What is computation

What Does A Computer Do

- Fundamentally:
  - performs calculations 계산이 핵심 기능
    - a billion calculations per second! 초당 수 억 개의 계산 가능
  - remembers results 저장공간에 결과를 저장 가능
    - 100s of gigabytes of storage!
- What kinds of calculations? 연산의 종류?
  - built-in to the language 언어가 정한 명령
  - ones that you define as the programmer 프로그래머가 정의한 명령
- Computers only know what you tell them 시킨 일만 할 수 있음

Types Of Knowledge

- declarative knowledge is statements of fact. 선언적 지식은 사실만 말함

- imperative knowledge is a recipe or “how-to”. 명령형 지식은 방법을 다룸

A numerical Example

- square root of a number x is y such that y*y = x 제곱 근의 계산

- recipe for deducing square root of a number x (16) 제곱 근 계산의 방법
  1. Start with a guess, g 추측
  2. If g*g is close enough to x, stop and say g is the answer 제곱이 비슷하면 종료
  3. Otherwise make a new guess by averaging g and x/g 새로운 추측 값 계산
  4. Using the new guess, repeat process until close enough 비슷해질 때까지 반복

g	g*g	x/g	(g+x/g)/2
3	9	16/3	4.17
4.17	17.36	3.837	4.0035
4.0035	16.0277	3.997	4.000002

What is a recipe

1. Sequence of simple steps 단순한 단계
2. flow of control process that specifies when each step is executed 흐름 제어가 필요
3. a means of determining when to stop 종료 조건이 필요

1 + 2 + 3 = an algorithm ! 위 3개가 알고리즘의 정의

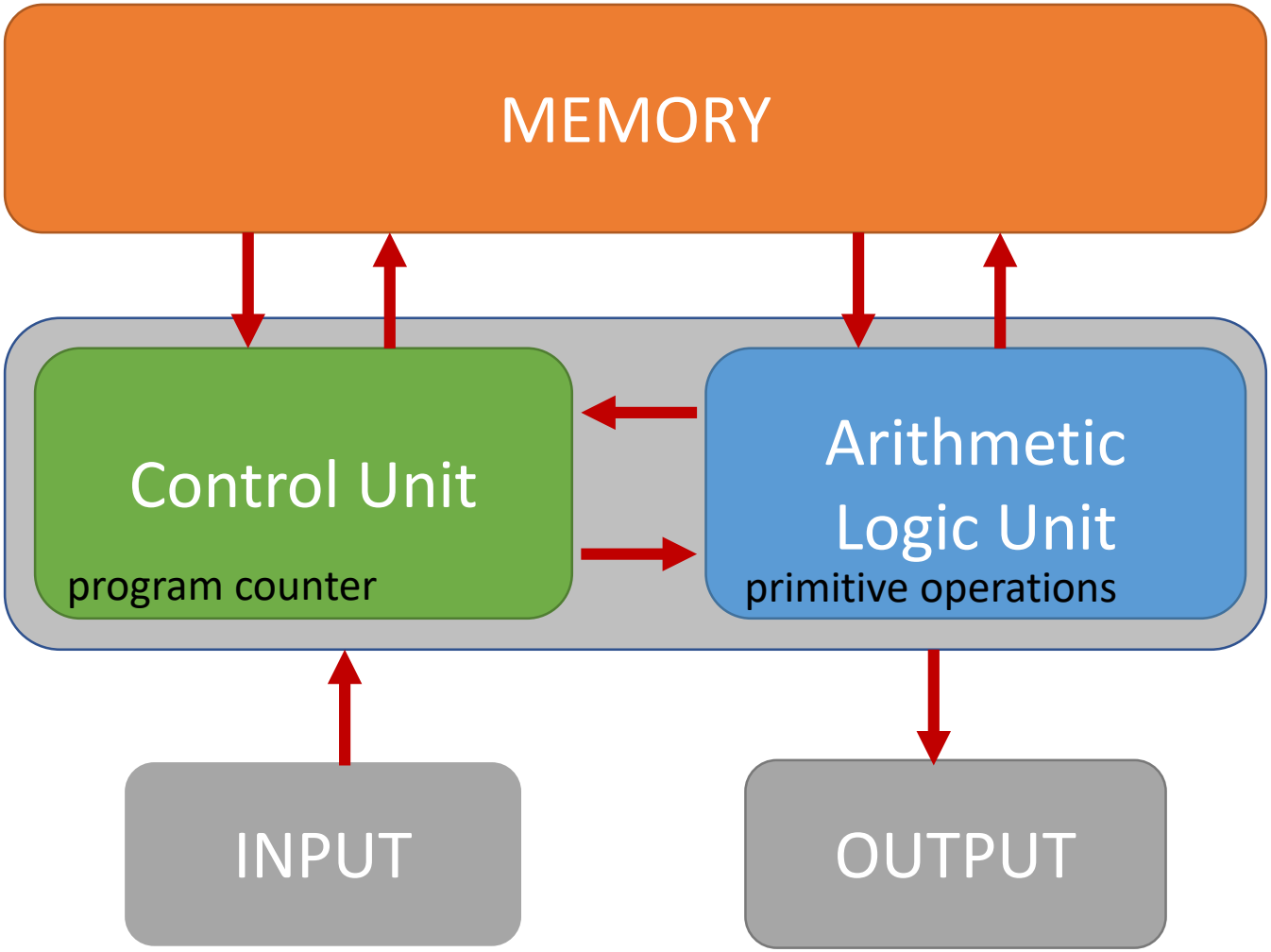
Computers are machines

- How to capture a recipe in a mechanical process 기계적 방법

1. fixed program computer: Calculator 고정형 프로그램 컴퓨터: 계산기

2. Stored Program computer: machine stores and executes instructions
   저장형 프로그램 컴퓨터: 저장된 명령어 실행

Basic Machine Architecture



Stored Program Computer

- Sequence of instructions stored inside computer 컴퓨터에 저장된 명령의 순서
  - built from predefined set of primitive instructions 기본적 명령의 조합으로 정의됨
    1. arithmetic and logic 수식과 논리 계산
    2. simple tests 간단한 검사
    3. moving data 데이터의 이동
- special program (interpreter) executes each instruction in order 특별한 프로그램인 해석기가 각 명령을 실행함
  - use tests to change flow of control through sequence 검사를 통해 실행을 제어함
  - stop when done 다 실행 후 종료

Basic Primitives

- Turing showed that you can compute anything using 6 primitives
  튜링이 6가지 기본 명령이면 어떤 것이든 계산할 수 있다고 증명
  - Move left, Move right, Print, Scan, Erase, Do nothing
  - if you are interested to learn more about it: reference or small video
- modern programming languages have more convenient set of primitives
  현대 프로그래밍 언어는 좀 더 편한 기본 명령어들이 있음
- can abstract methods to create new primitives 새 기본 명령 만들기 가능
- anything computable in one language is computable in any other programming language 한 언어로 연산 가능한 것은 다른 언어에서도 연산 가능

Creating Recipes

- a programming language provides a set of primitive operations 기본 명령어 집합이 제공됨
- expressions are complex but legal combinations of primitives in a programming language 표현식이 복잡하더라도 기본 명령 조합이면 해석 가능
- expressions and computations have values and meanings in a programming language 표현식과 연산에는 값과 의미가 있음

Aspects of Languages

- Primitive constructs: Syntax and Semantic 문법과 의미
  - Syntax: Defines the grammar 문법을 정의
  - Semantic: is the meaning associated with syntactically correct symbols with no semantic errors 사용된 심볼들이 의미에 맞는지 확인

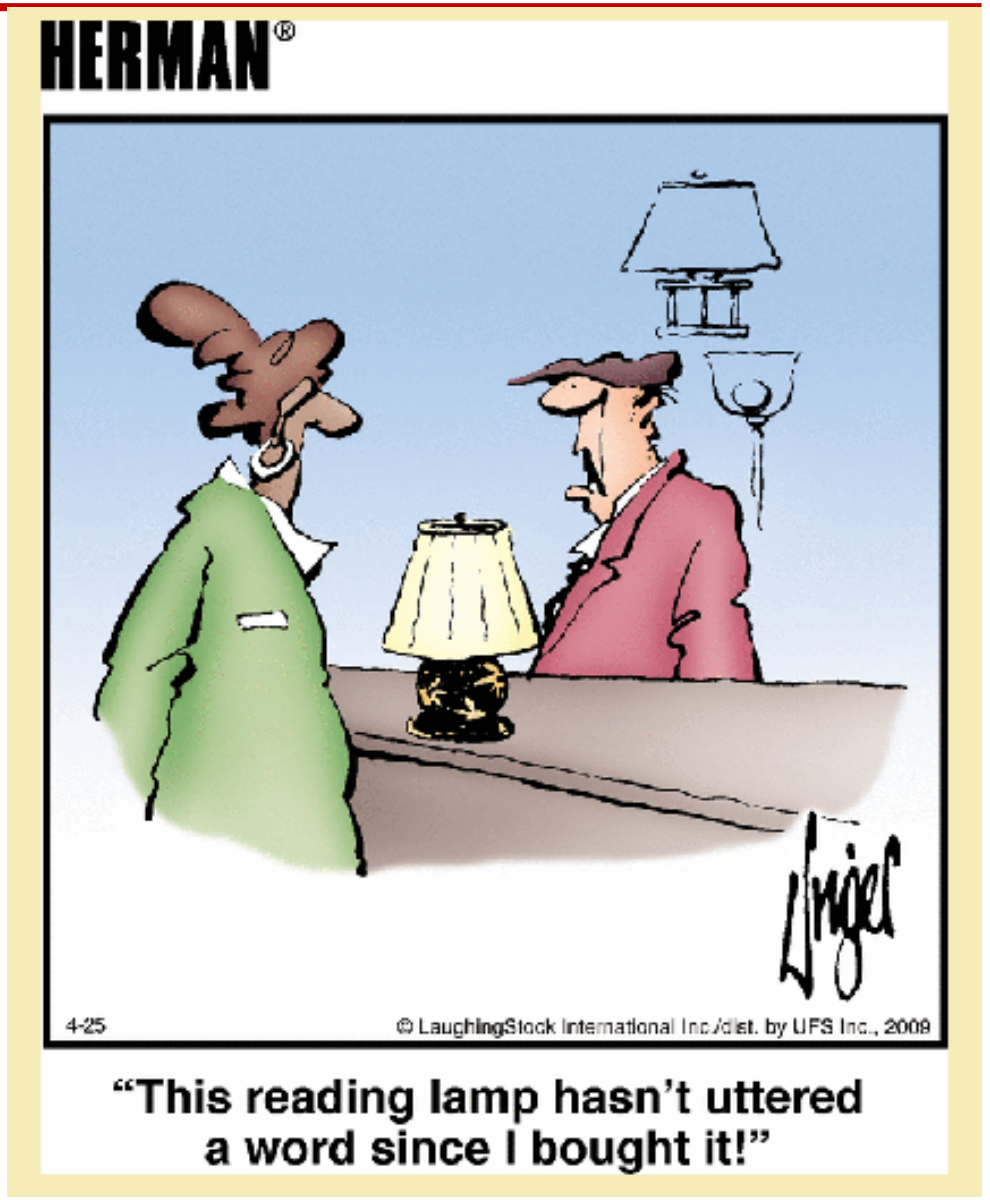
  - English/Korean: Words 단어
    - 아버지 가방에 들어가신다 – syntactically valid but semantically not correct
    - 아기 고기 다리 – not syntactically valid

  - Programming Language: Numbers, Strings, Simple operators 숫자 문자, 연산자
    - 3.14*8 – syntactically valid
    - “hi”5 – not syntactically valid

Aspects of Languages

- Natural languages have many meanings
  자연어는 다중적 의미를 갖음

- Programming Languages have only one meaning but may not be what programmer intended
  프로그래밍 언어는 유일한 의미만 갖음



Where things go wrong

- syntactic errors 문법 오류
  - common and easily caught 쉽게 고칠 수 있음
- static semantic errors 정적 의미 오류
  - some languages check for these before running program 의미가 맞는지 미리 검사하기도 함
  - can cause unpredictable behavior 예측 불가능한 오류
- no semantic errors but different meaning than what programmer intended 오류가 없지만 프로그래머의 의도와 다를 수 있음
  - program crashes, stops running 오동작, 멈춤
  - program runs forever 무한 반복 실행
  - program gives an answer but different than expected 예상 외의 결과