



Universidade de Brasília

Departamento de Ciência da Computação

Aula 17

Implementação RISC-V

Pipeline – Unidade Operativa e Controle



Controle do pipeline

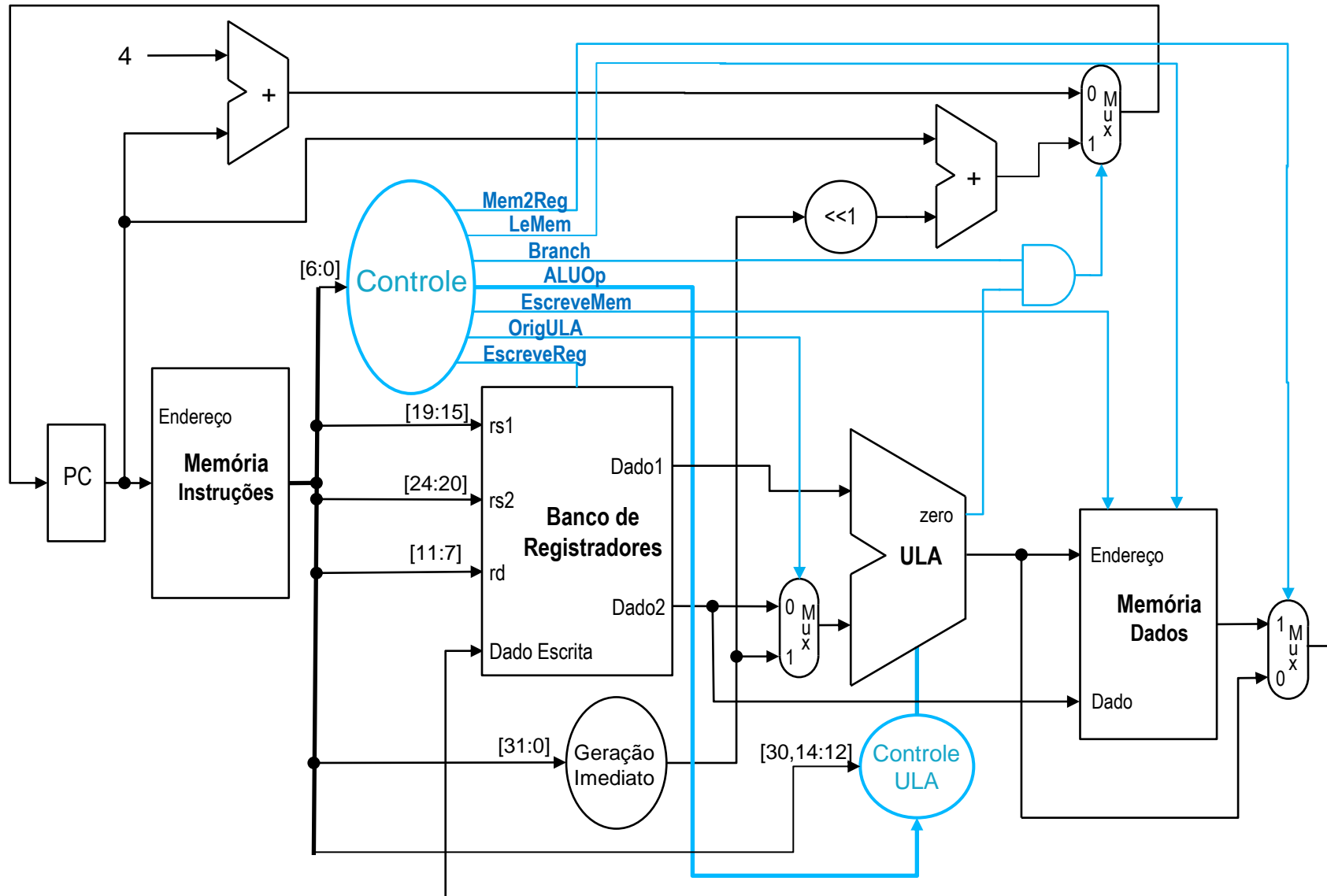
- Temos 5 estágios.

O que deve ser controlado em cada estágio?

IF:	<i>Instruction Fetch e PC Increment</i>
ID:	<i>Instruction Decode / Register Fetch</i>
EX:	<i>Execution</i>
MEM:	<i>Memory Stage</i>
WB:	<i>Write Back</i>

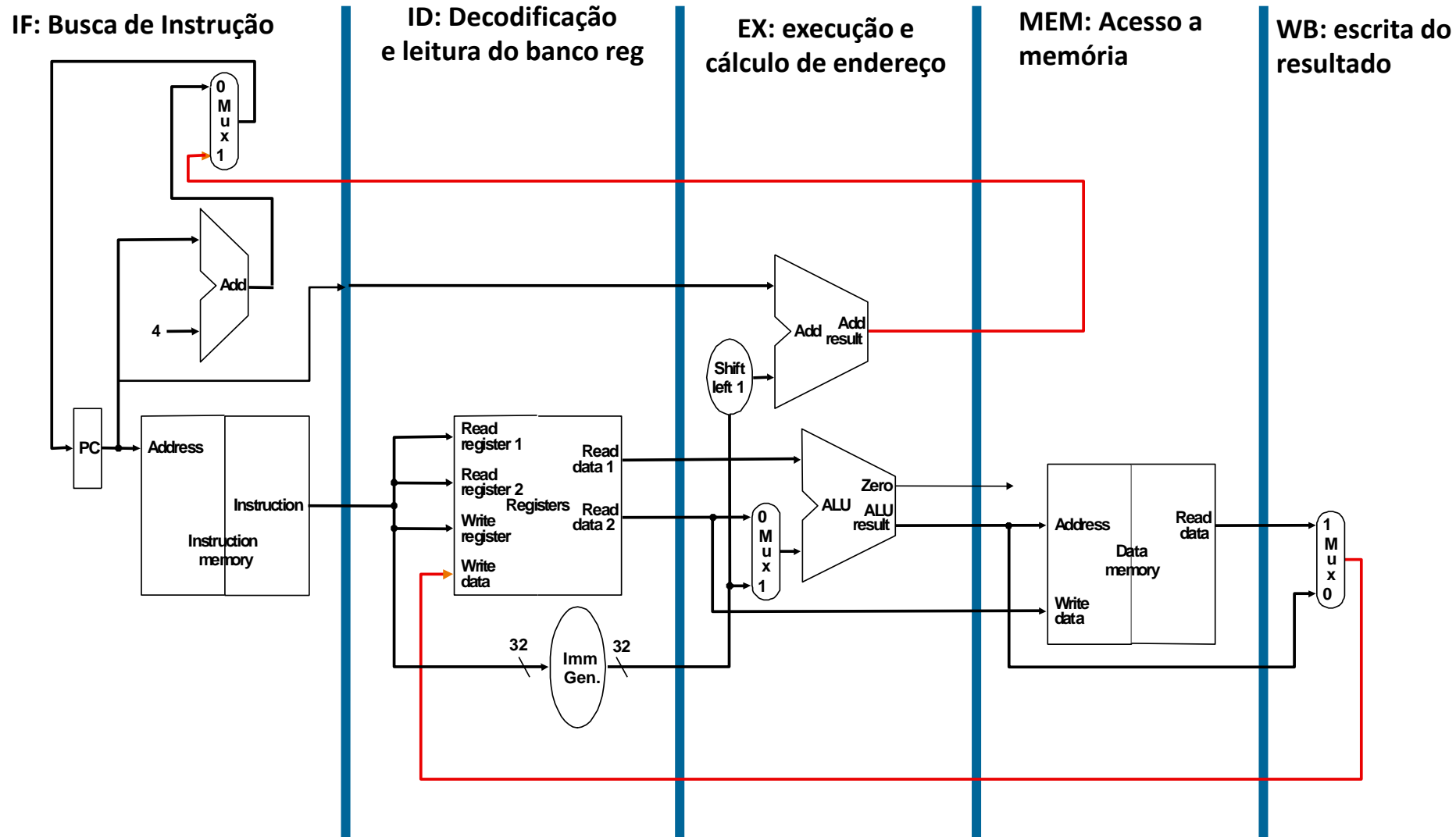


Caminho de Dados UNICICLO





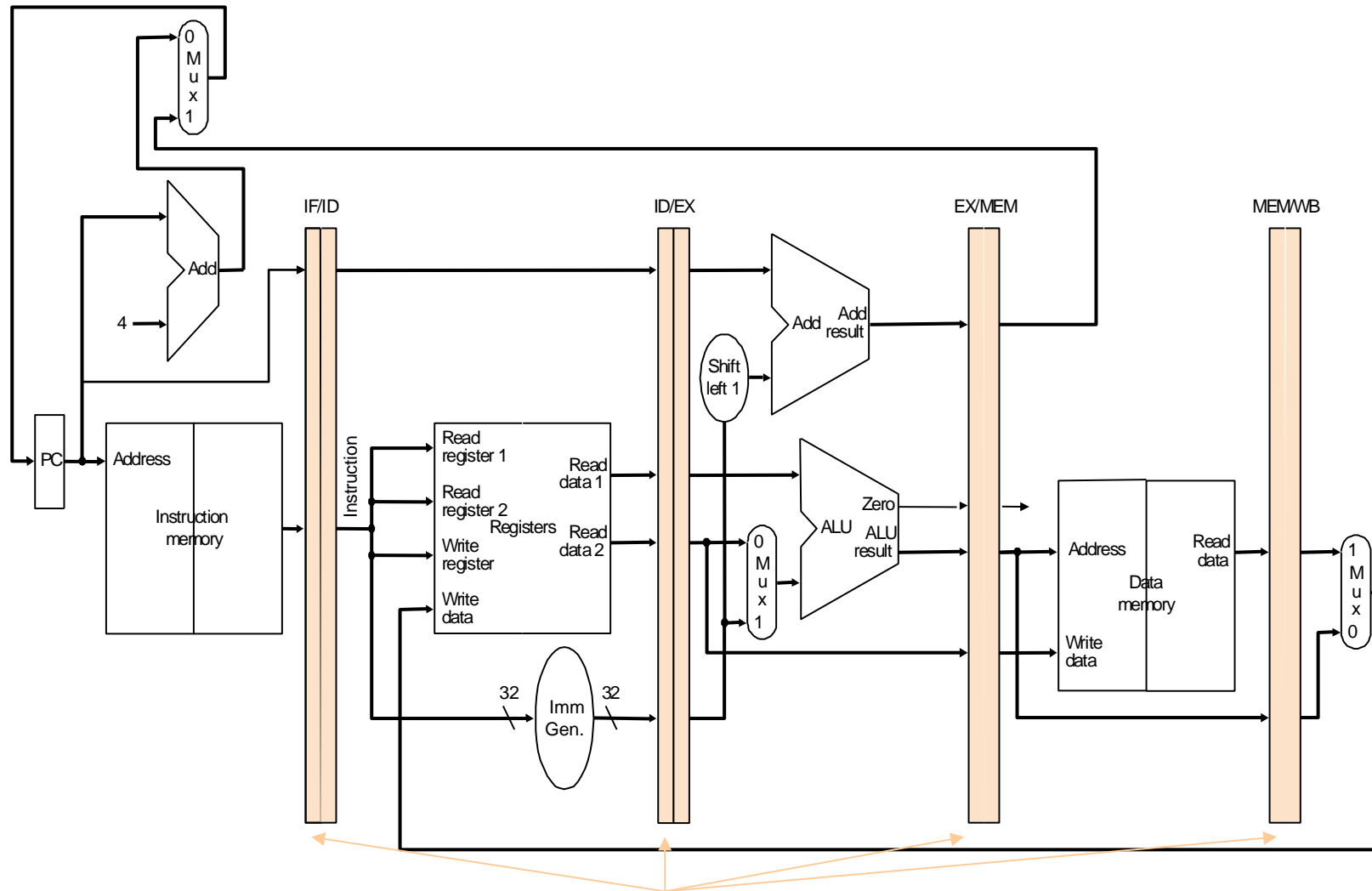
Caminho de dados UNICICLO



Fluxo de dados → com 2 exceções



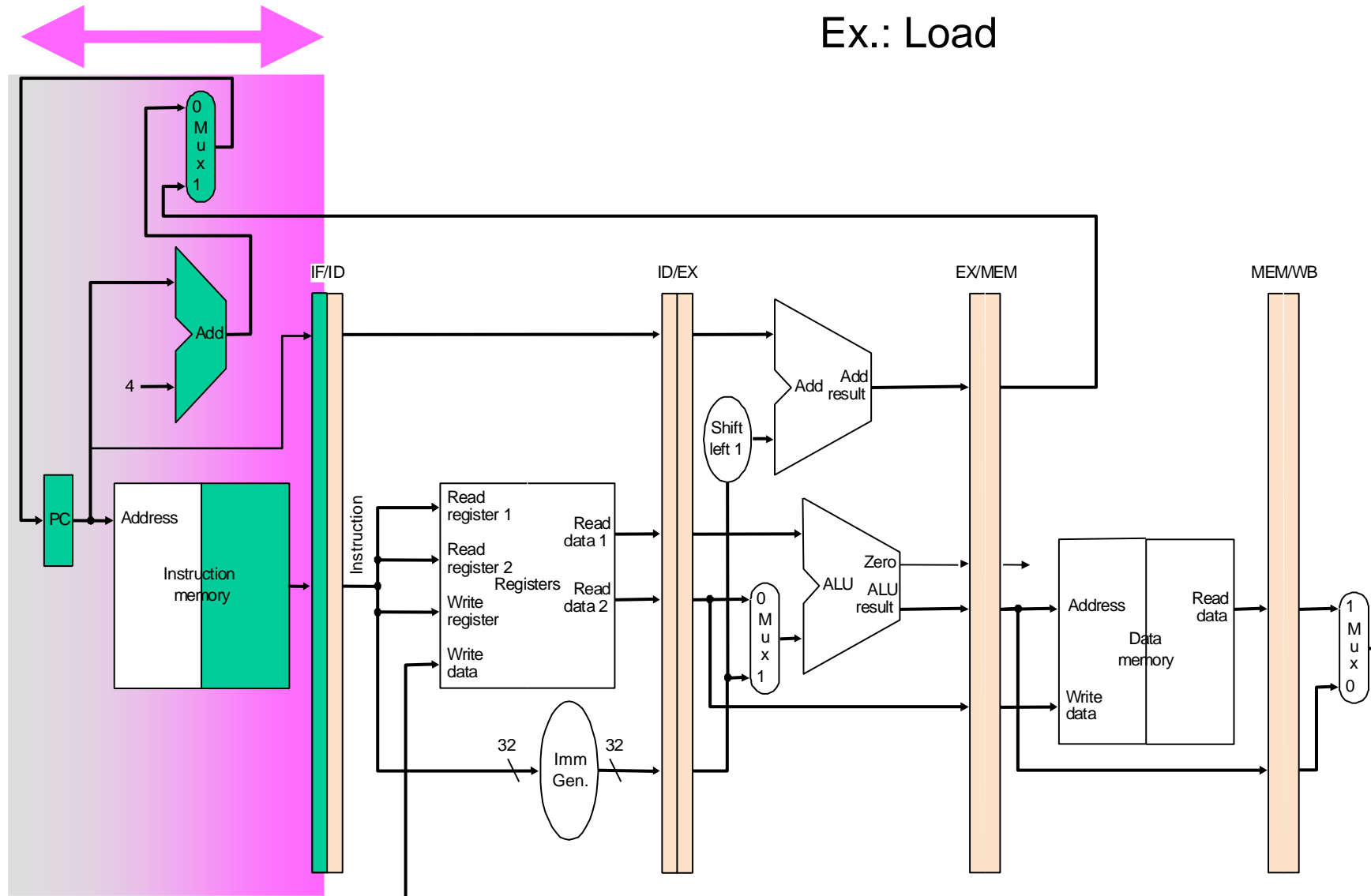
Caminho de dados com *pipeline*



Registradores dos estágios do pipeline
Quais seus tamanhos?

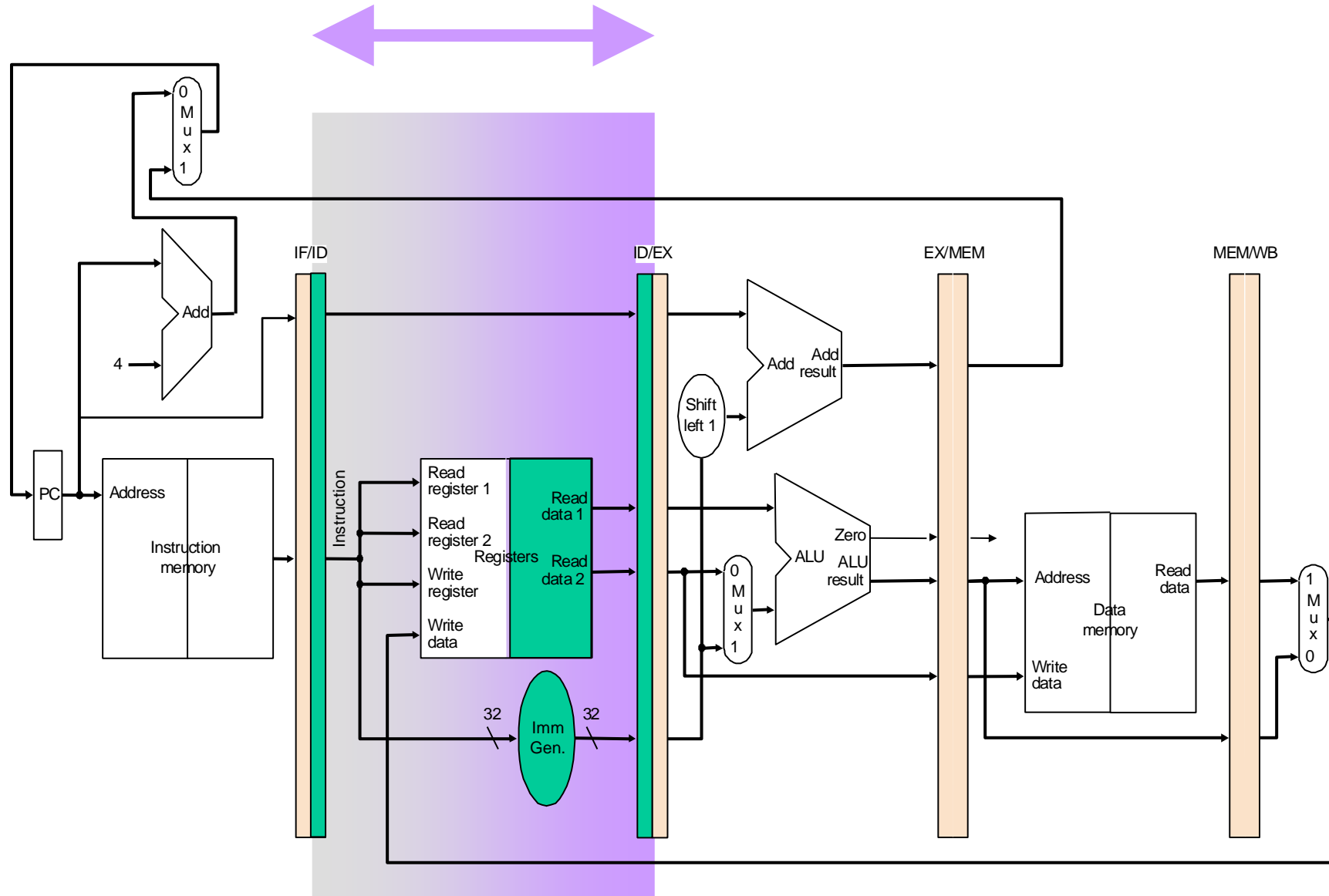
Busca de instrução

Ex.: Load



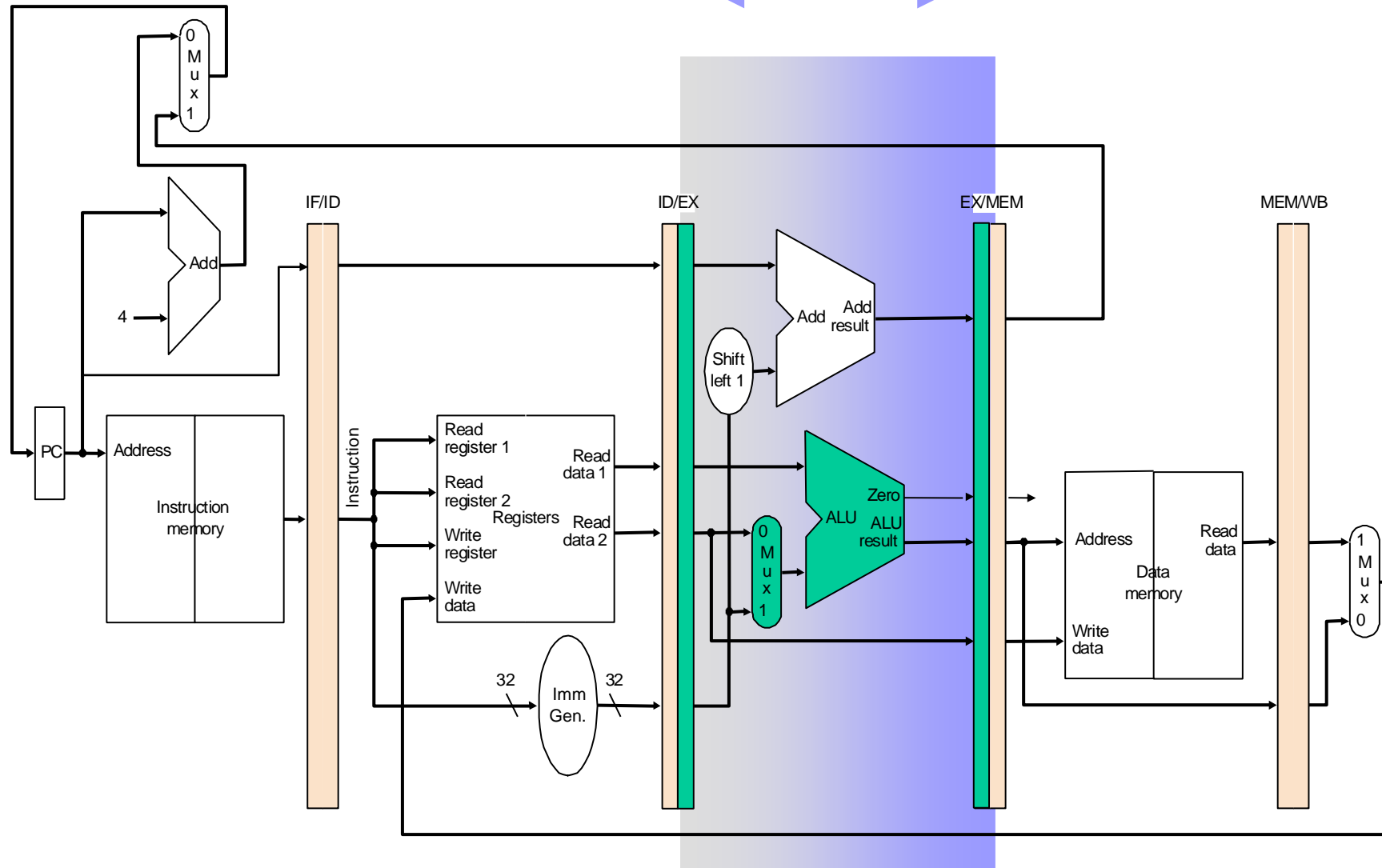


Decodificação/leitura de registros



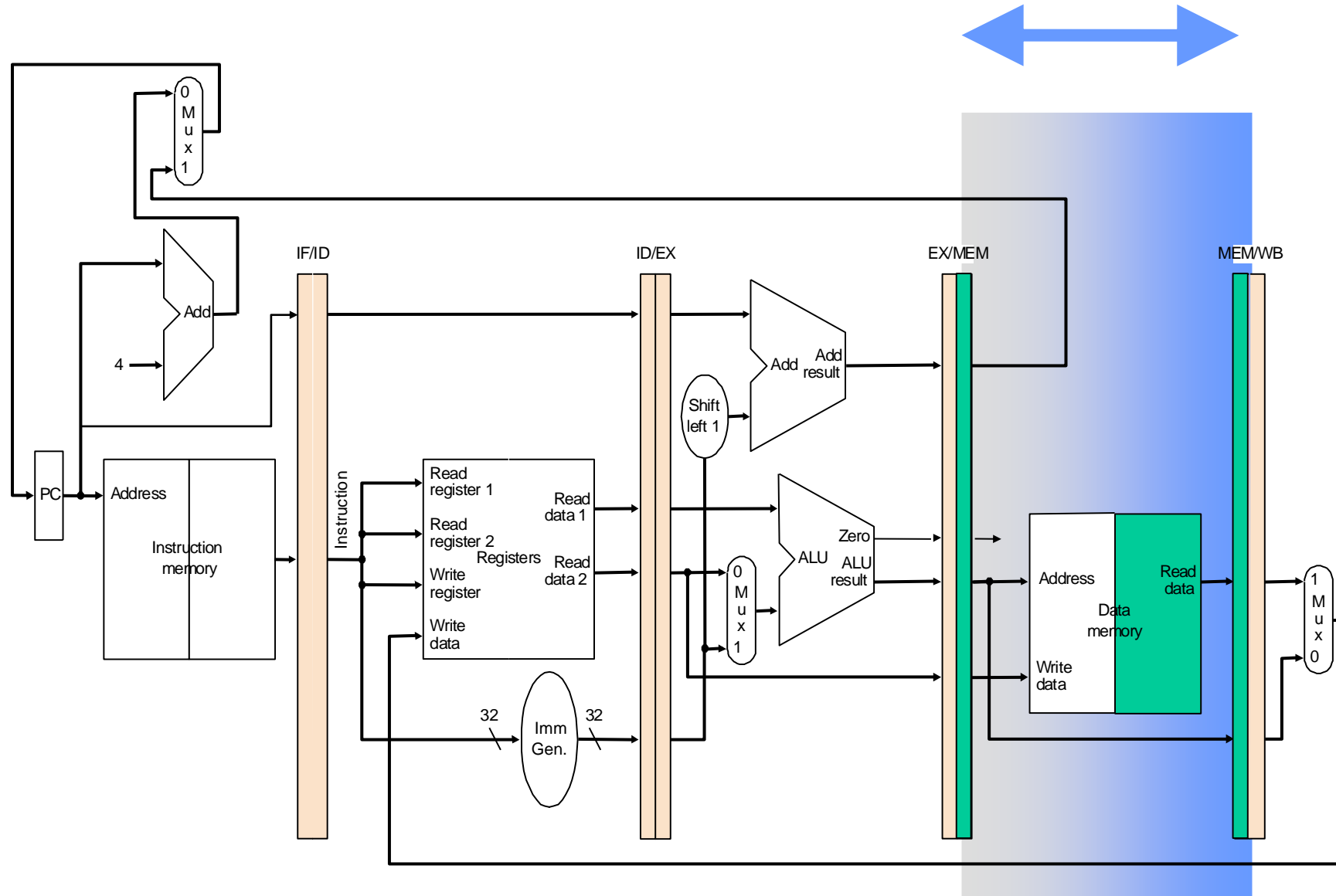


Execução/cálculo de endereço



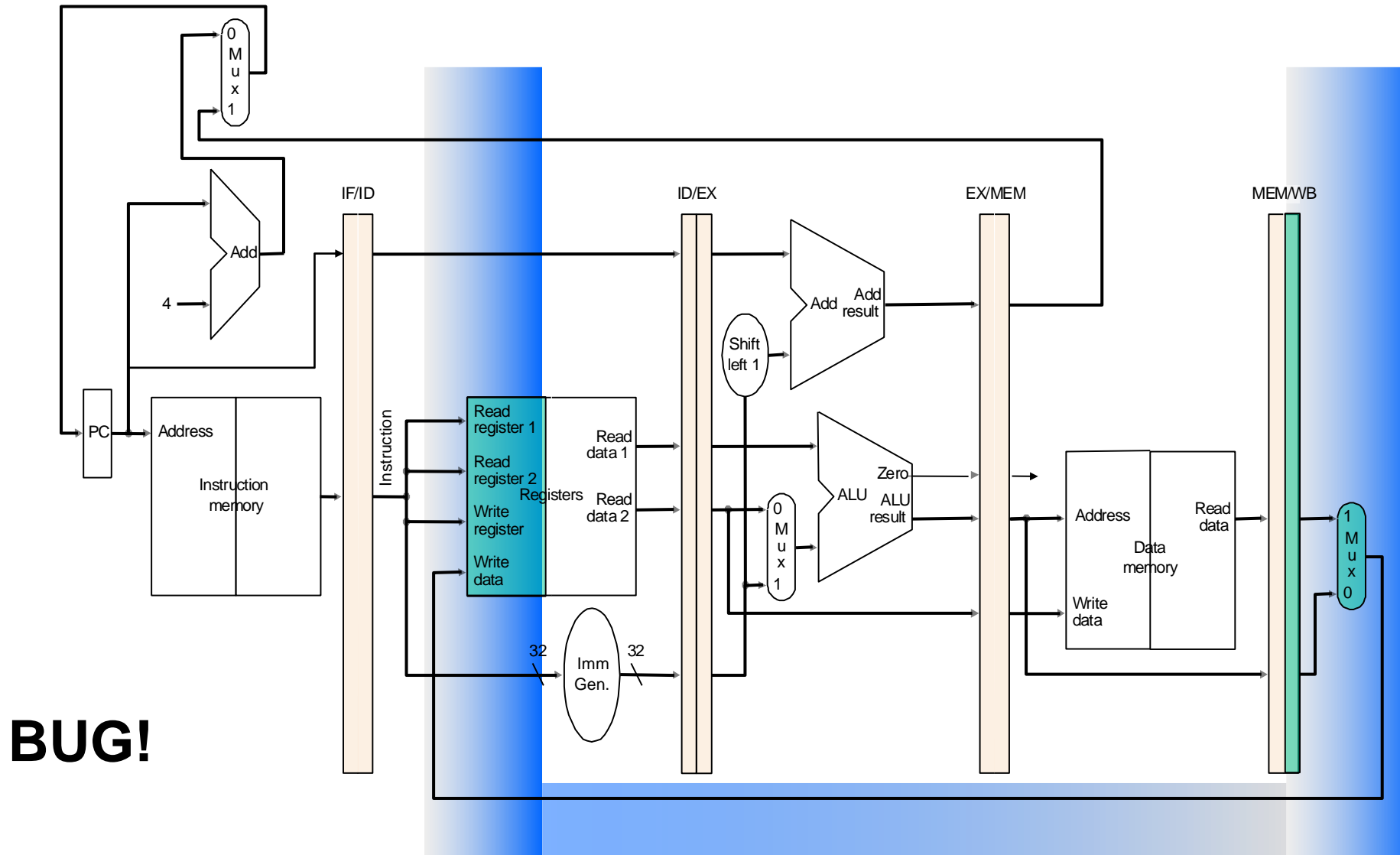


Acesso à memória



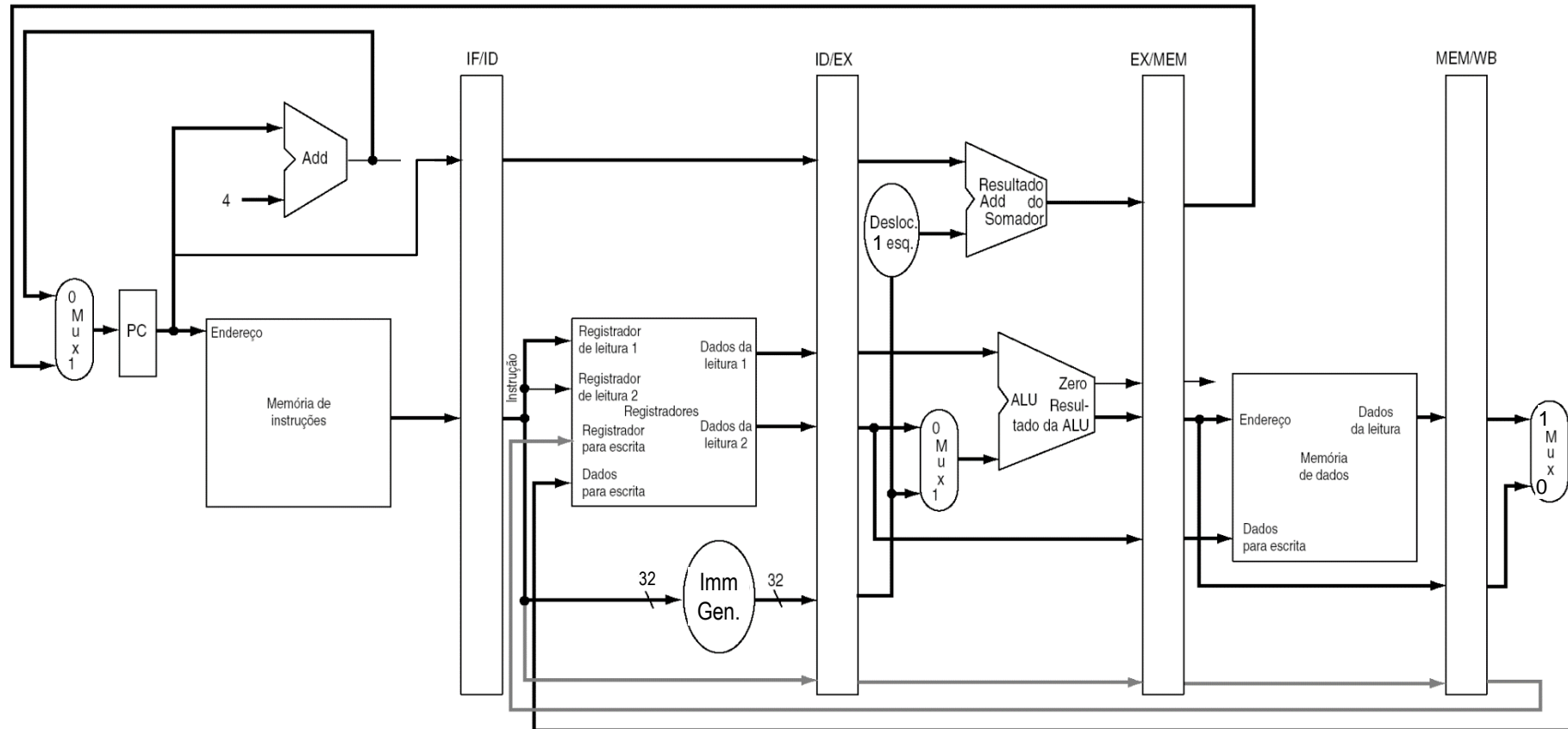


Escrita do Resultado



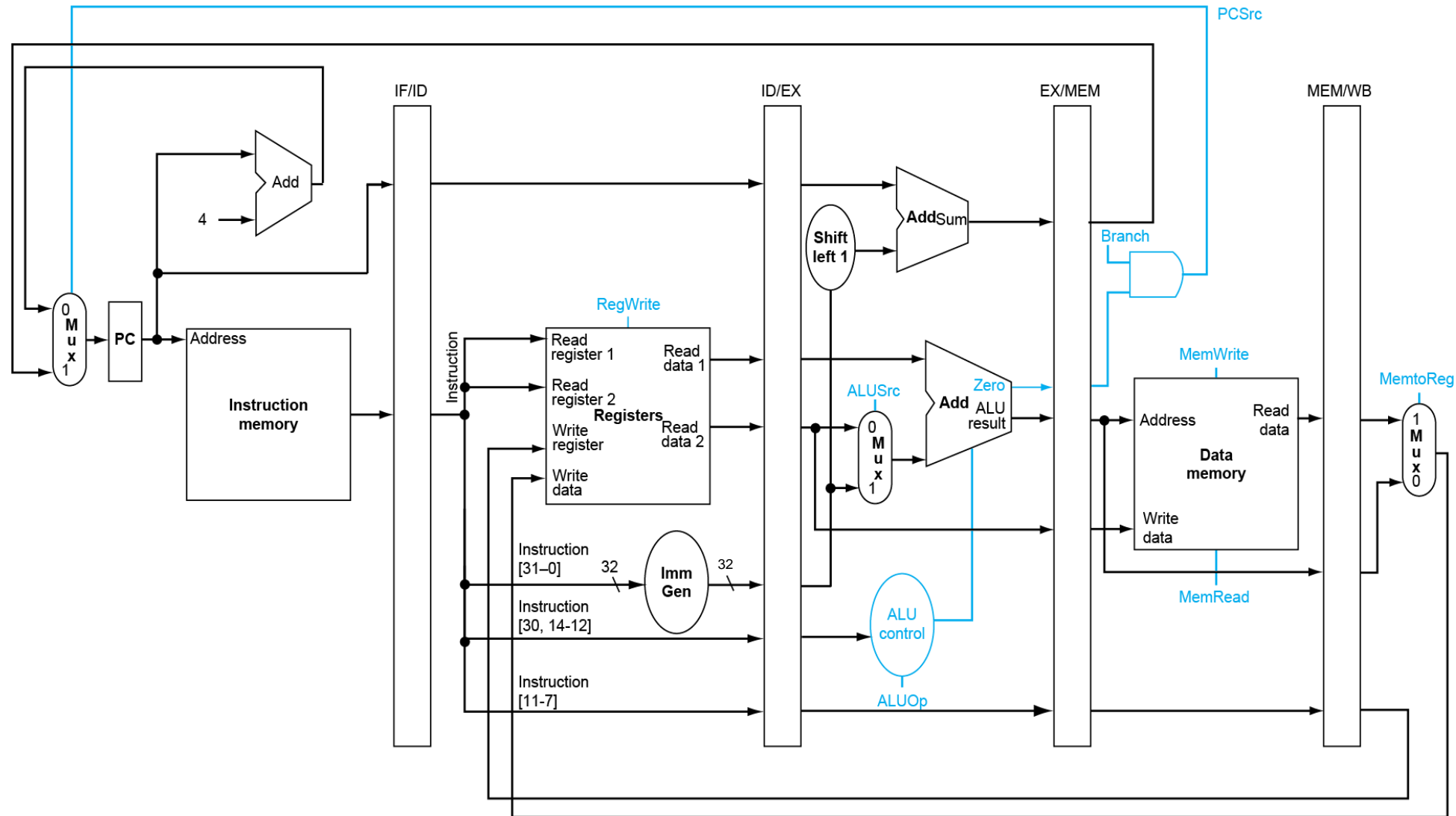


Corrigindo o BUG!!!





Controle do pipeline – Sinais de Controle





Controle do pipeline

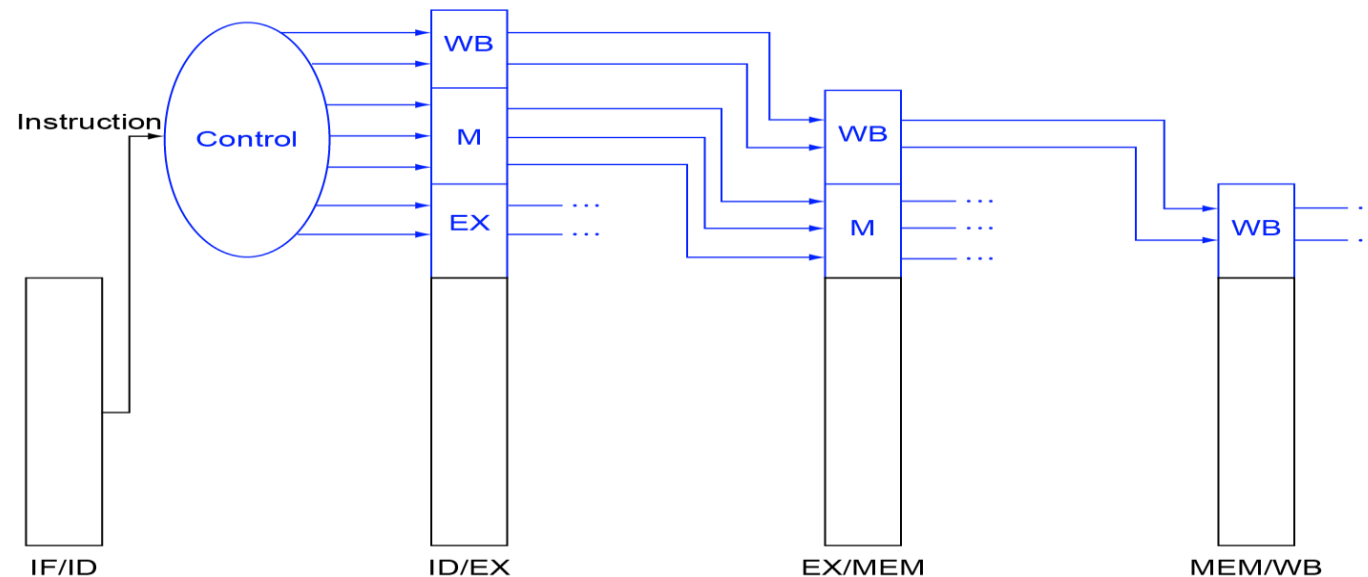
- Temos 5 estágios. O que precisa ser controlado em cada estágio?
 - Busca da instrução:
 - Nenhum sinal de controle
 - Decodificação da instrução / leitura do banco de regs.:
 - Nenhum sinal de controle
 - Execução / cálculo de endereço:
 - ALUOp, ALUSrc
 - Acesso a memória:
 - Branch, MemRead, MemWrite
 - Escrita do resultado
 - Mem2Reg, RegWrite



Controle do *Pipeline*

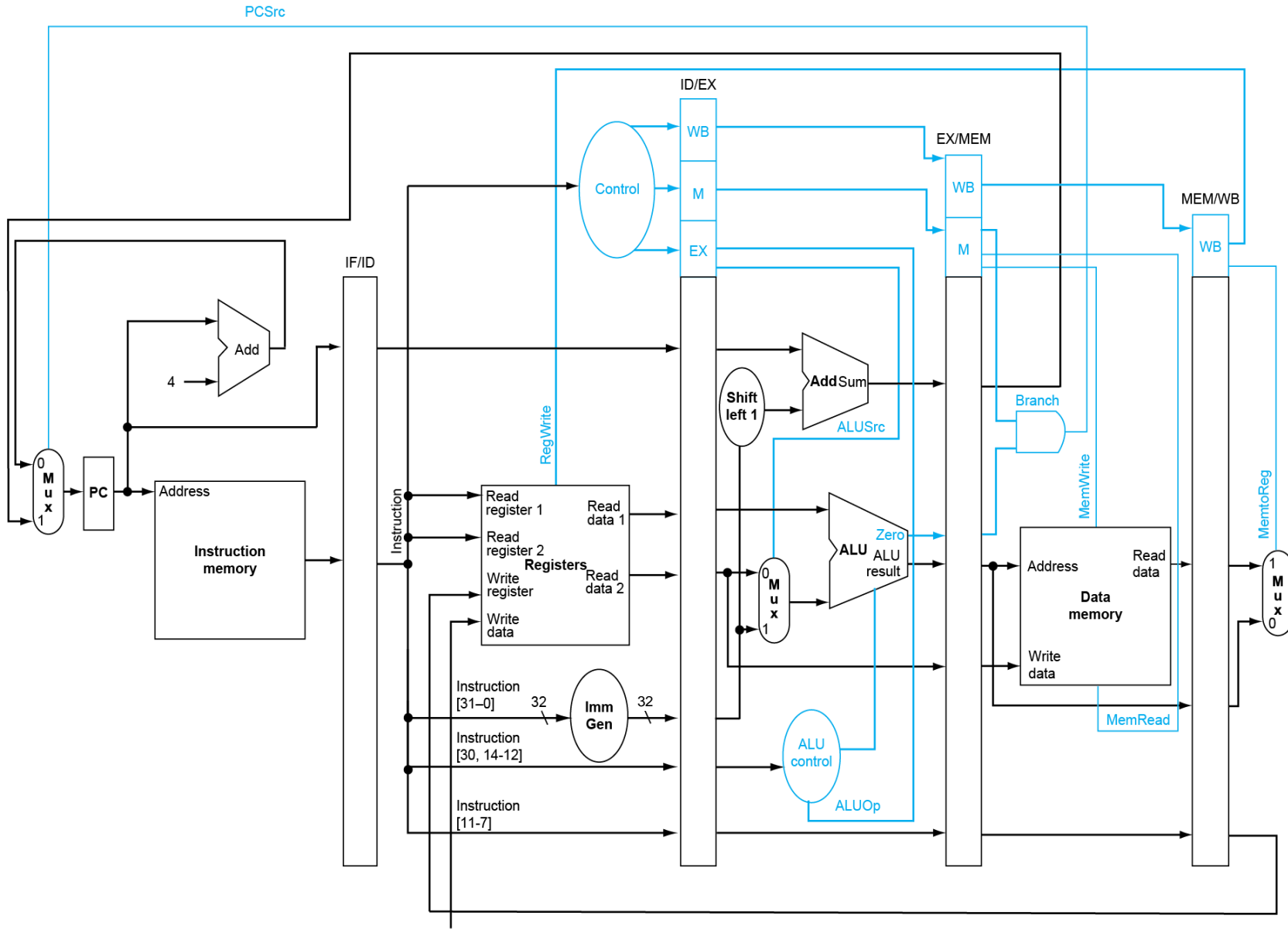
Sinais de controle são transmitidos da mesma forma que os dados

	Estágio de EX		Estágio MEM			Estágio WB	
Instrução	ALUSrc	ALUOp	Branch	MemRead	MemWrite	RegWrite	Mem2Reg
Tipo-R	0	10	0	0	0	1	0
lw	1	00	0	1	0	1	1
sw	1	00	0	0	1	0	X
beq	0	01	1	0	0	0	X





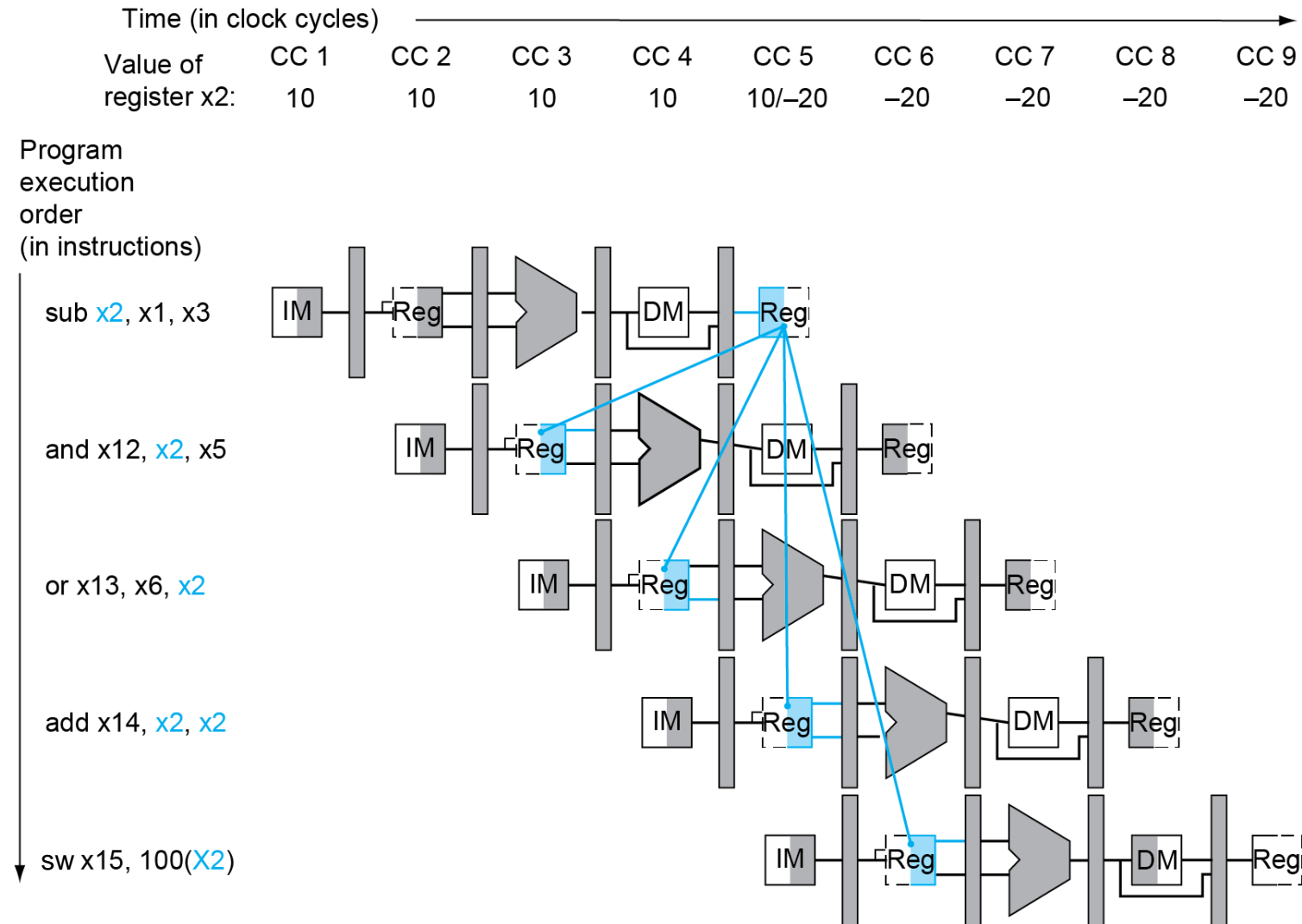
Caminho de dados pipeline com controle





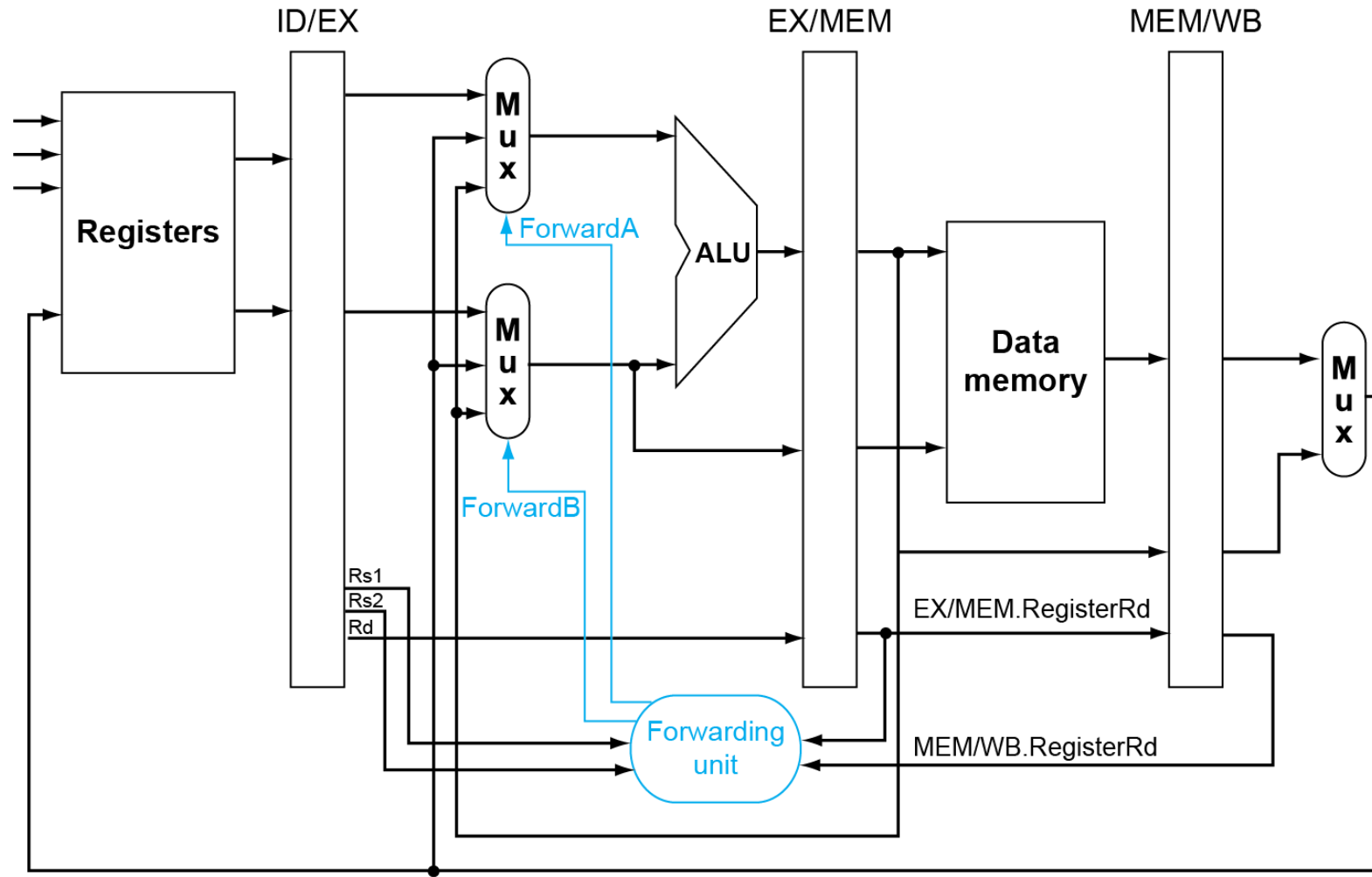
HAZARDS

■ Hazard de Dados: Forwarding





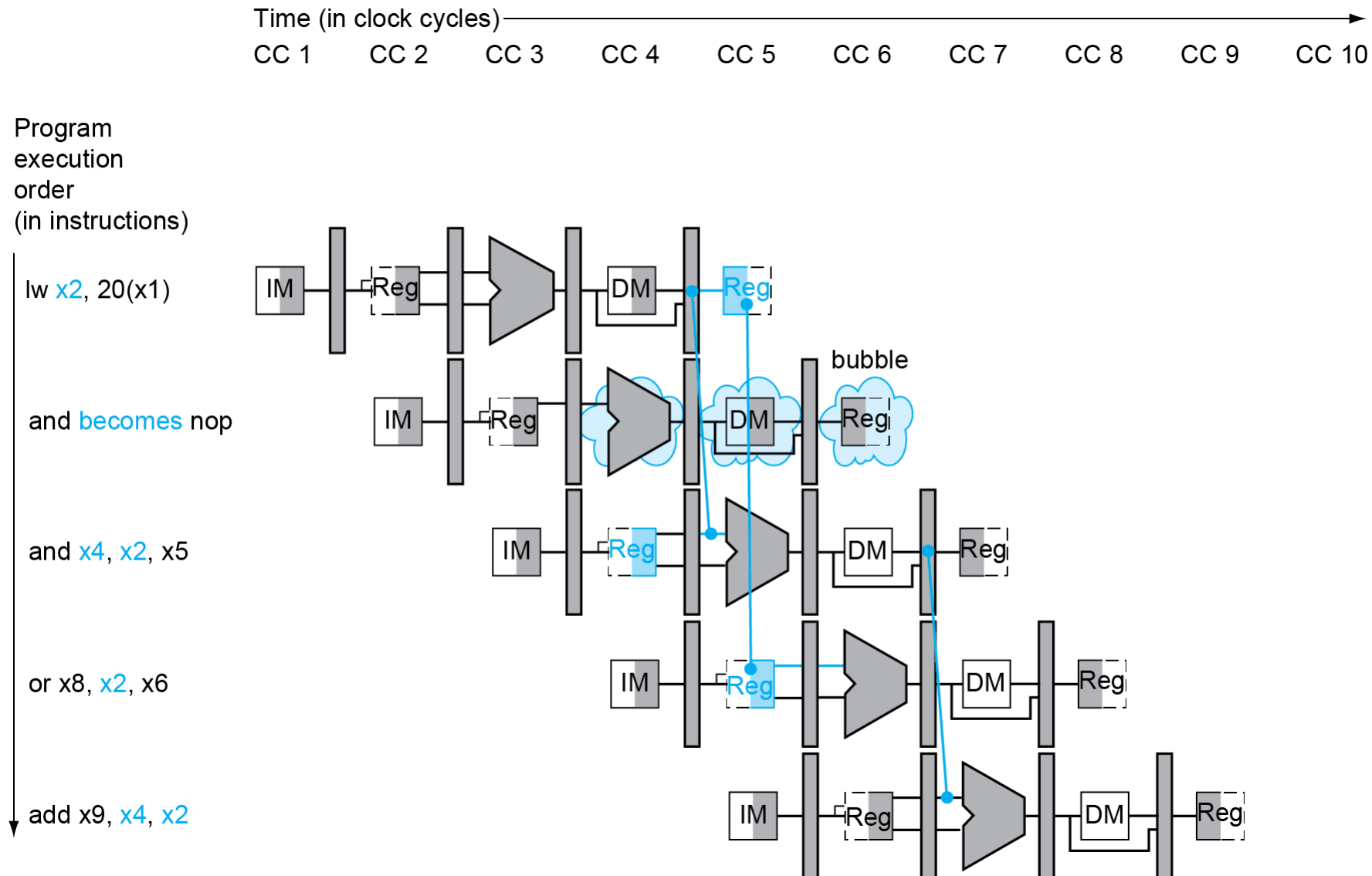
Hazard de dados: Forwarding

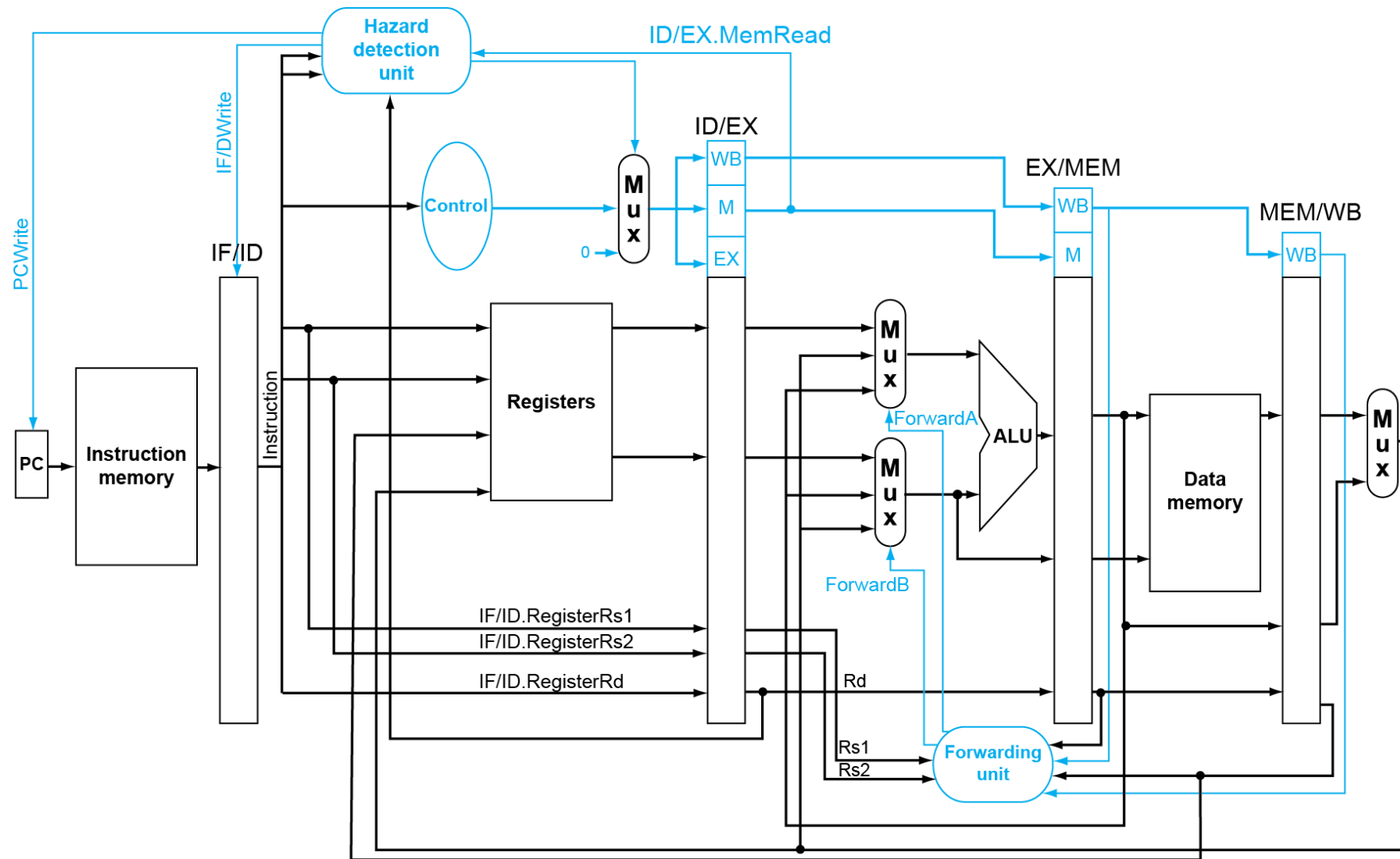


Obs.: Faltava o mux OrigALU após o ForwardB!



Quando forward não é possível







Deteção de Hazards

- Comparar índices dos registradores a serem lidos e escritos, armazenados nos registradores do *pipeline*:
- Deve-se comparar igualmente se a instrução escreve em um registrador (EscreveReg = 1) ou lê da memória (LeMem = 1)

Obs.:

Flush descarta uma instrução do pipeline.

Stall paralisa o pipeline.



Detecção de Hazard de dados

■ 1. Hazard EX

if (EX/MEM.EscreveReg
 and (EX/MEM.RegistradorRd \neq 0)
 and (EX/MEM.RegistradorRd = ID/EX.RegistradorRs1)) ForwardA = 10

if (EX/MEM.EscreveReg
 and (EX/MEM.RegistradorRd \neq 0)
 and (EX/MEM.RegistradorRd = ID/EX.RegistradorRs2)) ForwardB = 10

Não se deve dar forward no x0!!!

■ 2. Hazard MEM

if (MEM/WB.EscreveReg
 and (MEM/WB.RegistradorRd \neq 0)
 and (MEM/WB.RegistradorRd = ID/EX.RegistradorRs1)) ForwardA = 01

if (MEM/WB.EscreveReg
 and (MEM/WB.RegistradorRd \neq 0)
 and (MEM/WB.RegistradorRd = ID/EX.RegistradorRs2)) ForwardB = 01

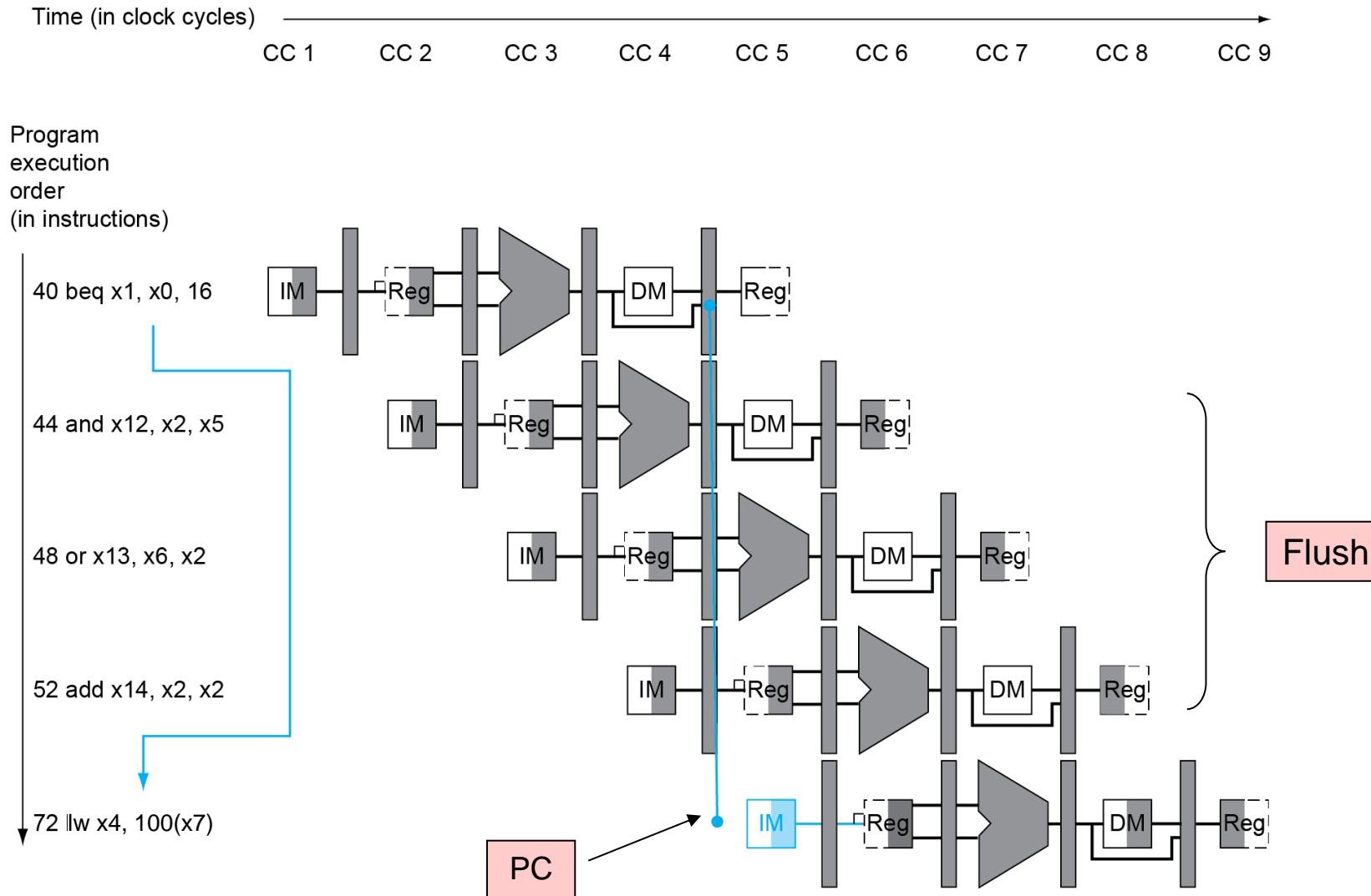
■ 3. LW

if (ID/EX.LeMem and ((ID/EX.RegistradorRd = IF/ID.RegistradorRs1)
 or (ID/EX.RegistradorRd = IF/ID.RegistradorRs2))) Ocasiona stall no pipeline



Hazard de Controle

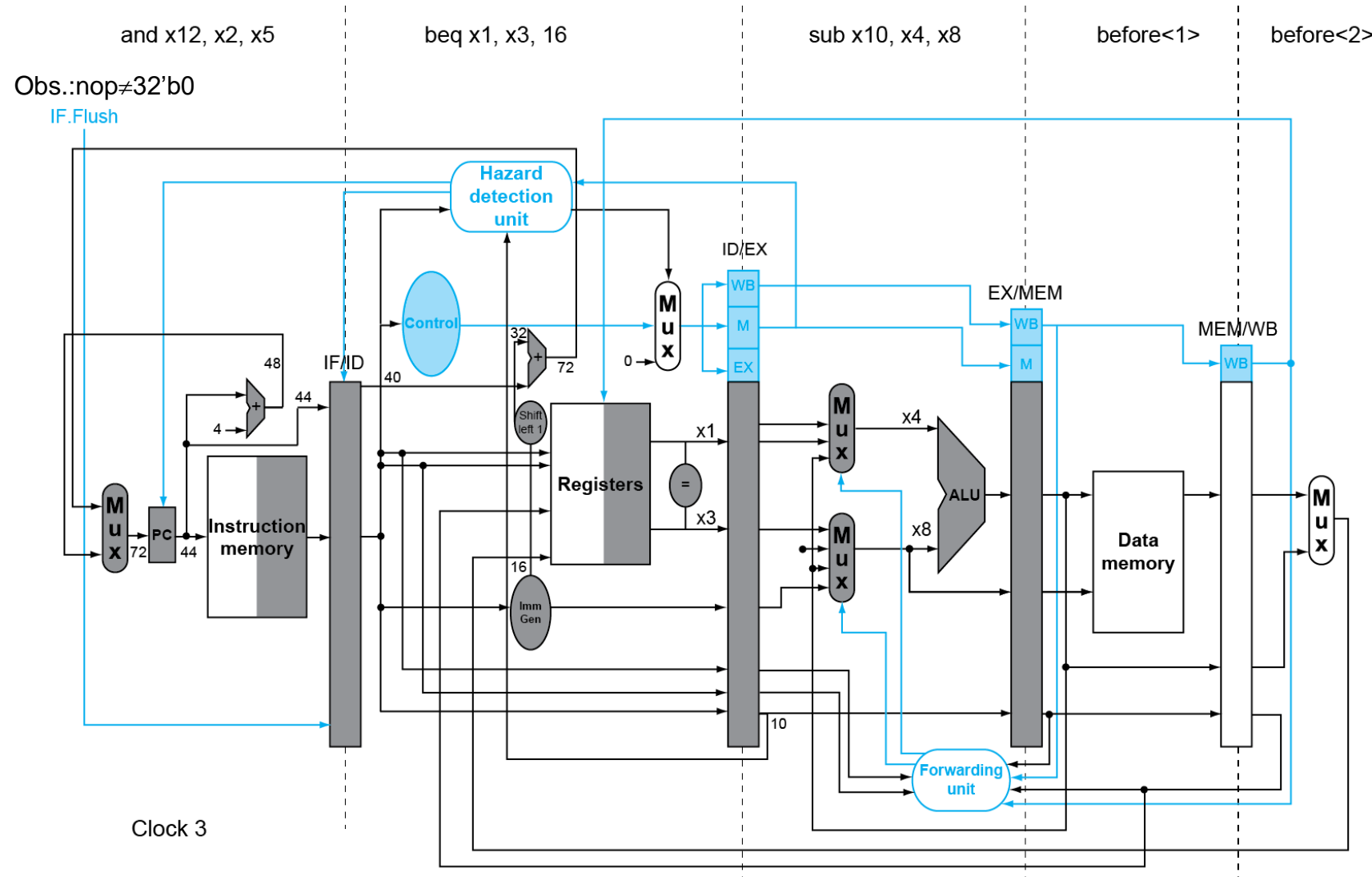
■ Prevendo: Desvio não-tomado





Hazard de Controle

■ Hardware otimizado para 1 bolha

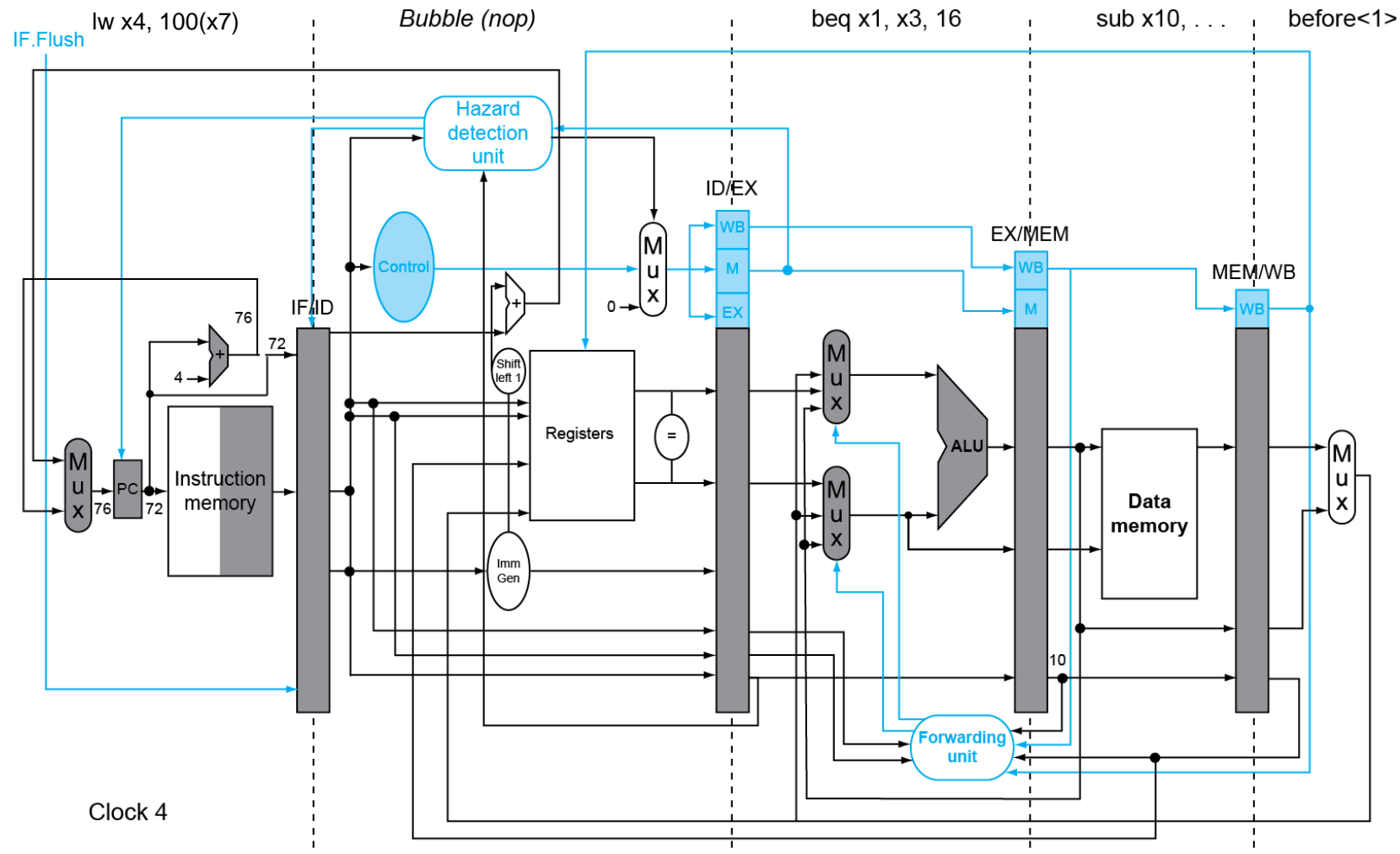


Como ficaria se fosse sub x1,x4,x8 ? Forward beq



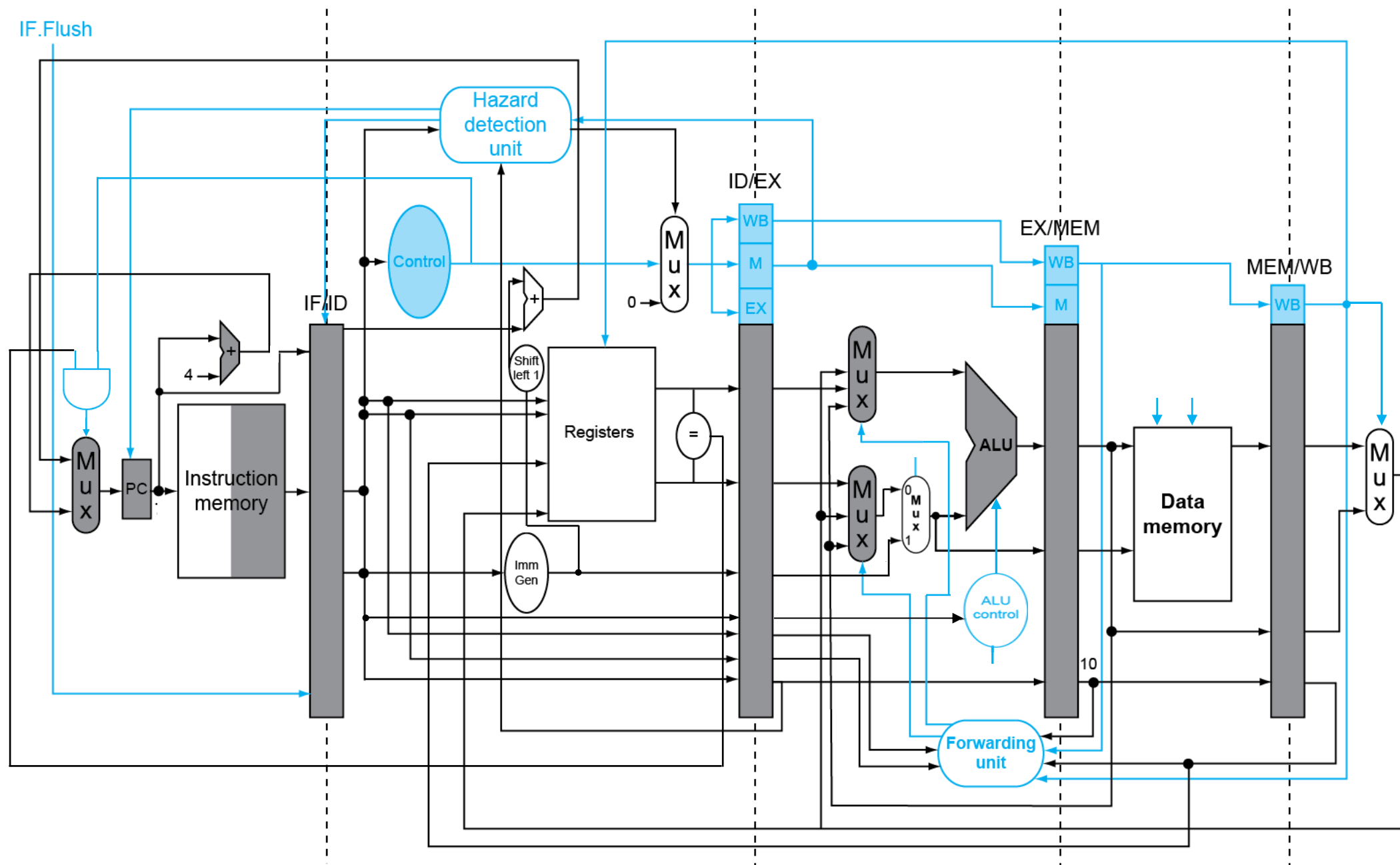
Hazard de Controle

■ Hardware otimizado para 1 bolha



Clock 4

Ver Ripes





Comparação: Uniciclo x Multiciclo x Pipeline

■ Suponha os seguintes tempos das unidades operativas:

- Acesso a memória: 200ps
- Operação da ULA: 100ps
- Acesso ao banco de registradores: 50ps

■ Dado o workload composto de:

- 25% loads
- 10% stores
- 11% branches
- 2% jumps
- 52% operações com a ULA

Poderia considerar o
Setup Time dos registradores: PC,
do Multiciclo e do Pipeline

Compare o desempenho, considerando para o pipeline

- 50% dos loads é seguido de uma operação que requer o argumento
- 25% dos desvios são previstos erradamente e que o atraso da previsão errada é 1 ciclo de clock
- jumps usam 2 ciclos de clock
- ignore outros hazards.



Solução

Uniciclo:

período de clock: $200+50+100+200+50 = 600\text{ps}$

$$\text{CPI}_{\text{média}} = 1$$

Tempo de execução médio da instrução: $600 \times 1 = 600\text{ps}$

Multiciclo:

período de clock: 200ps

$$\text{CPI}_{\text{média}} = 0.25 \times 5 + 0.1 \times 4 + 0.11 \times 3 + 0.02 \times 3 + 0.52 \times 4 = 4.12$$

Tempo de execução médio da instrução: $200 \times 4.12 = 824\text{ps}$

Pipeline:

período de clock: 200ps

$$\text{CPI}_{\text{média}} = 0.25 \times 1.5 + 0.1 \times 1 + 0.11 \times 1.25 + 0.02 \times 2 + 0.52 \times 1 = 1.17$$

Tempo de execução médio da instrução: $200 \times 1.17 = 234\text{ps}$



Conclusão

- Todos processadores modernos possuem pipeline complexos:
 - 80486 : 5 estágios (CISC , puro escalar)
 - Pentium: 5 estágios + 2 ULAs inteiros
 - Pentium II: 10 estágios + super-escalar (núcleo RISC, **nível de micro instruções**)
 - Pentium III : 10 estágios
 - Pentium IV: 20 estágios
 - Pentium D: 31 estágios (Maior frequência, maior num. bolhas, maior consumo)
 - Core2 : 14 estágios (porém dual/quad core real)
 - Core i3,i5,i7,i9 : 16 estágios
- PowerPC e Pentium: tabela de histórico de desvios (buffer de previsão)
- Escalonamento de instruções: Que instrução executar agora?
 - Historicamente: tarefa do compilador
 - Hoje em dia: o hardware do processador já realiza.

Obs: Meltdown e Spectre falhas de segurança (2017/2018)