Computing machines architecture

Dr. Giuseppe Maggiore

Hogeschool Rotterdam
Rotterdam, Netherlands

Introduction

Lecture topics

- We discuss the actual computational elements of a computer
- We bridge what we have seen in the previous lecture with actual computer architectures

Structure of a computer

Computational elements at a glance

- CPU
- Memory

CPU and instructions

CPU

- Read the current instruction from memory based on the PC
- Evaluate the instruction
  - Read and write memory elements as needed
- Write the PC of the next instruction

CPU and instructions

CPU instructions

- *Machine instructions*
- Significantly smaller than what we use
  - Register manipulation `add`, `sub`, `mul`, ...
  - Memory manipulation by integer address `lw`, `sw`
- Concrete programming languages instructions equal many machine instructions

CPU and instructions

Machine vs programming language instructions

- There are different sorts of programming languages
- Some higher level, some lower level
- Lower level languages instructions equal few (even as low as one) machine instructions

CPU and instructions

Machine vs programming language instructions

- There are different sorts of programming languages
- Some higher level, some lower level
- Lower level languages instructions equal few (even as low as one) machine instructions
- Higher level languages instructions equal many (even as high as tens) machine instructions

Memory and data

Memory

- Data is stored into memory (and also the instructions).
- Memory is just a long linear stream of bytes
- CPU queries memory by address
- CPU updates memory with address and data

Memory and data

Different kinds of memory

- There are two kinds of memory in a computer: Random Access Memory (RAM), and hard drives (HD).
- RAM is volatile: the data is lost when the computer is powered off.
- HD memory is permanent. Data remain after switching off the power.
- The memory we are referring to in this lesson is the RAM.

Different kinds of memory

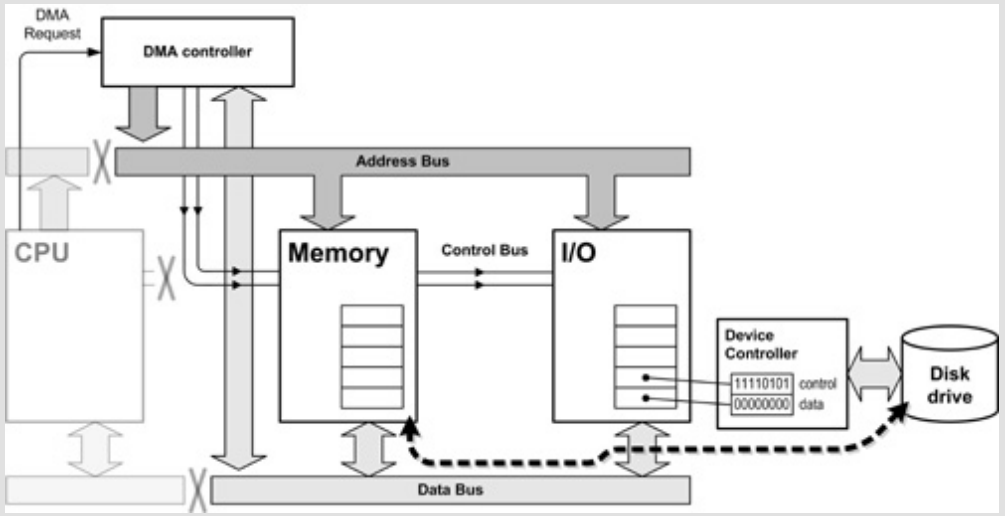

Figure: Architecture of a computer

.

Capabilities of programming languages

Typical programming language elements

- Intuitive instruction structure
- Higher level flow-control operators
  `if, while, foreach, ...`
- Labelled data through variables
  `int studentAge = 19`
- Higher level data manipulation operators
  `hypotenuse = sqrt(x * x, y * y)`

Capabilities of programming languages

Instruction names

- Machine instructions have names that are hard to pierce

Capabilities of programming languages

Instruction names

- Machine instructions have names that are hard to pierce
  - What is the meaning of instruction `0xDEBC318A`?

Capabilities of programming languages

Instruction names

- Machine instructions have names that are hard to pierce
  - What is the meaning of instruction `0xDEBC318A`?
  - What is the meaning of instruction `currentUserAge := currentUserAge + 1`?

Capabilities of programming languages

Higher level flow-control operators

- Machine instructions are tiny
- Many standardized behaviors require lots of machine instructions

Consider a *fictional* machine language listing vs its high-level equivalent:

```
lw r1 r3
cmpi r0 r3 18
jmpsz ELSE
lw r4 r3
addi r3 r3 1
sw r4 r3
jmp END
ELSE:
lw r5 r3
addi r3 r3 1
sw r5 r3
END:
...
```

```
if userAge >= 18 then
  adultUsers := adultUsers + 1
else
  youngUsers := youngUsers + 1
...
```

Capabilities of programming languages

Variables

- Program data is stored into variables
- Variables *label* memory data
- *Labels* simplify reasoning

Capabilities of programming languages

Variables

- Program data is stored into variables
- Variables *label* memory data
- *Labels* simplify reasoning
  - What is the meaning of `0xA0DF9931`?

Capabilities of programming languages

Variables

- Program data is stored into variables
- Variables *label* memory data
- *Labels* simplify reasoning
  - What is the meaning of `0xA0DF9931`?
  - What is the meaning of `userAge`?

Capabilities of programming languages

Variables and types

- Program data in memory has no fixed structure
- We can read 48 bytes instead of 32, and get 16 bytes of garbage for free
- This causes errors

Capabilities of programming languages

Variables and types

- Variables give a *type* to memory data
- *Types* simplify reasoning

Capabilities of programming languages

Variables and types

- Variables give a *type* to memory data
- *Types* simplify reasoning
  - How many bytes should I read at address `0xA0DF9931`?

Capabilities of programming languages

Variables and types

- Variables give a *type* to memory data
- *Types* simplify reasoning
  - How many bytes should I read at address `0xA0DF9931`?
  - How many bytes should I read for integer `userAge`[a]?

[a] Knowing that integers are 4 bytes on 32 bit machines and 8 bytes on 64 bit machines

Capabilities of programming languages

Yet more

- Higher level programming languages do even more
- Handle custom and complex computations (functions, events, continuations, lambda's)
- Handle custom and complex data structures (structs, classes, tuples, ...)

This is it!

The best of luck, and thanks for the attention!