Ch.4-2 The Processor - pipeline

▼ Performance issues

1. longest delay가 clock period가 됨
   - 가장 긴 놈 : **lw** instruction
     - instruction mem → register file → ALU → data mem → register file

   ⇒ design principle 다 망침

   ⇒ pipelining 사용해보자!

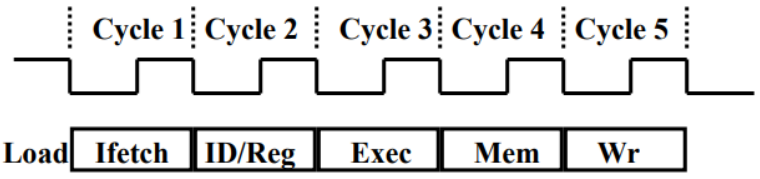
▼ Pipelining Lessons

1. pipelining : single task에 대한 latency는 영향 없음

   → but, 특정 시간 내의 처리↑ throughput 증가
2. pipeline 속도 : 가장 느린 pipeline stage에 의해 제한됨 longest delay
3. 여러 task가 다른 resource를 가지고 각자 independent하게 일함 ⇒ parallel
4. potential speedup ⇒ pipe stage의 개수 (나중에 stall, hazard의 문제로 인해 정비례하지 x)
5. pipe stage의 unbalnced한 길이는 speed up 속도 줄임
6. pipeline을 가득 채우는 시간과 비우는 시간은 speedUp을 줄임 (fill) (drain)
7. dependency → stall

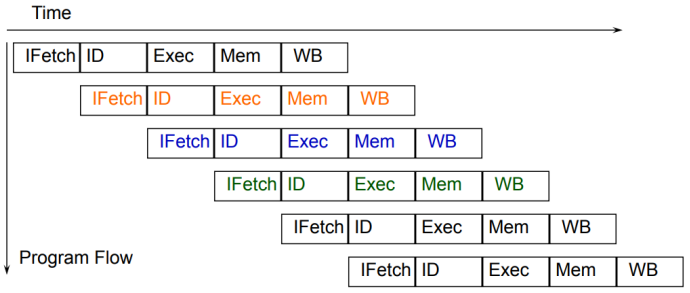
▼ MIPS pipeline

▼ Pipelined Execution

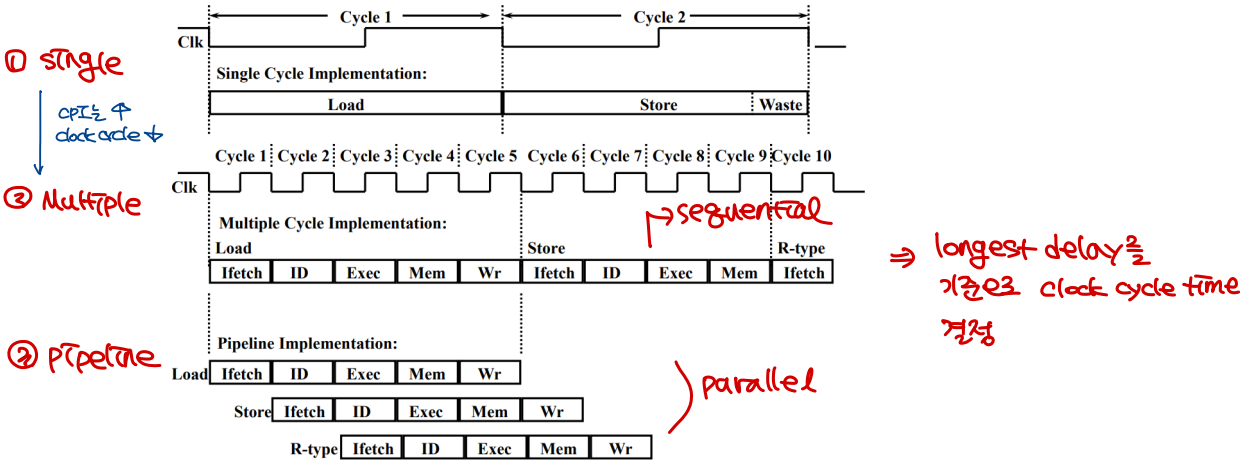


- five stage for instruction → stage마다 하나의 step

1. IF : instructon fetch → memory에서 instruction 가져옴
2. ID : instruction decode + register read → ① instruction read & decode ② register read 동시에
3. EX : execute operation + calculate address
4. MEM : access memory operand (피연산자)
5. WB : write result back to register (register에 result 쓰기)



- 여러 개의 instruction → 동시에 다양한 stage
- 각 instruction이 5번의 cycle 내에 실행된다고 해보자



▼ Pipeline Performance

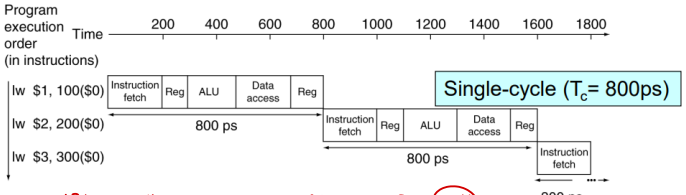
- assumption
  1. register r/w → 100ps
  2. other → 200ps

⇒ pipeline vs single - cycle

① single

Instr	Instr fetch	Register read	ALU op	Memory access	Register write	Total time
lw	200ps	100 ps	200ps	200ps	100 ps	800ps
sw	200ps	100 ps	200ps	200ps		700ps
R-format	200ps	100 ps	200ps		100 ps	600ps
beq	200ps	100 ps	200ps			500ps

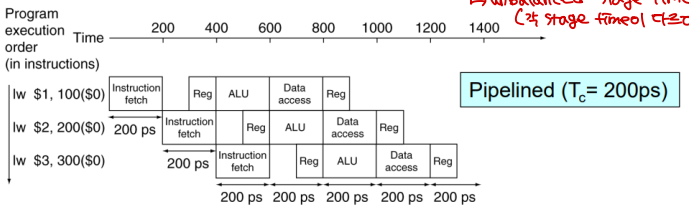
single



⇒ 이상적인 speedUp은 stage가 5개이므로 5배, But 4배

↳ unbalanced stage time 때문이다. (각 stage time이 다르다)

pipeline



- pipeline speedup → throughput 증가때문에 발생함. not instruction time!

$$T\ btw\ inst_{pipelined} = \frac{T\ btw\ inst_{no\ pipelined}}{Number\ of\ stages}$$

+) example

cycle time × CPI × instruction

time/cycle × cycle/instruction × inst.

가정
① instruction : 100개
② single-cycle의 cycle time : 45ns
③ Multi-cycle, pipelined cycle time : 10ns
④ Multi-cycle의 CPI : 3.6

i) single : 45ns/cycle × 1CPI × 100inst = 4500ns

ii) Multi : 10ns/cycle × 3.6CPI × 100inst = 3600ns

iii) Idle pipelined : 10ns/cycle × (1CPI × 100inst + 4cycle) = 1040ns

↳ pipeline fill, drain

⇒ SpeedUp = 4500ns / 1040ns = 4.33

▼ Pipeline datapath

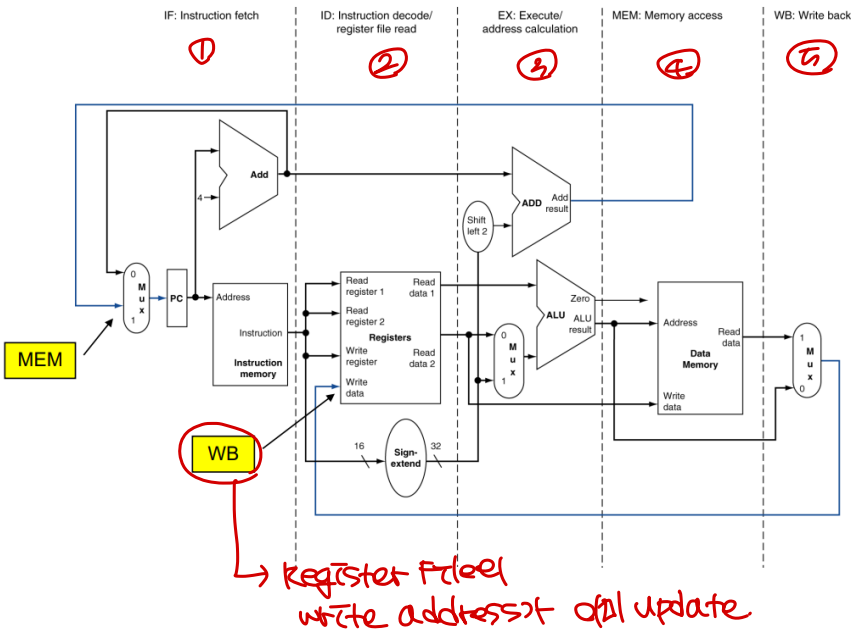
- pipelining and ISA design

1. 모든 instruction은 32bit ⇒ 같은 길이를 가진다
   - fetch, decode가 더 쉽다.
2. instruction의 format이 굉장히 적다 → 고정된 형식
   - decode + read reg → 한 stage 내에 가능
3. Lw/Sw addressing → memory access stage가 모두 정해져 있다
4. memory access는 오직 한 cycle 내에서만 진행된다
   ↳ operand가 memory에 ordered

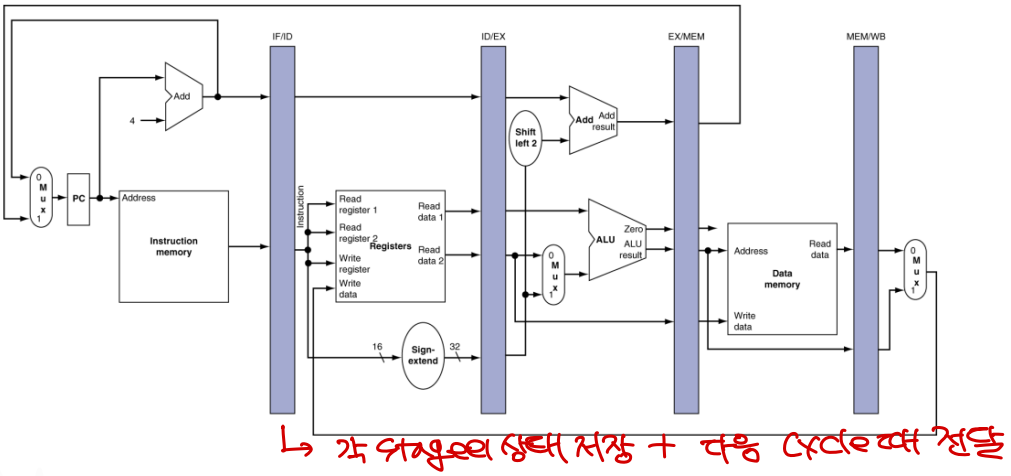
③ EX에서 address 계산

④ MEM에서 memory access

- MIPS Pipelined Datapath



- pipeline registers
  - stage 사이에 register 필요 → 이전 cycle에서 만들어진 일시적인 결과 모두 저장하고 있어야 함



▼ Pipeline Operation

Instruction: cycle-by-cycle flow

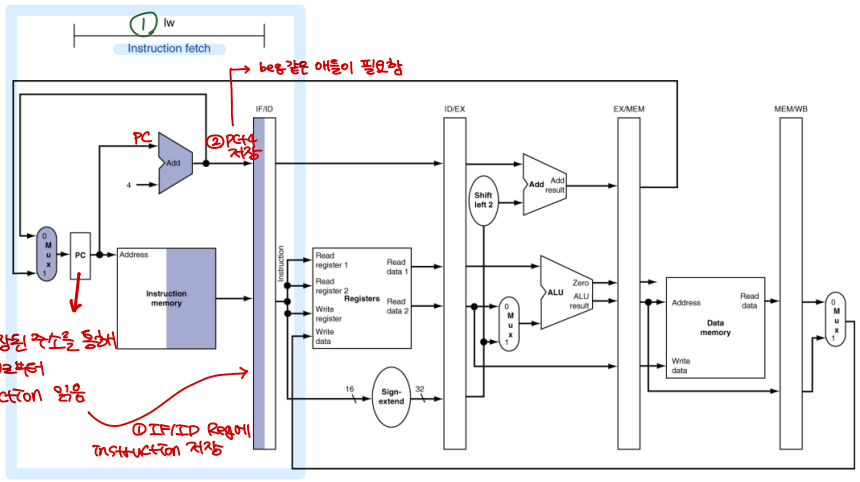
- 우리는 Load/Store 명령어에 대해 우선 "Single-clock-cycle" diagram을 통해 앞으로 살펴볼 것이다.
- 참고:

  Single-Clock-Cycle Diagram: 특정 cycle에서 사용되는 부분을 표시하고, 단계별로 어떤 instruction을 실행하고 있는지 표시한 Diagram

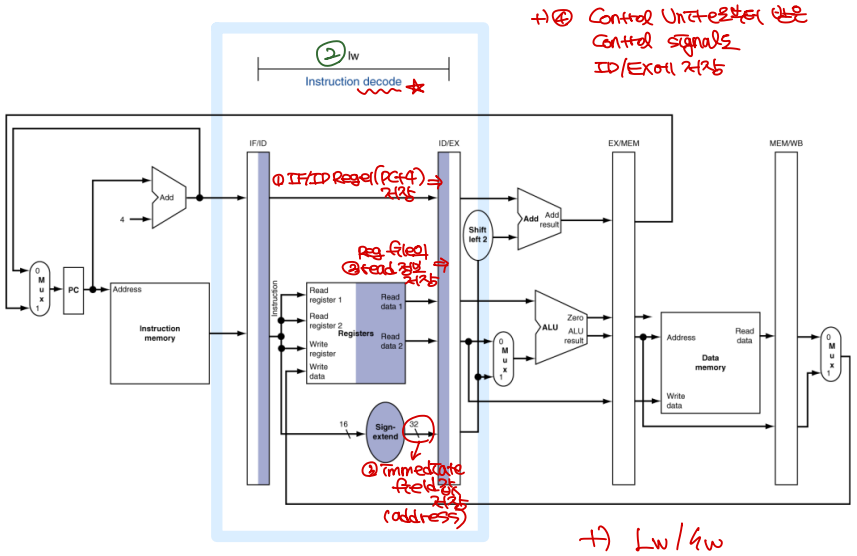
  Multi-Clock-Cycle Diagram: 시간 진행에 따라 operation의 동작을 나타낸 Graph (Graph of operation over time)

① single clock cycle Diagram

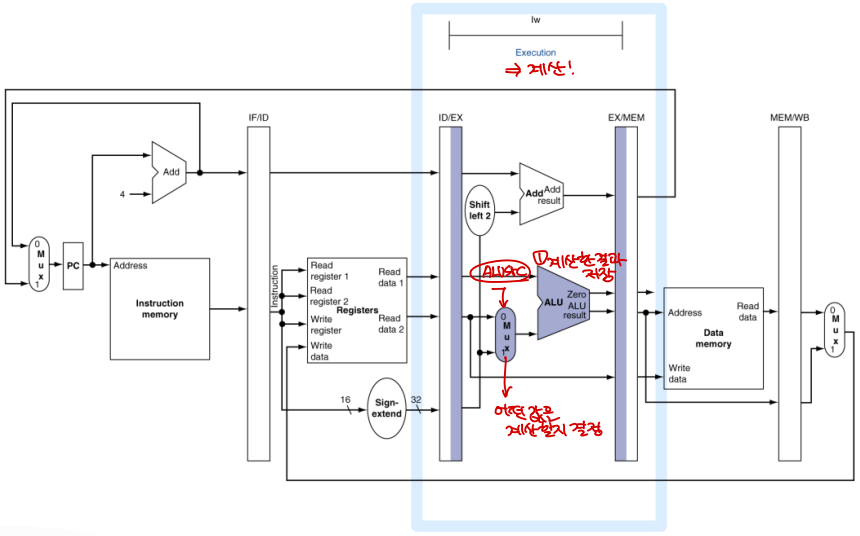
- Load
  - IF



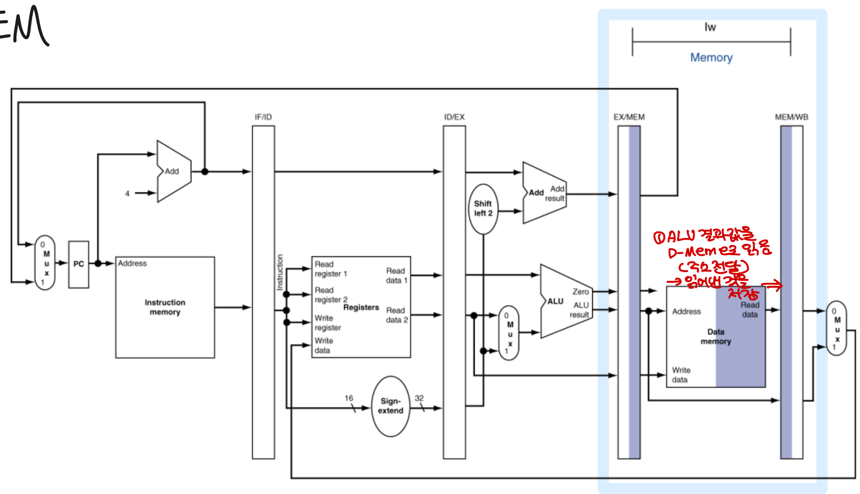
  - ID



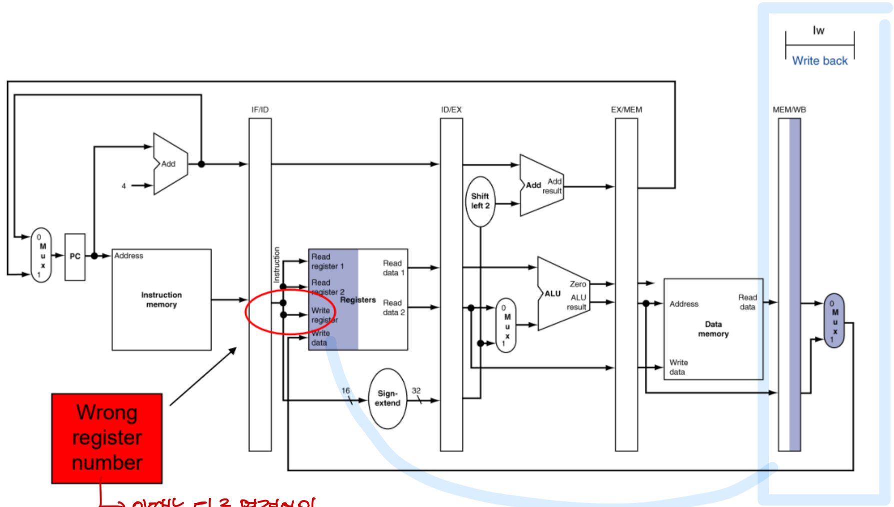
  - EX



- MEM



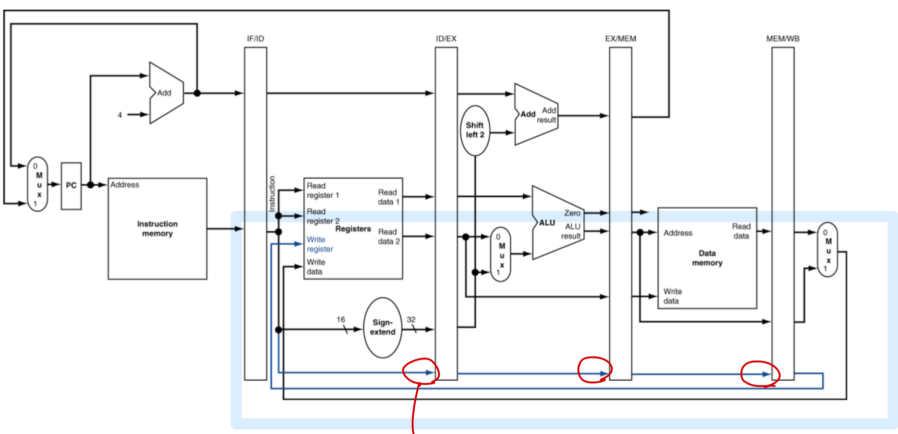
- WB



↳ 이때는 다른 명령어의 rt값이 전달될 수 있다.

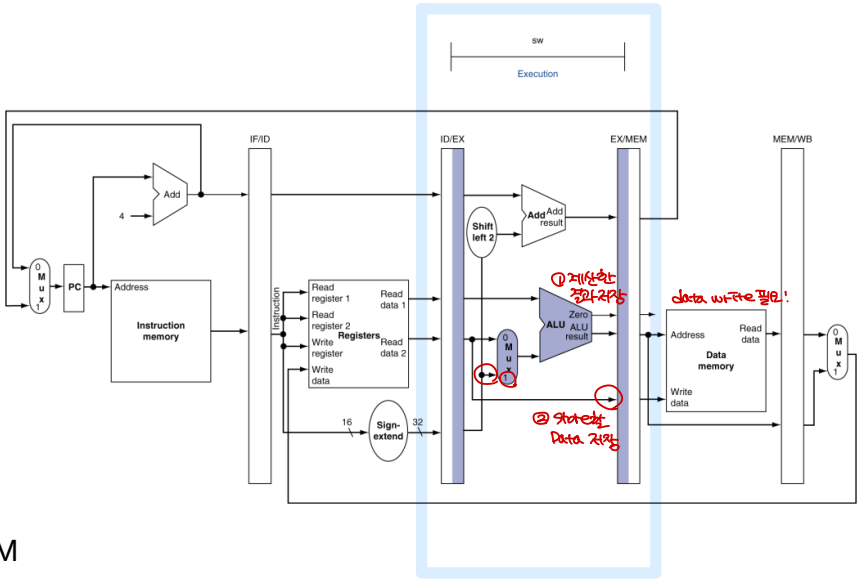
① Reg file에 data write

⇒ concurrency 적용

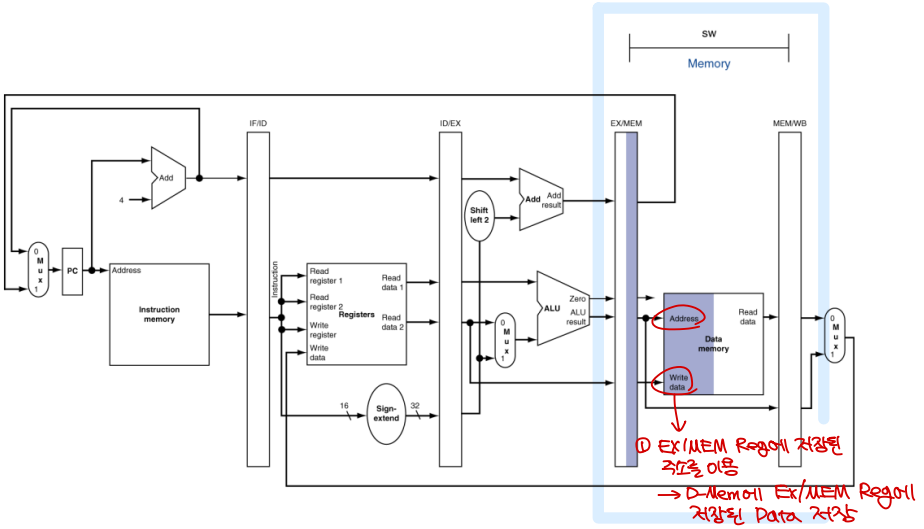


rt field의 값을 계속 저장
↳ 올바른 reg file에 최종결과 write할 수 있도록!

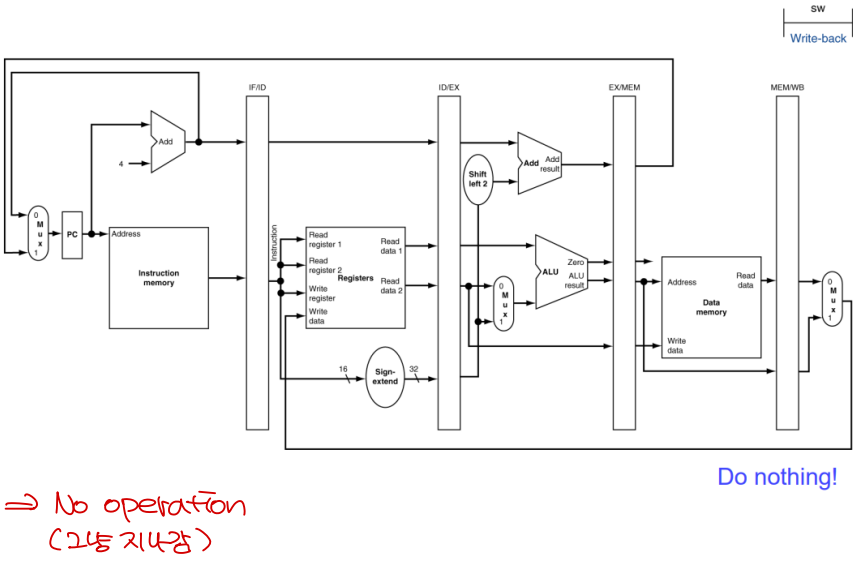
- Store
  - IF, ID는 Load와 동일
  - EX



  - MEM



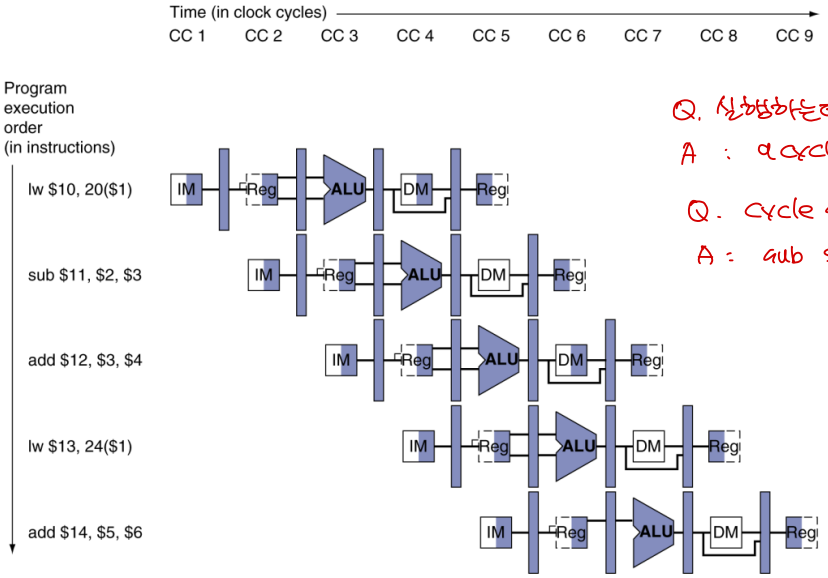
  - WB



⇒ No operation
(그냥 지나감)

② Multi-Cycle Pipeline diagram

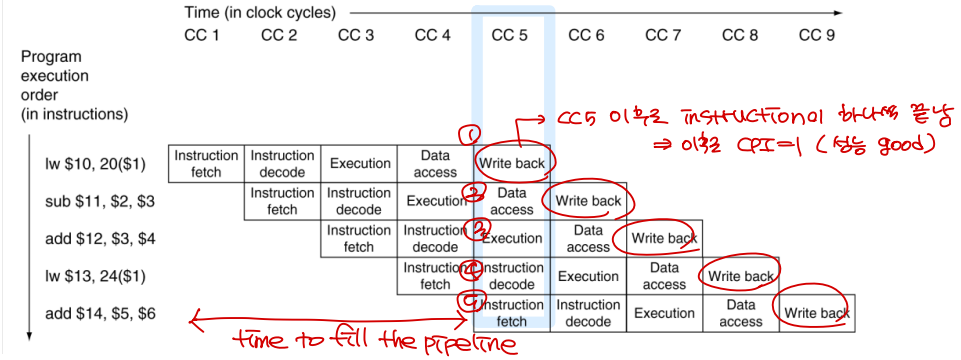
- Multi vs Single
  - ~~Multi-cycle~~ pipeline diagram



Q. 실행하는데 걸리는 총 cycle
A : 9 cycle

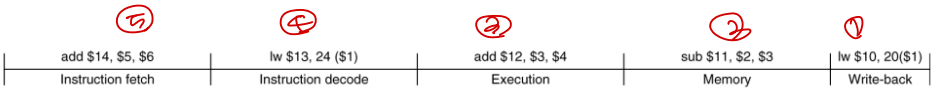
Q. cycle 4에서 ALU가 하는일?
A : sub $11, $2, $3

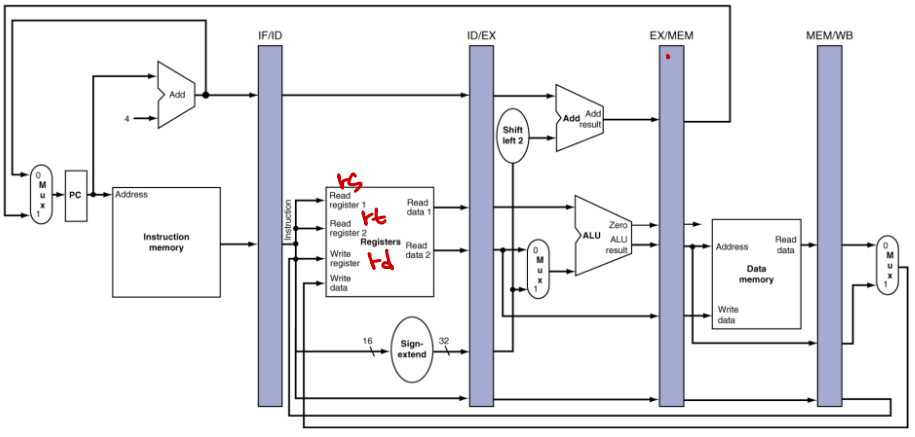
*resource usage form

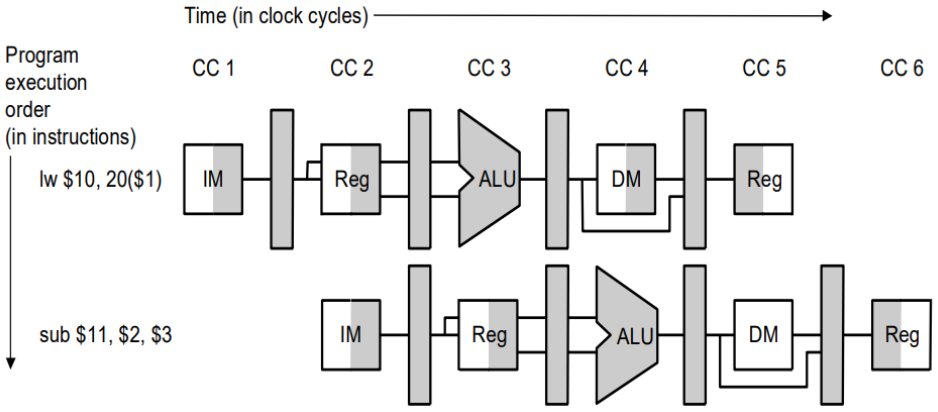


CC5 이후로 instructional 하나씩 끝남
⇒ 이상적 CPI=1 (성능 good)

  - Single-cycle pipeline diagram
    - clock cycle 5번째일 때







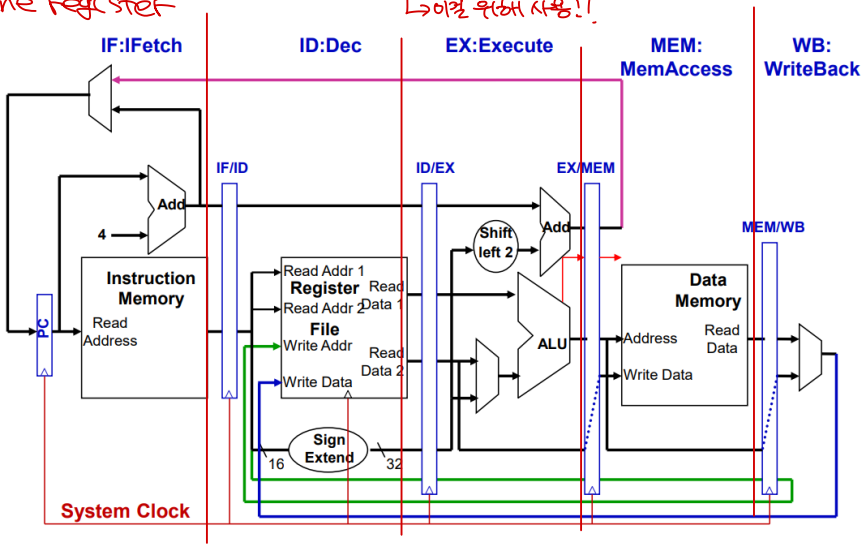
▼ MIPS datapth addtions/mods

- State registers : 각 pipeline stage를 독립적으로 나눔

≠ pipeline register

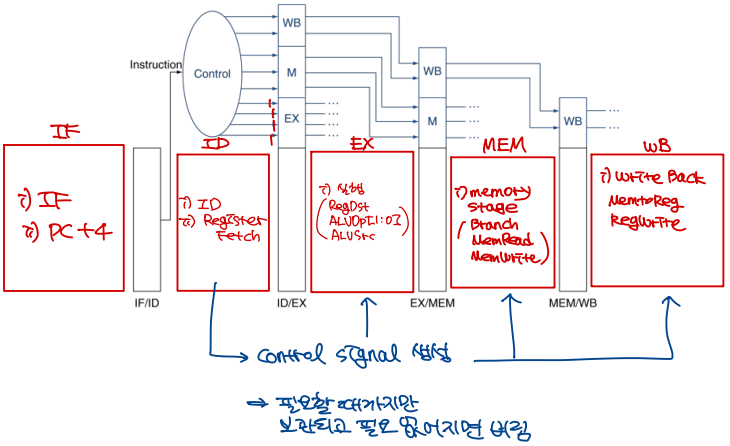
↳ 이것 위해 사용!

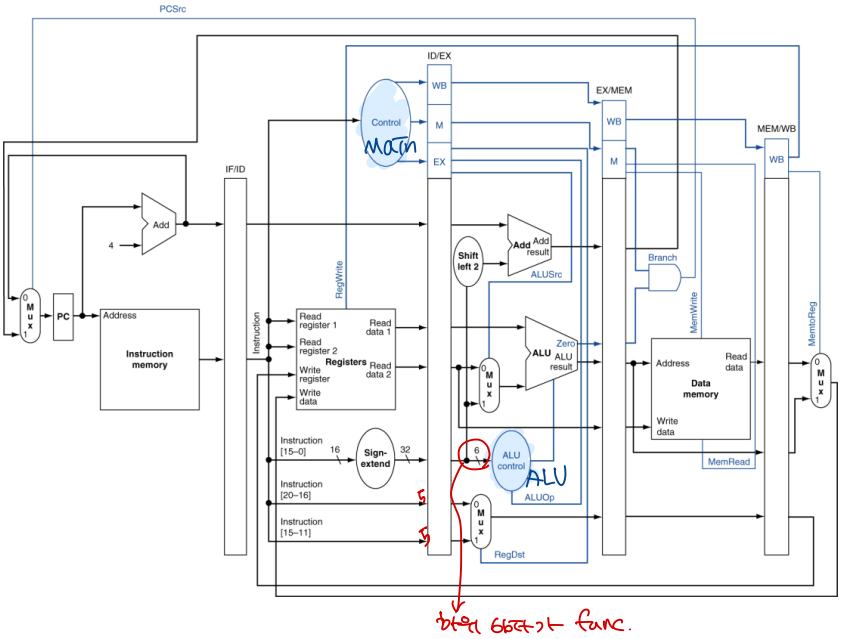
독립성 보장
data 전파
pipeline Hazard 주의

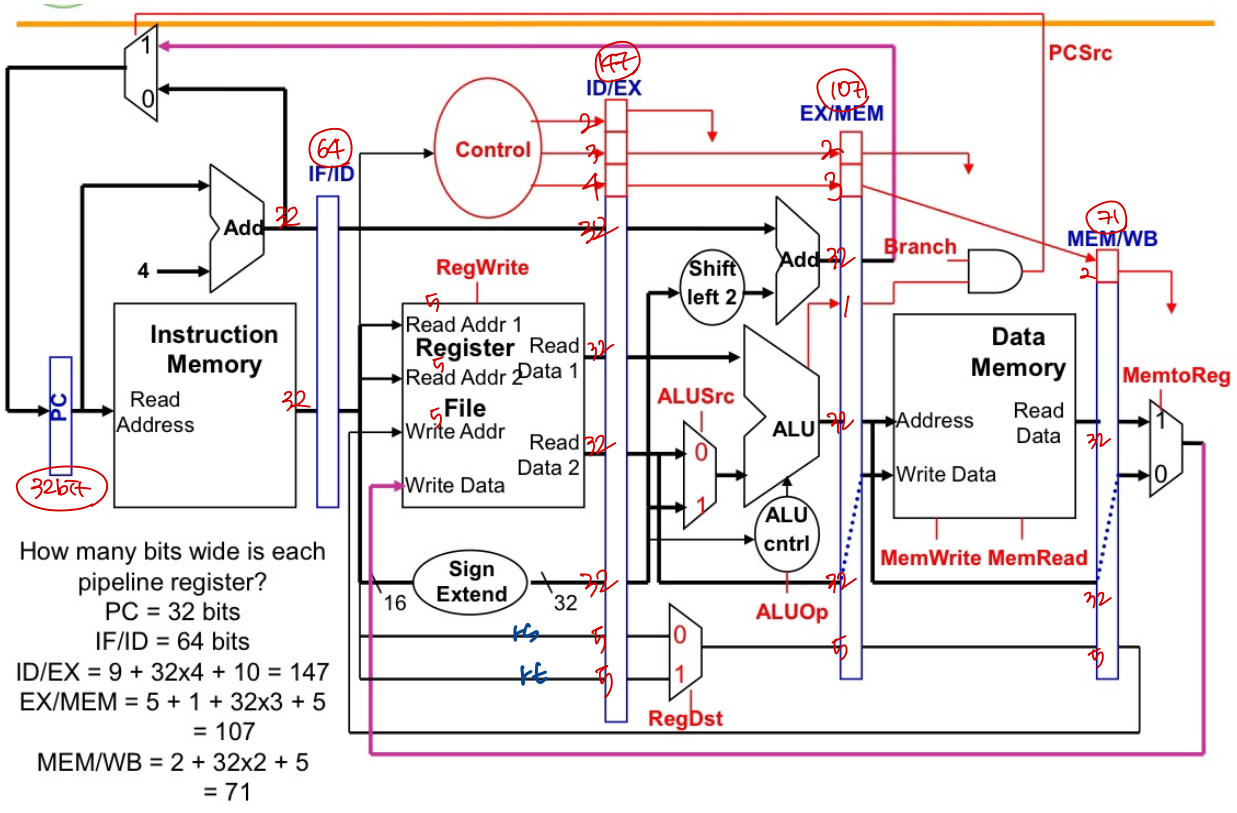


▼ Pipelined Control ⇒ instruction을 위한 control signal 생성해내기!

Instruction	Execution/Address Calculation stage control lines				Memory access stage control lines			stage control lines	
	Reg Dst	ALU Op1	ALU Op0	ALU Src	Branch	Mem Read	Mem Write	Reg write	Mem to Reg
R-format	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







How many bits wide is each pipeline register?
PC = 32 bits
IF/ID = 64 bits
ID/EX = 9 + 32x4 + 10 = 147
EX/MEM = 5 + 1 + 32x3 + 5 = 107
MEM/WB = 2 + 32x2 + 5 = 71