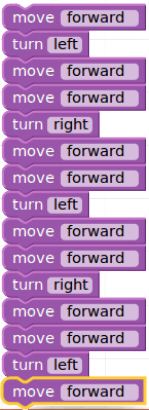


Oprogramowanie komputerów

wer. 14 z drobnymi modyfikacjami!

Wojciech Myszka

2023-11-06 17:48:15 +0100

Politechnika Wrocławska

Od czego zależy szybkość komputerów?

Od czego zależy szybkość komputerów?

1. Częstość zegara: Intel, AMD, obecnie ponad 4 GHz (max. 5+ GHz)

Od czego zależy szybkość komputerów?

1. Częstość zegara: Intel, AMD, obecnie ponad 4 GHz (max. 5+ GHz)
2. Szybkość pamięci.

Od czego zależy szybkość komputerów?

1. Częstość zegara: Intel, AMD, obecnie ponad 4 GHz (max. 5+ GHz)
2. Szybkość pamięci.
3. Długość słowa:
   - krótkie słowo:
     - prostsza budowa,
     - szybsze przesyłanie do pamięci,
     - dłuższe operacja na długich liczbach,

Od czego zależy szybkość komputerów?

1. Częstość zegara: Intel, AMD, obecnie ponad 4 GHz (max. 5+ GHz)
2. Szybkość pamięci.
3. Długość słowa:
   - krótkie słowo:
     - prostsza budowa,
     - szybsze przesyłanie do pamięci,
     - dłuższe operacja na długich liczbach,
   - długie słowo:
     - bardziej złożona budowa,
     - czasami marnotrawstwo zasobów,
     - szybkie wykonywanie operacji na długich liczbach.

Od czego zależy szybkość komputerów?

1. Częstość zegara: Intel, AMD, obecnie ponad 4 GHz (max. 5+ GHz)
2. Szybkość pamięci.
3. Długość słowa:
   - krótkie słowo:
     - prostsza budowa,
     - szybsze przesyłanie do pamięci,
     - dłuższe operacja na długich liczbach,
   - długie słowo:
     - bardziej złożona budowa,
     - czasami marnotrawstwo zasobów,
     - szybkie wykonywanie operacji na długich liczbach.
4. Wewnętrzna konstrukcja komputera:
   - liczba jednostek arytmetycznych,
   - sposób wykonywania operacji.

Przetwarzanie potokowe I

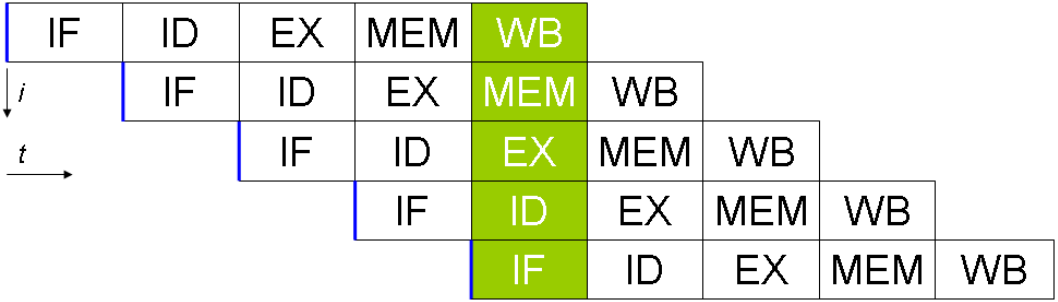
Pipeline

IF — pobieranie instrukcji, **ID** — dekodowanie instrukcji, **EX** — wykonanie, **MEM** — zapis wyników (cache), **WB** — zapis do pamięci

Rysunek: Przetwarzanie danych przez procesor

Przetwarzanie potokowe II

Pipeline

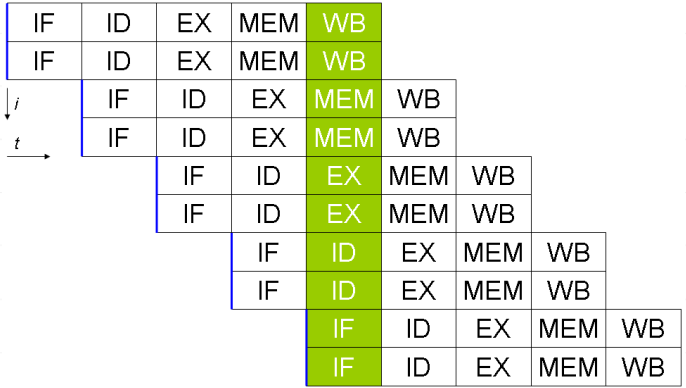


IF — pobieranie instrukcji, **ID** — dekodowanie instrukcji, **EX** — wykonanie, **MEM** — zapis wyników (cache), **WB** — zapis do pamięci

Rysunek: Przetwarzanie danych przez współczesny procesor (pipeline)

Przetwarzanie potokowe

Pipeline + dwa procesory



IF — pobieranie instrukcji, **ID** — dekodowanie instrukcji, **EX** — wykonanie, **MEM** — zapis wyników (cache), **WB** — zapis do pamięci

Rysunek: Przetwarzanie potokowe w przypadku wielu procesorów

Od czego zależy jeszcze szybkość?

Procesory wektorowe

- Procesor wektorowy (tablicowy) to CPU skonstruowane w taki sposób, że zawiera polecenia wykonania operacji matematycznej na wielu elementach danych.
- SIMD — Single Instruction, Multiple Data
- Podstawa „superkomputerów” z lat 80 i 90.
- W roku 2000 IBM, Toshiba i Sony współpracowały nad stworzeniem procesora Cell zawierającego jeden procesor skalarny (odwrotność procesora wektorowego) i osiem procesorów wektorowych, który znalazł zastosowanie (między innymi) w PlayStation 3.

Różne akronimy

1. CISC *Complex Instruction Set Computer*

Różne akronimy

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

Różne akronimy

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

Różne akronimy

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

Różne akronimy

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

Różne akronimy

1. **x86** Najpopularniejsza architektura (CISC) komputerów PC.

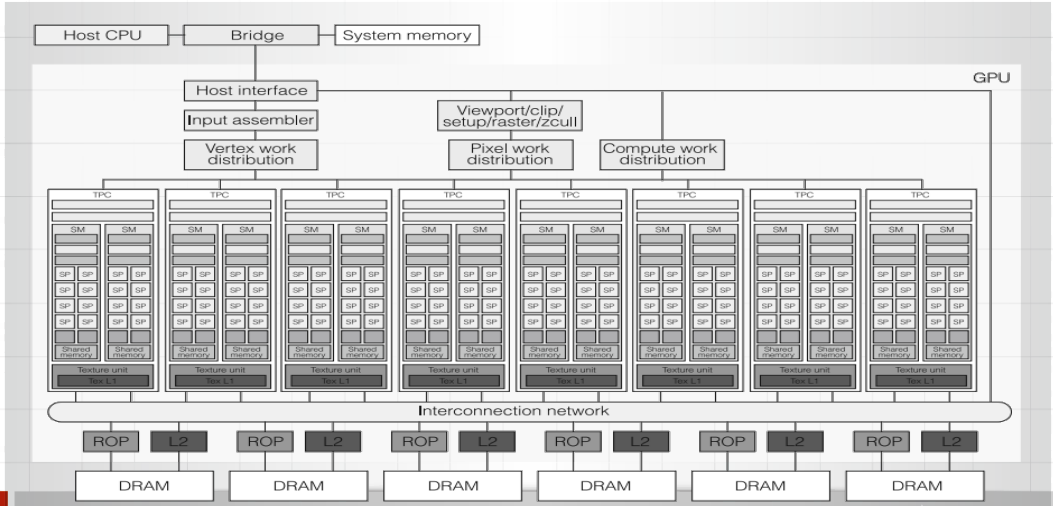
Różne akronimy

1. **x86** Najpopularniejsza architektura (CISC) komputerów PC.
2. **x86-64** Architektura 64 bitowa (będąca rozszerzeniem wprowadzana przez AMD dominująca dziś na rynku.

Różne akronimy

1. **x86** Najpopularniejsza architektura (CISC) komputerów PC.
2. **x86-64** Architektura 64 bitowa (będąca rozszerzeniem wprowadzana przez AMD dominująca dziś na rynku.
3. **ARM** architektura (RISC), która zdominowała rynek współczesnych urządzeń mobilnych, ale nie tylko.
   Najnowsze urządzenia Apple (z procesorami M1 i'lub M2) z niej korzystają.
4. **RISC-V** – otwarty model programowy procesora (ISA) oparty o zasady RISC.

NVIDIA CUDA



Rysunek: Wewnętrzna architektura karty graficznej Tesla

Superkomputery

- Pojęcie superkomputer jest nierozłącznie związane z szybkością obliczeń.
- Zainteresowanych odsyłam do studiowania listy Top 500
- Najwięcej superkomputerów mają USA (150); drugie na liście są Chinye (134)
- Wtej chwili (czerwiec 2023) Polsce są trzy superkoputery z listy znajdujące sie w Krakowie (miejsca 123 i 362) i Poznaniu (miejsce 186).

Dane zmieniają się bardzo szybko.

Zadanie domowe

Zapoznać się z wymienionymi skrótowcami.

Co trzeba żeby komputer działał?

1. Co to jest komputer?

Co trzeba żeby komputer działał?

1. Co to jest komputer?
2. Rodzaj kalkulatora (ma arytmometr/procesor).

Co trzeba żeby komputer działał?

1. Co to jest komputer?
2. Rodzaj kalkulatora (ma arytmometr/procesor).
3. Ma pamięć…

Co trzeba żeby komputer działał?

1. Co to jest komputer?
2. Rodzaj kalkulatora (ma arytmometr/procesor).
3. Ma pamięć…
4. …ale co „popycha” go do działania?

Co trzeba żeby komputer działał?

1. Co to jest komputer?
2. Rodzaj kalkulatora (ma arytmometr/procesor).
3. Ma pamięć…
4. …ale co „popycha” go do działania?
5. Program

Co trzeba żeby komputer działał?

1. Co to jest komputer?
2. Rodzaj kalkulatora (ma arytmometr/procesor).
3. Ma pamięć…
4. …ale co „popycha” go do działania?
5. Program

Co trzeba żeby komputer działał?

1. Co to jest komputer?
2. Rodzaj kalkulatora (ma arytmometr/procesor).
3. Ma pamięć…
4. …ale co „popycha” go do działania?
5. Program?

Włączamy komputer…

…i co się dzieje

1. Jak wszystko jest OK procesor automatycznie próbuje wykonać program znajdujący się w **ustalonym** miejscu pamięci.

[1]BIOS to nazwa własna tłumacząca się jako Basic Input Output System, ale też ogólna nazwa każdego oprogramowania uruchamianego zaraz po starcie komputera.

Włączamy komputer...

...i co się dzieje

1. Jak wszystko jest OK procesor automatycznie próbuje wykonać program znajdujący się w **ustalonym** miejscu pamięci.
2. W tym miejscu pamięci **musi** być jakiś program...

[1]BIOS to nazwa własna tłumacząca się jako Basic Input Output System, ale też ogólna nazwa każdego oprogramowania uruchamianego zaraz po starcie komputera.

Włączamy komputer…

…i co się dzieje

1. Jak wszystko jest OK procesor automatycznie próbuje wykonać program znajdujący się w **ustalonym** miejscu pamięci.
2. W tym miejscu pamięci **musi** być jakiś program…
3. Zazwyczaj w tym obszarze pamięci znajduje się „pamięć stała” (*Read Only Memory* — ROM).

[1]BIOS to nazwa własna tłumacząca się jako Basic Input Output System, ale też ogólna nazwa każdego oprogramowania uruchamianego zaraz po starcie komputera.

Włączamy komputer…

…i co się dzieje

1. Jak wszystko jest OK procesor automatycznie próbuje wykonać program znajdujący się w **ustalonym** miejscu pamięci.
2. W tym miejscu pamięci **musi** być jakiś program…
3. Zazwyczaj w tym obszarze pamięci znajduje się „pamięć stała” (*Read Only Memory* — ROM).
4. W tym miejscu znajduje się program zwany BIOS[1] (*Basic Input Output System*).

[1]BIOS to nazwa własna tłumacząca się jako Basic Input Output System, ale też ogólna nazwa każdego oprogramowania uruchamianego zaraz po starcie komputera.

Włączamy komputer…

…i co się dzieje

1. Jak wszystko jest OK procesor automatycznie próbuje wykonać program znajdujący się w **ustalonym** miejscu pamięci.
2. W tym miejscu pamięci **musi** być jakiś program…
3. Zazwyczaj w tym obszarze pamięci znajduje się „pamięć stała” (*Read Only Memory* — ROM).
4. W tym miejscu znajduje się program zwany BIOS[1] (*Basic Input Output System*).
5. BIOS sprawdza wszystkie komponenty komputera.

[1]BIOS to nazwa własna tłumacząca się jako Basic Input Output System, ale też ogólna nazwa każdego oprogramowania uruchamianego zaraz po starcie komputera.

Włączamy komputer…

…i co się dzieje

1. Jak wszystko jest OK procesor automatycznie próbuje wykonać program znajdujący się w **ustalonym** miejscu pamięci.
2. W tym miejscu pamięci **musi** być jakiś program…
3. Zazwyczaj w tym obszarze pamięci znajduje się „pamięć stała” (*Read Only Memory* — ROM).
4. W tym miejscu znajduje się program zwany BIOS[1] (*Basic Input Output System*).
5. BIOS sprawdza wszystkie komponenty komputera.
6. BIOS ładuje z dysku system operacyjny.

[1]BIOS to nazwa własna tłumacząca się jako Basic Input Output System, ale też ogólna nazwa każdego oprogramowania uruchamianego zaraz po starcie komputera.

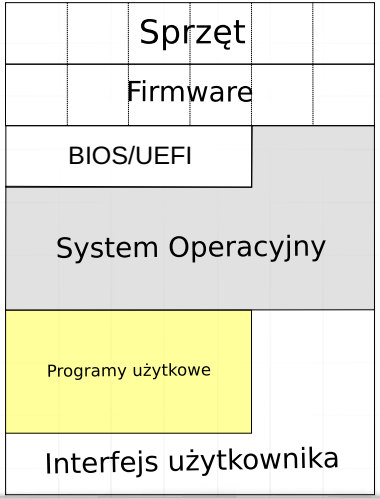
Włączamy komputer…

…i co się dzieje

1. Jak wszystko jest OK procesor automatycznie próbuje wykonać program znajdujący się w **ustalonym** miejscu pamięci.
2. W tym miejscu pamięci **musi** być jakiś program…
3. Zazwyczaj w tym obszarze pamięci znajduje się „pamięć stała” (*Read Only Memory* — ROM).
4. W tym miejscu znajduje się program zwany BIOS[1] (*Basic Input Output System*).
5. BIOS sprawdza wszystkie komponenty komputera.
6. BIOS ładuje z dysku system operacyjny.
7. System operacyjny uruchamia aplikacje użytkowe.

[1]BIOS to nazwa własna tłumacząca się jako Basic Input Output System, ale też ogólna nazwa każdego oprogramowania uruchamianego zaraz po starcie komputera.

BIOS


Sprzęt
Firmware
BIOS/UEFI
System Operacyjny
Programy użytkowe
Interfejs użytkownika

Programy

- Oprogramowanie (i jego jakość) wpływa bardzo mocno na efektywną szybkość komputerów.
- Na czym polega programowanie komputera?

Przykładowy program

c

```c
int main()
{
        int a=1;
        int b=2;
        int c;
        c=a+b;
        return 0;
}
```

Przykładowy program I

Assembler

```
        .file   "p.c"
        .text
        .globl  main
        .type   main, @function
main:
.LFB0:
        .cfi_startproc
        pushq   %rbp    #
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp      #,
        .cfi_def_cfa_register 6
# p.c:3:        int a=1;
        movl    $1, -12(%rbp)   #, a
# p.c:4:        int b=2;
        movl    $2, -8(%rbp)    #, b
# p.c:6:        c=a+b;
        movl    -12(%rbp), %edx # a, tmp93
        movl    -8(%rbp), %eax  # b, tmp94
        addl    %edx, %eax      # tmp93, tmp92
        movl    %eax, -4(%rbp)  # tmp92, c
# p.c:7:        return 0;
        movl    $0, %eax        #, _4
# p.c:8: }
        popq    %rbp    #
        .cfi_def_cfa 7, 8
        ret
```

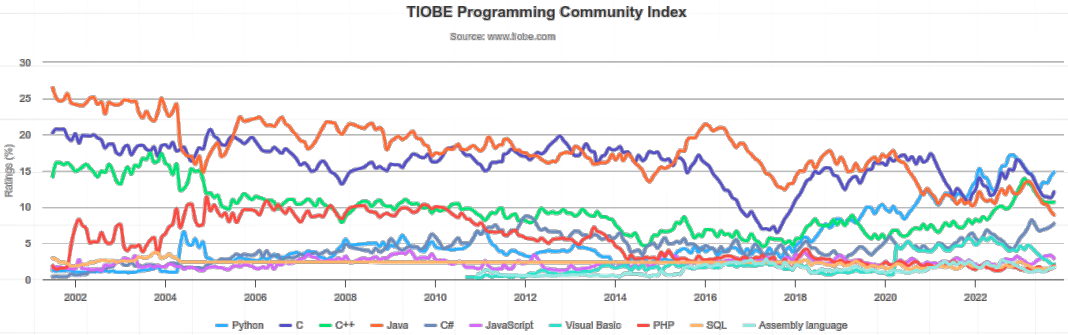
Przykładowy program II

Assembler

```
        .cfi_endproc
.LFE0:
        .size   main, .-main
        .ident  "GCC: (Ubuntu 7.3.0-27ubuntu1~18.04) 7.3.0"
        .section        .note.GNU-stack,"",@progbits
```

Czy umiejętność programowania jest ważna?

Języki programowania



Rysunek: TIOBE programming community index, źródło:
https://www.tiobe.com/tiobe-index/

Proste zadania

Suma liczb

Zadanie polega na tym, że mamy dodać, powiedzmy, 1000 liczb (dostarczonych na papierku). Jak to robić:

Proste zadania

Suma liczb

Zadanie polega na tym, że mamy dodać, powiedzmy, 1000 liczb (dostarczonych na papierku). Jak to robić:

- ręcznie,

Proste zadania

Suma liczb

Zadanie polega na tym, że mamy dodać, powiedzmy, 1000 liczb (dostarczonych na papierku). Jak to robić:

- ręcznie,
- ręcznie z użyciem kalkulatora,

Proste zadania

Suma liczb

Zadanie polega na tym, że mamy dodać, powiedzmy, 1000 liczb (dostarczonych na papierku). Jak to robić:

- ręcznie,
- ręcznie z użyciem kalkulatora,
- za pomocą gotowego programu,

Proste zadania

Suma liczb

Zadanie polega na tym, że mamy dodać, powiedzmy, 1000 liczb (dostarczonych na papierku). Jak to robić:

- ręcznie,
- ręcznie z użyciem kalkulatora,
- za pomocą gotowego programu,
- za pomocą programu napisanego przez siebie?

Bardziej zaawansowany program techniczny

Okres drgań wahadła matematycznego

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

Bardziej zaawansowany program techniczny

Okres drgań wahadła matematycznego

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

- Mamy, powiedzmy, 100 wartości l, mamy policzyć okresy drgań wahadła; jak to zrobić”

Bardziej zaawansowany program techniczny

Okres drgań wahadła matematycznego

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

- Mamy, powiedzmy, 100 wartości l, mamy policzyć okresy drgań wahadła; jak to zrobić”
  - ręcznie?? (bez kalkulatora będzie trudno)

Bardziej zaawansowany program techniczny

Okres drgań wahadła matematycznego

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

- Mamy, powiedzmy, 100 wartości l, mamy policzyć okresy drgań wahadła; jak to zrobić”
  - ręcznie?? (bez kalkulatora będzie trudno)
  - pisać program?

Bardziej zaawansowany program techniczny

Okres drgań wahadła matematycznego

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

- Mamy, powiedzmy, 100 wartości l, mamy policzyć okresy drgań wahadła; jak to zrobić”
  - ręcznie?? (bez kalkulatora będzie trudno)
  - pisać program?
  - skorzystać z „gotowca” (arkusz kalkulacyjny)

Bardziej zaawansowany program techniczny

Okres drgań wahadła matematycznego

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

- Mamy, powiedzmy, 100 wartości l, mamy policzyć okresy drgań wahadła; jak to zrobić”
  - ręcznie?? (bez kalkulatora będzie trudno)
  - pisać program?
  - skorzystać z „gotowca” (arkusz kalkulacyjny)
  - narysować wykres funkcji?

Wykres

Rysunek: Wykres funkcji $T = 2\pi\sqrt{\frac{l}{g}}$

Labirynt

Postawienie problemu

- mamy labirynt (najprostszy),

Labirynt

Postawienie problemu

- mamy labirynt (najprostszy),
- mamy wejście (i jesteśmy przy wejściu),

Labirynt

Postawienie problemu

- mamy labirynt (najprostszy),
- mamy wejście (i jesteśmy przy wejściu),
- jest jedno wyjście,

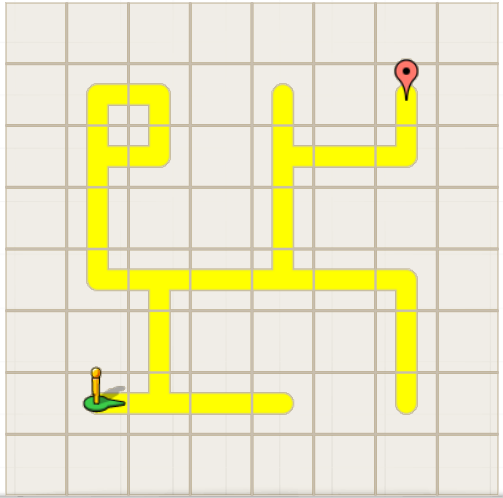
Labirynt

Postawienie problemu

- mamy labirynt (najprostszy),
- mamy wejście (i jesteśmy przy wejściu),
- jest jedno wyjście,
- trzeba znaleźć drogę prowadzącą do wyjścia.

Bardziej złożony problem

labirynt



Rysunek: Labirynt z Google games

Język programowania Google Blockly

Dalsze przykłady będę programował w języku Google 

Jest to *visual programming editor*…

Język programowania Google Blockly

Dalsze przykłady będę programował w języku Google 

Jest to *visual programming editor*...

- Można korzystać on-line:
  `https://blockly-demo.appspot.com/static/demos/code/index.html`
- Można ściągnąć na komputer
  `https://github.com/google/blockly/zipball/master`
  - rozpakować w jakiejś kartotece
  - znaleźć w tej kartotece podkartotekę `google-blockly-xxxxxx` (gdzie xxxxxx to będzie wyglądający na losowy ciąg znaków; przemianować tę kartotekę na `google-blockly` wejść do podkartoteki `demos/code`, znaleźć plik `index.html` i otworzć go w ulubionej przeglądarce.

- Prosty program

- Prosty program
- Losowe ruchy: idź do skrzyżowania i podejmij losową decyzję w którą stronę skręcić.

- Prosty program
- Losowe ruchy: idź do skrzyżowania i podejmij losową decyzję w którą stronę skręcić.
- Zadanie domowe: jak tę ideę zaprogramować w Blockly? I czy się da?

- Prosty program
- Losowe ruchy: idź do skrzyżowania i podejmij losową decyzję w którą stronę skręcić.
- Zadanie domowe: jak tę ideę zaprogramować w Blockly? I czy się da?
- Reguła lewej/prawej ręki: posuwaj się, żeby mieć ścianę zawsze po lewej/prawej stronie.

Największy Wspólny Dzielnik

- Mamy dwie liczby całkowite, dodatnie i różne od zera m i n.

Największy Wspólny Dzielnik

- Mamy dwie liczby całkowite, dodatnie i różne od zera m i n.
- Szukamy takiej liczby x która jest podzielnikiem i m i n i jest to największa liczba wśród wszystkich takich podzielników.

Największy Wspólny Dzielnik

Prosty algorytm z definicji

- znajdź wszystkie podzielniki pierwszej liczby,

Największy Wspólny Dzielnik

Prosty algorytm z definicji

- znajdź wszystkie podzielniki pierwszej liczby,
- znajdź wszystkie podzielniki drugiej liczby,

Największy Wspólny Dzielnik

Prosty algorytm z definicji

- znajdź wszystkie podzielniki pierwszej liczby,
- znajdź wszystkie podzielniki drugiej liczby,
- znajdź wszystkie wspólne podzielniki,

Największy Wspólny Dzielnik

Prosty algorytm z definicji

- znajdź wszystkie podzielniki pierwszej liczby,
- znajdź wszystkie podzielniki drugiej liczby,
- znajdź wszystkie wspólne podzielniki,
- znajdź największy wśród nich.

Znajdowanie wszystkich podzielników liczby

- czy liczba n dzieli się przez 1
- czy liczba n dzieli się przez 2
- $\ldots$
- czy liczba n dzieli się przez $n - 1$

Znajdowanie wszystkich podzielników

czy można to uprościć?

- Wystarczy startować od dwójki (wszystkie liczby dzielą się przez 1)

Znajdowanie wszystkich podzielników

czy można to uprościć?

- Wystarczy startować od dwójki (wszystkie liczby dzielą się przez 1)
- Kiedy skończyć?

Znajdowanie wszystkich podzielników

czy można to uprościć?

- Wystarczy startować od dwójki (wszystkie liczby dzielą się przez 1)
- Kiedy skończyć?
- Wystarczy kontynuować do $\sqrt{n}$

Wspólna część dwu zbiorów *N* i *M*

1. Weź pierwszy element ze zbioru *N*

Wspólna część dwu zbiorów *N* i *M*

1. Weź pierwszy element ze zbioru *N*
2. Sprawdź czy znajduje się w zbiorze *M*?

Wspólna część dwu zbiorów *N* i *M*

1. Weź pierwszy element ze zbioru *N*
2. Sprawdź czy znajduje się w zbiorze *M*?
3. Jeżeli tak — zapisz w zbiorze wynikowym.

Wspólna część dwu zbiorów *N* i *M*

1. Weź pierwszy element ze zbioru *N*
2. Sprawdź czy znajduje się w zbiorze *M*?
3. Jeżeli tak — zapisz w zbiorze wynikowym.
4. Jeżeli nie przejrzałeś wszystkich elementów w zbiorze *N*, weź element następny i przejdź do kroku 2

Szukanie wartości największej w zbiorze

Szukanie wartości największej w zbiorze

1. weź pierwszy element: będzie „wzorem”

Szukanie wartości największej w zbiorze

1. weź pierwszy element: będzie „wzorem”
2. czy został jakiś element w zbiorze? jeżeli nie — KONIEC

Szukanie wartości największej w zbiorze

1. weź pierwszy element: będzie „wzorem”
2. czy został jakiś element w zbiorze? jeżeli nie — KONIEC
3. weź następny element ze zbioru

Szukanie wartości największej w zbiorze

1. weź pierwszy element: będzie „wzorem”
2. czy został jakiś element w zbiorze? jeżeli nie — KONIEC
3. weź następny element ze zbioru
4. czy większy od „wzoru?”

Szukanie wartości największej w zbiorze

1. weź pierwszy element: będzie „wzorem”
2. czy został jakiś element w zbiorze? jeżeli nie — KONIEC
3. weź następny element ze zbioru
4. czy większy od „wzoru?”
5. jeżeli nie — przejdź do punktu 2

Szukanie wartości największej w zbiorze

1. weź pierwszy element: będzie „wzorem”
2. czy został jakiś element w zbiorze? jeżeli nie — KONIEC
3. weź następny element ze zbioru
4. czy większy od „wzoru?”
5. jeżeli nie — przejdź do punktu 2
6. jeżeli tak — wstaw w miejsce „wzoru”

Algorytm Euklidesa

E1. Niech *r* będzie resztą z dzielenia *m* przez *n*

Algorytm Euklidesa

E1. Niech r będzie resztą z dzielenia m przez n

E2. Jeżeli $r = 0$ koniec

Algorytm Euklidesa

E1. Niech r będzie resztą z dzielenia m przez n

E2. Jeżeli $r = 0$ koniec

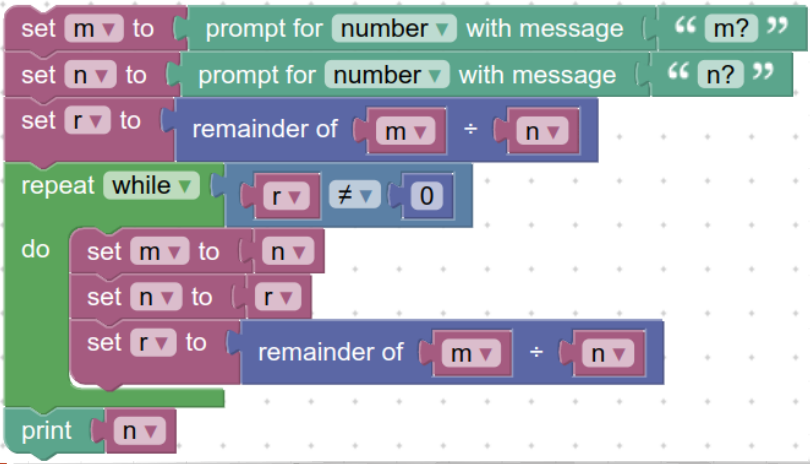
E3. W przeciwnym razie
$m = n$
$n = r$
przejdź do E1

NWD

Program w Blockly

NWD

Program w Blockly



Zadanie domowe

- Znaleźć inne warianty algorytmu Euklidesa
- Zaprogramować w Blockly?

Algorytm B

1. Przyjmij $k \leftarrow 0$, a następnie powtarzaj operacje: $k \leftarrow k+1$, $u \leftarrow u/2$, $v \leftarrow v/2$ zero lub więcej razy do chwili gdy przynajmniej jedna z liczb u i v przestanie być parzysta.
2. Jeśli u jest nieparzyste to przyjmij $t \leftarrow -v$ i przejdź do kroku 4. W przeciwnym razie przyjmij $t \leftarrow u$.
3. (W tym miejscu t jest parzyste i różne od zera). Przyjmij $t \leftarrow t/2$.
4. Jeśli t jest parzyste to przejdź do 3.
5. Jeśli $t > 0$, to przyjmij $u \leftarrow t$, w przeciwnym razie przyjmij $v \leftarrow -t$.
6. Przyjmij $t \leftarrow u - v$. Jeśli $t \neq 0$ to wróć do kroku 3. W przeciwnym razie algorytm zatrzymuje się z wynikiem $u \cdot 2^k$.

Zadanie domowe?

Algorytm B w Blockly?

Zadanie domowe?

Algorytm B w Blockly?

Yyyyy… za trudne

Zadanie domowe?

Algorytm B w Blockly?

Yyyyy… za trudne

Rozwiązać ręcznie dla wybranych *u* i *v* (mniejszych niż 1000).

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.