Pipeline

Sumário

- Introdução;
- Pipeline Hazards:
  - Hazards Estruturais
  - Hazards de Dados
  - Hazards de Controle
- Caminho de Dados usando Pipeline;
- Representação Gráfica do Pipeline;
- Forwarding;
- Bibliografia.

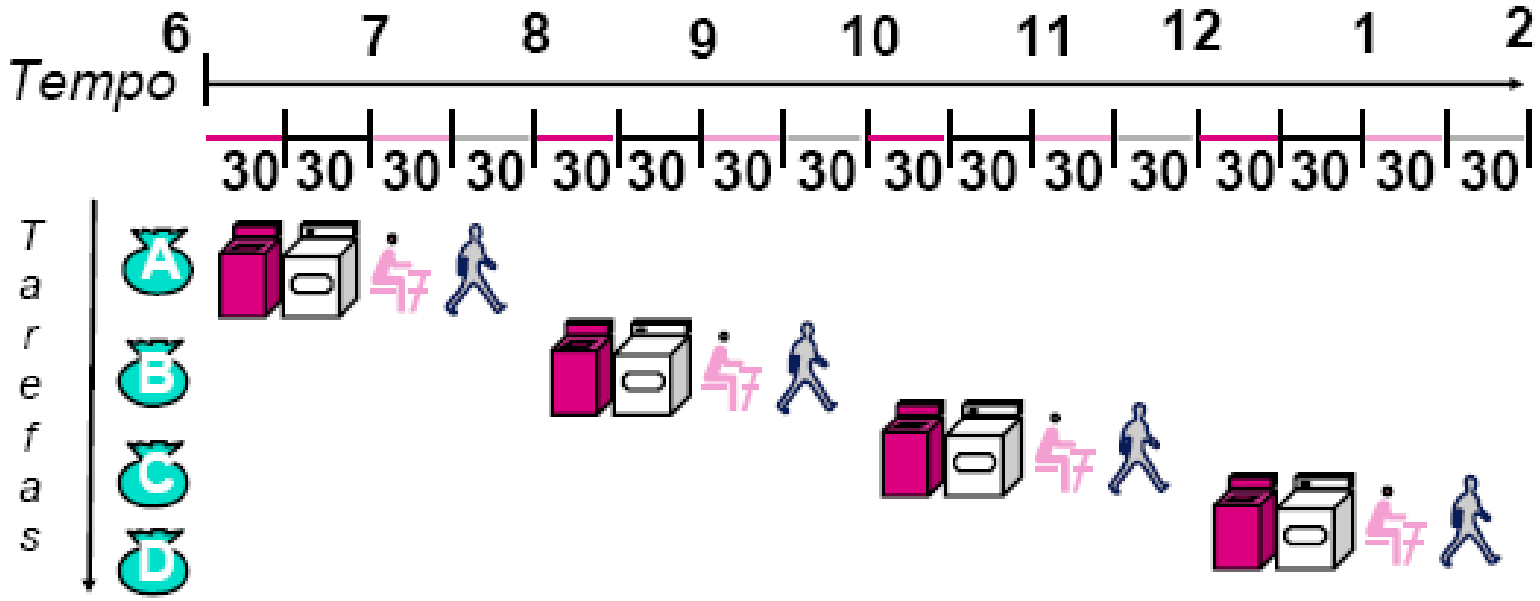
Introdução

- Pipelining é uma técnica de implementação em que várias instruções são sobrepostas na execução;

- Analogamente temos o exemplo de uma pessoa lavando roupas:
  - 1) Coloca a trouxa suja na lavadora;
  - 2) Quando a lavadora terminar. Colocar a trouxa na secadora;
  - 3) Quando a secadora terminar. Colocar a trouxa seca na mesa e passar
  - 4) Quando terminar de passar. Guardar as roupas

Introdução

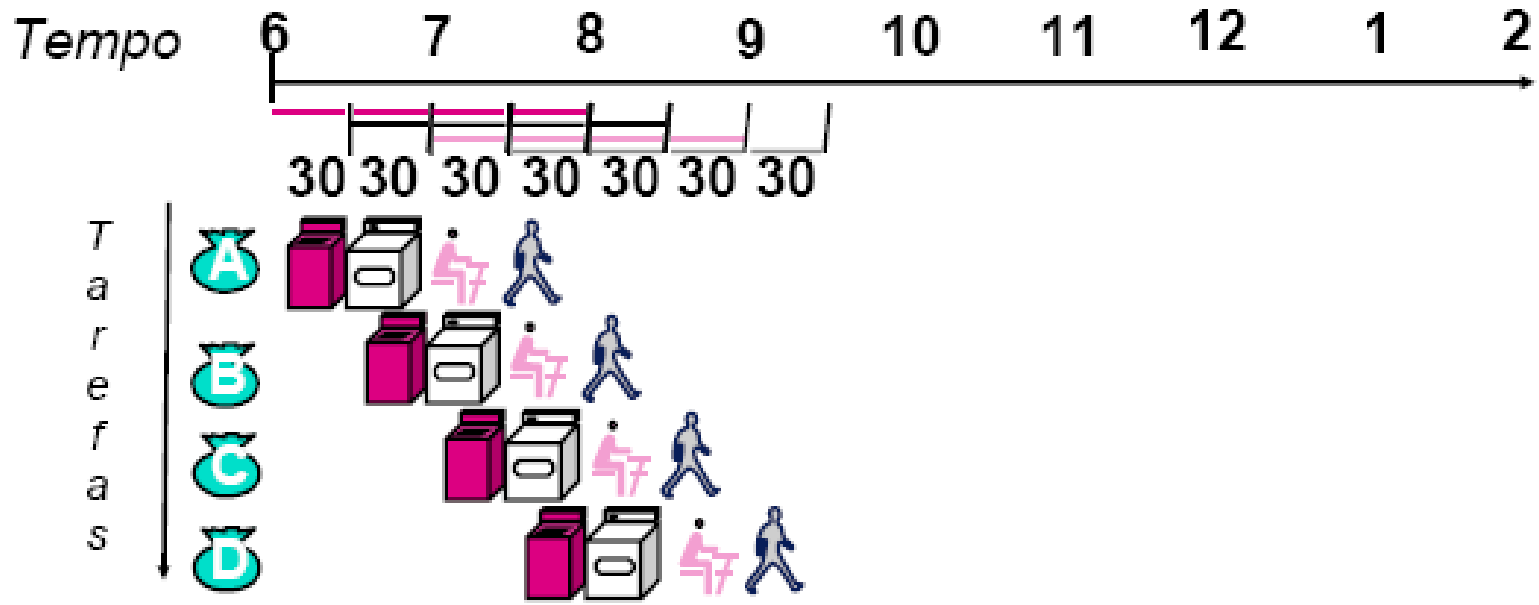
- Vejamos, nas figuras a seguir, a diferença do tempo que levaríamos sem pipeline e com pipeline:



Sem Pipeline

Introdução

- Com pipeline levaríamos menos tempo, como podemos ver nas figuras a seguir:

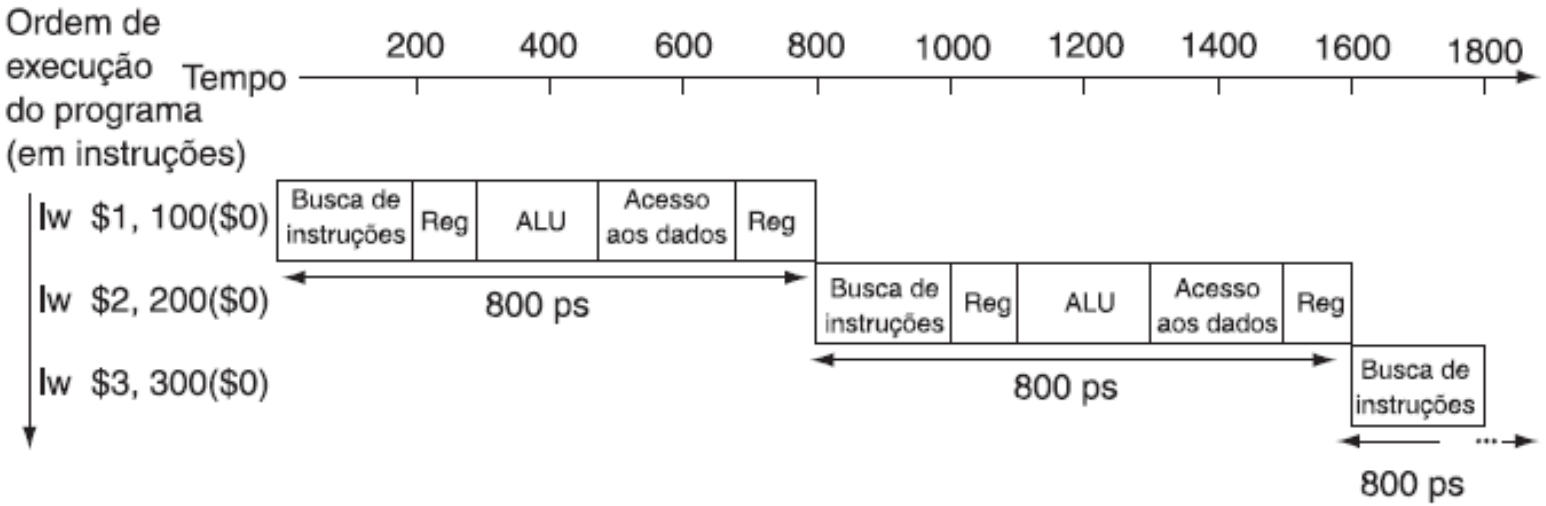


Introdução

- Como vimos no caminho de dados e de controle multiciclo, as instruções MIPS normalmente exigem cinco etapas:
  - 1) Busca da instrução na memória;
  - 2) Ler registradores enquanto a instrução é decodificada
  - 3) Executa a operação ou calcular um endereço;
  - 4) Acessar um operando na memória de dados;
  - 5) Escrever o resultado em um registrador;

Introdução

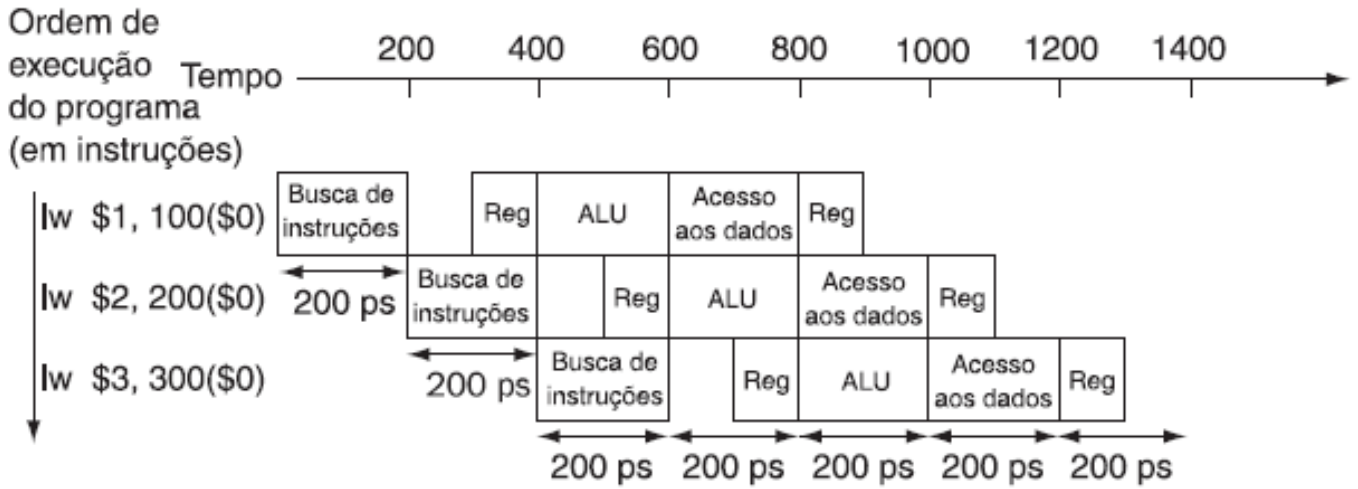
- Vejamos, a execução de três load word sem pipeline:



  - Considerando:
    - Busca da Instrução = 200ps
    - Leitura de registradores = 100ps
    - Operação da ULA = 200ps
    - Acesso a dados = 200ps

Introdução

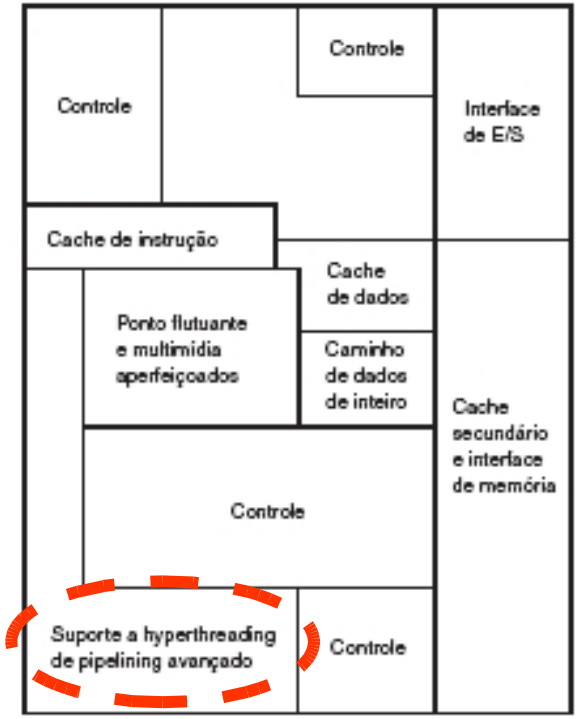
- Vejamos, a execução de três load word com pipeline:



  - Os tempos de estágio do pipeline dos computadores são limitados pelo recurso mais lento, seja a operação da ULa ou o acesso à memória

Introdução

- Vejamos o processador Pentium IV e sua área destinada ao pipeline:



Introdução

- Facilidades do Pipeline:
  - Todas as instruções possuem o mesmo tamanho;
  - Apenas alguns formatos de instrução
  - Operandos de memória aparecem apenas em loads e stores

- O que dificulta:
  - Riscos estruturais: apenas uma memória, por exemplo;
  - Riscos de Controle: Necessidade de se preocupar com instruções ramificadas;
  - Riscos de dados: uma instrução depende de uma instrução anterior;

Pipeline Hazards

- Existem situações em pipelining em que a próxima instrução não pode ser executada no ciclo de clock seguinte.

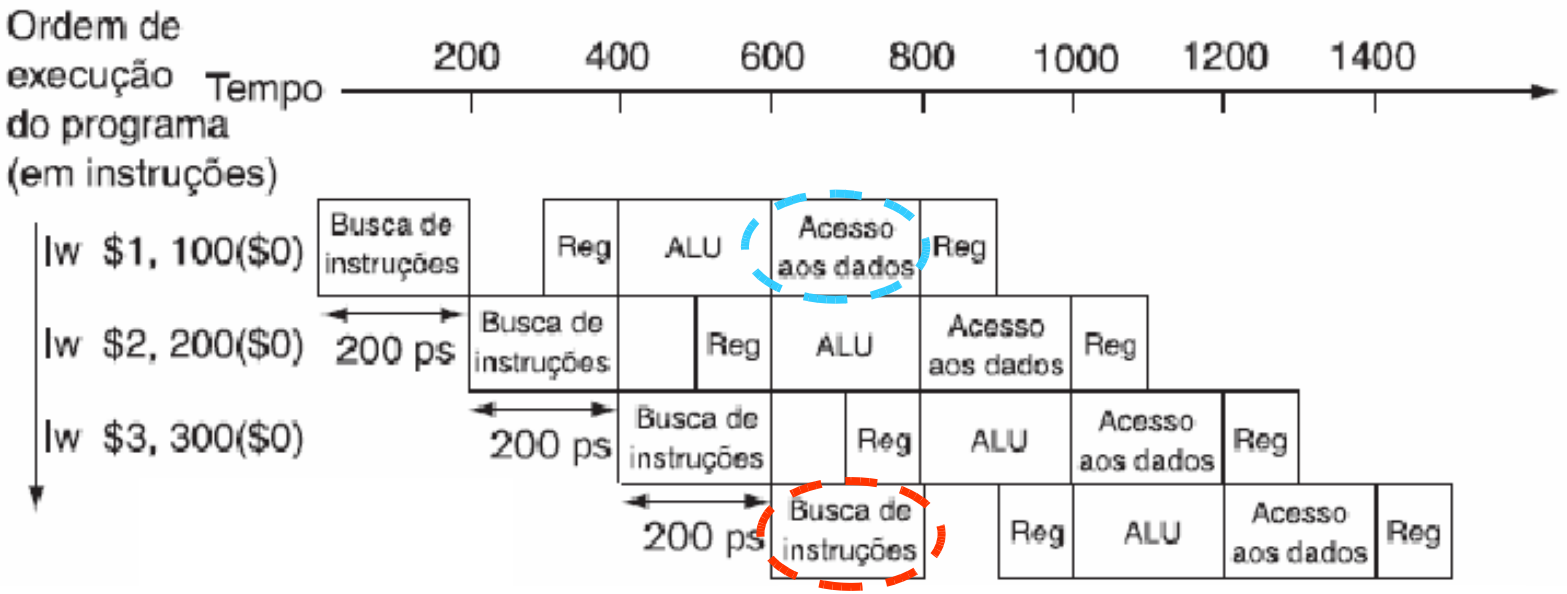
- Esses eventos são chamados de hazards (Riscos), e existem três tipos diferentes:
  - Hazards Estruturais
  - Hazards de Dados
  - Hazards de Controle

Hazards Estruturais

- Significa quando o hardware não pode admitir a combinação de instruções que queremos executar
  - Suponha que tivéssemos uma única memória, em vez de duas;
  - Se o pipeline tivesse uma quarta instrução, veríamos que, no mesmo ciclo de clock em que a primeira instrução está acessando dados da memória, a quarta instrução está buscando uma instrução dessa mesma memória;
  - Sem duas memórias, nosso pipeline poderia ter um hazard estrutural;

Hazards Estruturais

- Vejamos na figura:



Hazards de Dados

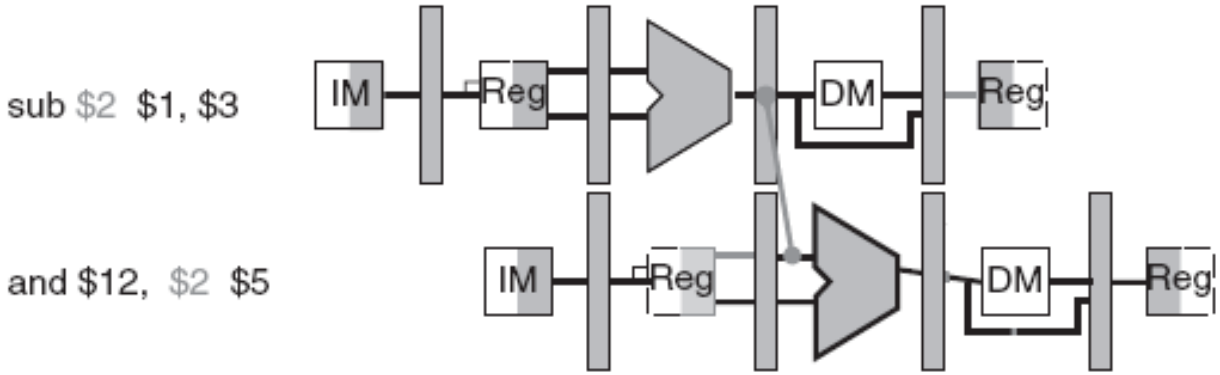
- Ocorrem quando o pipeline precisa ser interrompido porque uma etapa precisa esperar até que outra seja concluída;

- Por exemplo, suponha que tenhamos uma instrução add seguida imediatamente por uma instrução subtract que usa a soma($s0);

```
add $s0, $t0, $t1
sub $t2, $s0, $t3
```

Hazards de Dados

- A solução mais viável para esse tipo de risco é baseada na observação de que não precisamos esperar que a instrução termina antes de tentar resolver o hazard de dados;



- O acréscimo de hardware extra para ter o item que falta antes do previsto, diretamente dos recursos internos, é chamado de **forwarding** e **bypassing**;

Hazard de Controle

- Decorre da necessidade de tomar uma decisão com base nos resultados de uma instrução enquanto outras estão sendo executadas;

- Para os casos de desvio, os computadores usam a previsão e sempre prever que os desvios não serão tomados;
  - Se ele estiver certo, o pipeline prosseguirá a toda velocidade;
  - Quando os desvios são tomados, o pipeline sofre um stall (bolhas)

Hazard de Controle

- Um técnica popular para a previsão dinâmica de desvios é manter um histórico de cada desvio como tomado ou não tomado;
- Em seguida, o comportamento passado servirá para prever o futuro;
- Estudos mostram que a previsão de desvios consiste em uma técnica com acertos superiores a 90%;

Hazard de Controle

- A última solução é conhecida como decisão adiada. Supomos o trecho de código MIPS a seguir:

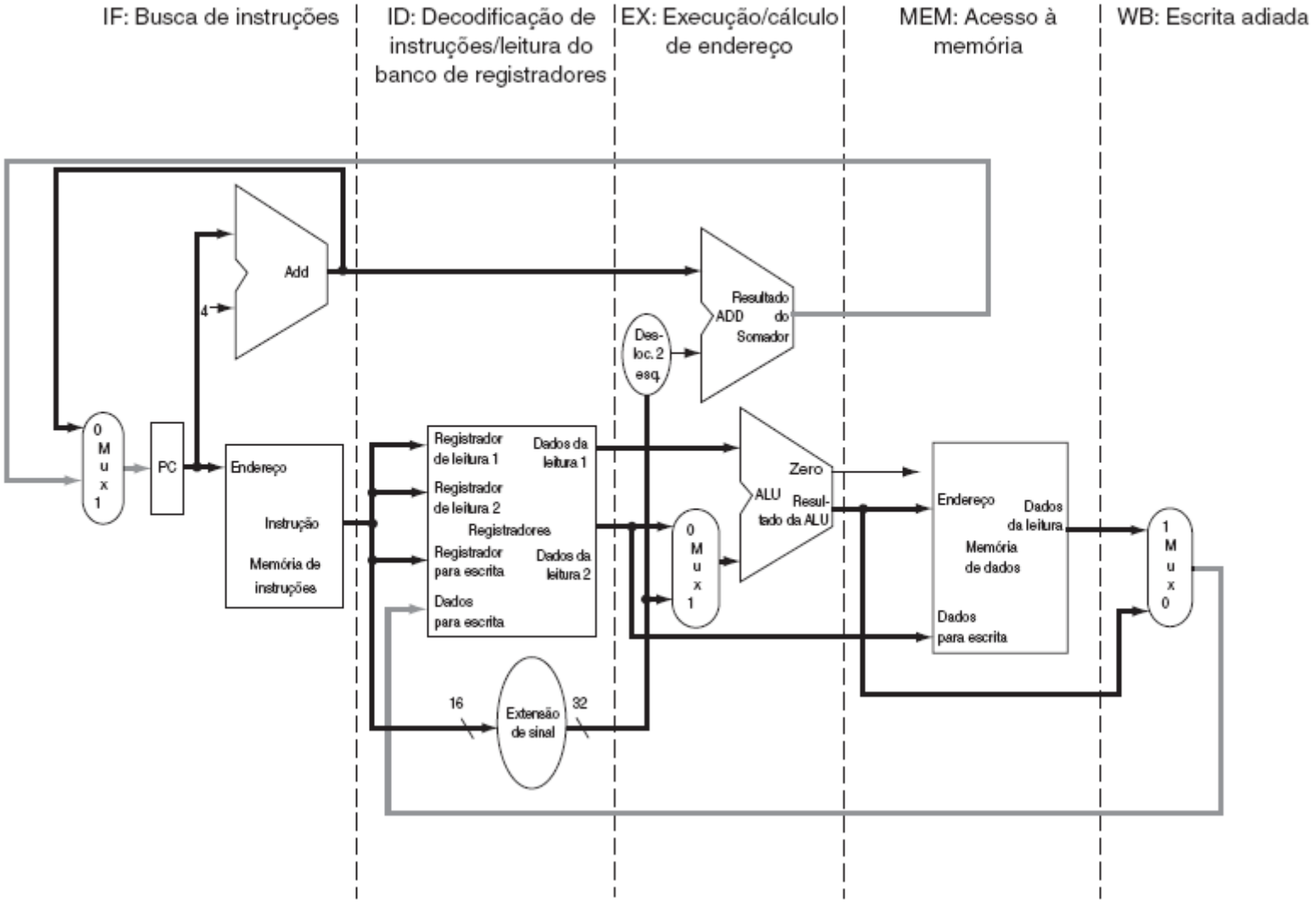
```
add $s4, $s5, $s6
beq $s1, $s2, L1
lw $s3, 3000($s3)
```

- Nesse caso, a arquitetura MIPS colocará uma instrução imediatamente após a instrução de desvio, que não é afetada pelo desvio (add $s4, $s5, $s6);

Caminho de Dados usando Pipeline

- Tomemos o caminho de dados de ciclo único para a divisão de uma instrução em cinco estágios:
  - IF (Instruction Fetch): Busca das Instruções
  - ID (Instruction Decode): Decodificação de Instruções e Leitura do banco de registradores
  - EX (Execution): Execução ou cálculo de endereço
  - MEM(Memory): Acesso à memória de dados
  - WB(Qrite Back): Escrita do resultado

Caminho de Dados usando

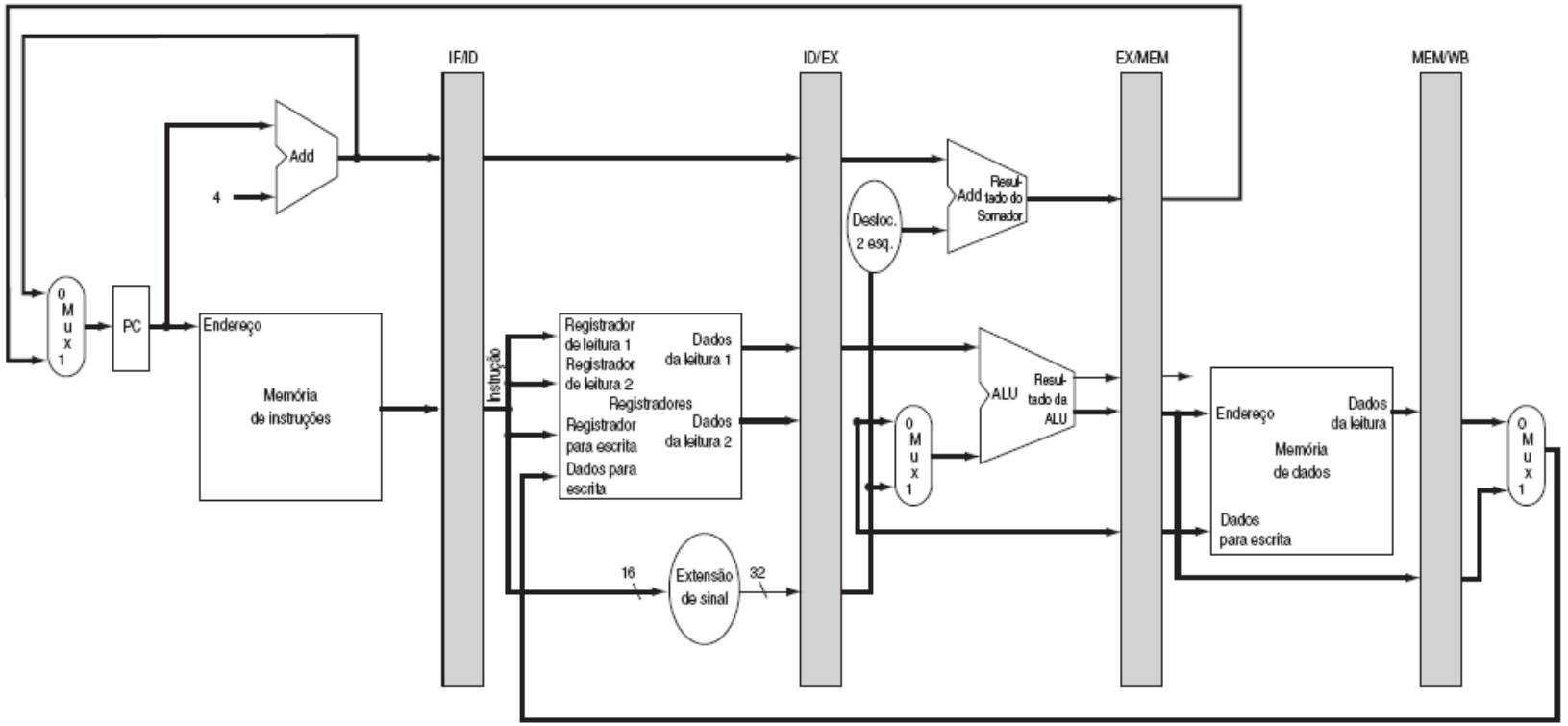


Caminho de Dados usando Pipeline

- Considerações:
  - Cada etapa da instrução pode ser mapeada no caminho de dados da esquerda para a direita;
  - As únicas exceções são:
    - a atualização do PC, linhas em negrito, que envia o resultado da ULA e
    - a etapa de escrita do resultado, onde os dados da memória são enviados para a esquerda (escrita no banco de registradores)

Caminho de Dados usando Pipeline

- Como vimos no caminho de dados de múltiplos ciclos é necessários a inserção de registradores para “salvar” cada estágio do pipeline;

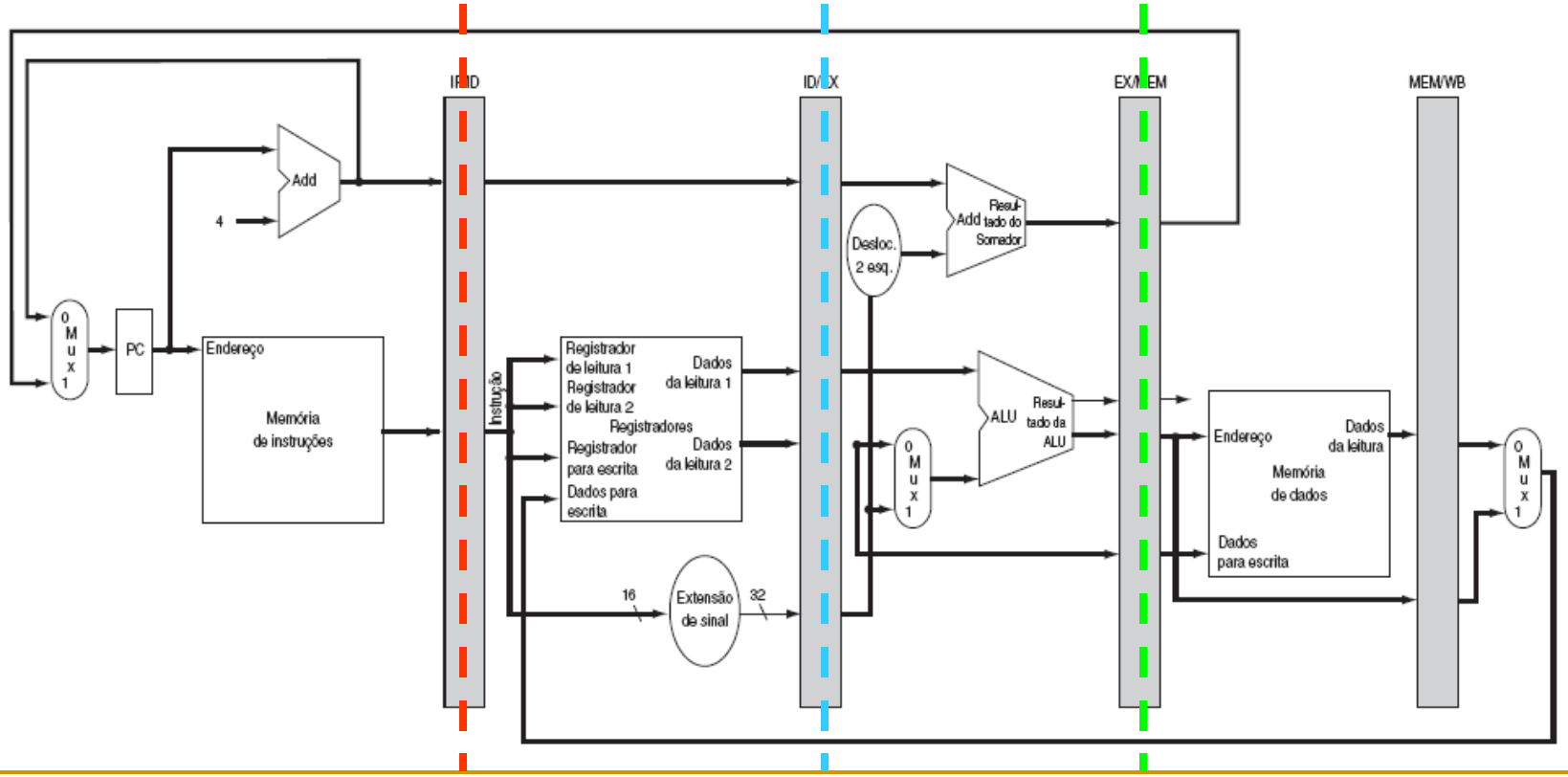


Caminho de Dados usando Pipeline

- Considerações:
  - Os registradores são rotulados pelos nomes dos estágios que separam, por exemplo, IF/ID porque separa os estágios de busca de instruções e decodificação de instrução;
  - Os registradores precisam ser grandes o suficiente para armazenar todos os dados correspondentes às linhas que passam por eles. Por exemplo, IF/ID precisa ter 64 bits de largura, pois precisa manter a instrução de 32 bits lida da memória e o endereço incrementado de 32 bits no PC;
    - ID/EX = 128 bits;
    - EX/MEM = 97 bits;
    - MEM/WB = 64 bits;

Caminho de Dados usando Pipeline

- Vejamos como a instrução lw passa pelos cinco estágios de execução do pipeline:

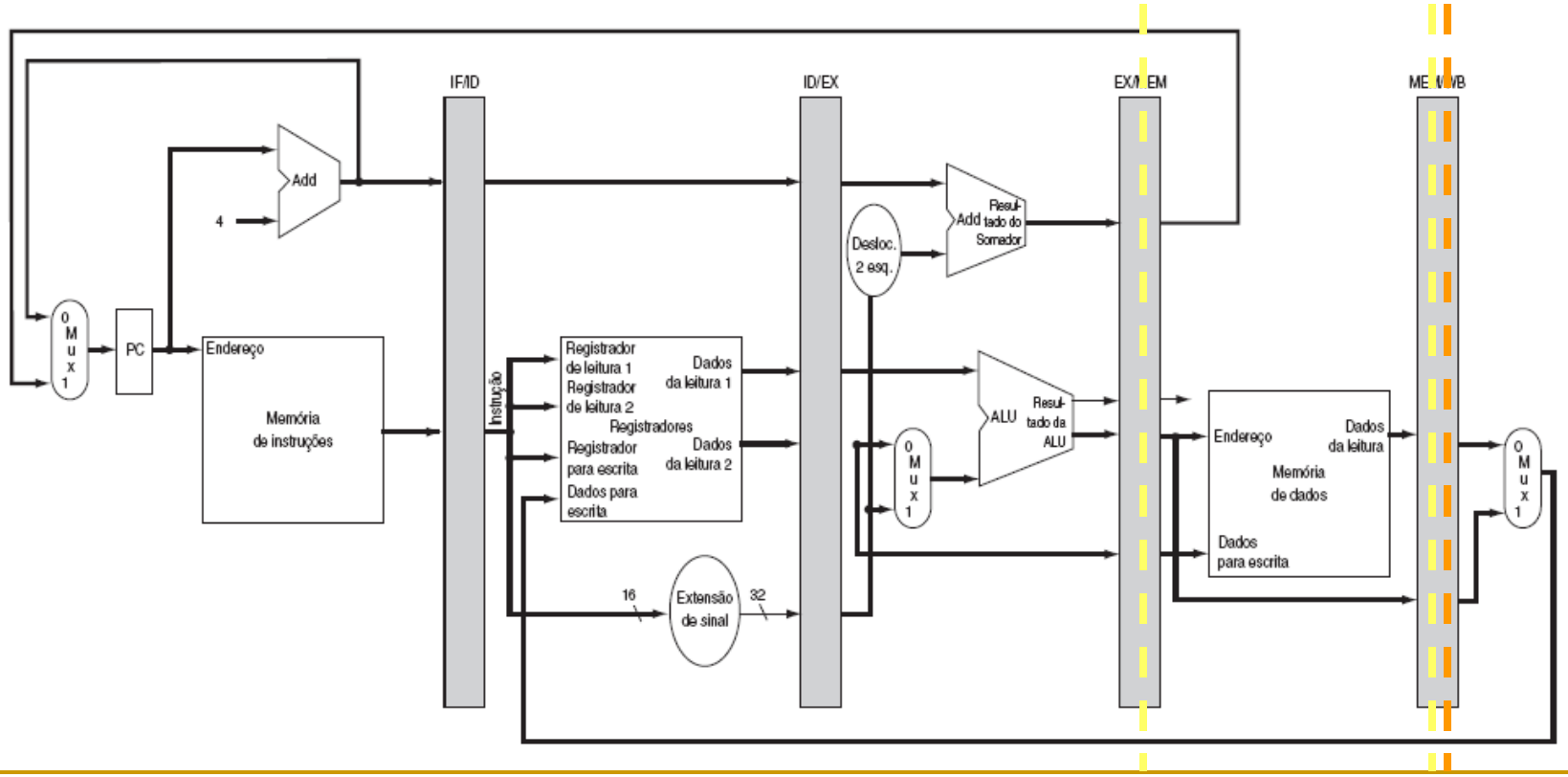


Caminho de Dados usando Pipeline

- Considerações:
  - Embora o load só precise do registrador de cima no segundo estágio, o processador não sabe que instrução está sendo decodificada, de modo que estende o sinal da constante de 16 bits e lê os dois registradores para o registrador de pipeline ID/EX;
  - Não precisamos de todos os três operando, mas simplifica o controle manter todos os três;
  - No terceiro estágio, o registrador é acrescentado ao imediato com sinal estendido e a soma é colocada no registrador de pipeline EX/MEM;

Caminho de Dados usando Pipeline

- Vejamos como a instrução lw pelos dois últimos estágios de execução do pipeline:

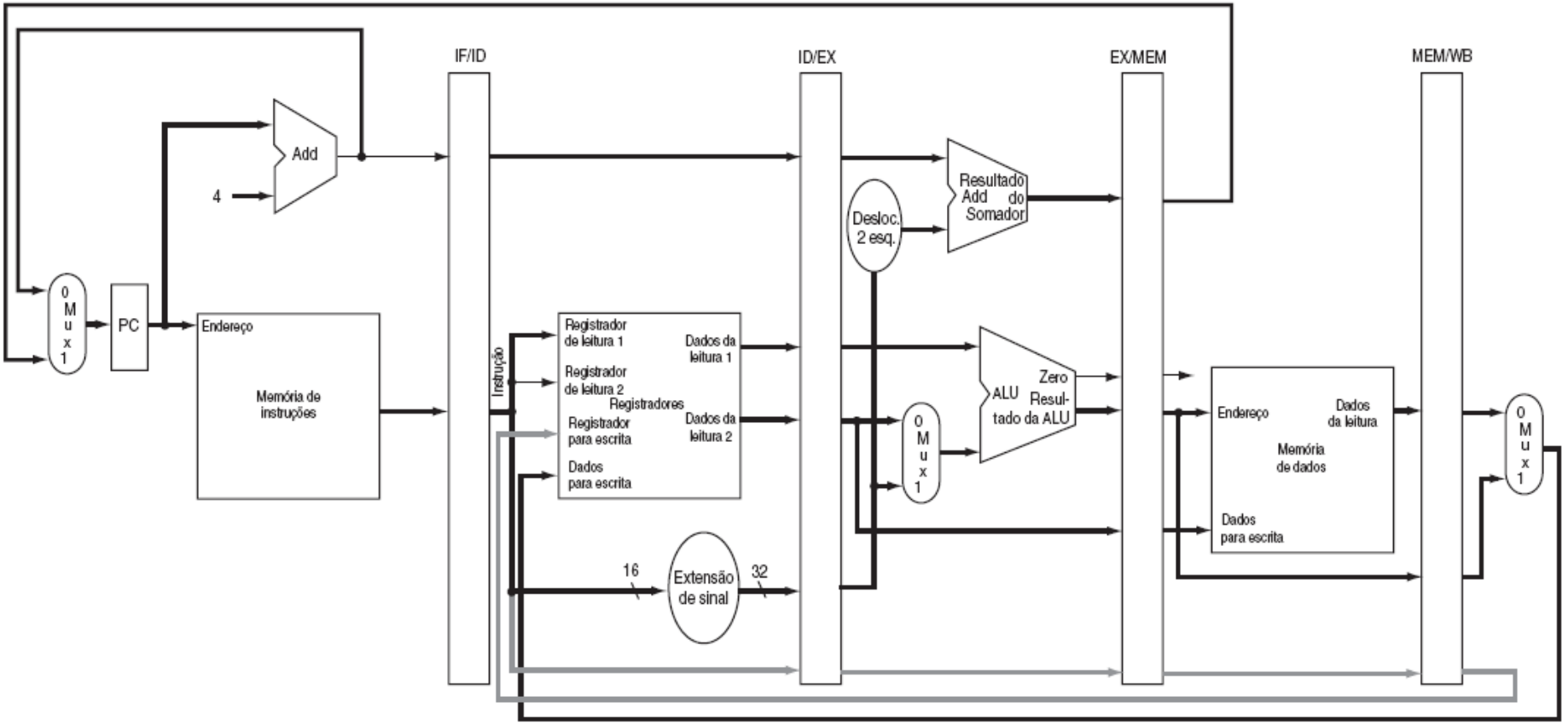


Caminho de Dados usando Pipeline

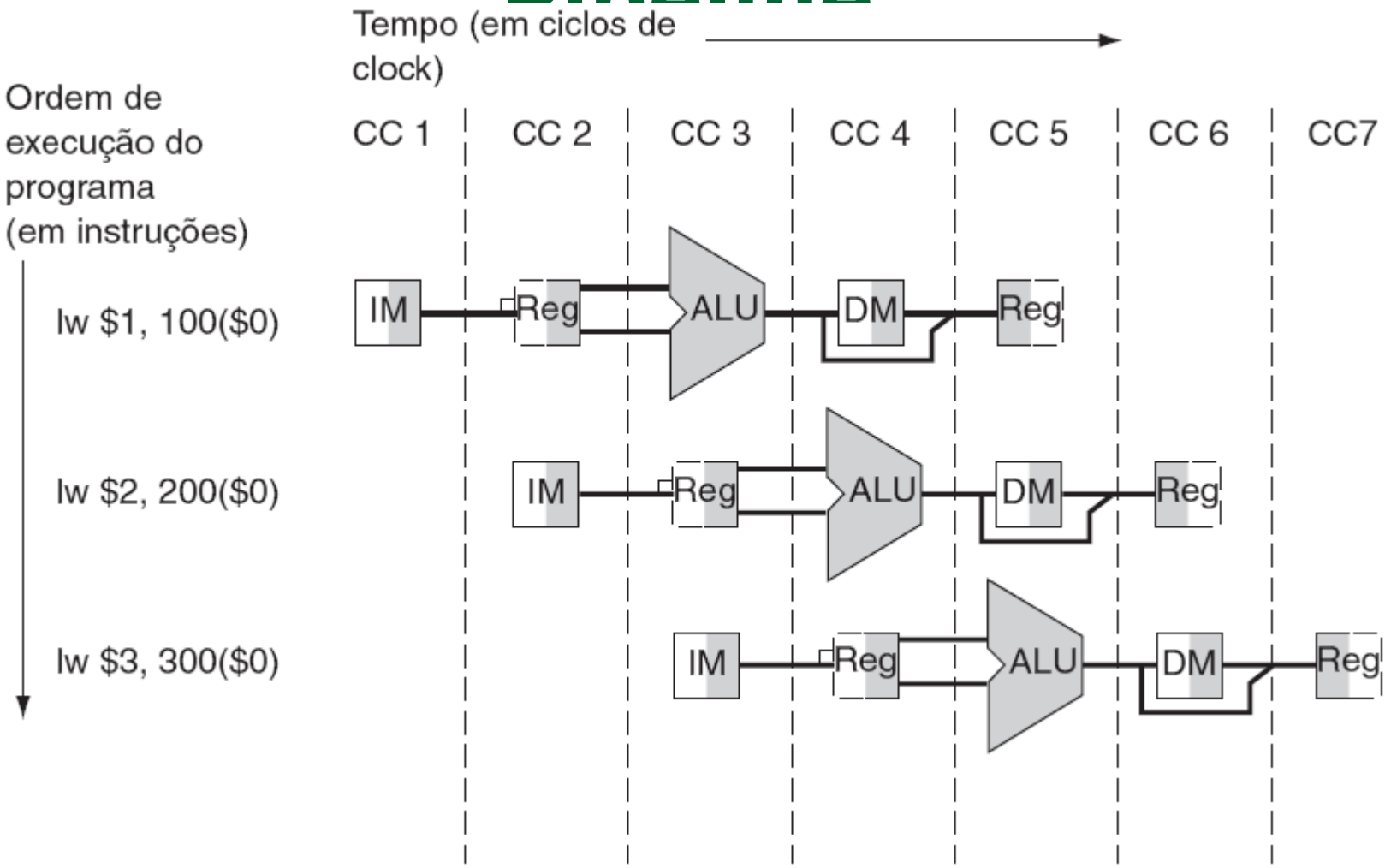
- Considerações:
  - No quarto estágio a memória de dados é lida por meio do endereço no registrador de pipeline EX/MEM, e os dados são colocados no registrador de pipeline MEM;WB;
  - No quinto e último estágio, os dados são lidos do registrador de pipeline MEM/WB e escritos no banco de registradores no meio do caminho de dados;
  - Qual o bug que existe no projeto da instrução load? Por exemplo, qual a instrução fornece o número do registrador de escrita? Resposta: IF/ID, logo precisamos preservar o número do registrador de destino na instrução load;

Caminho de Dados usando Pipeline

- Caminho de dados em pipeline corrigido para lidar corretamente com a instrução load:



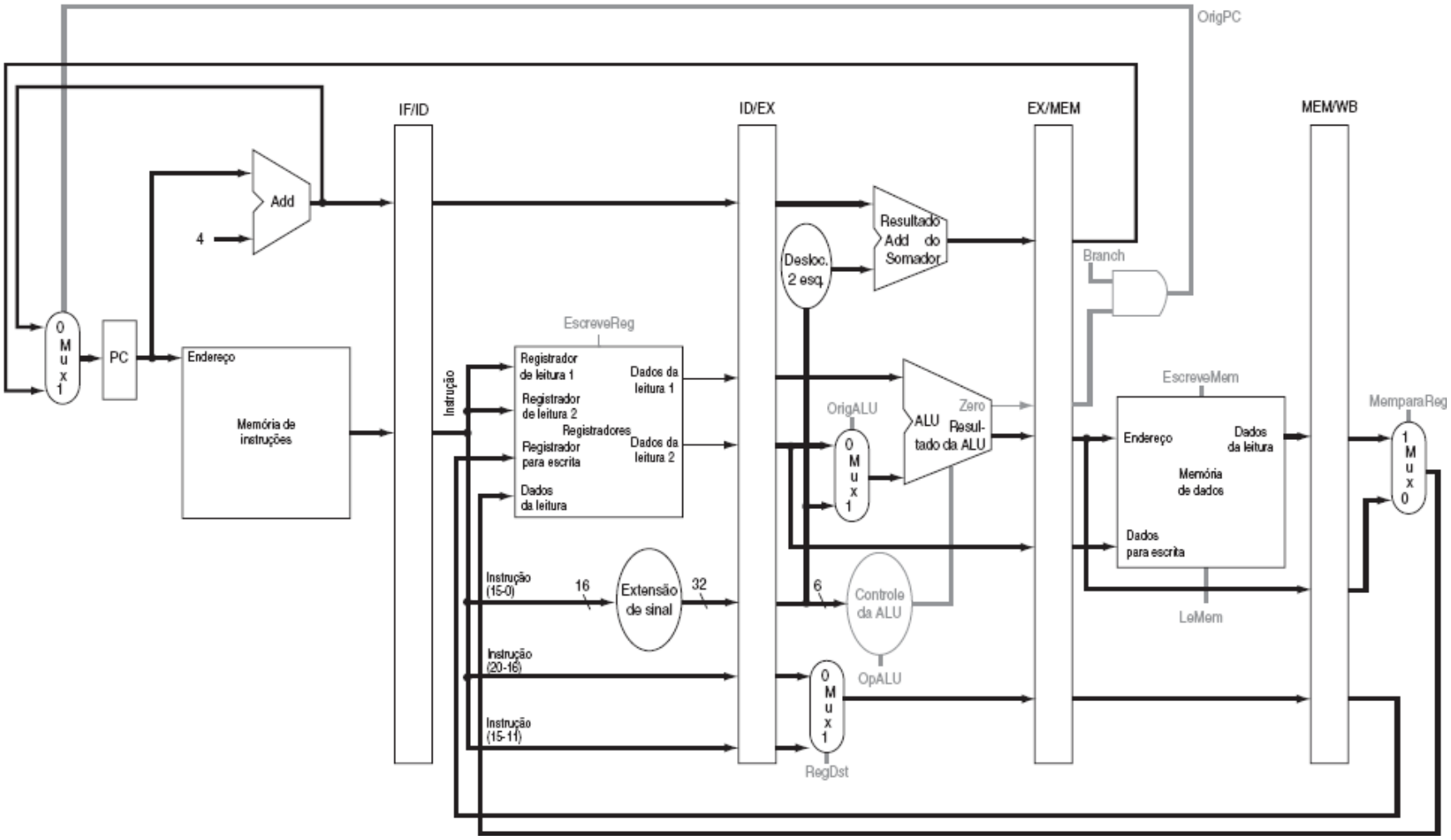
Representação Gráfica do Pipeline



Representação Gráfica do Pipeline

- Considerações:
  - Pode ajudar a responder questões como estas:
    - quantos ciclos leva para executar esse código
    - qual é a ALU sendo executada durante o ciclo 4?
    - use essa representação para ajudar a entender os caminhos de dados

Controle de um Pipeline



Controle de um Pipeline

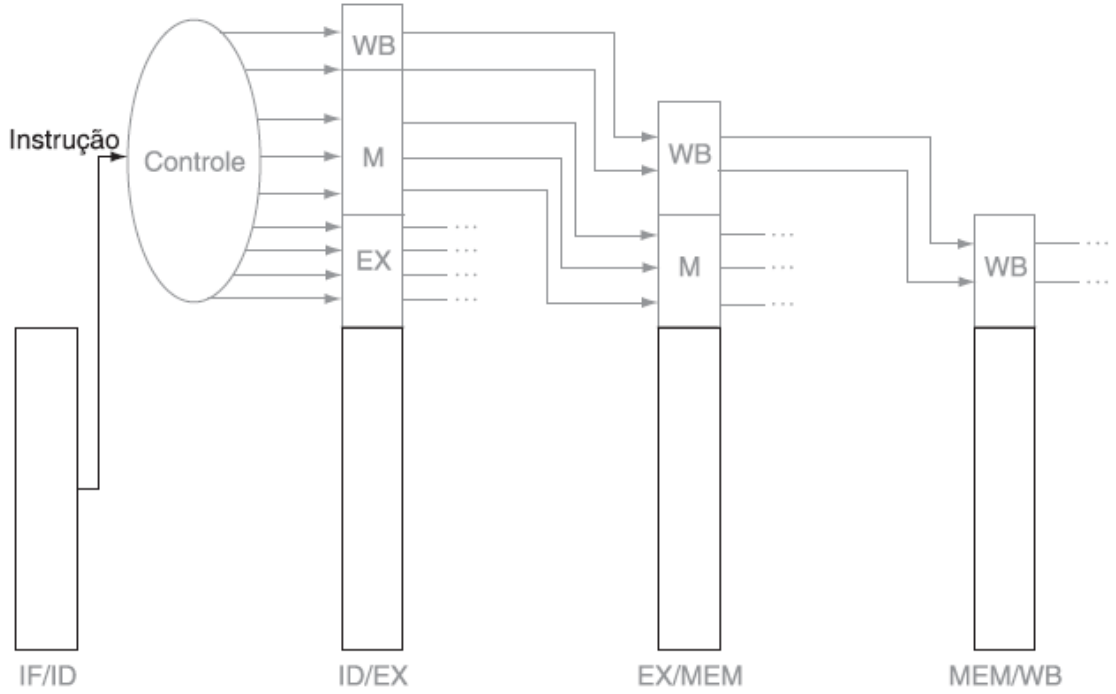
- As duas tabelas são idênticas, mas a segunda foi organizada em três grupos, correspondentes aos três últimos estágios do pipeline:

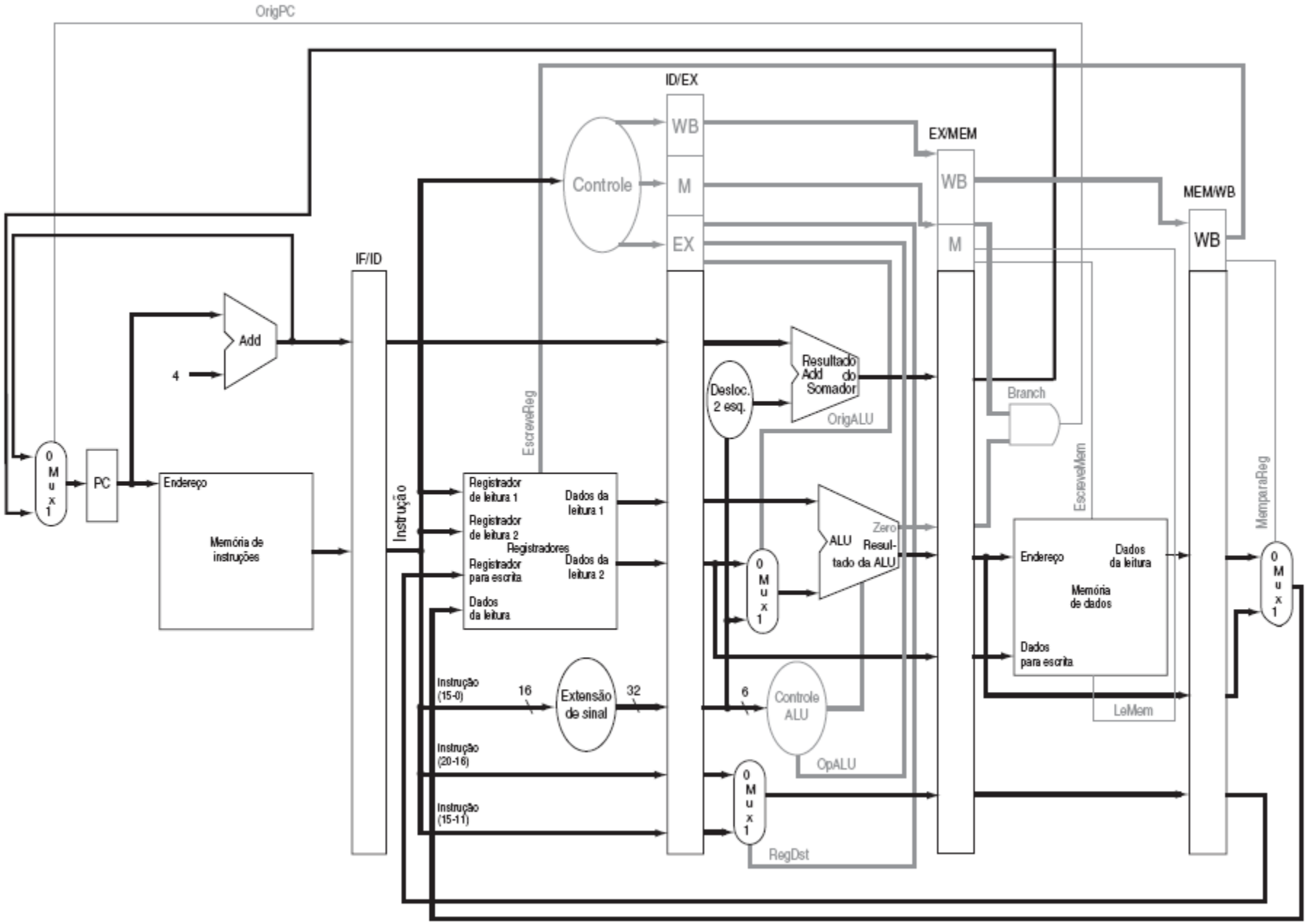
Instrução	Linhas de controle do estágio de cálculo de endereço/execução				Linhas de controle do estágio de acesso à memória			Linhas de controle do estágio de escrita do resultado	
	RegDst	OpALU1	OpALU0	OrigALU	Branch	LeMem	Escreve Mem	Escreve Reg	Mem para Reg
Formato R	1	1	0	0	0	0	0	1	0
lw	0	0	0	1	0	1	0	1	1
sw	X	0	0	1	0	0	1	0	X
beq	X	0	1	0	1	0	0	0	X

Instrução	RegDst	OrigALU	Mempara Reg	Escreve Reg	Le Mem	Escreve Mem	Branch	ALUOpl	ALUOp0
formato 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

Controle de um Pipeline

- Antes de finalizar nosso pipeline com sinais de controle, precisamos inserir sinais nos registradores de pipeline:




OrigPC
ID/EX
EX/MEM
MEM/WB
WB
M
EX
Controle
IF/ID
Add
4
0
M
u
x
1
PC
Endereço
Memória de instruções
Instrução
EscreveReg
Registrador de leitura 1
Registrador de leitura 2
Registradores
Registrador para escrita
Dados da leitura
Dados da leitura 1
Dados da leitura 2
Desloc. 2 esq.
Resultado Add do Somador
OrigALU
Branch
ALU
Zero
Resul-tado da ALU
Instrução (15-0)
16
Extensão de sinal
32
6
Controle ALU
OpALU
Instrução (20-16)
Instrução (15-11)
RegDst
EscreveMem
LeMem
MemparaReg
Endereço
Dados da leitura
Memória de dados
Dados para escrita

Forwarding

- Bastante usado para solucionar o risco de dados, o forwarding, usa resultados temporários antes mesmo que eles sejam escritos;

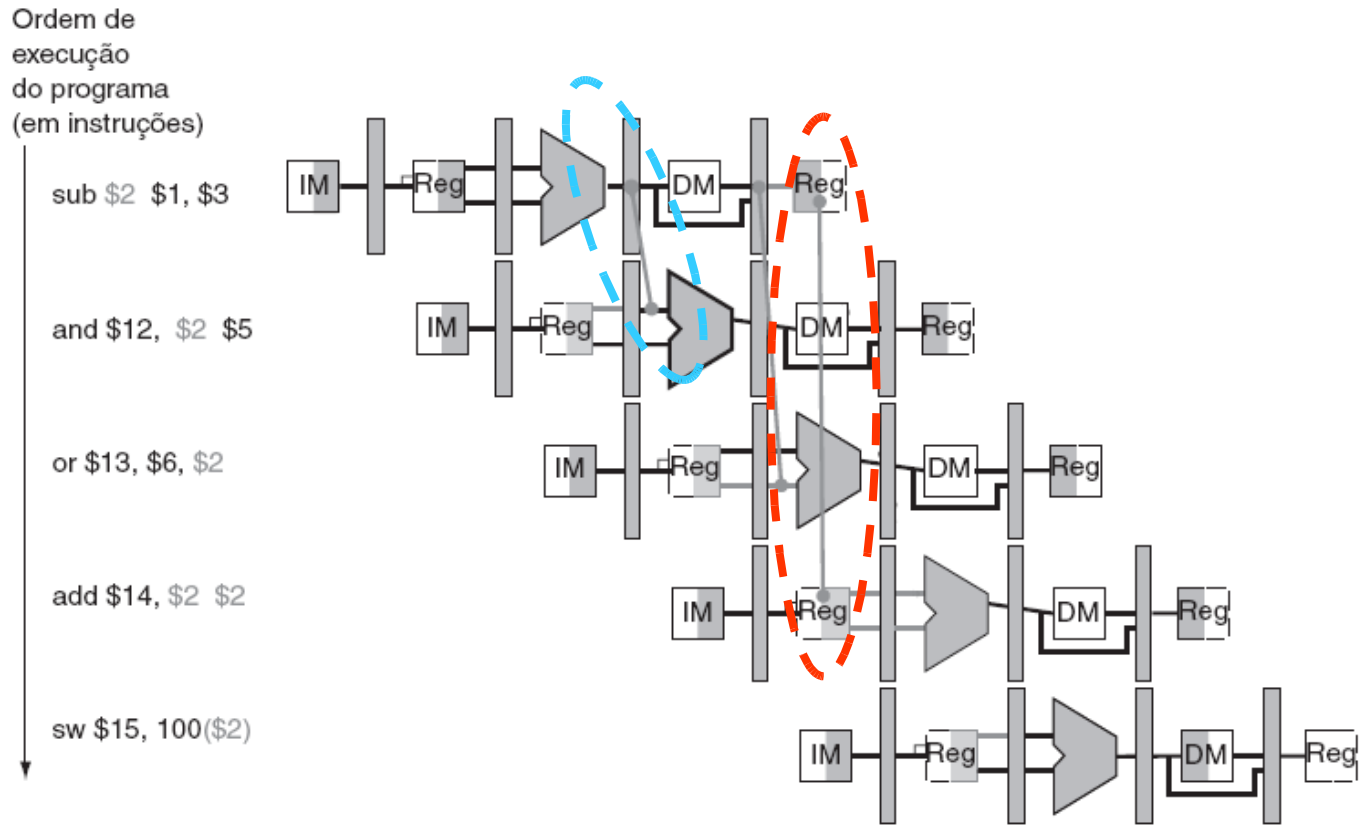
- Existem dois tipos de forwarding:
  - forwarding do arquivo de registrador para manipular read/write para o mesmo registrador
  - forwarding da ALU;

- Vejamos na figura a seguir uma representação gráfica do forwarding;

Forwarding

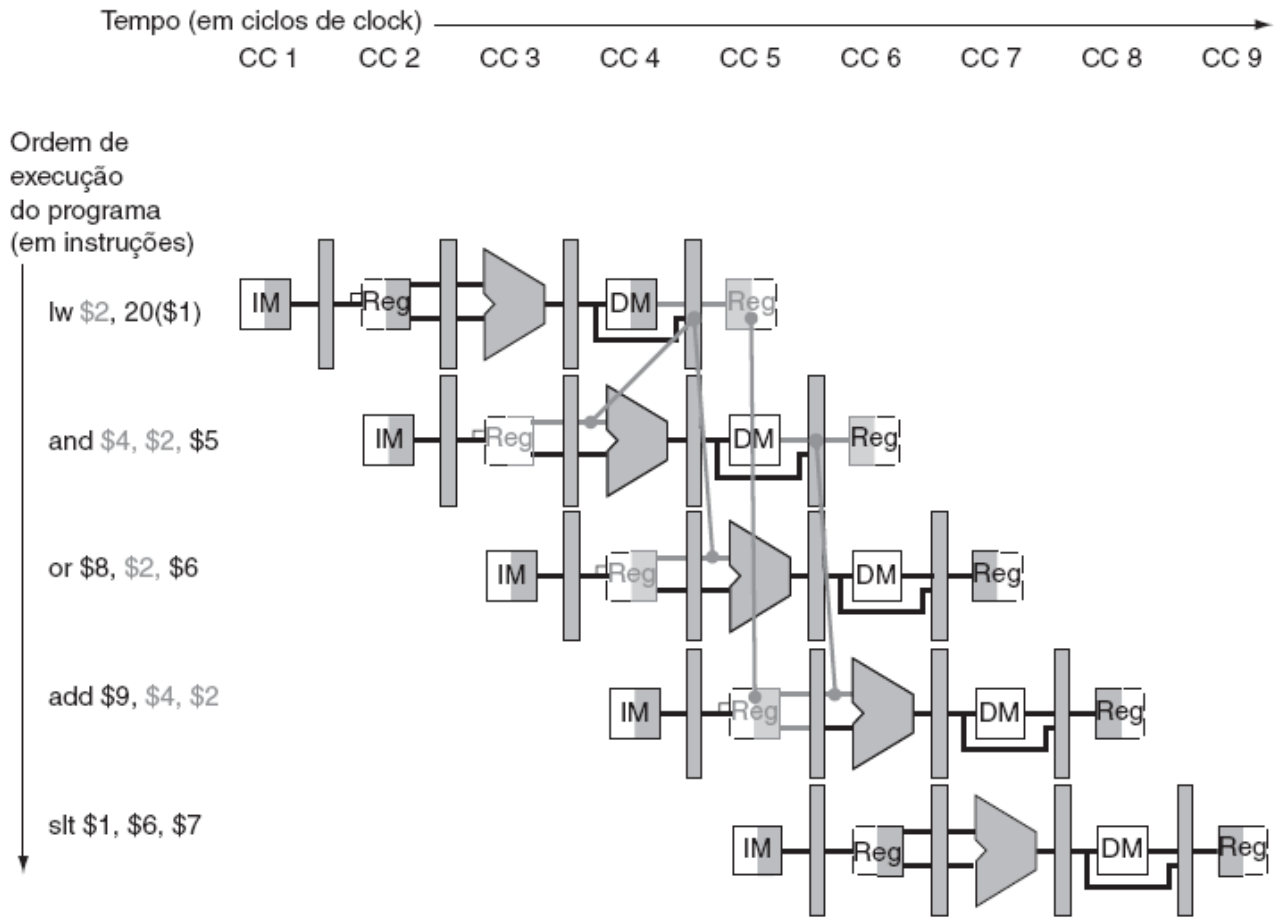
Tempo (em ciclos de clock) →

	CC 1	CC 2	CC 3	CC 4	CC 5	CC 6	CC 7	CC 8	CC 9
Valor do registrador $2:	10	10	10	10	10/–20	–20	–20	–20	–20
Valor de EX/MEM:	X	X	X	–20	X	X	X	X	X
Valor de MEM/WB:	X	X	X	X	–20	X	X	X	X



Forwarding

- Nem sempre é possível aplicar o forwarding, vejamos:



Bibliografia

- PATERSON, D. A. & HENNESSY, J. L. *Organização e Projeto de Computadores: a interface hardware/software,* Editora Campus. 3ª ed. RJ: 2005.