Slide 0

1장. 마이크로프로세서의 구조및 시스템의 개요

마이크로 프로세서의 성능

Slide 1

- 성능은 MIPS (Million Instruction Per Second) 로 나타냄
  $MIPS = \frac{F}{CPI}$ (F: 입력 clock 주파수, CPI: 명령어당 평균 clock 수)
  예) 40MHz, CPI = 2 이면 20 MIPS
- 계산속도로 측정
  - FLOPS (FLoating-point Operations Per Second)
  - 초당 수행하는 부동 소수점 연산 명령
- 일을 수행하는데 걸리는 시간
  프로세서 시간 = $NI * CPI * C$ (NI: 수행 총 명령어, C: 하나의 clock 시간)

마이크로 프로세서의 성능 (Cont'd)

Slide 2

- CISC 와 RISC 의 비교

	RISC 방식	CISC 방식
명령어 형식	단순한 고정길이	복잡한 가변길이
Decoding	hardwired	micro-programmed
레지스터	많은 내부 레지스터 (~ 256 개)	적은 내부 레지스터 (보통 32개)
명령어 분류	대부분이 R-T-R 명령어 (속도 향상)	대부분이 memory 관련 명령어 (속도 늦음)
CPI	보통 1	보통 2 이상
NI	CISC 에 비해 1.8배 내지 2배	
버스	보통 분리된 명령어/데이타 버스 사용 (Harvard 구조)	

→현재는 RISC 와 CISC 의 장점을 복합하는 방향으로 발전

시스템에 관한 문제

Slide 3

- BUS master : BUS 를 구동시킬수 있는 기능을 갖는 device (보통 CPU 및 복잡한 기능을 하는 것들)
- BUS slave : BUS master 의 요구에 의하여 데이터를 주고 받는 device

시스템에 관한 문제 (Cont'd)

Slide 4

- 메모리 장치
  - SRAM (Static RAM) : 캐쉬에 사용 (고속)
  - DRAM (Dynamic RAM) : 주기적으로 데이터 복구 (SRAM 보다 저속)
- 입출력 장치
  - I/O mapped I/O : I/O device 들을 위한 특별한 명령어 사용 (Intel 계열)
  - Memory mapped I/O : memory access 와 같은 명령어 사용 (Motorola 계열)
  - DMA (Direct Memory Access) : 시스템과 외부 장치와의 고 속의 데이터 전송 (Bus master)
- 특수 보조 프로세서
  - FPU (Floating point unit) : 실수 연산
  - 음성, 통신 등의 보조 프로세서

시스템에 관한 문제 (Cont'd)

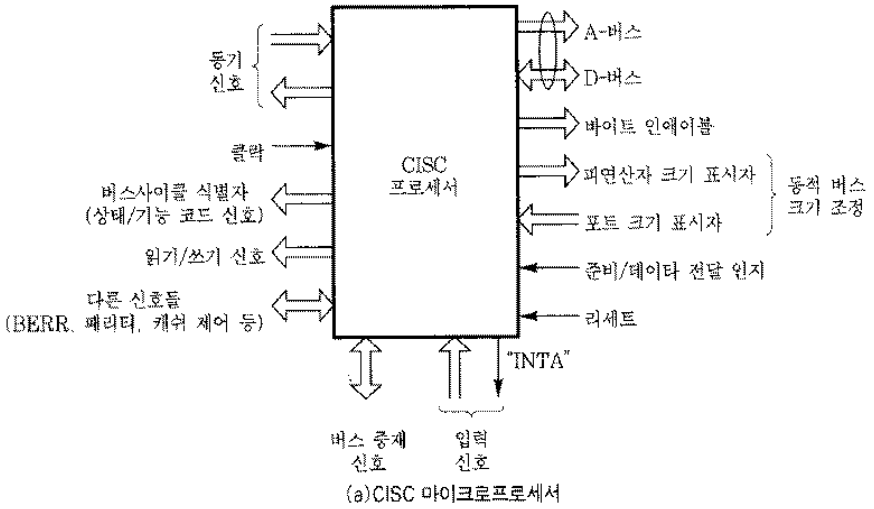
Slide 5

- MMU (Memory Management Unit)
  - logical memory space →physical memory space
- 캐쉬 : 마이크로프로세서와 메모리 사이의 고 속의 데이터 전달을 위한 장치
  - spatial locality 및 temporal locality 응용 한 것.
- 버스의 계층 구조
  - processor bus : FPU, MMU 등이 CPU 와 접속하는 bus
  - local bus : ROM, RAM, I/O devices 등이 접속되는 bus
  - global bus or system bus: 여러개의 보드 사이가 접속되는 bus (multiboard system 에서)
  - bus arbiter : 여러개의 bus master 가 버스를 요청했을 때 중재하는 장치

마이크로프로세서 : 외부구조

Slide 6

- I/O 신호 (그림 1.6)

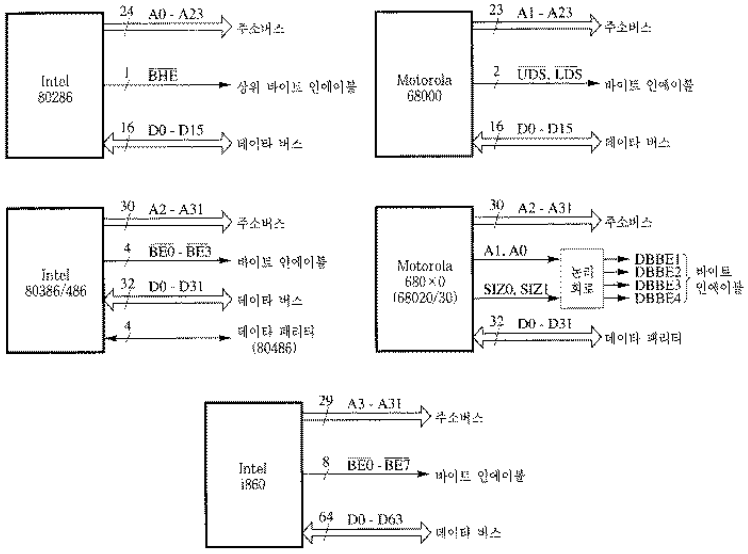


(a)CISC 마이크로프로세서

마이크로프로세서 : 외부구조 (Cont'd)

Slide 7

– 주소 버스 와 데이터 버스 : multiplex 된것과 안된 것이 있음 (그림 1.7)



마이크로프로세서 : 외부구조 (Cont'd)

Slide 8

– 버스 사이클 식별자 : 현재의 버스 사이클의 형태 표시 (그림 1.8)

INTEL 80386(4)

M/IO#	D/C#	W/R#	버스 사이클 형태		잠김(Locked)?
Low	Low	Low	인터럽트 인지		Yes
Low	Low	High	일어나지 않음		—
Low	High	Low	I/O 데이타 읽기		No
Low	High	High	I/O 데이타 쓰기		No
High	Low	Low	메모리 코드 읽기		No
High	Low	High	정지(Halt)	셧다운	No
			주소 = 2	주소 = 0	
			(BE0)# High	(BE0)# Low	
			BE1# High	BE1# High	
			BE2# Low	BE2# High	
			BE3# High	BE3# High	
			A2-A31 Low)	A2-A31 Low)	
High	High	Low	메모리 데이타 읽기	및 사이클	
High	High	High	메모리 데이타 쓰기	및 사이클	

(4) Intel사의 허가를 얻어 게재함 © 1986 Intel Corp.

MOTOROLA 68020/30(8)

FC2	FC1	FC0	주 소 공 간
0	0	0	(미정의, 예약됨)*
0	0	1	사용자 데이타 공간
0	1	0	사용자 프로그램 공간
0	1	1	(미정의, 예약됨)*
1	0	0	(미정의, 예약됨)*
1	0	1	수퍼바이저 데이타 공간
1	1	0	수퍼바이저 프로그램 공간
1	1	1	CPU공간(우측 표 참조)

*주소 공간 3은 사용자 정의를 위해 예약됨.
주소 공간 0과 4는 Motorola의 미래용으로 예약됨.

A19-A16				CPU공간 형태
1	1	1	1	인터럽트 인지
0	0	1	0	보조프로세서 통신
0	0	0	1	액세스 단계 제어 (CALLM, RETM)
0	0	0	0	중지점 인지

(b)버스 사이클 식별자

마이크로프로세서 : 외부구조 (Cont'd)

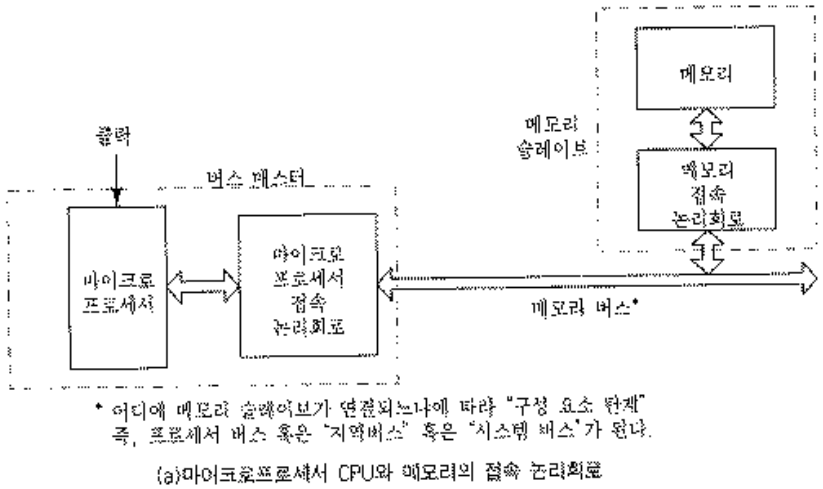
Slide 9

– 동기 신호
  * address strobe →주소가 유호함, data strobe →데이터가 유효함
  * ready (intel 계열), DTACK# (Data Transfer ACknowledge, Motorola 계열) : 버스 사이클의 종료를 알림
  * 피연산자와 포트의 크기 표시
    1. 피연산자 크기 : SIZ0, SIZ1 (Motorola) , BE# (Intel)
    2. 포트의 크기 : DSACK0, DSACK1 (M) , BS16#, BS8# (I)
  * 인터럽트 및 버스 중재 신호
    1. 인터럽트 신호 : 인터럽트 관련된 신호
    2. 버스중재 신호 : 버스 할당과 관련된 신호
  * 다른 제어 신호들
    1. clock 입력 : 모든 동기 신호에 사용
    2. reset 신호 : 프로세서를 초기화 (가장 높은 우선순위의 인터럽트)

마이크로프로세서 : 외부구조 (Cont'd)

- 지역버스 접속 구성 요소 (그림 1.9)
  - 버스를 구동하는 소자 : Open collector 나 tri-state 출력

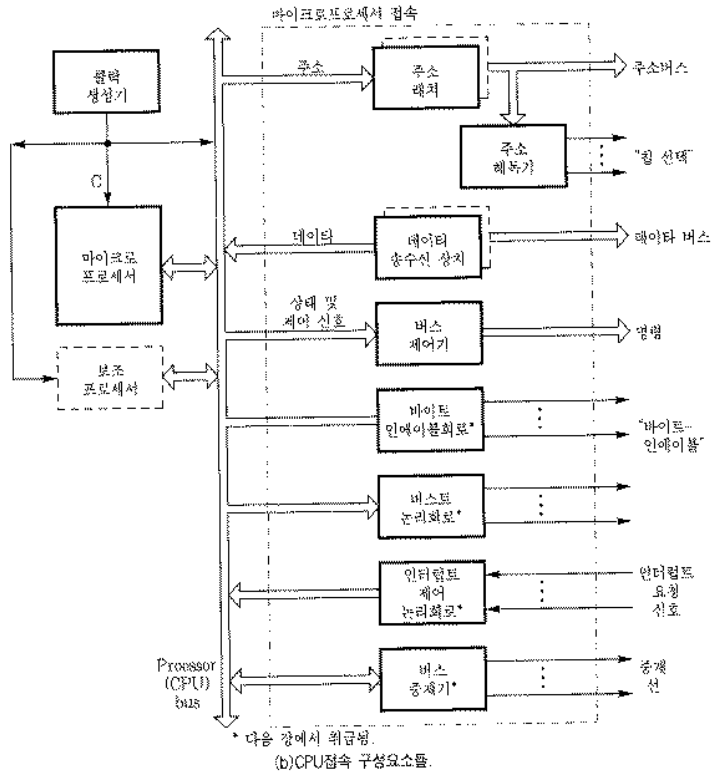
Slide 10



(a)마이크로프로세서 CPU와 메모리의 접속 논리회로

마이크로프로세서 : 외부구조 (Cont'd)

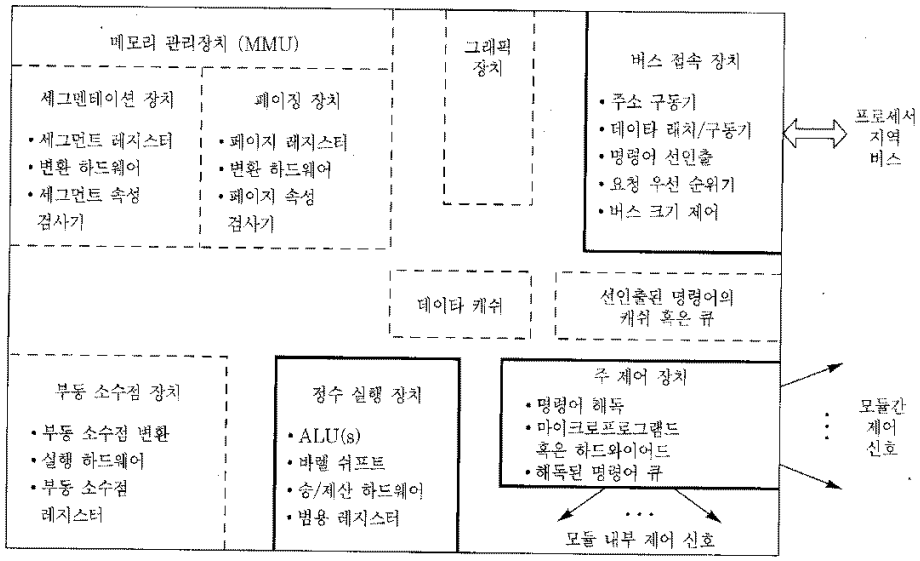
Slide 11



(b)CPU접속 구성요소들.

마이크로프로세서 : 내부구조 (그림 1.10)

Slide 12



시스템 동작

- 버스 사이클, clock 사이클, "상태"
  - 버스 사이클 (그림 1.13)

Slide 13

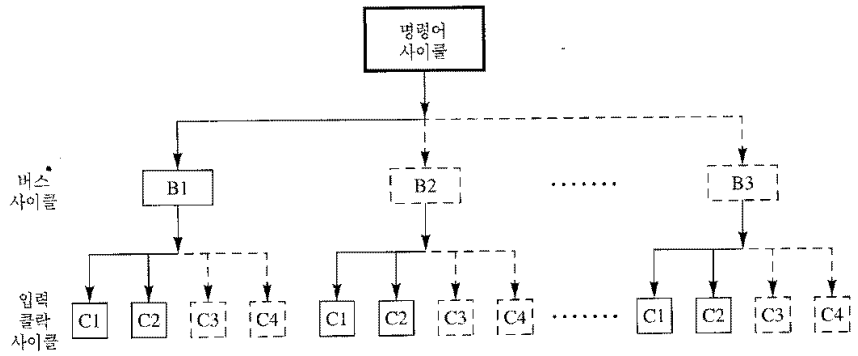


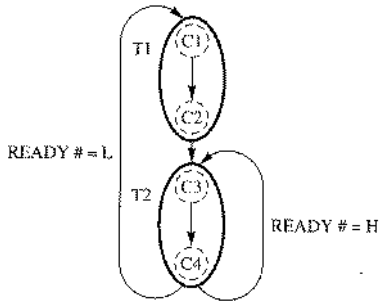
그림 1.13 한 명령어 사이클은 버스 사이클 B1, B2,...(적어도 하나)로 세분된다. 한 버스 사이클은 다시 C1, C2,...의 입력 클락 사이클로 세분된다 (버스 사이클은 마이크로프로세서에 따라 하나 또는 두 개, 세 개 이상의 입력 클락 사이클로 되어 있다).

* 외부 메모리나 I/O 와 데이터 교환시 시작

시스템 동작 (Cont'd)

- 마이크로 동작과 상태 전이도
  - 마이크로 동작 : 명령어 하나 하나를 수행하는데 필요한 동작
    예) 메모리 읽기 마이크로 동작 (그림 1.16)

Slide 14



(Cp = 2C, "T 상태" = 2C)

(a) 80386에서는 클럭 사이클 C를 "CLK2 사이클", 2개의 CLK2 사이클을 "T 상태" 혹은 "CLK 사이클"이라고 한다.

시스템 동작 (Cont'd)

Slide 15

C1: A_{31}-A_2 ← 주소의 일부, M/IO# ← H, BE0#-BE3# ← 적절한 값
W/R# ← L, ADS# ← L
C2:
C3: ADS# ← H. BS16#을 검출한다.(2장에서 설명됨)
C4: READY# = H 이면, 추가의 T_2 상태를 실행한다.
READY# = L 이면, D_{31}-D_0에서 데이타를 읽고 사이클을 마친다.

(b)"메모리 읽기" 마이크로 동작

C1: A_{31}-A_2 ← 주소의 일부, M/IO# ← H, BE0#-BE3# ← 적절한 값
W/R# ← H, ADS# ← L
C2: D_{31}-D_0 ← 데이타 출력
C3: ADS# ← H,
C4: READY# = H 이면, 추가의 T_2 상태를 실행한다.
READY# = L 이면, 사이클을 마친다.

(c)"메모리 쓰기" 마이크로 동작

파이프라인, 수퍼파이프라인, 수퍼스칼라 마이크로 프로세서

Slide 16

- 수행 속도를 빠르게 하는 것 →CPI 를 줄임 →내부 병렬성 이용
- 방법
  - 시간적 병렬성 이용 : 파이프라인 →공통된 하드웨어 →CPI 가 1 이됨
    수퍼파이프라인 : 파이프라인의 깊이 를 증가 →CPI 가 1 보다 작음
  - 공간적 병렬성 이용 : 수퍼스칼라 →분리된 하드웨어 →CPI 가 1 보다 작음
- 파이프 라이닝
  - 한명령어 수행시 마이크로프로세서의 모든 부분이 사용되지 않음
  - 여러 단계로 분할
  - 파이프라인의 단계가 다 차면 →입력속도 == 출력속도
  - n개의 파이프라인 이 다 차면 : n개의 명령을 동시에 수행하는것과 같아짐
  - 하나의 명령어 수행시간 : 변함없음

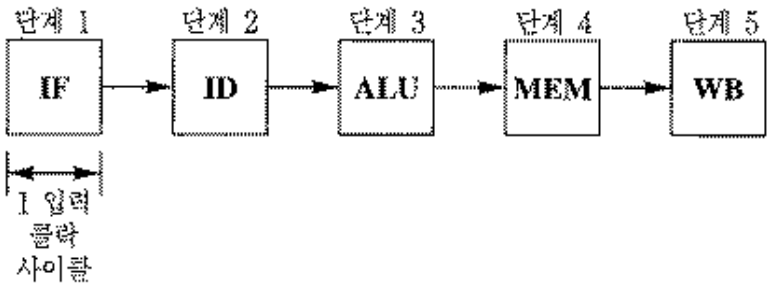
파이프라인, 수퍼파이프라인, 수퍼스칼라 마이크로 프로세서 (Cont'd)

Slide 17

- 5단계의 파이프라인 (그림 1.19)
  1. IF (Instruction Fetch) : 명령어 인출
  2. ID (Instruction Decode) : 명령어 해독
  3. ALU (ALU op.) : ALU 연산
  4. MEM (Memory Acess) : 메모리 적재/저장
  5. WB (Write Back) : 레지스터 쓰기

파이프라인, 수퍼파이프라인, 수퍼스칼라 마이크로 프로세서 (Cont'd)

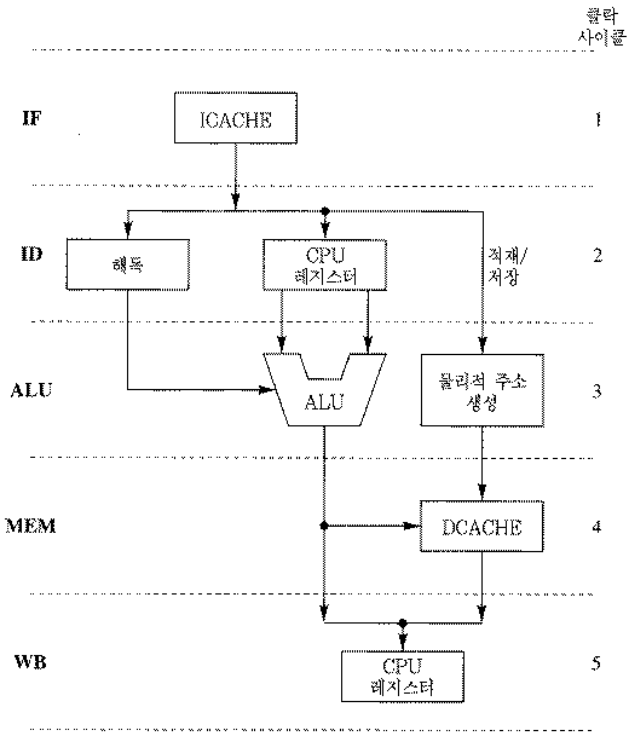
Slide 18



(a)단순화된 5-단계 명령어 파이프라인의 기능적인 표현

파이프라인, 수퍼파이프라인, 수퍼스칼라 마이크로 프로세서 (Cont'd)

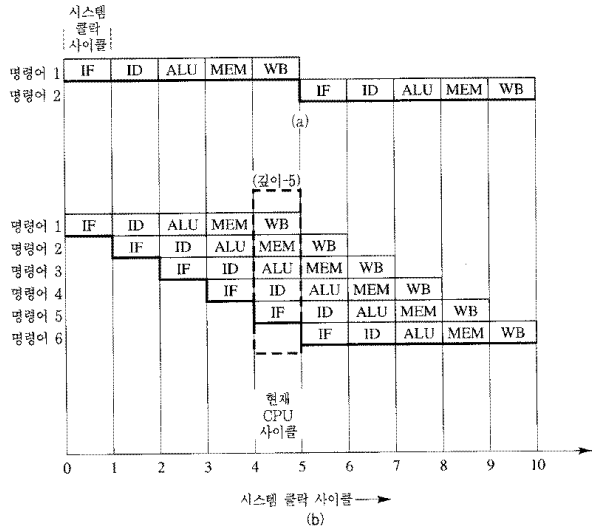
Slide 19



파이프라인, 수퍼파이프라인, 수퍼스칼라 마이크로 프로세서 (Cont'd)

Slide 20

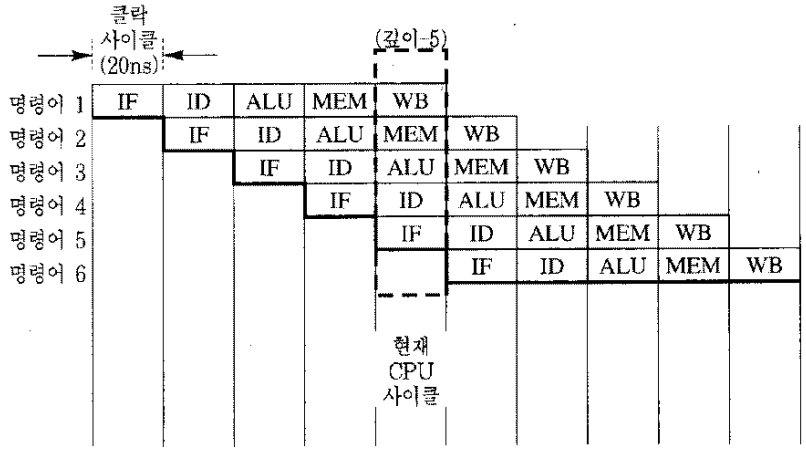
- 조건 분기 명령과 제어 해저드 때문에 이상적인 동작 불가
- 비파이프라인과 파이프라인의 비교 (그림 1.20)



파이프라인, 수퍼파이프라인, 수퍼스칼라 마이크로 프로세서 (Cont'd)

Slide 21

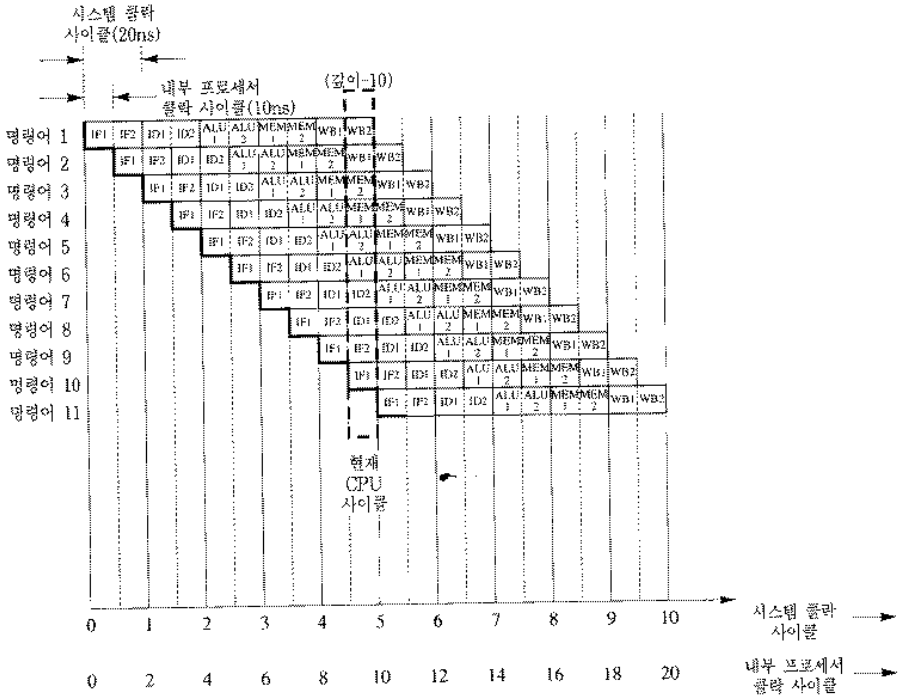
- 수퍼 파이프라인
  - 파이프라인의 단계를 늘림 (그림 1.21)



(a)5-단계 파이프라인:시스템 클락 = 내부 프로세서 클락 = 50MHz,
각 단계는 20ns 걸림, 20ns마다 한 명령어 나옴;이상적으로 CPI = 1

파이프라인, 수퍼파이프라인, 수퍼스칼라 마이크로 프로세서 (Cont'd)

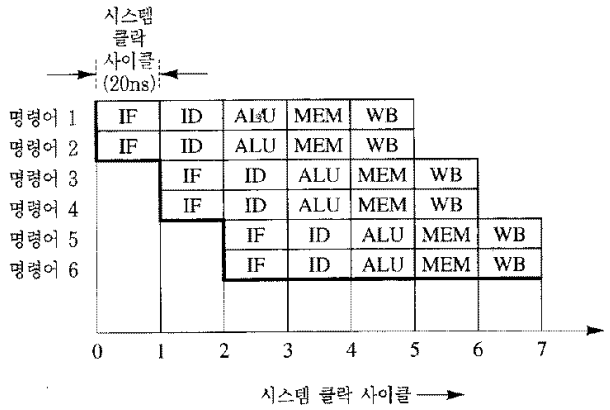
Slide 22



파이프라인, 수퍼파이프라인, 수퍼스칼라 마이크로 프로세서 (Cont'd)

- 수퍼 스칼라
  - 한사이클에 하나 이상의 명령어 인출 하여 여러개의 장치에서 동시에 수행
  - 88110 의 경우 : 두 개의 명령어 인출, 동시 해독, 두 개의 분리된 장치에서 실행됨 (그림 1.22)

Slide 23



파이프라인, 수퍼파이프라인, 수퍼스칼라 마이크로 프로세서 (Cont'd)

Slide 24

- 수퍼 파이프라인과 수퍼 스칼라 비교
  - 수퍼 파이프라인 : 내부 단계를 더욱 세분 (하드웨어 비교적 간단, 어느이상 세분 효과없음)
  - 수퍼 스칼라 : 분리된 장치를 둠 (하드웨어 복잡)
  - CPI 는 이상적으로 둘다 0.5