제 17강. 4.8 데이터 해저드: 전방전달 대 지연, 4.9 제어 해저드

4.8 데이터 해저드: 전방전달 대 지연

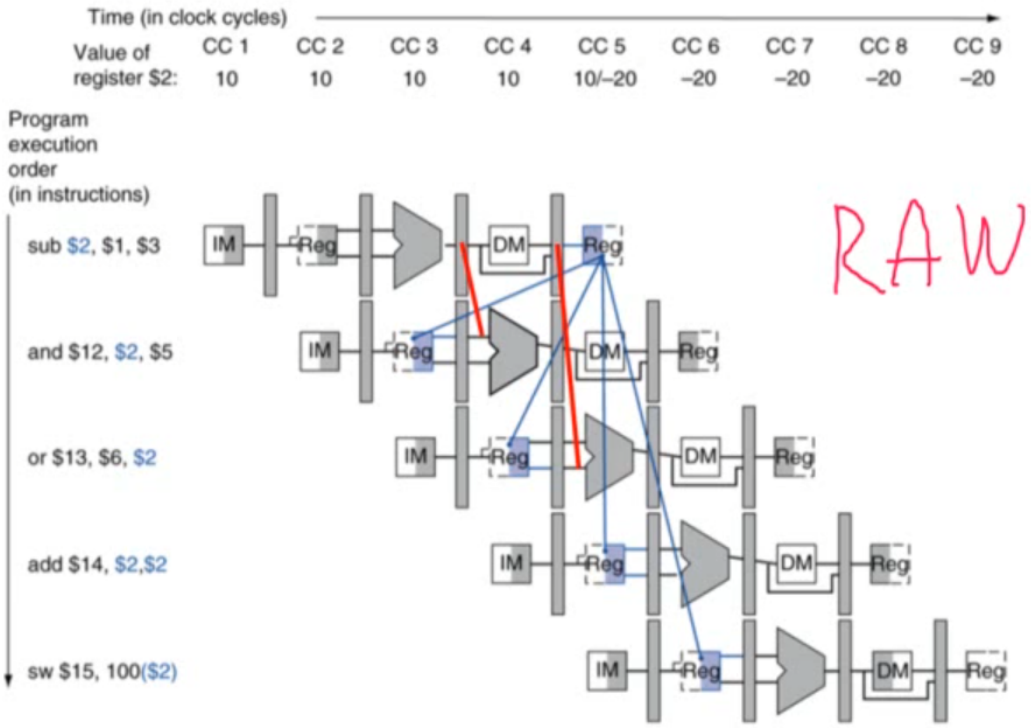


그림 4.52

레지스터 $2를 읽은 값은 sub 명령어의 결과값이 아니다.

필요한 결과값이 만들어지는 것은 EX 단계, 클럭 사이클 CC3의 끝.

AND와 OR 명령어가 실제로 데이터를 필요로 하는 시점은

ID 단계가 아니라 EX 단계, 즉 CC 4,5 시작할 때 각각 필요

전방전달(forward)하면 해결

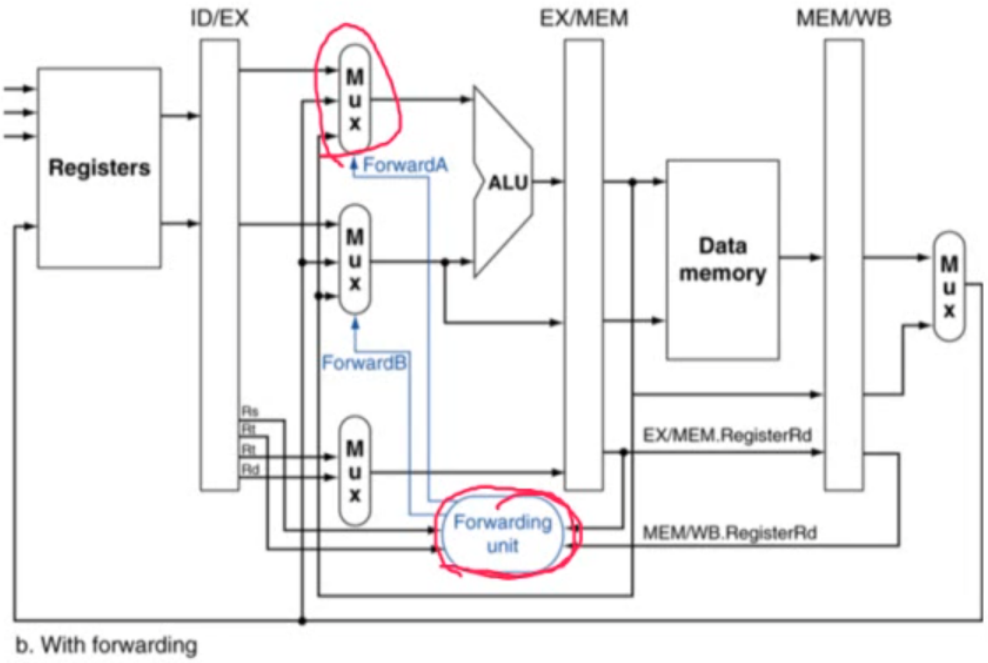
참고. 파이프라인 데이터패스

1. IF : 명령어 인출
2. ID : 명령어 해독 및 레지스터 파일 읽기

3. EX : 실행 또는 주소 계산
4. MEM : 데이터 메모리 접근
5. WB : 쓰기 (Write-back)

데이터 해저드가 일어나는 경우

1. EX/MEM RegisterRd (destination Register) = **ID/EX RegisterRs** (source register)
2. EX/MEM RegisterRd = ID/Ex Register Rt(target register)
   ㄴ EX/MEM 스테이지가 끝난 후
3. MEM/WB. RegisterRd = ID/EX.RegisterRs
4. MEM/WB. RegisterRd = ID/EX.RegisterRt
   ㄴ MEM/WB 스테이지 끝난 후



Forwarding unit

3x1 MUX를 쓰게 됨

forwarding unit은 ALU의 input을 결정

포워딩 조건

- EX hazard
  - if(EX/MEM. RegWrite and (EX/MEM.RegisterRd ≠ 0)
    and (EX/MEM.RegisterRd = ID/EX.RegisterRs))
    ForwardA = 10
  - if(EX/MEM. RegWrite and (EX/MEM.RegisterRd ≠ 0)
    and (EX/MEM.RegisterRd = ID/EX.RegisterRt))
    ForwardB = 10

EX/MEM RegisterRd 필드는

- ALU 명령어의 레지스터 목적지 (명령어의 Rd 필드)
- 적재 명령어의 레지스터 목적지 (명령어의 Rt 필드) 둘다 될 수 있음

파이프라인 레지스터 EX/MEM에서 값을 받도록 MUX를 제어

- MEM hazard
  - if(MEM/WB. RegWrite and (MEM/WB.RegisterRd ≠ 0)
    and (MEM/WB.RegisterRd = ID/EX.RegisterRs))
    ForwardA = 01
  - if(MEM/WB. RegWrite and (MEM/WB.RegisterRd ≠ 0)
    and (MEM/WB.RegisterRd = ID/EX.RegisterRt))
    ForwardB = 01

Double Data Hazard

MEM hazard

- if(MEM/WB. RegWrite and (MEM/WB.RegisterRd ≠ 0)
  and not(EX/MEM. RegWrite and (EX/MEM.RegisterRd ≠ 0)
  and (EX/MEM.RegisterRd = ID/EX.RegisterRs))
  and (MEM/WB.RegisterRd = ID/EX.RegisterRs))
  ForwardA = 01

- if(MEM/WB. RegWrite and (MEM/WB.RegisterRd ≠ 0)

  and not(EX/MEM. RegWrite and (EX/MEM.RegisterRd ≠ 0)

  and (EX/MEM.RegisterRd = ID/EX.RegisterRt))

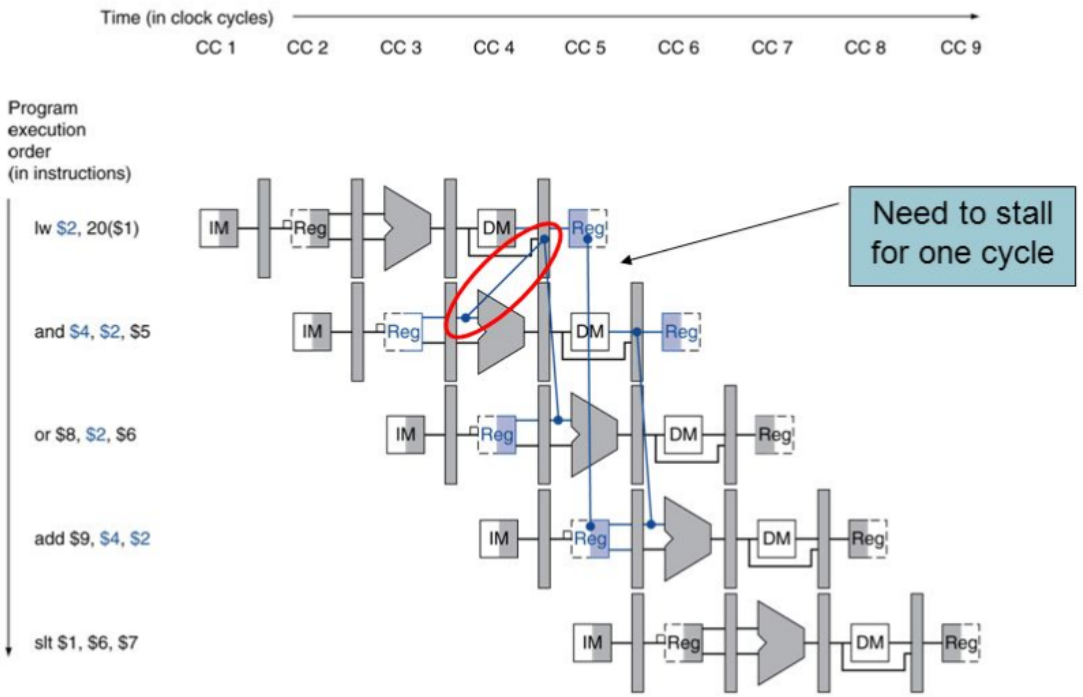
  and (MEM/WB.RegisterRd = ID/EX.RegisterRt))

  ForwardB = 01

데이터 해저드와 지연

전방전달이 해결 못하는 문제 하나는

적재 명령어를 따르는 명령어가 적재 명령어에서 쓰기를 행하는 레지스터를 읽으려고 시도할 때



Load-Use hazard가 발생하면

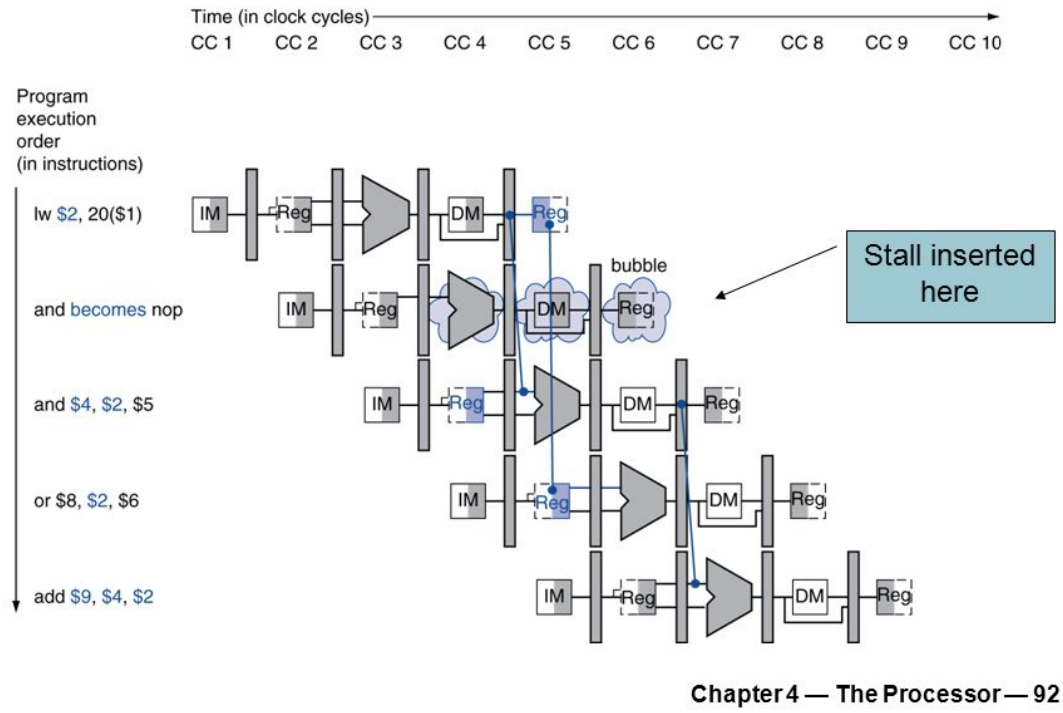
stall하고 bubble을 넣어 준다

control에다가 ID/EX register에 0값을 추가

- PC 와 IF/ID 레지스터 값이 업데이트되면 안 됨
  - 디코딩 다시

- IF fetch 다시
- 1-cycle stall은 MEM에서 데이터를 읽는 것을 허용
  - EX stage에서 forward하는 것 가능

Stall/Bubble in the Pipeline



Chapter 4 — The Processor — 92

and 명령어를 nop으로 바꿈으로써 CC4부터 거품이 삽입

and 명령어가 실제로는 CC 2,3에서 인출되고 해독되지만,

EX단계는 CC 5까지 연기됨. (지연이 없었다면 CC4에서 EX 단계에 들어갔을 것)

RECAP

- stall은 성능을 저하시킨다.

- 올바른 결과를 얻기 위해선 필요
- hazards와 stall을 피하기 위해서는 컴파일러가 코드를 이해해야 한다.
  - 파이프라인 구조에 대한 이해 필요

4.9 제어 해저드

분기가 일어나지 않는다고 가정

분기 지연은 너무 느리다

→ 더 좋은 방법은 분기가 일어나지 않는다 예측하고 명령어들을 순서대로 계속 실행하는 것

분기가 일어나면 해독되었던 명령어를 버리고 분기 목적지에서 실행을 계속함

명령어 버리기

적재-사용 데이터 해저드의 경우

원래의 제어값을 0으로 바꾸면 됨

파이프라인의 IF,ID,EX 단계에 있는 명령어를 쓸어내야(flush) 한다는 것 의미