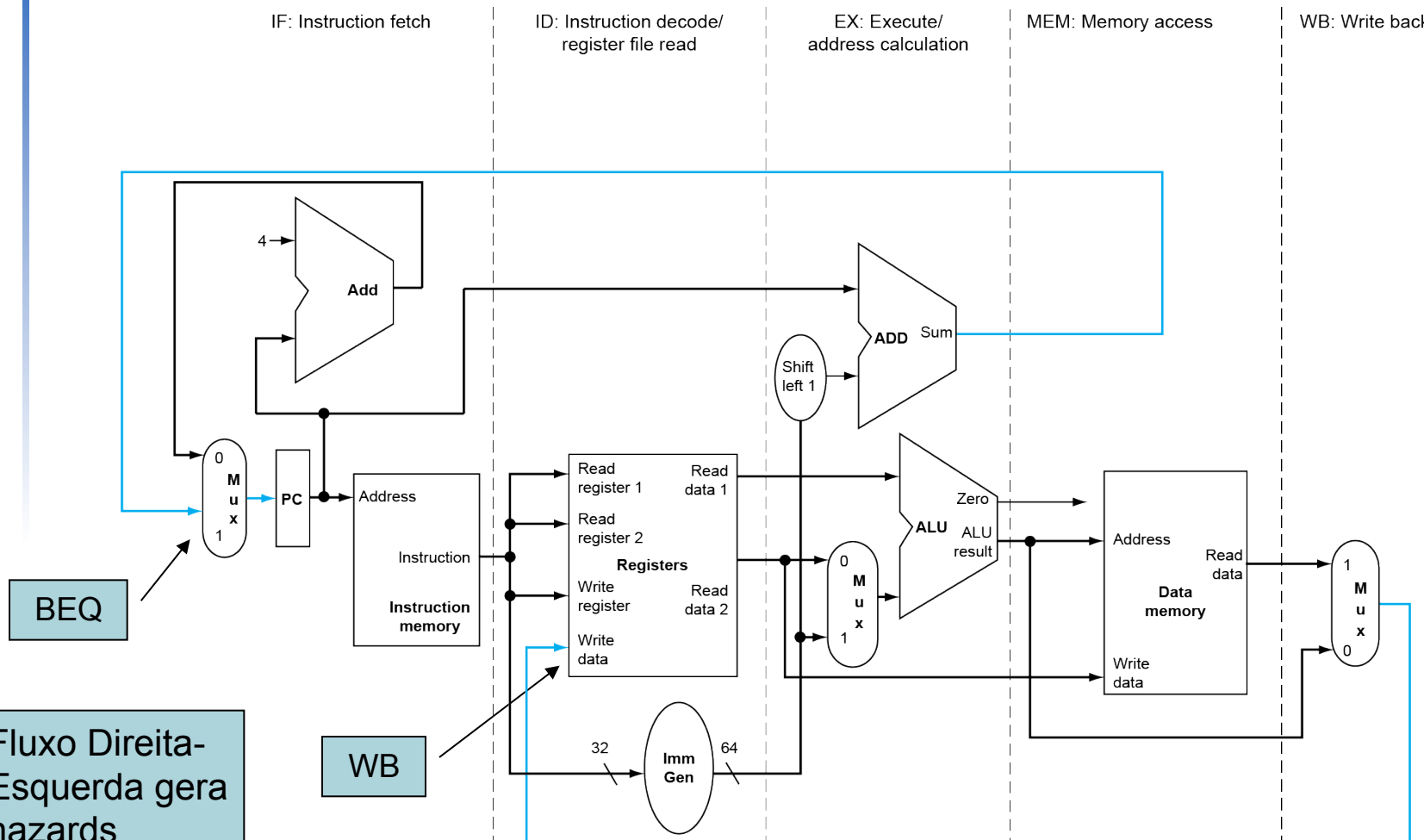
Organização de Computadores I
DCC006

Aula 9 – O Processador – Pipeline

Prof. Omar Paranaiba Vilela Neto

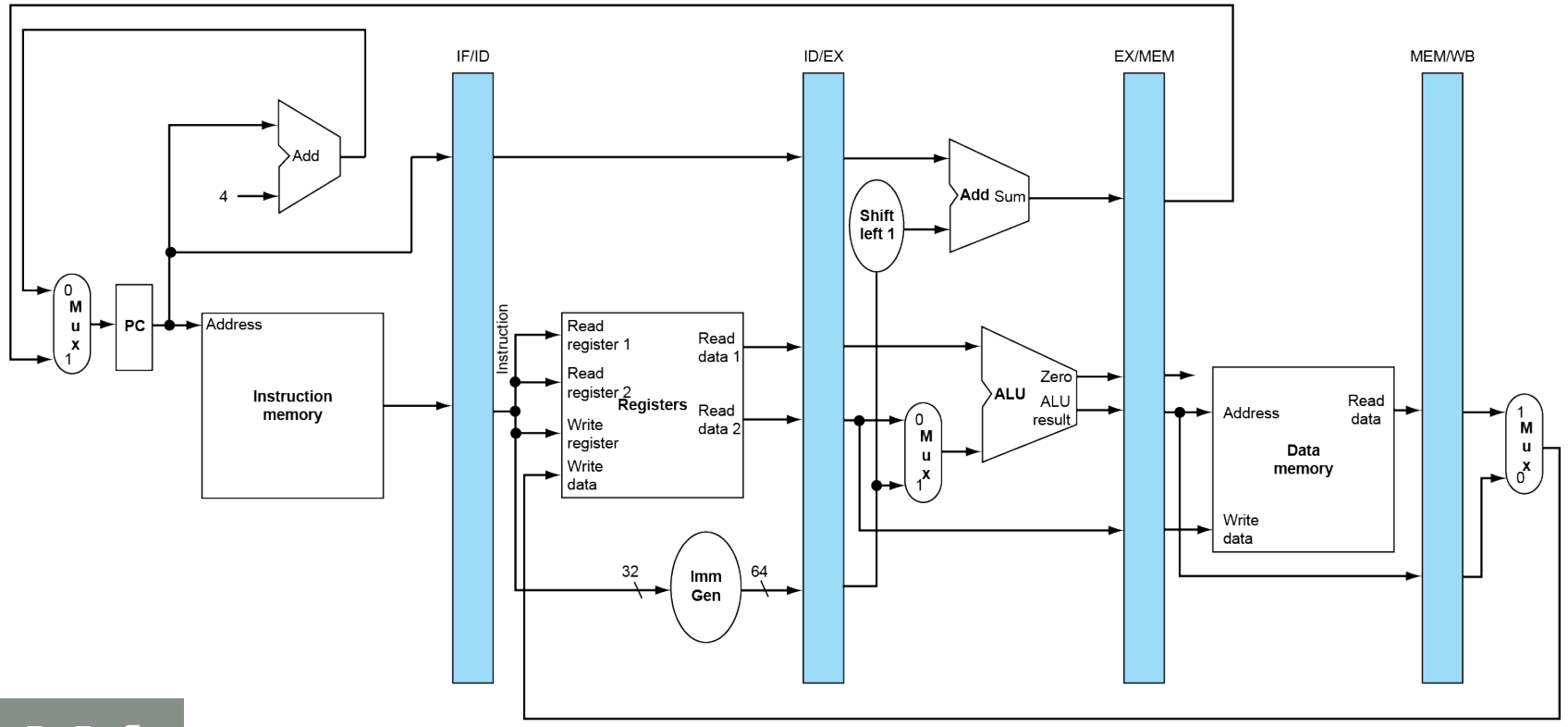
UFMG - ICEx
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

RISC-V Pipelined Datapath



Pipeline registradores

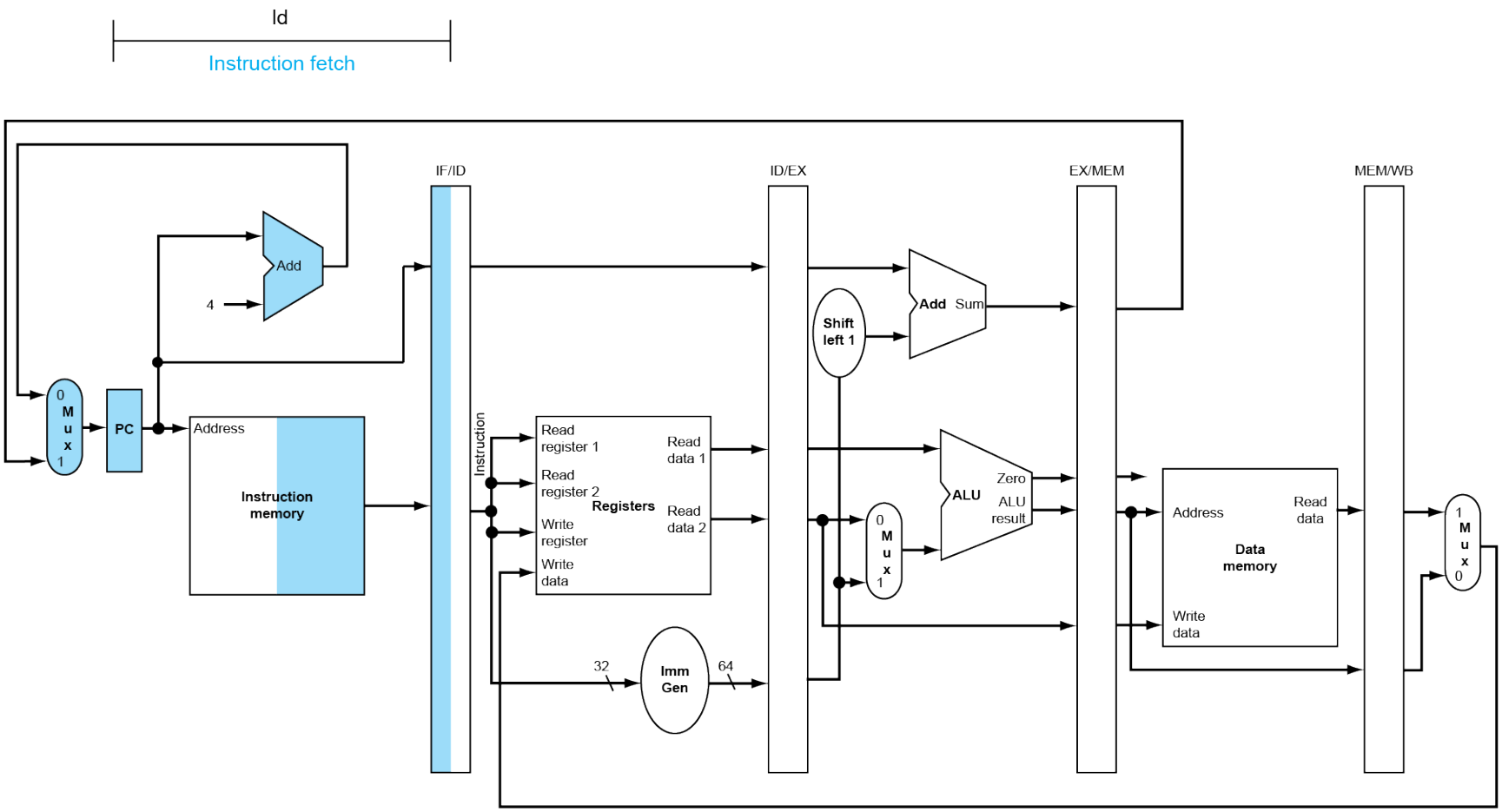
- Precisa de registradores entre estágios
  - Armazenar informação produzida no ciclo anterior



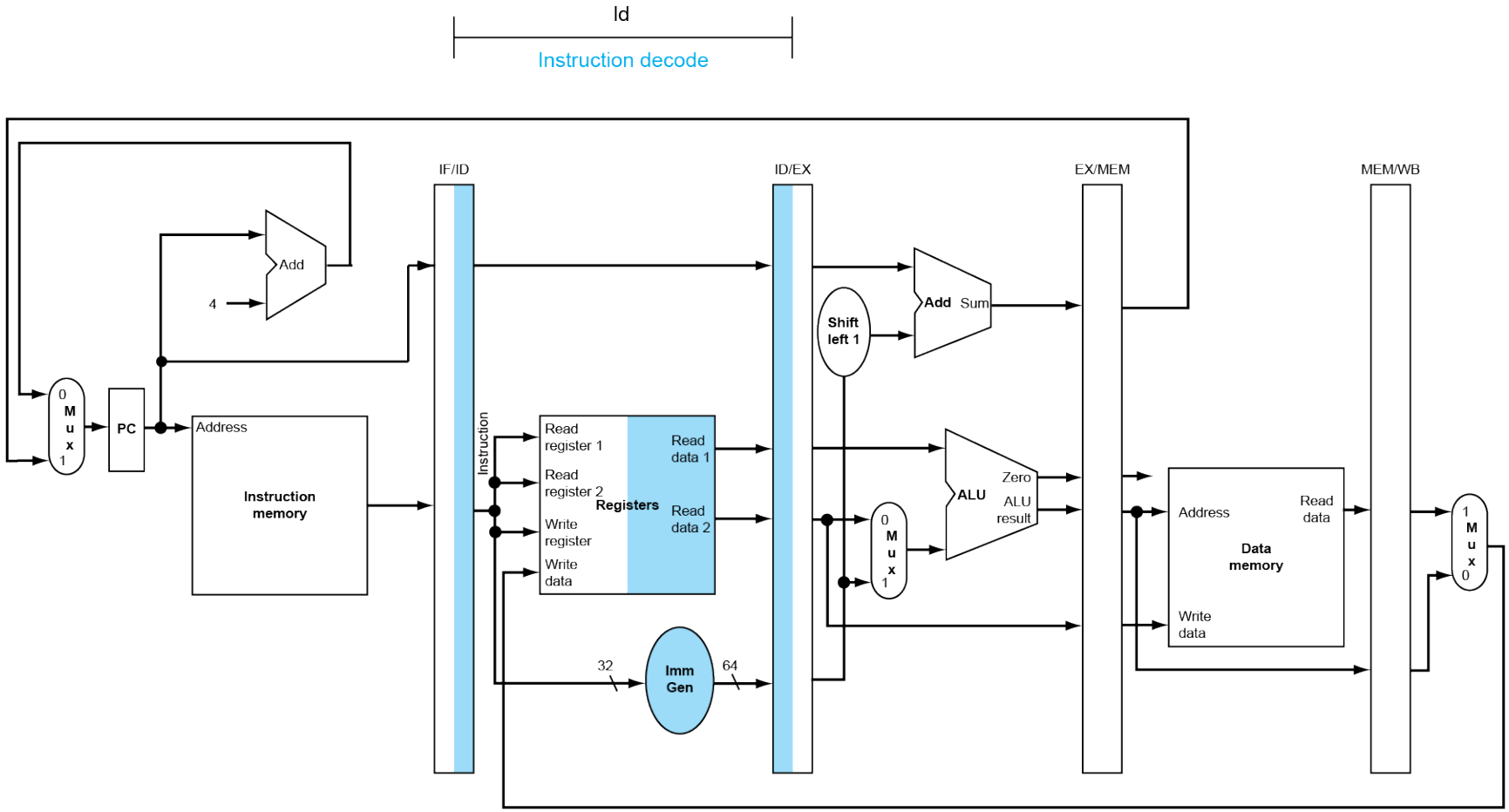
Operação do Pipeline

- Fluxo de instruções através do pipeline Ciclo-por-Ciclo
  - "Simples-ciclo-clock" diagrama de pipeline
    - Mostra o uso do pipeline em um simples ciclo
    - Destaca uso de recurso
  - "Múltiplos-ciclos-clock" diagrama de pipeline
    - Gráfico de operação ao longo do tempo
- Vamos ver o diagrama "Simples-ciclo-clock" para load & store

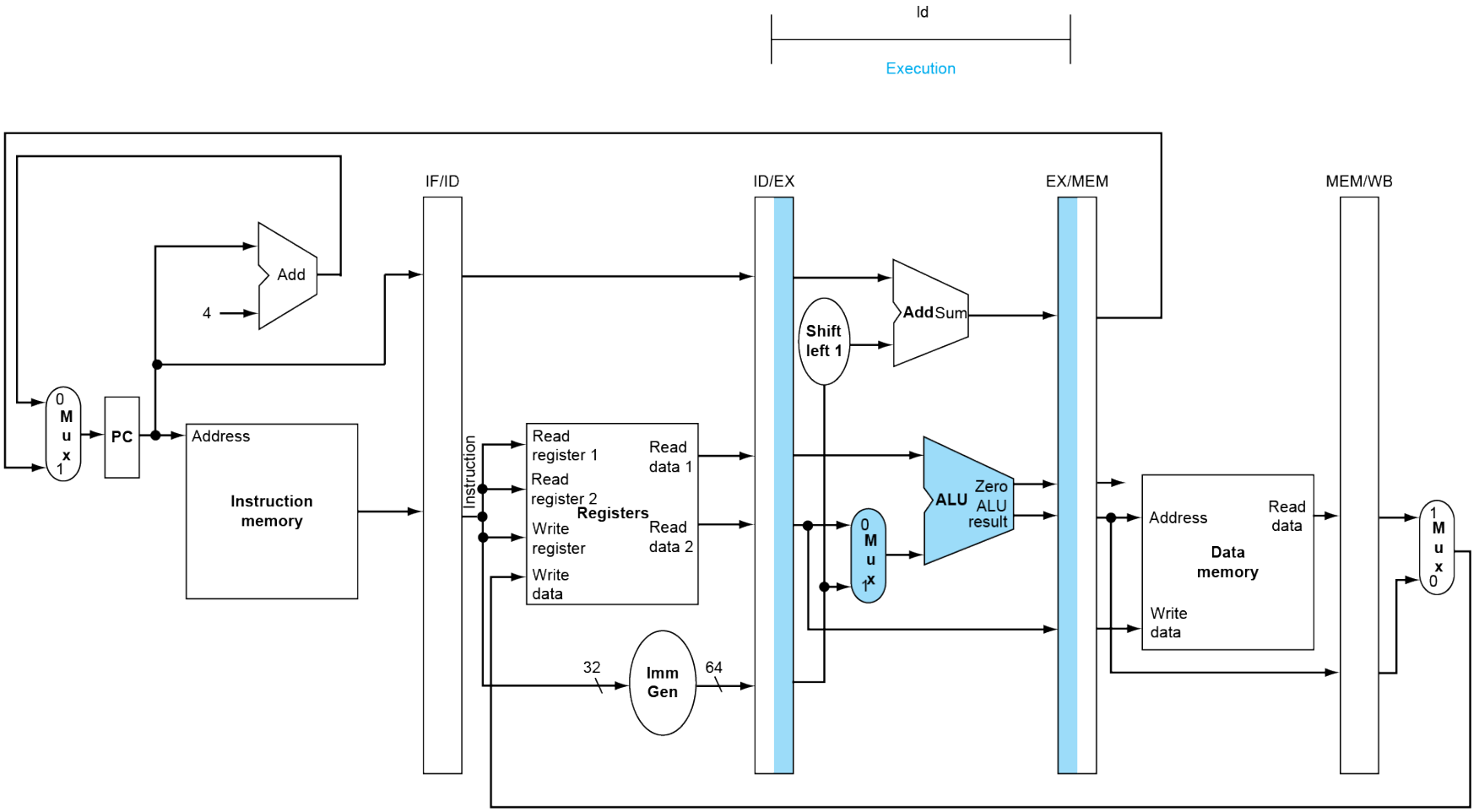
IF para Load, Store, …



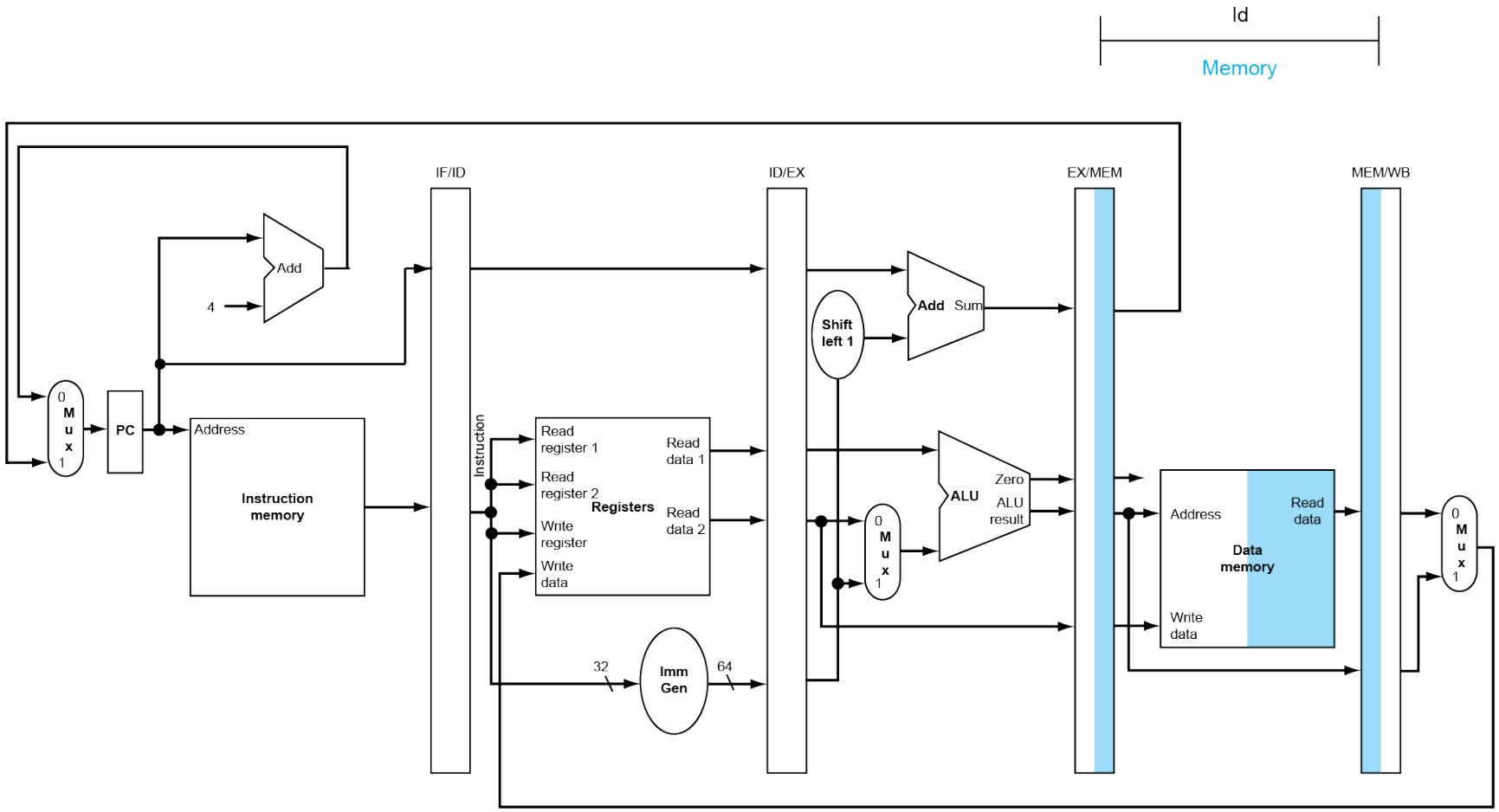
ID para Load, Store, …



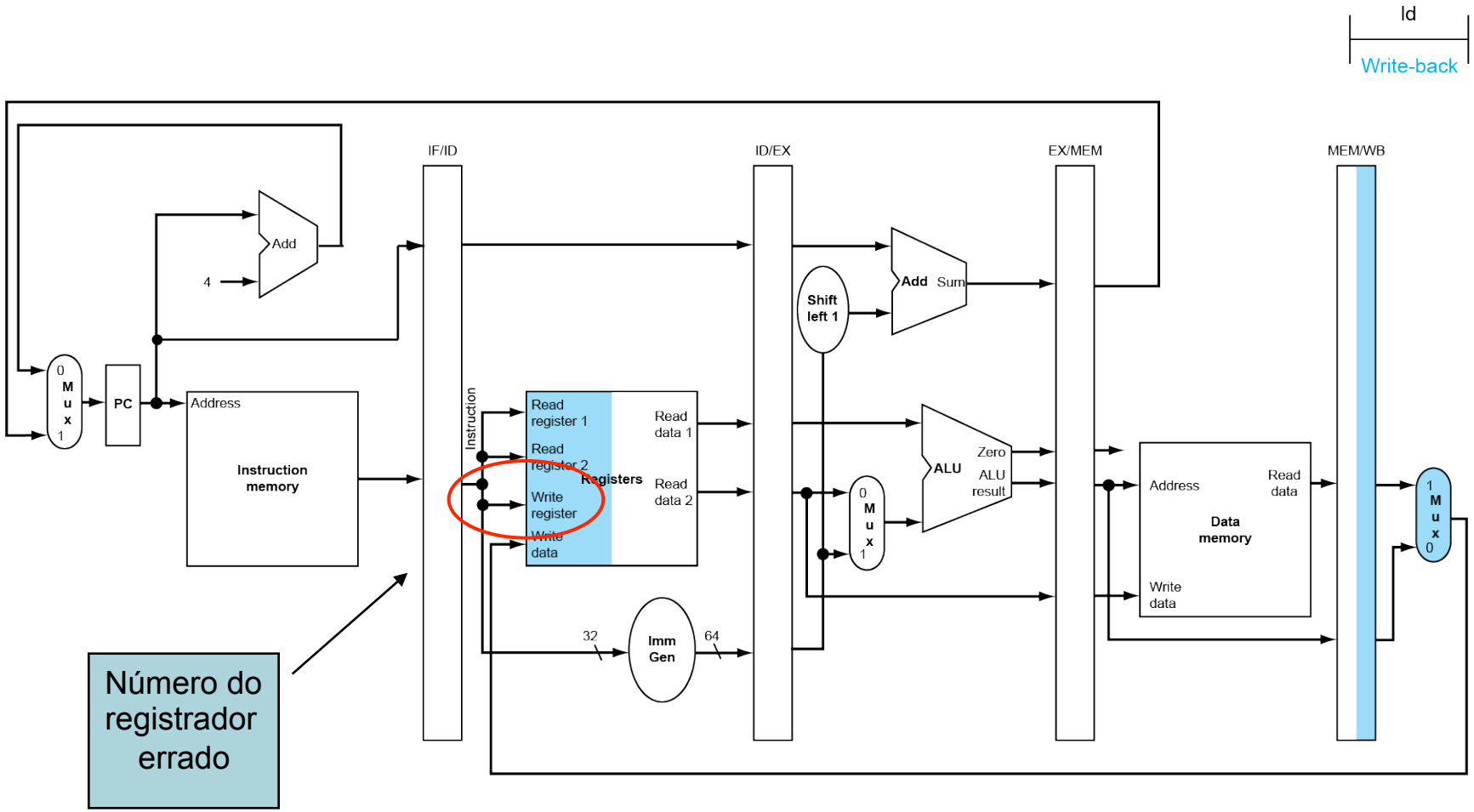
EX para Load


ld
Execution
IF/ID
ID/EX
EX/MEM
MEM/WB
Add
4
0
M
u
x
1
PC
Address
Instruction memory
Instruction
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
32
Imm Gen
64
Shift left 1
Add
Sum
ALU
Zero
ALU result
0
M
u
x
1
Address
Read data
Data memory
Write data
1
M
u
x
0

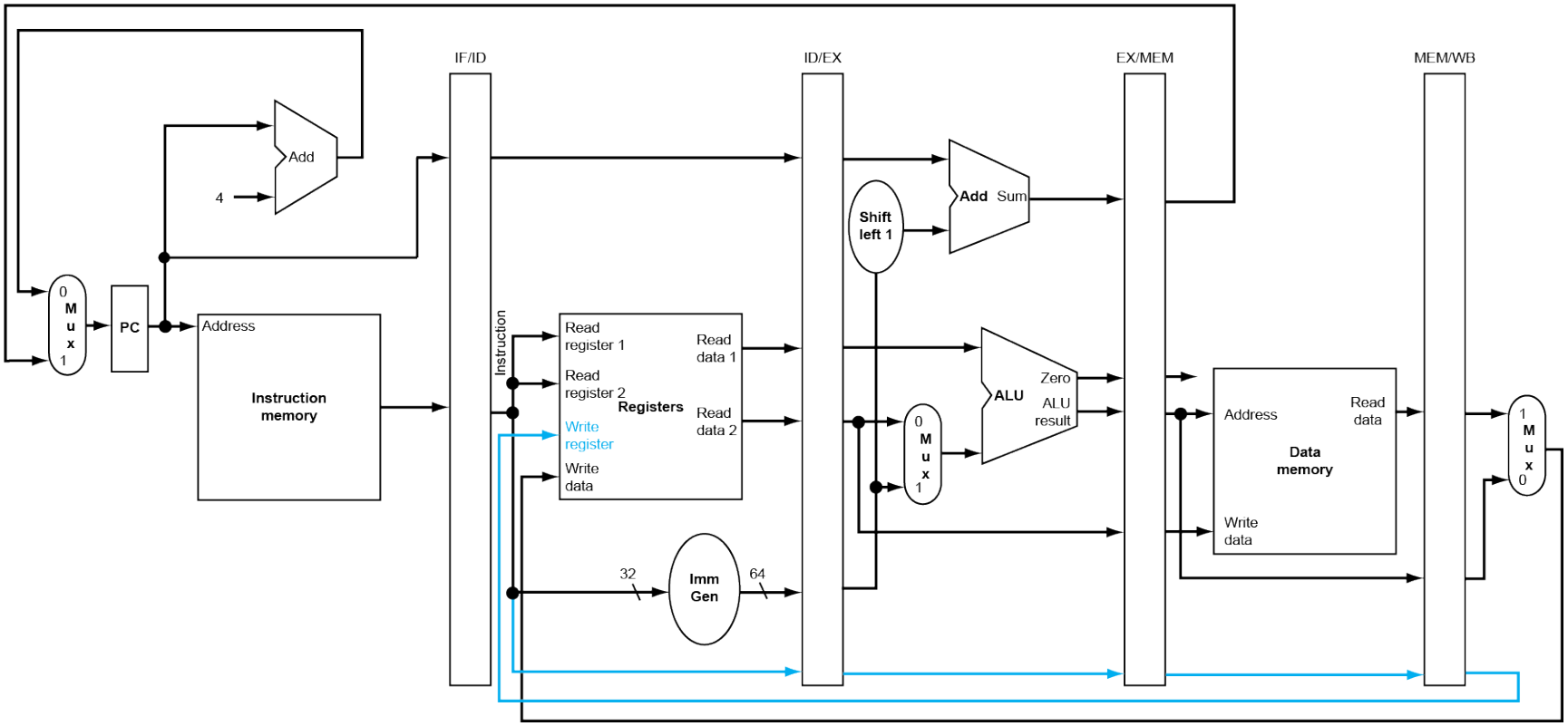
MEM para Load



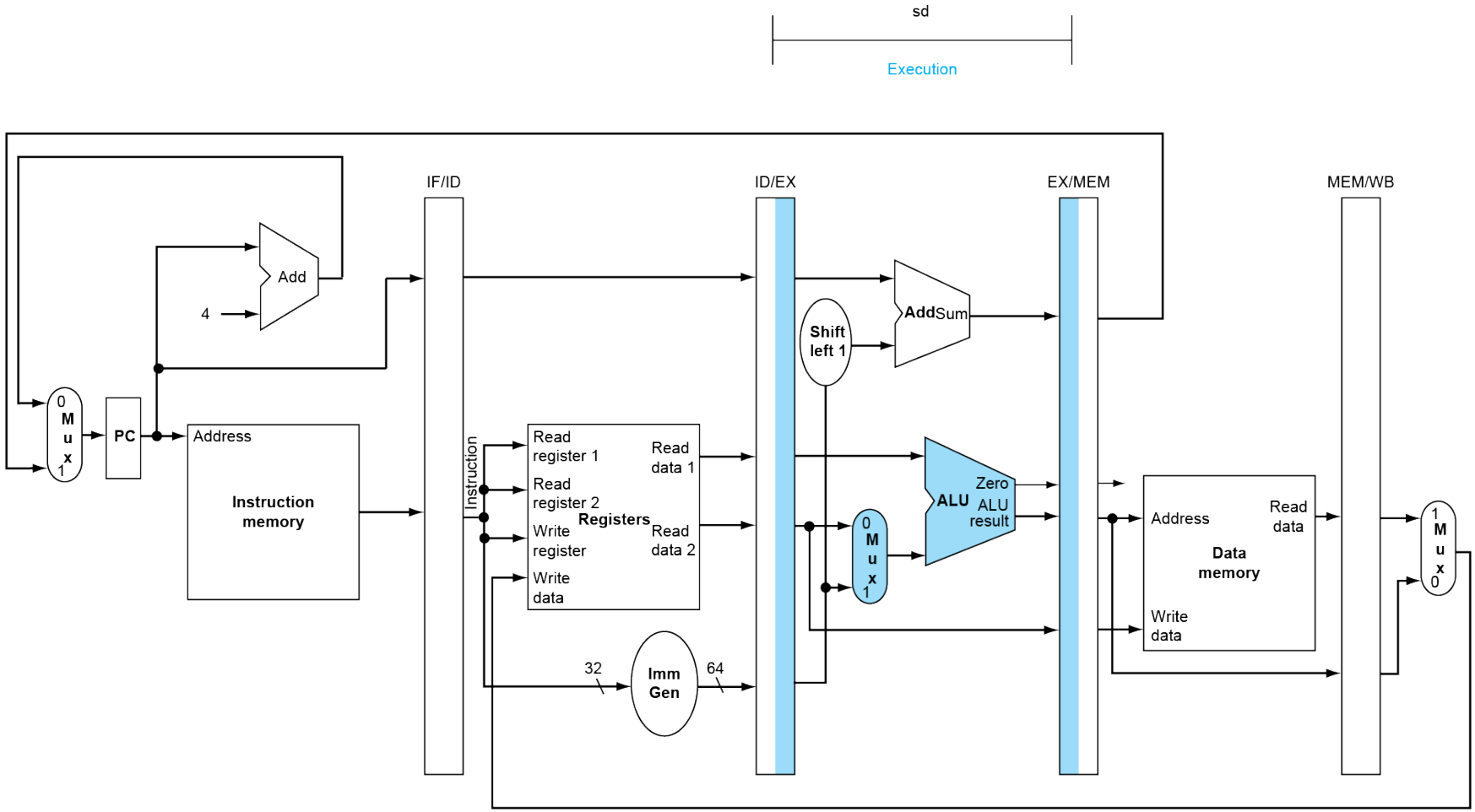
WB para Load


ld
Write-back
IF/ID
ID/EX
EX/MEM
MEM/WB
Add
4
M
u
x
0
1
PC
Address
Instruction memory
Instruction
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
Shift left 1
Add
Sum
ALU
Zero
ALU result
32
Imm Gen
64
Address
Read data
Data memory
Write data
Número do registrador errado

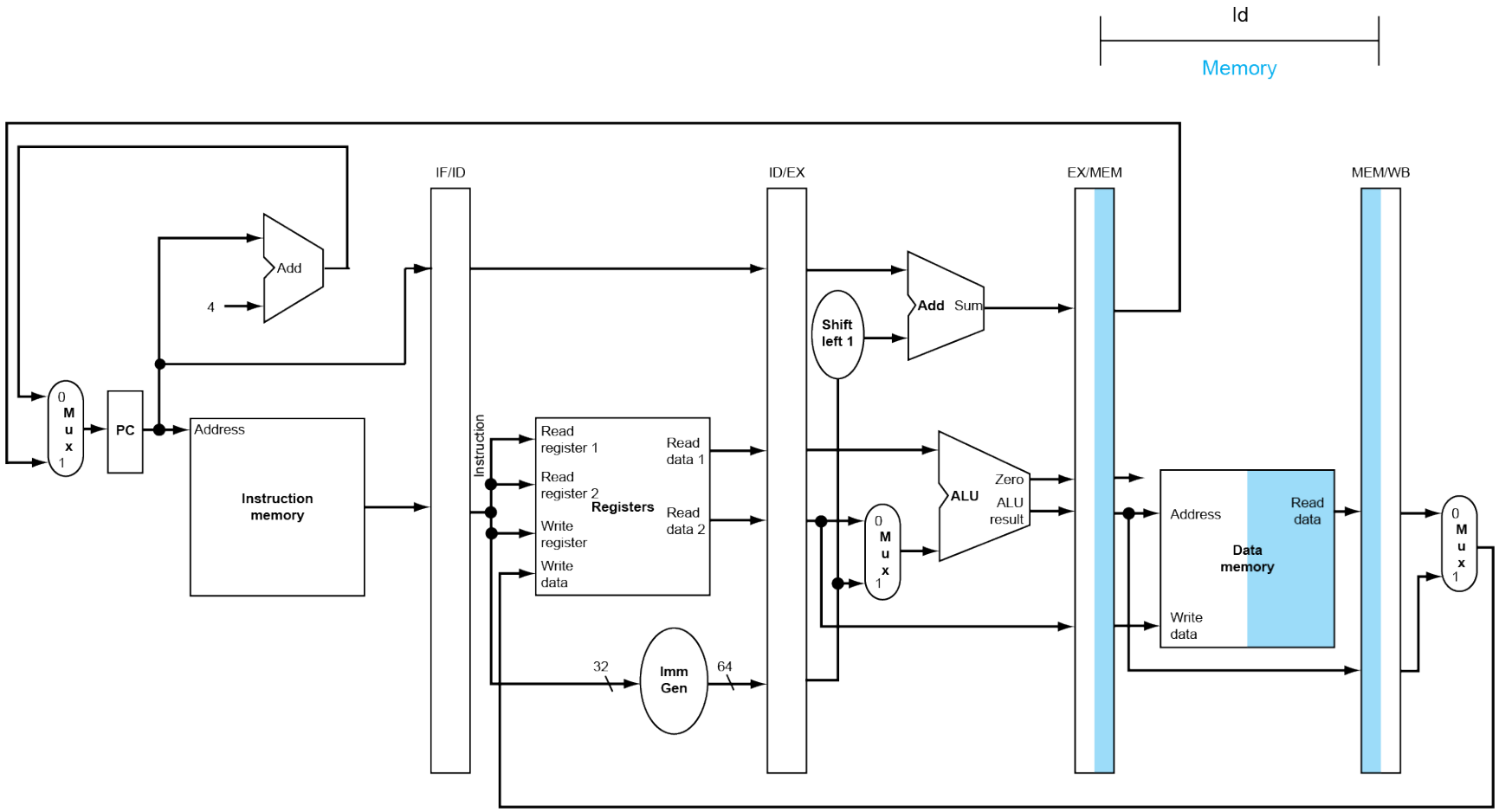
Datapath para Load



EX para Store



MEM para Store



WB para Store

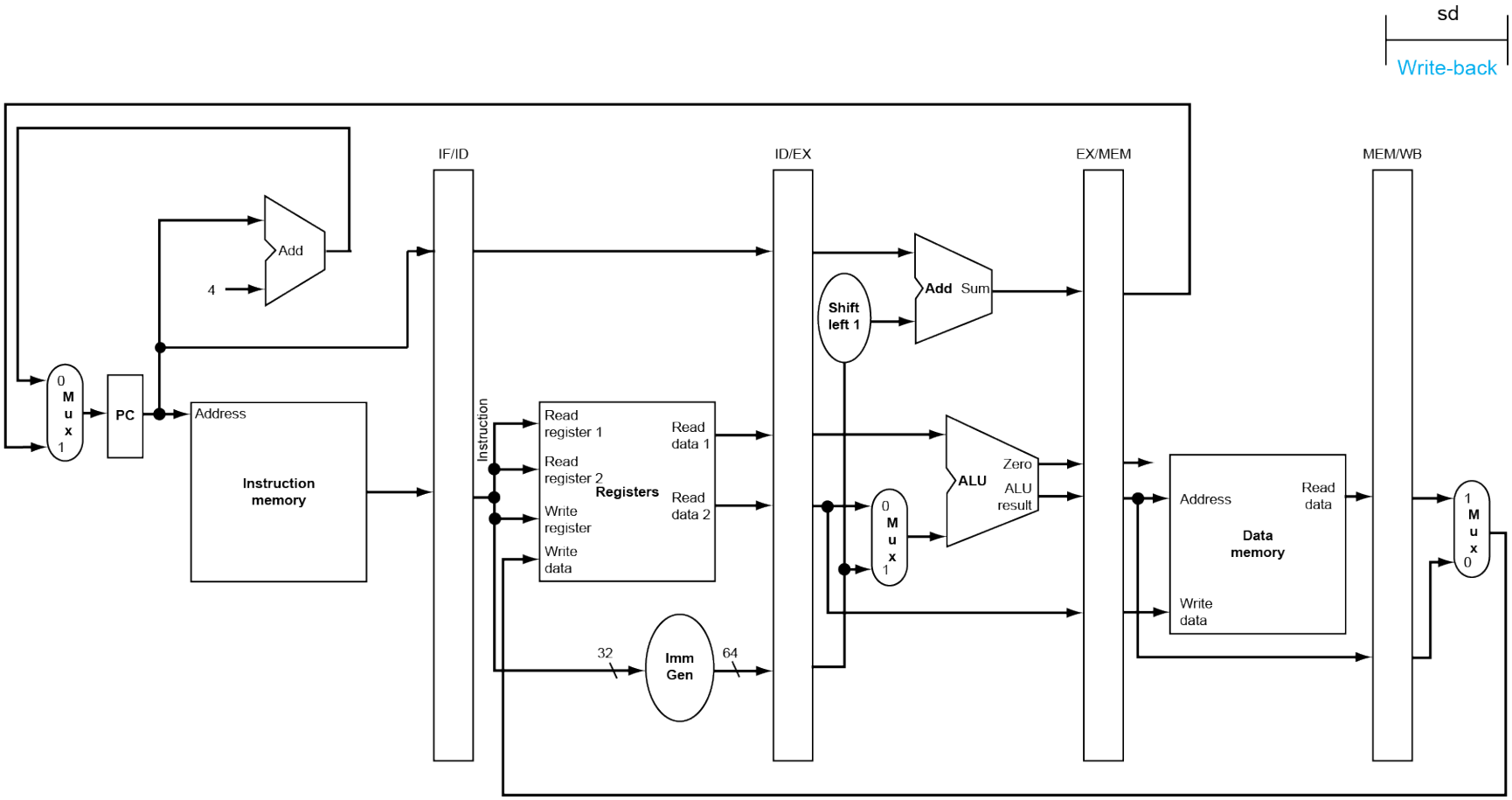


Diagrama Pipeline Múltiplos-Ciclos

- Mostrando o uso de recursos

Diagrama Pipeline Múltiplos-Ciclos

- Mostrando o uso de recursos

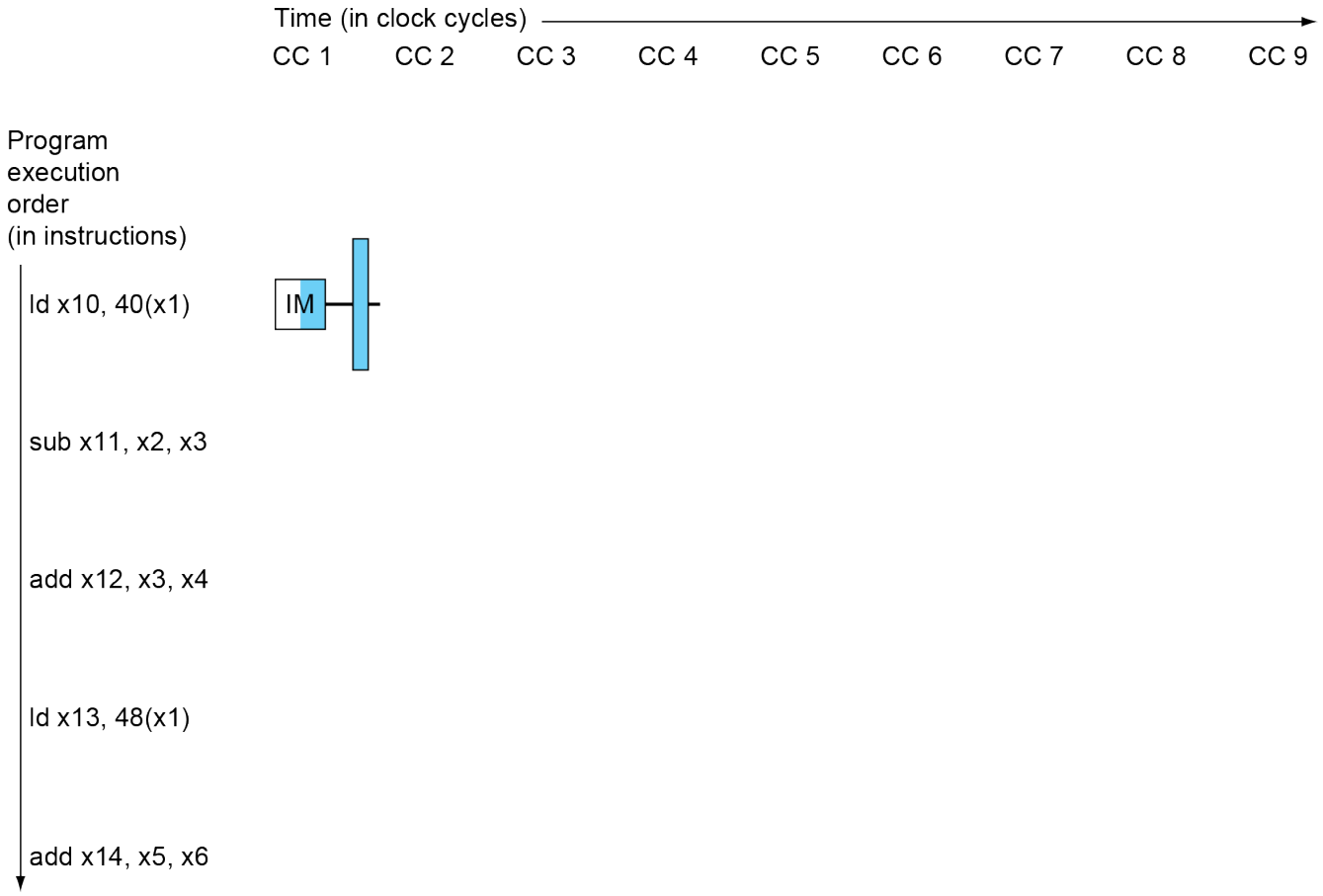


Diagrama Pipeline Múltiplos-Ciclos

- Mostrando o uso de recursos

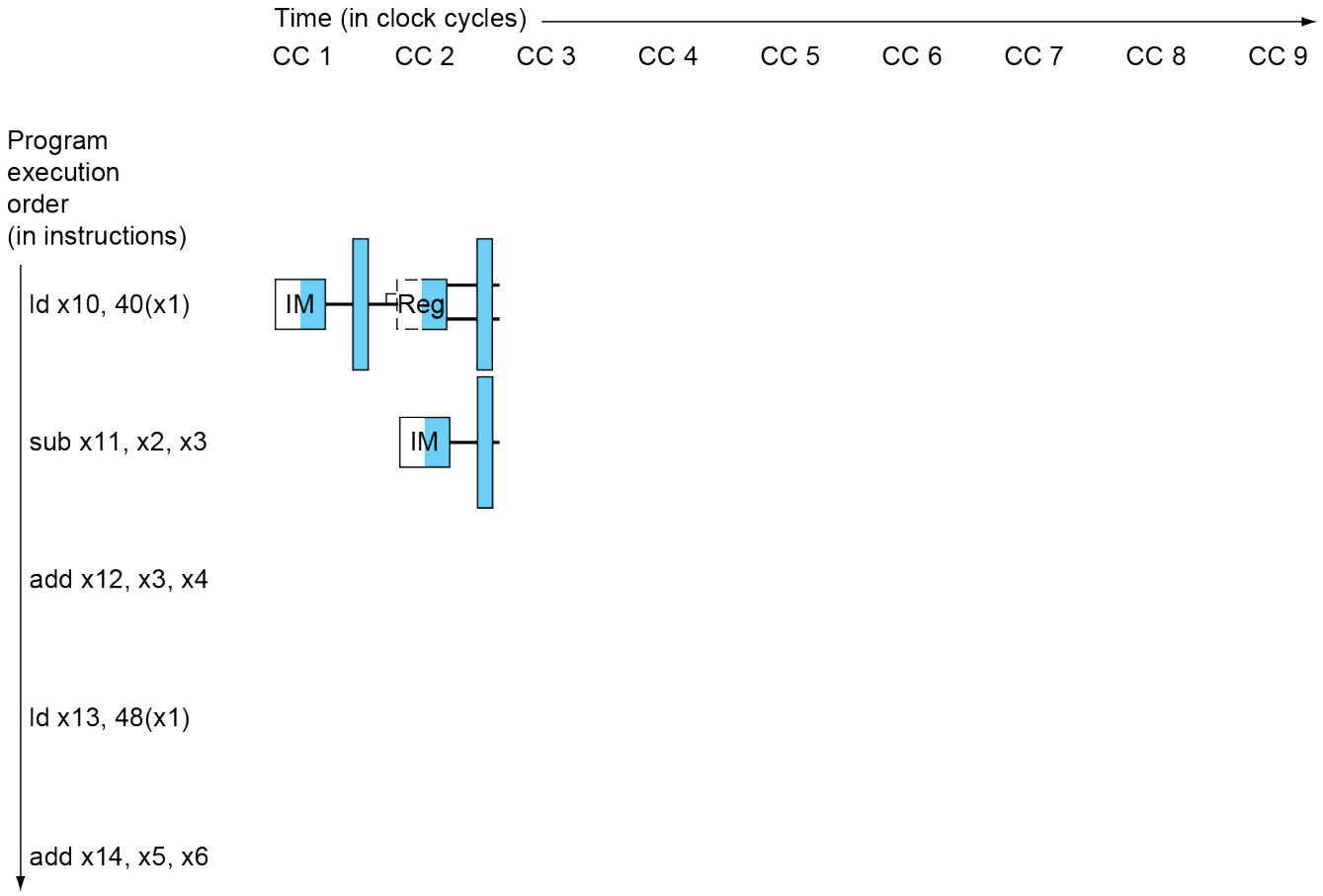


Diagrama Pipeline Múltiplos-Ciclos

- Mostrando o uso de recursos

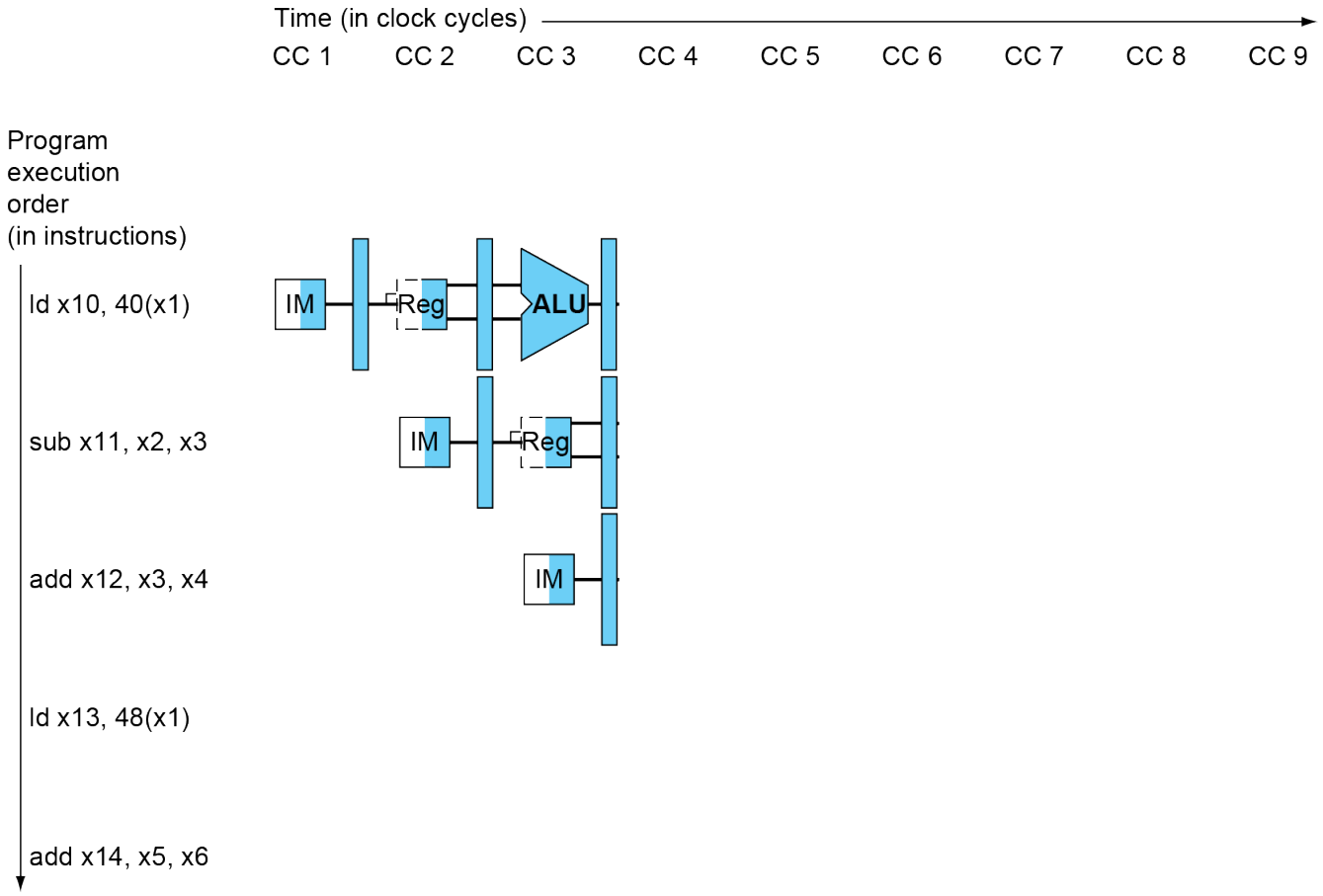


Diagrama Pipeline Múltiplos-Ciclos

- Mostrando o uso de recursos

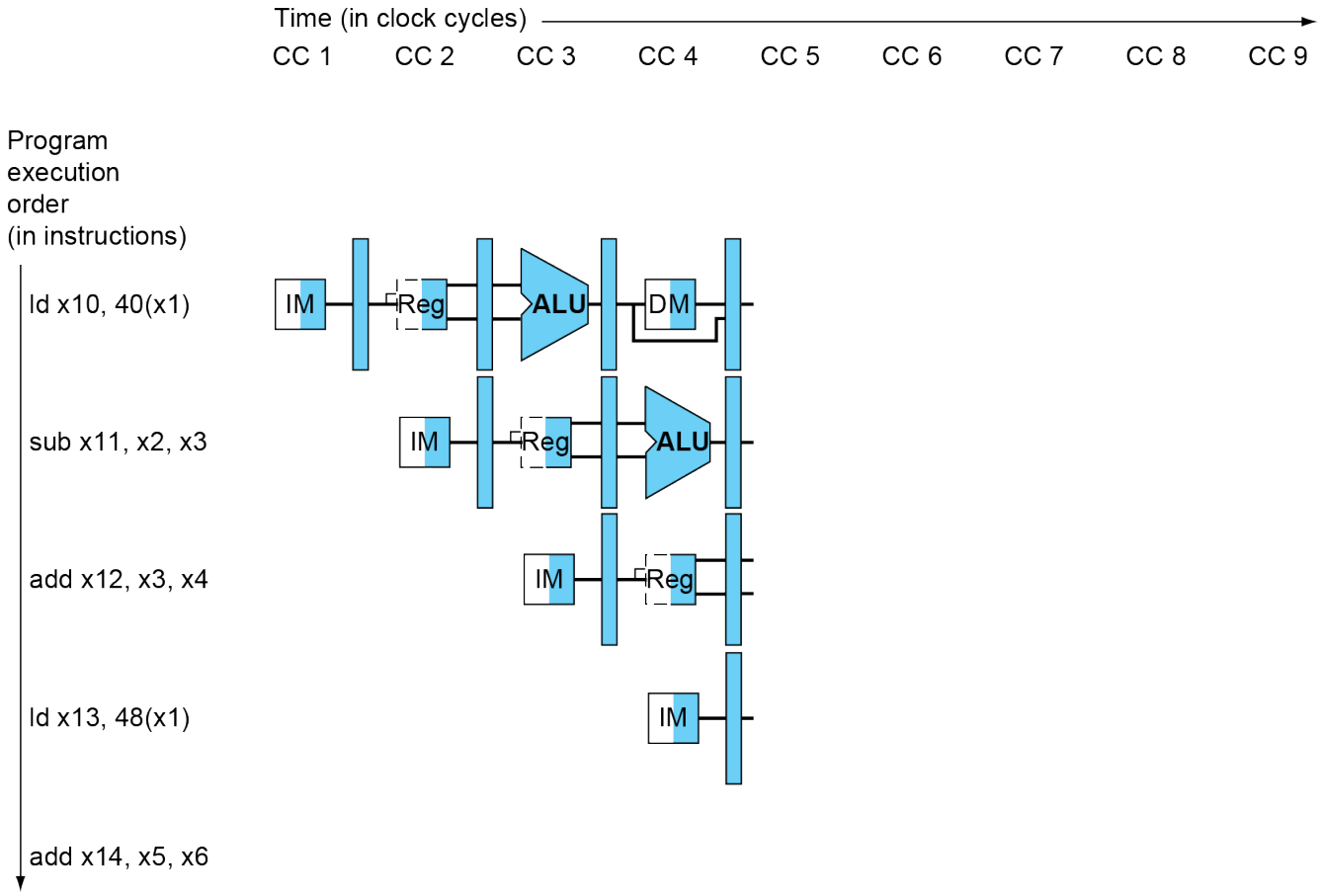


Diagrama Pipeline Múltiplos-Ciclos

- Mostrando o uso de recursos

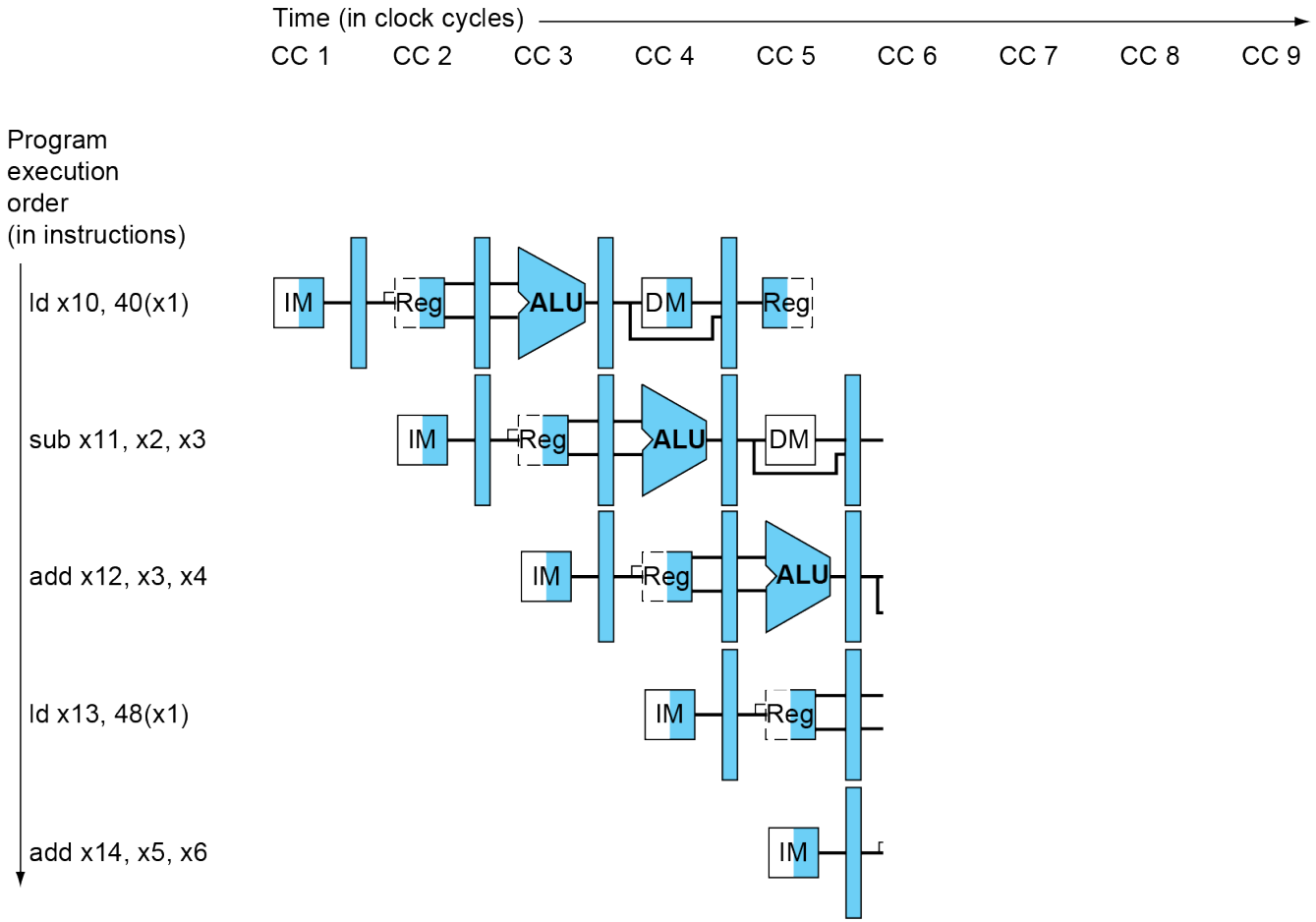


Diagrama Pipeline Múltiplos-Ciclos

- Mostrando o uso de recursos

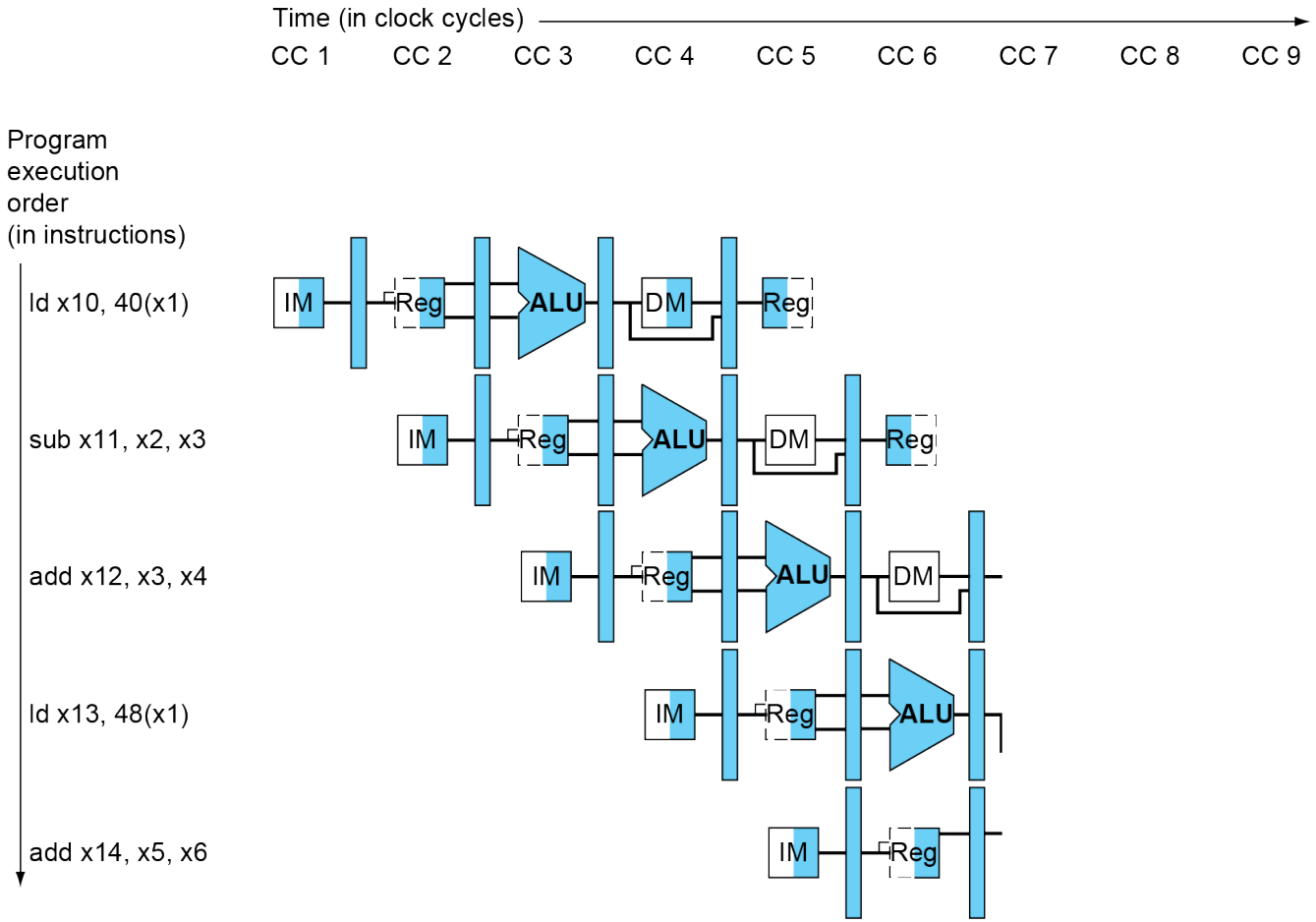


Diagrama Pipeline Múltiplos-Ciclos

- Mostrando o uso de recursos

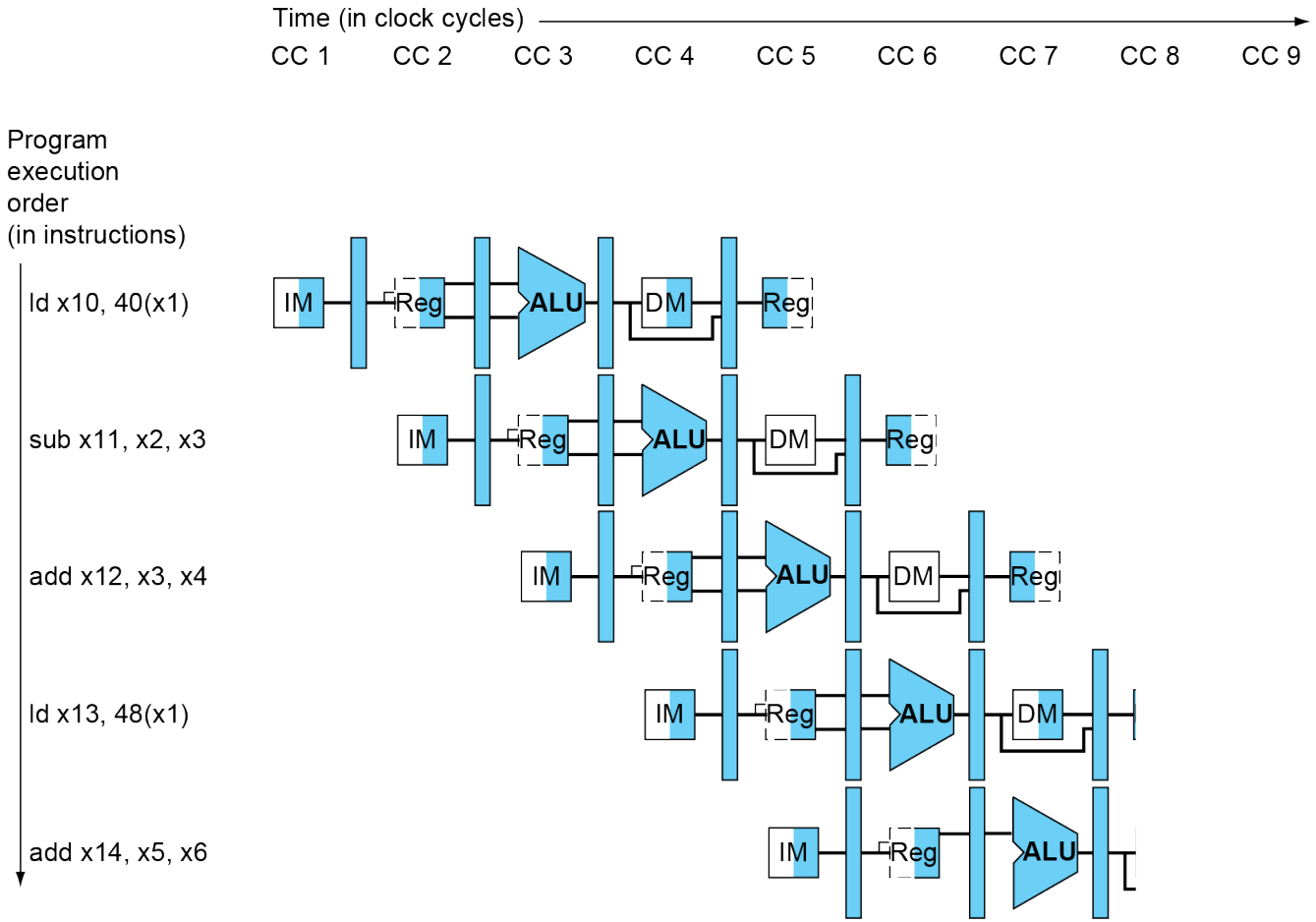


Diagrama Pipeline Múltiplos-Ciclos

- Mostrando o uso de recursos

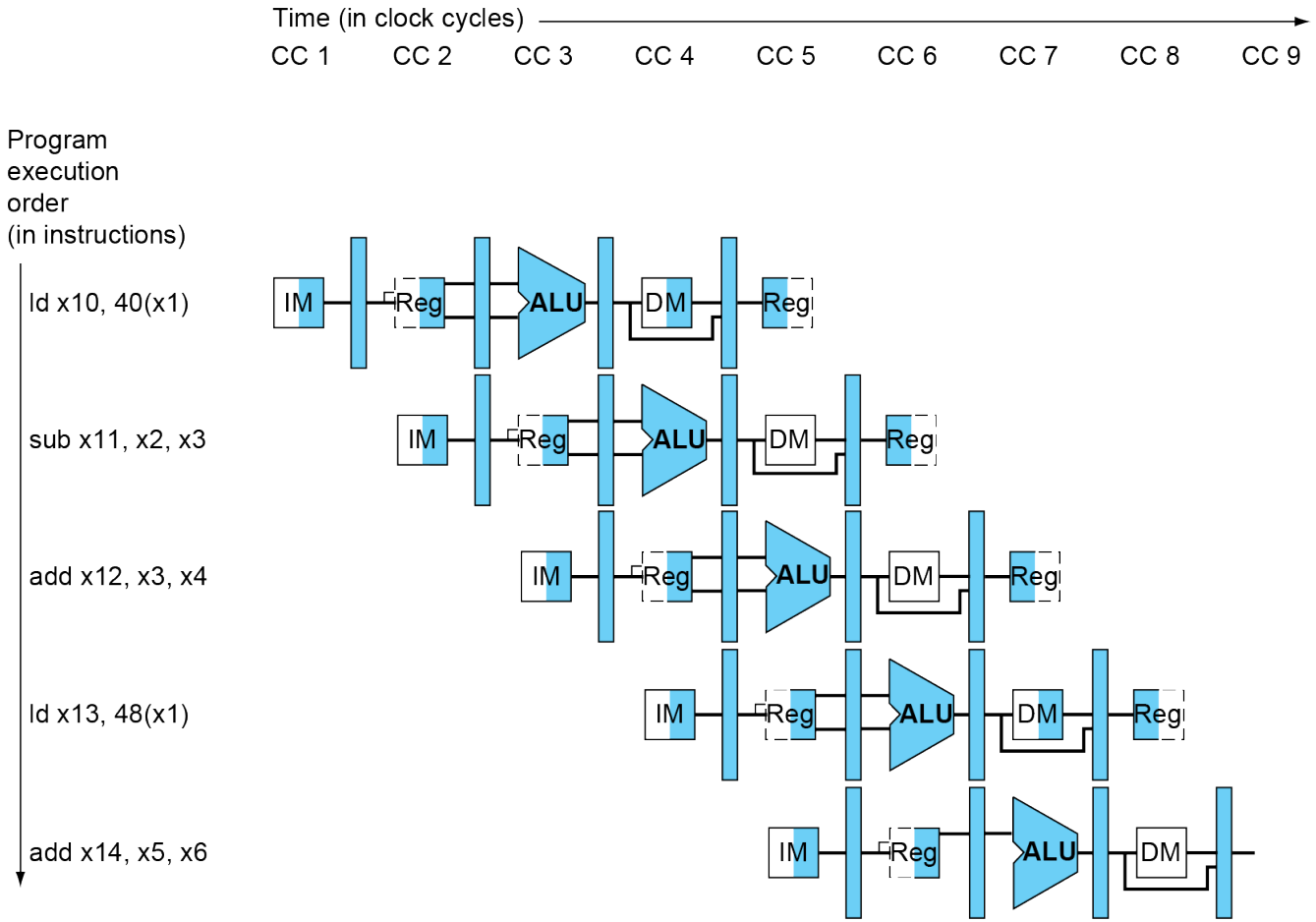


Diagrama Pipeline Múltiplos-Ciclos

- Mostrando o uso de recursos

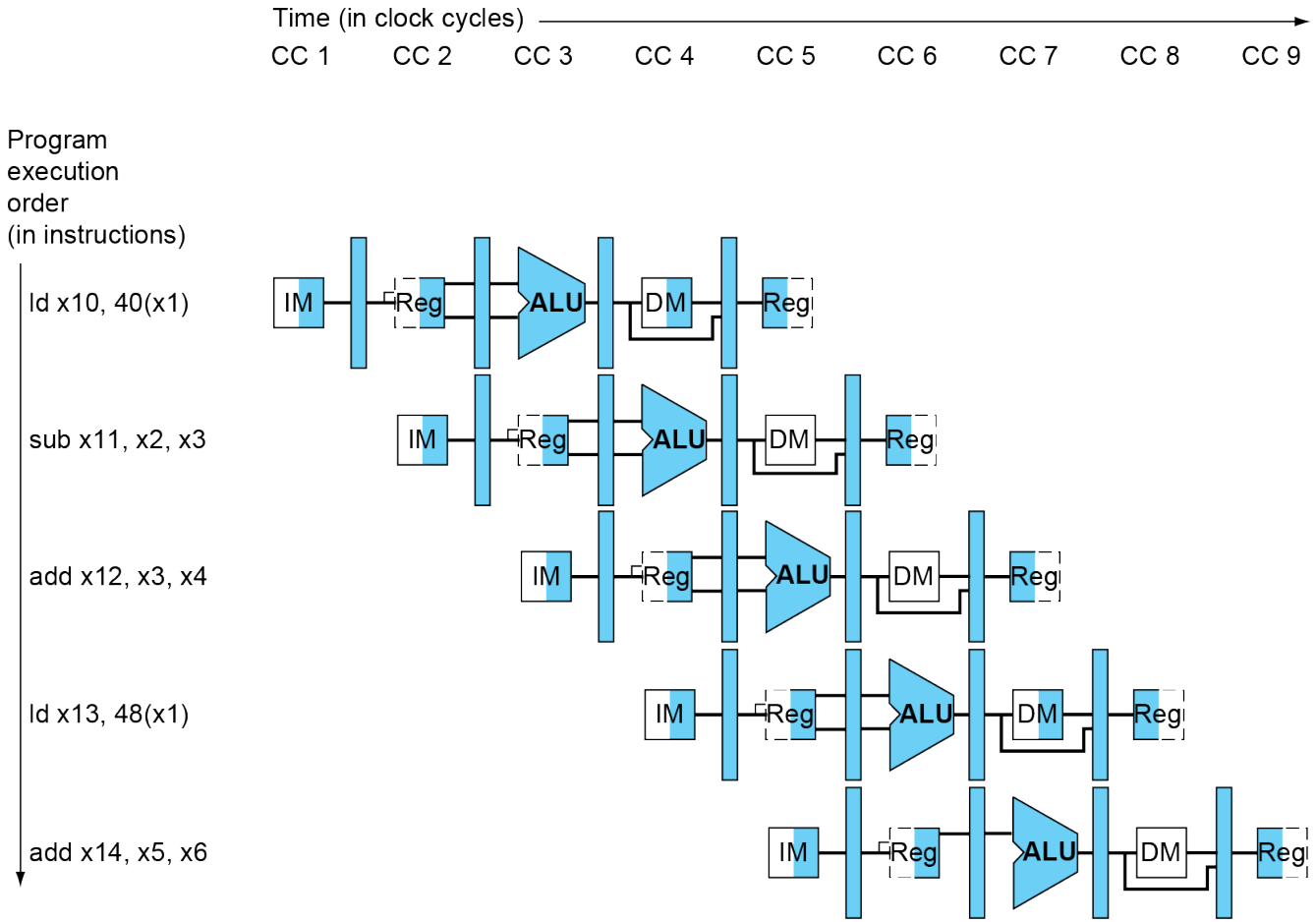


Diagrama Pipeline Múltiplos-Ciclos

- Forma tradicional

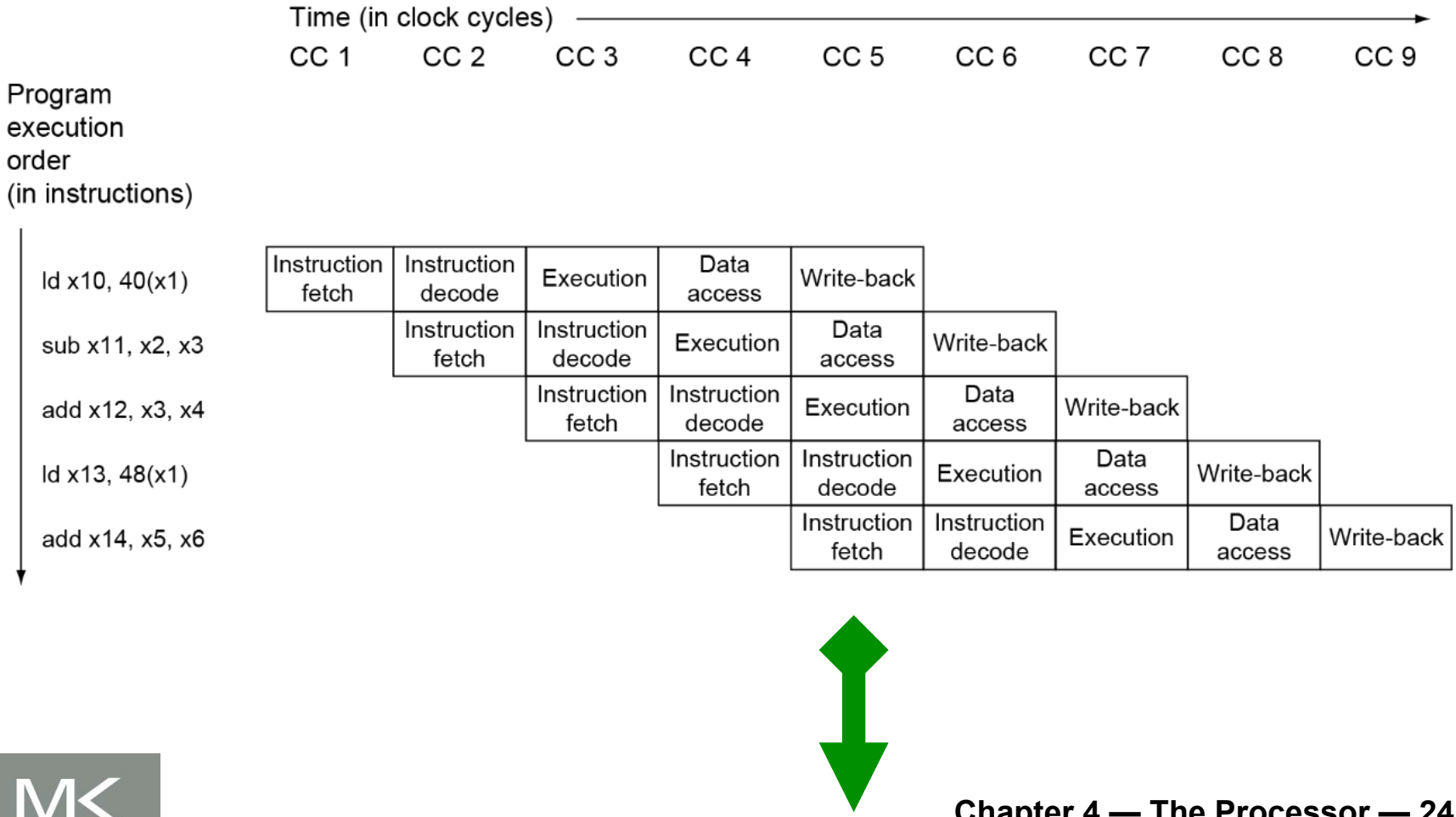
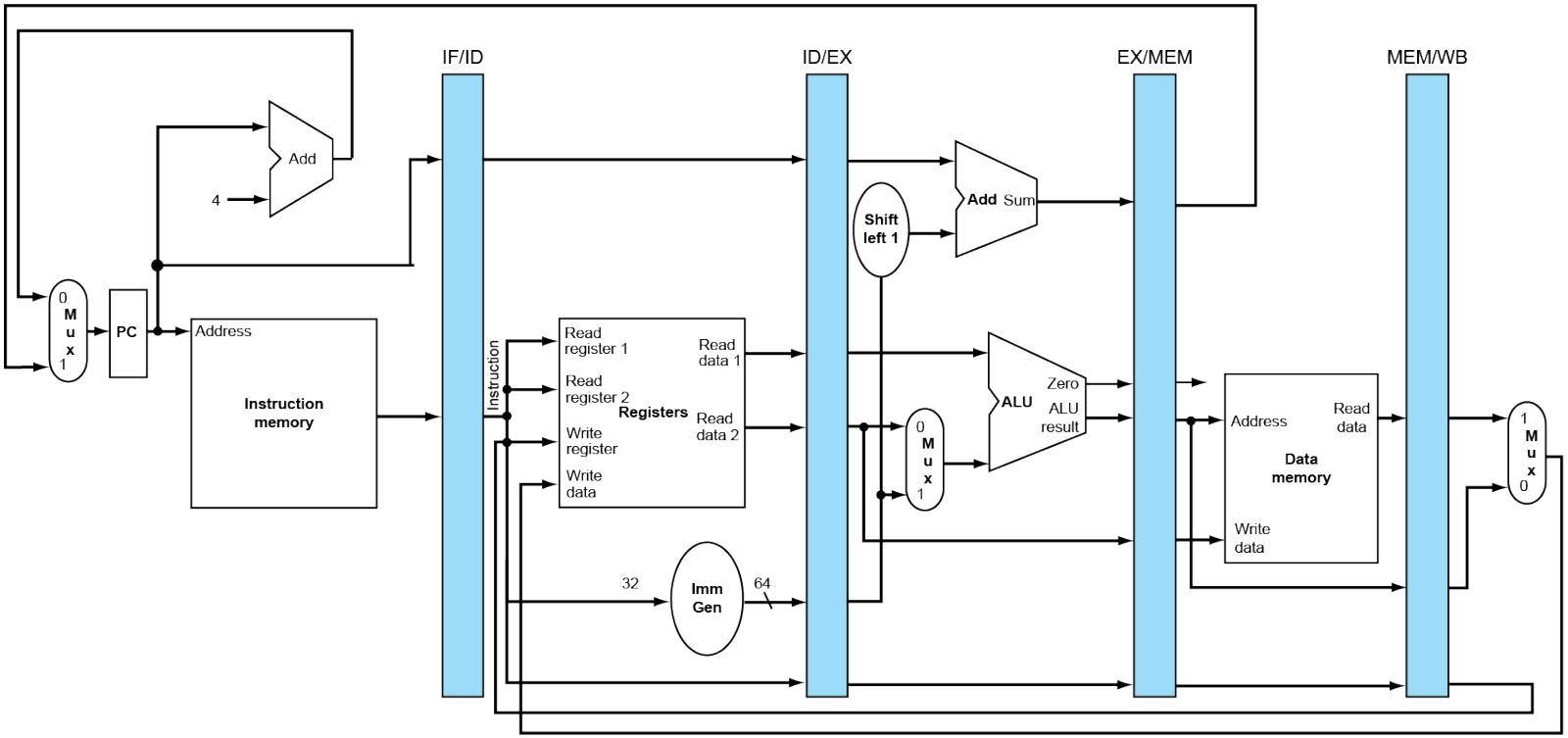


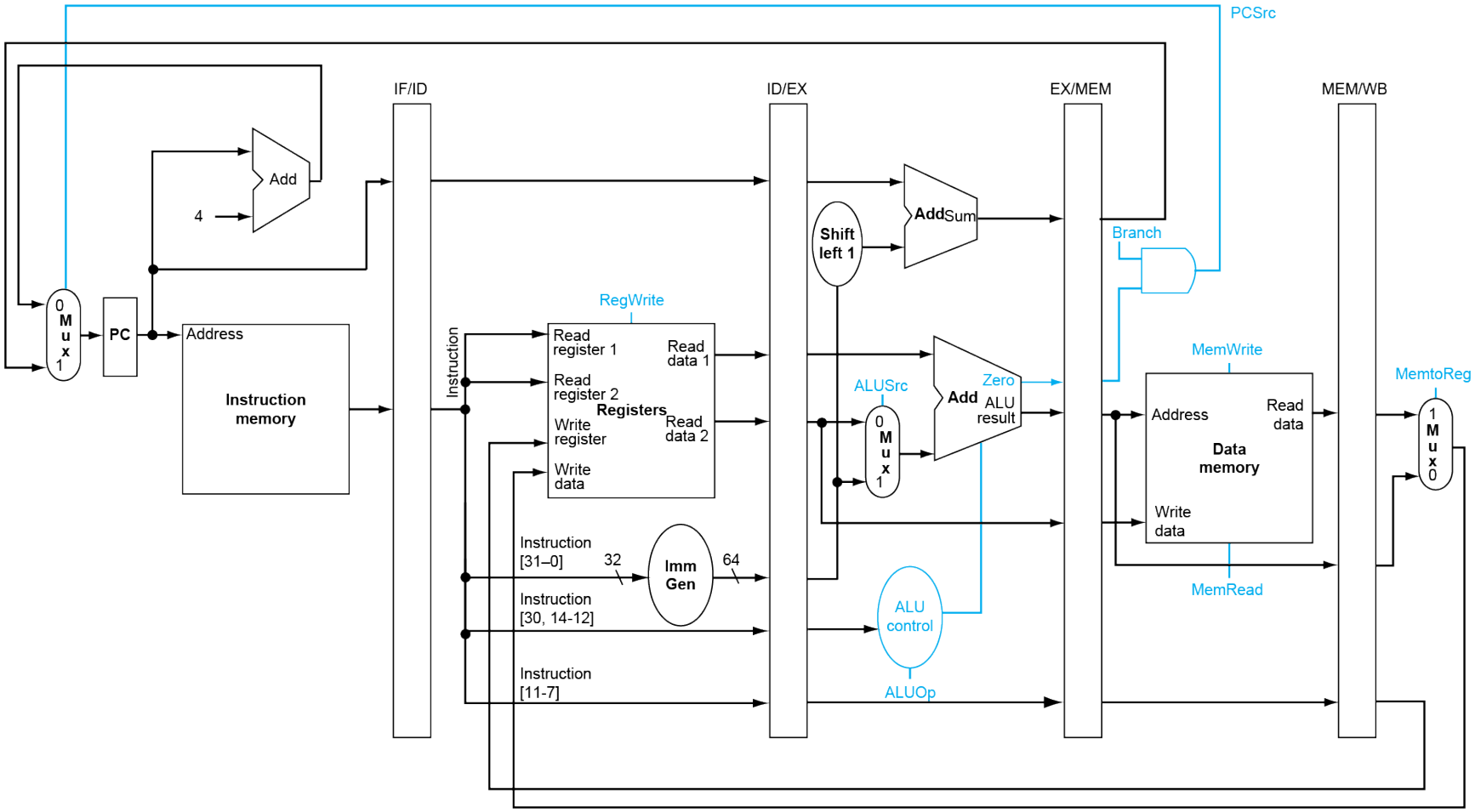
Diagrama Pipeline Simples-Ciclo

- Estado do pipeline em um simples ciclo

add x14, x5, x6	ld x13, 48(x1)	add x12, x3, x4	sub x11, x2, x3	ld x10, 40(x1)
Instruction fetch	Instruction decode	Execution	Memory	Write-back

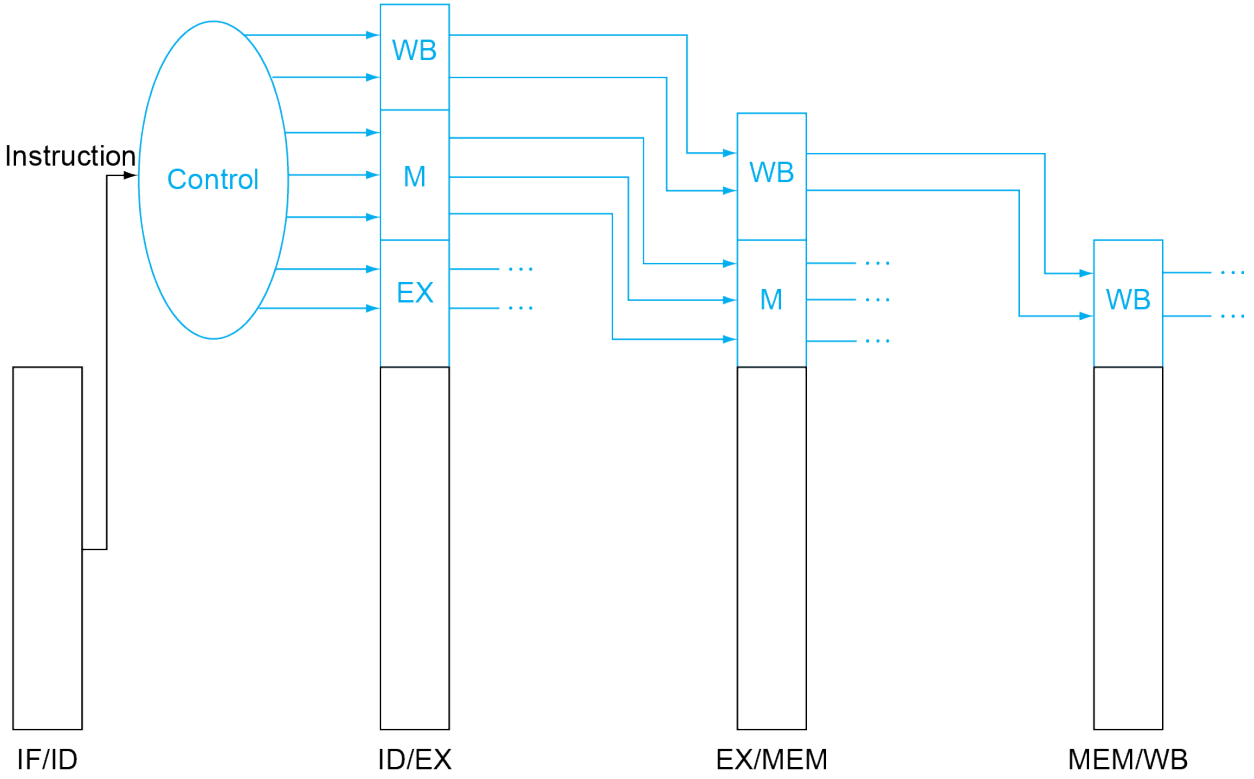


Controle Pipelined (Simplificado)

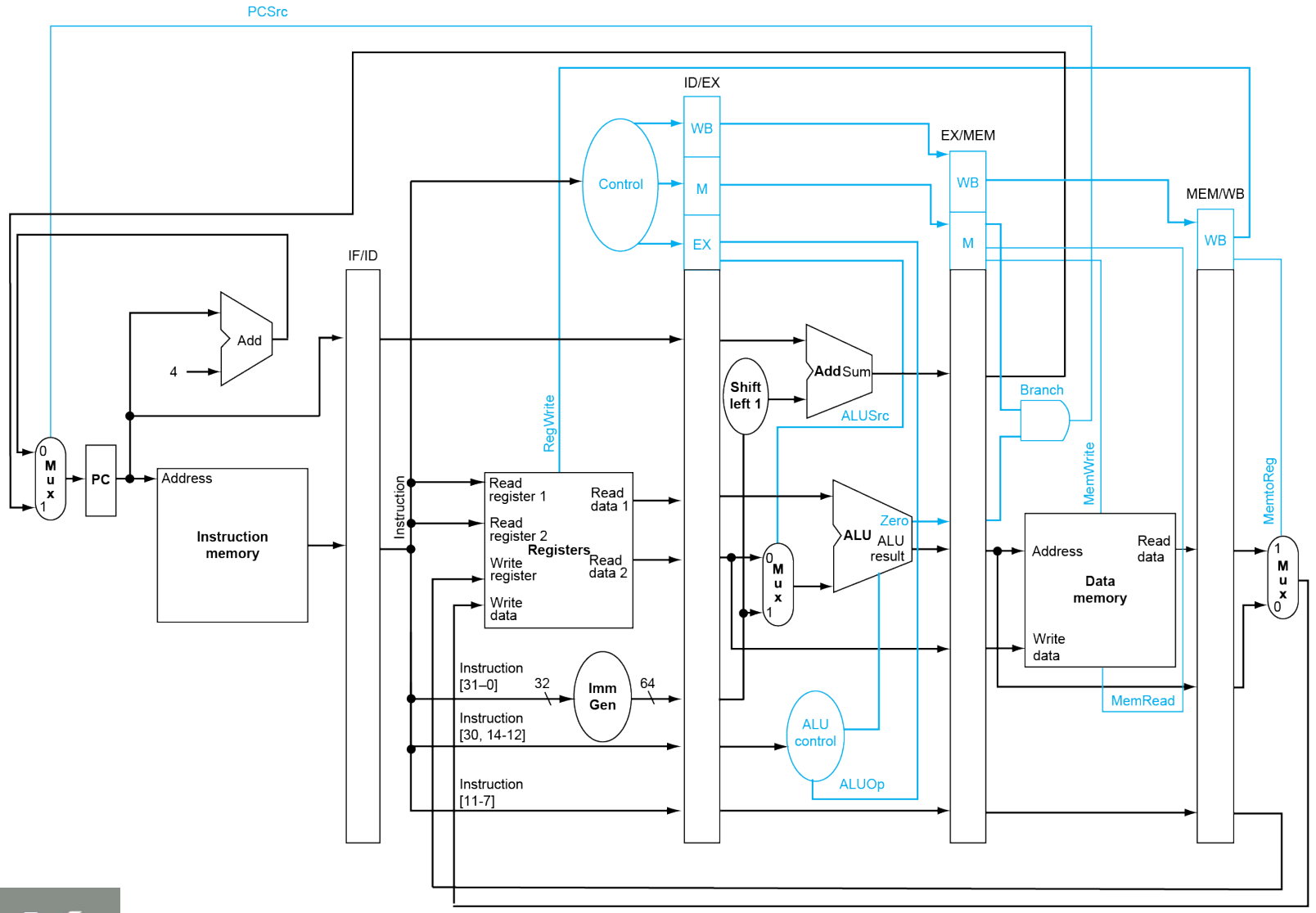


Controle pipelined

- Sinais de controle derivados da instrução
  - Como no ciclo único



Controle pipelined



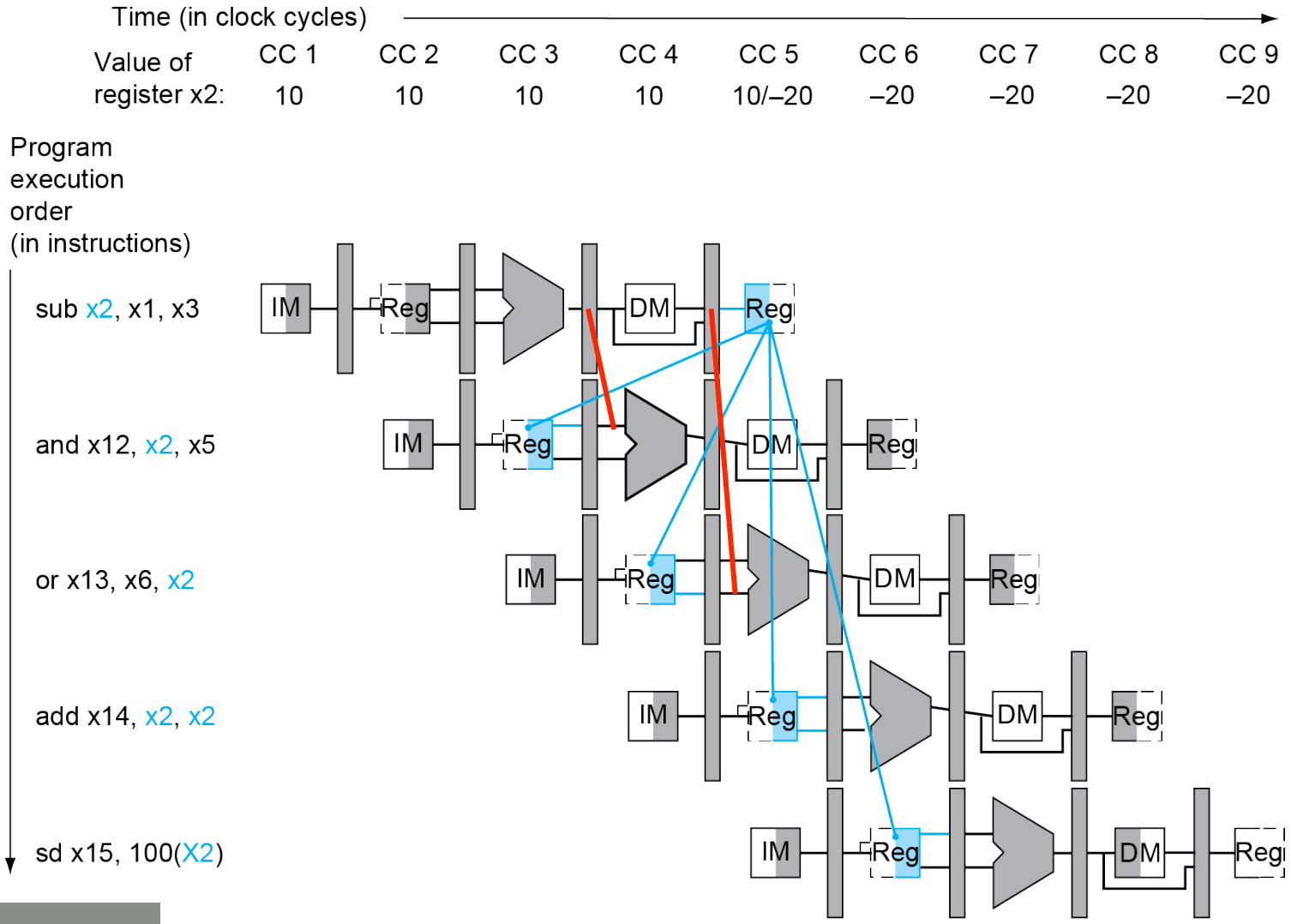
Hazards de dados em instruções ALU

- Considere esta sequência

```
sub  x2, x1,x3
and  x12,x2,x5
or   x13,x6,x2
add  x14,x2,x2
sd   x15,100(x2)
```

- Pode-se resolver hazards por encaminhamento
  - Como detectar quando usar?

Dependências e Encaminhamentos



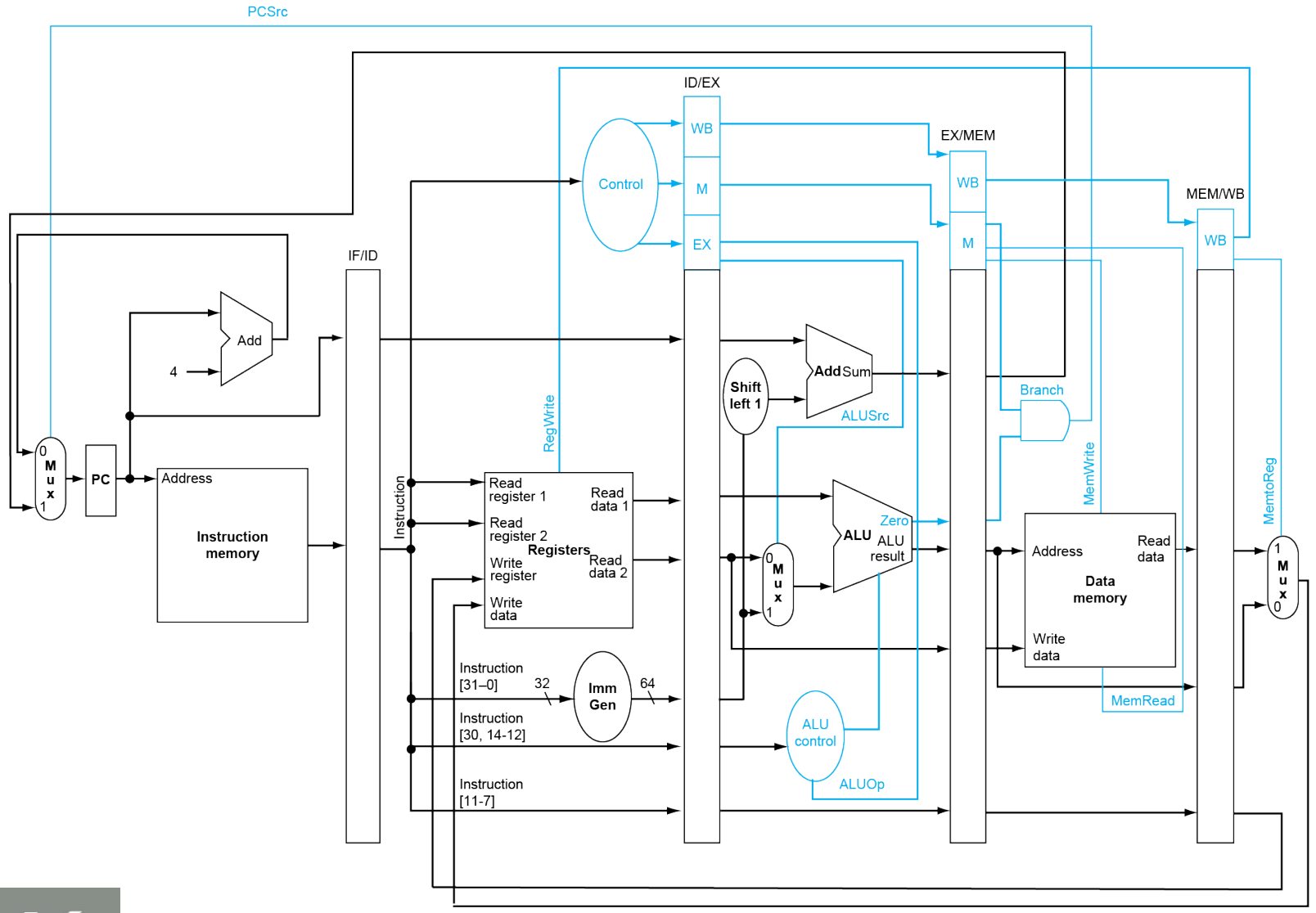
Detectando Encaminhamento

- Transmita os números dos registradores
  - ex., ID/EX.RegisterRs1 = número do registrador para Rs1 no registrador ID/EX
- Número do registradores operandos na ALU no estágio EX é dado por:
  - ID/EX.RegisterRs1, ID/EX.RegisterRs2
- Hazards de dados quando
  1a. EX/MEM.RegisterRd = ID/EX.RegisterRs1
  1b. EX/MEM.RegisterRd = ID/EX.RegisterRs2
  2a. MEM/WB.RegisterRd = ID/EX.RegisterRs1
  2b. MEM/WB.RegisterRd = ID/EX.RegisterRs2

Fwd do EX/MEM pipeline reg

Fwd do MEM/WB pipeline reg

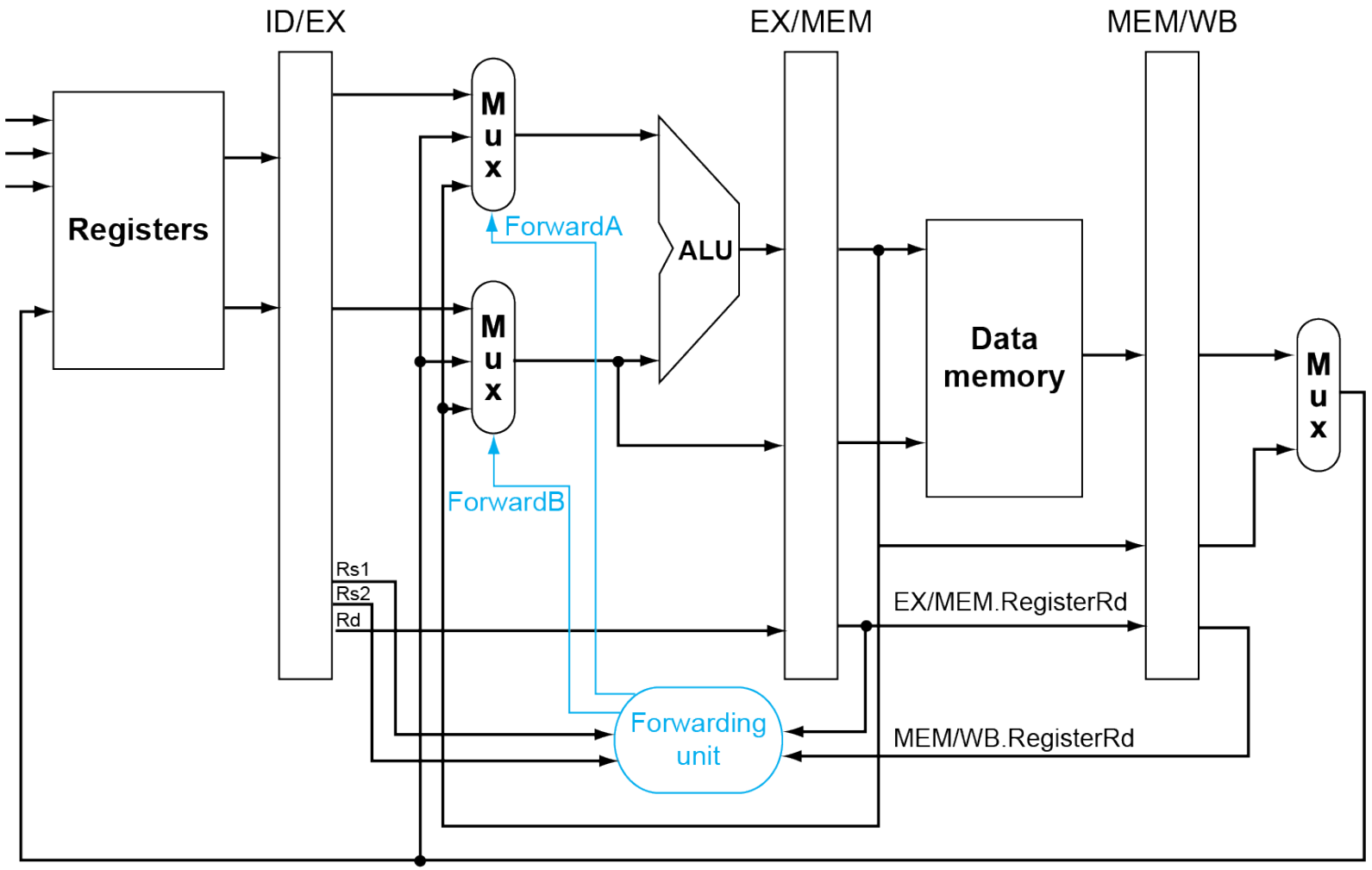
Controle pipelined



Detectando Encaminhamento

- Mas somente se instrução de encaminhamento vai escrever em um registrador
  - EX/MEM.RegWrite, MEM/WB.RegWrite
- E somente se Rd para esta instrução não é x0
  - EX/MEM.RegisterRd ≠ 0, MEM/WB.RegisterRd ≠ 0

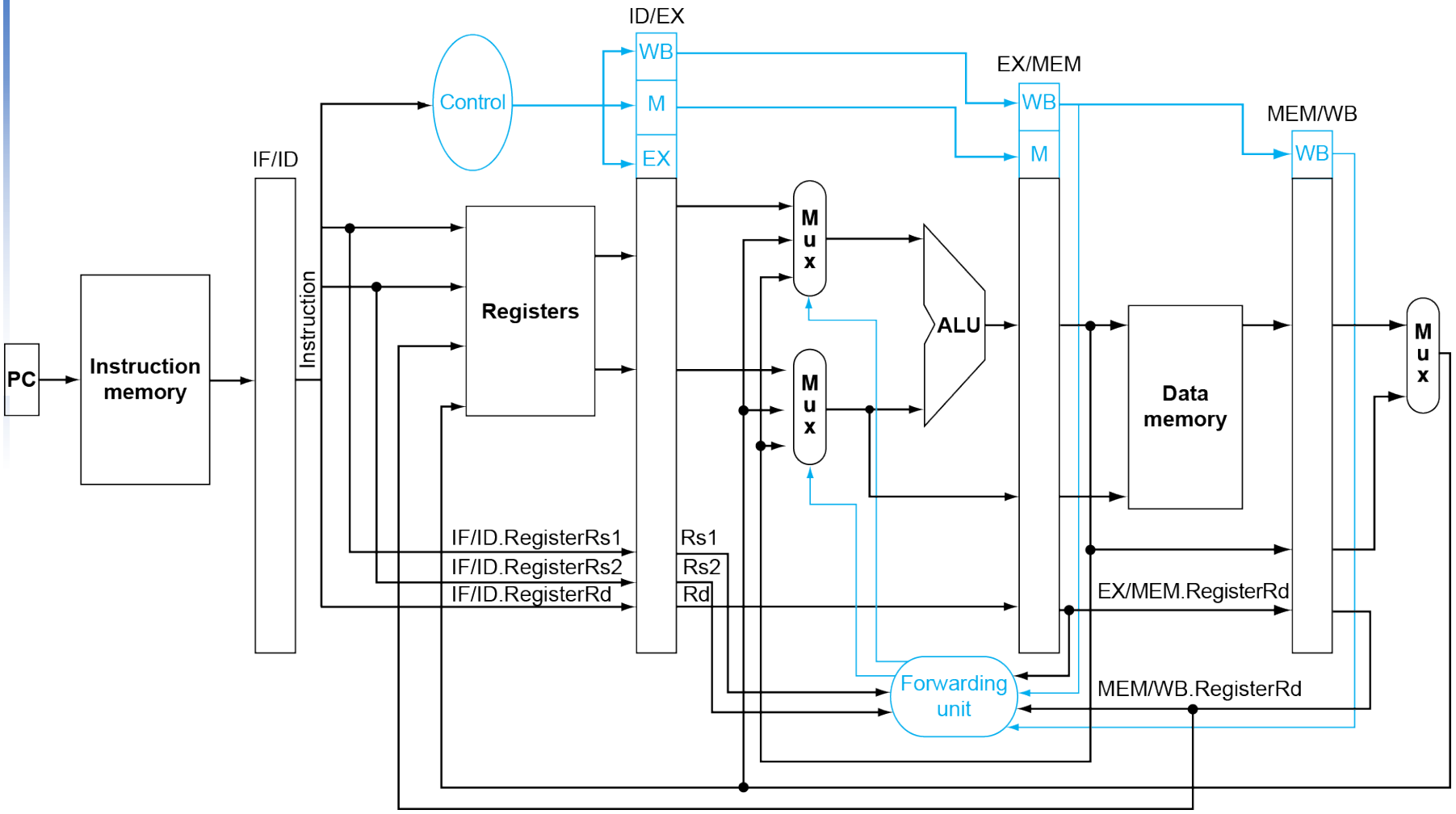
Caminhos do Encaminhamento



Condições de Encaminhamento

Mux controle	Origem	Explicação
ForwardA = 00	ID/EX	Primeiro operando da ALU vem do banco de registradores
ForwardA = 10	EX/MEM	Primeiro operando da ALU vem do cálculo anterior da ALU
ForwardA = 01	MEM/WB	Primeiro operando da ALU vem da memória ou de cálculo anterior da ALU
ForwardB = 00	ID/EX	Segundo operando da ALU vem do banco de registradores
ForwardB = 10	EX/MEM	Segundo operando da ALU vem do cálculo anterior da ALU
ForwardB = 01	MEM/WB	Segundo operando da ALU vem da memória ou de cálculo anterior da ALU

Datapath com Forwarding



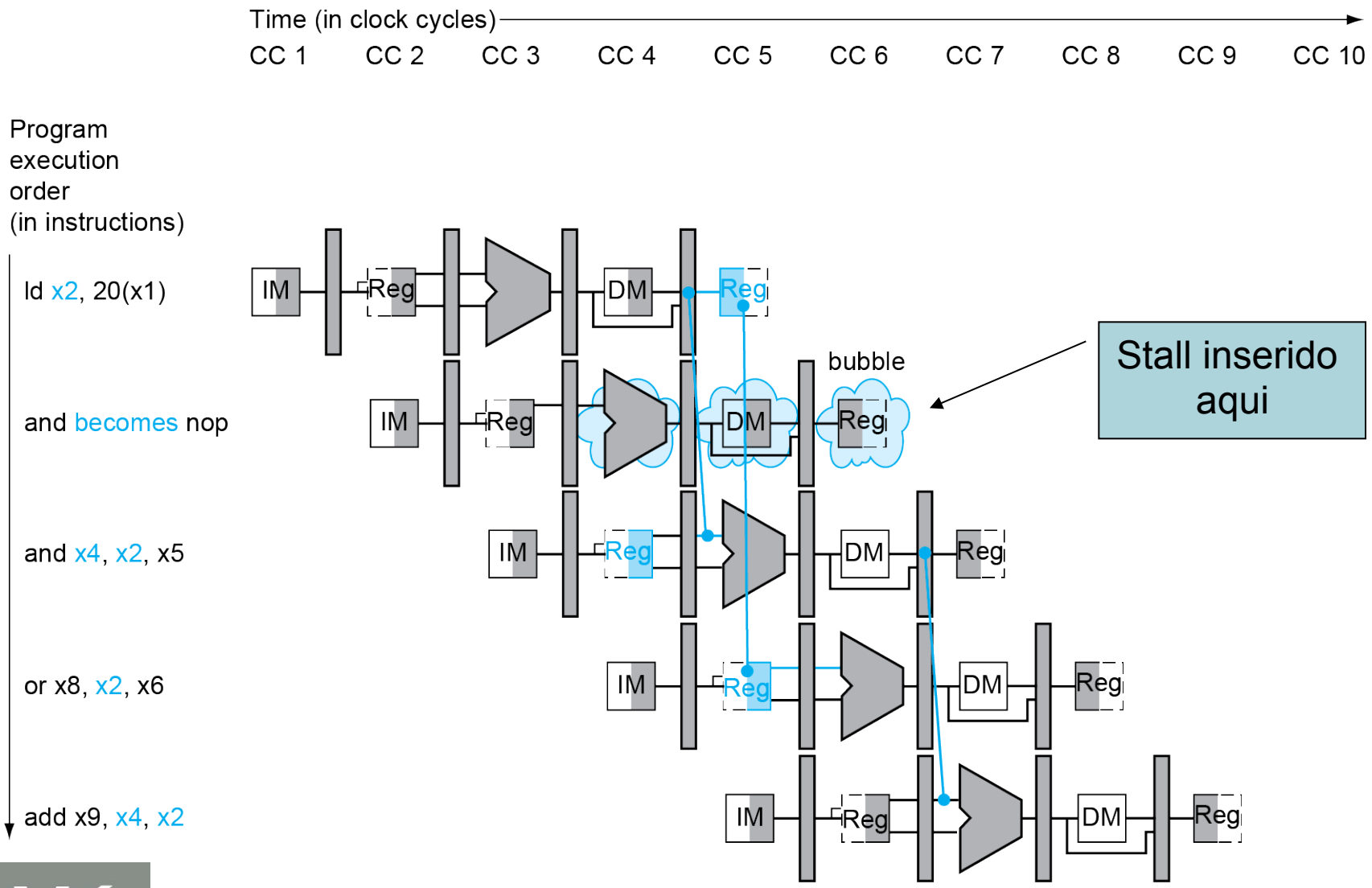
Load Detecção de Hazard

- Verifique a instrução no estágio de decodificação − ID
- Números dos registradores operandos da ALU no estágio ID
  - IF/ID.RegisterRs1, IF/ID.RegisterRs2
- Load hazard quando
  - ID/EX.MemRead and
    ((ID/EX.RegisterRd = IF/ID.RegisterRs1) or
    (ID/EX.RegisterRd = IF/ID.RegisterRs2))
- Se detectado, para (Stall)

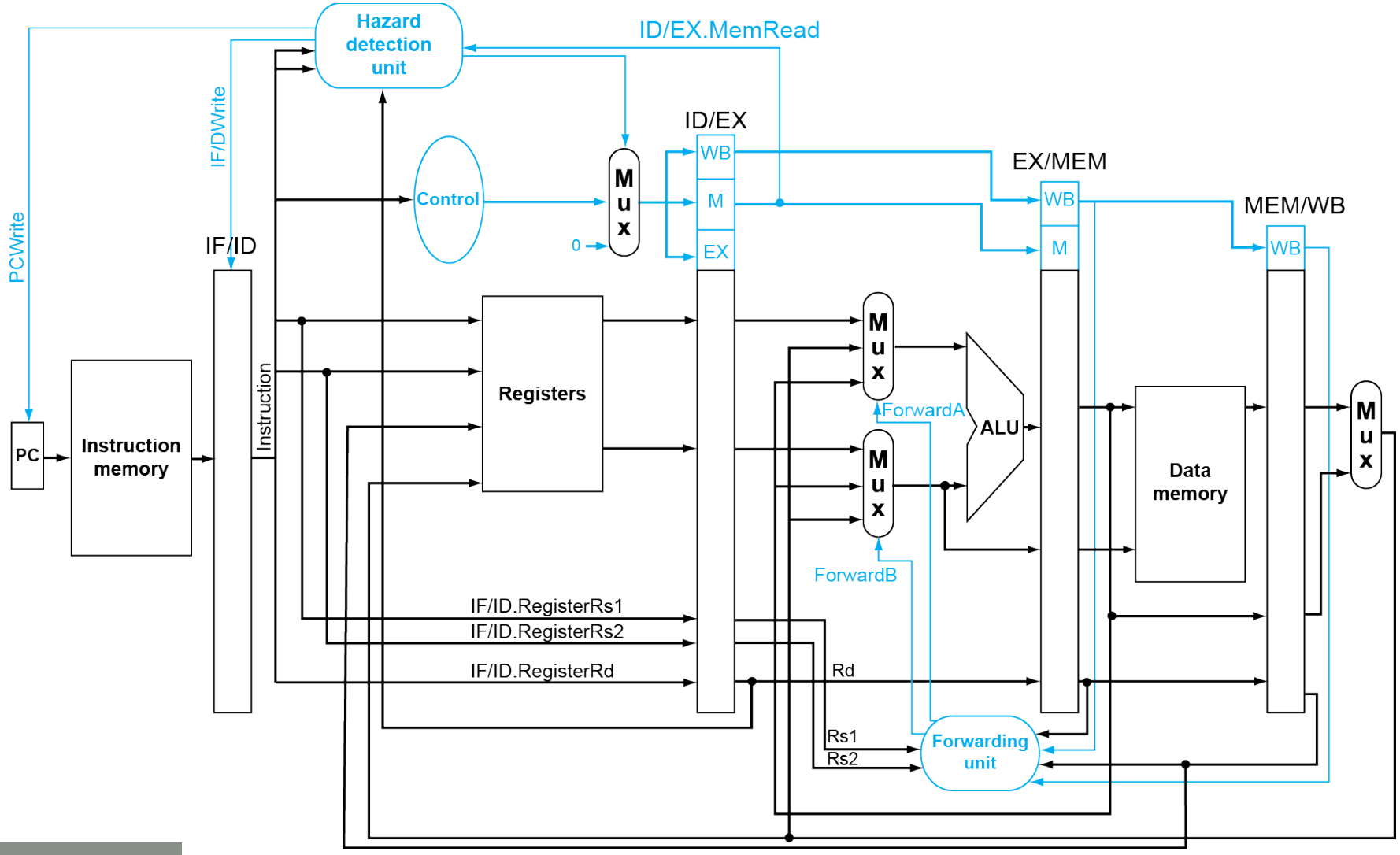
Como fazer Stall no Pipeline

- Força valores de controle no registrador ID/EX para 0
  - EX, MEM e WB fazem nop (*no-operation*)
- Previne alterar o PC e o registrador IF/ID
  - Instrução é decodificada novamente
  - Próxima instrução é buscada novamente
  - 1 ciclo de stall permite MEM ler o dado para ld
    - Em seguida encaminha para o EX

Load Detecção de Hazard



Datapath com detecção Hazard



Stalls e Desempenho

The BIG Picture

- Stalls reduz desempenho
  - Mas é necessário para ter valores corretos
- Compilador pode arranjar o código para evitar hazards e stalls
  - Requer conhecimento da estrutura do pipeline