Outline

1. Digital Systems
   - transistors
   - logic gates
2. Dictionaries and Conditional Statements
   - intrinsic operations
   - writing numbers in words

MCS 260 Lecture 9
Introduction to Computer Science
Jan Verschelde, 23 June 2023

Digital Systems

introduction to electronic circuits

A computer is a synchronous binary digital system.

digital: all information is discrete (not continuous)

binary: only zero and one are used
a binary digit is a bit

synchronous: functioning is ruled by the system clock

Basic elements to represent bits are switches that can be open (1) or closed (0).

Transistors are electronic circuits to represent bits.

transistors and gates
intrinsic operations

1. Digital Systems
   - transistors
   - logic gates

2. Dictionaries and Conditional Statements
   - intrinsic operations
   - writing numbers in words

Transistors

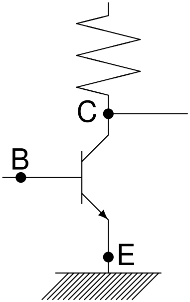
electronic circuits to represent bits

Transistors have three connections to the outside:

1. base: input voltage
2. collector: output voltage
3. emittor: to ground

High Voltage: 1

Low Voltage: 0



transistors and gates
intrinsic operations

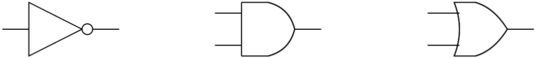
Logic Gates

implement logic operators

Logic gates are circuits that correspond to logic operators.

Representations of NOT, AND, OR:

NOT AND OR



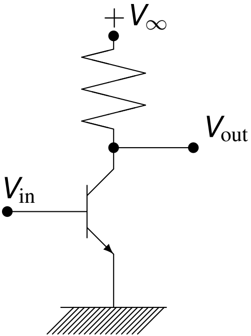
x NAND y
= NOT (x AND y)

x NOR y
= NOT (x OR y)

NAND NOR

A NOT Gate

as realized by a transistor



Input Voltage V_{in}

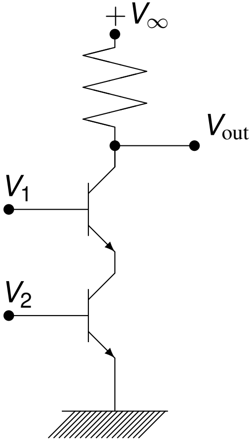
$V_{\mathrm{in}} =$ low
$\Rightarrow$ switch is open
$\Rightarrow V_{\mathrm{out}} = +V_\infty$

$V_{\mathrm{in}} =$ high
$\Rightarrow$ switch is closed
$\Rightarrow V_{\mathrm{out}} =$ low

A NOT gate converts a low input voltage to high and a high input voltage to low.

A NAND Gate

two transistors in series



Input voltages V_1 and V_2

If either V_1 or V_2 is low:

$\Rightarrow$ switch is open

$\Rightarrow V_{out} = +V_\infty$

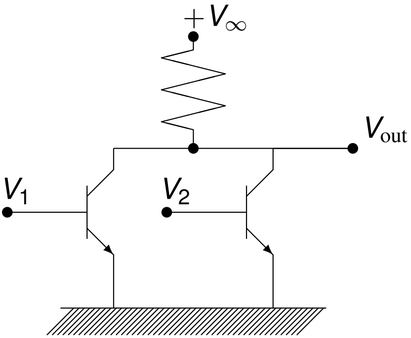
If both V_1 and V_2 are high:

$\Rightarrow$ switch is closed

$\Rightarrow V_{out} =$ low

A NOR Gate

two transistors in parallel



Input voltages V_1 and V_2
if either V_1 or V_2 is high $\Rightarrow$ closed switch $\Rightarrow$ $V_{\rm out} = $ low;
if both V_1 or V_2 are low $\Rightarrow$ open switch $\Rightarrow$ $V_{\rm out} = +V_\infty$.

transistors and gates
intrinsic operations

intrinsic operations

Intrinsic operations are those operations that belong to the standard library.

For every variable `x`, the function

- id(`x`) returns the address of `x`,
- type(`x`) returns the type of `x`.

Python has dynamic typing and garbage collection.

To see the operations defined on strings, do

```
>>> dir(str)
```

which returns the list of operations defined on strings. Then,

```
>>> help(str.split)
```

shows the help on the `split()` method on strings.

transistors and gates
intrinsic operations

1. Digital Systems
   - transistors
   - logic gates

2. Dictionaries and Conditional Statements
   - intrinsic operations
   - writing numbers in words

writing numbers in words – applying dictionaries

On a check, the amount is spelled out in words.

Program specification: Input: n, a natural number < 1000.
Output: a string expressing n in words.

An example session with `write_numbers.py`:

```
$ python write_numbers.py
give a natural number : 125
125 is one hundred and twenty five
```

the dictionary: numbers spelled out in English

For all $n \leq 20$ and multiples of 10:

```
DIC = { \
    0:'zero', 1:'one', 2:'two', 3:'three', \
    4:'four', 5:'five', 6:'six', 7:'seven', \
    8:'eight', 9:'nine', 10:'ten', \
    11:'eleven', 12:'twelve', 13:'thirteen', \
    14:'fourteen', 15:'fifteen', 16:'sixteen', \
    17:'seventeen', 18: 'eighteen', 19:'nineteen', \
    20:'twenty', 30:'thirty', 40:'forty', \
    50:'fifty', 60:'sixty', 70:'seventy', \
    80:'eighty', 90:'ninety', 100:'hundred' \
}
```

The dictionary lookup `DIC[n]` handles special cases.

idea for the algorithm

case analysis

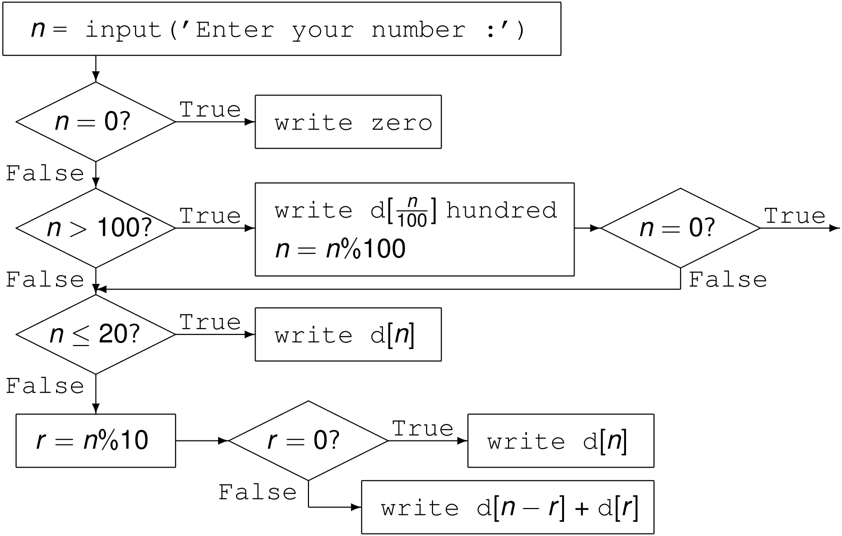
We distinguish three cases:

1. the trivial case: $n = 0$
   This is the only case we write `zero`.

2. large numbers $n \geq 100$
   We start writing $n/100$ `hundred`
   and then continue with

3. the rest: $0 < n < 100$:
   1. for $n \leq 20$: dictionary lookup
   2. for $20 < n < 100$: compute $r = n\%10$ and $n - r$

flowchart for `write_numbers.py`



Exercises

1. Draw all transistors needed to realize an OR gate and describe its working.
2. Construct truth tables for
    1. (A OR B) OR NOT (A AND B)
    2. NOT ((A OR C) OR B) OR (A AND C)
3. Draw the logic gates to realize the expressions of the previous exercise.
4. Let `secret` be a secret number the user of a Python program has to guess. Give code for prompting the user for a guess and for printing feedback.
5. Write a script to use `dbm` to store the dictionary `d` to spell numbers out in English.
6. Modify the `write_numbers.py` program so it uses the `dbm` file made in the previous exercise.