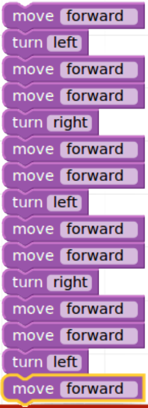


Software

ver. 14 z drobnymi modyfikacjami!

Wojciech Myszka

2023-11-06 17:48:24 +0100

HR EXCELLENCE IN RESEARCH

Wrocław University of Science and Technology

What determines the speed of computers?

What determines the speed of computers?

1. Clock frequency: Intel, AMD. Now about 4+ GHz (max 5.7 GHz)

What determines the speed of computers?

1. Clock frequency: Intel, AMD. Now about 4+ GHz (max 5.7 GHz)
2. Memory “speed”.

What determines the speed of computers?

1. Clock frequency: Intel, AMD. Now about 4+ GHz (max 5.7 GHz)
2. Memory “speed”.
3. Word length:

What determines the speed of computers?

1. Clock frequency: Intel, AMD. Now about 4+ GHz (max 5.7 GHz)
2. Memory “speed”.
3. Word length:
   - short word
     - simpler design
     - faster transfer to memory (less data!)
     - longer processing of extended data

What determines the speed of computers?

1. Clock frequency: Intel, AMD. Now about 4+ GHz (max 5.7 GHz)
2. Memory “speed”.
3. Word length:
   - short word
     - simpler design
     - faster transfer to memory (less data!)
     - longer processing of extended data
   - long word
     - more complicated design
     - wasting of resources (sometimes)
     - faster operation on long numbers

What determines the speed of computers?

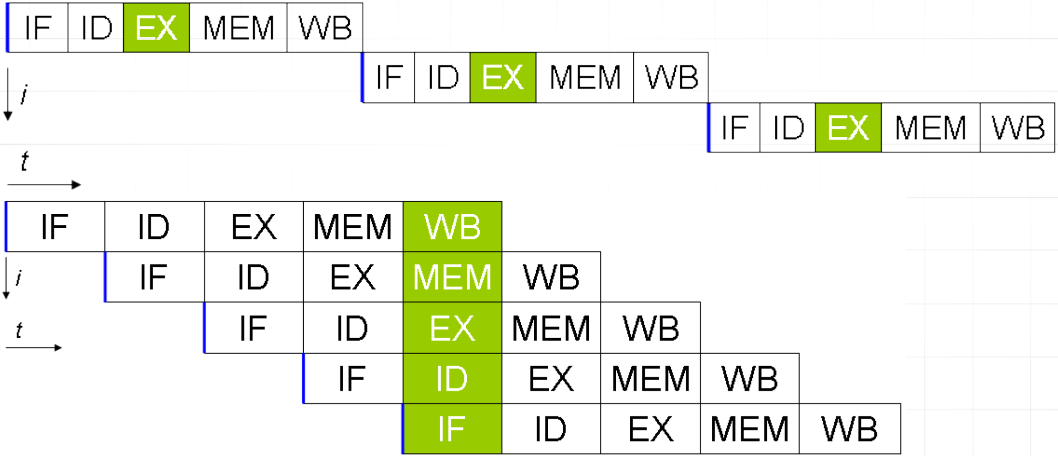
1. Clock frequency: Intel, AMD. Now about 4+ GHz (max 5.7 GHz)
2. Memory “speed”.
3. Word length:
   - short word
     - simpler design
     - faster transfer to memory (less data!)
     - longer processing of extended data
   - long word
     - more complicated design
     - wasting of resources (sometimes)
     - faster operation on long numbers
4. Computer design
   - number of arithmetic units
   - way of performing arithmetic operations

Pipelining

Pipeline

Pipelining

Pipeline



IF — indtruction fetch, **ID** — instruction decode, **EX** — execution, **MEM** — storing results into cache, **WB** — writing back (from cache to memory)

Pipelining

Pipeline + two processors/cores

i ↓, t →

IF	ID	EX	MEM	WB				
IF	ID	EX	MEM	WB				
	IF	ID	EX	MEM	WB			
	IF	ID	EX	MEM	WB			
		IF	ID	EX	MEM	WB		
		IF	ID	EX	MEM	WB		
			IF	ID	EX	MEM	WB	
			IF	ID	EX	MEM	WB	
				IF	ID	EX	MEM	WB
				IF	ID	EX	MEM	WB

IF — indtruction fetch, **ID** — instruction decode, **EX** — execution, **MEM** — storing results into cache, **WB** — writing back (from cache to memory)

What else determines the speed of computing?

Vector processors

1. Vector (array) processor has instructions allowing to perform operations on one dimensional arrays of data. This means that at the same time it performs several operations at once.
2. This is called SIMD — Single Instruction, Multiple Data
3. Basis of a “supercomputers” from 80 and 90.
4. In 2000, IBM, Toshiba and Sony worked together on the development of the Cell processor containing one scalar processor (the inverse of a vector processor) and eight vector processors, which it was used (among other things) in the PlayStation 3.

Various abbreviations

1. CISC

Various abbreviations

1. CISC *Complex Instruction Set Computer*
2. RISC

Various abbreviations

1. CISC *Complex Instruction Set Computer*
2. RISC *Reduced Instruction Set Computer*
3. VLIW

Various abbreviations

1. CISC *Complex Instruction Set Computer*
2. RISC *Reduced Instruction Set Computer*
3. VLIW *Very Long Instruction Word*
4. EPIC

Various abbreviations

1. CISC *Complex Instruction Set Computer*
2. RISC *Reduced Instruction Set Computer*
3. VLIW *Very Long Instruction Word*
4. EPIC *Explicitly Parallel Instruction Computing*

Homework: Read about this abbreviations!

Various abbreviations

1. CISC *Complex Instruction Set Computer*
2. RISC *Reduced Instruction Set Computer*
3. VLIW *Very Long Instruction Word*
4. EPIC *Explicitly Parallel Instruction Computing*

Homework: Read about this abbreviations!

Various abbreviations

1. **x86** The most popular architecture of PC computers (32 bit version is now obsolete)

Various abbreviations

1. **x86** The most popular architecture of PC computers (32 bit version is now obsolete)
2. **x86-64** 64-bit architecture introduced by AMD (extension of x86)

Various abbreviations

1. **x86** The most popular architecture of PC computers (32 bit version is now obsolete)
2. **x86-64** 64-bit architecture introduced by AMD (extension of x86)
3. **ARM** a family of reduced instruction set computing (RISC) architectures for computer processors, configured for various environments.
   Apple new processors (M1, M2) belongs to this family.
   ARM means Advanced RISC Machines

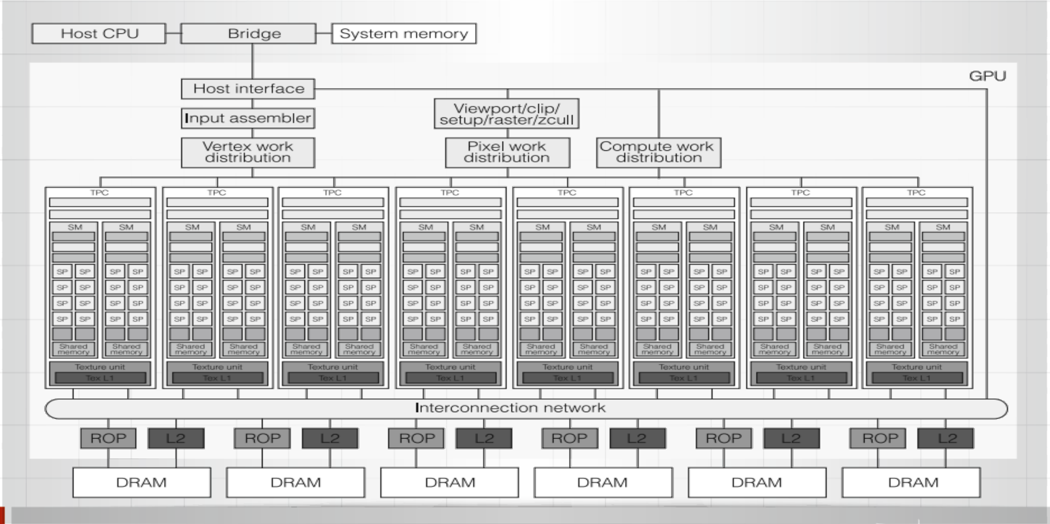
Various abbreviations

1. **x86** The most popular architecture of PC computers (32 bit version is now obsolete)
2. **x86-64** 64-bit architecture introduced by AMD (extension of x86)
3. **ARM** a family of reduced instruction set computing (RISC) architectures for computer processors, configured for various environments.
   Apple new processors (M1, M2) belongs to this family.
   ARM means Advanced RISC Machines
4. **RISC-V** is an open standard instruction set architecture (ISA) based on established reduced instruction set computer (RISC) principles. Unlike most other ISA designs, RISC-V is provided under royalty-free open-source licenses. Becomes more and more popular

Various abbreviations

1. **x86** The most popular architecture of PC computers (32 bit version is now obsolete)
2. **x86-64** 64-bit architecture introduced by AMD (extension of x86)
3. **ARM** a family of reduced instruction set computing (RISC) architectures for computer processors, configured for various environments.
   Apple new processors (M1, M2) belongs to this family.
   ARM means Advanced RISC Machines
4. **RISC-V** is an open standard instruction set architecture (ISA) based on established reduced instruction set computer (RISC) principles. Unlike most other ISA designs, RISC-V is provided under royalty-free open-source licenses. Becomes more and more popular
5. **CUDA** (Compute Unified Device Architecture) is a parallel computing platform and application programming interface (API) model created by Nvidia.

NVIDIA CUDA


Host CPU
Bridge
System memory
GPU
Host interface
Input assembler
Viewport/clip/
setup/raster/zcull
Vertex work
distribution
Pixel work
distribution
Compute work
distribution
TPC
SM
SP
Shared
memory
Texture unit
Tex L1
Interconnection network
ROP
L2
DRAM

Homework

Read something about all this mentioned acronyms.

Why does computer work?

1. What is a computer?

Why does computer work?

1. What is a computer?
2. Kind of a calculator (it has an arithmetic unit).

Why does computer work?

1. What is a computer?
2. Kind of a calculator (it has an arithmetic unit).
3. it has a memory. . .

Why does computer work?

1. What is a computer?
2. Kind of a calculator (it has an arithmetic unit).
3. it has a memory. . .
4. . . . but what turns it into operation?

Why does computer work?

1. What is a computer?
2. Kind of a calculator (it has an arithmetic unit).
3. it has a memory. . .
4. . . . but what turns it into operation?
5. Program

Why does computer work?

1. What is a computer?
2. Kind of a calculator (it has an arithmetic unit).
3. it has a memory…
4. …but what turns it into operation?
5. Program

Why does computer work?

1. What is a computer?
2. Kind of a calculator (it has an arithmetic unit).
3. it has a memory. . .
4. . . . but what turns it into operation?
5. Program?

Turn on the computer. . .

. . . and what happens?

1. When everything is OK processor automatically tries to execute the program from a specified part of the memory.

Turn on the computer…

…and what happens?

1. When everything is OK processor automatically tries to execute the program from a specified part of the memory.
2. But, there **must be** some program (code) in this memory…

Turn on the computer…

…and what happens?

1. When everything is OK processor automatically tries to execute the program from a specified part of the memory.
2. But, there **must be** some program (code) in this memory…
3. Typically, in this memory area is a “permanent memory” (*Read-Only Memory* — ROM, NVRAM)…

Turn on the computer…

…and what happens?

1. When everything is OK processor automatically tries to execute the program from a specified part of the memory.
2. But, there **must be** some program (code) in this memory…
3. Typically, in this memory area is a “permanent memory” (*Read-Only Memory* — ROM, NVRAM)…
4. …containing program called BIOS (*Basic Input Output System*) replaced now by *Unified Extensible Firmware Interface* (UEFI).

Turn on the computer…

…and what happens?

1. When everything is OK processor automatically tries to execute the program from a specified part of the memory.
2. But, there **must be** some program (code) in this memory…
3. Typically, in this memory area is a “permanent memory” (*Read-Only Memory* — ROM, NVRAM)…
4. …containing program called BIOS (*Basic Input Output System*) replaced now by *Unified Extensible Firmware Interface* (UEFI).
5. BIOS/UEFI checks all components of the computer…

Turn on the computer…

…and what happens?

1. When everything is OK processor automatically tries to execute the program from a specified part of the memory.
2. But, there **must be** some program (code) in this memory…
3. Typically, in this memory area is a “permanent memory” (*Read-Only Memory* — ROM, NVRAM)…
4. …containing program called BIOS (*Basic Input Output System*) replaced now by *Unified Extensible Firmware Interface* (UEFI).
5. BIOS/UEFI checks all components of the computer…
6. UEFI checks integrity of the Operating System …

Turn on the computer. . .

. . . and what happens?

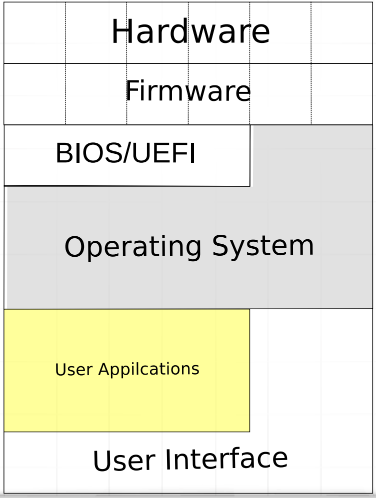
1. When everything is OK processor automatically tries to execute the program from a specified part of the memory.
2. But, there **must be** some program (code) in this memory. . .
3. Typically, in this memory area is a “permanent memory” (*Read-Only Memory* — ROM, NVRAM). . .
4. . . . containing program called BIOS (*Basic Input Output System*) replaced now by *Unified Extensible Firmware Interface* (UEFI).
5. BIOS/UEFI checks all components of the computer. . .
6. UEFI checks integrity of the Operating System . . .
7. . . . and loads an Operating System from the disc.

Turn on the computer…

…and what happens?

1. When everything is OK processor automatically tries to execute the program from a specified part of the memory.
2. But, there **must be** some program (code) in this memory…
3. Typically, in this memory area is a “permanent memory” (*Read-Only Memory* — ROM, NVRAM)…
4. …containing program called BIOS (*Basic Input Output System*) replaced now by *Unified Extensible Firmware Interface* (UEFI).
5. BIOS/UEFI checks all components of the computer…
6. UEFI checks integrity of the Operating System …
7. …and loads an Operating System from the disc.
8. Operating System runs applications.

BIOS

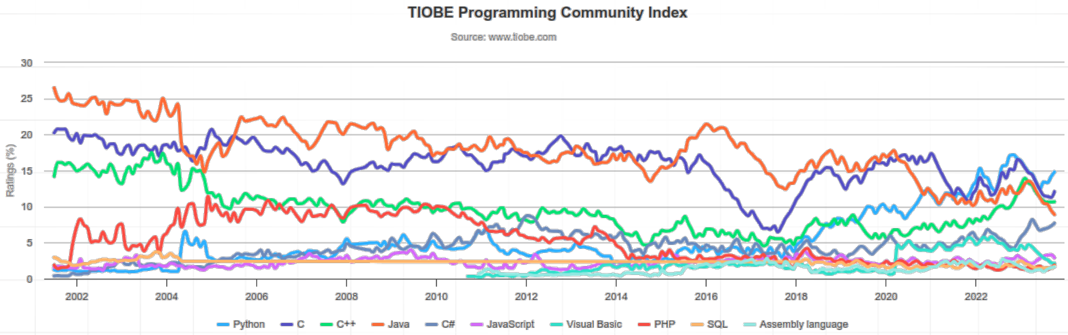

Hardware
Firmware
BIOS/UEFI
Operating System
User Appilcations
User Interface

Programs

- Software (and its quality) influences the effective speed of computers.
- What is computer programming?

Are the programming skills important?

Programming languages



Simple tasks

The sum of numbers

The task is that we have to add, say, 1000 numbers (provided on paper). How to do it:

Simple tasks

The sum of numbers

The task is that we have to add, say, 1000 numbers (provided on paper). How to do it:

- “by hand”?

Simple tasks

The sum of numbers

The task is that we have to add, say, 1000 numbers (provided on paper). How to do it:

- "by hand"?
- by hand with help of a calculator?

Simple tasks

The sum of numbers

The task is that we have to add, say, 1000 numbers (provided on paper). How to do it:

- “by hand”?
- by hand with help of a calculator?
- using some application?

Simple tasks

The sum of numbers

The task is that we have to add, say, 1000 numbers (provided on paper). How to do it:

- "by hand"?
- by hand with help of a calculator?
- using some application?
- using a self-made computer program?

More advanced technical problem

Period of oscillation of a pendulum

is given by the equation

$$T = 2\pi\sqrt{\frac{l}{g}}$$

More advanced technical problem

Period of oscillation of a pendulum

is given by the equation

$$T = 2\pi\sqrt{\frac{l}{g}}$$

- Let say that we have 100 values of l

More advanced technical problem

Period of oscillation of a pendulum

is given by the equation

$$T = 2\pi\sqrt{\frac{l}{g}}$$

- Let say that we have 100 values of l
  - by hand?? (difficult without a calculator)

More advanced technical problem

Period of oscillation of a pendulum

is given by the equation

$$T = 2\pi\sqrt{\frac{l}{g}}$$

- Let say that we have 100 values of l
  - by hand?? (difficult without a calculator)
  - develop an application?

More advanced technical problem

Period of oscillation of a pendulum

is given by the equation

$$T = 2\pi\sqrt{\frac{l}{g}}$$

- Let say that we have 100 values of l
  - by hand?? (difficult without a calculator)
  - develop an application?
  - use a spreadsheet?

More advanced technical problem

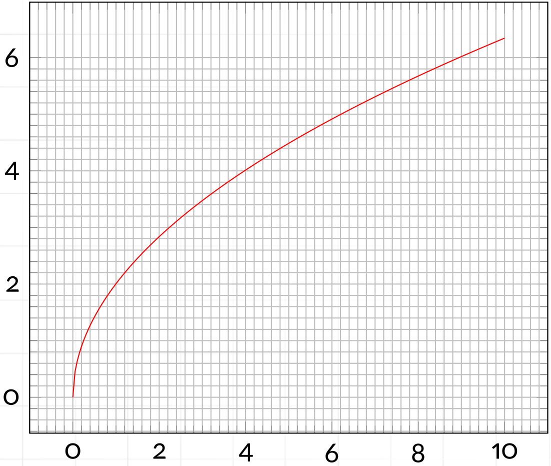
Period of oscillation of a pendulum

is given by the equation

$$T = 2\pi\sqrt{\frac{l}{g}}$$

- Let say that we have 100 values of l
  - by hand?? (difficult without a calculator)
  - develop an application?
  - use a spreadsheet?
  - plot the function, using, for example, Gnuplot?

Plot



Maze

Problem statement

- we have the simplest maze

Maze

Problem statement

- we have the simplest maze
- there is an “entry”

Maze

Problem statement

- we have the simplest maze
- there is an “entry”
- there is an “exit”

Maze

Problem statement

- we have the simplest maze
- there is an “entry”
- there is an “exit”
- you have to find a way from the entry to the exit…

More complicated problem

Maze

Programming language: Google Blockly

There are two versions:

1. Blockly Games
2. Google Blockly is a visual programming editor.

Programming language: Google Blockly

There are two versions:

1. Blockly Games Blockly Games
2. Google Blockly Blockly is a visual programming editor.

- Can be used on-line: https://blockly.games/
- Can be downloaded to ones computer https://github.com/google/blockly-games/wiki/Offline
  - unpack in some directory
  - and find file `/blockly-read-only/demos/index.html` in that directory and open it in web browser.

Maze

How to solve the maze?

- You can direct Pegman (tell the way) to find the exit (example in the browser).

Maze

How to solve the maze?

- You can direct Pegman (tell the way) to find the exit (example in the browser).
- Random turns: go to crossing and randomly choose a direction.

Maze

How to solve the maze?

- You can direct Pegman (tell the way) to find the exit (example in the browser).
- Random turns: go to crossing and randomly choose a direction.
- Homework: how to realize this strategy in Blockly?

Maze

How to solve the maze?

- You can direct Pegman (tell the way) to find the exit (example in the browser).
- Random turns: go to crossing and randomly choose a direction.
- Homework: how to realize this strategy in Blockly?
- Left-/right- hand walk: follow the wall touching it using your left/right hand.

Greatest Common Divisor

Problem statement

- There are two natural (whole) numbers m and n, such that $m > 0, n > 0$.

Greatest Common Divisor

Problem statement

- There are two natural (whole) numbers m and n, such that $m > 0, n > 0$.
- We are searching for x which divides both m and n and is the greatest of all such dividers. In other words: the largest positive integer that divides the numbers without a remainder.

Greatest Common Divisor

Problem statement

- There are two natural (whole) numbers m and n, such that $m > 0, n > 0$.
- We are searching for x which divides both m and n and is the greatest of all such dividers. In other words: the largest positive integer that divides the numbers without a remainder.
- What the reminder is?

Greatest Common Divisor

The simple algorithm “from the definition”

- find all divisors of the first number.

Greatest Common Divisor

The simple algorithm “from the definition”

- find all divisors of the first number.
- find all divisors of the second number.

Greatest Common Divisor

The simple algorithm “from the definition”

- find all divisors of the first number.
- find all divisors of the second number.
- find common numbers (divisors)

Greatest Common Divisor

The simple algorithm “from the definition”

- find all divisors of the first number.
- find all divisors of the second number.
- find common numbers (divisors)
- find the greatest one.

Find all divisors

- check if 1 divides n
- check if 2 divides n
- $\dots$
- check if $n - 1$ divides n

Finding all divider

Can we simplify this

- It is enough to start from 2 (all numbers are divided by 1)

Finding all divider

Can we simplify this

- It is enough to start from 2 (all numbers are divided by 1)
- When stopping this procedure?

Finding all divider

Can we simplify this

- It is enough to start from 2 (all numbers are divided by 1)
- When stopping this procedure?
- It is enough to finish at $\sqrt{n}$ (any whole number close to $\sqrt{n}$).

The intersection of two sets

1. Take the first object from the set N (divisors of n).

The intersection of two sets

1. Take the first object from the set N (divisors of n).
2. Check if it belongs to the set M?

The intersection of two sets

1. Take the first object from the set *N* (divisors of *n*).
2. Check if it belongs to the set *M*?
3. If so, put it to the resulting set *X*

The intersection of two sets

1. Take the first object from the set N (divisors of n).
2. Check if it belongs to the set M?
3. If so, put it to the resulting set X
4. If you have not passed through all the objects in the set N, take the next element and go to step 2.

Searching for the maximum element

Searching for the maximum element

1. Take the first object from the set. It will be the **current maximum value**.

Searching for the maximum element

1. Take the first object from the set. It will be the **current maximum value**.
2. Are there any objects left in the set? *If* NO, *then* STOP.

Searching for the maximum element

1. Take the first object from the set. It will be the **current maximum value**.
2. Are there any objects left in the set? *If* NO, *then* STOP.
3. *else* take the next one

Searching for the maximum element

1. Take the first object from the set. It will be the **current maximum value**.
2. Are there any objects left in the set? *If* NO, *then* STOP.
3. *else* take the next one
4. is it greater than the **current maximum value**?

Searching for the maximum element

1. Take the first object from the set. It will be the **current maximum value**.
2. Are there any objects left in the set? *If* NO, *then* STOP.
3. *else* take the next one
4. is it greater than the **current maximum value**?
5. if NO, then *go to* step 2

Searching for the maximum element

1. Take the first object from the set. It will be the **current maximum value**.
2. Are there any objects left in the set? *If* NO, *then* STOP.
3. *else* take the next one
4. is it greater than the **current maximum value**?
5. if NO, then *go to* step 2
6. else, it will be the **current maximum value**.

Euclidean algorithm

E1. Let r be the remainder from the division of m by n

Euclidean algorithm

E1. Let r be the remainder from the division of m by n

E2. If $r = 0$ STOP. The solution is n.

Euclidean algorithm

E1. Let r be the remainder from the division of m by n

E2. If $r = 0$ STOP. The solution is n.

E3. Else

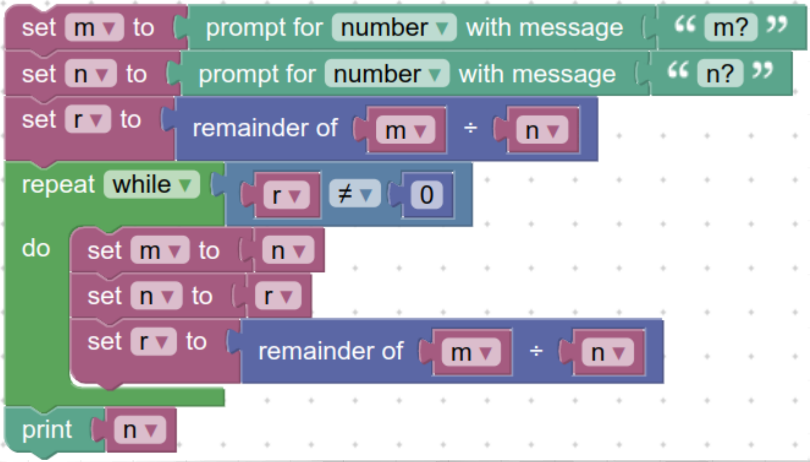
$m \leftarrow n$

$n \leftarrow r$

Go to E1

GCD

Blockly implementation



Homework

- Find other variants of Euclidean algorithm…
- …and program it in Blockly

Algorithm B

(Binary greatest common divisor algorithm)

1. $k \leftarrow 0$
2. while u is even and v is even
   $u \leftarrow u/2$
   $v \leftarrow v/2$
   $k \leftarrow k + 1$
   now u or v (or both) are odd
3. if u is odd, let $t \leftarrow -v$ and go to step 5 else let $t \leftarrow u$
4. (At this point t is even and not equal 0.) Let $t \leftarrow t/2$
5. If t is even then go to step 4
6. If $t > 0$ then let $u \leftarrow t$ else let $v \leftarrow -t$.
7. Let $t \leftarrow u - v$. If $t \neq 0$ then goto 4. Else, the result is $u \cdot 2^k$

Homework?

Program Algorithm B in Blockly. . . ?

Homework?

Program Algorithm B in Blockly. . . ?

Ups. . . probably to difficult. . . !

Homework?

Program Algorithm B in Blockly. . . ?

Ups. . . probably to difficult. . . !

Solve it for chosen *u* and *v* (both less then 1000) “by hand”: using paper and pencil.

Bibliography

- Suits D.B., *Playing with mazes*, URL https://davidsuits.net/PlayingWithMazes.pdf 1994.
- Pullen W.D., *Maze classification*, URL http://www.astrolog.org/labyrnth/algrithm.htm 2015.
- Pullen W.D., *Technical maze terms*, URL http://www.astrolog.org/labyrnth/glossary.htm 2015.
- Pullen W.D., *Making difficult mazes*, URL http://www.astrolog.org/labyrnth/psych.htm 2015.