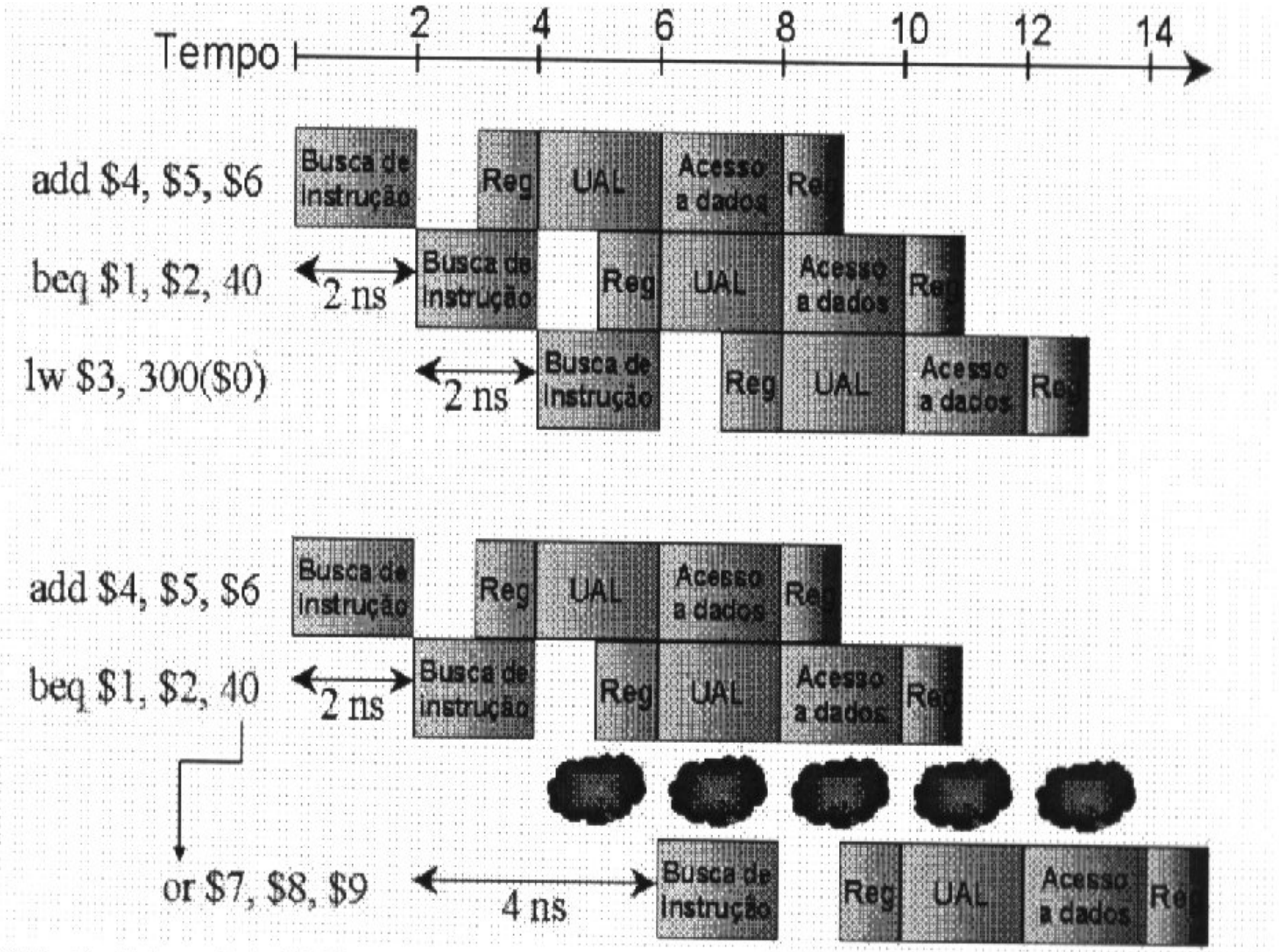
Age atrasando o pipeline para solucionar os conflitos



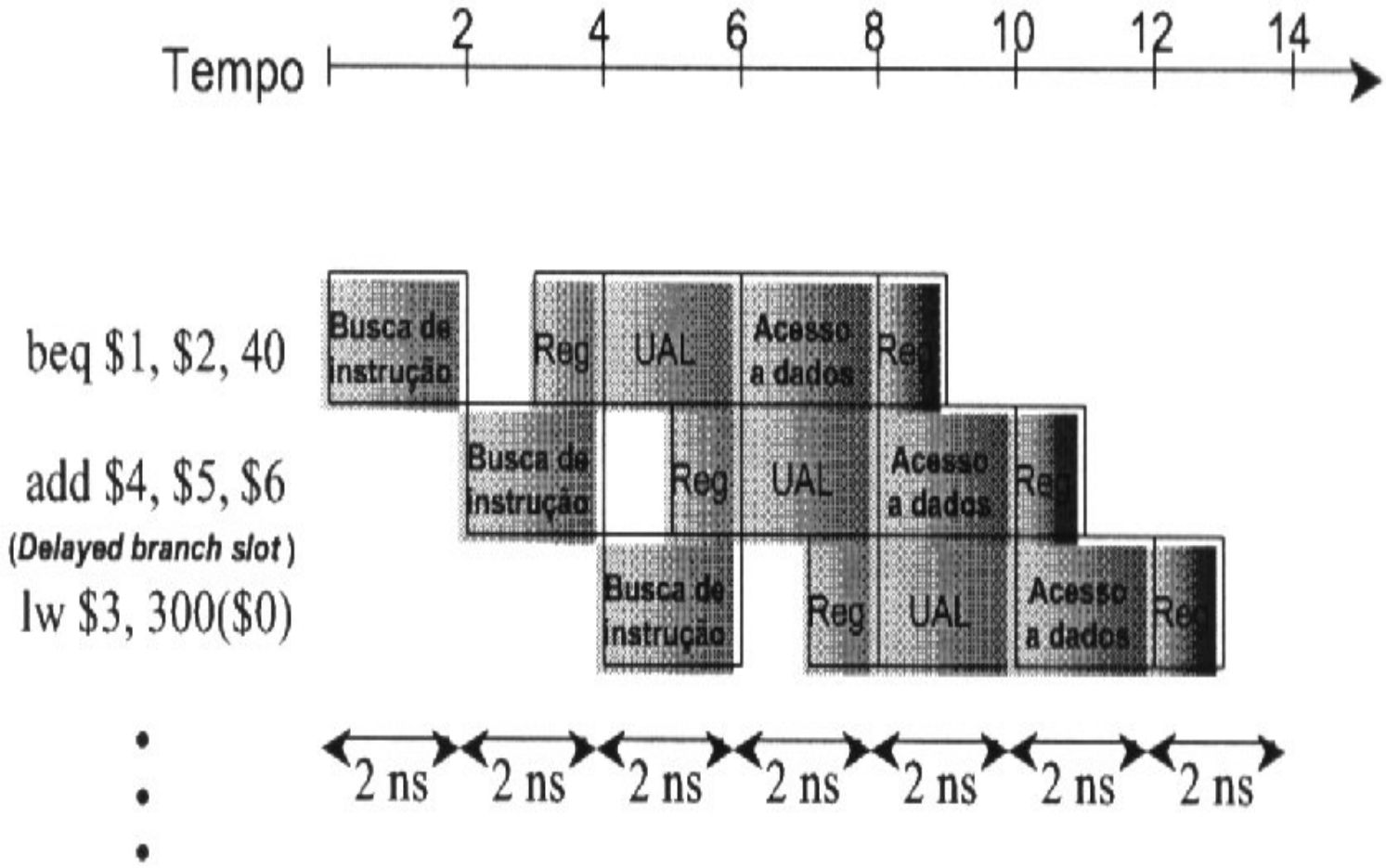
Conflito de controle: parada(bolha)

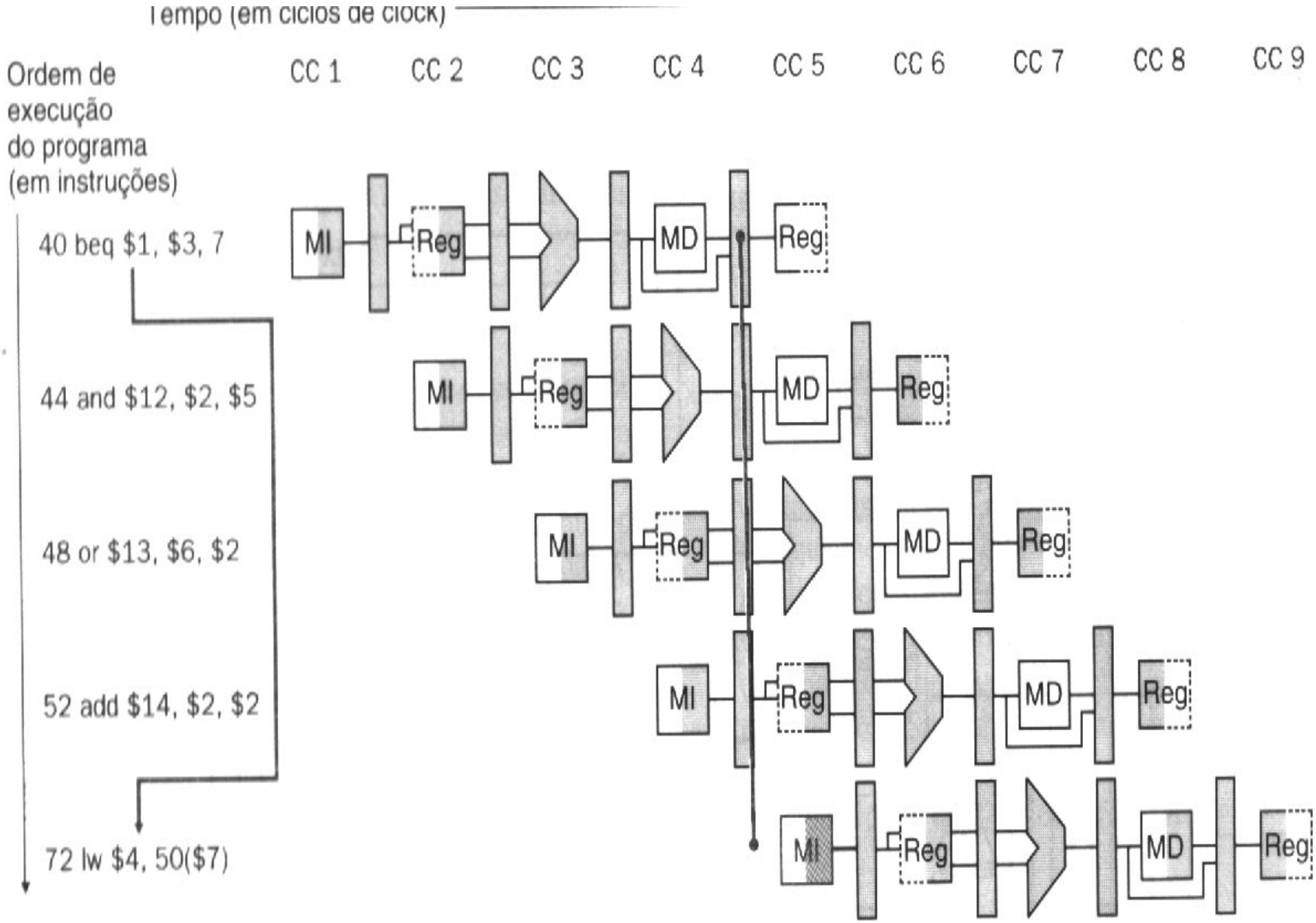


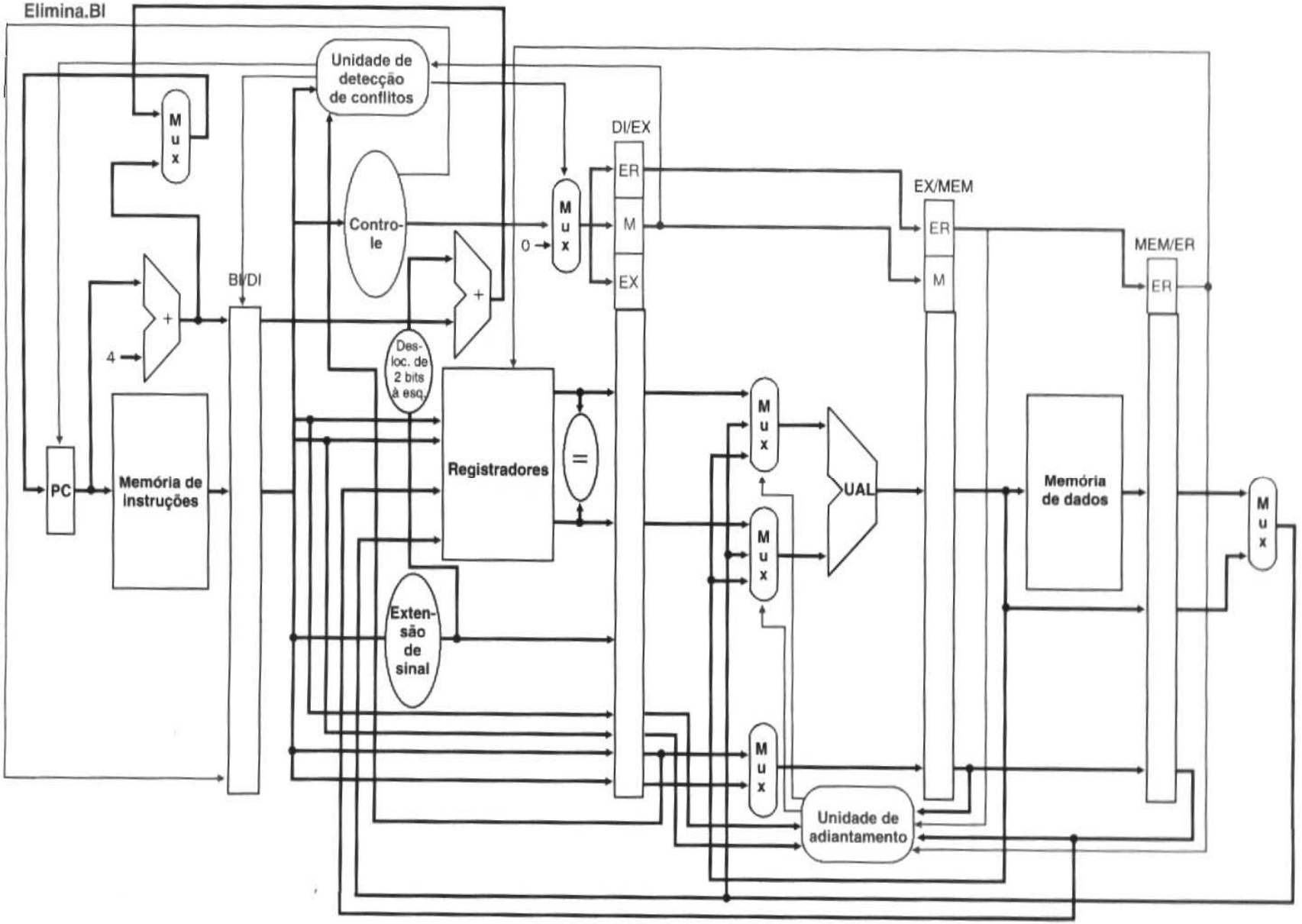
Conflito de controle: predição

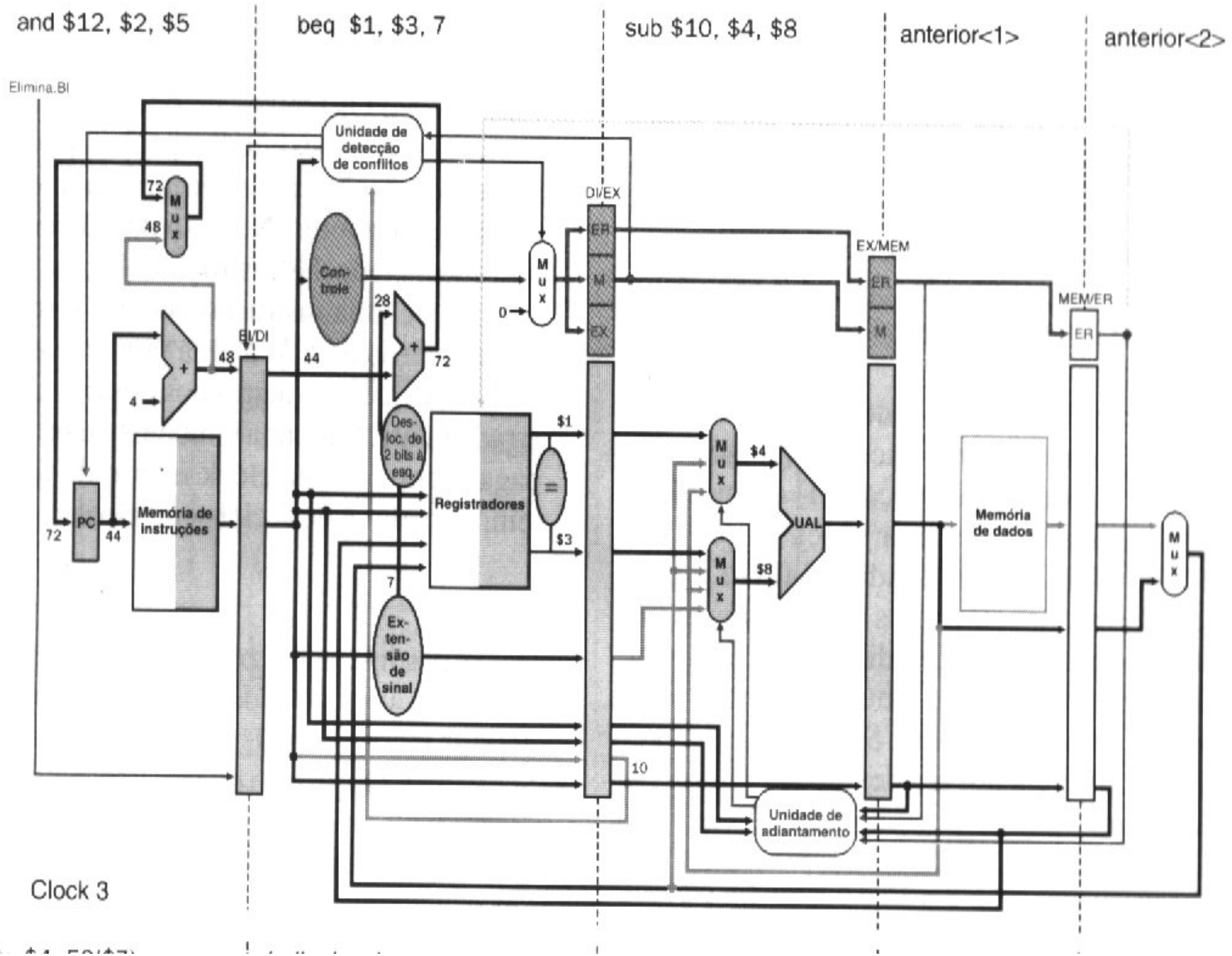


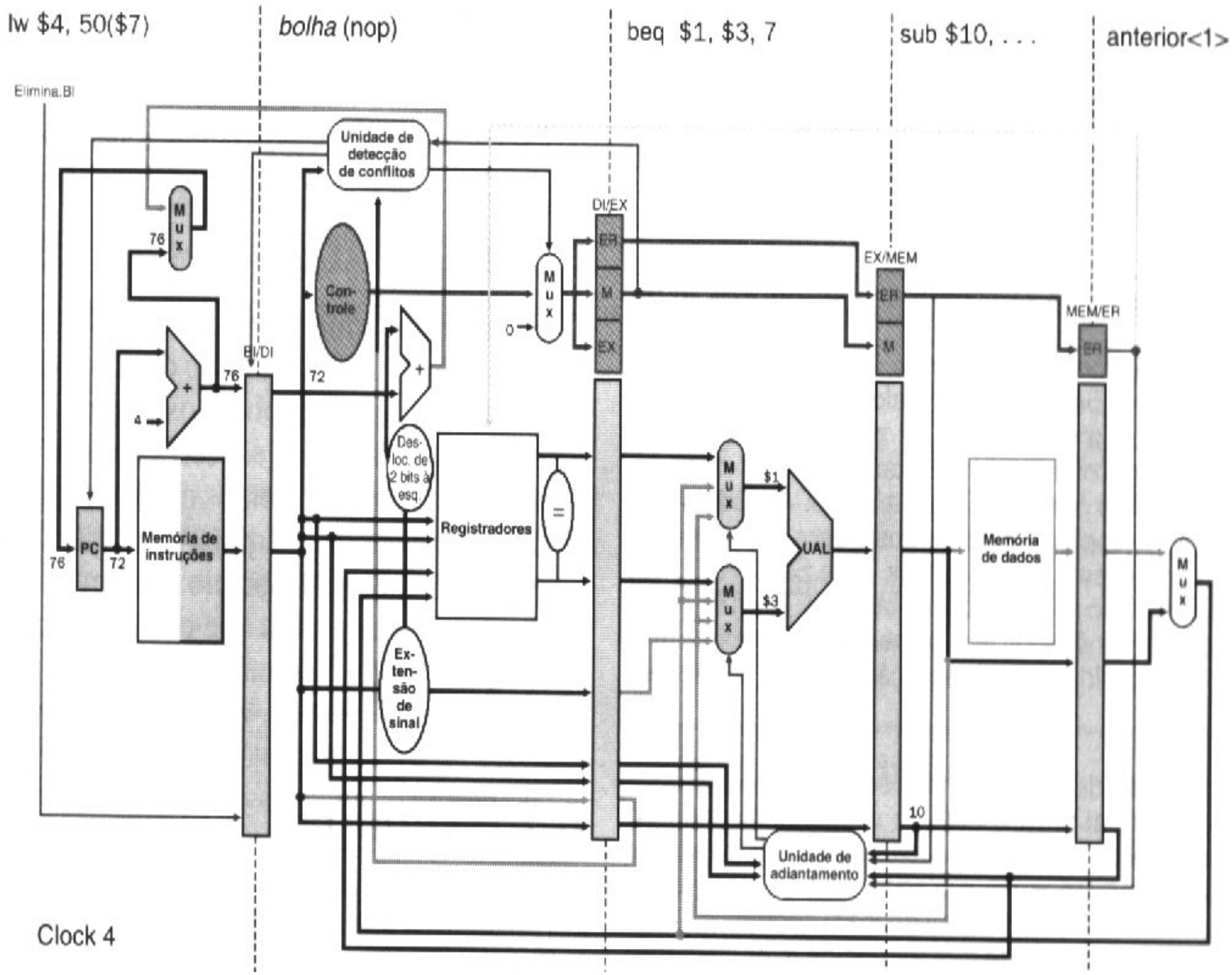
Solução para MIPS




Tempo (em ciclos de clock)
Ordem de execução do programa (em instruções)
CC 1
CC 2
CC 3
CC 4
CC 5
CC 6
CC 7
CC 8
CC 9
40 beq $1, $3, 7
44 and $12, $2, $5
48 or $13, $6, $2
52 add $14, $2, $2
72 lw $4, 50($7)
MI
Reg
MD
Reg


Elimina.BI
Unidade de detecção de conflitos
M u x
Contro-le
0
M u x
DI/EX
ER
M
EX
EX/MEM
ER
M
MEM/ER
ER
BI/DI
+
4
+
Des-loc. de 2 bits à esq.
PC
Memória de instruções
Registradores
=
M u x
M u x
M u x
UAL
Memória de dados
M u x
Exten-são de sinal
Unidade de adiantamento


and $12, $2, $5
beq $1, $3, 7
sub $10, $4, $8
anterior<1>
anterior<2>
Elimina.BI
Unidade de detecção de conflitos
72
48
Mux
Con-trole
28
+
72
0
Mux
DI/EX
ER
M
EX
EX/MEM
ER
M
MEM/ER
ER
4
+
48
BI/DI
44
PC
72
44
Memória de instruções
Des-loc. de 2 bits à esq.
Registradores
$1
=
$3
7
Ex-ten-são de sinal
Mux
$4
Mux
$8
UAL
Memória de dados
Mux
10
Unidade de adiantamento
Clock 3


lw $4, 50($7)
bolha (nop)
beq $1, $3, 7
sub $10, . . .
anterior<1>
Elimina.BI
Unidade de detecção de conflitos
Mux
76
Con-trole
DI/EX
ER
M
EX
EX/MEM
MEM/ER
0
BI/DI
72
4
Des-loc. de 2 bits à esq.
Registradores
=
PC
Memória de instruções
Ex-ten-são de sinal
$1
$3
UAL
Memória de dados
10
Unidade de adiantamento
Clock 4

Diminuindo a penalidade do ***Branch taken***

• no esquema anterior

- a decisão só é tomada no estágio MEM
- caso ocorra ***branch taken*** é necessário limpar os estágios IF, ID e EX
- 3 clocks perdidos

• para diminuir a penalidade:

- antecipar a decisão do estágio MEM para o estágio ID
- economia de dois clocks
- flush (limpa) somente na instrução sendo lida na memória (IF)

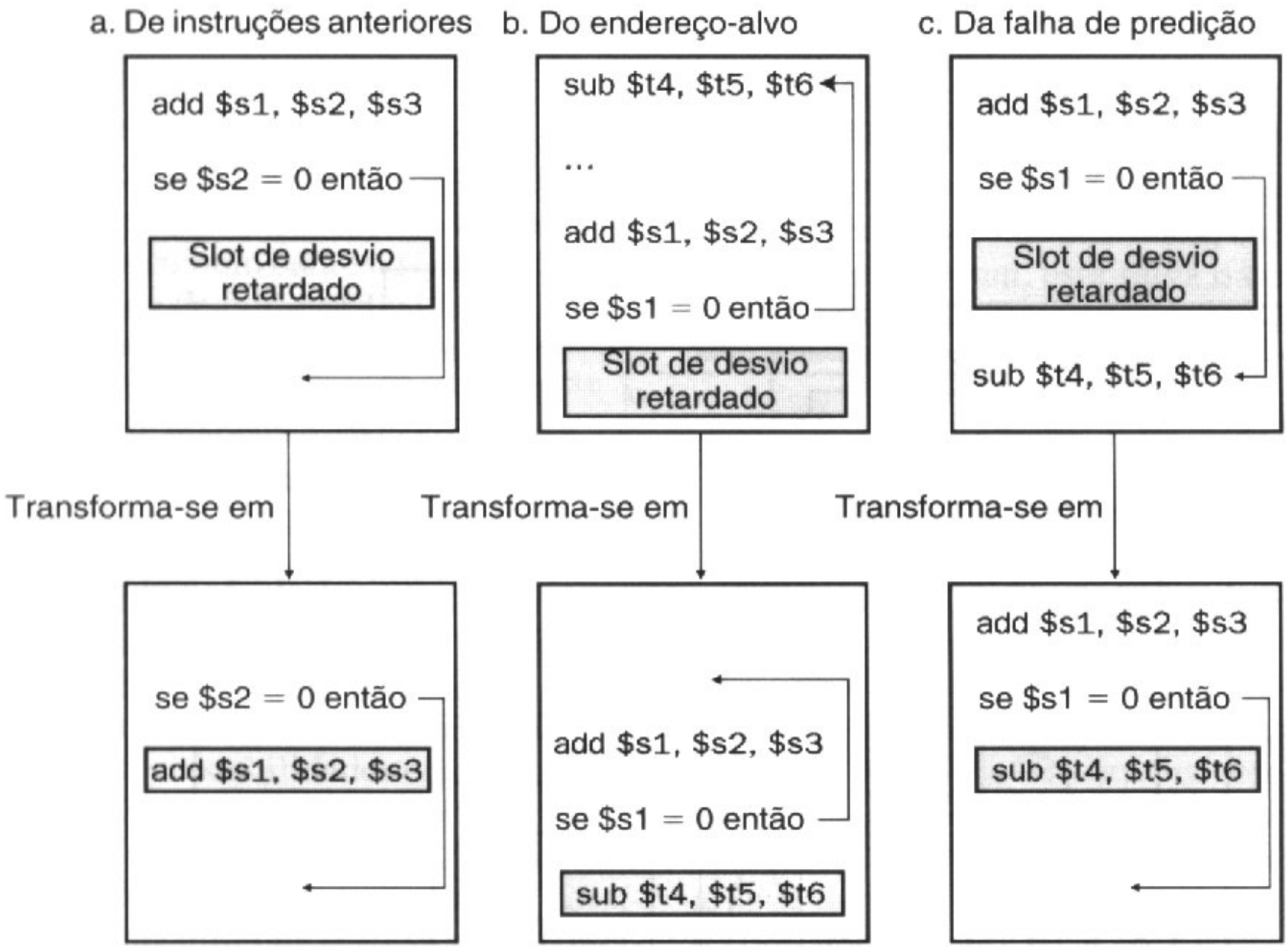
• mudanças no hardware :

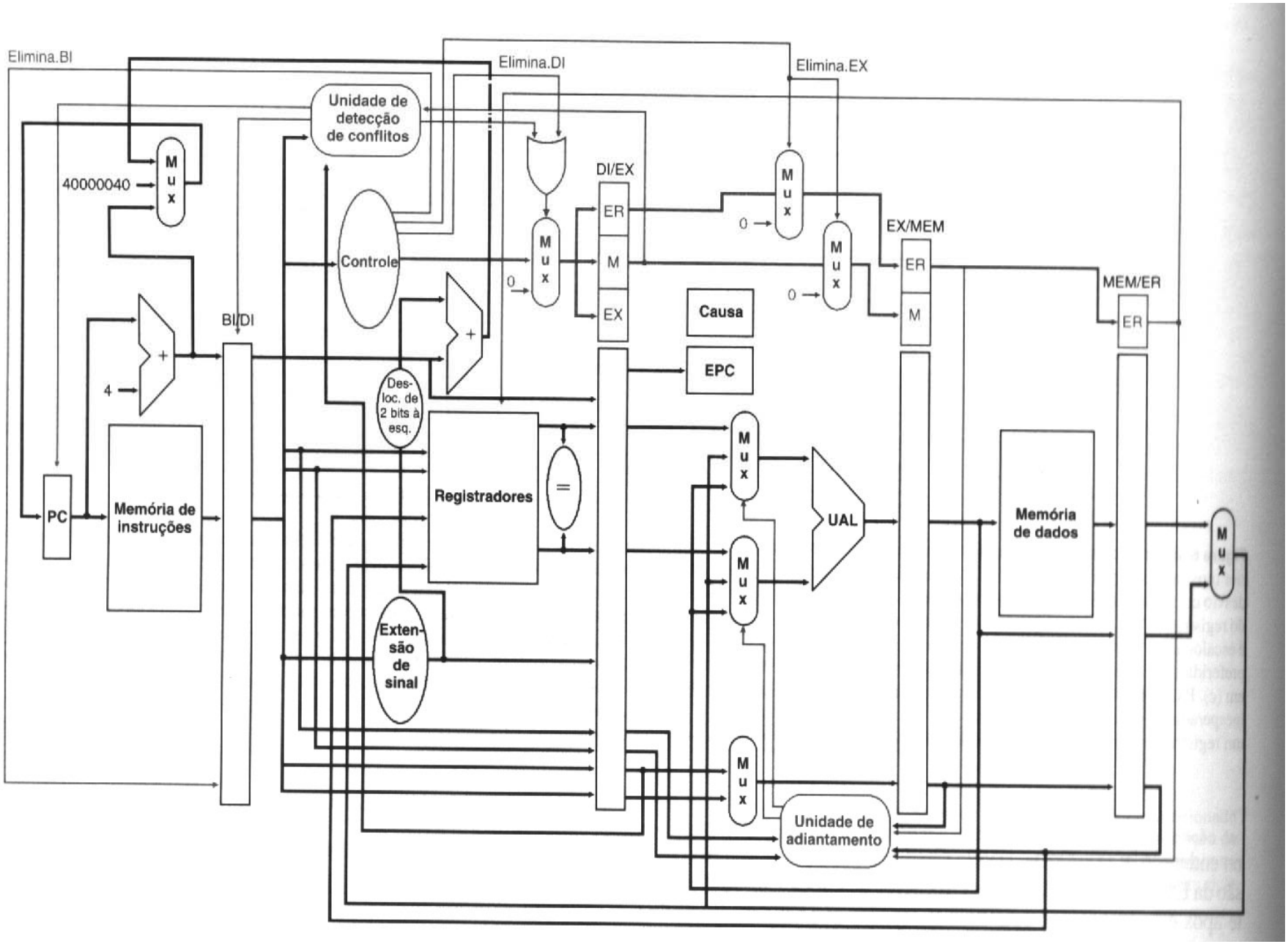
- cálculo do endereço do desvio
–
- comparação dos registradores

> 32 XORs com uma saída

> mais rápido do que a ALU


Realizado
Previsto e realizado
Não realizado
Previsto e realizado
Realizado
Realizado
Não realizado
Previsto e não realizado
Não realizado
Previsto e não realizado
Realizado
Não realizado


a. De instruções anteriores
b. Do endereço-alvo
c. Da falha de predição
add $s1, $s2, $s3
se $s2 = 0 então
Slot de desvio retardado
sub $t4, $t5, $t6
...
add $s1, $s2, $s3
se $s1 = 0 então
Slot de desvio retardado
add $s1, $s2, $s3
se $s1 = 0 então
Slot de desvio retardado
sub $t4, $t5, $t6
Transforma-se em
Transforma-se em
Transforma-se em
se $s2 = 0 então
add $s1, $s2, $s3
add $s1, $s2, $s3
se $s1 = 0 então
sub $t4, $t5, $t6
add $s1, $s2, $s3
se $s1 = 0 então
sub $t4, $t5, $t6


Elimina.BI
Elimina.DI
Elimina.EX
Unidade de detecção de conflitos
40000040
M u x
DI/EX
ER
M
EX
EX/MEM
MEM/ER
Controle
0
Causa
EPC
BI/DI
4
+
Des-loc. de 2 bits à esq.
PC
Memória de instruções
Registradores
=
UAL
Memória de dados
Exten-são de sinal
Unidade de adiantamento