COMPILERS
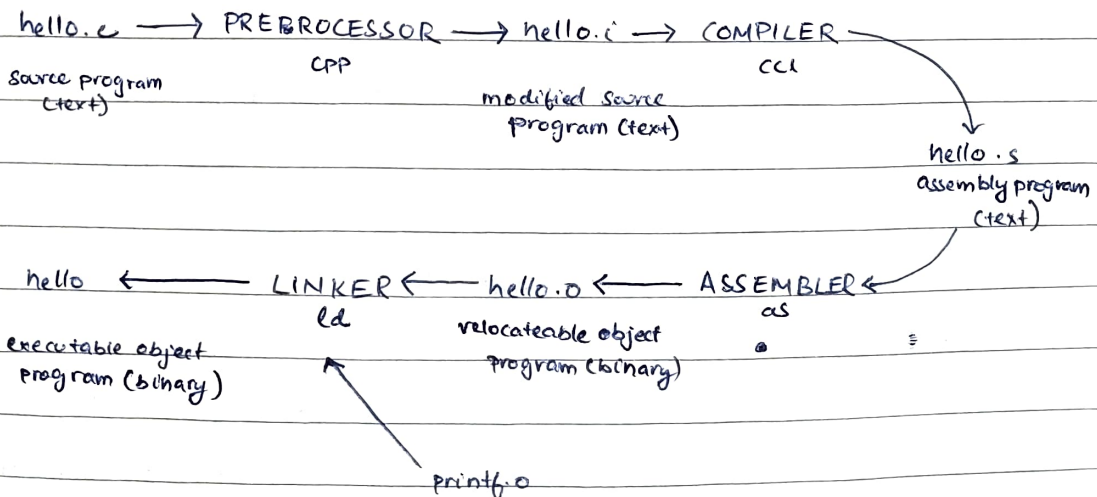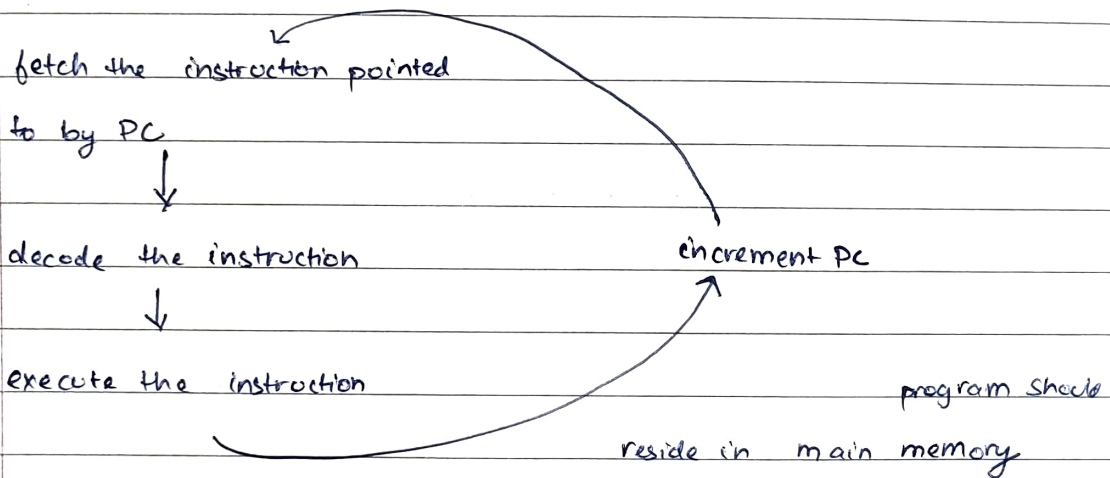
instruction set architecture ISA

hardware — software interface

advantage: multiple implementations for the same ISA.

VON NEUMANN MODEL OF COMPUTING

fetch the instruction pointed
to by PC

↓

decode the instruction

↓

execute the instruction

increment PC

program should

reside in main memory

hello.c ⟶ PREPROCESSOR ⟶ hello.i ⟶ COMPILER
source program   CPP   modified source   CCI
(text)   program (text)

hello.s
assembly program
(text)

hello ⟵ LINKER ⟵ hello.o ⟵ ASSEMBLER
         ld   relocateable object   as
executable object   program (binary)
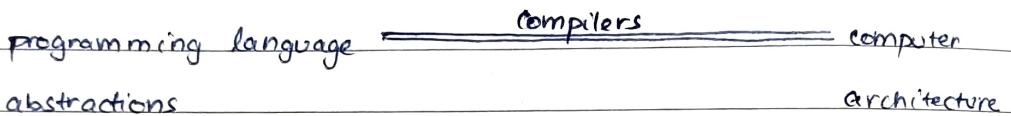program (binary)

printf.o

algorithms + data structures = programs

niklaus wirth

PROGRAMMING LANGUAGES

a programming language provides:

- data abstractions
- data processing abstraction
- control abstractions
(-) object-oriented language abstractions

programming language   Compilers   computer
abstractions   architecture

Q. what principles guide the design of language abstractions?

COMPILER

a compiler translates a program in a source language to an equivalent program in target language.

- speed   • space   • feedback   • debugging   • compile-time efficiency

PARALLELISM GRANULARITIES

instruction level parallelism: processor level parallelism:

- pipelined processors   - multicore architectures
- superscalar processors   - hyperthreading architectures
- VLIW processors

john backus talking on compilation speed (fortran).

INTERPRETER

an ea interpreter executes the operations specified by a source program on its inputs.

source program ──┐
                 └─→ INTERPRETER ──→ results
inputs ──────────┘
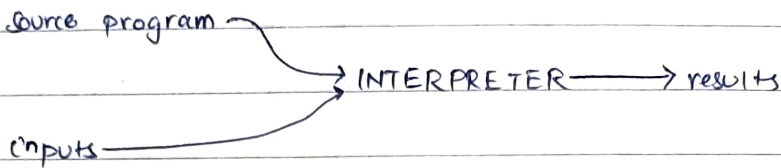
may or may not use an intermediate IR

Just In Time compilers : Java Hotspot.
- frequently called methods   ⎫
                                      ⎬ ──→ native code
- methods containing intensive loops ⎭

computer architects perpetually create new challenges for the compiler by building more complex machines.

- compilers must hide that complexity from the programmer. a successful compiler requires mastery of the many complex interactions between its constituent parts.

<u>note</u>   if compilers can hide HW complexity from the programmer, can they hide sw complexity as well?   DSLs.

compiler construction involves ideas from many different areas of computer science:

- Artificial intelligence: greedy algorithms, heuristic search techs.
- algorithms: graph algorithms, union-find, dynamic programming
- theory: DFAs & PDAs, pattern matching, fixed-point matching
- systems: allocation & naming, synchronization, locality.
- architecture: pipeline & hierarchy mgmt, instr. set use.

parallel programs require sophisticated compilers.