Introduction

Assembler is NOT just another programming language. It includes also:

- Instruction set architecture
- Exception handling
- Operating system control

Why learn this stuff?

- you will understand the underlying machine instructions when debugging a high-level program
- to write compilers you need to know assembler
- to design a processor you need to know the instruction set architecture

That is: assembly programming is a part of the whole thing, i.e. if you want to call yourself a "computer scientist" you should know assembly programming.

Computers

What is a computer?

- CPU
- memory (disk drives, DRAM, SRAM, CD)
- input (mouse, keyboard)
- output (display, printer)
- network
- software

We need levels of abstraction ...

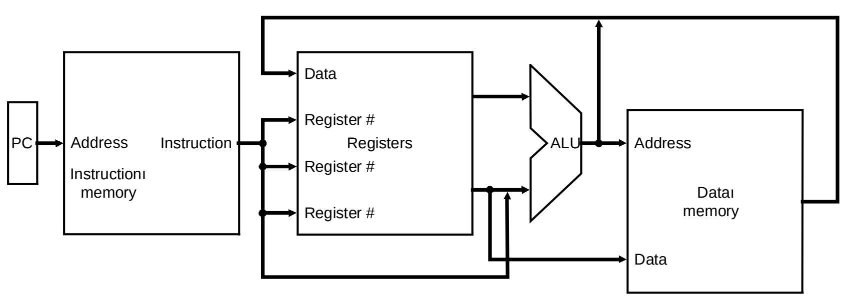
Levels of abstraction

Software:
- Application
- Operating system
- Firmware

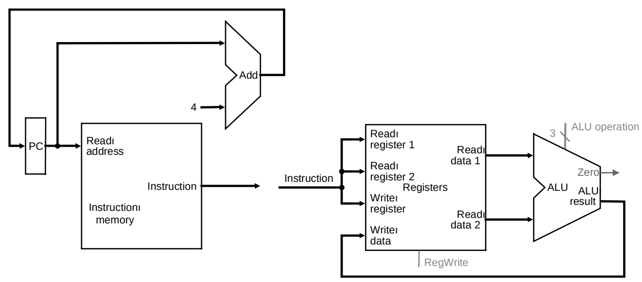
Instruction set architecture:
- Data type and structures: encodings and machine representation
- Instruction set
- Instruction formats
- Addressing modes and accessing data and instructions

Hardware:
- Instruction set processing
- I/O System
- Digital design
- Circuit design
- Layout

MIPS machine



Fetch & execute R-type instructions



Example: adding two variables

C:

- A = B + C

Assembler

- B -> $s1, C -> $s2
- add $t0, $s1, $s2
- $t0 -> A

Machine instruction:

	op	rs	rt	rd	…	funct
• decimal:	0	17	18	8	0	32
• binary:	000000	10001	10010	01000	00000	100000

Example: Datapath and control