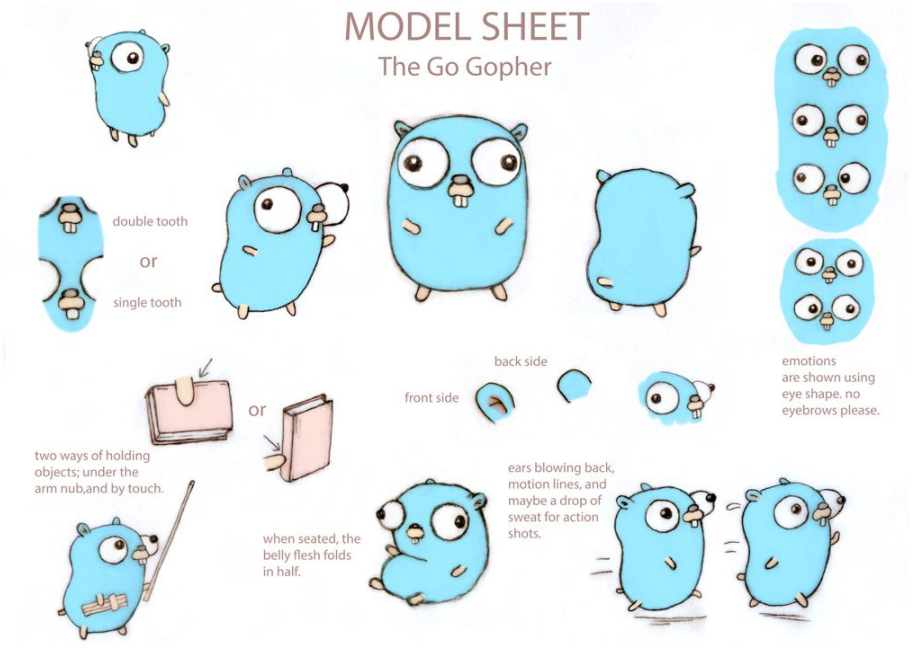
Chapter 1 - 2

Property	
날짜	@October 6, 2021
발표자	장예서



Gopher의 MODEL SHEET 짱귀엽다

Chapter 1 컴퓨터 원리

1.1 비트의 탄생과 트랜지스터

트랜지스터 → 2가지 실리콘, N형과 P형으로 구성

예) NPN 트랜지스터 : 이미터(N), 베이스(P), 콜렉터(N)

1.1.1 트랜지스터와 0과1

트랜지스터 기능

1. 증폭
2. **스위치**: 베이스 부분에 전압을 가하면 전류가 흐른다

1.1.2 2진수

트랜지스터 1개 → 1비트

1.2 트랜지스터에서 계산기로

1.2.1 논리 소자와 트랜지스터

AND, OR, XOR, NOT 등 논리 소자를 트랜지스터로 만들 수 있다

1.2.2 논리 소자로 사칙연산 구현하기

예) 1비트 가산기 : 자리올림수 - AND, 합의 결과 - XOR

트랜지스터 → 논리 소자 → 계산기

1.3 계산기에서 컴퓨터로

1.3.1 최초의 상상 속 컴퓨터 : 튜링 머신

상상으로 구현한 기계, a-machine ~~(OMG... 앨런 튜링... 동성애법 금지... 청산가리 사과...ㅠㅠ)~~

1.3.2 컴퓨터의 완성 : 폰 노이만 구조

CPU, 메모리, 외부 입출력 장치 → 현대 컴퓨터 기본 구조 ~~(헉 핵폭탄...)~~

1.4 컴퓨터 동작 원리

휘발성 메모리 (램) / 비휘발성 메모리 (하드 디스크)

1. 프로그램 로드 : OS가 실행 파일을 메모리에 복사
2. 데이터 로드 및 캐싱 : 메모리에서 연산에 필요한 데이터를 캐시로 복사
3. 연산 및 저장 : CPU가 연산에 사용할 데이터를 레지스터로 복사 → 메모리에 저장할 수도

~ 캐시 미스 발생 가능

Chapter 2 프로그래밍 언어

2.1 초창기 프로그래밍 언어

OP 코드 : 수행할 명령어를 나타내는 부호 (Operation code = 명령 코드)

예) **ADD** 3 4 = **0011** 0011 0100

초기 프로그래밍: 천공카드 구멍 뚫어 기계어 작성

2.2 어셈블리어의 등장

어셈블리어 : 기계어와 1:1로 대응되는, 인간이 쉽게 읽고 쓸 수 있는 언어

현재까지도 임베디드 프로그래밍에 많이 사용됨

2.3 고수준 언어의 등장

어셈블리어의 문제점 : 코드 양이 많다 → 전체 동작 이해에 어려움, 버그 발생, 다른 아키텍처로 이식이 어려움

고수준 언어: 높은 생산성(작성 시간 단축), 높은 가독성(쉬운 이해, 낮은 오류 가능성), 유연한 이식성 제공

2.3.1 고수준 코드가 실행되기까지

고수준 언어 코드 → 컴파일 → 기계어

각 언어들은 자신만의 고유한 컴파일러가 있다. 예) Go 컴파일러, C 컴파일러

2.4 프로그래밍 언어의 구분

2.4.1 정적 컴파일 언어 vs 동적 컴파일 언어

.

Aa Name	설명	장점	단점
정적 컴파일 언어	미리 기계어로 변환해 두었다가 사용하는 방식	실행할 때 빠름, 높은 타입 안전성	칩셋과 OS마다 바이너리 코드를 표현하는 형식이 다르고 CPU 아키텍처와 OS에 따라 실행환경이 달라진다. → 많은 빌드 과정 요구
동적 컴파일 언어	실행 시점에 기계어로 변환하는 방식, 정적보다 이후에 개발	모든 플랫폼에서 실행 가능(범용성)	정적보다 더 느리게 동작

GO는 정적 컴파일 언어 : 각 플랫폼에 맞는 실행 파일을 따로 생성해야함, 내부환경 변수만 바꿔서 비교적 쉽게 대응, 매우 빠른 실행 속도

2.4.2 약 타입 언어 vs 강 타입 언어

.

Aa Name	설명	장점	단점
강 타입 언어 / 정적 타입 언어	타입 검사를 강하게 하는 언어, 서로 다른 타입 간 연산에 엄격	타입으로 생길 수 있는 문제 미연에 방지	사용하기 까다로움
약 타입 언어 / 동적 타입 언어	타입 검사를 약하게 하는 언어, 서로 다른 타입 간 연산에 관대	편한 코딩	예기치 못한 버그 발생 가능

GO는 최강 타입 언어 : 자동 타입 변환 지원 X, 사용하기 까다롭지만 타입이 달라서 발생하는 문제점 전혀 X

2.4.3 가비지 컬렉터 유무

.

Aa Name	설명	장점	단점
가비지 컬렉터 O	메모리에서 불필요한 영역을 치워줌	메모리 관련 문제 감소	CPU 성능 사용됨
가비지 컬렉터 X	프로그래머가 메모리 할당/해제 책임	대체로 더 빠름	메모리 누수 현상, 버그 발생

GO는 가비지 컬렉터 제공 : 자동으로 불필요한 메모리 해제, 매우 발전된 형태로 성능 손실이 크지 않다