주제/단락	내용
명령어 파이프라이닝	CPU 내부 하드웨어를 여러 단계로 분할해 서로 겹치게 처리하여 프로그램 실행 속도를 높이는 기술이다.
2단계 파이프라인 구조	명령어 인출 단계와 실행 단계로 분리해 독립 파이프라인으로 동작시킨다.
2단계 동작 타이밍	클럭을 맞추면 1주기에는 첫 명령어 인출, 2주기에는 첫 명령어 실행과 동시에 두 번째 명령어 선인출이 이뤄진다.
초기 속도 향상 예시	6개 작업을 4주기 처리로 환산해 속도 향상은 1.5배이며, 명령어 수가 늘수록 2배에 근접한다.
단계 시간 불일치 문제	두 단계의 처리 시간이 다르면 파이프라인 이득이 줄어든다.
단계 세분화 해결	단계를 더 잘게 나눠 각 단계의 처리 시간을 비슷하게 맞추면 전체 속도 향상이 커진다.
IF 단계	명령어를 기억장치에서 인출하는 단계이다.
ID 단계	제어 유니트의 해독기를 사용해 명령어를 해독하는 단계이다.
OF 단계	연산에 필요한 오퍼랜드를 기억장치에서 인출하는 단계이다.
EX 단계	지정된 연산을 수행하고 결과를 저장하는 단계이다.
파이프라인 총 시간 식	단계 수 k, 명령어 수 N, 단계당 1주기일 때 총시간 $Tk = k + (N - 1)$이다.
비파이프라인 시간 식	파이프라인을 쓰지 않으면 총시간 $T1 = k \times N$이다.
파이프라인 속도 향상 식	$Sp = T1/Tk = (k \times N) / (k + N - 1)$로 표현된다.
4단계 1GHz 예제	첫 명령어 4ns 후 완료, 이후 매 1ns마다 완료되어 10개 처리 시간은 13ns, 속도 향상은 약 3.08배이다.
N 증가에 따른 한계	N=100일 때 약 3.88배, N=1000일 때 약 3.988배, N=10000일 때 약 3.9988배로 $N \to \infty$이면 이론상 4배이다.
이론적 상한 일반화	K단계 파이프라이닝의 이론적 성능 향상은 실행 시간 기준 최대 K배이다.
단계 통과 일관성	일부 명령은 오퍼랜드 인출이 불필요해도 하드웨어 단순화를 위해 모든 명령이 동일 단계들을 통과하도록 설계한다.
파이프라인 클럭 결정	가장 오래 걸리는 단계 시간을 기준으로 클럭이 정해진다.
느린 단계 예시	IF, ID, OF가 1ns여도 EX가 1.5ns면 각 단계 처리 시간은 1.5ns로 맞춰야 한다.

주제/단락	내용
슈퍼 파이프라이닝	단계를 더 세분화해 한 클럭 내 여러 미세단계를 수행함으로써 처리 속도를 높이는 기법이다.
파이프라인 장애 분류	구조적 장애, 데이터 장애, 제어 장애의 세 종류가 존재한다.
구조적 장애 정의	서로 다른 단계의 명령이 동시에 같은 하드웨어 자원을 사용하려 할 때 발생한다.
구조적 장애 예시	IF와 OF가 동시에 기억장치 접근 시 메모리 충돌로 지연이 생긴다.
구조적 장애 해결	하버드 구조 채택 또는 명령어 캐시와 데이터 캐시를 분리해 충돌을 줄인다.
데이터 장애 정의	한 명령의 결과가 다음 명령의 입력으로 필요할 때 결과 준비 전 접근으로 인해 지연이 생긴다.
데이터 의존성 예시	앞선 ALU 결과가 레지스터에 기록되기 전에 다음 명령이 그 값을 요구하면 병목이 발생한다.
데이터 장애 해결 1	파이프라인 스톨을 삽입해 1~2단계 지연시키는 방법이 있다.
데이터 장애 해결 2	명령어 재배치로 실행 순서를 바꿔 의존성을 회피한다.
데이터 장애 해결 3	레지스터 기록 전에 ALU 결과를 다음 명령으로 직접 전달하는 데이터 포워딩을 사용한다.
포워딩 예시 설명	ADD의 EX 결과가 완료되기 전에 SUB가 같은 레지스터 값을 필요로 하면 EX 결과를 바로 전달해 지연을 줄인다.
제어 장애 정의	조건 분기 수행 시 미리 인출해 둔 명령어들이 무효화되어 성능 손실이 발생한다.
제어 장애 예시	조건 충족 시 특정 주소로 점프하는 분기에서 잘못 인출된 명령들이 폐기된다.
분기 예측	분기 발생 여부를 예측해 그에 맞춰 인출하며, 분기 역사 표를 이용해 최근 결과를 참고한다.
분기 목적지 선인출	분기 다음 명령과 분기 목표 명령을 함께 인출해 조건 결과에 따라 유효 경로를 선택한다.
루프 버퍼	최근 인출된 n개 명령을 저장하는 소형 고속 메모리를 IF 단계에 두어 반복문 성능을 높인다.
지연 분기	분기 명령 위치를 재배치해 파이프라인 버블을 줄이는 방법이다.
슈퍼스칼라 정의	하나의 프로세서에 두 개 이상 명령어 파이프라인을 포함해 동시 다중 인출·실행을 수행하는 구조다.
슈퍼스칼라 동작	매 클럭마다 각 파이프라인이 서로 다른 명령을 인출·실행해 이론적으로 파이프라인 수만큼 속도가 향상된다.

주제/단락	내용
m-way 표기	파이프라인의 수가 m이면 m-way 슈퍼스칼라라 한다.
단일 파이프라인 시간	실행할 명령어 수 N일 때 T1 = k + N − 1이다.
m-way 실행 시간	Tm = k + (N − m)/m로 표현된다.
슈퍼스칼라 속도 향상	Sp = T(1)/T(m) = (k+N−1) / (k + (N−m)/m) = m(k+N−1) / (N + m(k−1))이다.
명령어 수 무한대 한계	N이 매우 크면 Sp는 m에 수렴해 이론상 m배 속도 향상이 가능하다.
데이터 의존성 영향	명령 간 결과 전달 필요로 인해 동시 실행이 제한되면 이상적 m배 달성이 어렵다.
자원 경합 영향	레지스터, 캐시, 메모리 등의 하드웨어 경합으로 동시 실행 가능 명령 수가 m보다 작아질 수 있다.
성능 개선 전략 1	실행 순서 재배치로 데이터 의존성을 줄인다.
성능 개선 전략 2	ALU, 레지스터, 캐시 등 하드웨어를 중복 추가해 자원 경합을 감소시킨다.
동적 실행 정의	컴파일 순서를 따르지 않고 데이터 의존성을 고려해 실행 순서를 재배치해 유휴 사이클을 줄이는 기법이다.
동적 실행 단계 1	Fetch 단계에서 원래 순서대로 명령어를 인출한다.
동적 실행 단계 2	Decode 단계에서 해독 정보를 명령어 풀에 저장한다.
동적 실행 단계 3	의존성 검사로 동시에 실행 불가한 연속 명령을 파악하고 독립 명령을 선택한다.
동적 실행 단계 4	Dispatch에서 동시 처리 가능한 명령을 ALU나 Load/Store 유닛에 배정한다.
동적 실행 단계 5	Retire 단계에서 프로그램 순서대로 결과를 확정해 레지스터와 메모리에 반영한다.
재배치 실행 예	I1과 I2가 의존이면 I1과 I3, 이어서 I2와 I4를 묶어 동시 실행하고, 완료된 명령은 순서대로 확정한다.
CPU 코어 정의	명령어 실행에 필요한 슈퍼스칼라 모듈, ALU, FPU, 레지스터, 제어장치, 온칩 캐시 등을 포함한 핵심 하드웨어 묶음이다.
멀티코어 정의	여러 개의 CPU 코어를 하나의 칩에 포함한 프로세서로 칩 레벨 다중프로세서라고도 한다.

주제/단락	내용
코어 수 명칭	두 개는 듀얼 코어, 네 개는 쿼드 코어, 여섯 개는 헥사 코어, 여덟 개는 옥타 코어라 한다.
듀얼 코어 특성	단일 코어 슈퍼스칼라 대비 이론상 최대 2배 성능 향상이 가능하며 내부 캐시와 버스 인터페이스만 공유한다.
듀얼 코어 실행	각 코어가 독립 프로그램을 실행해 멀티태스킹과 멀티스레딩을 지원한다.
멀티태스킹 정의	여러 코어를 사용해 독립 프로세스들을 동시에 실행하는 방식이다.
멀티스레딩 정의	하나의 코어가 여러 스레드를 동시에 실행해 동시성을 높이는 방식이다.
프로세스와 스레드	프로세스는 자원 할당의 독립 단위이고 스레드는 독립 실행 가능한 최소 단위로 프로세스 자원을 공유한다.
단일 스레드 모델	각 코어가 하나의 스레드를 처리하며 RS에 PC, SP, 상태 레지스터 등 실행 상태를 저장한다.
멀티스레드 모델	각 코어가 두 개의 RS를 보유해 최대 두 스레드를 동시에 처리하며 ALU, FPU, 캐시를 공유한다.
상태 분리	스레드별 PC, SP, 상태 레지스터는 각자의 RS에 저장되어 독립성이 보장된다.
듀얼코어 멀티스레드 결과	두 물리 코어가 네 개의 논리 프로세서로 보이며 동시 처리 스레드 수가 증가한다.
혼합형 멀티코어 정의	성능과 전력 효율이 다른 코어를 한 칩에 함께 배치해 작업 특성에 따라 적합한 코어가 처리한다.
혼합형 활용	저전력이 중요한 랩탑·스마트폰에서 단순 작업과 복잡 작업이 혼재된 환경을 효율적으로 처리한다.
혼합형 예시	고성능 P-코어는 복잡 작업과 멀티스레딩을 담당하고, 저전력 E-코어는 작은 작업과 백그라운드 처리를 맡으며 멀티스레딩은 미지원이다.