Computers

Programmable machines

Outline

- Hardware and software
- Numeral systems and binary digits (bits)
- Binary truth functions and logic gates
- Computer architecture, CPU, and instructions
- Algorithm, program, and compiler

Slides available at it-course.ch/Computers.pdf

Good news

- Computers do exactly what you tell them to do.
- They do it incredibly fast.

Bad news

- Computers do exactly what you tell them to do.
- They do it incredibly fast.

Hardware and software

Hardware:

- Computers (physical)
- General purpose machines
- Expensive to design
- Expensive to copy
- Subjected to wear
- Difficult to fix once shipped

Software:

- Programs (intangible)
- Special purpose instructions
- Expensive to design
- Free to copy
- Wear-free
- Never really finished, requires maintenance

Numeral systems

- **Decimal numeral system**: ten symbols, called digits (from Latin "digitus"): 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9. Example: $123 = 1 \cdot 10^2 + 2 \cdot 10^1 + 1 \cdot 10^0$.
- **Binary numeral system**: only two symbols, called bits (short for "binary digits"): 0 and 1. Example: $1011 = 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0$. 8 bits = 1 byte.
- **Task**: Count and add in binary.

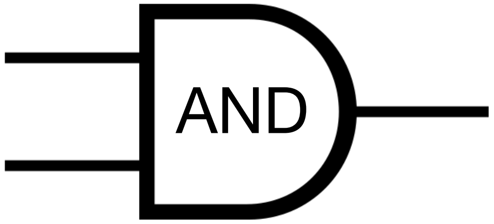
Truth functions with truth tables

A	B	A **AND** B
0	0	0
0	1	0
1	0	0
1	1	1

A	B	A **OR** B
0	0	0
0	1	1
1	0	1
1	1	1

A	B	A **XOR** B
0	0	0
0	1	1
1	0	1
1	1	0

Implemented in hardware as logic gates usually with transistors. Depicted as:

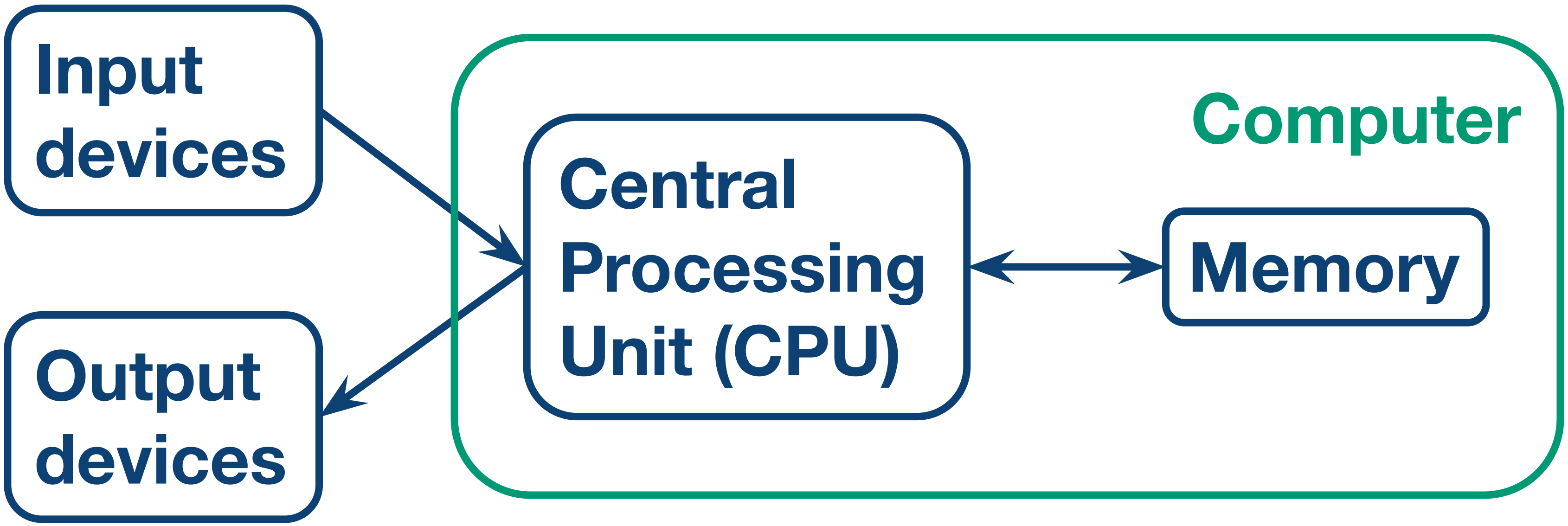


Tasks

1. Design logic gates which implement these binary truth functions using water (1 as on, 0 as off) coming from two pipes collected in a third pipe. (The simplicity of logic gates is why all modern computers use binary representations.)
2. Design an electronic circuit using logic gates which forms the sum S and the carry C from two input bits A and B.
3. Inputs: A, B, and C_{in}; outputs: S and C_{out}

Solutions: visualization, half adder, full adder

Computer (heavily simplified)



- There is volatile and non-volatile memory.
- Performance is increased with caches.
- Also: Graphics Processing Unit (GPU).

Instructions (heavily simplified)

The CPU stores its data in so-called registers.

The CPU executes five types of instructions:

- Load data from memory to registers,
- Store data from registers to memory,
- Perform arithmetic operations on registers,
- Perform logical operations on registers, and
- Jump to a given instruction, which can depend on so-called flags set by the previous operation.

Example (simplified)

```
        mov 0, sum
        mov 1, num
loop:   add num, sum
        add 1, num
        cmp num, 100
        ble loop
        halt
```

Explanation: mov: move; cmp: compare;
ble: branch if less or equal

Question: What is the above code doing?

Algorithm

An algorithm is a procedure for solving a specified problem in a finite number of steps, i.e. it has to produce an output eventually.

Examples: Sorting, finding the shortest path, etc.

A program is a list of instructions that can be run.

Compiler

A compiler is a program which translates code from a source language into a target language.

A codebase can be compiled to different targets.

For now: source code → machine code, which is usually displayed in an assembly language.

Example (real deal)

Sum all numbers from 1 to 100 in the C programming language.

Compiled with the GNU compiler collection `gcc -S source.c` to x86 AT&T assembly syntax.

```
#include <stdio.h>
int main() {
  int result = 0;
  int number = 1;
  while (number <= 100) {
    result = result + number;
    number = number + 1;
  }
  printf("Result: %d\n", result);
  return 0;
}
```

⇨

```
_main:
  pushq %rbp
  movq  %rsp, %rbp
  subq  $16, %rsp
  movl  $0, -4(%rbp)
  movl  $0, -8(%rbp)
  movl  $1, -12(%rbp)
LBB0_1:
  cmpl  $100, -12(%rbp)
  jg    LBB0_3
  movl  -8(%rbp), %eax
  addl  -12(%rbp), %eax
  movl  %eax, -8(%rbp)
  movl  -12(%rbp), %eax
  addl  $1, %eax
  movl  %eax, -12(%rbp)
  jmp   LBB0_1
LBB0_3:
  movl  -8(%rbp), %esi
  leaq  L_.str(%rip), %rdi
  movb  $0, %al
  callq _printf
  xorl  %eax, %eax
  addq  $16, %rsp
  popq  %rbp
  retq
L_.str:
  .asciz "Result: %d\n"
```

Explanations

- Labels end with `:` (LBB: local block begin).
- Literal values start with `$`, registers with `%`.
- Suffix of commands determines the bit-length: **b** for byte (1 byte = 8 bits), **w** for word (2 bytes = 16 bits), **l** for double word (4 bytes = 32 bits), and **q** for quad word (8 bytes = 64 bits).
- Program flow: `jmp` means jump unconditionally; `jg` means jump on greater than based on the `cmp` (comparison) in the previous instruction.

Abstraction

Managing registers and memory (allocation and deallocation of space for variables) is a hassle.

This is why almost all programs are written in high-level programming languages, which abstract from the details of the current computing platform.

Low-level aspects are then handled by a compiler.

You find more information about assembly here.

Example: Prime factorization

```
#include <stdio.h>                        // Comments:
int main() {
   int number;                            // Variable declaration
   printf("Number: ");                    // Print to standard output
   scanf("%d", &number);                  // Read a number from input
   while (number > 0) {                   // Loop while condition true
      printf("Factors: ");
      int factor = 2;                     // Assign value to variable
      while (factor * factor <= number) {
         if (number % factor == 0) {      // Check remainder (modulo)
            printf("%d, ", factor);
            number = number / factor;
         } else {
            factor = factor + 1;
         }
      }
      printf("%d\n", number);
      printf("Number: ");
      scanf("%d", &number);
   }
   return 0;
}
```

Task: Come up with an algorithm

The greatest common divisor of two integers is the largest positive integer which divides both integers without a remainder.

Geometric interpretation: Largest size of square tile which tiles a rectangle/room without remainders.

Task: Come up with an algorithm which finds the size of this square tile. Example: gcd(51, 21) = 3.

Solution: Euclidean algorithm