제2장 CPU의 구조와 기능 (2)

- 명령어 실행 과정에 대한 실습
- 간접 사이클(Indirect Cycle)
- 인터럽트 사이클(Interrupt Cycle)
- 명령어 파이프라이닝(Pipelining)
  - Speedup
  - Super Scalar 방식

CPU에서의 명령어 실행 과정

예

주소	명령어		기계 코드
100	LOAD	250	1250
101	ADD	251	5251
102	STA	251	2251
103	JUMP	170	8170

- LOAD 명령 : 1
- STA 명령 : 2
- ADD 명령 : 5
- JUMP 명령 : 8

t0 : MAR ← PC
t1: MBR ←M[MAR], PC ←PC+1
t2: IR ←MBR

프로그램 실행 과정

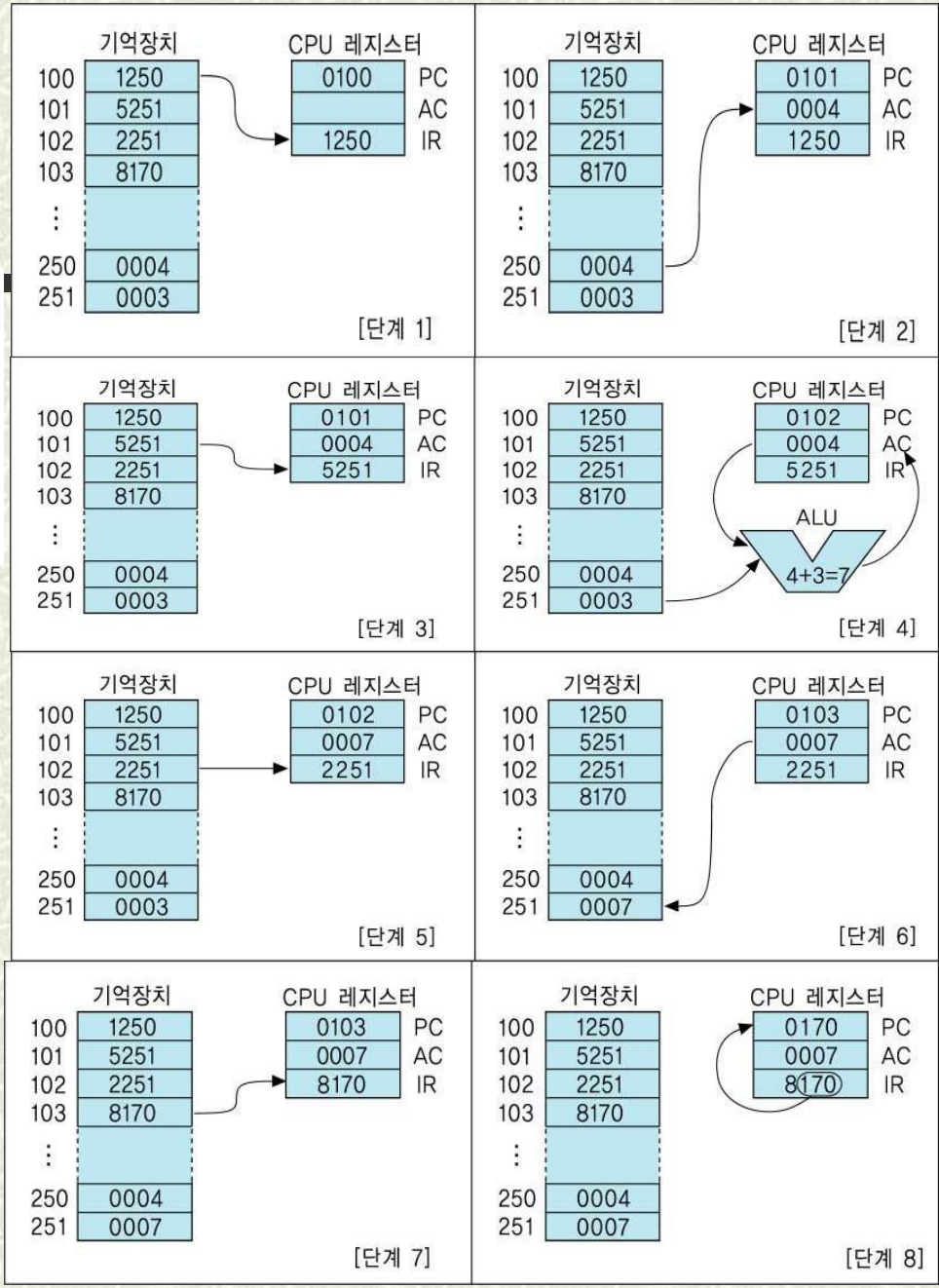
주소	명령어		기계 코드
100	LOAD	250	1250
101	ADD	251	5251
102	STA	251	2251
103	JUMP	170	8170

t0 : MAR ← IR(*addr*)
t1: MBR ←M[MAR]
t2: AC ←MBR

t0 : MAR ←IR(*addr*)
t1: MBR ←M [MAR]
t2: AC ←AC + MBR

t0 : MAR ← IR(*addr*)
t1: MBR ←AC
t2: M[MAR] ← MBR

t0 : PC ← IR(*addr*)



2.2.4 간접 사이클(indirect cycle)

- 명령어에 포함되어 있는 주소를 이용하여, 그 **명령어 실행에 필요한 데이터의 주소**를 인출하는 사이클

  → 간접 주소지정 방식(indirect addressing mode)에서 사용

- 인출 사이클과 실행 사이클 사이에 위치
- 간접 사이클에서 수행될 마이크로-연산

  t_0 : MAR ← IR(addr)

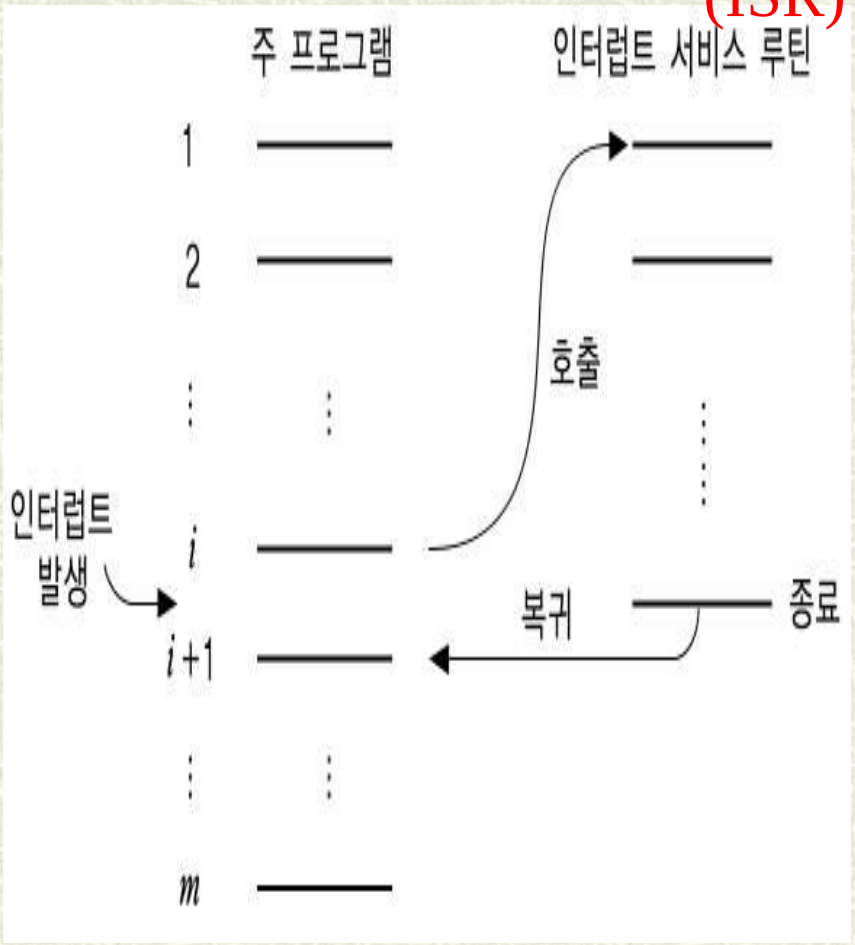
  t_1 : MBR ← M[MAR]

  t_2 : IR(addr) ← MBR

- 인출된 명령어의 주소필드 내용을 이용하여 기억장치로부터 데이터의 실제 주소를 인출하여 **IR의 주소필드에 저장**

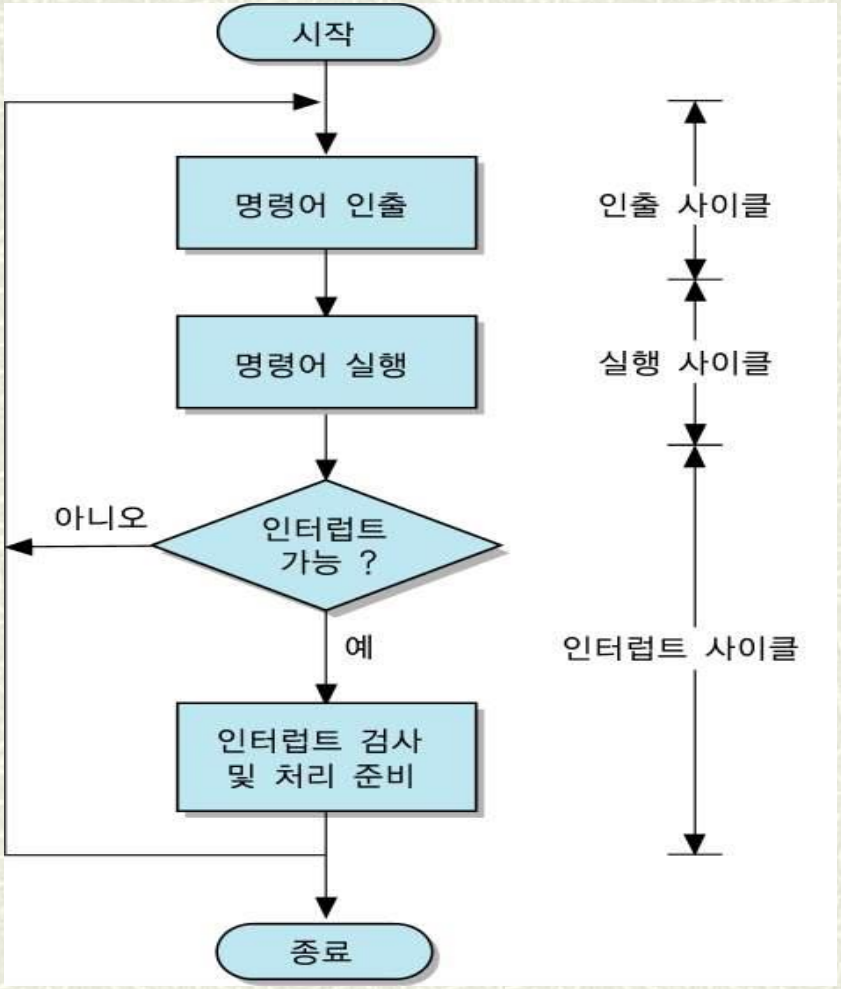
인터럽트 사이클 (Interrupt Cycle)

- CPU의 정상적인 처리를 중단 시킴
- 외부로부터 인터럽트 요구가 들어오면, CPU는 원래의 프로그램 수행을 중단하고, 요구된 인터럽트 처리
  - 인터럽트 서비스 루틴 (Interrupt Service Routine)



인터럽트 사이클 (Interrupt Cycle)

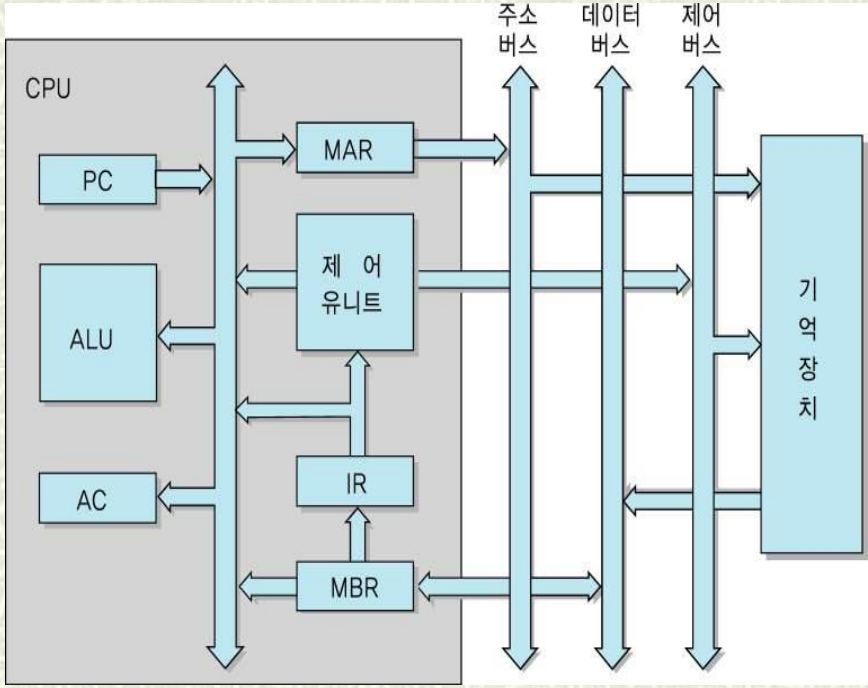
- CPU는 각 명령 실행 cycle후 에 인터럽트 요구신호를 검사함



인터럽트 사이클 (Interrupt Cycle)

- **인터럽트 요구시 CPU 동작**
  - 현재 명령실행을 끝낸 즉시, 다음 명령의 주소(PC)를 stack에 저장함.
  - ISR를 호출하기 위하여 루틴의 시작 주소를 PC에 적재한다.

- t0 : MBR← PC
- t1: MAR ← SP, PC ← ISR시작주소
- t2: M[MAR] ← MBR

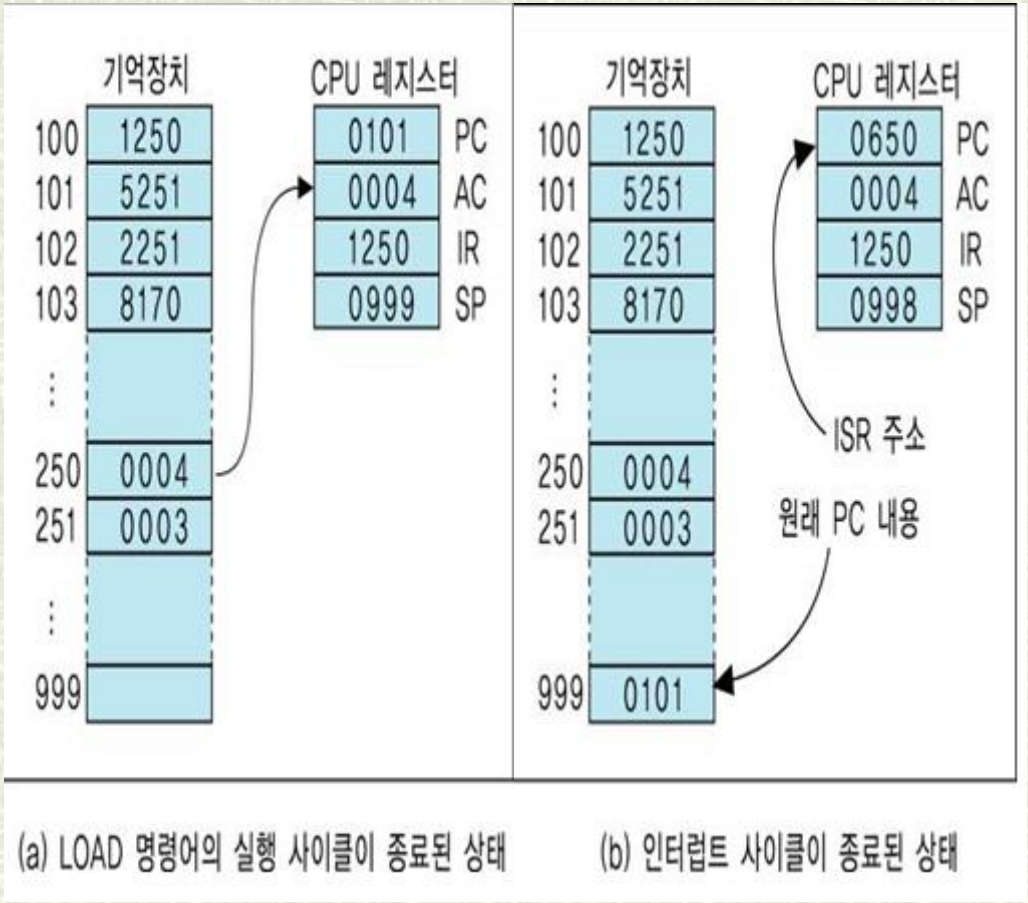


인터럽트 사이클
(LOAD 명령어 수행동안 인터럽트 발생했을 때)

t0 : MBR← PC
t1: MAR ← SP, PC ← ISR시작주소
t2: M[MAR] ← MBR

주소	명령어		기계 코드
100	LOAD	250	1250
101	ADD	251	5251
102	STA	251	2251
103	JUMP	170	8170

ISR 시작주소 :
650번지



다중 인터럽트(multiple interrupt)

- ISR을 수행하는 동안에 다른 인터럽트가 발생하는 것
- 다중 인터럽트의 처리방법

1. CPU가 ISR을 처리하고 있는 도중에는 새로운 인터럽트 요구가 발생해도 CPU가 인터럽트 사이클을 수행하지 않도록 방지

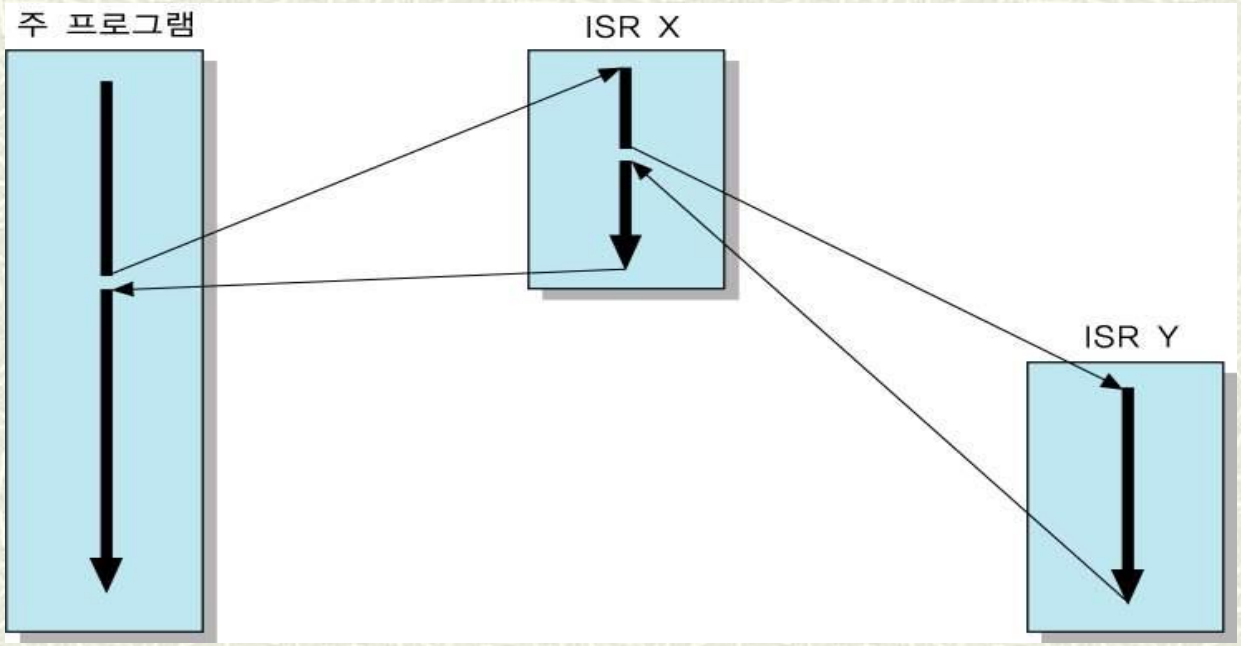
   * interrupt flag = 인터럽트 불가능(interrupt disabled)

     - 시스템 운영상 중요한 프로그램이나 도중에 중단할 수 없는 데이터 입출력 동작 등을 위한 인터럽트를 처리하는데 사용

2. 인터럽트의 우선 순위를 정하고, 우선 순위가 낮은 인터럽트가 처리되고 있는 동안에 우선순위가 더 높은 인터럽트가 들어오면 현재의 인터럽트 서비스 루틴의 수행을 중단하고 새로운 인터럽트를 처리

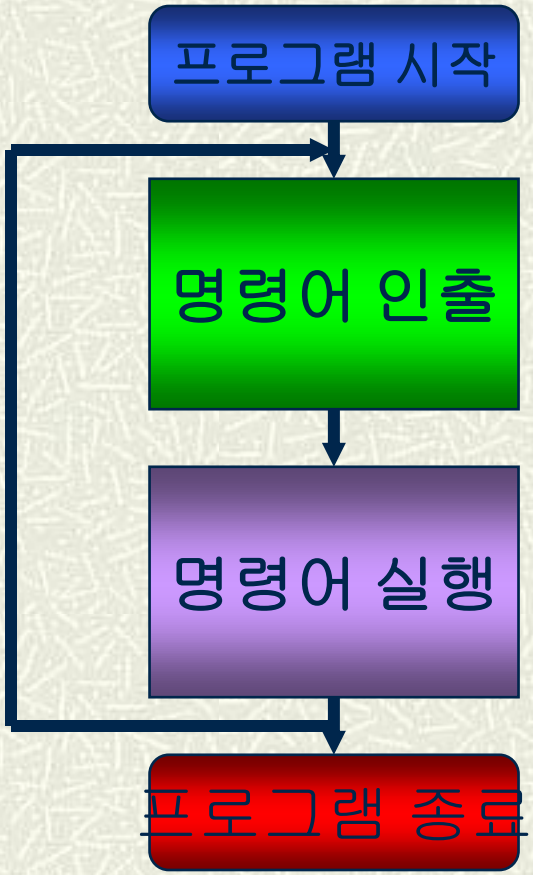
다중 인터럽트 처리

- 장치 X를 위한 ISR X를 처리하는 도중에
  - 우선 순위가 더 높은 장치 Y로부터 인터럽트 요구 발생
  - 우선순위가 높은 것을 먼저 처리되는 경우 제어의 흐름



CPU 처리 속도를 향상 시키는 방법?

(MIPS : Million Instructions Per Sec.)




```
LOAD  X
ADD   Y
STOR  Z
```

Fetch Cycle

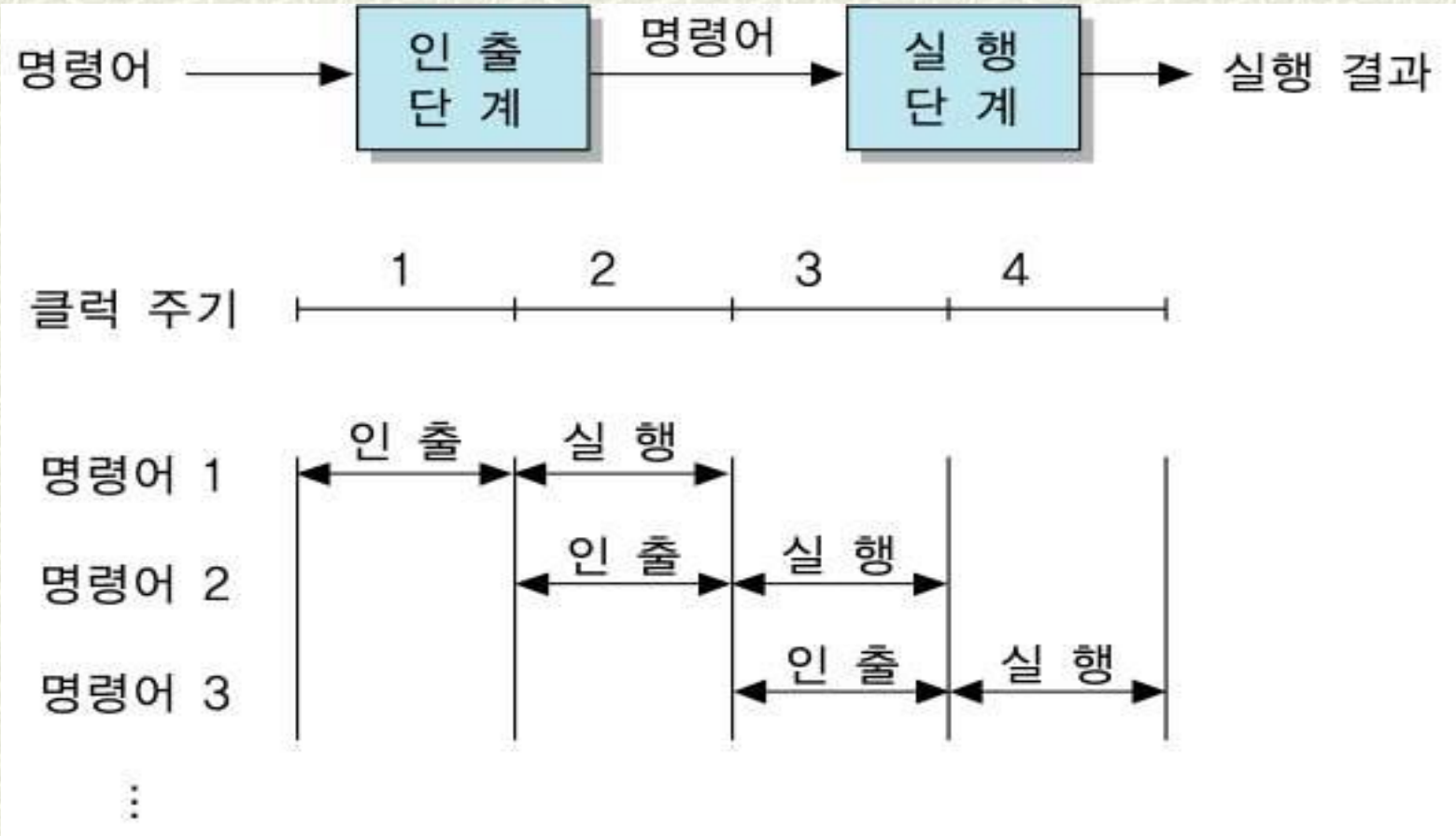
Execution Cycle

Overlap

명령어 파이프라이닝 (Pipelining)

- 명령어 파이프라이닝 이란 ?
  - CPU 속도를 향상 시키는 방법
- 방법
  - 각 사이클(fetch,execution) 처리모듈을 따로 둠
    - 즉, ***fetch stage module***, ***execution stage module***
    - 각 모듈 별로 서로 다른 명령어를 동시에 처리
    - instruction prefetch(overlap)을 통하여 한 주기에 최대 한개의 명령어 처리 완료 가능

2단계 명령어 파이프라인과 시간 흐림도



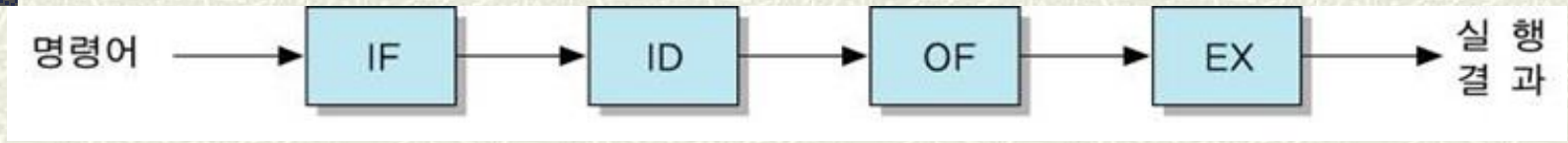
명령어 파이프라이닝 (Pipelining)

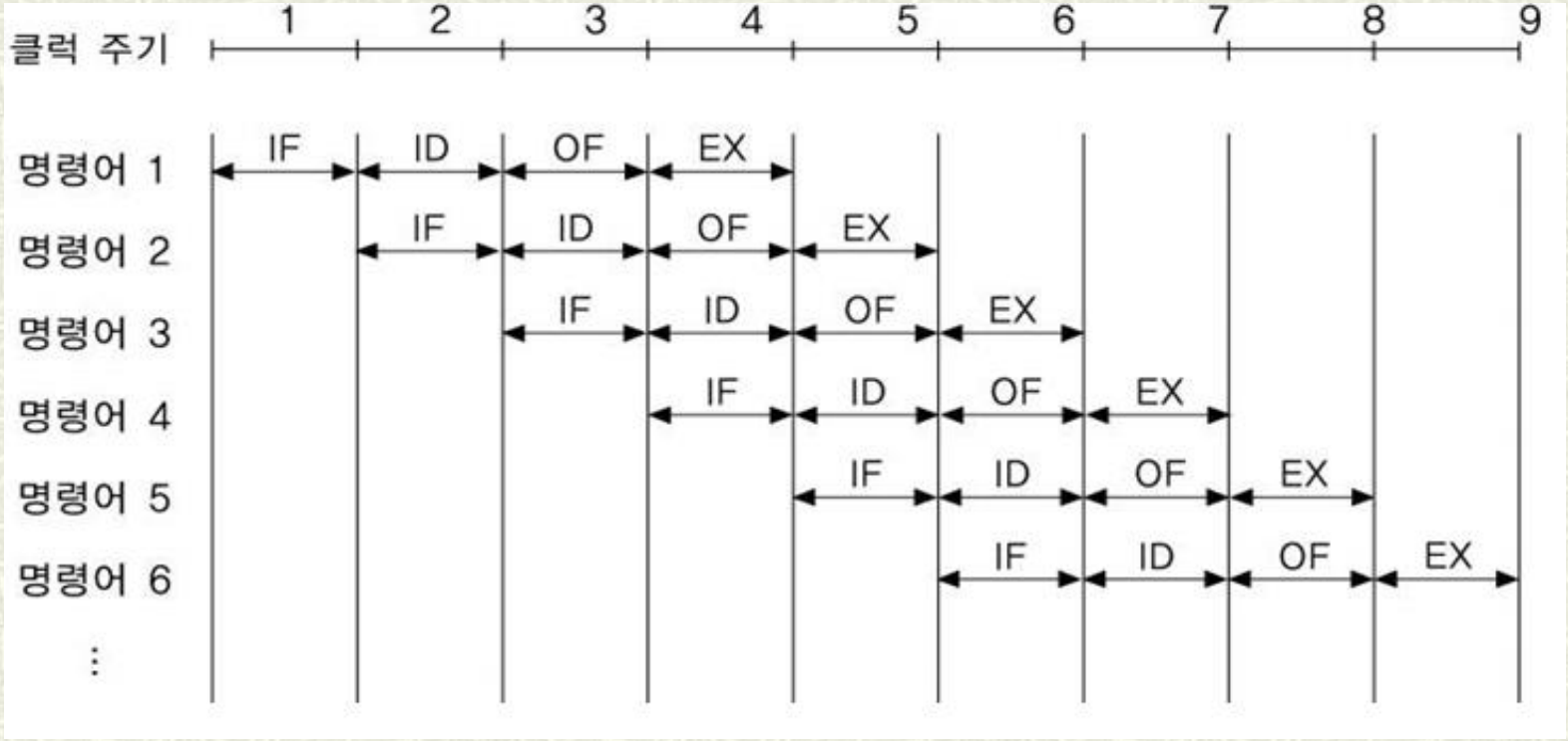
- 문제점
  - 각 모듈 별 처리시간이 다름
    - 실행단계 시간이 인출 단계 보다 오래 걸림
      - 인출 단계는 한 명령어를 인출한 후에 다음 명령어 인출 처리가 즉시 완료가 안됨.
  - 따라서 최대 성능을 발휘 못함.
- 해결책
  - 처리시간이 긴 파이프 라인으로 더 나눈다. *(다음)*

명령어 파이프라이닝 (Pipelining)

- 예)
  - 명령어 인출(IF) :
    - 다음 명령어를 기억장치에서 fetch
  - 명령어 해독(ID) :
    - 해독기(decoder)를 이용하여 명령어를 decode
  - 오퍼랜드 인출(OF) :
    - 기억장치로부터 operand(data)를 읽어 온다.
  - 실행(EX)
    - decoding 명령어를 execution

4단계 명령어 파이프라인 & 시간 흐름도





명령어 파이프라이닝 (Pipelining)

- 성능 계산
  - 클럭 주기를 $1\mu s$ 라고 할때
  - 첫 번째 명령을 수행 완료하는데 걸리는 시간 ?
    - $4\mu s$
  - 그 이후 명령어들은 몇 초마다 실행 될까?
    - $1\mu s$
  - 10개 명령어를 처리할때 전체 수행 시간은 ?
    - 1 x $4\mu s$ + (10-1) x $1\mu s$ = $13\mu s$

명령어 파이프라이닝 (Pipelining)

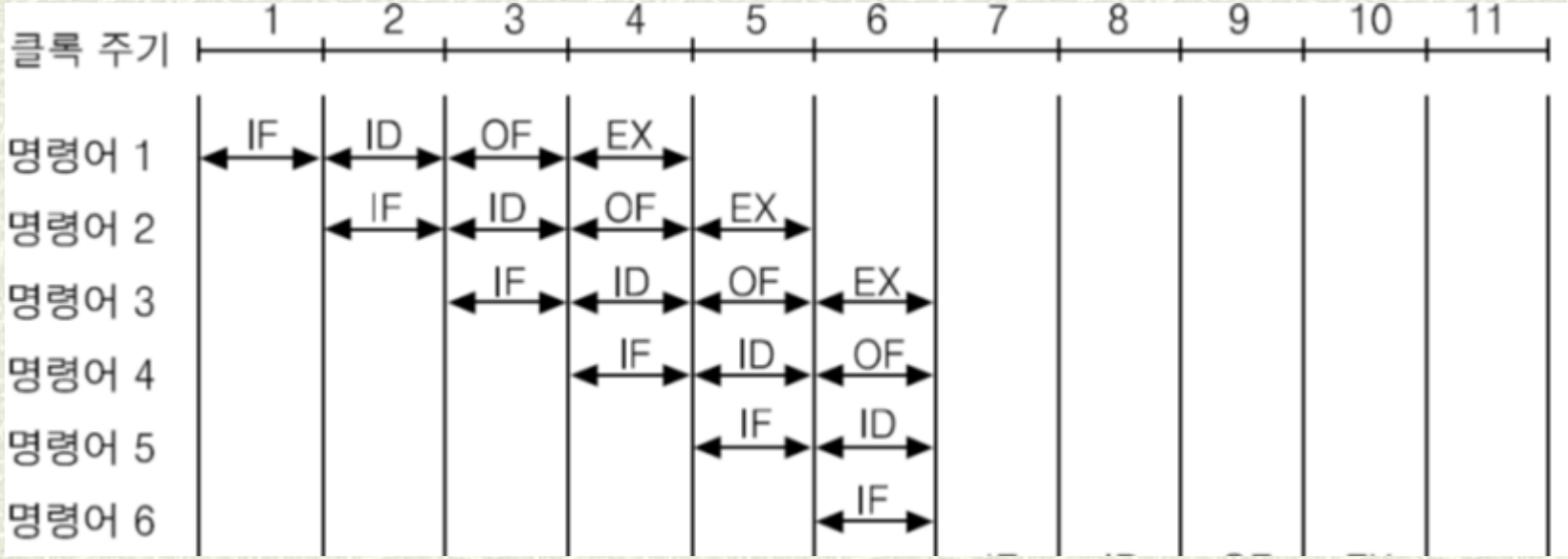
- 성능 계산
  - 파이프라이닝처리가 아닐 경우에 처리 시간?
    - 10 x $4\mu s$
- 성능 향상
  - $\frac{40}{13} \approx 3.08$ 배 성능 향상 (Speed up)

성능척도

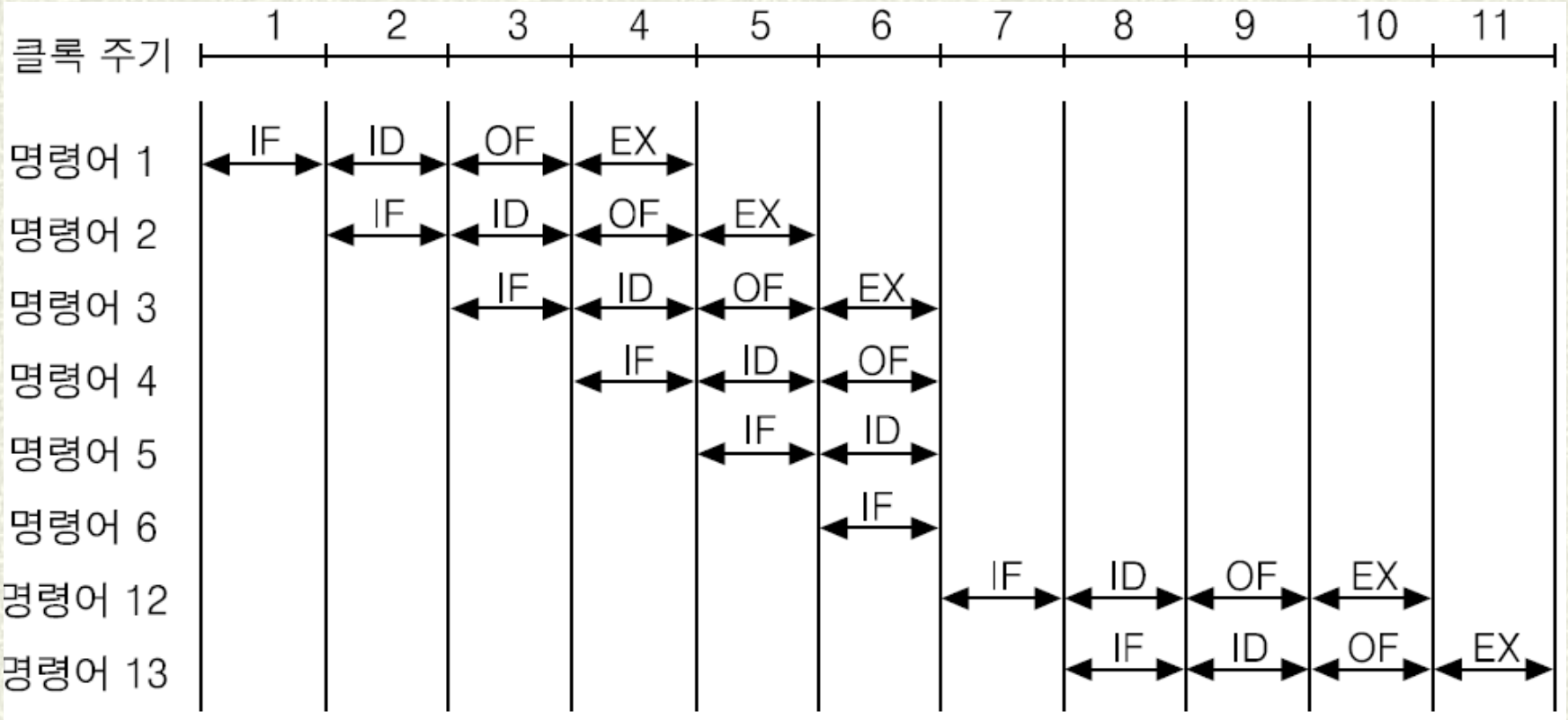
- 성능의 비교로서 동일 작업에 대한 실행 시간의 비(Speedup : 시간 개선도)
  - ***Speedup*** = **기존의 실행 시간 ÷ 개선된 실행시간**
  - *Speedup*은 1보다 큰 경우에 성능이 개선된 것이라고 말할 수 있음.

파이프 라인으로 최대 k배 속도 향상 ? (k는 stage 수)

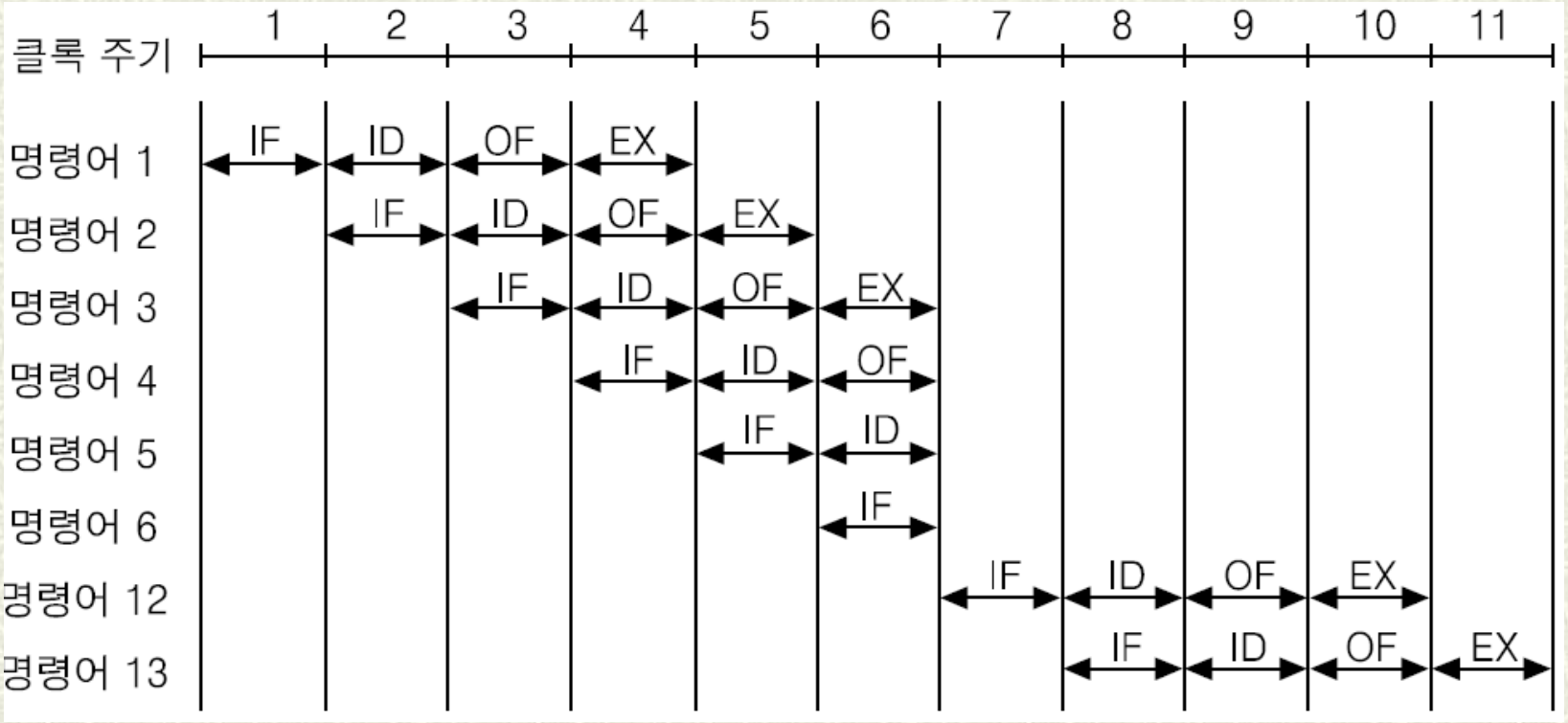
- 장애 요인
  - 조건 분기(Conditional Branch)
    - (예) **명령어3이 명령어12로 분기하라는 명령**



조건 분기 가 존재하는 경우의 시간 흐름도



파이프 라인으로 최대 k배 속도 향상 ? (k는 stage 수)

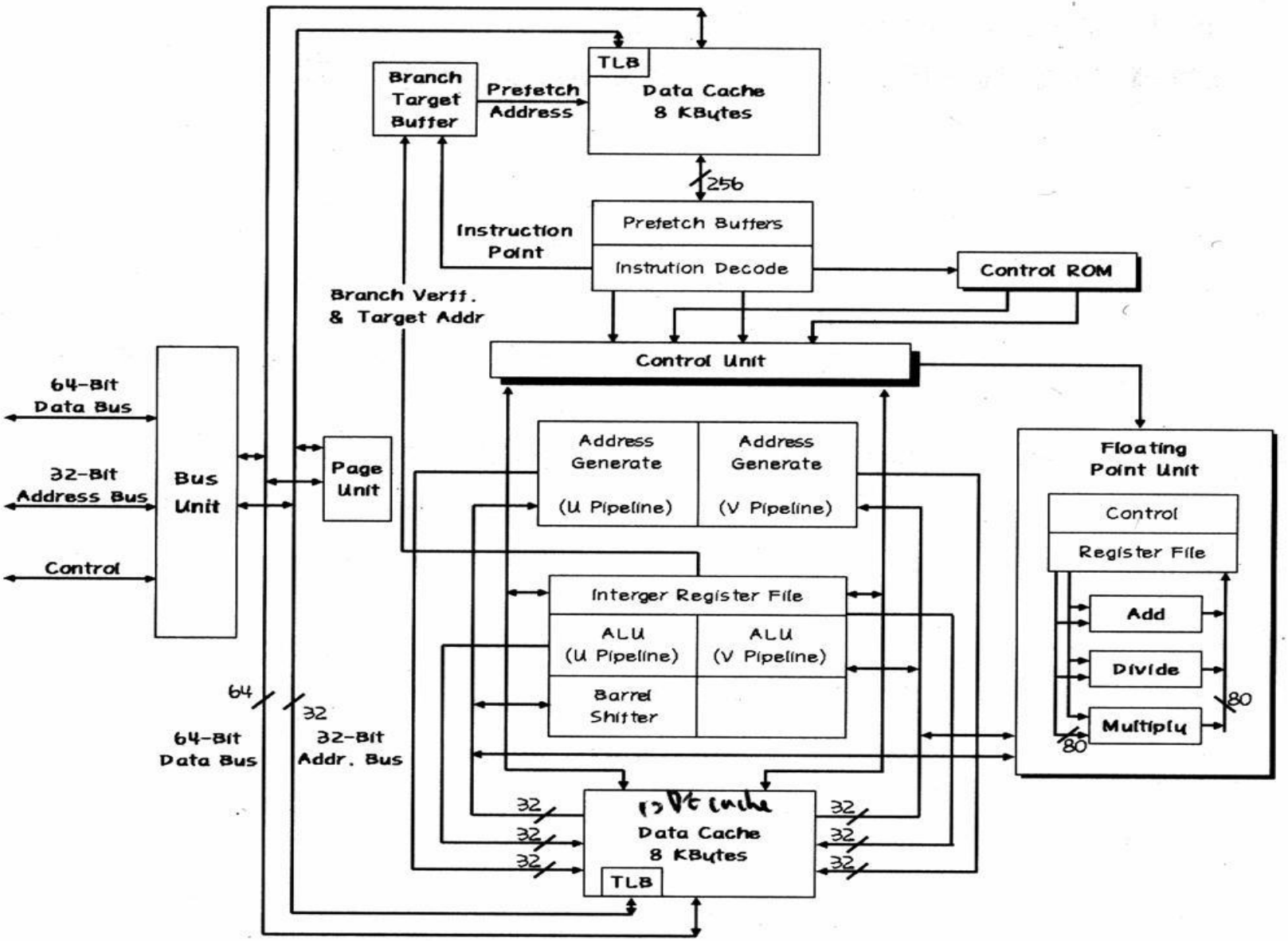


- stage 7-9는 비어 있게 됨.
  - 3개 cycle 낭비 초래

파이프 라인으로 최대 k배 속도 향상 ? (k는 stage 수)

- 성능저하 최소화 방법
  - Prefetch branch target(분기 목적지 선인출)
  - Loop Buffer (루프 버퍼)
  - Branch prediction (분기 예측)
  - Delayed branch(지연 분기)

Pentium Processor 내부 구조



Superscalar

- Superscalar는 CPU 처리 속도를 더욱 높이기 위해 두개 이상의 명령어 파이프라인들을 포함시킨 구조
- 즉, 2-way superscalar는 매주기마다 명령어가 두 개씩 인출되어 동시에 처리된다는 것을 말함.

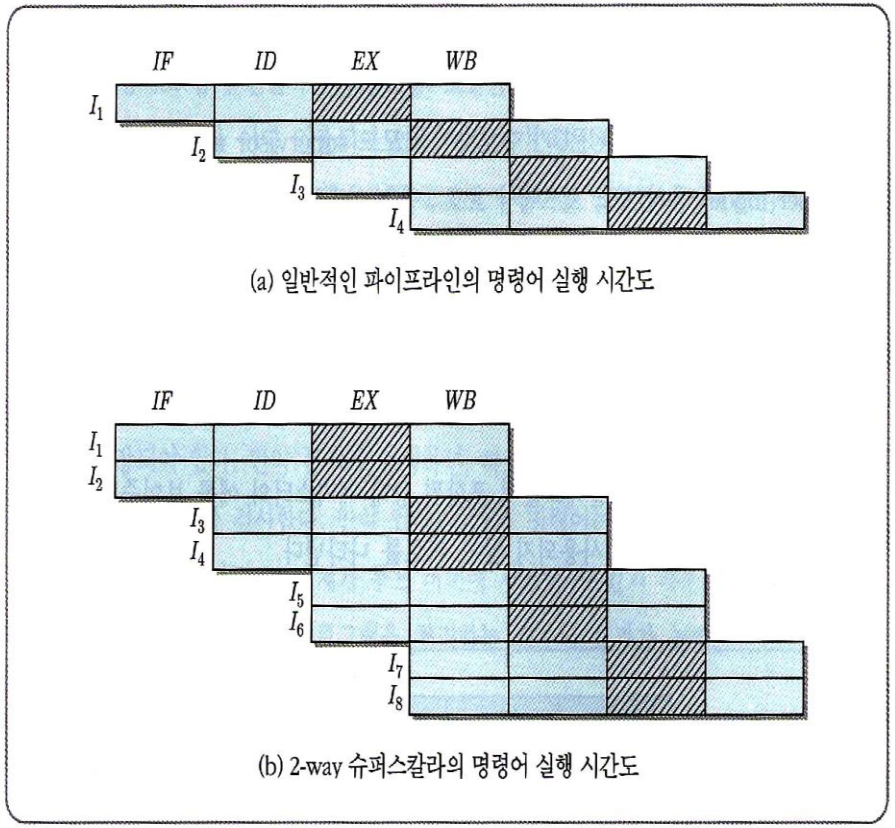
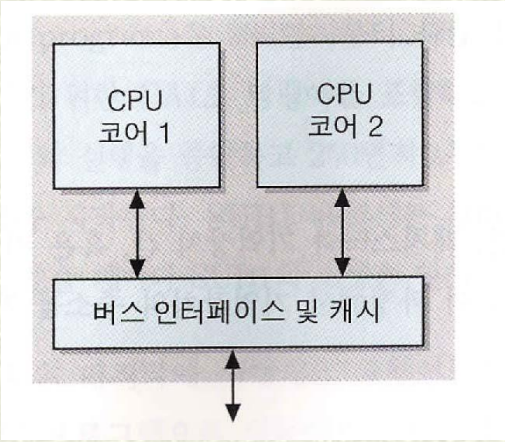


그림 2-15 일반적인 명령어 파이프라인과 슈퍼스칼라 구조의 명령어 실행 흐름도 [JOH91]

Dual-Core & Multi-Core

- CPU Core
  - 명령어 실행에 필요한 핵심 모듈
    - ALU, Register set, Superscalar module등
  - Core 여러개를 하나의 칩에 넣은 CPU
    - Chip-level mutiprocessor (on-chip)
    - Multi-core processor
      - Dual-core, quad-core, hexa-core, octa-core



- 참고
  https://www.seminarstopics.com/seminar/5758/multi-core-processor-seminar-report-pdf