1장

컴퓨터 추상화 및 기술

컴퓨터 혁명

- 컴퓨터 기술의 발전
  - 도메인별 액셀러레이터로 지원
- 새로운 애플리케이션 실현 가능
  - 자동차의 컴퓨터
  - 휴대폰

- 인간 게놈 프로젝트
- 월드 와이드 웹
- 검색 엔진

- 컴퓨터가 널리 보급되어 있습니다.

컴퓨터 클래스

- 개인용 컴퓨터
  - 범용, 다양한 소프트웨어
  - 비용/성능 트레이드오프에 따라 달라질 수 있습니다
    .
- 서버 컴퓨터
  - 네트워크 기반

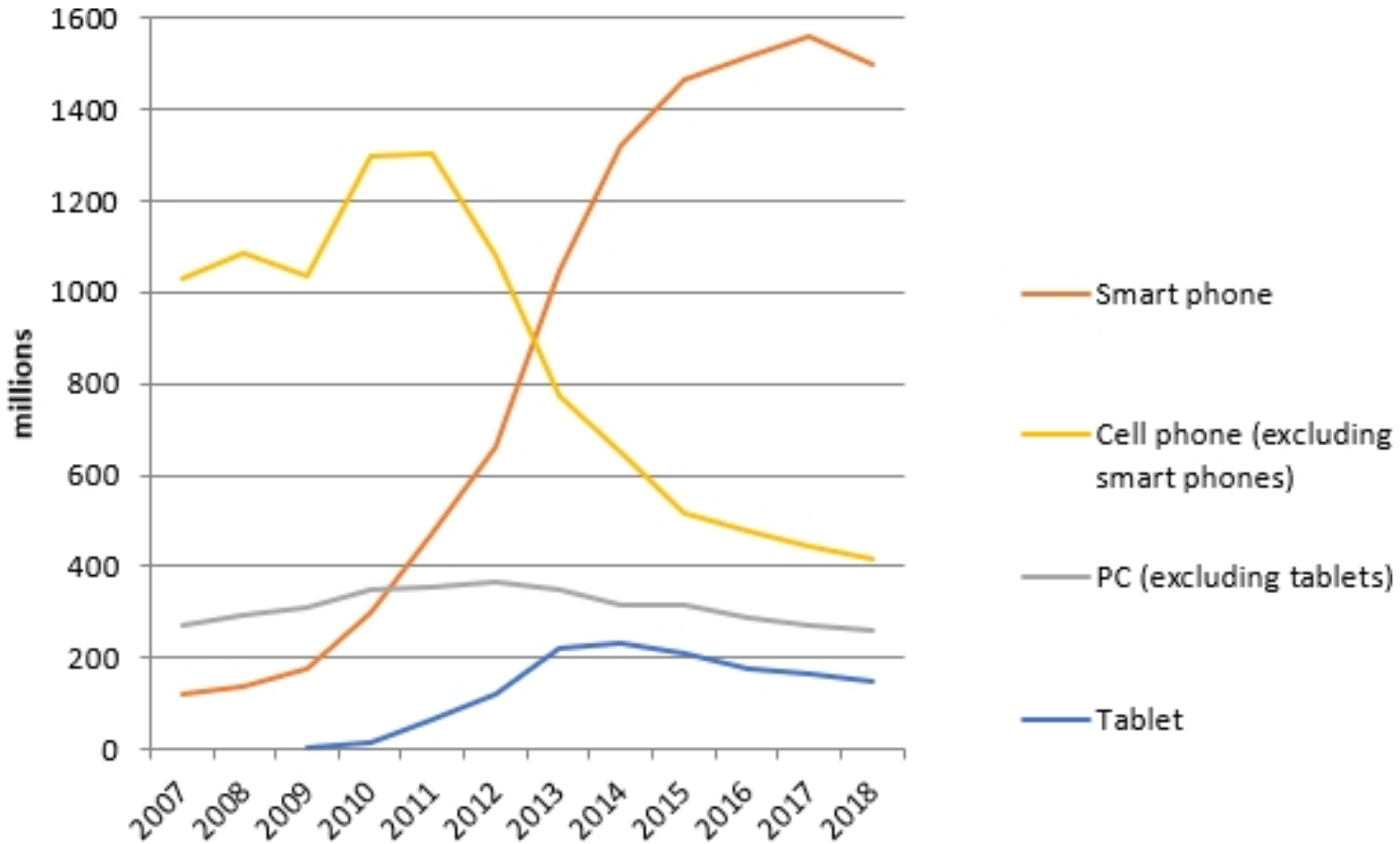
- 고용량, 성능, 안정성
- 소규모 서버부터 빌딩 규모의 서버까지

컴퓨터 클래스

- 슈퍼컴퓨터
  - 서버 유형
  - 고급 과학 및 엔지니어링 계산
  - 최고의 성능을 자랑하지만 전체 컴퓨터 시장에서 차지하는 비중은 극히 일부에 불과합니다.

- 임베디드 컴퓨터
  - 시스템의 구성 요소로 숨겨짐
  - 엄격한 전력/성능/비용 제약 조건

포스트 PC 시대


1600
1400
1200
1000
800
600
400
200
0
millions
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
Smart phone
Cell phone (excluding smart phones)
PC (excluding tablets)
Tablet

포스트 PC 시대

- 개인 모바일 장치(PMD)
  - 배터리로 작동
  - 인터넷에 연결
  - 수백 달러
  - 스마트폰, 태블릿, 전자 안경
- 클라우드 컴퓨팅

- 창고 규모 컴퓨터(WSC)
- 서비스형 소프트웨어(SaaS)
- 일부 소프트웨어는 PMD에서 실행되고 일부는 클라우드에서 실행됩니다.
- Amazon 및 Google

학습 내용

- 프로그램이 기계어로 번역되는 방법
  - 그리고 하드웨어가 이를 실행하는 방법
- 하드웨어/소프트웨어 인터페이스
- 프로그램 성능을 결정하는 요소
  - 그리고 이를 개선할 수 있는 방법

- 하드웨어 설계자가 성능을 개선하는 방법
- 병렬 처리란 무엇인가요?

성능 이해

- 알고리즘
  - 실행된 작업 수를 결정합니다.
- 프로그래밍 언어, 컴파일러, 아키텍처
  - 작업당 실행되는 기계 명령어 수 확인
- 프로세서 및 메모리 시스템
  - 명령 실행 속도 결정하기

- I/O 시스템(OS 포함)
  - I/O 작업이 실행되는 속도를 결정합니다.

7가지 훌륭한 아이디어

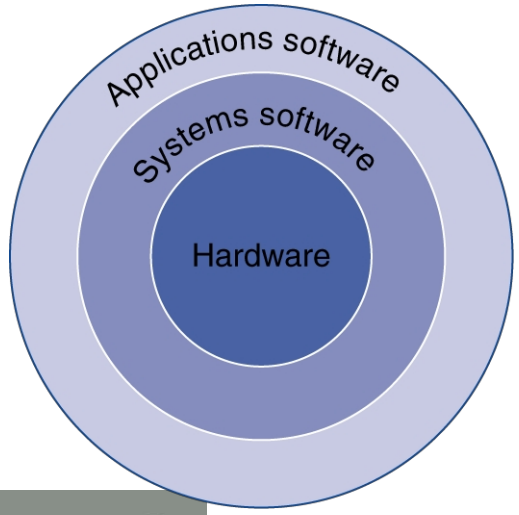
- ***추상화를*** 사용하여 디자인 간소화
- ***일반 케이스를 빠르게*** 만들기
- ***병렬 처리를*** *통한* 성능
- ***파이프라이닝을*** *통한* 성능 향상
- ***예측을*** *통한* 성능 향상

- 기억의 ***계층 구조***

- 이중화를 *통한* ***신뢰성***

프로그램 아래

- 애플리케이션 소프트웨어
  - 고급 언어로 작성
- 시스템 소프트웨어
  - 컴파일러: HLL 코드를 머신 코드로 변환합니다.
  - 운영 체제: 서비스 코드
    - 입력/출력 처리

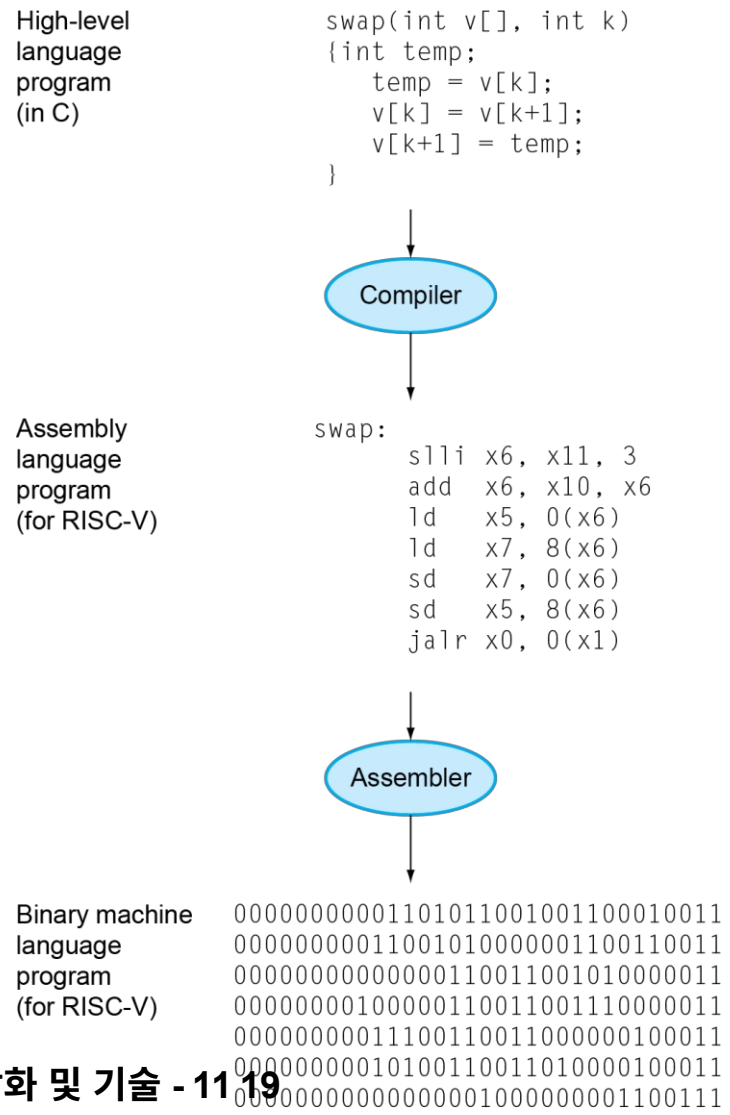


- 메모리 및 스토리지 관리
- 작업 예약 및 리소스 공유

- 하드웨어
  - 프로세서, 메모리, I/O 컨트롤러

프로그램 코드 수준

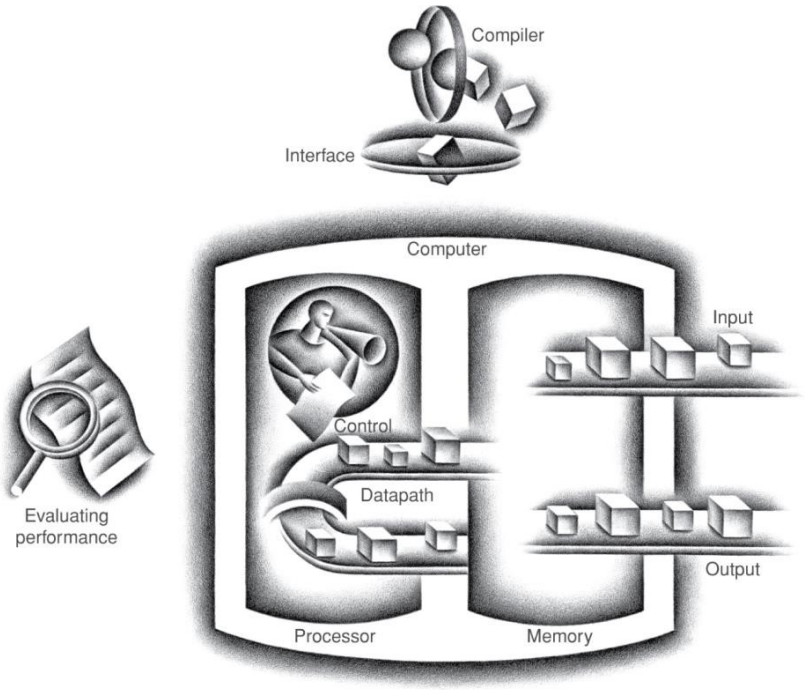
- 고급 언어
  - 추상화 수준 근접 문제 도메인에
  - 생산성 및 휴대성 제공
- 어셈블리 언어
  - 지침의 텍스트 표현
- 하드웨어 표현
  - 이진 숫자(비트)



- 인코딩된 지침 및 데이터

컴퓨터의 구성 요소

큰 그림



- 모든 종류의 컴퓨터에서 동일한 구성 요소: 프로세서, 메모리, I/O
  - 데스크톱, 서버, 임베디드
- 입력/출력에는 다음이 포함됩니다.

- 사용자 인터페이스 장치
  - 디스플레이, 키보드, 마우스
- 저장 장치
  - 하드 디스크, CD/DVD, 플래시
- 네트워크 어댑터
  - 다른 컴퓨터와 통신하는 경우

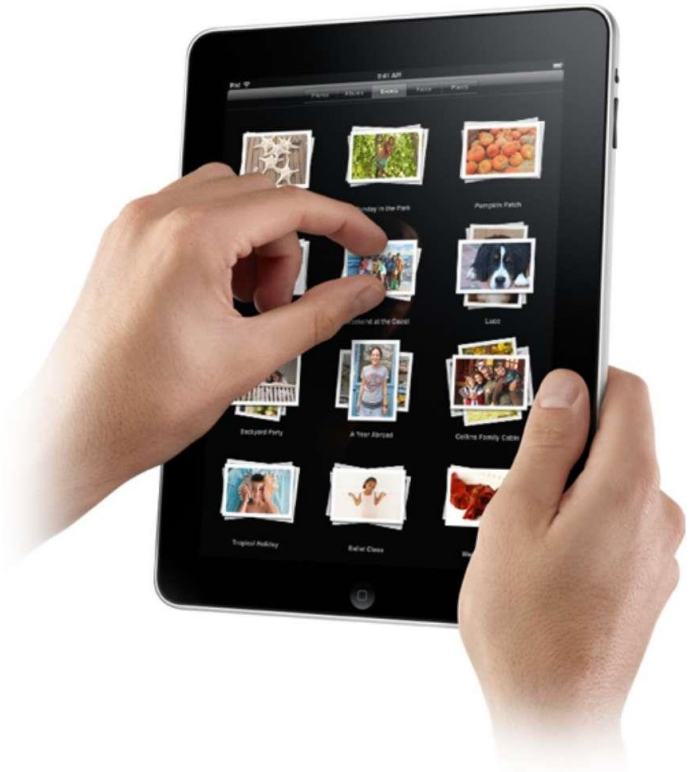
터치스크린

- 포스트PC 장치
- 키보드 및 마우스 대체
- 저항성 및 정전 용

  량 유형

  - 대부분의 태블릿, 스마

    트폰은 정전식을 사용합

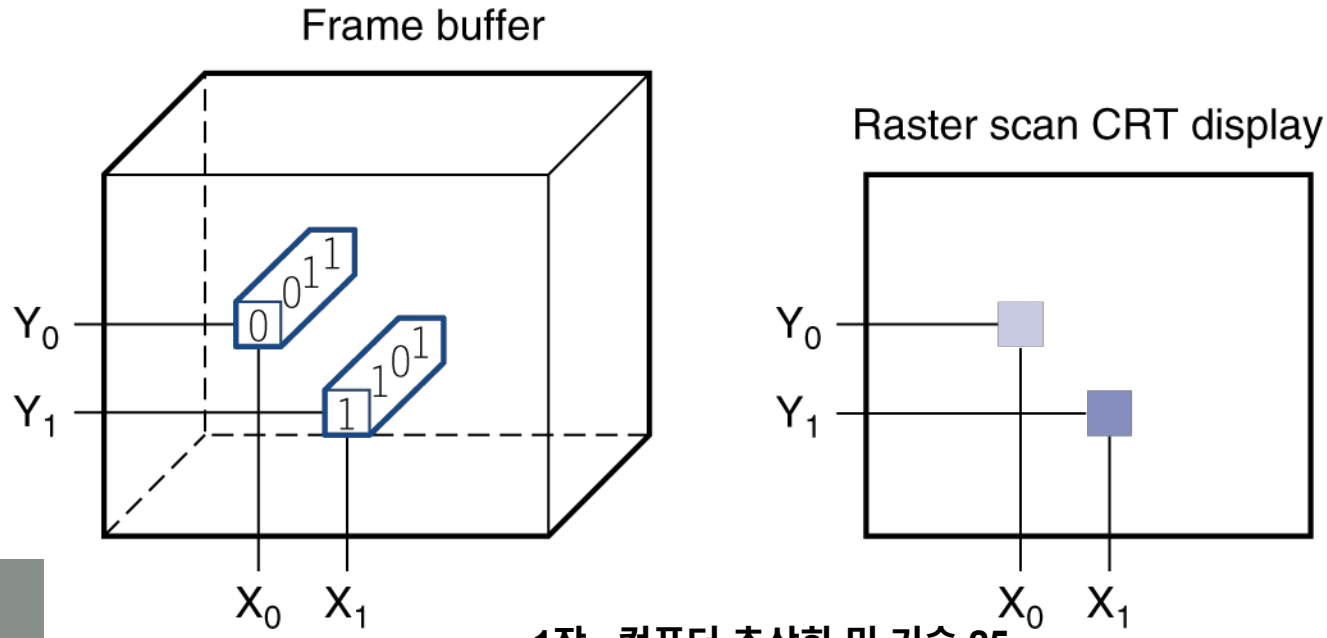


니다.

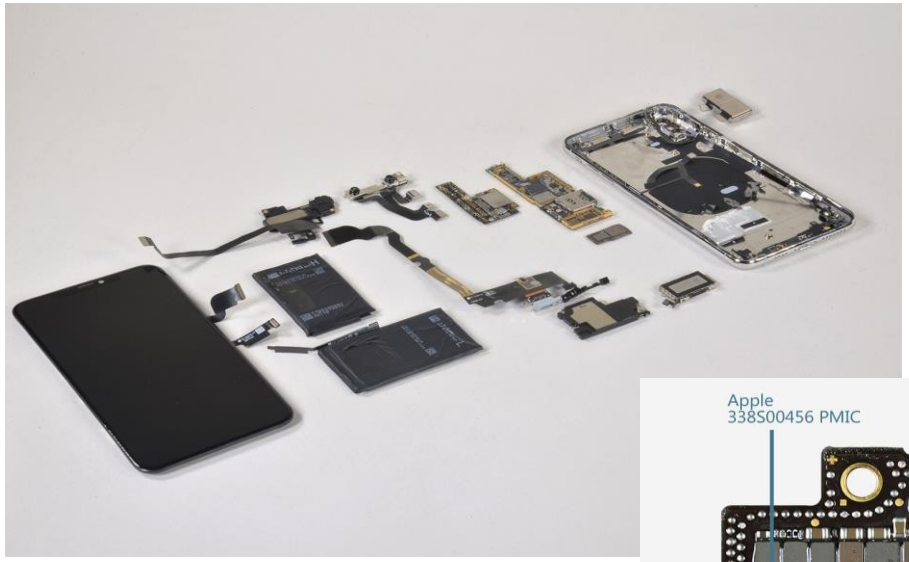
- 정전식으로 동시에 여러 번의 터치 가능

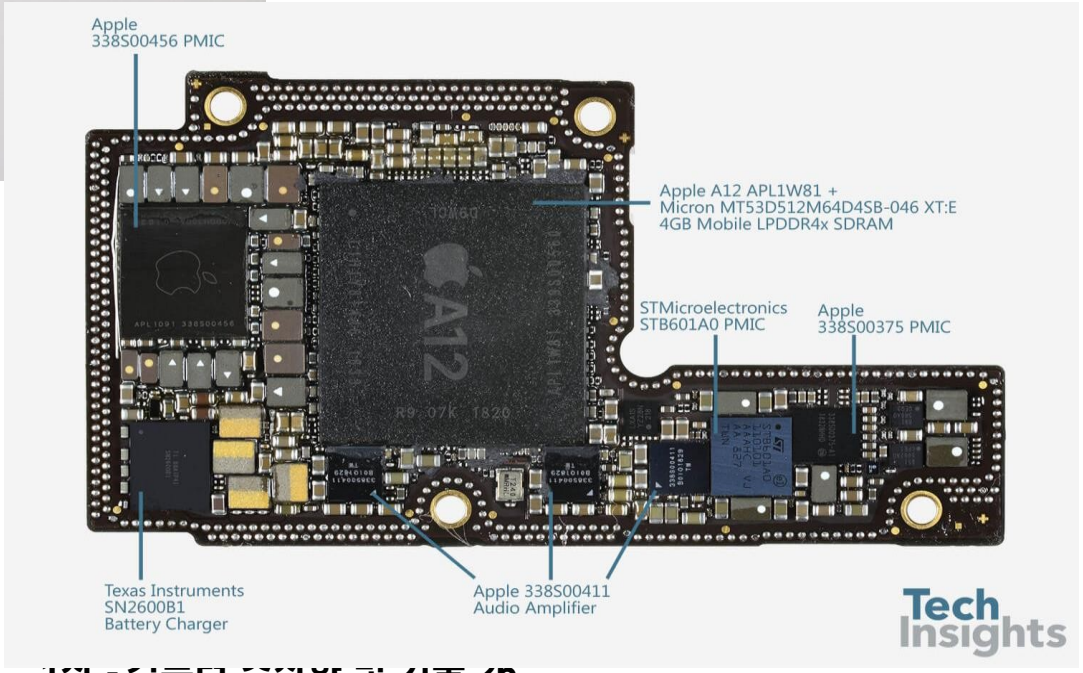
거울을 통해

- LCD 화면: 사진 요소(픽셀)
  - 프레임 버퍼 메모리의 콘텐츠를 미러링합니다.



상자 열기



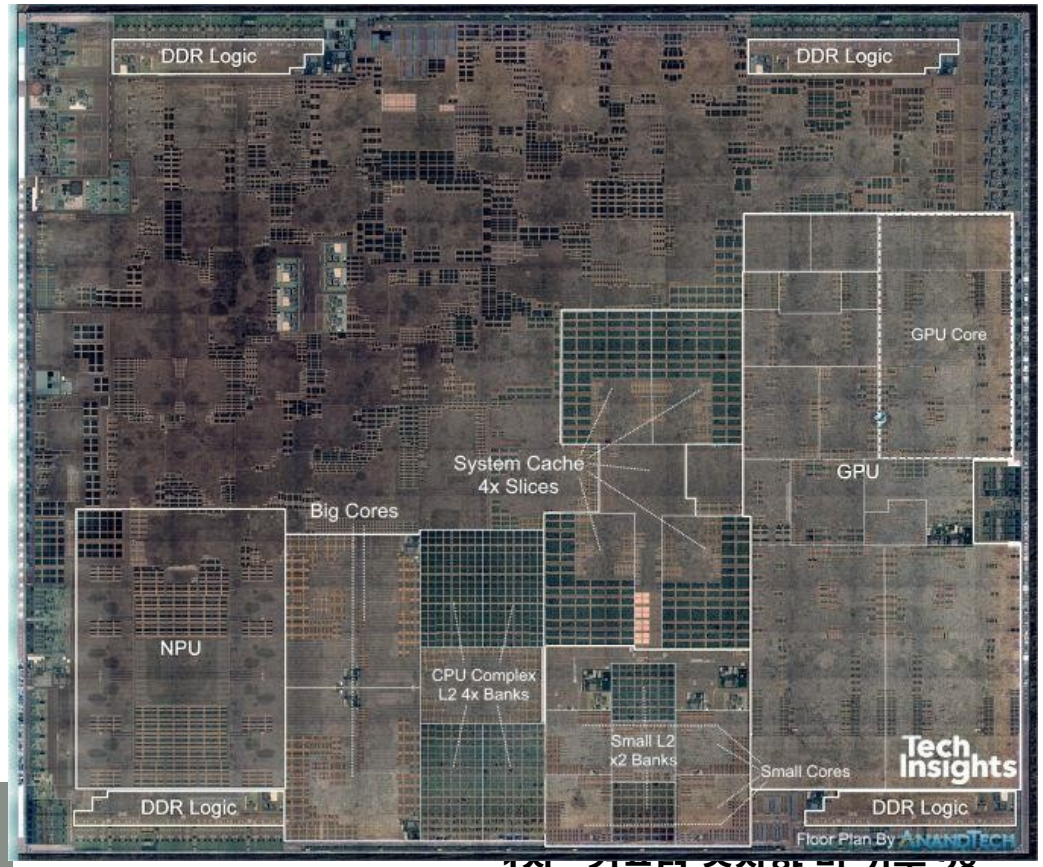

Apple
338S00456 PMIC
Apple A12 APL1W81 +
Micron MT53D512M64D4SB-046 XT:E
4GB Mobile LPDDR4x SDRAM
STMicroelectronics
STB601A0 PMIC
Apple
338S00375 PMIC
Texas Instruments
SN2600B1
Battery Charger
Apple 338S00411
Audio Amplifier
Tech Insights

프로세서 내부(CPU)

- 데이터 경로: 데이터에 대한 작업 수행
- 제어: 시퀀스 데이터 경로, 메모리, ...
- 캐시 메모리
  - 데이터에 즉시 액세스할 수 있는 작고 빠른 SRAM 메모리

프로세서 내부

- A12 프로세서



추상화

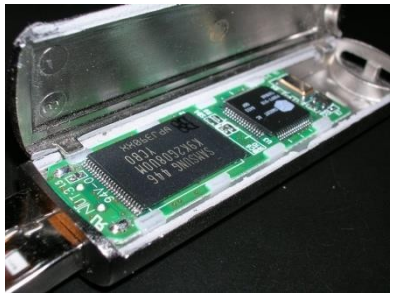
큰 그림

- 추상화는 복잡성을 처리하는 데 도움이 됩니다.
  - 하위 수준 세부 정보 숨기기
- 명령어 집합 아키텍처(ISA)
  - 하드웨어/소프트웨어 인터페이스

- 애플리케이션 바이너리 인터페이스
  - ISA 플러스 시스템 소프트웨어 인터페이스
- 구현
  - 기본 및 인터페이스 세부 정보

데이터를 위한 안전한 장소

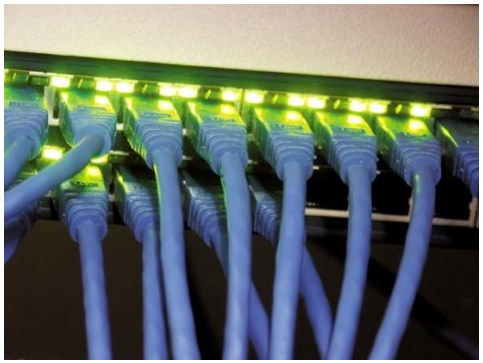
- 휘발성 메인 메모리
  - 전원이 꺼지면 지침 및 데이터 손실
- 비휘발성 보조 메모리
  - 자기 디스크
  - 플래시 메모리
  - 광디스크(CD롬, DVD)




DVD
RW

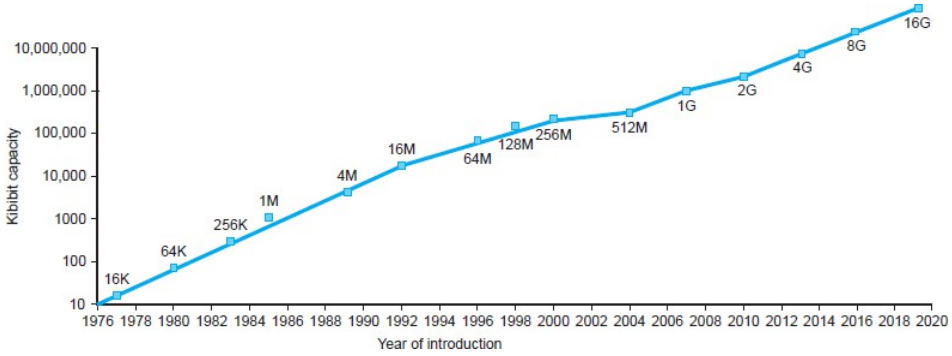
네트워크

- 커뮤니케이션, 리소스 공유, 비로컬 액세스
- LAN(로컬 영역 네트워크): 이더넷
- 광역 네트워크(WAN): 인터넷
- 무선 네트워크: WiFi, 블루투스



기술 트렌드

- 전자 기술은 계속 발전하고 있습니다.
  - 용량 및 성능 향상
  - 비용 절감



DRAM 용량

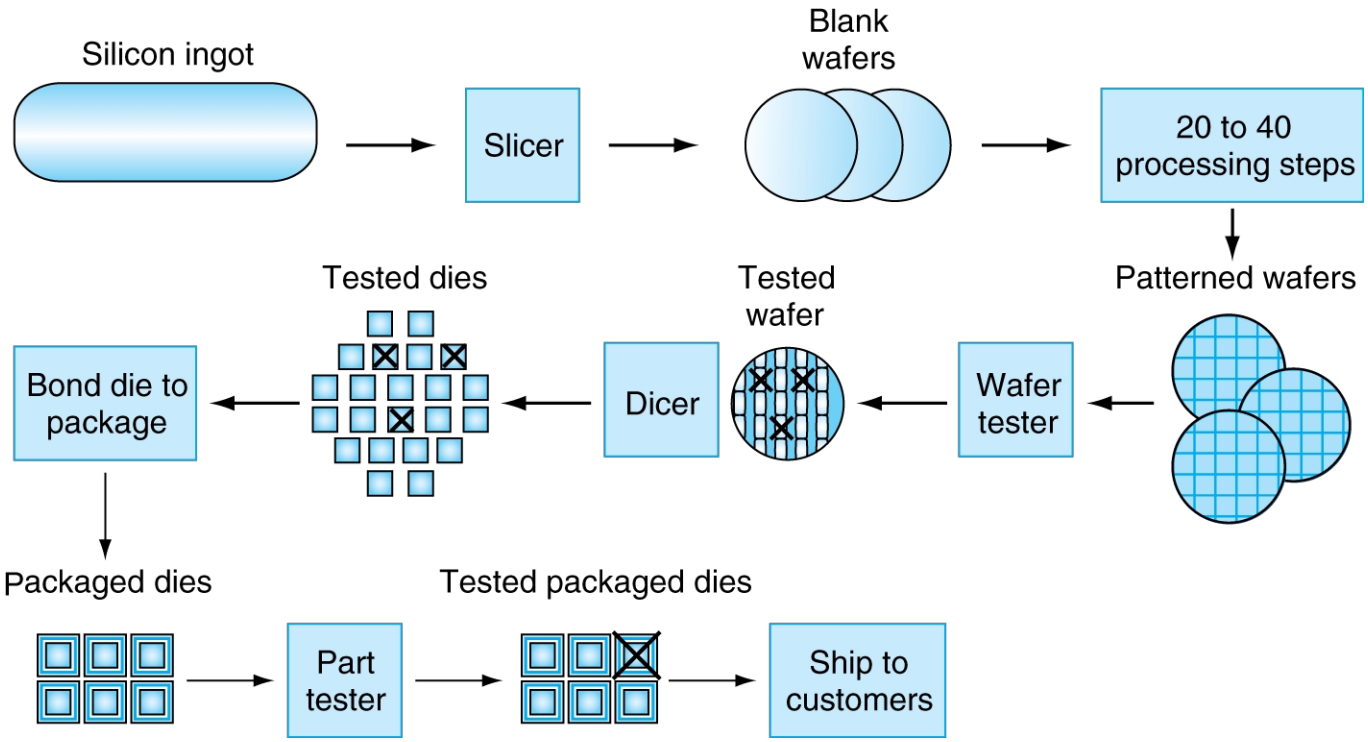
연도	기술	상대적 성능/비용
1951	진공관	1
1965	트랜지스터	35

1975	집적 회로(IC)	900
1995	초대형 IC(VLSI)	2,400,000
2013	초대형 IC	250,000,000,000

반도체 기술

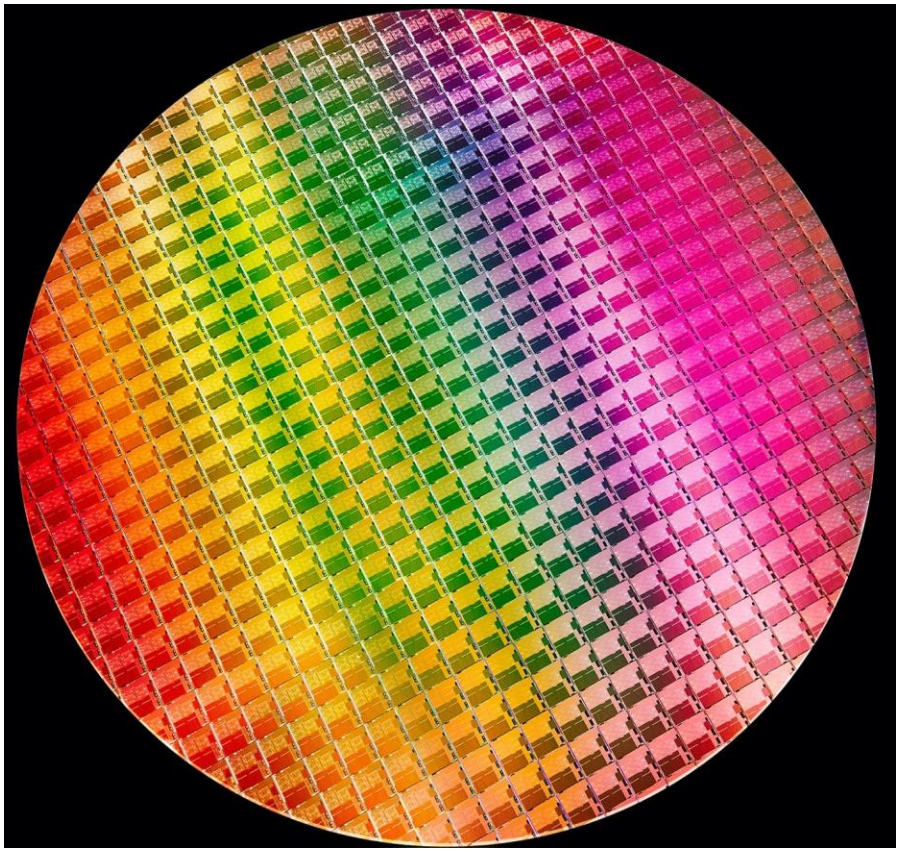
- 실리콘: 반도체
- 머티리얼을 추가하여 프로퍼티를 변형합니다:
  - 지휘자
  - 절연체
  - 스위치

IC 제조


Silicon ingot
Slicer
Blank wafers
20 to 40 processing steps
Patterned wafers
Wafer tester
Tested wafer
Dicer
Tested dies
Bond die to package
Packaged dies
Part tester
Tested packaged dies
Ship to customers

- 수율: 웨이퍼당 작업 다이의 비율

인텔® 코어 10세대



- 300mm 웨이퍼, 506개 칩, 10nm 기술
- 각 칩의 크기는 11.4 x 10.7mm입니다.

집적 회로 비용

$$\text{주사위당 비용} = \frac{\text{웨이퍼당 비용}}{\text{웨이퍼당 다이 수} \times \text{수율}}$$

$$\text{웨이퍼당 다이} \approx \frac{\text{웨이퍼 면적}}{\text{다이 면적}}$$

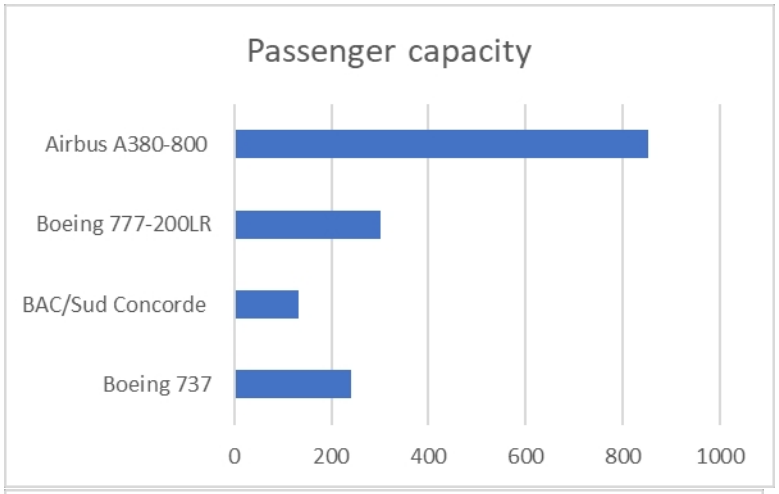
$$\text{수익률} = \frac{1}{(1+(\text{결함당 면적} \times \text{다이 면적}/2))^2}$$

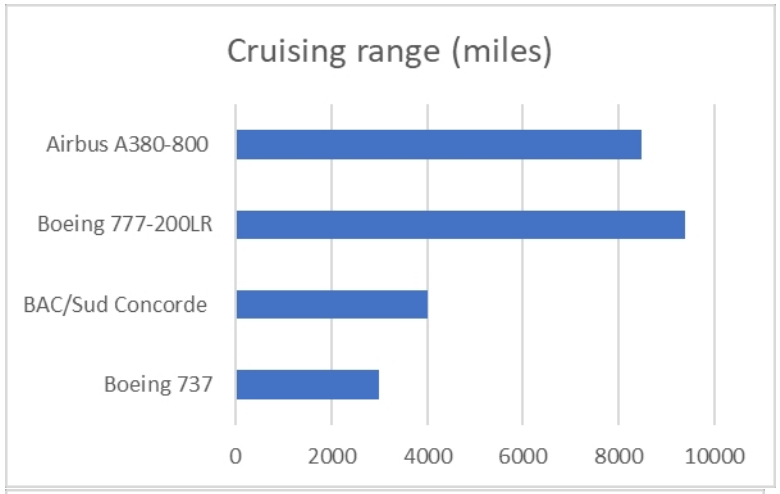
- 면적 및 불량률과의 비선형 관계
  - 웨이퍼 비용 및 면적 고정

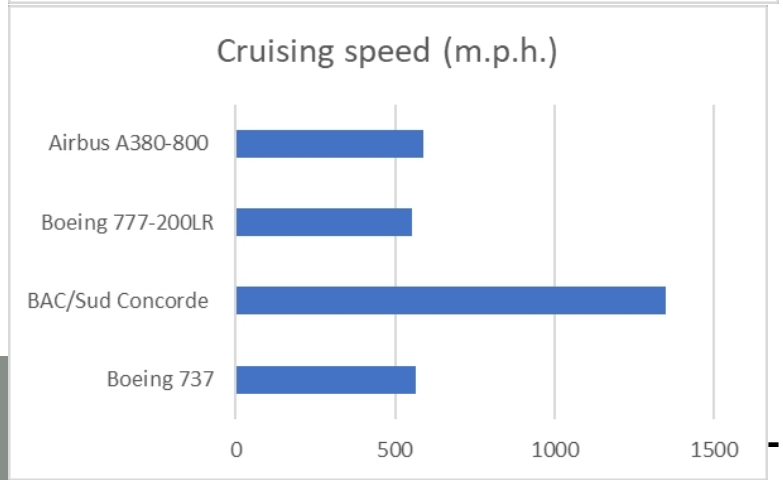
- 제조 공정에 따라 결정되는 불량률
- 아키텍처 및 회로 설계에 따라 결정되는 다이 면적

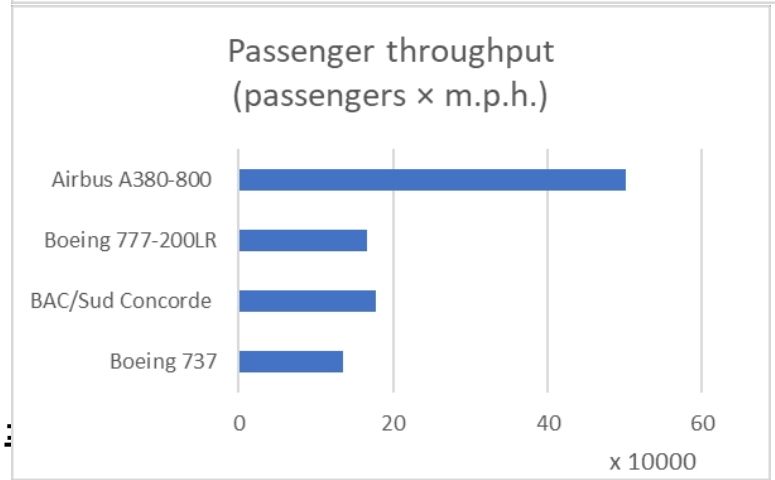
성능 정의

- 어떤 비행기가 가장 성능이 좋을까요?









응답 시간 및 처리량

- 응답 시간
  - 작업을 수행하는 데 걸리는 시간
- 처리량
  - 단위 시간당 수행한 총 작업 수
    - 예: 시간당 작업/거래/... 등
- 응답 시간과 처리량은 다음에 의해 영향을 받습니까 ?

- 프로세서를 더 빠른 버전으로 교체하시나요?
- 프로세서를 더 추가하시나요?

- 지금은 응답 시간에 집중하겠습니다...

상대적 성능

- 성능 = 1/실행 시간 정의
- "X가 Y보다 n시간 빠릅니다."

$$\frac{\text{성능}_X}{\text{성능}_Y} = \frac{\text{실행 시간}_Y}{\text{실행 시간}_X} = n$$

- 예: 프로그램 실행에 소요되는 시간

- A는 10초, B는 15초
- 실행 시간$_B$ / 실행 시간$_A$
  = 15초/10초 = 1.5
- 따라서 A는 B보다 1.5배 빠릅니다.

실행 시간 측정

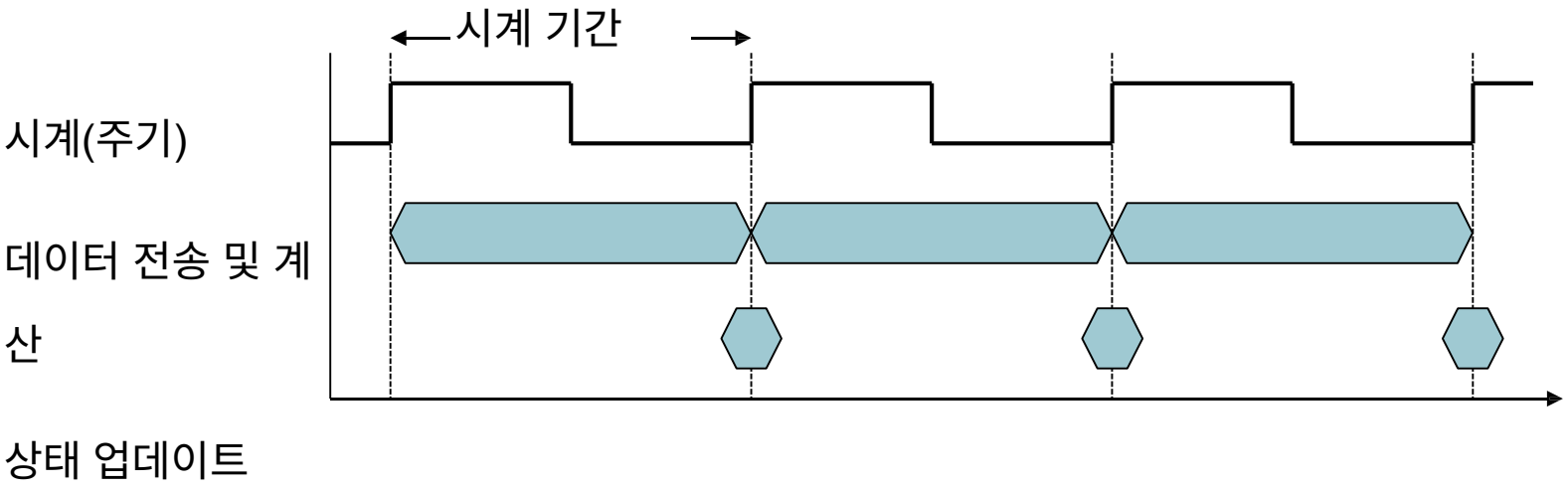
- 경과 시간
  - 모든 측면을 포함한 총 응답 시간
    - 처리, I/O, OS 오버헤드, 유휴 시간
  - 시스템 성능 결정
- CPU 시간
  - 주어진 작업을 처리하는 데 소요된 시간

- I/O 시간, 다른 작업의 공유 시간 할인

- 사용자 CPU 시간과 시스템 CPU 시간으로 구성
- 프로그램마다 CPU 및 시스템 성능에 따라 다르게 영향을 받습니다.

CPU 클럭킹

- 고정 속도 클럭에 의해 제어되는 디지털 하드웨어의 작동



- 시계 주기: 시계 주기 기간
  - 예: 250ps = 0.25ns = 250×10-12초
- 클록 주파수(속도): 초당 주기
  - 예: 4.0GHz = 4000MHz = 4.0×109Hz

CPU 시간

$$\text{CPU 시간} = \text{CPU 클럭 주기} \times \text{클럭 주기 시간}$$

$$= \frac{\text{CPU 클럭 주기}}{\text{클럭 속도}}$$

- 다음과 같이 성능이 향상되었습니다.
  - 클록 주기 수 감소
  - 클럭 속도 증가

- 하드웨어 설계자는 종종 클럭 속도와 사이클 수를 비교해야 합니다.

CPU 시간 예시

- 컴퓨터 A: 2GHz 클럭, 10초 CPU 시간
- 컴퓨터 B 설계
  - 6초 CPU 시간 목표
  - 더 빠른 클럭을 수행할 수 있지만 1.2배의 클럭 주기가 발생합니다.
- 컴퓨터 B의 클럭은 얼마나 빨라야 하나요?

$$\text{클럭 속도}_B = \frac{\text{클록 주기}_B}{\text{CPU 시간}_B} = \frac{1.2\text{배 클록 사이클}_A}{6\text{s}}$$

$$\text{클럭 주기}_A = \text{CPU 시간}_A \times \text{클럭 속도}_A$$

$$= 10\text{초} \times 2\text{GHz} = 20 \times 10^9$$

$$\text{클럭 속도}_B = \frac{1.2 \times 20 \times 10^9}{6\text{s}} = \frac{24 \times 10^9}{6\text{s}} = 4\text{GHz}$$

인스트럭션 수 및 CPI

클럭 사이클 = 명령어 수 × 명령어당 사이클 CPU 시간 = 명령

어 수 × □□□ × 클럭 사이클 시간

$$= \frac{\text{명령어 수} \times \text{CPI}}{\text{클럭 속도}}$$

- 프로그램의 명령어 수
  - 프로그램, ISA 및 컴파일러에 의해 결정됩니다.

- 인스트럭션당 평균 주기
  - CPU 하드웨어에 의해 결정
  - 지침마다 CPI가 다른 경우
    - 인스트럭션 믹스의 영향을 받는 평균 CPI

CPI 예시

- 컴퓨터 A: 사이클 시간 = 250ps, CPI = 2.0
- 컴퓨터 B: 사이클 시간 = 500ps, CPI = 1.2
- 동일한 ISA
- 어느 것이 더 빠르며, 얼마나 빠를까요?

$$\text{CPU 시간}_A = \text{명령어 수} \times \text{CPI}_A \times \text{주기 시간}_A$$

$$= \boxtimes \times 2.0 \times 250\text{ps} = \boxtimes \times 500\text{ps}$$

A가 더 빠릅니다...

$$\text{CPU 시간}_B = \text{명령어 수} \times \text{CPI}_B \times \text{주기 시간}_B$$

$$= \boxtimes \times 1.2 \times 500\text{ps} = \boxtimes \times 600\text{ps}$$

$$\frac{\text{CPU 시간}_B}{\text{CPU 시간}_A} = \frac{\boxtimes \times 600\text{ps}}{\boxtimes \times 500\text{ps}} = 1.2$$

...이만큼

CPI 자세히 알아보기

- 인스트럭션 클래스마다 사이클 횟수가 다른 경우

$$\text{시계 주기} = \sum_{i=1}^{n} (\text{CPI}_i \times \text{인스트럭션 수})_i$$

  - 가중 평균 CPI

$$\text{CPI} = \frac{\text{시계 주기}}{\text{명령어 개수}} = \sum_{i=1}^{n} \left(\text{CPI}_i \times \underbrace{\frac{\text{명령어 개수}_i}{\text{명령어 개수}}}_{\text{상대적 빈도}} \right)$$

CPI 예시

- 클래스 A, B, C의 명령어를 사용하는 대체 컴파일된 코드 시퀀스

클래스	A	B	C
클래스용 CPI	1	2	3
시퀀스 1의 IC	2	1	2
시퀀스 2의 IC	4	1	1

- 시퀀스 1: IC = 5
  - 시계 주기
    = 2×1 + 1×2 + 2×3
    = 10
  - Avg. cpi = 10/5 = 2.0

- 시퀀스 2: IC = 6
  - 시계 주기
    = 4×1 + 1×2 + 1×3
    = 9
  - Avg. cpi = 9/6 = 1.5

성능 요약

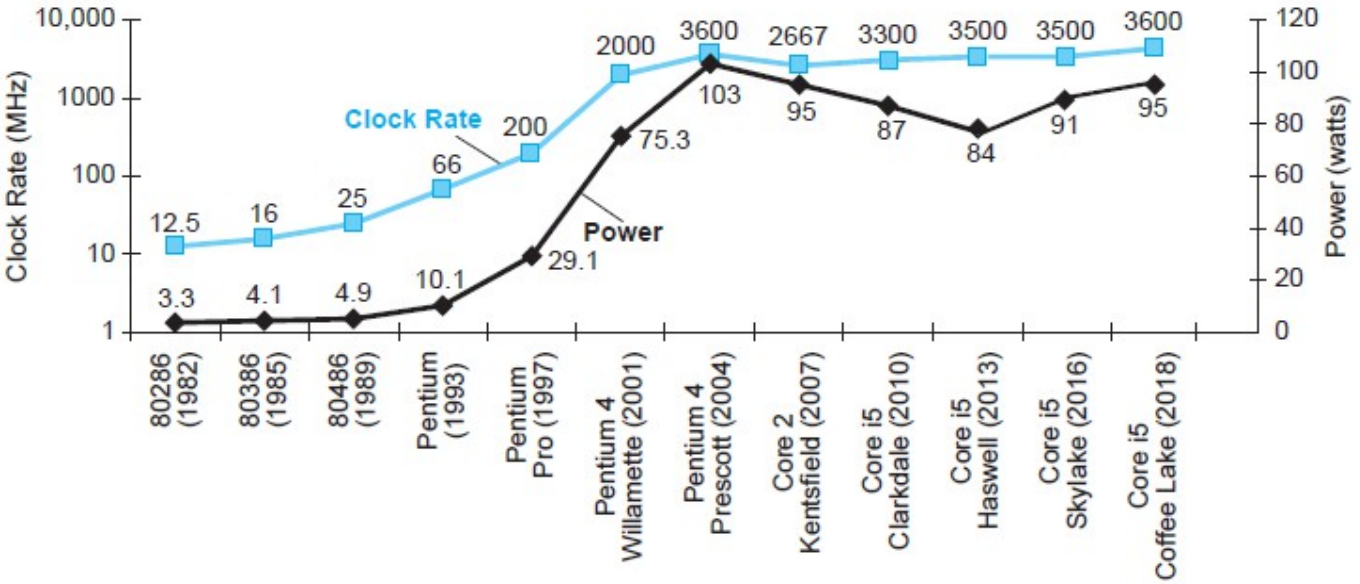
큰 그림

$$\text{CPU Time} = \frac{\text{지침}}{\text{프로그램}} \times \frac{\text{시계 주기}}{\text{지침}} \times \frac{\text{초}}{\text{시계 주기}}$$

- 성능은 다음에 따라 달라집니다.
  - 알고리즘: IC, CPI에 영향을 미칩니다.

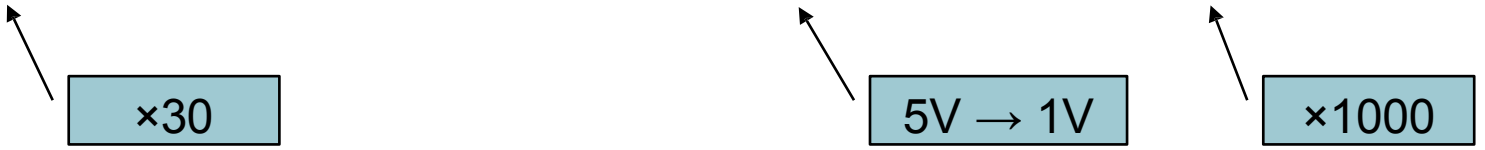
- 프로그래밍 언어: IC, CPI에 영향을 미칩니다.
- 컴파일러: IC, CPI에 영향
- 명령어 집합 아키텍처: IC, CPI, T에 영향을 미칩니다.c

파워 트렌드



- CMOS IC 기술에서

$$\text{전력} = \text{정전 용량 부하} \times \text{전압}^2 \times \text{주파수}$$



전력 절감

- 새 CPU에 다음이 있다고 가정해 보겠습니다.
  - 구형 CPU의 정전 용량 부하 85%
  - 전압 15% 및 주파수 15% 감소

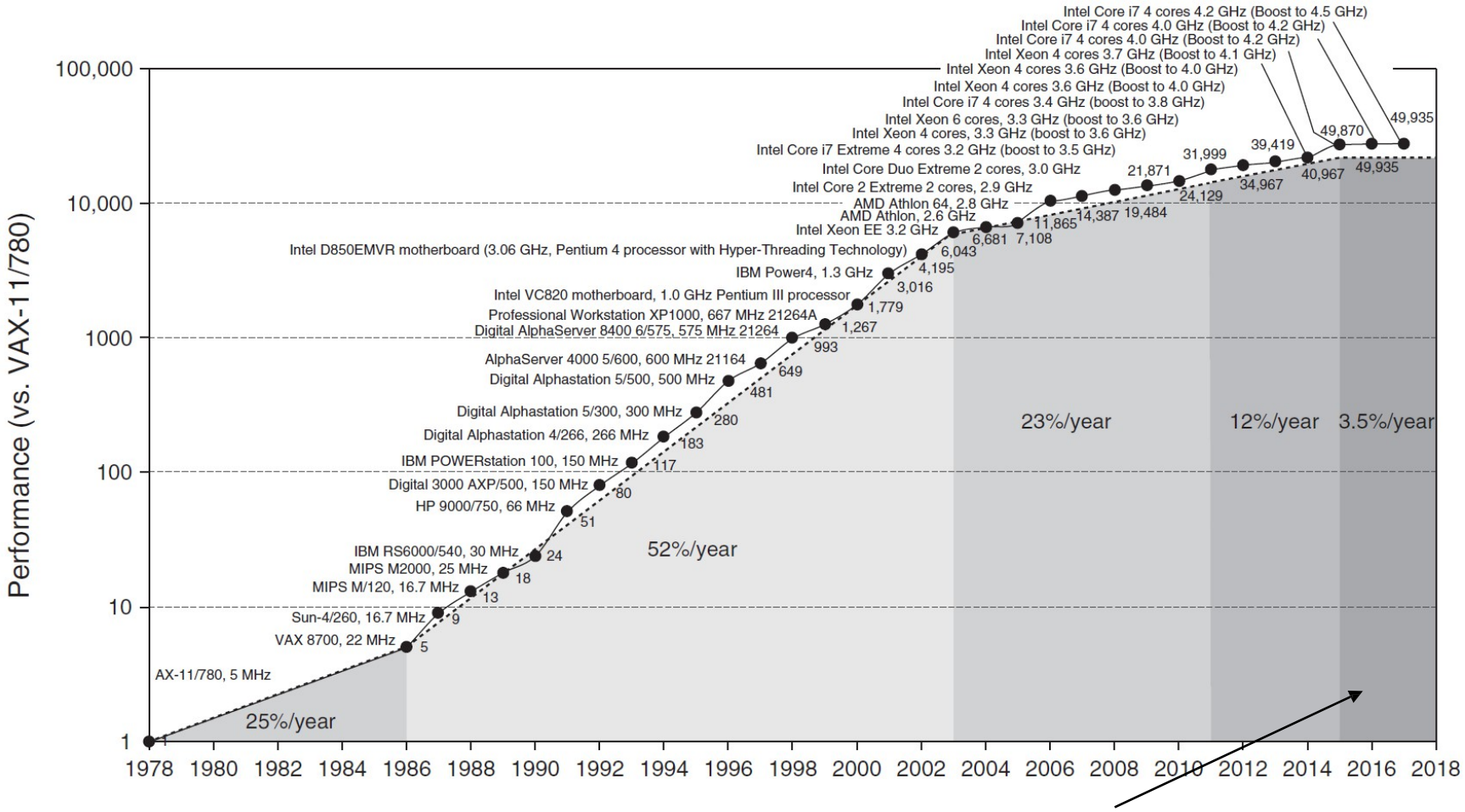
$$\frac{P_{new}}{P_{old}} = \frac{C_{old} \times 0.85 \times (V_{old} \times 0.85)^2 \times f_{old} \times 0.85}{\text{콜드} \times \text{볼드}^2 \times F_{old}} = 0.85^4 = 0.52$$

- 파워 월

- 전압을 더 이상 낮출 수 없습니다.
- 더 이상 열을 제거할 수 없습니다.

- 성능을 개선할 수 있는 다른 방법은 무엇인가요?

유니프로세서 성능



전력, 명령어 수준 병렬 처리, 메모리 지연 시간의 제약을 받습니다.

멀티프로세서

- 멀티코어 마이크로프로세서
  - 칩당 하나 이상의 프로세서
- 명시적으로 병렬 프로그래밍이 필요합니다.
  - 명령어 수준 병렬 처리와 비교
    - 하드웨어는 한 번에 여러 명령을 실행합니다.
    - 프로그래머에게 숨겨짐

- 어려운 일
  - 성능을 위한 프로그래밍
  - 로드 밸런싱
  - 커뮤니케이션 및 동기화 최적화

사양 CPU 벤치마크

- 성능 측정에 사용되는 프로그램
  - 실제 워크로드의 일반적인 경우
- 표준 성능 평가 공사(SPEC)
  - CPU, I/O, 웹, ...에 대한 벤치마크를 개발합니다.
- 사양 CPU2006
  - 선택한 프로그램을 실행하기까지 경과된 시간
    - I/O가 미미하므로 CPU 성능에 중점을 둠

- 기준 머신을 기준으로 정규화
- 성능 비율의 기하평균으로 요약합니다.
  - CINT2006(정수) 및 CFP2006(부동 소수점)

$$\sqrt[n]{\prod_{i=1}^{n} \text{실행 시간 비율}_i}$$

SPECspeed 2017 정수 벤치마크에서

1.8GHz 인텔 제온 E5-2650L

Description	*Name*	*Instruction Count x 10^9*	*CPI*	*Clock cycle time (seconds x 10^–9)*	*Execution Time (seconds)*	*Reference Time (seconds)*	*SPECratio*
Perl interpreter	perlbench	2684	0.42	0.556	627	1774	2.83
GNU C compiler	gcc	2322	0.67	0.556	863	3976	4.61
Route planning	mcf	1786	1.22	0.556	1215	4721	3.89
Discrete Event simulation - computer network	omnetpp	1107	0.82	0.556	507	1630	3.21
XML to HTML conversion via XSLT	xalancbmk	1314	0.75	0.556	549	1417	2.58
Video compression	x264	4488	0.32	0.556	813	1763	2.17
Artificial Intelligence: alpha-beta tree search (Chess)	deepsjeng	2216	0.57	0.556	698	1432	2.05
Artificial Intelligence: Monte Carlo tree search (Go)	leela	2236	0.79	0.556	987	1703	1.73
Artificial Intelligence: recursive solution generator (Sudoku)	exchange2	6683	0.46	0.556	1718	2939	1.71
General data compression	xz	8533	1.32	0.556	6290	6182	0.98
Geometric mean							2.36

사양 전력 벤치마크

- 다양한 워크로드 수준에서 서버의 전력 소비량
  - 성능: ssj_ops/sec(서버 측 Java 작업)
  - 전력: 와트(줄/초)

$$\text{오버올즈j\_옵스퍼 와트} = \left(\sum_{i=0}^{10} \text{ssj\_ops}_i \right) \Big/ \left(\sum_{i=0}^{10} \text{power}_i \right)$$

제온 E5-2650L용 SPECpower_ssj2008

Target Load %	Performance (ssj_ops)	Average Power (watts)
100%	4,864,136	347
90%	4,389,196	312
80%	3,905,724	278
70%	3,418,737	241
60%	2,925,811	212
50%	2,439,017	183
40%	1,951,394	160
30%	1,461,411	141
20%	974,045	128
10%	485,973	115
0%	0	48
Overall Sum	26,815,444	2,165
∑ssj_ops / ∑power =		12,385

함정: 암달의 법칙

- 컴퓨터의 한 측면을 개선하면 전체 성능의 비례적인 향상을 기대할 수 있습니다.

$$T_{\text{개선됨}} = \frac{\text{영향을 받은 개선 요인}}{} + T_{\text{영향을 받지 않음}}$$

- 예: 80대/100대 계정 곱하기
  - 멀티플 성능이 전체적으로 5배 향상되려면 얼마나 개선되어야 하나요?

$$20 = \frac{80}{n} + 20$$

- 할 수 없습니다!

- 결론: 공통 사례를 빠르게 만들기

오류: 사용률이 낮은 컴퓨터는 전력을 거의 사용하지 않습니다.

- i7 성능 벤치마크 다시 보기
  - 100% 로드 시: 258W
  - 50% 로드 시: 170W (66%)
  - 10% 로드 시: 121W (47%)
- Google 데이터 센터

- 대부분 10% - 50% 부하에서 작동
- 100% 로드 시 1% 미만의 시간

- 부하에 따라 전력이 비례하도록 프로세서를 설계하는 것을 고려하세요.

함정: 성능으로서의 MIPS 메트릭(건너뛰기)

- MIPS: 초당 수백만 건의 명령어
  - 다음을 설명하지 않습니다.
    - 컴퓨터 간 ISA의 차이점
    - 지침 간의 복잡성 차이

$$\text{MIPS} = \frac{\text{인스트럭션 수}}{\text{실행 시간} \times 10^6}$$

$$= \frac{\text{인스트럭션 수}}{\frac{\text{명령어 수} \times \text{CPI}}{\text{클럭 속도}} \times 10^6} = \frac{\text{클럭 속도}}{\text{CPI} \times 10^6}$$

- CPI는 특정 CPU의 프로그램마다 다릅니다.

결론

- 비용 대비 성능이 개선되고 있습니다.
  - 기반 기술 개발로 인해
- 계층적 추상화 계층
  - 하드웨어와 소프트웨어 모두에서
- 명령어 집합 아키텍처
  - 하드웨어/소프트웨어 인터페이스

- 실행 시간: 최고의 성능 측정
- 전력은 제한 요소입니다.
  - 병렬 처리를 사용하여 성능 향상