

Podstawy Informatyki

Katedra Telekomunikacji, EiT

dr inż. Jarosław Bułat

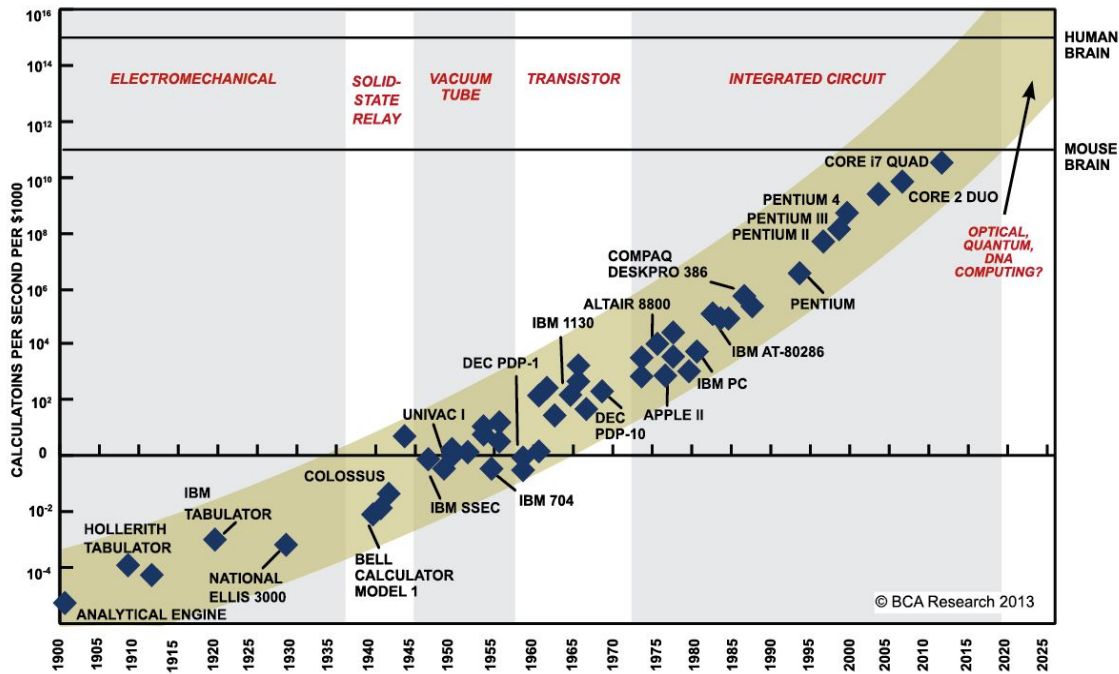
kwant@agh.edu.pl

High level

- » Współcześni programiści zbyt często piszą w “ultrawygodnych językach”, zbyt wysoko (w sensie abstrakcji) od sprzętu. Przez to nie czują jakie ograniczenia ma sprzęt.
- » Prawo Moor’a jest martwe (2020), w procesach technologicznych ≤ 7 nm nie osiąga się taktowania 5+ GHz ze względu na ograniczenia fizyki
- » Pojedynczy rdzeń/wątek nie będzie szybszy niż jest
- » Wniosek:

w przyszłości będzie nacisk na optymalizację kodu

Prawo Moora



SOURCE: RAY KURZWEIL, "THE SINGULARITY IS NEAR: WHEN HUMANS TRANSCEND BIOLOGY", P.67, THE VIKING PRESS, 2006. DATAPPOINTS BETWEEN 2000 AND 2012 REPRESENT BCA ESTIMATES.

Moore's Law: The number of transistors on microchips doubles every two years

Our World
in Data

Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years. This advancement is important for other aspects of technological progress in computing – such as processing speed or the price of computers.

Transistor count

50,000,000,000

10,000,000,000

5,000,000,000

1,000,000,000

500,000,000

100,000,000

50,000,000

10,000,000

5,000,000

1,000,000

500,000

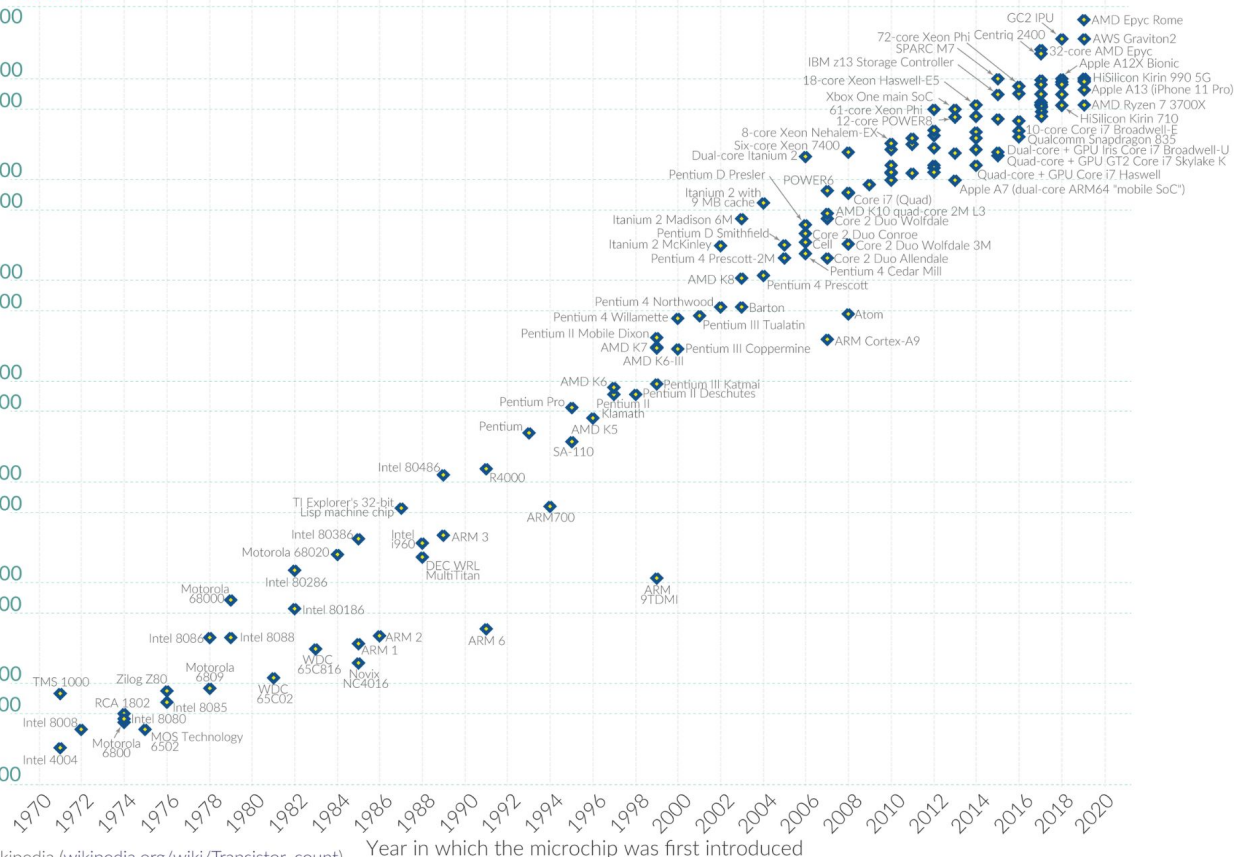
100,000

50,000

10,000

5,000

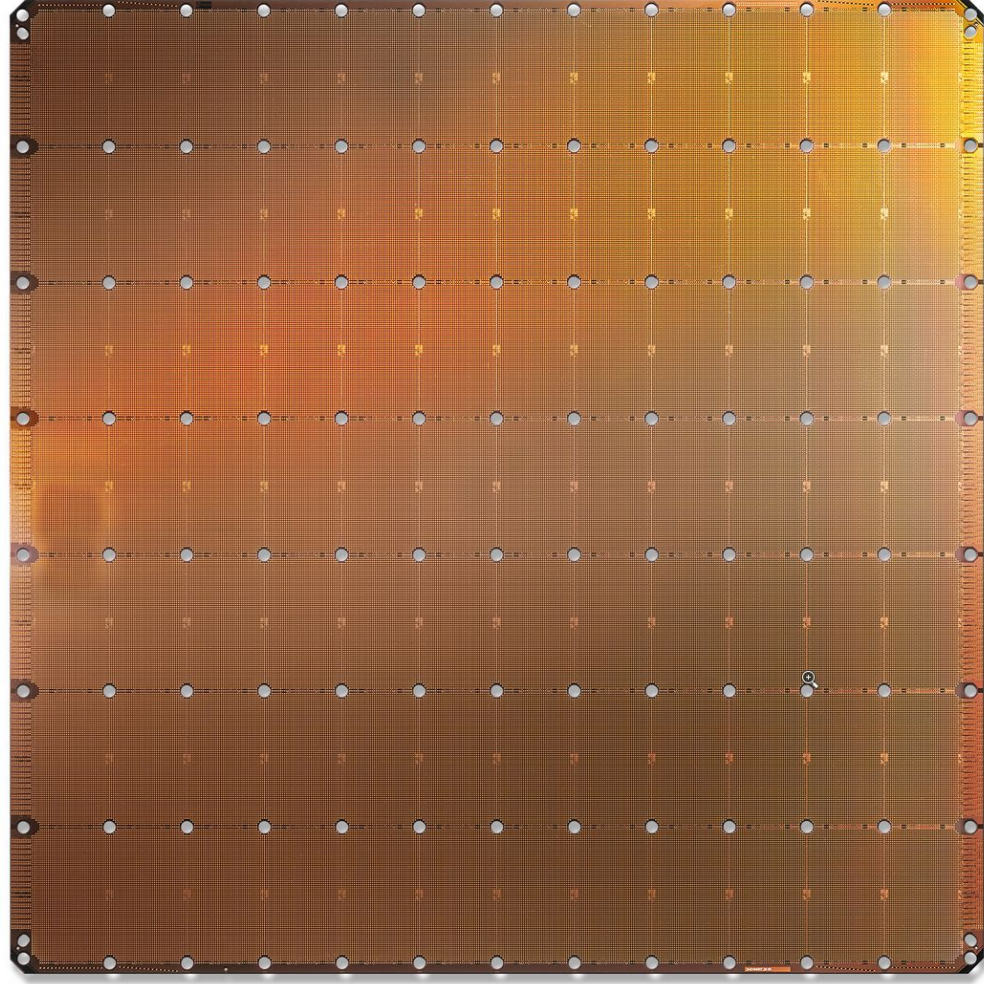
1,000

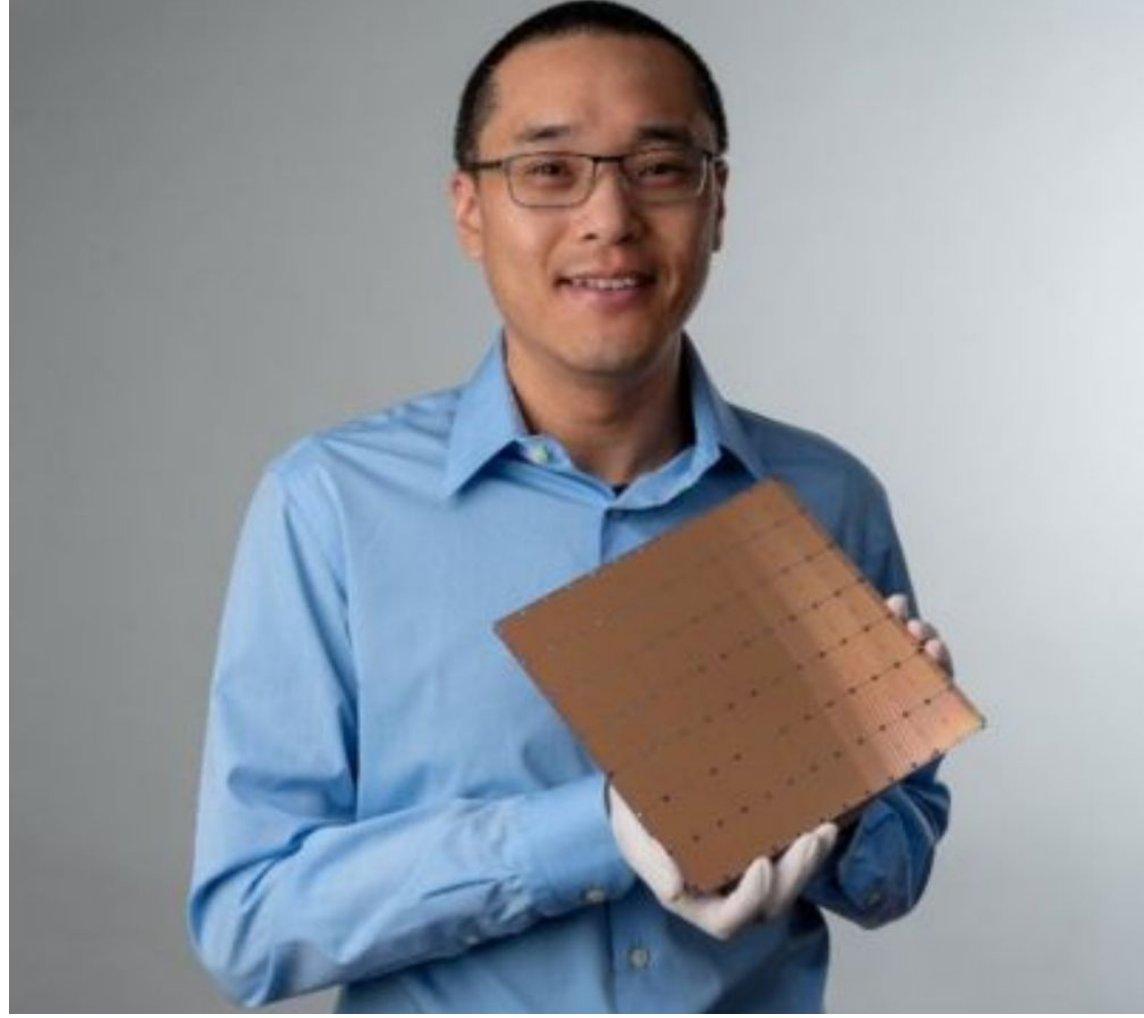


Data source: Wikipedia (wikipedia.org/wiki/Transistor_count)

OurWorldinData.org – Research and data to make progress against the world's largest problems.

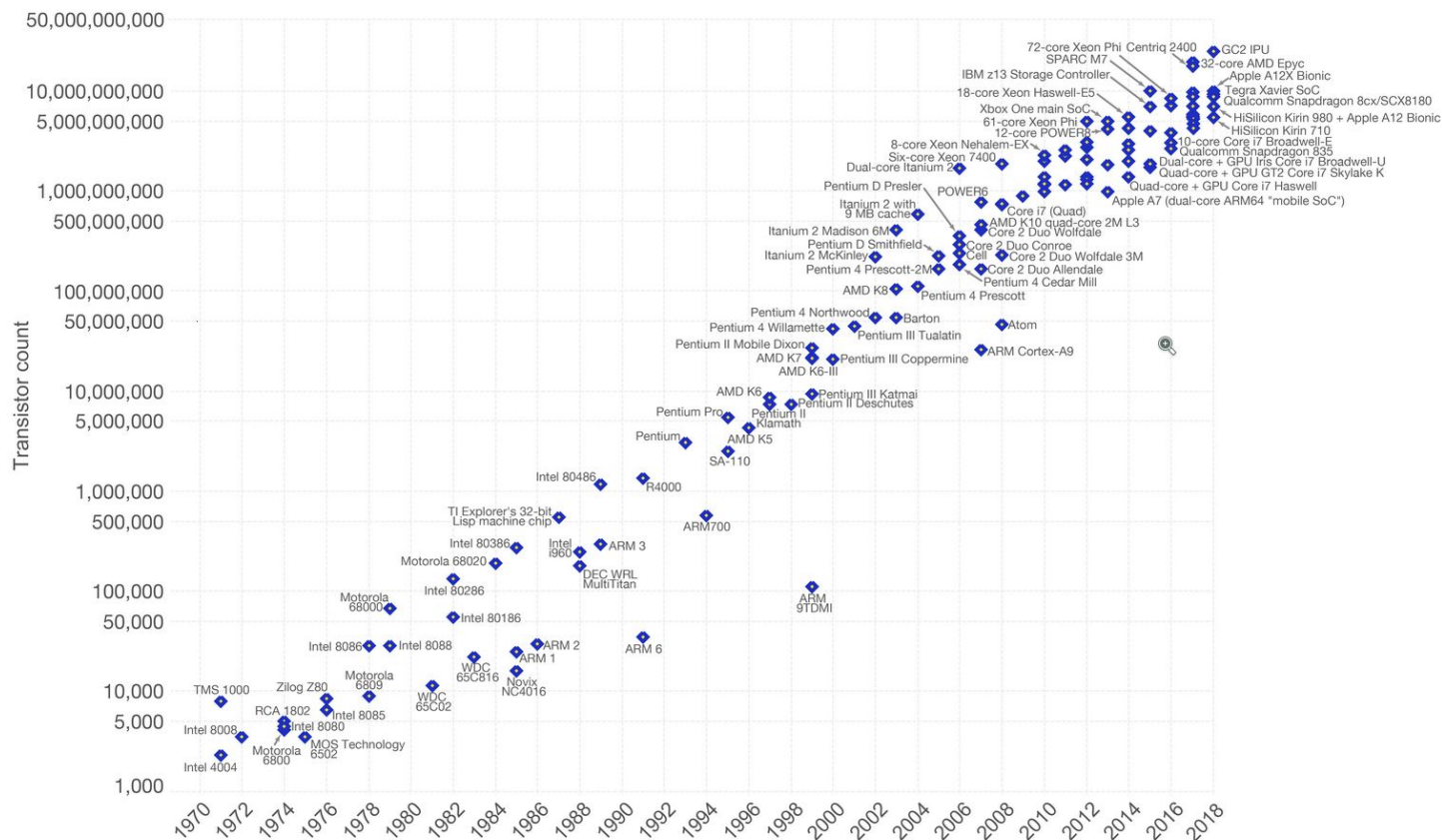
Licensed under CC-BY by the authors Hannah Ritchie and Max Roser.





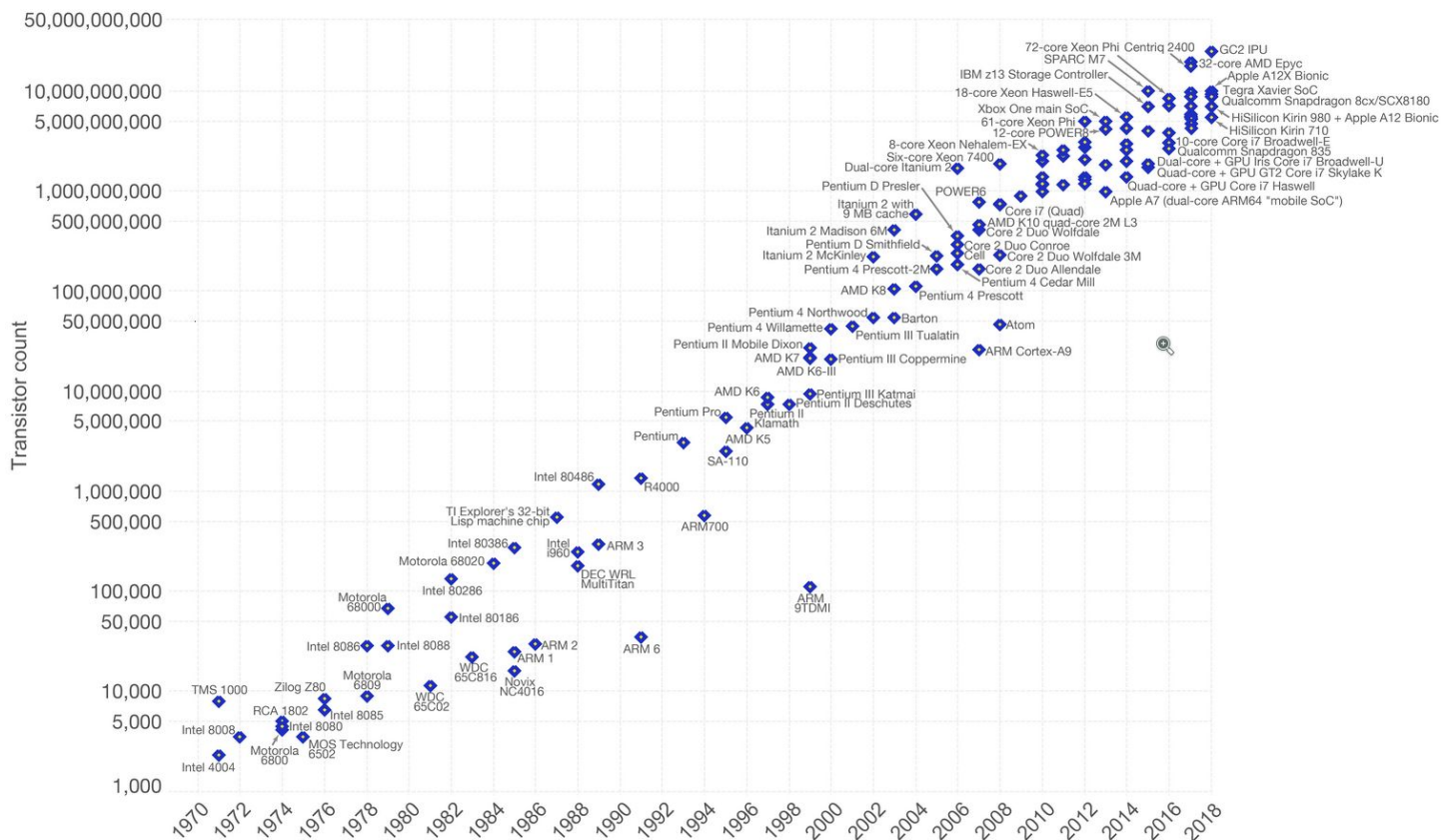
Cerebras Systems

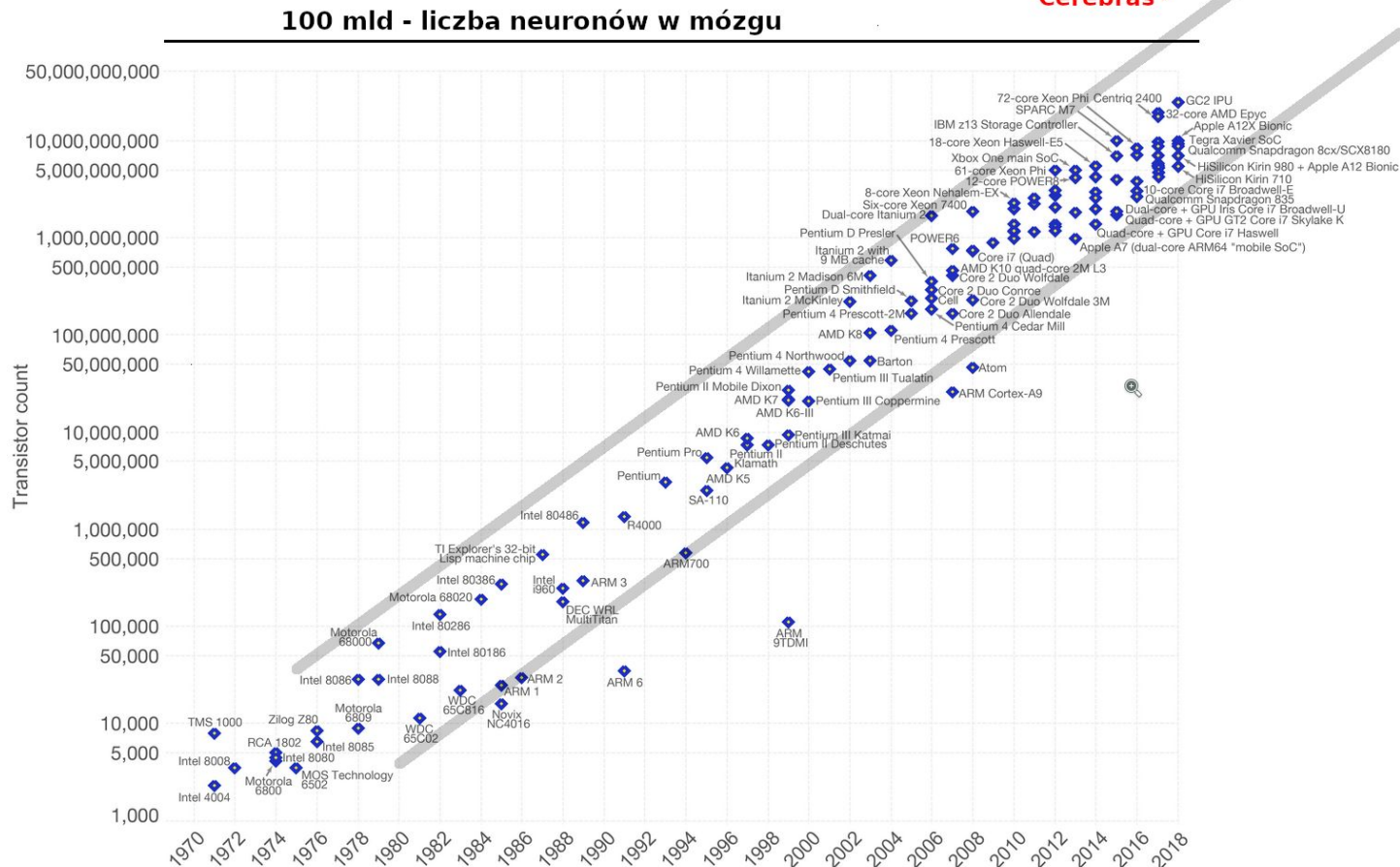
- » 46,225 mm²
- » 400,000 cores (~80x Nvidia)
- » 18 GB on-chip SRAM
- » 100 Pb/s bandwidth
- » 1.2 biliony tranzystorów (1.2 US-tryliony)
1.2 E+12





Cerebras





Wzrost wykładniczy

- » **AMD EPYC 2:** 64 cores (128 threads)
 - 40 mld tranzystorów == $4 \text{ E}+10$

Wzrost wykładniczy

- » **AMD EPYC 2:** 64 cores (128 threads)
 - 40 mld tranzystorów == $4 \text{ E}+10$
 - podwajamy liczbę tranzystorów co ~~18 miesięcy~~
24 miesiące

Wzrost wykładniczy

- » **AMD EPYC 2:** 64 cores (128 threads)
 - 40 mld tranzystorów == $4 \text{ E}+10$
 - podwajamy liczbę tranzystorów co ~~18 miesięcy~~
24 miesiące
 - rok 1970, liczba tranzystorów: 1383
 - rok 2019, liczba tranzystorów: $4 \text{ E}+10$

Wzrost wykładniczy

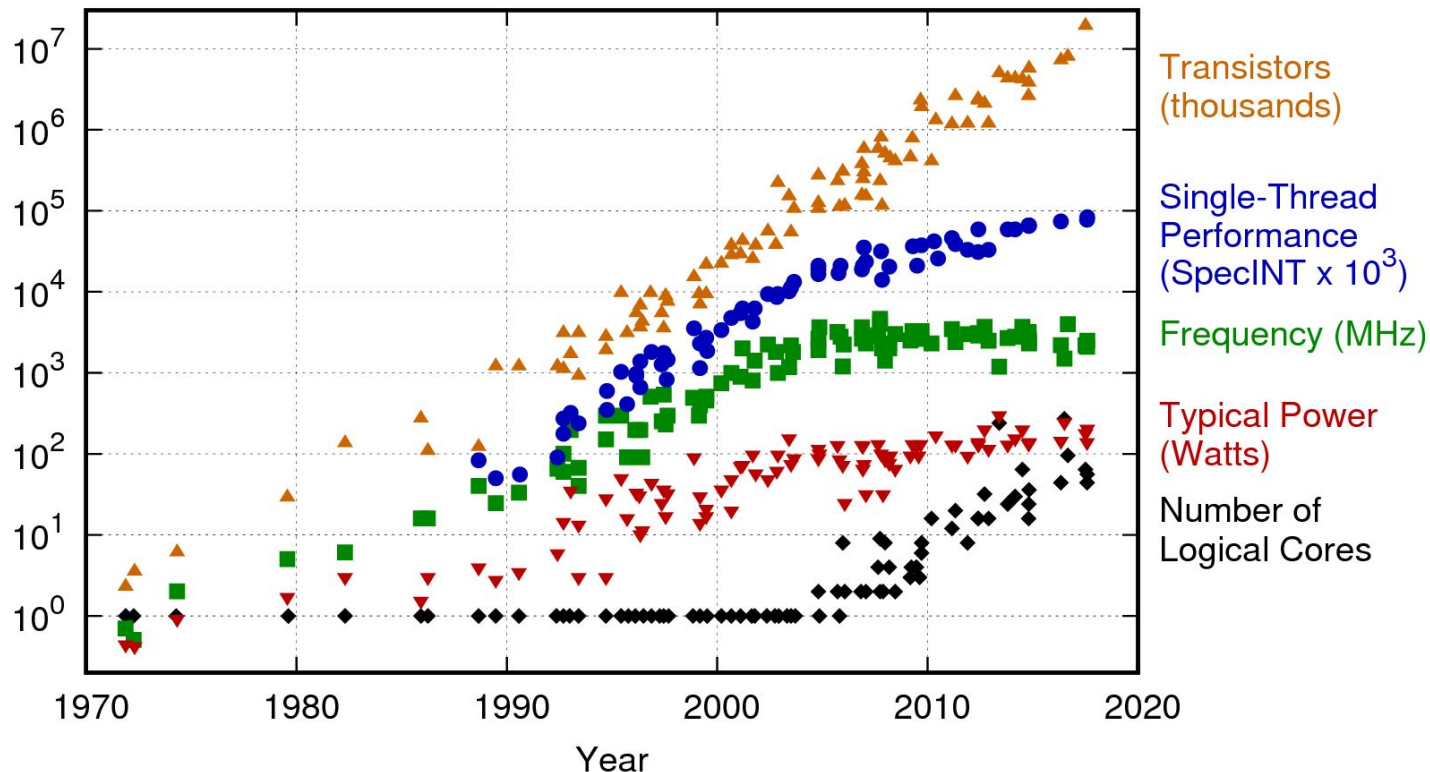
- » **AMD EPYC 2:** 64 cores (128 threads)
 - 40 mld tranzystorów == $4 \text{ E}+10$
 - podwajamy liczbę tranzystorów co ~~18 miesięcy~~
24 miesiące
 - rok 1970, liczba tranzystorów: 1383
 - rok 2019, liczba tranzystorów: $4 \text{ E}+10$
 - rok 2282, liczba tranzystorów: $1.2 \text{ E}+50$
 - rok 2468, liczba tranzystorów: $1.2 \text{ E}+78$

Wzrost wykładniczy

- » **AMD EPYC 2:** 64 cores (128 threads)
 - 40 mld tranzystorów == $4 \text{ E}+10$
 - podwajamy liczbę tranzystorów co ~~18 miesięcy~~
24 miesiące
 - rok 1970, liczba tranzystorów: 1383
 - rok 2019, liczba tranzystorów: $4 \text{ E}+10$
 - rok 2282, liczba tranzystorów: **$1.2 \text{ E}+50$**
 - rok 2468, liczba tranzystorów: **$1.2 \text{ E}+78$**
- » Co oznaczają te liczby?

Wzrost wykładniczy

42 Years of Microprocessor Trend Data



Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2017 by K. Rupp

Plan prezentacji

- » Ankieta (co stwarza problem na zajęciach)
- » **Pętle** - przykłady
 - dużo przykładów
 - instrukcja **break**;
 - instrukcja **continue**;
- » **Tablice** (ang. arrays)
- » Git - rejected push

git / konsola

- » shell: **strzałka w górę**
- » skrypty: **bash/tcsh**
- » praca pod konsolą: **fish**
- » autouzupełnianie: **tab** (fish: **ctrl -f**)
- » schowek:
 - Ctr-c + Ctrl-v
 - 3 przycisk myszy
 - mouse3+ Ctrl-Shift-v

Odliczanie wstecz

```
#include <iostream>
```

```
int main(){
```

```
    size_t size = 10;
```

```
    for (int x = size; x >= 0; --x ) {  
        cout << x << endl;
```

```
    }
```

```
}
```

» Iterator nie musi być zmieniany
o +=1, może być
dekrementowany

» Rezultat:

10

9

...

1

0

Iterator zmieniany co 2

```
#include <iostream>
```

```
int main(){
```

```
    size_t size = 10;
```

```
    for (int x = 0; x < size; x+=2 ) {  
        cout << x << endl;
```

```
    }
```

```
}
```

» Iterator nie musi być zmieniany

o +=1

» Rezultat:

0

2

4

6

8

Zmiana iteratora w pętli

```
#include <iostream>
```

```
int main(){  
    size_t size = 10;  
  
    for (int x = 0; x < size; ++x ) {  
        cout << ++x << endl;  
    }  
}
```

» Zmiana iteratora

» Rezultat:

1

3

5

7

9

» Never Ever !!!

dwa iteratory pętli

```
for(int i=0, j=10; i<5 && j>5; ++i, --j ){  
    cout << i << " ";  
    cout << j << endl;  
}
```

wyrażenie 1: `int i=0, j=10`

wyrażenie 2: `i<5 && j>5`

wyrażenie 3: `++i, --j`

- » Dwa iteratory “i” oraz “j”
- » Warunek jeden, może być złożony
- » Wyrażenie 3, może zmieniać oba iteratory
- » Rezultat:
0 10
1 9
2 8
3 7
4 6
- » Zamiana kolejności elementów w tablicy*

modyfikacja działania pętli

```
#include <iostream>
```

```
int main() {  
    size_t width = 1920;  
  
    for (int x = 0; x < width; ++x ) {  
        if (x == 2) {  
            continue;  
        } else if (x == 5) {  
            break;  
        }  
        cout << x << endl;  
    }  
}
```

» `continue`; rozpoczyna iteracje od początku

» `break`; kończy pętlę

» Rezultat:

0

1

3

4

» Modyfikacja wykonania pętli

pętla w pętli - zagnieżdżenie

```
#include <iostream>
```

```
int main() {  
    size_t width = 1920;  
    size_t height = 1080;  
  
    for (int x = 0; x < width; ++x) {  
        for (int y = 0; y < height; ++y) {  
            // test each pixel of image  
        }  
    }  
}
```

- » Zagnieżdżona pętla
- » Iteracja po wszystkich pikselach obrazu FullHD
- » Pętla **zewnętrzna** (iterator x)
- » Pętla **wewnętrzna** (iterator y)
- » Dla jednego x, wykonają się wszystkie iteracje y
- » Wszystkie iteracje y wykonają się x razy (dla każdego x)
- » Dowolna liczba zagnieżdżeń, **sugerowanie nie więcej niż 3**

pętla w pętli - zagnieżdżenie

```
#include <iostream>
```

```
int main() {  
    size_t width = 1920;  
    size_t height = 1080;  
  
    for (int x = 0; x < width; ++x ) {  
        int z = 9;  
        for (int y = 0; y < height; ++y ) {  
            // test each pixel of image  
        }  
    }  
}
```

- » Każda iteracja pętli to wykonanie nowego bloku instrukcji
- » Zmienna “z” tworzona i inicjalizowana podczas każdej iteracji !!!

pętla w pętli - zagnieżdżenie

```
size_t width = 10;
```

```
for (int x = 0; x < width; ++x) {  
    for (int y = 0; y <= x; ++y) {  
        cout << "(" << x;  
        cout << "," << y << ") ";  
        // upper right triangle  
    }  
    cout << endl;  
}
```

- » Iteracja y kończy się na x
- » rezultat:

```
(0,0)  
(1,0) (1,1)  
(2,0) (2,1) (2,2)  
(3,0) (3,1) (3,2) (3,3)  
(4,0) (4,1) (4,2) (4,3) (4,4)  
(5,0) (5,1) (5,2) (5,3) (5,4) (5,5)  
(6,0) (6,1) (6,2) (6,3) (6,4) (6,5) (6,6)  
(7,0) (7,1) (7,2) (7,3) (7,4) (7,5) (7,6) (7,7)  
(8,0) (8,1) (8,2) (8,3) (8,4) (8,5) (8,6) (8,7) (8,8)  
(9,0) (9,1) (9,2) (9,3) (9,4) (9,5) (9,6) (9,7) (9,8) (9,9)
```



quiz

PI05_for1

socrative.com

- login
- student login

Room name:

KWANTAGH

Pętla nieskończona

```
for (;;) {  
    char c;  
    cin >> c;  
    if (c == 'x') {  
        break;  
    }  
}
```

```
char c;  
cin >> c;  
while (c != 'x') {  
    cin >> c;  
}
```

```
char c;  
do {  
    cin >> c;  
} while (c != 'x');
```

```
while (true) {  
    char c;  
    cin >> c;  
    if (c == 'x') {  
        break;  
    }  
}
```

- » Pętla nieskończona wtedy gdy nie znamy liczby iteracji
- » Koniec pętli po wprowadzeniu znaku 'x'
- » Deklaracja "c" wewnątrz pętli – zasięg!
- » Zawiązać zasięg zmiennych

Wyjście z wewnętrznej pętli

```
#include <iostream>
```

```
int main(){
```

```
    for (size_t x = 0; x < 10; ++x) {  
        for (size_t y = 0; y < 10; ++y) {  
            if (x > 4 && y > 5) {
```

```
                break;
```

```
                // does not work!!!
```

```
            }
```

```
        }
```

```
    }
```

```
    cout << x+10*y << endl;
```

```
}
```

- » Chcę opuścić **obie** pętlę jeżeli $x > 4$ AND $y > 5$ i wyświetlić $x+10*y$
- » Instrukcja **break**; opuści wyłącznie wewnętrzną pętlę

Wyjście z wewnętrznej pętli

```
#include <iostream>
```

```
int main(){
```

```
    for (size_t x = 0; x < 10; ++x) {  
        for (size_t y = 0; y < 10; ++y) {  
            if (x > 4 && y > 5) {
```

```
                break;
```

```
                // does not work!!!
```

```
            }
```

```
        }
```

```
    }
```

```
    cout << x+10*y << endl;
```

```
}
```

- » Chcę opuścić **obie** pętlę jeżeli $x > 4$ AND $y > 5$ i wyświetlić $x+10*y$
- » Instrukcja `break`; opuści wyłącznie wewnętrzną pętlę
- » **Jaki błąd zrobiłem w cout ???**

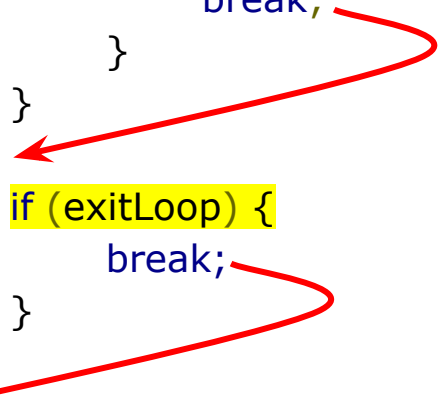
Wyjście z wewnętrznej pętli

```
for (size_t x = 0; x < 10; ++x) {  
    bool exitLoop = false;  
  
    for (size_t y = 0; y < 10; ++y) {  
        if (x > 4 && y > 5) {  
            exitLoop = true;  
            break;  
        }  
    }  
  
    if (exitLoop) {  
        break;  
    }  
}
```

- » Chcę opuścić **obie** pętlę jeżeli **x > 4** AND **y > 5**
- » Wyjście z wewnętrznej pętli + ustawienie znacznika exitLoop
- » Sprawdzanie znacznika na końcu zewnętrznej pętli

Wyjście z wewnętrznej pętli

```
for (size_t x = 0; x < 10; ++x) {  
    bool exitLoop = false;  
  
    for (size_t y = 0; y < 10; ++y) {  
        if (x > 4 && y > 5) {  
            exitLoop = true;  
            break;  
        }  
    }  
    if (exitLoop) {  
        break;  
    }  
}
```



- » Chcę opuścić **obie** pętlę jeżeli **x > 4** AND **y > 5**
- » Wyjście z wewnętrznej pętli + ustawienie znacznika exitLoop
- » Sprawdzanie znacznika na końcu zewnętrznej pętli

Wyjście z wewnętrznej pętli

```
size_t x = 0;
bool looping = true;

do {
    // bool looping = false; // not in scope !?!

    for (size_t y = 0; y < 10; ++y) {
        if (x > 4 && y > 5) {
            looping = false;
            break;
        }
    }
} while (looping && ++x < 10);
```

- » Chcę opuścić **obie** pętlę jeżeli **x > 4** AND **y > 5**
- » Wyjście z wewnętrznej pętli + ustawienie znacznika exitLoop
- » Pętla do-while sprawdza warunek na końcu
- » Deklaracja looping musi być na zewnątrz pętli

Wyjście z wewnętrznej pętli

```
#include <iostream>
```

```
int main(){
```

```
    for (size_t x = 0; x < 10; ++x) {  
        for (size_t y = 0; y < 10; ++y) {  
            if (x > 4 && y > 5) {  
                goto exitLoop;
```

```
            }
```

```
        }
```

```
    }
```

```
    exitLoop:
```

```
    cout << "end" << endl;
```

```
}
```

- » Chcę opuścić **obie** pętlę jeżeli **x > 4** AND **y > 5**
- » Skok bezwarunkowy **goto**
- » Pokusa jest duża ;-), ale **Never Ever !!!**

Wyjście z wewnętrznej pętli

```
#include <iostream>
```

```
int main(){
```

```
    for (size_t x = 0; x < 10; ++x) {  
        for (size_t y = 0; y < 10; ++y) {  
            if (x > 4 && y > 5) {  
                goto exitLoop;
```

```
exitLoop:
```

```
    cout << "end" << endl;  
}
```

- » Chcę opuścić **obie** pętlę jeżeli **x > 4 AND y > 5**
- » Skok bezwarunkowy **goto**
- » Pokusa jest duża ;-), ale **Never Ever !!!**



quiz

PI05_for2

socrative.com

- login
- student login

Room name:

KWANTAGH

Mam wiele takich
samych elementów
jak je przechować?

Tablice

- » Sposób na organizację wielu elementów jednego typu
 - każdy z elementów może być **indywidualnie adresowany**
 - wszystkie elementy w **ciągłej przestrzeni adresowej**
 - **brak** możliwości **zmiany rozmiaru** tablicy po utworzeniu
 - **tablica jest zmienną** (dotyczą wszystkie reguły dla zmiennych):
 - zasięg
 - niemożność zmiany typu
 - konieczność rezerwacji pamięci
 - nazwa

Prosta tablica

```
#include <iostream>
```

```
int main(){
```

```
    int tab[5];
```

```
    tab[0] = 1;
```

```
    tab[1] = 4;
```

```
    tab[2] = tab[0];
```

```
    tab[3] = -10;
```

```
    tab[4] = 4;
```

```
    for (size_t i = 0; i < 5; ++i) {
```

```
        tab[i] = i;
```

```
    }
```

```
}
```

» Deklaracja:

– typ

– rozmiar (liczba elementów)

» Adresowanie w nawiasach kwadratowych

» **tab[2]** jest typu **int** (w tym przypadku)

» **tab[5]** oznacza pięcio-elementową tablicę więc **tab[0]...tab[4]**

» ładnie wygląda razem z pętlą (**patrz warunek!!!**)

Prosta tablica

```
#include <iostream>
int main(){

    int tab[5];

    tab[0] = 1;
    tab[1] = 4;
    tab[2] = tab[0];
    tab[3] = -10;
    tab[4] = 4;
    // tab[5] do not exist !!!
    tab[5] = 1123;
    // will work and create
    // huge problem !

}
```

- » Kompilator/runtime nie sprawdza zakresu !!!
- » tab[5] zostanie wykonane pomimo że nie istnieje !!!
- » Najczęstsze źródło błędów “buffer overflow”
- » Bardzo efektywny sposób dostępu do pamięci ale niebezpieczny!!!
- » x=0;
tab[x-1];
- » valgrind służy do wyszukiwania błędów adresowania

Inicjalizacja tablicy

```
#include <iostream>
```

```
int main(){
```

```
    int tab[5] = {0,1,2,3,4};
```

```
    int tax[] = {0,1,2,3,4};
```

```
    for (int i = 4; i >= 0; --i) {  
        tab[i] = i*10;  
    }
```

```
    int x = tab[0];    // ? value ?
```

```
    tab[++x] = 7;
```

```
    tab[tax[4]] = tab[1];
```

```
}
```

- » Możliwa inicjalizacja podczas deklaracji
- » Nie trzeba podawać rozmiaru jeżeli inicjalizacja podczas deklaracji
- » Indeksowanie tablicy zawsze liczbą naturalną $\langle 0, 1, 2, 3, \dots, \text{size}-1 \rangle$
- » Często zerowanie pętla
- » Jaką wartość będzie miało x?
- » Na którą pozycję wpisane zostanie 7?
- » Indeksowanie pośrednie.

Akumulacja danych z tablicy

```
#include <iostream>
```

```
int tab[] = {0,1,2,3,4};
```

```
int main(){
```

```
    int result = 0;
```

```
    for (size_t i = 0; i < 5; ++i ) {
```

```
        result += tab[i];
```

```
    }
```

```
    cout << result;
```

```
    cout << endl;
```

```
}
```

- » Sumowanie wszystkich elementów z tablicy
- » Ważna inicjalizacja zmiennej result

Obliczanie rozmiaru tablicy

```
#include <iostream>
```

```
int tab[] = {0,1,2,3,4};
```

```
int main(){
```

```
    size_t size;
```

```
    size = sizeof(tab)/sizeof(tab[0]);
```

```
    int result = 0;
```

```
    for (size_t i = 0; i < size; ++i) {  
        result += tab[i];  
    }
```

```
    cout << result << endl;
```

```
}
```

- » To nie jest uniwersalne rozwiązanie
- » Nie będzie działać dla wskaźników (np. rezerwacja tablicy przez new/alloc).
- » `sizeof(tab)` podaje rozmiar w bajtach całej tablicy
- » `sizeof(tab[0])` podaje rozmiar w bajtach pojedynczego elementu tablicy

Deklaracja tablicy - rozmiar

```
#include <iostream>
```

```
int main(){
```

```
    size_t size = 10;
```

```
    int tab[size];           // c++98
```

```
    for (size_t i = 0; i < size; ++i ) {  
        tab[i] = 0;
```

```
    }
```

```
}
```

- » **Do c++98** rozmiar tablicy musiał być stałą (wartością znaną podczas kompilacji)
- » **Od c++98** włącznie, rozmiar może być zmienną (niejawna dynamiczna alokacja pamięci)

Wartość max w tablicy

```
int tab[] = {1,3,6,2,1,6753,2,341,0,1};
```

```
int max = 0;
```

```
for (size_t i = 0; i < 10; ++i) {
```

```
    if (max < tab[i]) {  
        max = tab[i];
```

```
    }
```

```
}
```

```
cout << max << endl;
```

- » Wyszukiwanie wartości maksymalnej w tablicy
- » **Inicjalizacja** zmiennej max
- » **Iteracja** po wszystkich elementach tablicy
- » **Porównanie** do każdego elementu
- » **Przypisanie** tab[i] do max jeżeli tab[i] jest większe
- » **Kiedy algorytm nie zadziała?**

Wartość max w tablicy

```
int tab[] = {1,3,6,2,1,6753,2,341,0,1};
```

```
int max = tab[0];
```

```
for (size_t i = 1; i < 10; ++i) {  
    if (max < tab[i]) {  
        max = tab[i];  
    }  
}
```

```
cout << max << endl;
```

- » Najszybciej
- » Zadziała dla ujemnych, dodatnich
- » W pierwszej iteracji porównywane jest `tab[0]` do `tab[1]`

Tablice wielowymiarowe

```
#include <iostream>
```

```
int main(){
```

```
    int tab2d[5][10];
```

```
    int tensor[2][3][7][5];
```

```
    // 210 cells
```

```
    tab2d[0][0] = 0;
```

```
    tab2d[4][9] = 4*9;
```

```
}
```

- » Dowolna liczba wymiarów
- » Zasady (deklaracja, indeksowanie) takie jak dla tablic jednowymiarowych

Tablica struktur

```
struct Person {  
    int age;  
    float salary;  
};  
  
Person employee[10];  
Person e = employee[0];  
e.age = 30;  
e.salary = 4000;  
  
employee[1] = e;  
e = employee[2];  
employee[3] = employee[2];
```

`e = employee;` // Błąd !!!

- » Tablica może być dowolnego typu więc również “mojego typu”
- » Każdy element tablicy jest pojedynczą strukturą **Person**
- » `employee[x]` jest typu **Person**
- » **employee NIE** jest typu **Person** !!!

Tablica struktur

```
struct Person {  
    int age;  
    float salary;  
};  
Person e = {30, 4000};  
  
Person employee[10];  
  
employee[2].age = 30;  
employee[2].salary = e.salary;  
  
employee.age; // Błąd !!!
```

- » Każdy element tablicy jest pojedynczą strukturą **Person**
- » Elementy struktury w tablicy można adresować bezpośrednio (operator .)
- » Zmienna employee NIE jest typu **Person**, więc nie można bezpośrednio adresować elementów struktury - **nie wiadomo którego elementu dotyczą**

dlaczego push
się nie powiódł?
dlatego bo masz konflikty...

GIT - rejected push

developer 1

```
> git add source1.cc  
> git commit -am "source1"
```

developer 2

GIT - rejected push

developer 1

```
> git add source1.cc  
> git commit -am "source1"
```

developer 2

```
> git add source2.cc  
> git commit -am "source2"  
> git push
```

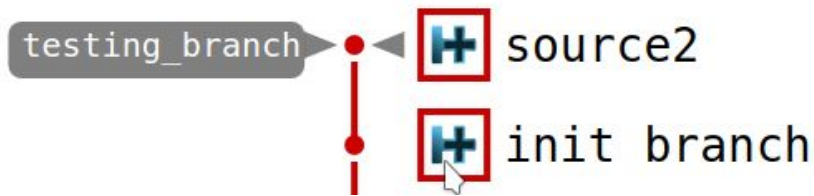
GIT - rejected push

developer 1

```
> git add source1.cc  
> git commit -am "source1"
```

developer 2

```
> git add source2.cc  
> git commit -am "source2"  
> git push
```



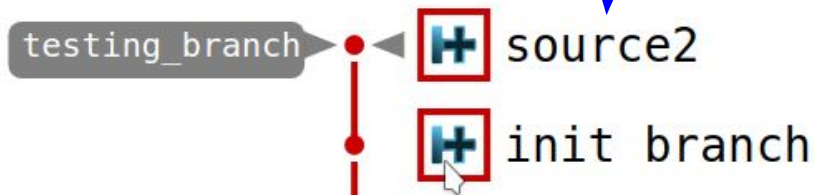
GIT - rejected push

developer 1

```
> git add source1.cc  
> git commit -am "source1"
```

developer 2

```
> git add source2.cc  
> git commit -am "source2"  
> git push
```



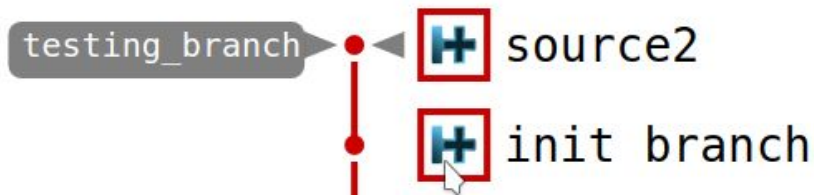
GIT - rejected push

developer 1

```
> git add source1.cc  
> git commit -am "source1"  
  
> git push
```

developer 2

```
> git add source2.cc  
> git commit -am "source2"  
> git push
```



GIT - rejected push

```
> git push
```

```
To https://git.sdr.kt.agh.edu.pl/dev_2016/testing-repo-2016.git
```

```
> git ! [rejected]      testing_branch -> testing_branch (fetch first)
```

```
> git error: failed to push some refs to
```

```
'https://git.sdr.kt.agh.edu.pl/dev_2016/testing-repo-2016.git'
```

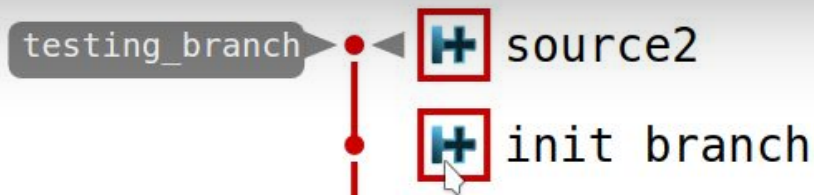
```
hint: Updates were rejected because the remote contains work that you do
```

```
hint: not have locally. This is usually caused by another repository pushing
```

```
hint: to the same ref. You may want to first integrate the remote changes
```

```
> > g hint: (e.g., 'git pull ...') before pushing again.
```

```
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```



GIT - rejected push

```
> git push
```

```
To https://git.sdr.kt.agh.edu.pl/dev_2016/testing-repo-2016.git
```

```
! [rejected]      testing_branch -> testing_branch (fetch first)
```

```
error: failed to push some refs to
```

```
'https://git.sdr.kt.agh.edu.pl/dev_2016/testing-repo-2016.git'
```

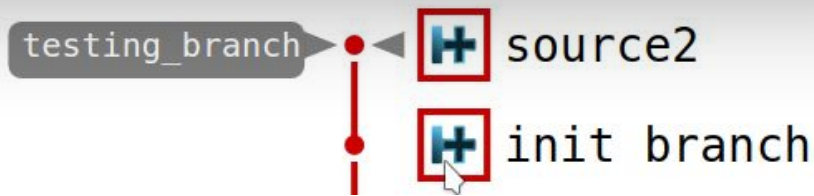
```
hint: Updates were rejected because the remote contains work that you do
```

```
hint: not have locally. This is usually caused by another repository pushing
```

```
hint: to the same ref. You may want to first integrate the remote changes
```

```
hint: (e.g., 'git pull ...') before pushing again.
```

```
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```



GIT - rejected push

```
~/g/testing-repo-2016 (testing_branch)> ls
```

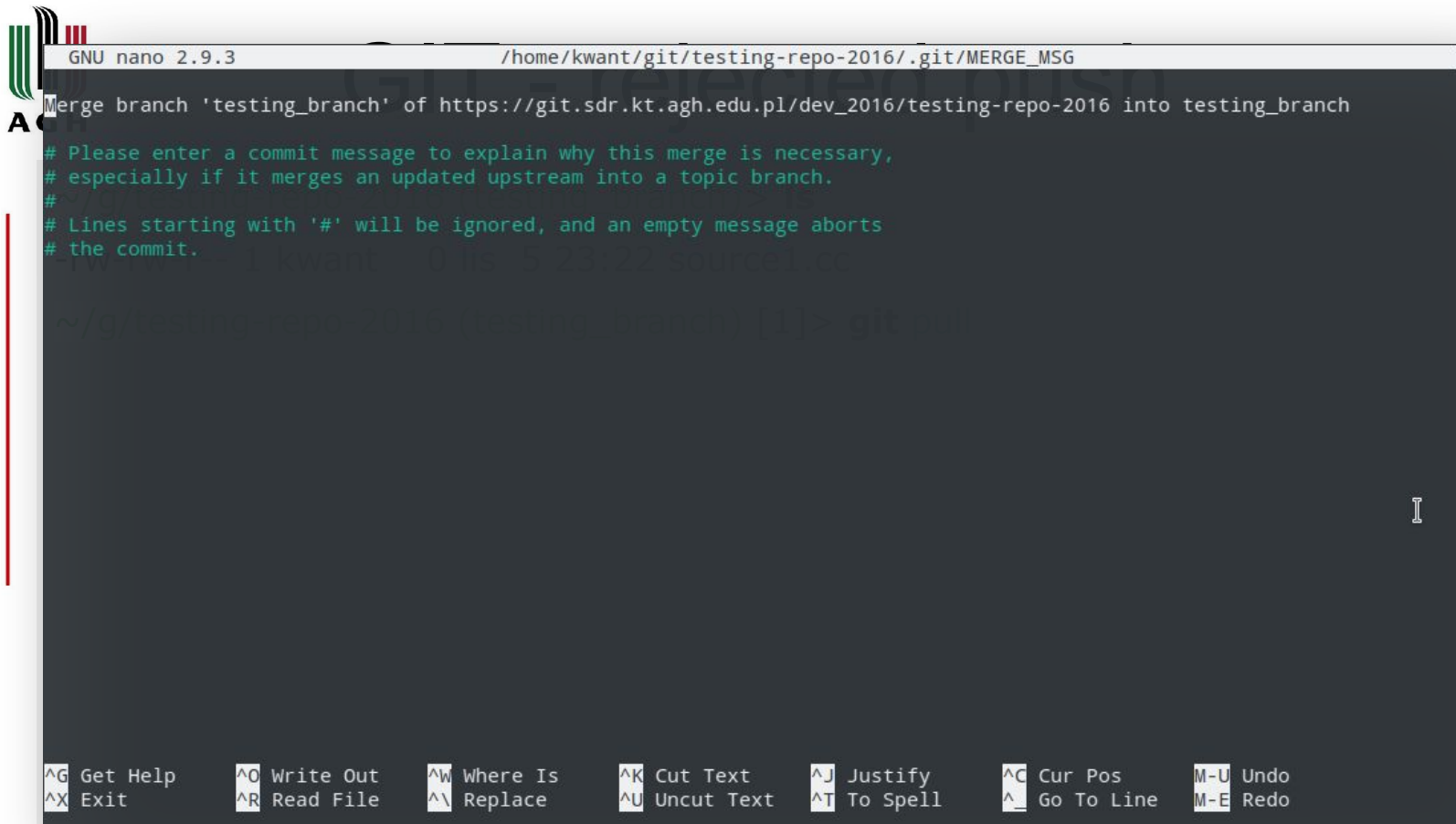
```
-rw-rw-r-- 1 kwant  0 lis  5 23:22 source1.cc
```

GIT - rejected push

```
~/g/testing-repo-2016 (testing_branch)> ls
```

```
-rw-rw-r-- 1 kwant  0 lis  5 23:22 source1.cc
```

```
~/g/testing-repo-2016 (testing_branch) [1]> git pull
```



The image shows a terminal window with the GNU nano 2.9.3 text editor. The editor is open to a file named `MERGE_MSG` in the directory `/home/kwant/git/testing-repo-2016/.git/`. The content of the file is a template for a merge commit message. The first line is a command to merge the `testing_branch` from a remote repository. The following lines are comments in green text, providing instructions on how to write the commit message. The editor's status bar at the bottom displays various keyboard shortcuts for navigation and editing.

```
GNU nano 2.9.3 /home/kwant/git/testing-repo-2016/.git/MERGE_MSG
Merge branch 'testing_branch' of https://git.sdr.kt.agh.edu.pl/dev_2016/testing-repo-2016 into testing_branch
# Please enter a commit message to explain why this merge is necessary,
# especially if it merges an updated upstream into a topic branch.
#
# Lines starting with '#' will be ignored, and an empty message aborts
# the commit.
~ /g/testing-repo-2016 (testing_branch) [1]> git pull
```

^G Get Help	^O Write Out	^W Where Is	^K Cut Text	^J Justify	^C Cur Pos	M-U Undo
^X Exit	^R Read File	^_ Replace	^U Uncut Text	^T To Spell	^_ Go To Line	M-E Redo

GIT - rejected push

```
~/g/testing-repo-2016 (testing_branch)> ls
```

```
-rw-rw-r-- 1 kwant  0 lis  5 23:22 source1.cc
```

```
~/g/testing-repo-2016 (testing_branch) [1]> git pull
```

```
From https://git.sdr.kt.agh.edu.pl/dev_2016/testing-repo-2016
```

```
ebd57ae..dc4ac53 testing_branch -> origin/testing_branch
```

Merge made by the 'recursive' strategy.

```
source2.cc | 0
```

```
1 file changed, 0 insertions(+), 0 deletions(-)
```

```
create mode 100644 source2.cc
```

GIT - rejected push

```
~/g/testing-repo-2016 (testing_branch)> ls
```

```
-rw-rw-r-- 1 kwant  0 lis  5 23:22 source1.cc
```

```
~/g/testing-repo-2016 (testing_branch) [1]> git pull
```

From https://git.sdr.kt.agh.edu.pl/dev_2016/testing-repo-2016

ebd57ae..dc4ac53 testing_branch -> origin/testing_branch

Merge made by the 'recursive' strategy.

```
source2.cc | 0
```

1 file changed, 0 insertions(+), 0 deletions(-)

create mode 100644 source2.cc

```
~/g/testing-repo-2016 (testing_branch)> ls
```

```
-rw-rw-r-- 1 kwant  0 lis  5 23:22 source1.cc
```

```
-rw-rw-r-- 1 kwant  0 lis  5 23:23 source2.cc
```

GIT - rejected push

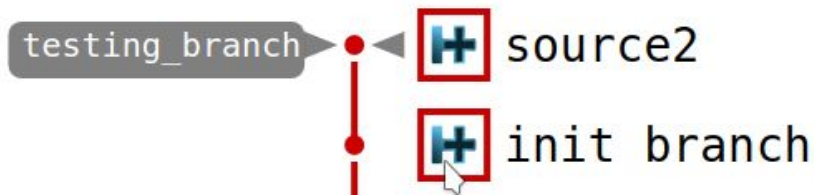
developer 1

```
> git add source1.cc  
> git commit -am "source1"
```

```
> git push
```

developer 2

```
> git add source2.cc  
> git commit -am "source2"  
> git push
```



GIT - rejected push

developer 1

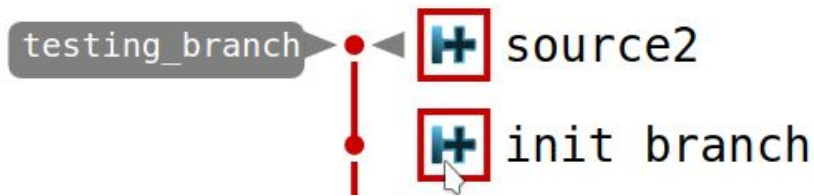
```
> git add source1.cc  
> git commit -am "source1"
```

```
> git push
```

```
> git pull
```

developer 2

```
> git add source2.cc  
> git commit -am "source2"  
> git push
```



GIT - rejected push

developer 1

```
> git add source1.cc  
> git commit -am "source1"
```

```
> git push
```

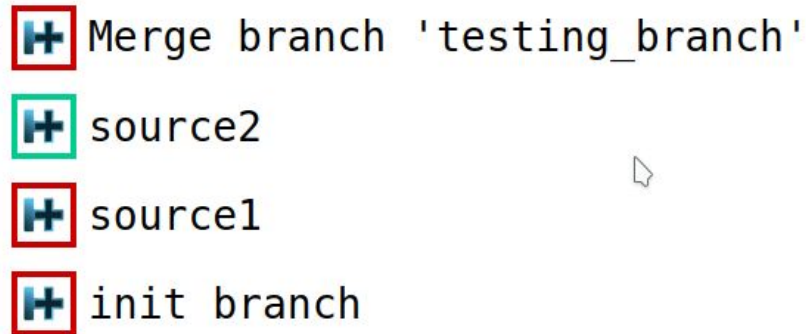
```
> git pull
```

```
> git push
```

developer 2

```
> git add source2.cc  
> git commit -am "source2"  
> git push
```

testing_branch



GIT - rejected push

- » Wnioski
 - “**pull**”-uj jak najczęściej
 - obowiązkowo zaczynaj pracę od **pull**
 - nie bój się **merge**, przyzwyczaj się, to jest “codzienność” gita

Dziękuję