Ministerstwo Nauki
i Szkolnictwa Wyższego

Regionalna Inicjatywa Doskonałości w Dyscyplinach
Informatyki, Elektrotechniki, Elektroniki, Automatyki i Robotyki
na Politechnice Częstochowskiej

Projekt w ramach programu Regionalna Inicjatywa Doskonałości, decyzja nr 020/RID/2018/19

Architektura Systemów Komputerowych

Procesory typu CISC i RISC, praca potokowa

dr hab. inż. Łukasz Szustak, prof. PCz
lszustak@icis.pcz.pl

Katedra Informatyki
Wydział Inżynierii Mechanicznej i Informatyki
Politechnika Częstochowska

Ministerstwo Nauki
i Szkolnictwa Wyższego

Regionalna Inicjatywa Doskonałości w Dyscyplinach
Informatyki, Elektrotechniki, Elektroniki, Automatyki i Robotyki
na Politechnice Częstochowskiej

Projekt w ramach programu Regionalna Inicjatywa Doskonałości, decyzja nr 020/RID/2018/19

Zamieszczane materiały służą wyłącznie do celów indywidualnego kształcenia. Nie wyrażam zgody na ich utrwalanie przekazywanie osobom trzecim ani rozpowszechnianie.

dr hab. inż. Łukasz Szustak, prof. PCz
lszustak@icis.pcz.pl

Katedra Informatyki
Wydział Inżynierii Mechanicznej i Informatyki
Politechnika Częstochowska

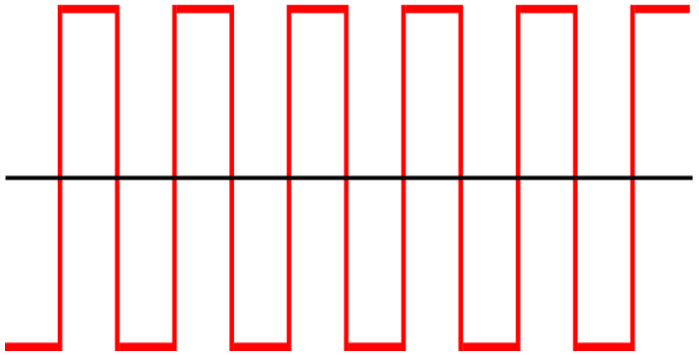
Wydział Inżynierii Mechanicznej i Informatyki
The Faculty of Mechanical Engineering and Computer Science

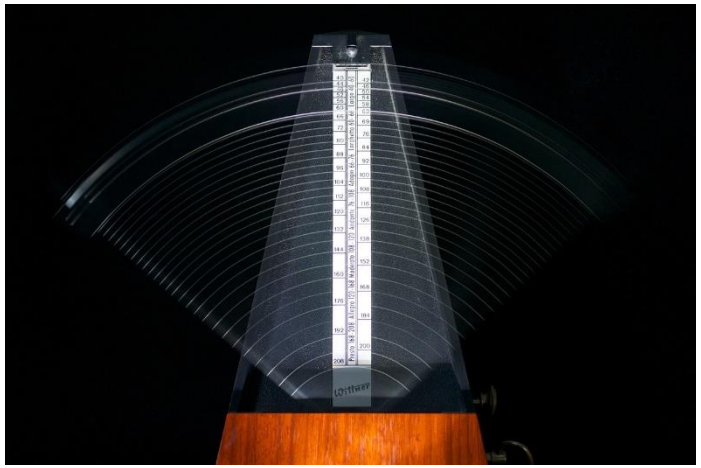
Komponenty

- Do podstawowych elementów architektury komputowej można zaliczyć:
  - Jednostkę centralna (**procesor** CPU), czyli przynajmniej jeden procesor (obecnie na popularności zyskują rozwiązania wieloprocesorowe)
  - Magistrale systemową poprzez którą komunikują się i przesyłaj dane komponenty systemu komputerowego
  - Pamięć, która służy do przechowywania danych i kodu instrukcji procesora
  - Urządzenia wejścia-wyjścia, np. klawiatura

Częstotliwość taktowania

- Większość procesorów, podobnie jak większość sekwencyjnych układów elektronicznych, jest z natury synchroniczna
- Oznacza to, że aby mogły wykonywa swoje zadania muszą mieć podany sygnał synchronizujący
- Taki sygnał, nazywany zegarowym, najczęściej przyjmuje formę prostokątnej fali w czasie:





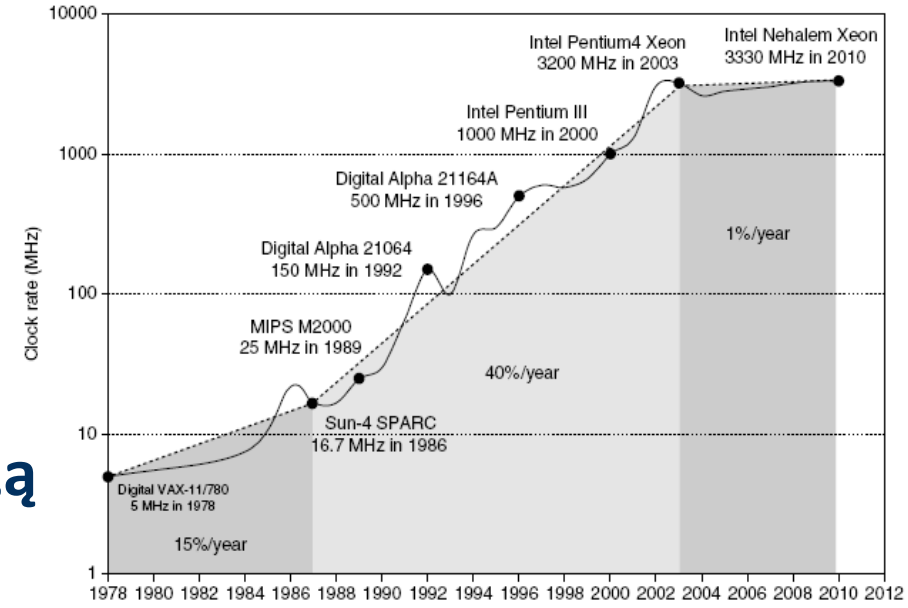
- **Miarą tego, ile razy sygnał zmienia się w ciągu sekundy, jest częstotliwość**

Częstotliwość taktowania – zegar bazowy

- Obecne systemy komputerowe zbudowane są z szeregowych łączy typu punkt-punkt
- Aby umożliwić komunikacje dla podzespołów (np. procesor, pamięci oraz łączy PCI Express), ich praca synchronizowana jest w oparciu o zegar bazowy (dla platform firmy Intel nazywany BCLK)
- W rezultacie częstotliwość taktowania danego podzespołu jest wyznaczana poprzez pomnożenie taktowanie zegara bazowego przez odpowiednie mnożniki
- Przykładowo: BLCK = 1000MHz oraz mnożnik = 2x odpowiada częstotliwości procesora równej 2000MHz (2GHz)
- Zegar bazowy jest podstawą do uzyskania częstotliwości taktowania innych podzespołów, a jego zwiększenie umożliwiania przyśpieszenie pracy całego systemu (wszystkich podzespołów, w tym procesora, pamięci oraz łączy PCI Express)
- Natomiast modyfikacja mnożnika zmienia szybkość pracy konkretnego komponentu, np. tylko procesora

Częstotliwość taktowania - procesor

- Procesor jest urządzeniem synchronicznym, co oznacza, że kolejne polecenie (kolejny krok wykonania instrukcji) jest inicjowane zmianą sygnału zegarowego
- Częstotliwość procesora określa liczę przetwarzanych cykli w ciągu 1 sekundy
- Zatem im większa częstotliwość (ilość zmian sygnału zegarowego), tym szybciej nasz procesor jest w stanie uruchamiać kolejne instrukcje/polecenia
- Niestety, każdy układ elektroniczny ma swoją graniczną częstotliwość powyżej której nie będzie działał stabilnie
- Częstotliwość zegara jest ważnym elementem wpływającym na wydajność procesora, jednakże **nie oznaczy to bezpośrednio, że procesor taktowany wyższą częstotliwością będzie zawsze szybszy niż inny - taktowany niższą**

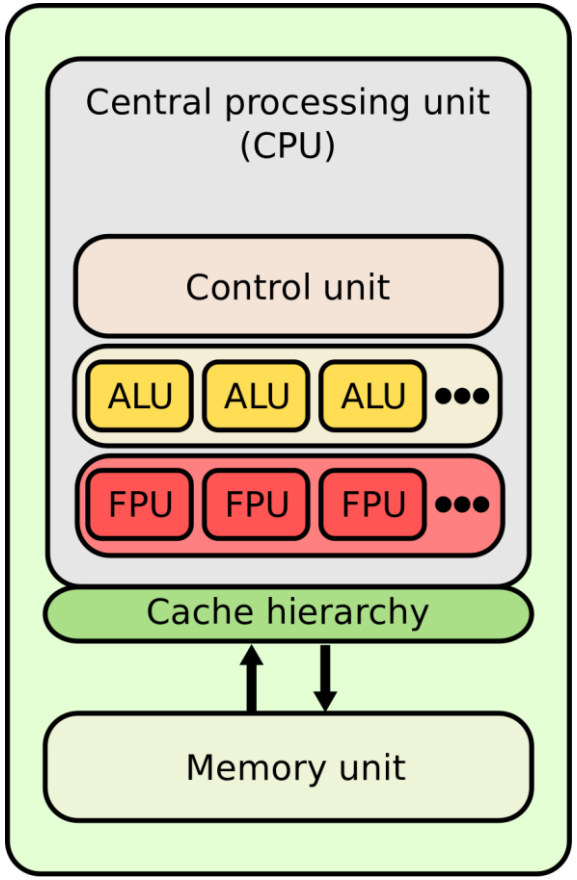


Jednostka centralna – procesor CPU

- Upraszczając, procesor można potraktować jako bardzo szybki i trochę bardziej skomplikowany kalkulator
- Procesor potrafi przekształcać liczby binarne (dodawać, odejmować, mnożyć, dzielić, negować, itp.) w sposób definiowany poprzez program składający się również z liczb binarnych
- Procesor (CPU) jest tym komponentem w systemie komputerowym, który interpretuje program i zgodnie z nim przetwarza dane
- Podstawową funkcją większości procesorów (niezależnie od ich konstrukcji fizycznej) jest wykonanie programu, czyli sekwencji instrukcji przechowywanych w pamięci
- Wykonanie każdej z tych instrukcji jest również rozbite na kroki

Jednostka centralna – procesor CPU

- Zazwyczaj wyróżnia się następujące główne kroki potrzebne do wykonania pojedynczej instrukcji:
  - Pobieranie rozkazów
  - Interpretacja rozkazów
  - Wykonanie rozkazów
  - Zapis wyników

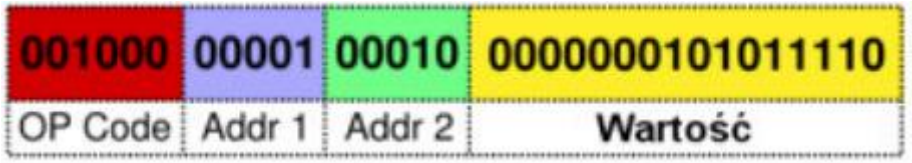


Jednostka centralna CPU: pobieranie rozkazów

- Czyli odczyt poleceń z pamięci
- Każde polecenie jest reprezentowane przez pewną liczbę binarną
- Liczbę "rozumie" nasz procesor - to już kwestia konstrukcji, ale zawsze jest to skończona liczba
- Położenie instrukcji w pamięci jest zdefiniowane przez specjalny blok, nazywany licznikiem rozkazów lub słowem stan programu (program counter, PC), który przechowuje adres aktualnie wykonywanej instrukcji w pamięci operacyjnej
- Po pobraniu następuje zwiększenie PC o długo pobranego rozkazu (w jednostkach pamięci, czyli najczęściej w bajtach)
- Natomiast w przypadku konieczności wykonania skoku - zawartość PC może być zmieniana przez odpowiednią instrukcję procesora

Jednostka centralna CPU: interpretacja rozkazów

- Po odczytaniu polecenia z pamięci, w pierwszym etapie przetwarzania, procesor interpretuje polecenie
- W trakcie tego kroku instrukcja jest rozbijana na części, które mają znaczenie dla poszczególnych bloków procesora
- Sposób w jaki procesor interpretuje instrukcje, jest zależny od jego architektury i zestawu instrukcji (instruction set architecture, ISA)
- Przykładowo, w jednej długiej liczbie będącej instrukcją, początkowa część bitów (nazywana opcode) jest znacznikiem operacji do wykonania, a pozostała część bitów zazwyczaj zawiera informacje wymagane przez instrukcję, tak jak adresy składników dla operacji dodawania



Jednostka centralna CPU: wykonanie rozkazów

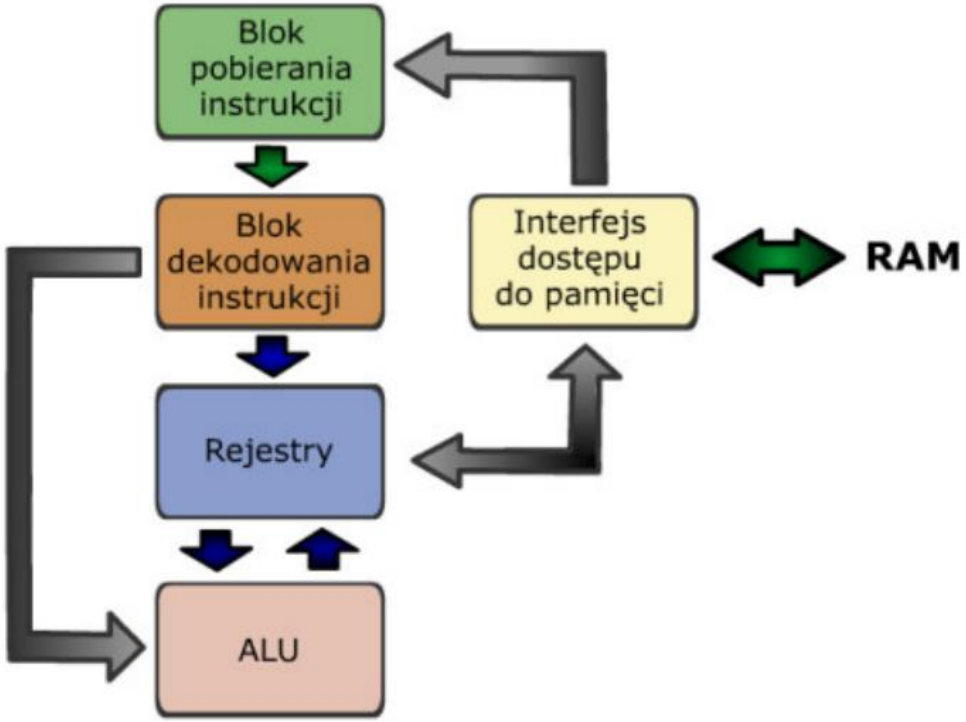
- Po pobraniu i zdekodowaniu przychodzi pora na rzeczywiste wykonanie polecenia
- Podczas tego kroku różne moduły CPU współpracuj ze sobą w celu wykonania polecenia
- Rozpatrzmy jako przykład dodawanie dwóch liczb: jednostka arytmetyczno-logiczna (ALU) zawiera układ elektroniczny, który jest w stanie wykonać proste operacje arytmetyczno - logiczne (jak dodawanie, czy operacje logiczne na bitach) dla danych wejściowych znajdujących się w rejestrach
- Układ ten jest więc uruchamiany, a następnie wynik dodawania jest umieszczany w rejestrze wyjściowym
- Należy zauważyć, że wykonanie instrukcji odbywa się na danych znajdujących się w rejestrach

Jednostka centralna CPU: zapis wyników

- Finalna faza - efekty pracy procesora są zapamiętywane w rejestrze wyjściowym, skąd mogą zostać odesłane do pamięci operacyjnej, lub też mogą być wykorzystane w kolejnym kroku obliczeń (przy wykonywaniu kolejnej instrukcji)
- Część instrukcji zmienia zawarto licznika instrukcji (PC) w celu wykonania skoku w programie – pozwala to na tworzenie pętli i instrukcji warunkowych

Jednostka centralna CPU: koncepcja działania

- Aby procesor miał możliwość wykonywania powyższych zadań musi dysponować małą pamięcią wewnętrzną, która wymagana jest do czasowego przechowywania danych i rozkazów
- „Większość” systemów komputerowych opiera się na założeniu, że procesor nie operuje na danych bezpośrednio umieszczonych w pamięci operacyjnej, lecz za każdym razem następuje przekazanie zarówno danych, jak i wyników obliczeń, z/do pamięci za pośrednictwem **rejestrów**



- Rejestrami procesora należy nazywać wydzielone, bardzo szybkie komórki pamięci, umieszczone w samym procesorze
- Umożliwiaj one szybki dostęp do potrzebnych danych lub rozkazów

Jednostka centralna CPU: koncepcja działania

- Ponieważ procesor to nic innego jak „skomplikowany kalkulator”, który dodatkowo pobiera informacje o czynnościach do wykonania z pamięci operacyjnej, należałoby zastanowić się w jaki sposób przebiegałoby zrealizowanie wyrażenia (x+y)*(x-z):
  - Najpierw zostają ustalone wartości danych oznaczonych we wzorze literami x, y i z
  - Wpisywana jest wartość x
  - Wybierana jest operacja +
  - Wpisywana jest wartość y
  - Wybierana jest operacja =
  - Odczytywana jest wynik z operacji, a następnie zapamiętywany
  - AC (kasuj)
  - Wpisywana jest wartość x
  - Wybierana jest operacja -
  - Wpisywana jest wartość z
  - Wybierana jest operacja =
  - Wybierana jest operacja *
  - Wpisywana jest wartość wcześniej zapamiętana
  - Wybierana jest operacja =
  - Odczytywana jest wynik z operacji, a następnie zapamiętywany

Jednostka centralna CPU: koncepcja działania

- Ponieważ procesor to nic innego jak „skomplikowany kalkulator”, który dodatkowo pobiera informacje o czynnościach do wykonania z pamięci operacyjnej, należałoby zastanowić się w jaki sposób przebiegałoby zrealizowanie wyrażenia (x+y)*(x-z):
  - Najpierw zostają ustalone wartości danych oznaczonych we wzorze literami x, y i z
  - Wpisywana jest wartość x
  - Wybierana jest operacja +
  - Wpisywana jest wartość y
  - Wybierana jest operacja =
  - Odczytywana jest wynik z operacji, a następnie zapamiętywany
  - AC (kasuj)
  - Wpisywana jest wartość x
  - Wybierana jest operacja -
  - Wpisywana jest wartość z
  - Wybierana jest operacja =
  - Wybierana jest operacja *
  - Wpisywana jest wartość wcześniej zapamiętana
  - Wybierana jest operacja =
  - Odczytywana jest wynik z operacji, a następnie zapamiętywany

Procesor realizuje to zadanie w zbliżony sposób, przy czym każda podstawowa czynność to pojedynczy rozkaz procesora

Jednostka centralna CPU: koncepcja działania

- Załóżmy, że rozkazy procesora mogą mieć jeden z trzech postaci:
  - pobierz dane z pamięci operacyjnej do rejestru (rejestr to "prywatna" jednostka pamięci należąca do procesora)
  - wykonaj operacje arytmetyczne na danych w rejestrach
  - prześlij dane z rejestru do pamięci operacyjnej
- Załóżmy też , że przed uruchomieniem obliczeń, gdzieś w pamięci operacyjnej, pod adresami nazwanymi dla wygody x, y, z umieszczone są wartości danych wejściowych, dla których ma zostać wykonane działanie (x+y)*(x-z)

Jednostka centralna CPU: koncepcja działania

- Najpierw zostają ustalone wartości danych oznaczonych we wzorze literami x, y i z
- Wpisywana jest wartość x
- Wybierana jest operacja +
- Wpisywana jest wartość y
- Wybierana jest operacja =
- Odczytywana jest wynik z operacji, a następnie zapamiętywany
- AC (kasuj)
- Wpisywana jest wartość x
- Wybierana jest operacja -
- Wpisywana jest wartość z
- Wybierana jest operacja =
- Wybierana jest operacja *
- Wpisywana jest wartość wcześniej zapamiętana
- Wybierana jest operacja =
- Odczytywana jest wynik z operacji, a następnie zapamiętywany

- Pobierz wartość spod adresu x do rejestru R1
- Pobierz wartość spod adresu y do rejestru R2
- Dodaj zawartość rejestrów R1 i R2 i wynik wstaw do R1
- Prześlij wynik z rejestru R1 pod adres t
- Pobierz warto spod adresu x do rejestru R1
- Pobierz warto spod adresu z do rejestru R2
- Odejmij zawarto rejestru R2 od rejestru R1 i wynik wstaw do R1
- Pobierz wartość spod adresu t do rejestru R2
- Pomnóż zawarto rejestrów R1 i R2 i wynik wstaw do R1
- Prześlij zawartość z rejestru R1 pod adres w

Jednostka centralna CPU: koncepcja działania

```
int concept(int x, int y, int z) {
    return (x+y)*(x-z);
}
```

```
concept(int, int, int):
        push    rbp
        mov     rbp, rsp
        mov     DWORD PTR [rbp-4], edi
        mov     DWORD PTR [rbp-8], esi
        mov     DWORD PTR [rbp-12], edx
        mov     edx, DWORD PTR [rbp-4]
        mov     eax, DWORD PTR [rbp-8]
        add     edx, eax
        mov     eax, DWORD PTR [rbp-4]
        sub     eax, DWORD PTR [rbp-12]
        imul    eax, edx
        pop     rbp
        ret
```

Pobranie wartości **x** i **y** do rejestrów **edx**, **eax**

Dodawanie i zapis wyniku w rejestrze **edx**

Pobranie wartości **z** do rejestru **eax**

Odejmowanie i zapis wyniku w **eax**

Mnożenie i zapis wyniku w **eax**

Jednostka centralna CPU: rejestry

- Rejestry są używane do przechowywania danych wewnętrznych w procesorze, a niektóre z nich zawierają informacje potrzebne do zarządzania wykonywaniem zadań
- Można pokusić się o stwierdzenie, że im więcej rejestrów, tym procesor będzie realizował bardziej efektywnie swoje zadania
- Liczba i rodzaj rejestrów we współczesnych procesorach różni się w zależności od producenta i generacji architektury

Jednostka centralna CPU: rejestry - rodzaje

- **danych** – służą tylko do przechowywania argumentów i wyników operacji arytmetyczno-logicznych
- **adresowe** – służą do przechowywania adresów i do operacji arytmetycznych na adresach
- **ogólnego przeznaczenia** – mogą pełnić rolę zarówno rejestrów danych, jak i adresowych
- **specjalizowane** – pełnią ściśle określoną funkcję, np. akumulator, czy wskaźnik stosu
- **stanu, znaczników** – przechowuje jednobitowe znaczniki stanu i sterowania procesora
- **licznik programu** – zawiera adres bieżącej instrukcji, która ma być wykonana

Jednostka centralna CPU: rejestry - rodzaje

- **zmiennopozycyjne** – służą do przechowywania liczb w formacie zmiennopozycyjnym
- **wektorowe** – przechowują wektor (tablicę) wartości
- **segmentowe** – służą do implementacji segmentowanego modelu pamięci
- Ponadto mogą występować różnego rodzaju rejestry, które nie są widoczne dla programisty lub są widoczne tylko w trybie uprzywilejowanym. Są to rejestry służące do:
  - **zarządzania pamięcią**
  - **Debugowania**
- **…**

Jednostka centralna CPU: rejestry - „rozmiar”

- Sposób w jaki procesor "pamięta" przetwarzane dane jest jego cech konstrukcyjną, która wpływa na praktycznie wszystkie dostępne funkcje
- Pojedynczy rejestr jest określony liczbą bitów, czyli logicznych zer lub jedynek
- **W zależności od ilości bitów przechowywanych w rejestrze, i co za tym idzie - przetwarzanych przez procesor, zmienia się rozmiar i precyzja danych na których operuje**
- Liczba bitów przetwarzanych w jednym cyklu procesora przez jednostki wykonawcze (ALU, FPU,…) jest nazywana słowem procesora
- Określenie procesor 32 bitowy, 64 bitowy, itd.. odnosi się do długości słowa przetwarzanego przez jednostki ALU i FPU procesora w jednej instrukcji

Jednostka centralna CPU: rejestry - „rozmiar”

- Długość poszczególnych rejestrów różni się dla różnych procesorów i architektury, czasami nawet różni się dla poszczególnych modułów jednego procesora
- Przykładowo 8-bitowy procesor jest w stanie w jednej instrukcji przetwarzać 8 bitów
  - Oznacza to, że dla liczb całkowitych procesor może przetwarzać liczby z zakresu od 0 do 255
- Jeśli rejestr używany do przekazywania adresów miałby również 8 bitów, to komputer oparty na takim procesorze miałby fizycznie ograniczoną wielkość pamięci do 256 komórek (maksymalna możliwa pojemność RAM wynosiłaby 256 bajtów!)
- Z tego powodu w starszych architektach CPU rejestry adresowe były dłuższe, przykładowo dla procesorów 8-bitowych miały 16 bitów, co dawało 65536 możliwych adresów (64 KiB RAM)
- „Dawna” architektura 32-bitowa (rejestry adresowe o rozmiarze 32 bity) umożliwiała przechowywanie 2^32 adresów co przekłada się na maksymalny obsługiwany rozmiar pamięci głównej 4GiB
- Obecna architektura 64-bitowa (rejestry adresowe o rozmiarze 64 bitów) umożliwiają przechowywanie 2^64 adresów co przekłada się na ogromny maksymalny rozmiar pamięci

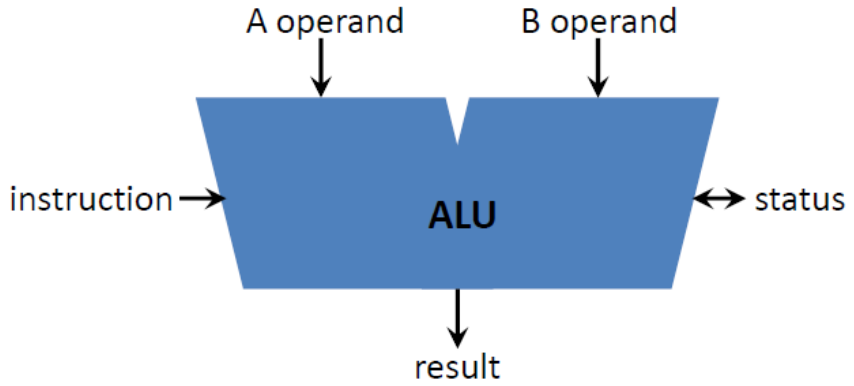
Jednostka centralna CPU: rejestry - „ilość”

- AVX-128 - rejestry o nazwach, xmm, np xmm0
- AVX-256 - rejestry o nazwach, ymm, np xmm0
- AVX-512 - rejestry o nazwach, zmm, np xmm0

Liczba rejestrów	Sandy Bridge	Haswell	Skylake
Integer Register File	160	168	180
FP Register File	144	168	168
SIMD Register File	16	16	32

Jednostka centralna CPU: jednostka ALU

- Jednostka arytmetyczno-logiczna (z ang. Arithmetic and Logical Unit lub Arithmetic Logic Unit, ALU) czyli układ cyfrowy, wykonujący operacje arytmetyczne, np. dodawanie, odejmowanie oraz logiczne
- Obecne systemy zawierają kilka ALU i mogą ich używać jednocześnie wykonując jednocześnie kolejne instrukcje
  - Jednakże, często ALU są różne: np.. trzy jednostki ALU potrafią wykonywać podstawowe operacje (dodawanie, odejmowanie i logiczne), a jedynie jedno potrafi mnożyć i dzielić
  - W takim przypadku procesor w jednym cyklu może wykonać np. 1 mnożenie i 2 dodawania, ale nie może wykonać 2 mnożeń



Jednostka centralna CPU: jednostka FPU

- Obecnie podstawowym komponentem procesora jest jednostka zmiennopozycyjna
- "Dawne" procesor wykonywał obliczenia jedynie na liczbach całkowitych - jakiekolwiek operacje na liczbach zmiennoprzecinkowych wymagały wykonywania wielu instrukcji prostych na liczbach całkowitych
  - Innymi słowy - dodanie 0.1 do 0.1 zajmowało kilkanaście razy więcej czasu niż dodanie 1+1
- Zazwyczaj ALU wykonuje operacje na liczbach w kodzie uzupełnienia do dwóch
- FPU jest odpowiednikiem ALU, tyle że pozwalającym wykonywać podstawowe operacje na zmiennych, które nie są liczbami całkowitymi, co powoduje znaczące przyspieszenie takich obliczeń

Jednostka centralna CPU: budowa

- Schemat blokowy rdzenia mikro architektury Intel Skylake Server

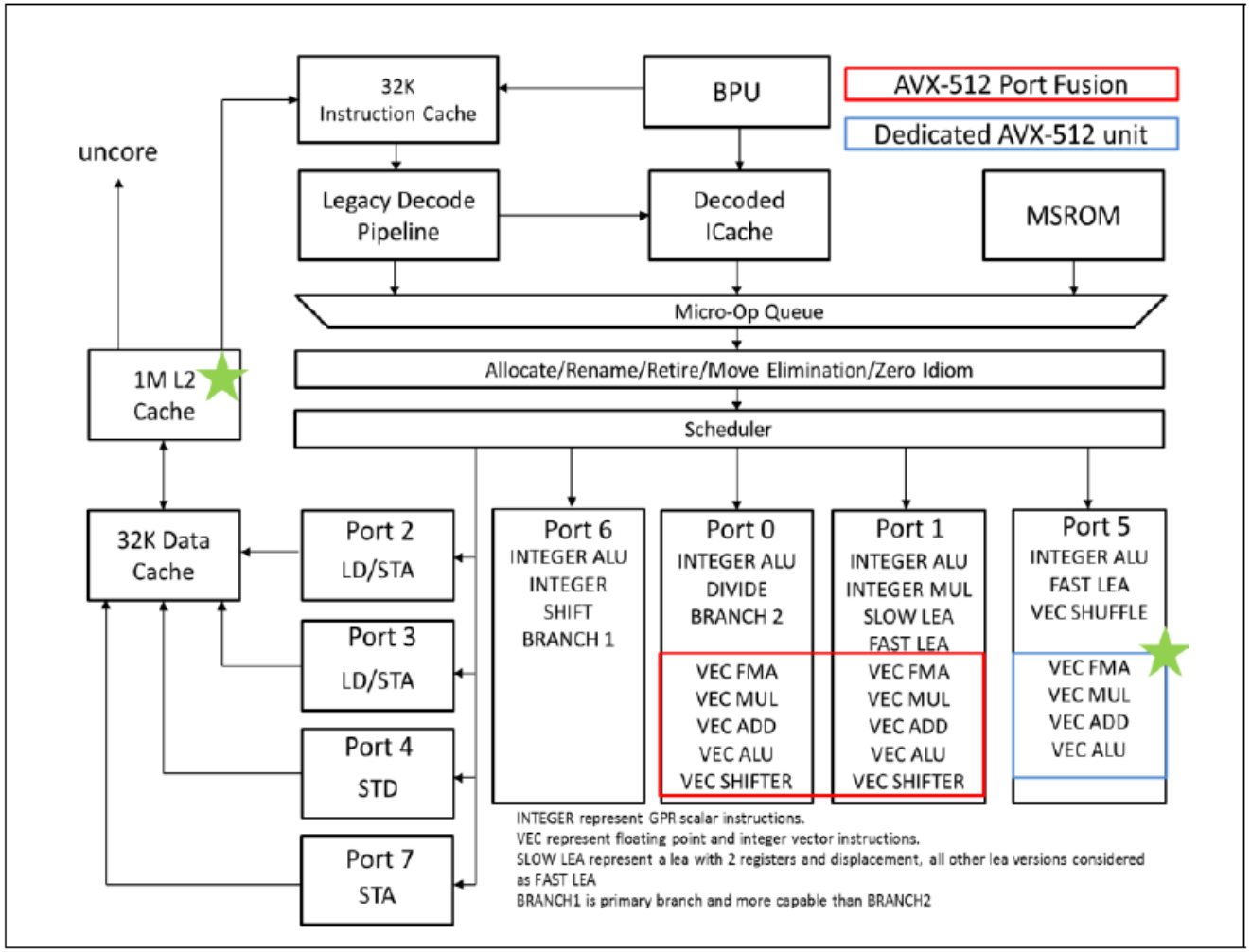


Figure 2-2. Processor Core Pipeline Functionality of the Skylake Server Microarchitecture

https://en.wikichip.org/wiki/intel/microarchitectures/skylake_(client)

Jednostka centralna CPU: budowa

- Schemat blokowy rdzenia mikro architektury Intel Skylake

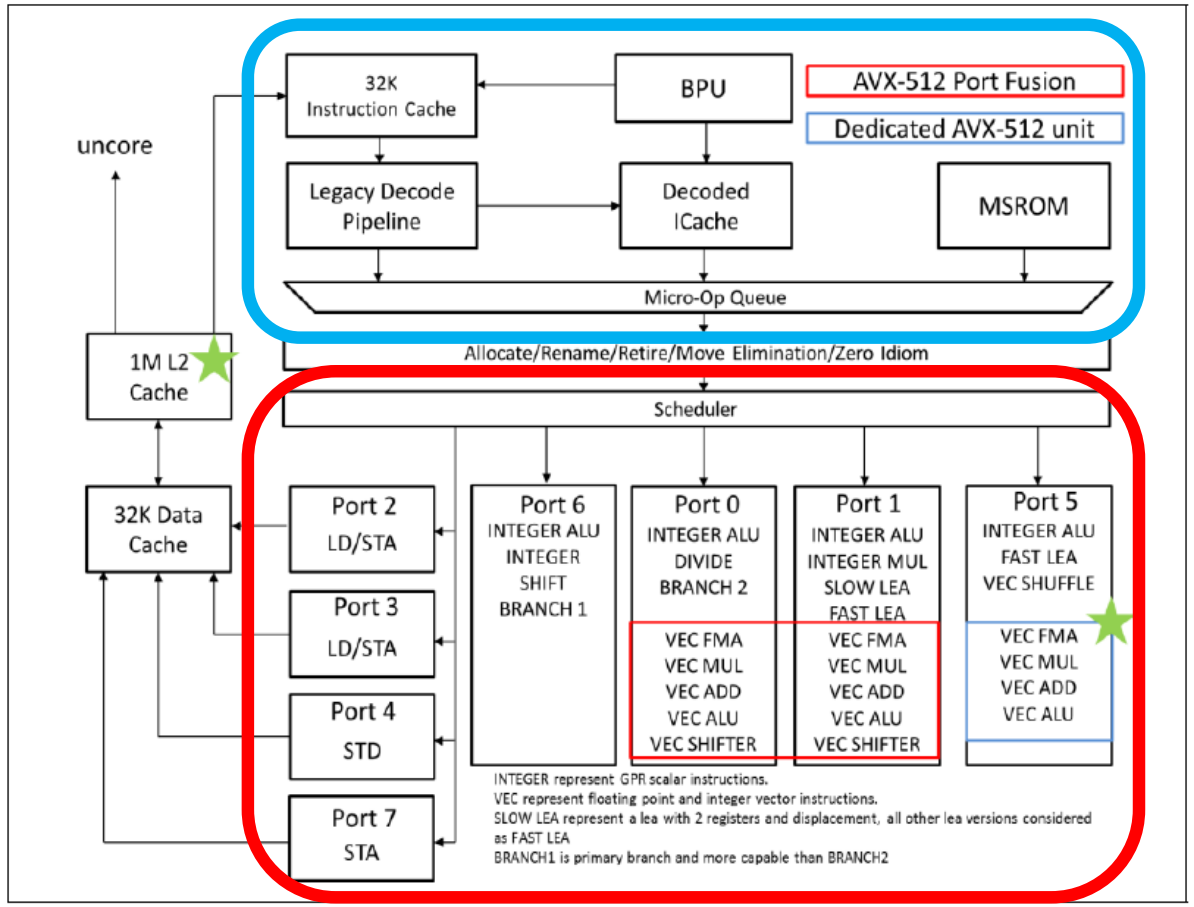


Figure 2-2. Processor Core Pipeline Functionality of the Skylake Server Microarchitecture

Front-end - jednostka odpowiedzialna za dekodowanie i kolejkowanie instrukcji

Out-of-Order Engine jednostka przetwarzająca instrukcje poza kolejności zdefiniowania

Jednostka centralna CPU: budowa - Front-end

- Jednostka Branch Prediction Unit wybiera kolejny blok kodu do wykonania z programu
- Blok kodu poszukiwany jest w zdekodowanym blokiem instrukcji, Icache, L2, LLC lub pamięć główna
- Następnie każda instrukcja jest pobierana i dekodowana na jedną lub więcej mikrooperacji, a następnie przesyłane są do bloku Rename, w którym są grupowane, selekcjonowane i zapisywane w kolejce FIFO
- Mikrooperacje trafiają do dyspozytora, gdzie są przetwarzane poza kolejnością

Jednostka centralna CPU: budowa - Out-of-Order Engine

- Jednostka the Out-of-Order Engine składa się z:
  - **Renamer:** przenosi mikroperacje z kolejki (micro-op queue) do dyspozytora, oraz przypisuje numer portu na którym ona będzie wykonywana
  - **Scheduler:** zarządza i kontroluje przydzielone do wykonania poza kolejnością mikroopracje na poszczególnych portach
  - **Execution Core**: część odpowiedzialna za wykonywaniem poza kolejnością instrukcji przy użyciu dostępnych portów. Procesor może wykonać do ośmiu mikrooperacji w jednym cyklu zegara
- Dyspozytor odpowiada za zmianę kolejności następujących po sobie instrukcji oraz ułożenie ich w taki sposób, aby można było wykonać jak najwięcej z nich jednocześnie

Jednostka centralna CPU: budowa - Out-of-Order Engine

- Dyspozytor zasobów może maksymalnie uruchamiać osiem mikro-operacji w każdym cyklu, z użyciem 8 portów
- Cztery porty utylizują zasoby obliczeniowe: każdy z tych portów odnosi się do innej jednostki ALU, dwa z nich odnoszą się do dedykowanych jednostek FMA, jak również jednostek SIMD oraz zmiennopozycyjnych
- Pozostałe cztery porty determinują wykonywanie instrukcji na pamięci
- Uwzględniając przetwarzanie potokowe, the out-of-order engine dla Skylake umożliwia jednoczesne przetwarzanie aż 224 mikrooperacji (Haswell - 192, Sandy Bridge - 168)

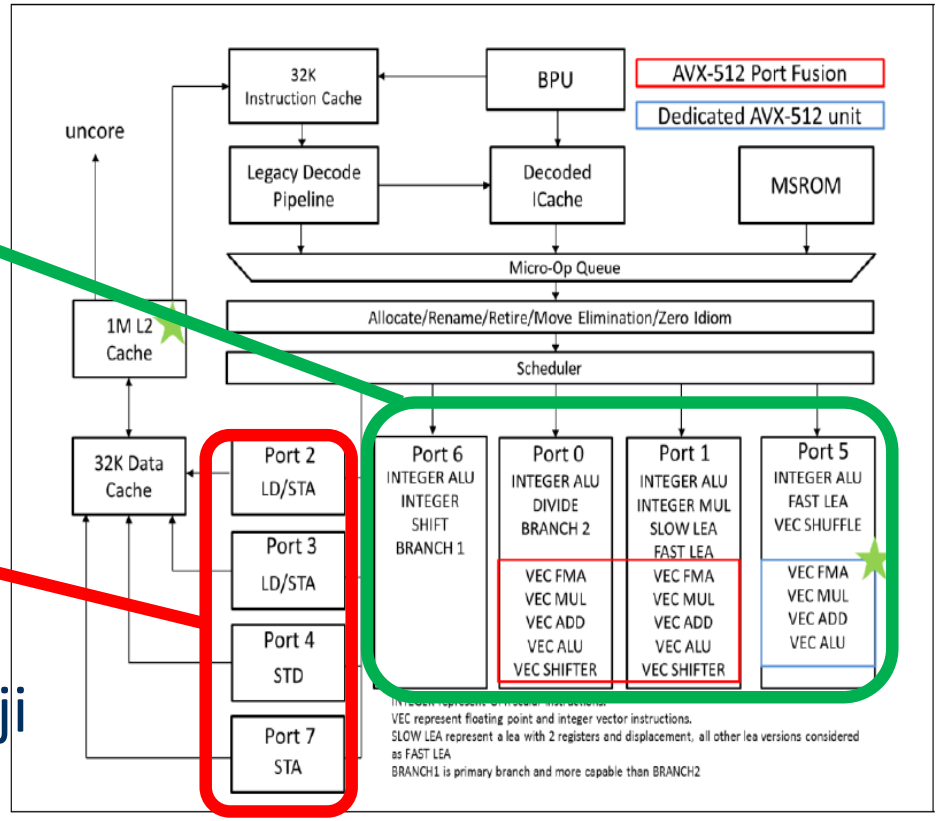


Figure 2-2. Processor Core Pipeline Functionality of the Skylake Server Microarchitecture

Jednostka centralna CPU: budowa - Out-of-Order Engine

- Rodzaj instrukcji realizowanych na poszczególnych portach (#of unit)

Table 2-3. Skylake Microarchitecture Execution Units and Representative Instructions[1]

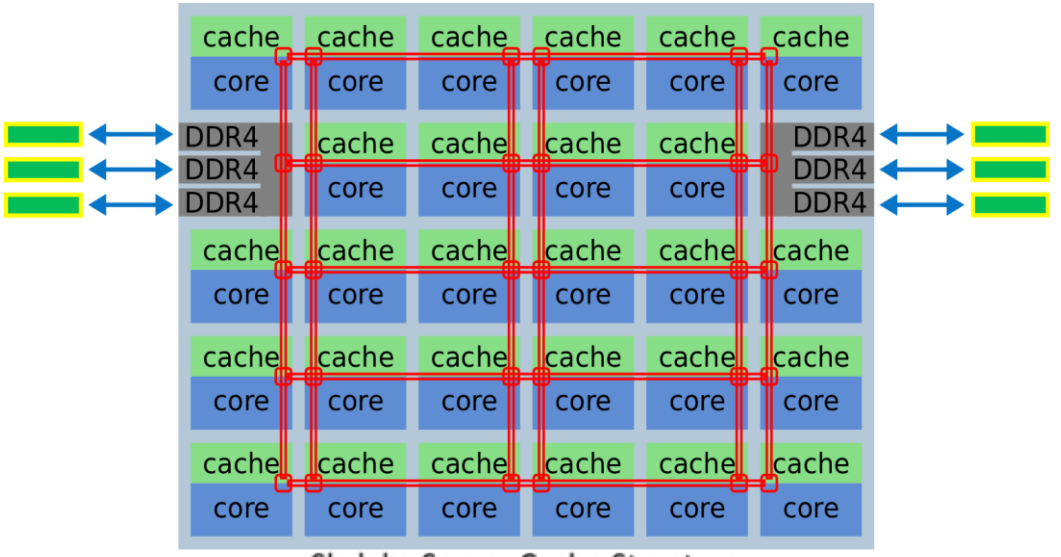
Execution Unit	# of Unit	Instructions
ALU	4	add, and, cmp, or, test, xor, movzx, movsx, mov, (v)movdqu, (v)movdqa, (v)movap*, (v)movup*
SHFT	2	sal, shl, rol, adc, sarx, adcx, adox, etc.
Slow Int	1	mul, imul, bsr, rcl, shld, mulx, pdep, etc.
BM	2	andn, bextr, blsi, blsmsk, bzhi, etc
Vec ALU	3	(v)pand, (v)por, (v)pxor, (v)movq, (v)movq, (v)movap*, (v)movup*, (v)andp*, (v)orp*, (v)paddb/w/d/q, (v)blendv*, (v)blendp*, (v)pblendd
Vec_Shft	2	(v)psllv*, (v)psrlv*, vector shift count in imm8
Vec Add	2	(v)addp*, (v)cmpp*, (v)max*, (v)min*, (v)padds*, (v)paddus*, (v)psign, (v)pabs, (v)pavgb, (v)pcmpeq*, (v)pmax, (v)cvtps2dq, (v)cvtdq2ps, (v)cvtsd2si, (v)cvtss2si
Shuffle	1	(v)shufp*, vperm*, (v)pack*, (v)unpck*, (v)punpck*, (v)pshuf*, (v)pslldq, (v)alignr, (v)pmovzx*, vbroadcast*, (v)pslldq, (v)psrldq, (v)pblendw
Vec Mul	2	(v)mul*, (v)pmul*, (v)pmadd*,
SIMD Misc	1	STTNI, (v)pclmulqdq, (v)psadw, vector shift count in xmm,
FP Mov	1	(v)movsd/ss, (v)movd gpr,
DIVIDE	1	divp*, divs*, vdiv*, sqrt*, vsqrt*, rcp*, vrcp*, rsqrt*, idiv

Jednostka centralna CPU: pamięć podręczna

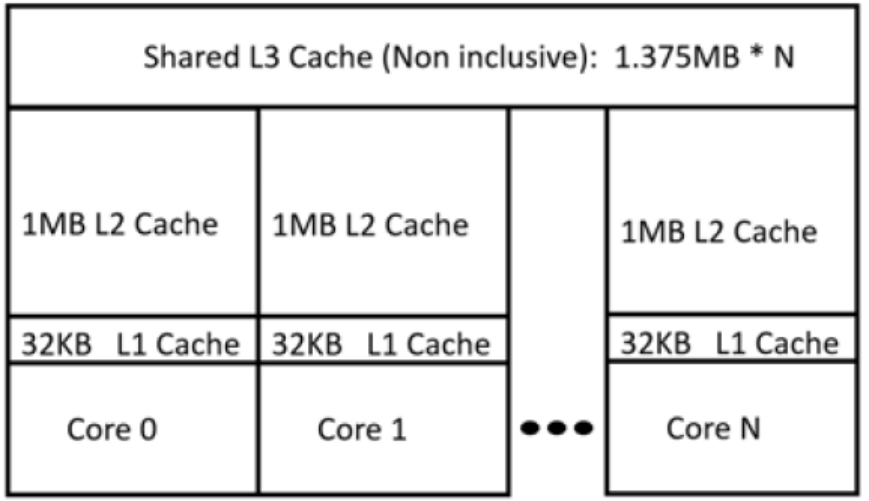
- Innym również ważnym elementem procesora jest pamięć podręczna
- Powszechnie dostęp do danych przechowywanych w pamięci operacyjnej jest zdecydowanie wolniejszy, aniżeli czas operacje wykonywanych przez procesor na tych danych
- Pamięć podręczna jest rozwiązaniem tego ograniczenia:
  - Pamięć cache jest miejscem, w którym ostatnio pobierane dane z pamięci operacyjnej są przechowywane w pamięci o zdecydowanie lepszych parametrach szybkościowych
  - Pamięć „cache” jest elementem właściwie wszystkich podsystemów - nie tylko procesor, ale równie np. dostępu do dysku, który może być również buforowany w pamięci głównej
- Obecnie wyróżniamy 2- lub 3- poziomowe pamięci cache:
- Każdy kolejny poziom pamięci cechuje coraz większy rozmiar
- Każdy kolejny poziom pamięci cache jest wolniejszy
- Pamięć podręczna w większości przypadków jest zintegrowana z procesorem

Jednostka centralna CPU: pamięć podręczna L1

- Hierarchia pamięci podręcznej na przykładzie procesora firmy Intel Skylake obejmuje:
  - Poziom pamięci podręcznej L1, który podzielony jest na blok danych i instrukcji: L1 ICache i L1 DCache
  - Poziom pamięci podręcznej L2
  - Każdy rdzeń posiada własną pamięć L1 i L2
  - L1 może być współdzielony przez dwa logiczne procesory (Simultaneous multithreading, SMT), jeżeli dany model procesora wspiera technologię np. „Intel HyperThreading”
  - Dane i instrukcje współdzielą L2
  - Wszystkie rdzenie są połączone do ostaniego poziomu pamięci podręcznej LLC (last level cache) poprzez dedykowany siatkę 2D

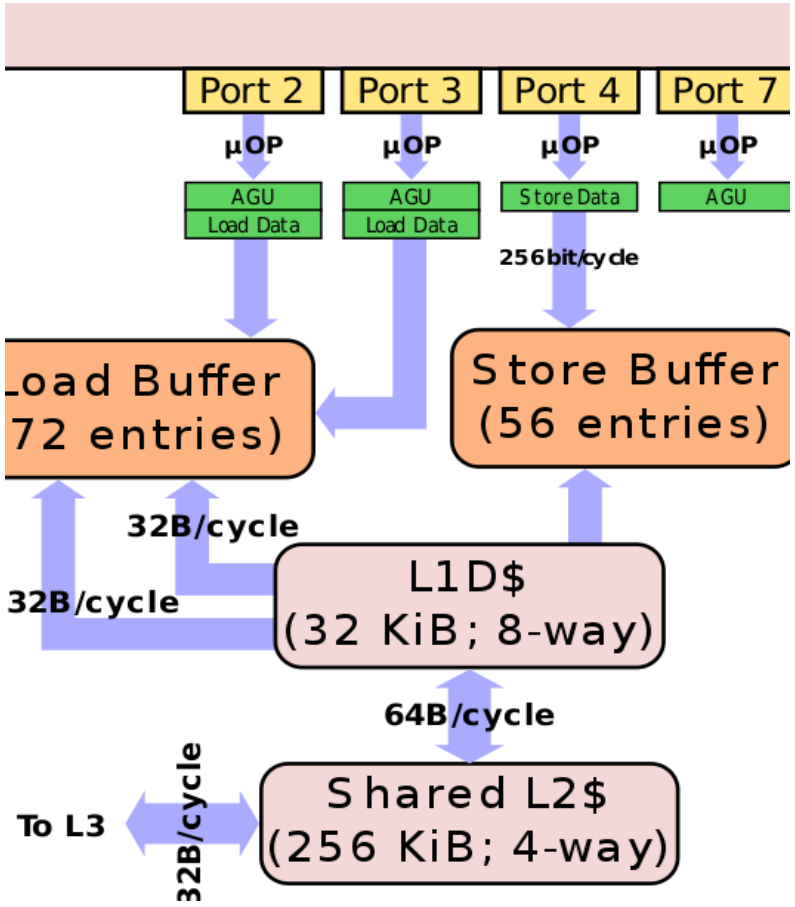


Skylake Server Cache Structure



Jednostka centralna CPU: pamięć podręczna L1

- L1 DCache sprawuję kontrolę nad przebiegiem instrukcji load i store
- Umożliwia wykonywanie instrukcji load i store poza kolejnością ich zdefiniowania przez kompilator/programiste
- Jednostka DCache jest zorganizowana jako 32 KiB blok, który składa się z kilkuset linijek pamięci cache o rozmiarze 64B
- Każda linijka zbudowana jest w postaci ośmiu 8-bajtowych banków
- Instrukcje load i store są wykonywane poprzez 32-bajtowe dostępy do Dcache, które są realizowane poprzez 8 wejść
- L1DC może obsłużyć dwie instrukcje load na 32-bajtowych danych oraz jedną instrukcje store na 32-bajtowych danych w każdym cyklu
  - Przepustowość: 64 B/cycle load i 32 B/cycle store
  - Porty 2 i 3 obsługują pobranie, port 4 zapis
- Dodatkowo, mikroarchitektura Skylake umożliwia buforowanie 72 operacji typu load i 56 typu store



https://en.wikichip.org/wiki/intel/microarchitectures/skylake_(client)

Jednostka centralna CPU: zestaw instrukcji

- Wśród aktualnie produkowanych procesorów obserwujemy dwa typy :
  - Procesory typu CISC: Compound Instruction Set Computer - komputery o złożonej liście rozkazów np. IA-32, IA-64, AMD64
  - Procesory typu RISC: Reduced Instruction Set Computer - komputery o zredukowanej liście rozkazów, np. ARM, AVR, PIC

IA - Intel Architecture

Jednostka centralna CPU: zestaw instrukcji - CISC

- Charakterystyka CISC (Complex Instruction Set Computers):
  - duża liczba instrukcji
  - mała optymalizacja – niektóre rozkazy potrzebują dużej liczby cykli procesora do wykonania
  - występowanie złożonych, specjalistycznych rozkazów
  - do pamięci może się odwoływać bezpośrednio duża liczba rozkazów
  - mniejsza od RISC-ów częstotliwość taktowania procesora
  - powolne działanie dekodera rozkazów
  - Jest to architektura zestawu instrukcji dla mikroprocesora, w którym każda instrukcja może wykonać kilka operacji niskiego poziomu, jak na przykład pobranie z pamięci, operację arytmetyczną, albo zapisanie do pamięci, a to wszystko w jednej instrukcji
  - Z reguły procesory wykonane w architekturze CISC działają wolniej niż procesory RISC

Jednostka centralna CPU: zestaw instrukcji - RISC

- Charakterystyka RISC (Reduced Instruction Set Computers):
  - zredukowana liczba rozkazów do niezbędnego minimum. Ich liczba wynosi kilkadziesiąt, podczas gdy w procesorach CISC sięga setek. Upraszcza to znacznie dekoder rozkazów
  - redukcja trybów adresowania, dzięki czemu kody rozkazów są prostsze, bardziej zunifikowane, co dodatkowo upraszcza dekoder rozkazów
  - ograniczenie komunikacji pomiędzy pamięcią, a procesorem. Do przesyłania danych pomiędzy pamięcią a rejestrami służy przede wszystkim dedykowane instrukcje, które zwykle nazywaj się load oraz store - pozostałe instrukcje mogą operować wyłącznie na rejestrach
  - Schemat działania a liczbach znajdujących się w pamięci jest następujący: załaduj dan z pamięci do rejestru, na zawartości rejestru wykonaj działanie, przepisz wynik z rejestru do pamięc
  - zwiększenie liczby rejestrów (przykładowo: 32, 192, 256, podczas gdy np. w architekturze x86 jest zaledwie 8 rejestrów), co równie ma wpływ na zmniejszenie liczby odwoła do pamięci
  - dzięki przetwarzaniu potokowemu (ang. pipelining) wszystkie rozkazy wykonuj się w jednym cyklu maszynowym, co pozwala na znaczne uproszczenie bloku wykonawczego,dodatkowo czas reakcji na przerwania jest krótszy

Jednostka centralna CPU: zestaw instrukcji - CICS vs RISC

RISC – Reduced Instruction Set Computers	CISC – Complex Instruction Set Computers
Zawierają ograniczony, prosty zbiór instrukcji.	Występują skomplikowane instrukcje wspierające języki wysokiego poziomu.
Zawierają dużą liczbę uniwersalnych rejestrów.	Zawierają małą liczbą rejestrów i/lub rejestry specjalizowane.
Instrukcje arytmetyczno-logiczne wykonywane są na rejestrach.	Instrukcje arytmetyczno-logiczne mogą pobierać argumenty z pamięci i umieszczać wynik w pamięci.
Kody instrukcji są stałej długości, typowo 4 bajty, i mają stałe rozmieszczenie pól, co ułatwia dekodowanie.	Kody instrukcji mają zmienną długość, typowo od jednego do kilkunastu bajtów. Występuje prefiksowanie instrukcji utrudniające dekodowanie.
Posiadają małą liczbę trybów adresowania.	Posiadają dużą liczbę trybów adresowania.
Dozwolone jest tylko adresowanie wyrównane.	Dozwolone jest adresowanie niewyrównane.

Jednostka centralna CPU: zestaw instrukcji - CICS vs RISC

- Przykład:

  A[i] = x-32;

CICS: x86-64 gcc 8.2

```
mov     eax, DWORD PTR [rbp-16]
cdqe
lea     rdx, [0+rax*4]
mov     rax, QWORD PTR [rbp-8]
add     rax, rdx
mov     edx, DWORD PTR [rbp-12]
sub     edx, 32
mov     DWORD PTR [rax], edx
```

RISC: RISC V x86-64 gcc 8.2

```
lw      a5,-32(s0)
slli    a5,a5,2
ld      a4,-24(s0)
add     a5,a4,a5
lw      a4,-28(s0)
addiw   a4,a4,-32
sext.w  a4,a4
sw      a4,0(a5)
```

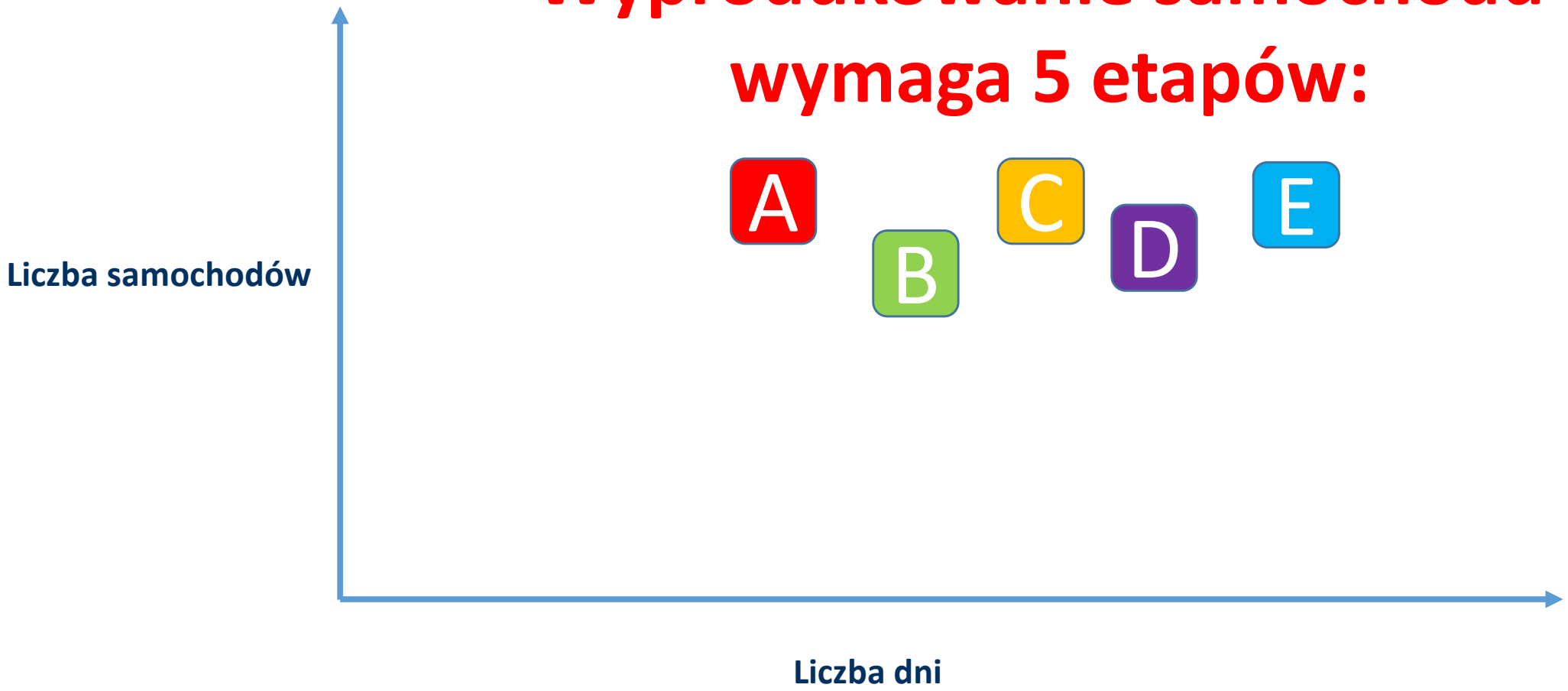
Jednostka centralna CPU: przetwarzanie potokowe

- Potokowość (ang. pipelining) jest techniką budowy procesorów polegającej na podziale logiki procesora odpowiedzialnej za proces wykonywania programu (instrukcji procesora) na specjalizowane grupy w taki sposób, aby każda z grup wykonywała część pracy związanej z wykonaniem rozkazu
- Zasadnicza idea przetwarzania potokowego polega na rozłożeniu wykonania pojedynczej instrukcji na ciąg prostych etapów, z których każdy może być wykonany w jednym cyklu zegara
- Grupy/etapy są połączone sekwencyjnie i wykonują pracę równocześnie, pobierając dane od poprzedniego elementu w sekwencji
- W każdej z tych grup/etapów rozkaz jest na innym stadium wykonania
- Wykonywanie instrukcji przypomina taśmę produkcyjną
- W każdym cyklu zegara instrukcje przemieszczają się wzdłuż potoku do następnego etapu

Jednostka centralna CPU: przetwarzanie potokowe - koncepcja

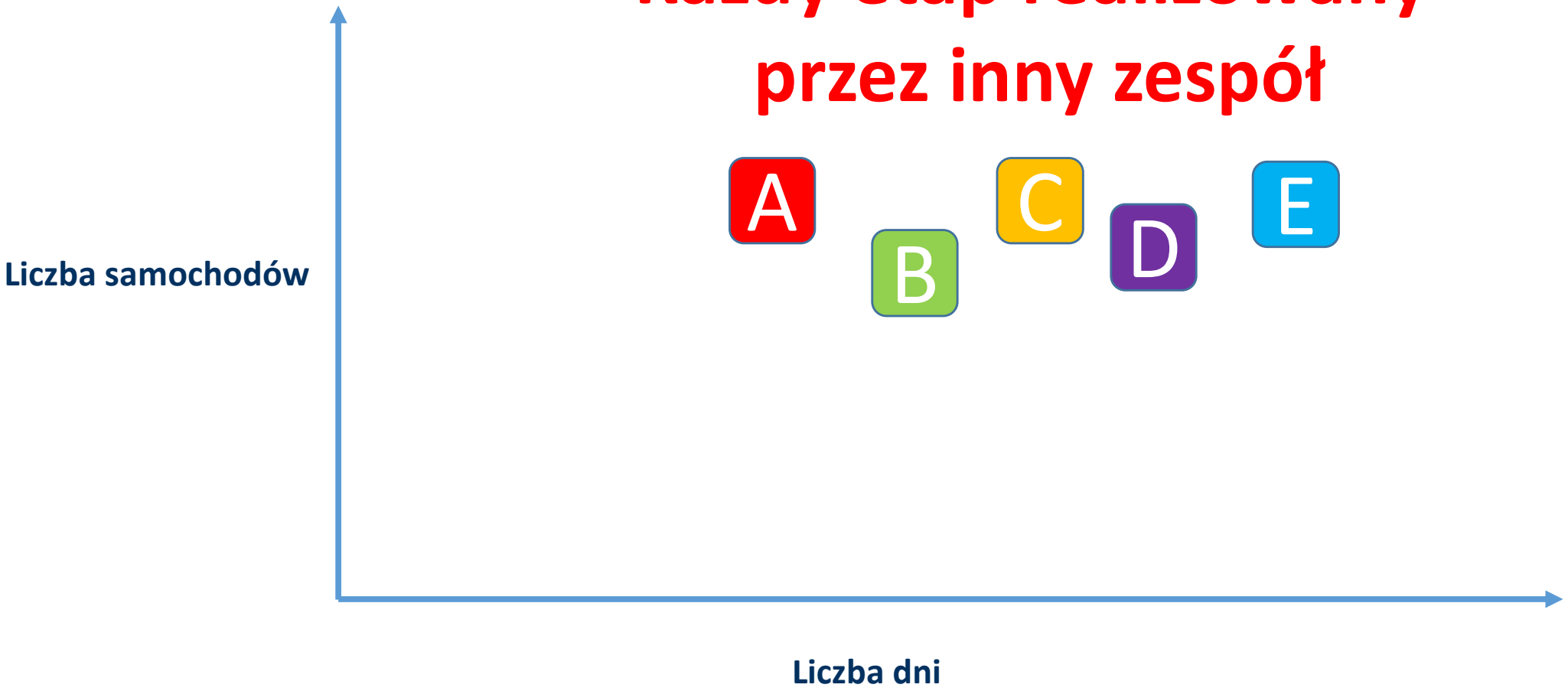
Jednostka centralna CPU: przetwarzanie potokowe - koncepcja

Wyprodukowanie samochodu wymaga 5 etapów:



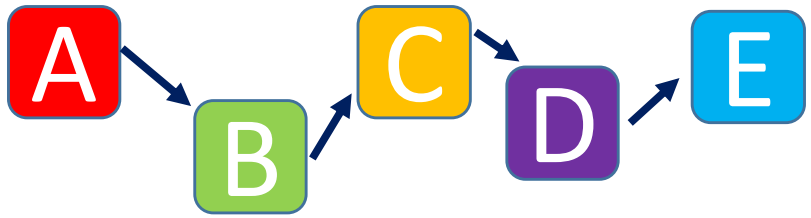
Jednostka centralna CPU: przetwarzanie potokowe - koncepcja

Każdy etap realizowany przez inny zespół

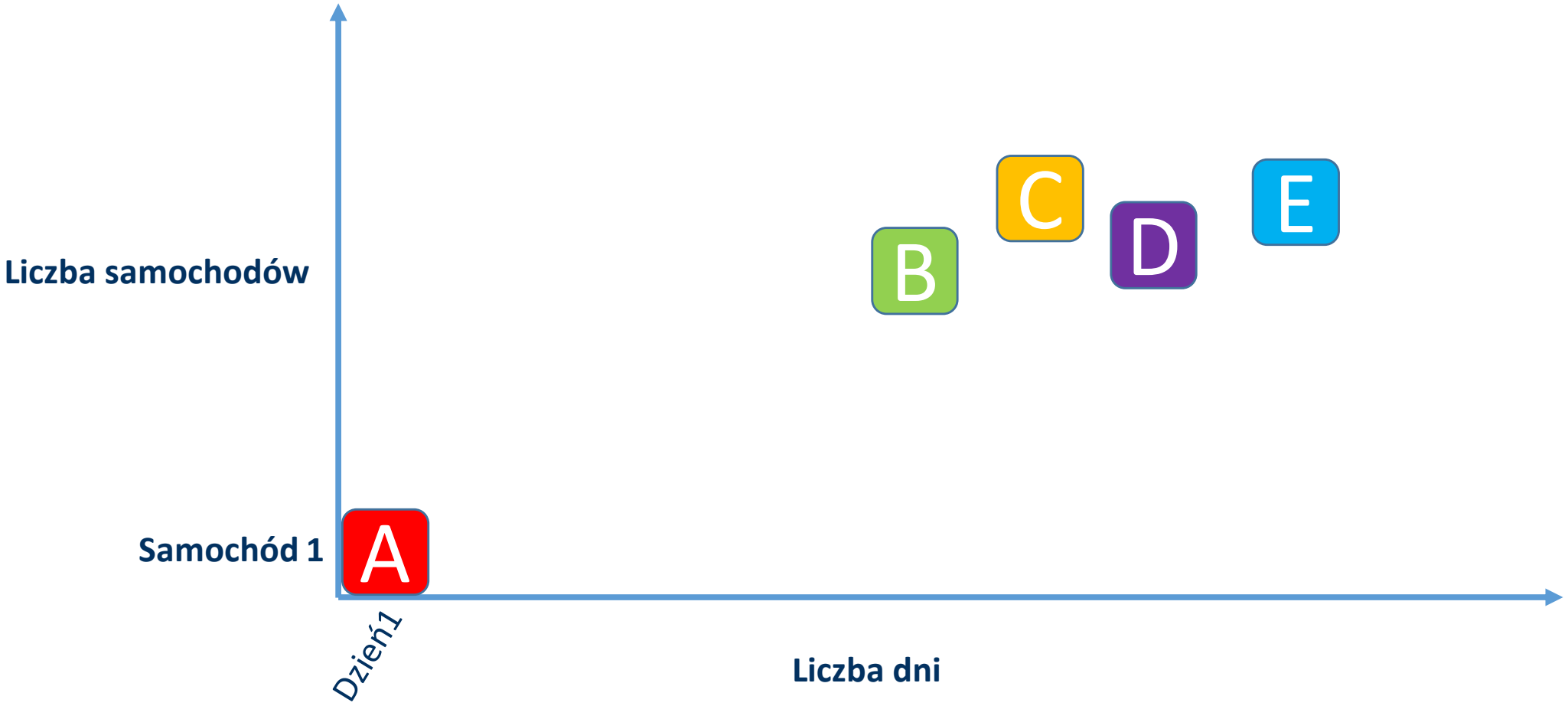


Jednostka centralna CPU: przetwarzanie potokowe - koncepcja

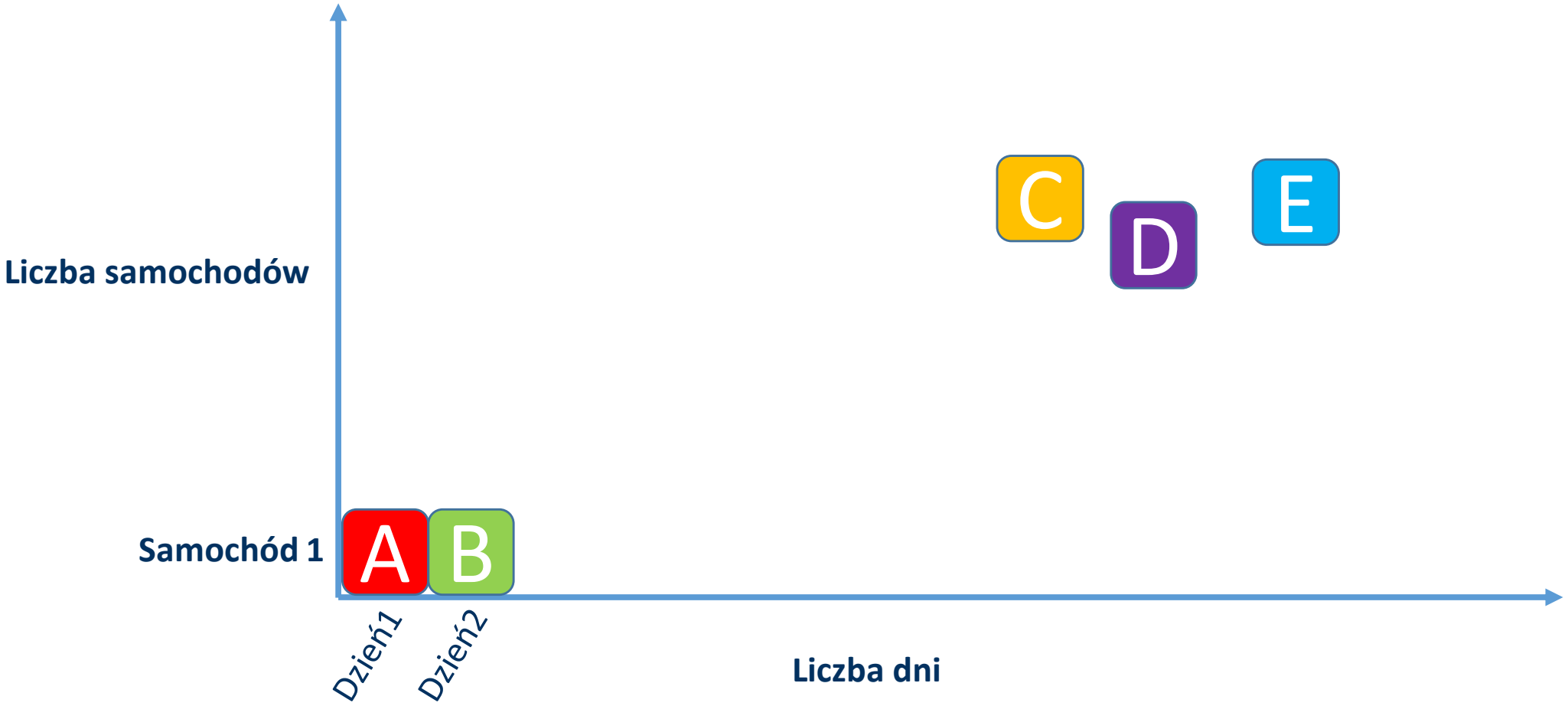
Realizacje danego etapu zależy od poprzedniego



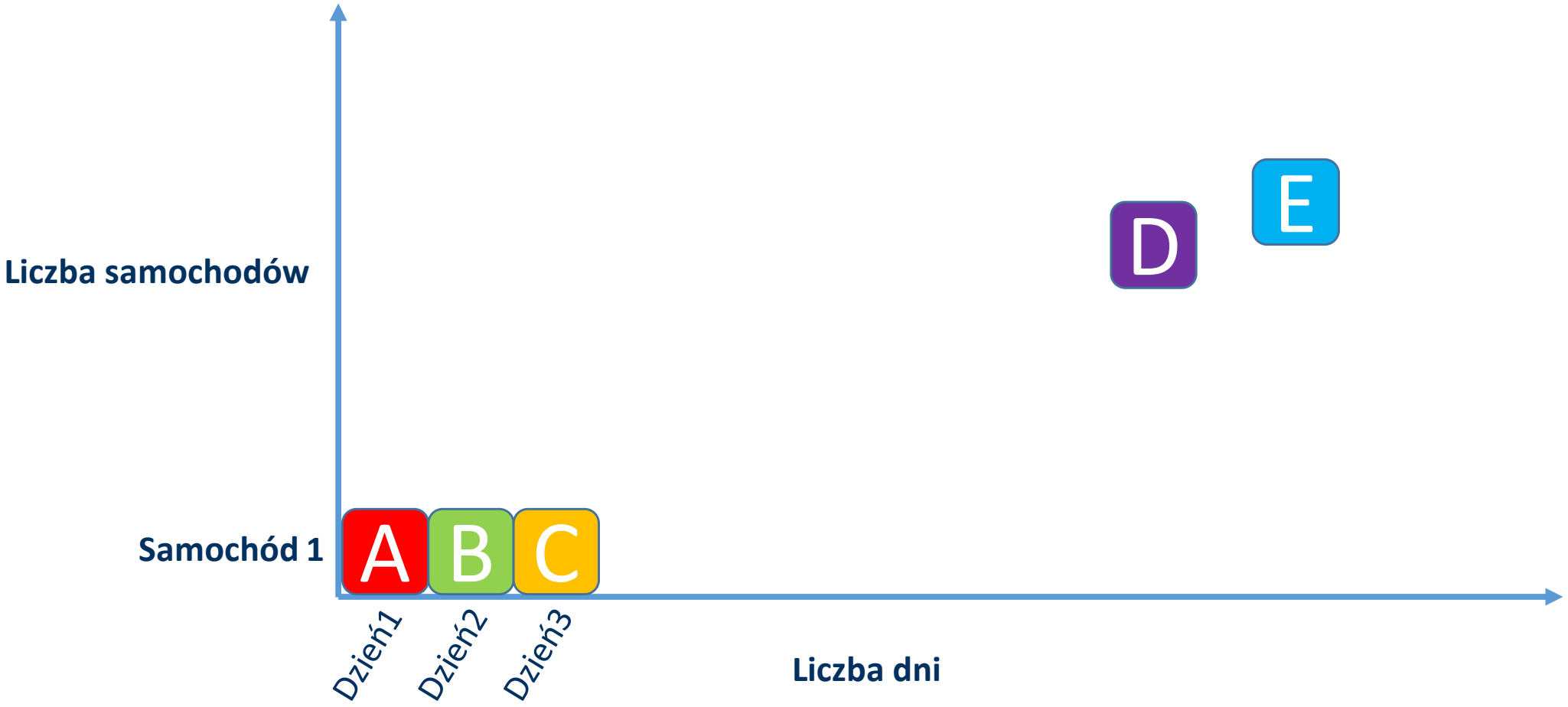
Jednostka centralna CPU: przetwarzanie potokowe - koncepcja



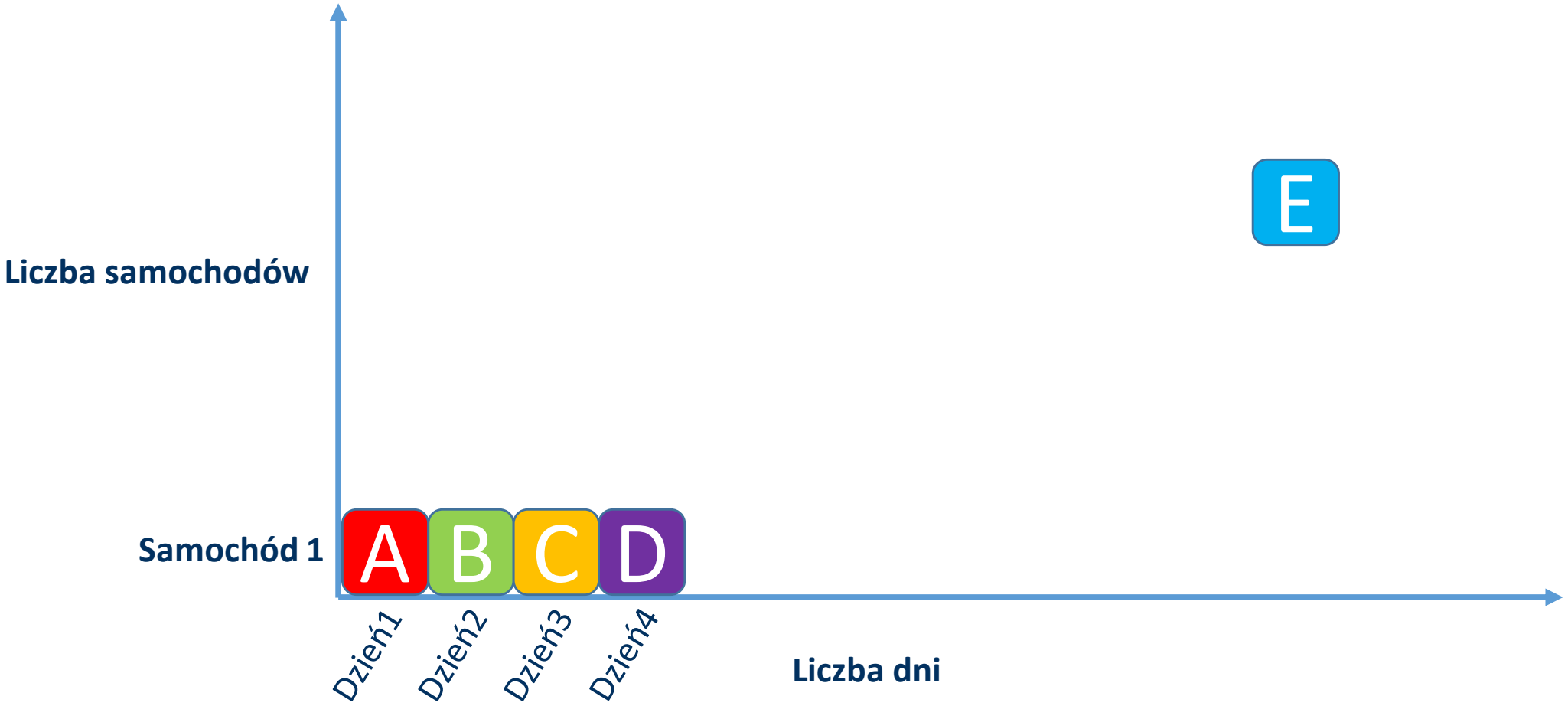
Jednostka centralna CPU: przetwarzanie potokowe - koncepcja



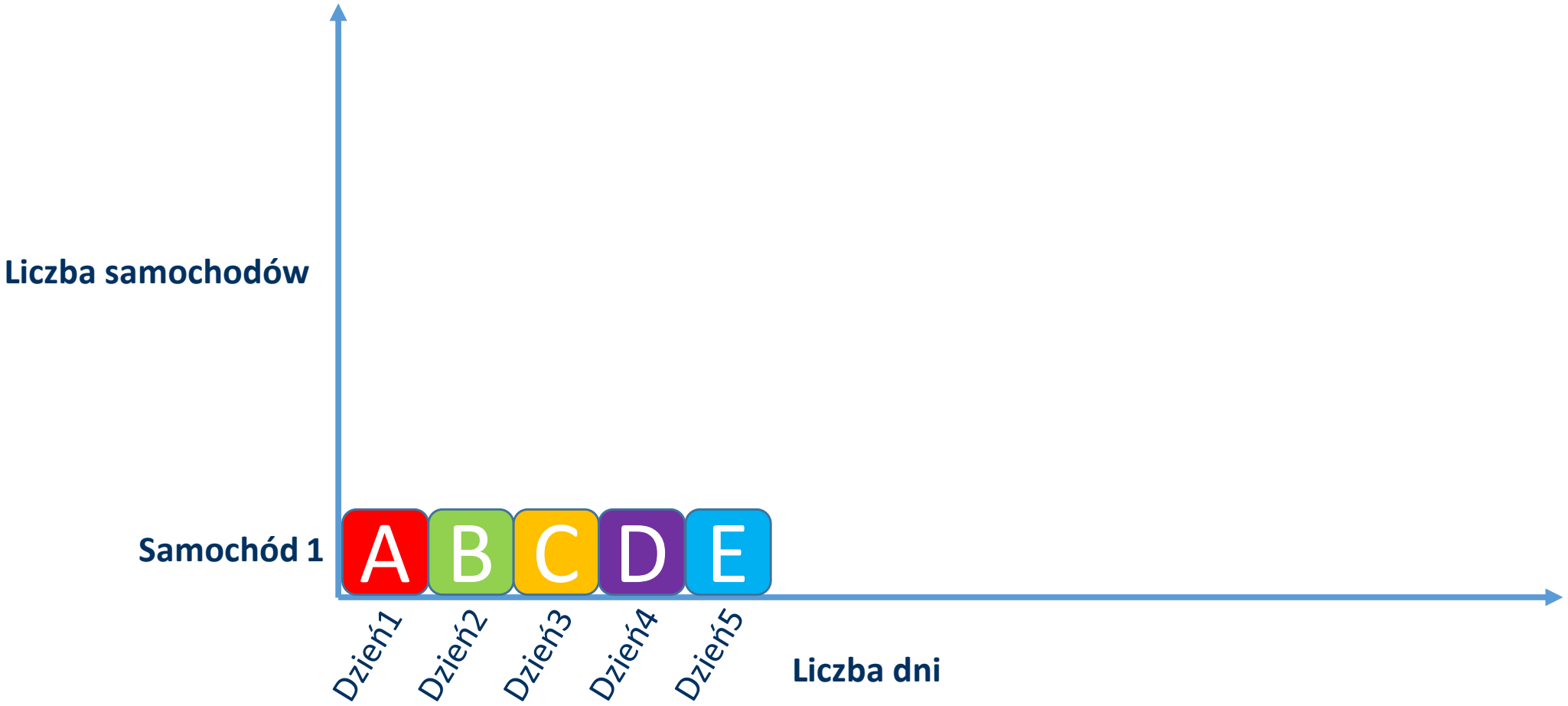
Jednostka centralna CPU: przetwarzanie potokowe - koncepcja



Jednostka centralna CPU: przetwarzanie potokowe - koncepcja



Jednostka centralna CPU: przetwarzanie potokowe - koncepcja



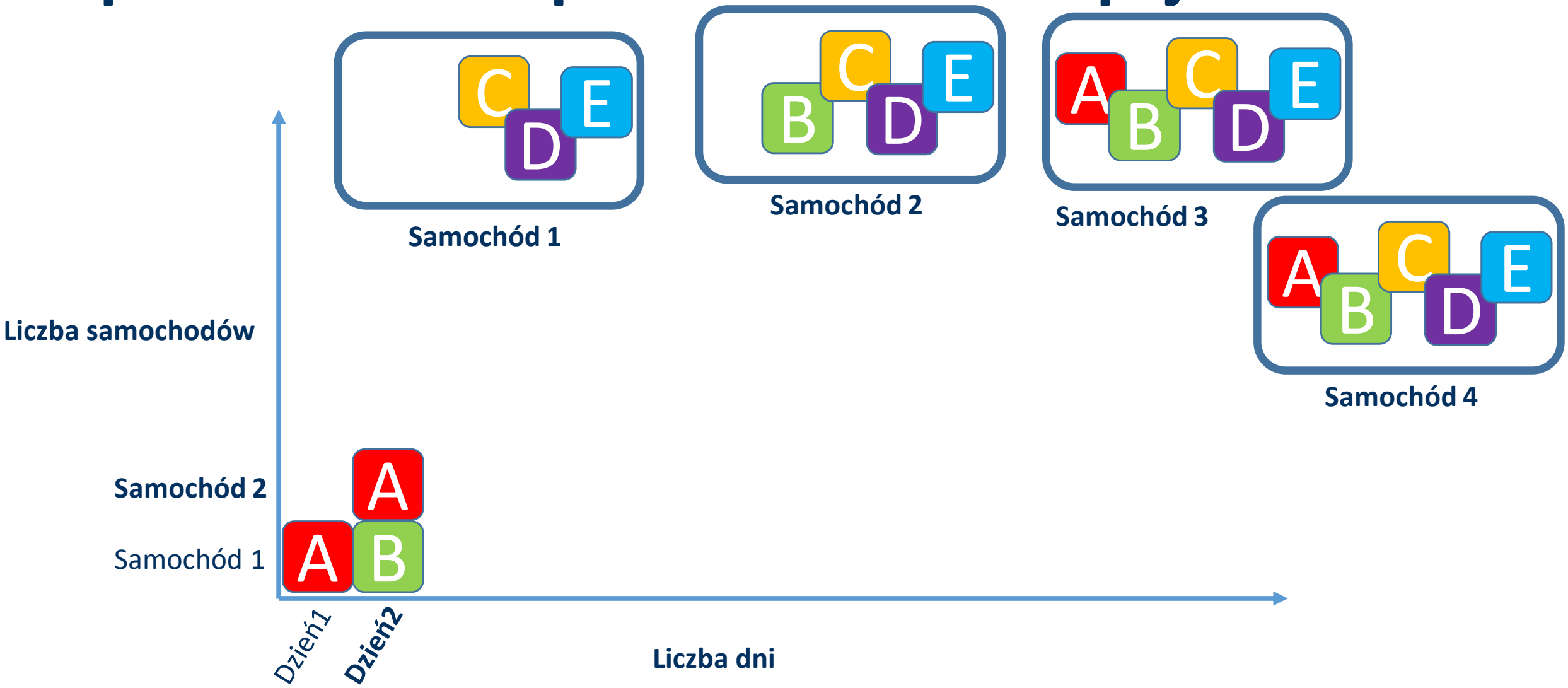
Jednostka centralna CPU: przetwarzanie potokowe - koncepcja

5 dni trwa wyprodukowanie jednego samochodu

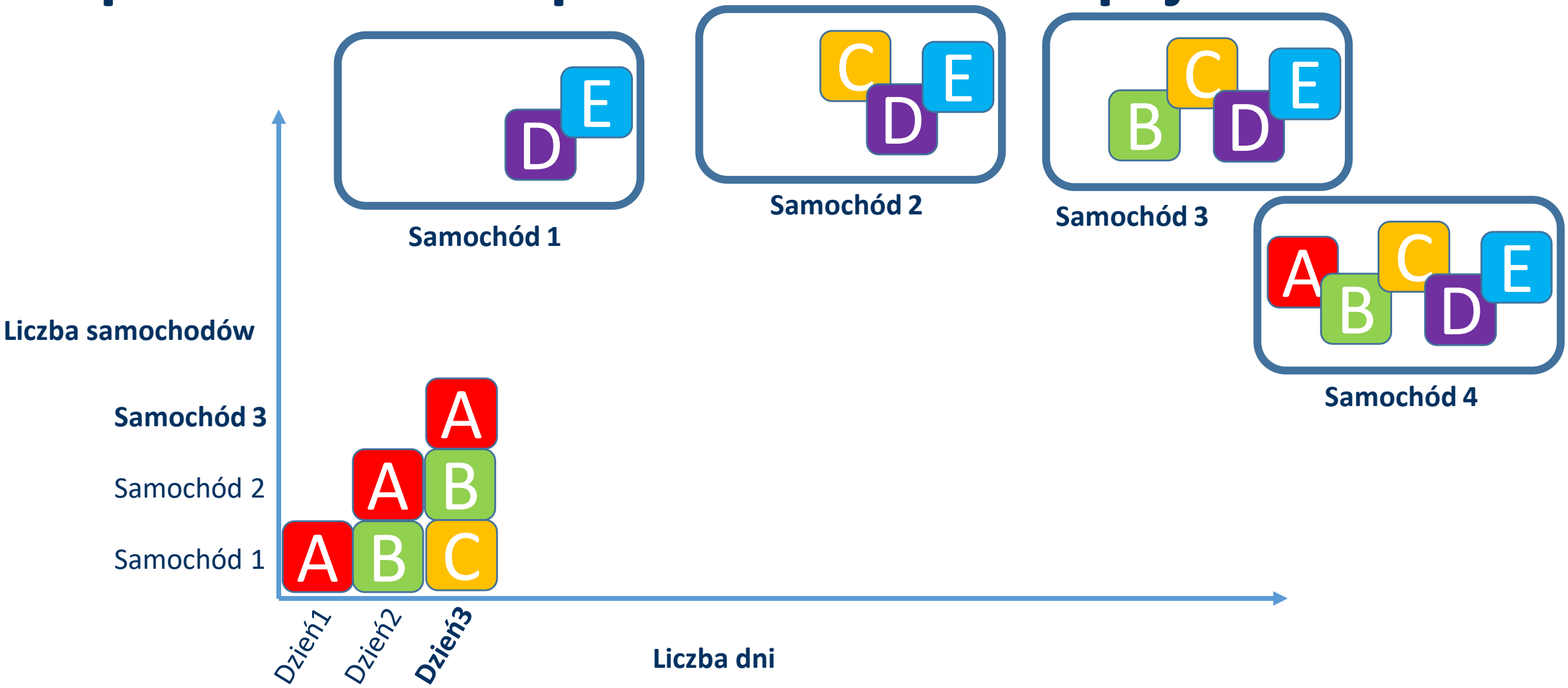
Jednostka centralna CPU: przetwarzanie potokowe - koncepcja

Jednostka centralna CPU: przetwarzanie potokowe - koncepcja

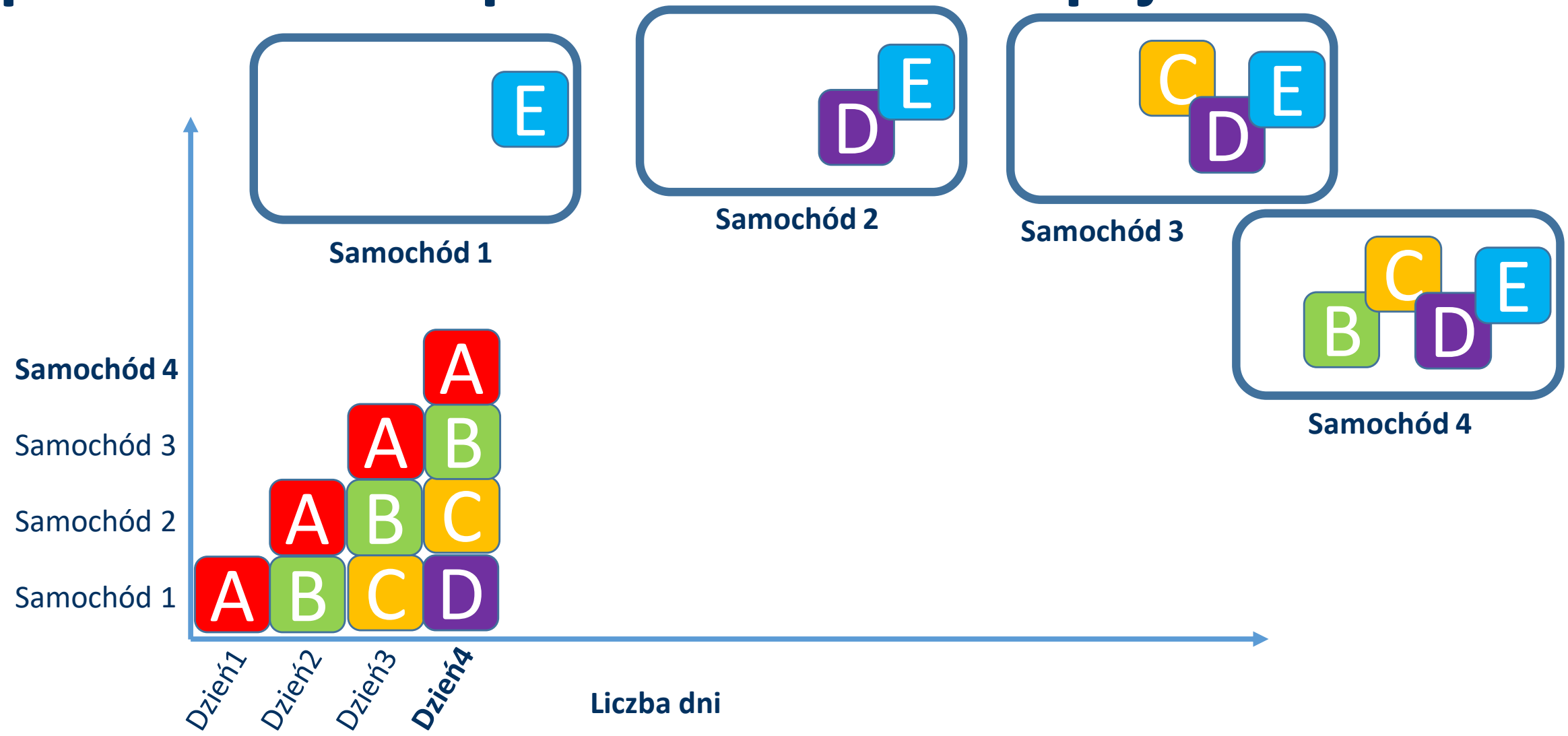
Jednostka centralna CPU: przetwarzanie potokowe - koncepcja



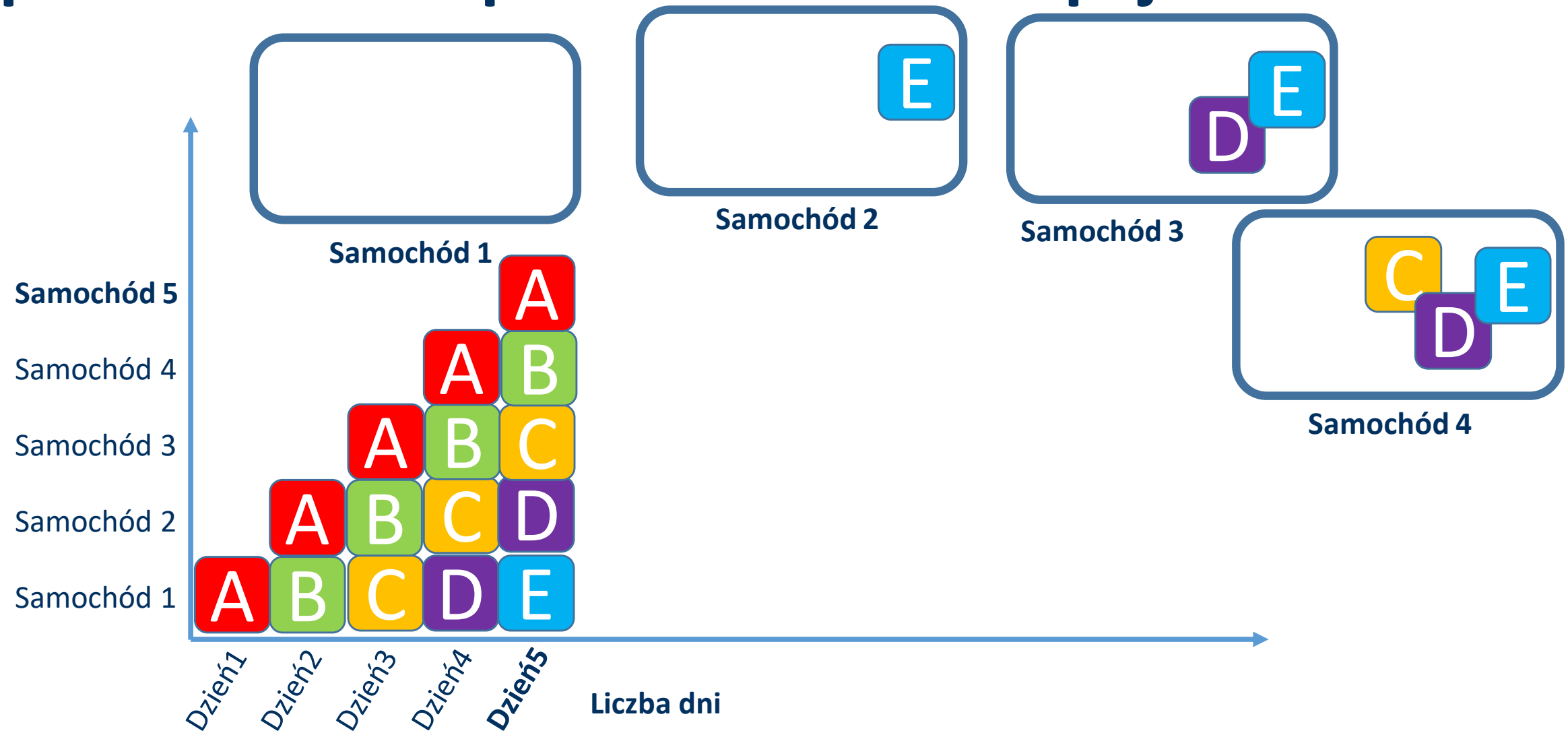
Jednostka centralna CPU: przetwarzanie potokowe - koncepcja



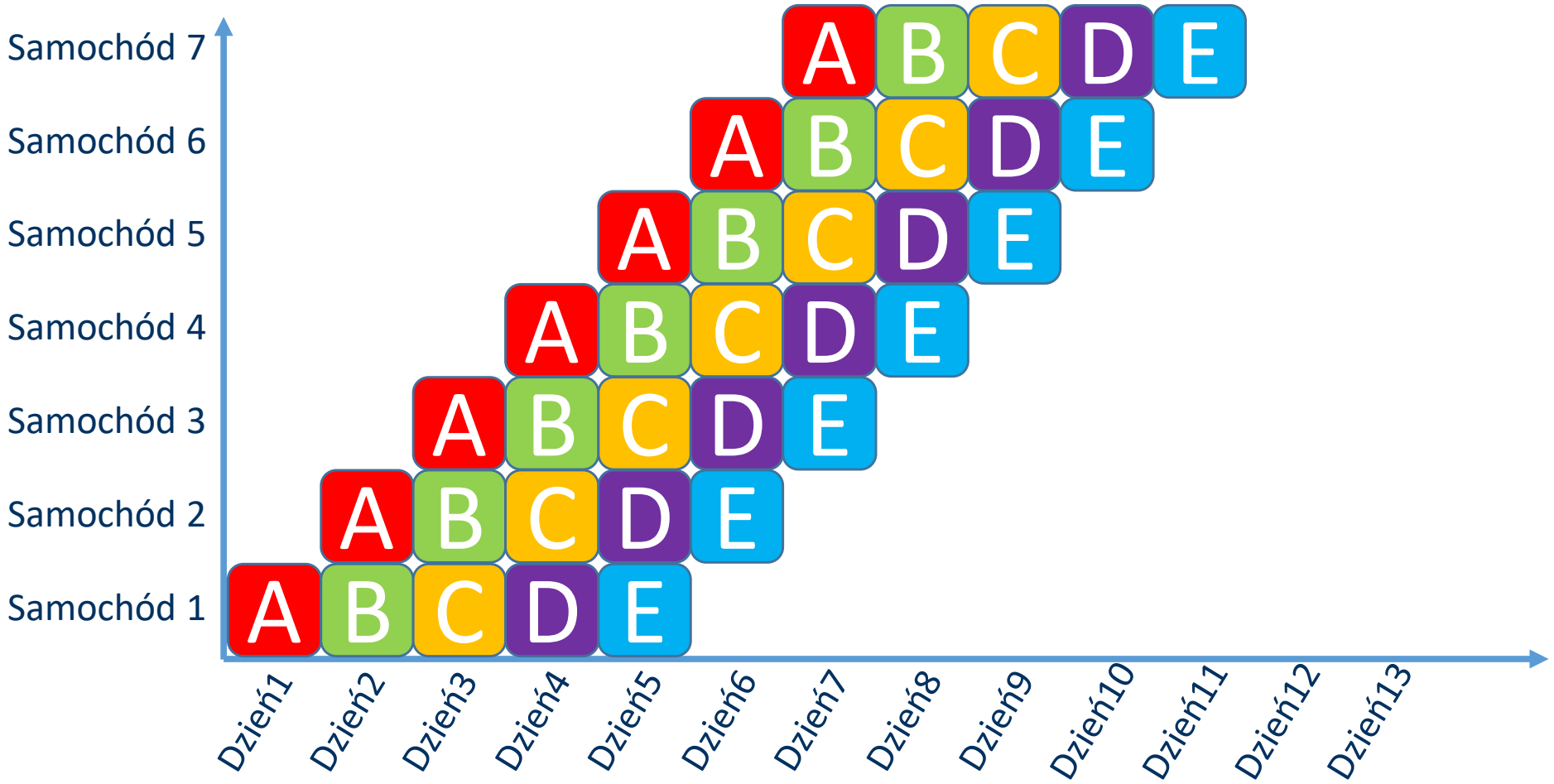
Jednostka centralna CPU: przetwarzanie potokowe - koncepcja



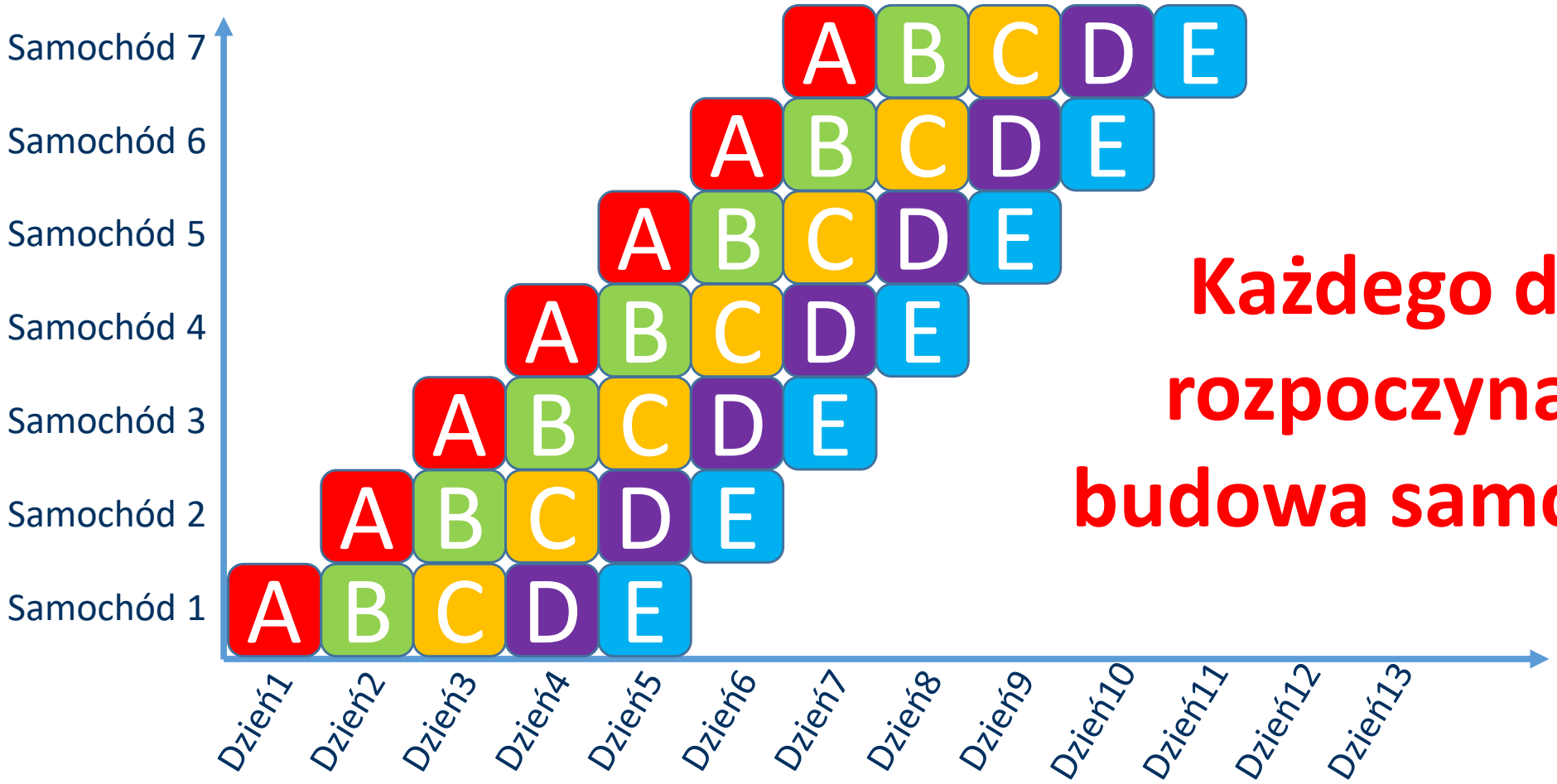
Jednostka centralna CPU: przetwarzanie potokowe - koncepcja



Jednostka centralna CPU: przetwarzanie potokowe - koncepcja

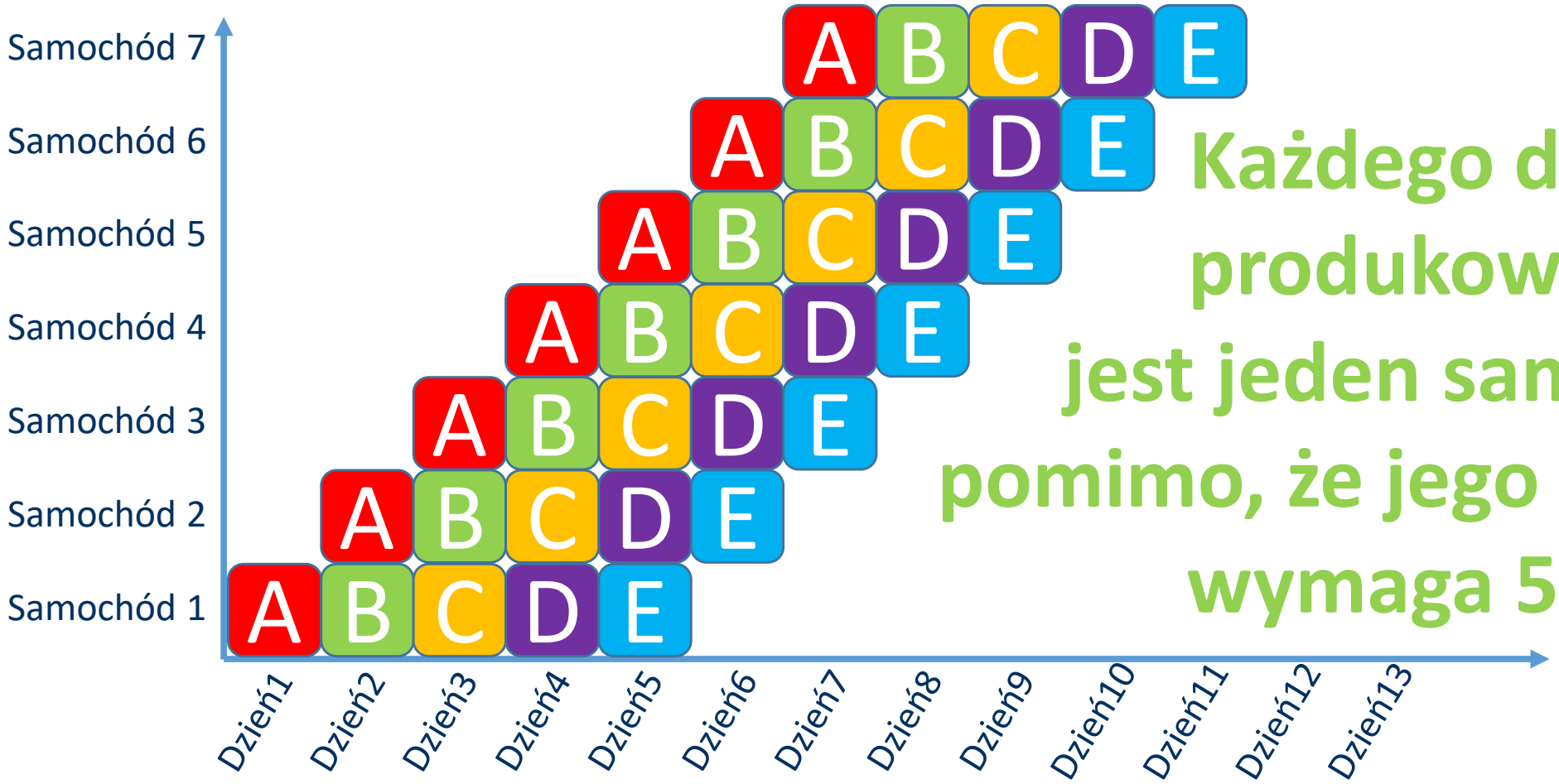


Jednostka centralna CPU: przetwarzanie potokowe - koncepcja



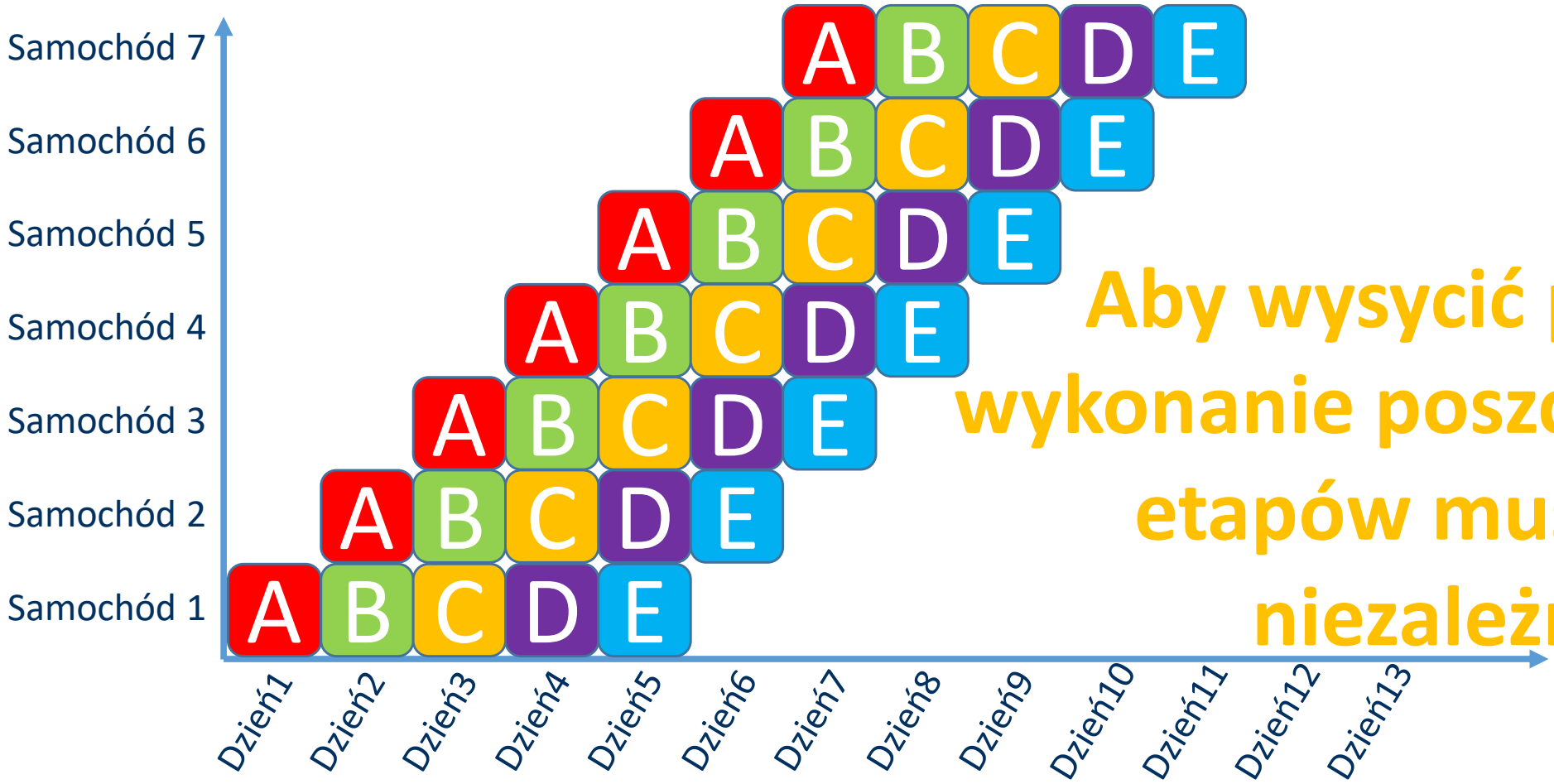
Każdego dnia rozpoczyna się budowa samochody

Jednostka centralna CPU: przetwarzanie potokowe - koncepcja



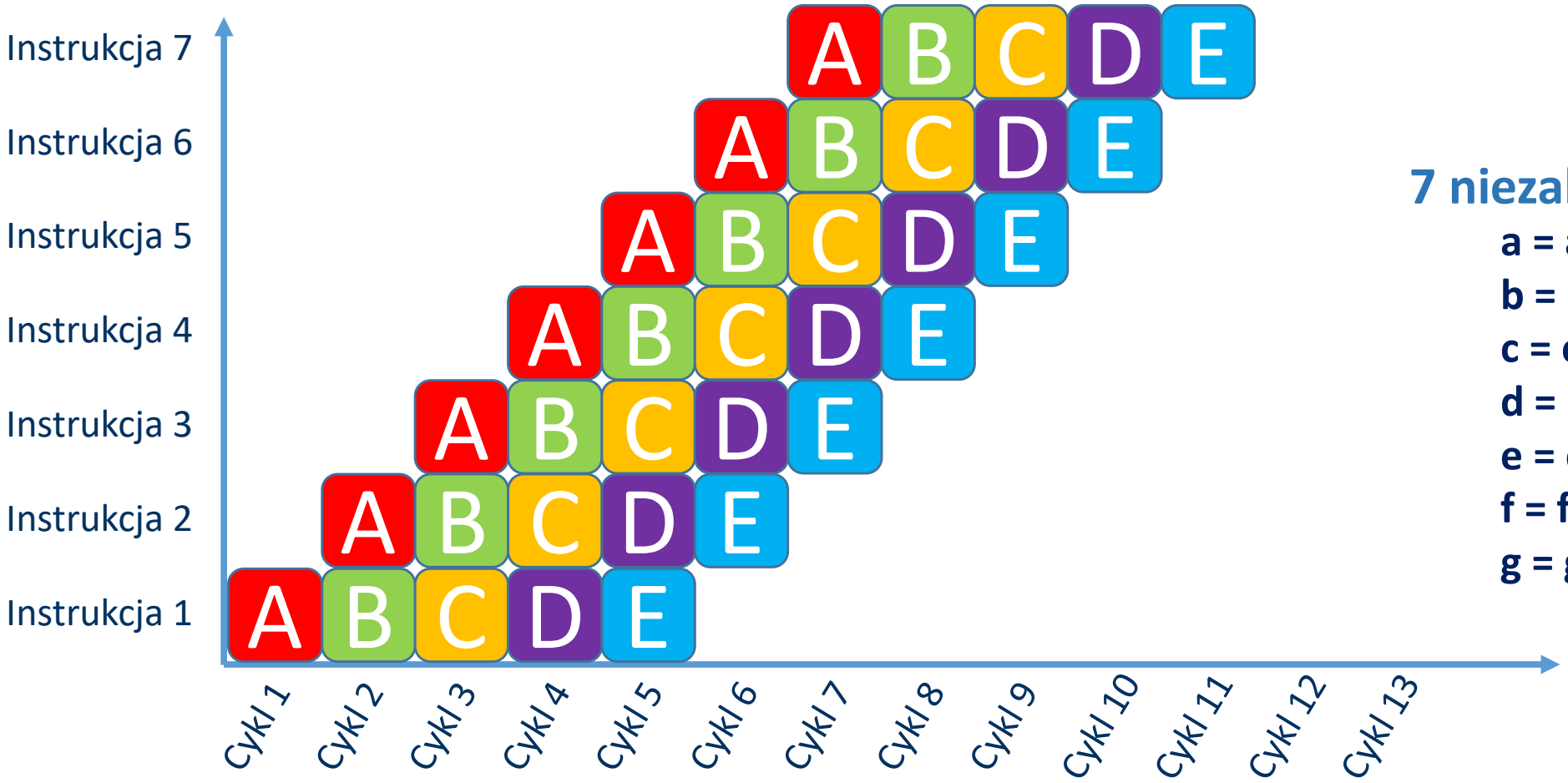
Każdego dnia produkowany jest jeden samochód pomimo, że jego produkcja wymaga 5 dni

Jednostka centralna CPU: przetwarzanie potokowe - koncepcja



Aby wysycić potok, wykonanie poszczególnych etapów musi być niezależne

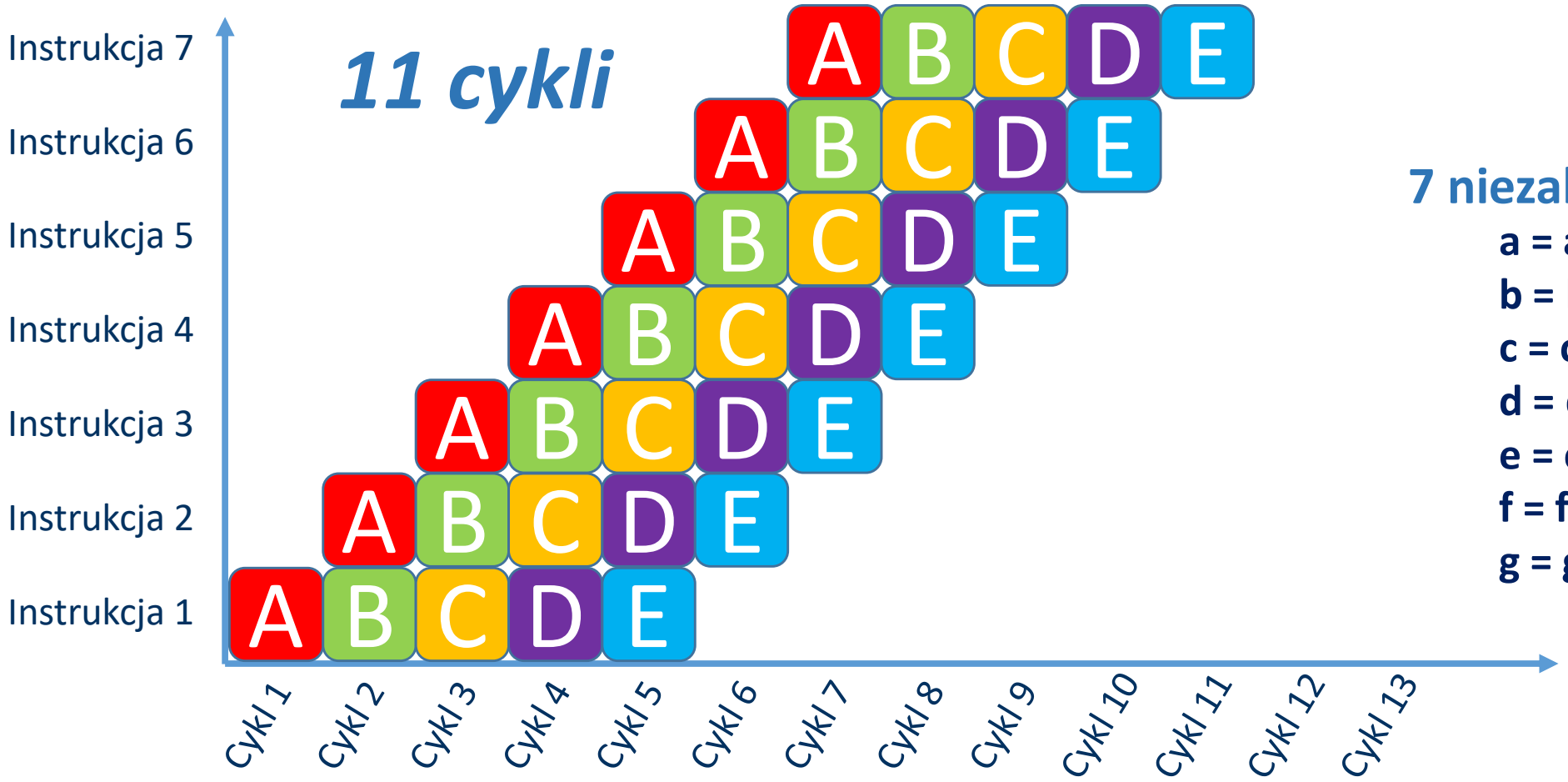
Jednostka centralna CPU: przetwarzanie potokowe - koncepcja



7 niezależnych instrukcji

a = a + 9 - instrukcja 1
b = b + 9 - instrukcja 2
c = c + 9 - instrukcja 3
d = d + 9 - instrukcja 4
e = e + 9 - instrukcja 5
f = f + 9 - instrukcja 6
g = g + 9 - instrukcja 7

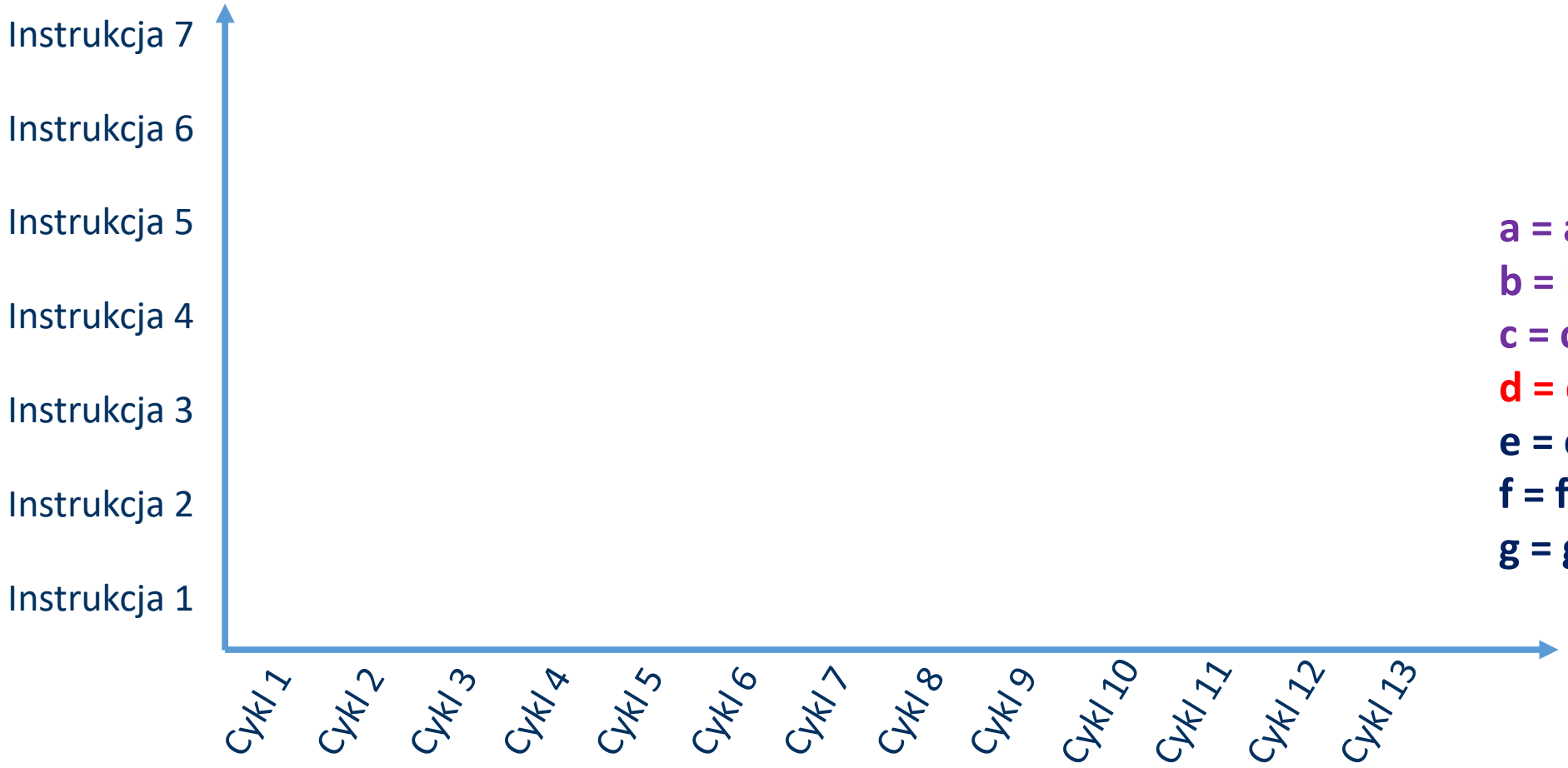
Jednostka centralna CPU: przetwarzanie potokowe - koncepcja



7 niezależnych instrukcji

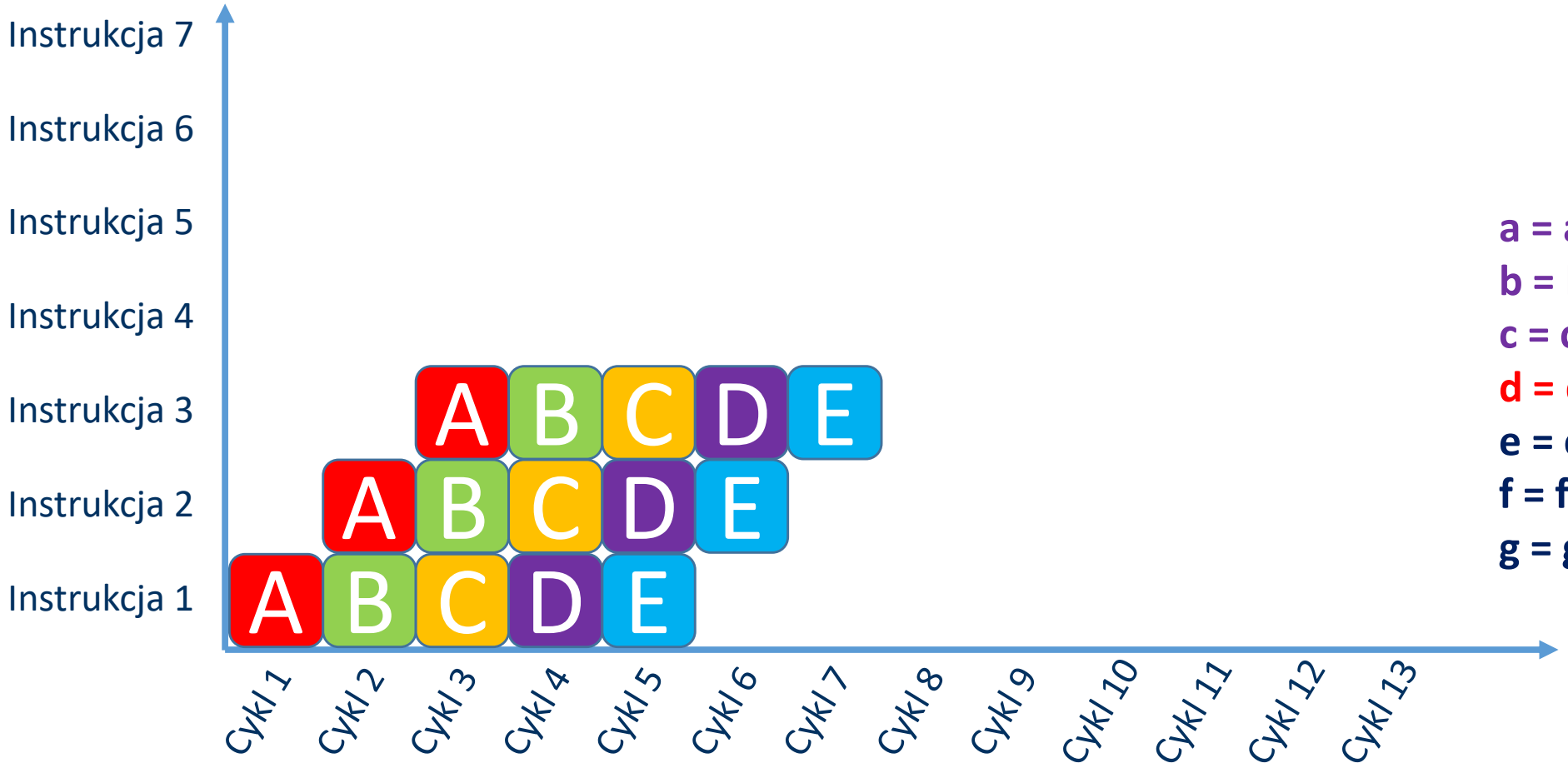
a = a + 9 - instrukcja 1
b = b + 9 - instrukcja 2
c = c + 9 - instrukcja 3
d = d + 9 - instrukcja 4
e = e + 9 - instrukcja 5
f = f + 9 - instrukcja 6
g = g + 9 - instrukcja 7

Jednostka centralna CPU: przetwarzanie potokowe - koncepcja



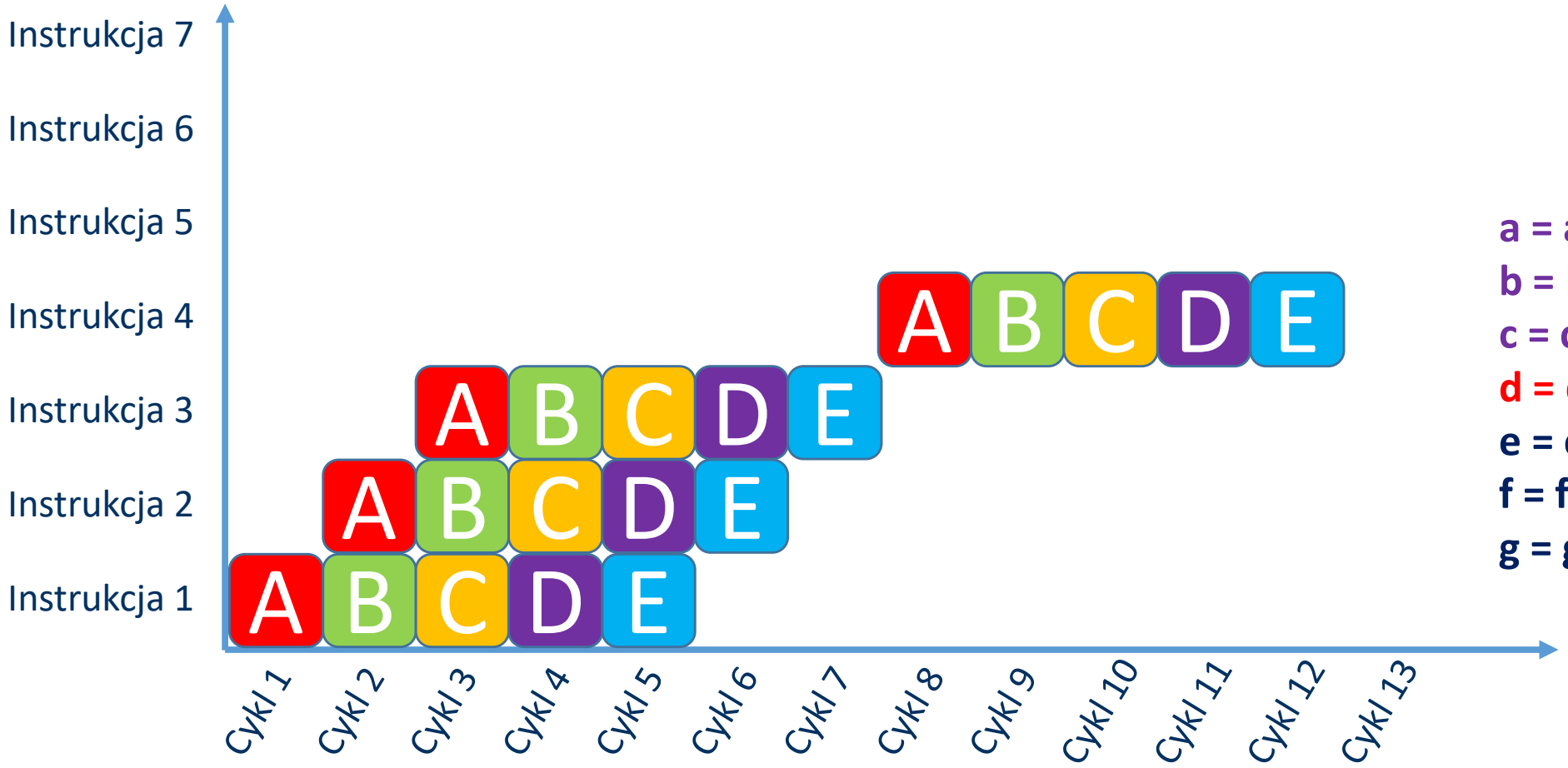
a = a + 9 - instrukcja 1
b = b + 9 - instrukcja 2
c = c + 9 - instrukcja 3
d = d + c - instrukcja 4
e = e + 9 - instrukcja 5
f = f + 9 - instrukcja 6
g = g + 9 - instrukcja 7

Jednostka centralna CPU: przetwarzanie potokowe - koncepcja



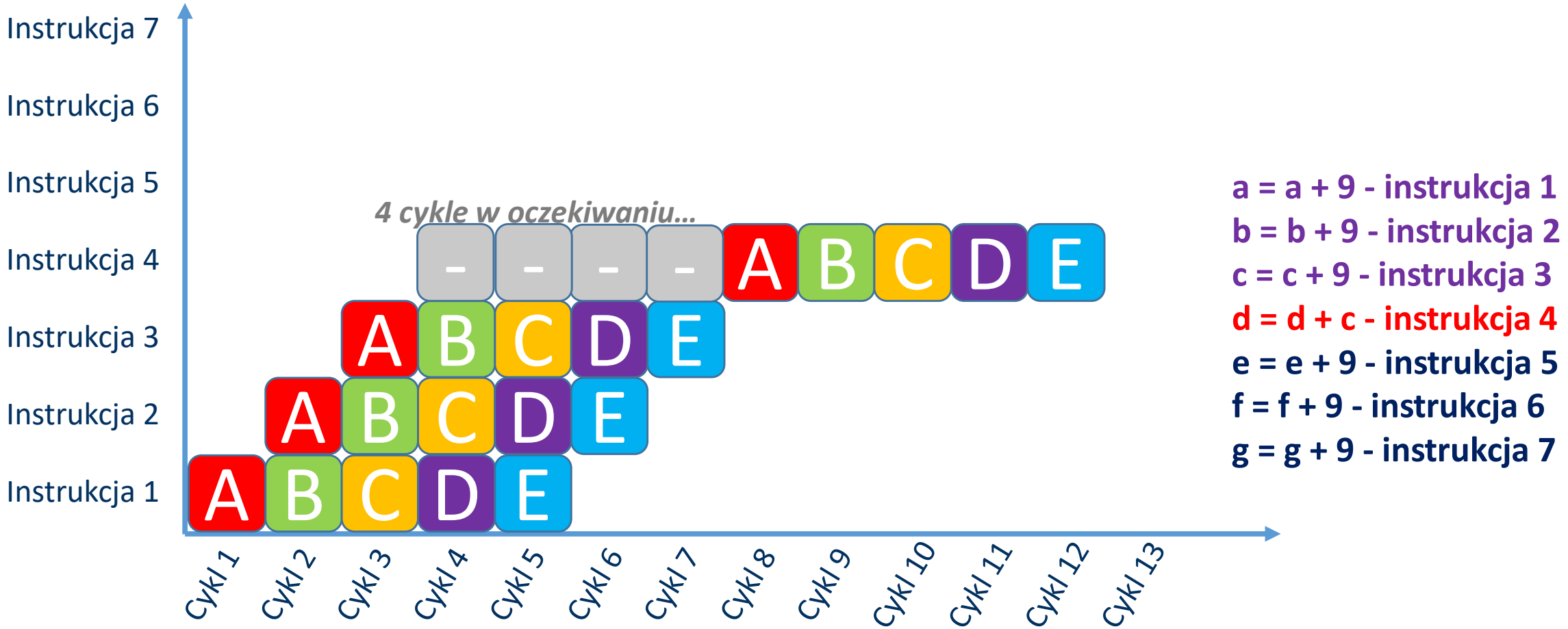
a = a + 9 - instrukcja 1
b = b + 9 - instrukcja 2
c = c + 9 - instrukcja 3
d = d + c - instrukcja 4
e = e + 9 - instrukcja 5
f = f + 9 - instrukcja 6
g = g + 9 - instrukcja 7

Jednostka centralna CPU: przetwarzanie potokowe - koncepcja



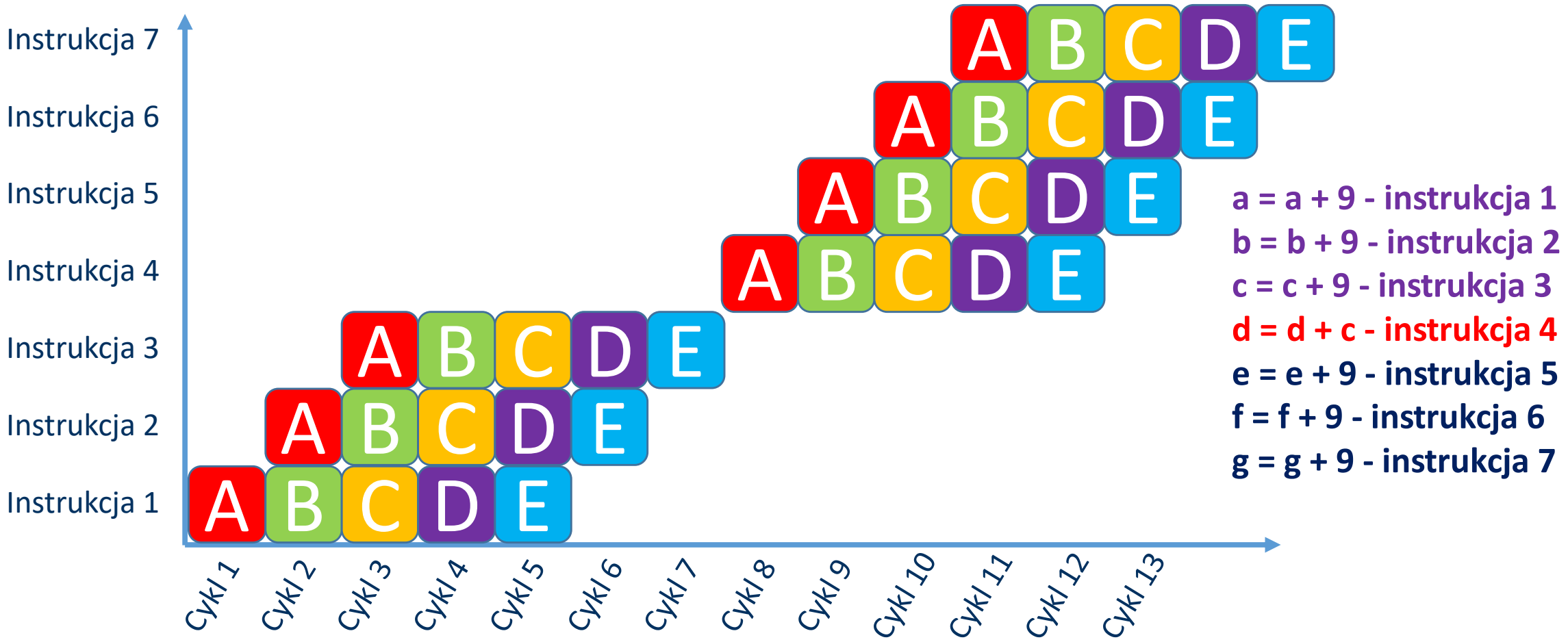
a = a + 9 - instrukcja 1
b = b + 9 - instrukcja 2
c = c + 9 - instrukcja 3
d = d + c - instrukcja 4
e = e + 9 - instrukcja 5
f = f + 9 - instrukcja 6
g = g + 9 - instrukcja 7

Jednostka centralna CPU: przetwarzanie potokowe - koncepcja

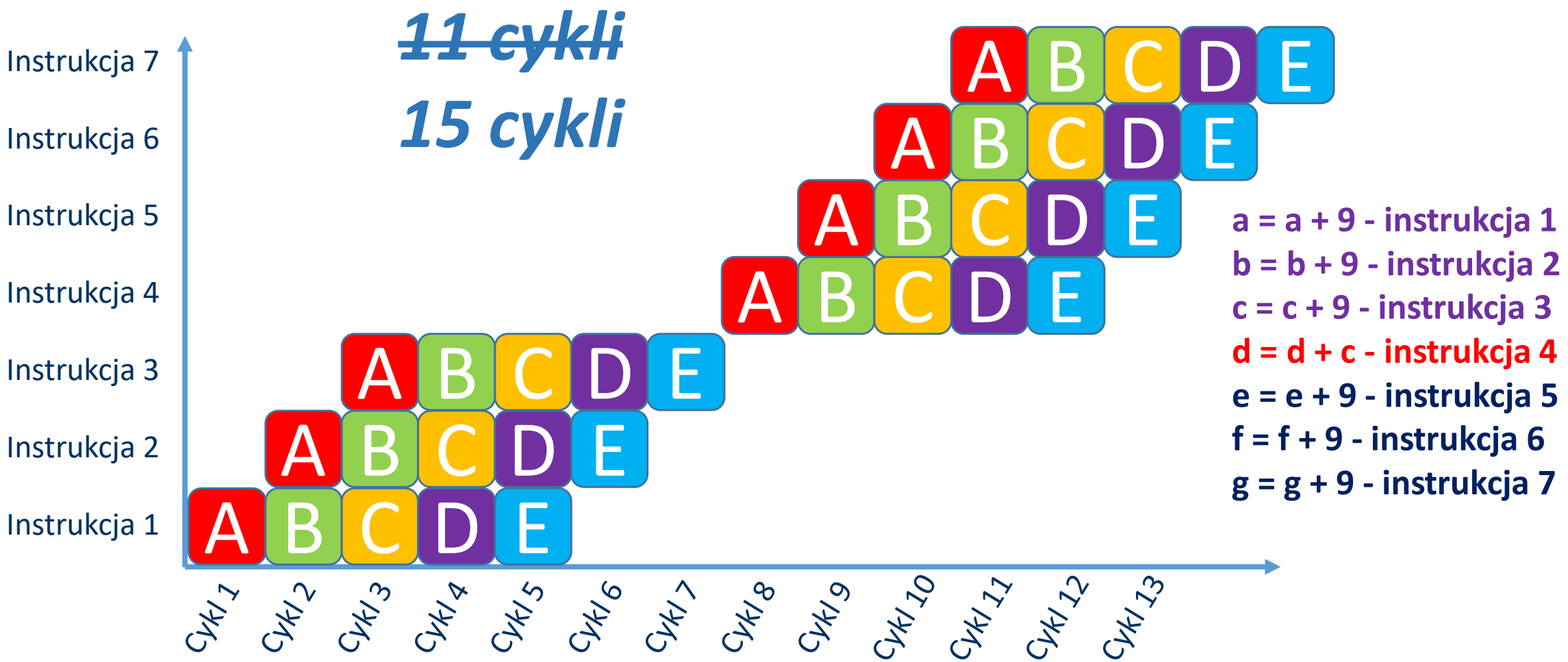


a = a + 9 - instrukcja 1
b = b + 9 - instrukcja 2
c = c + 9 - instrukcja 3
d = d + c - instrukcja 4
e = e + 9 - instrukcja 5
f = f + 9 - instrukcja 6
g = g + 9 - instrukcja 7

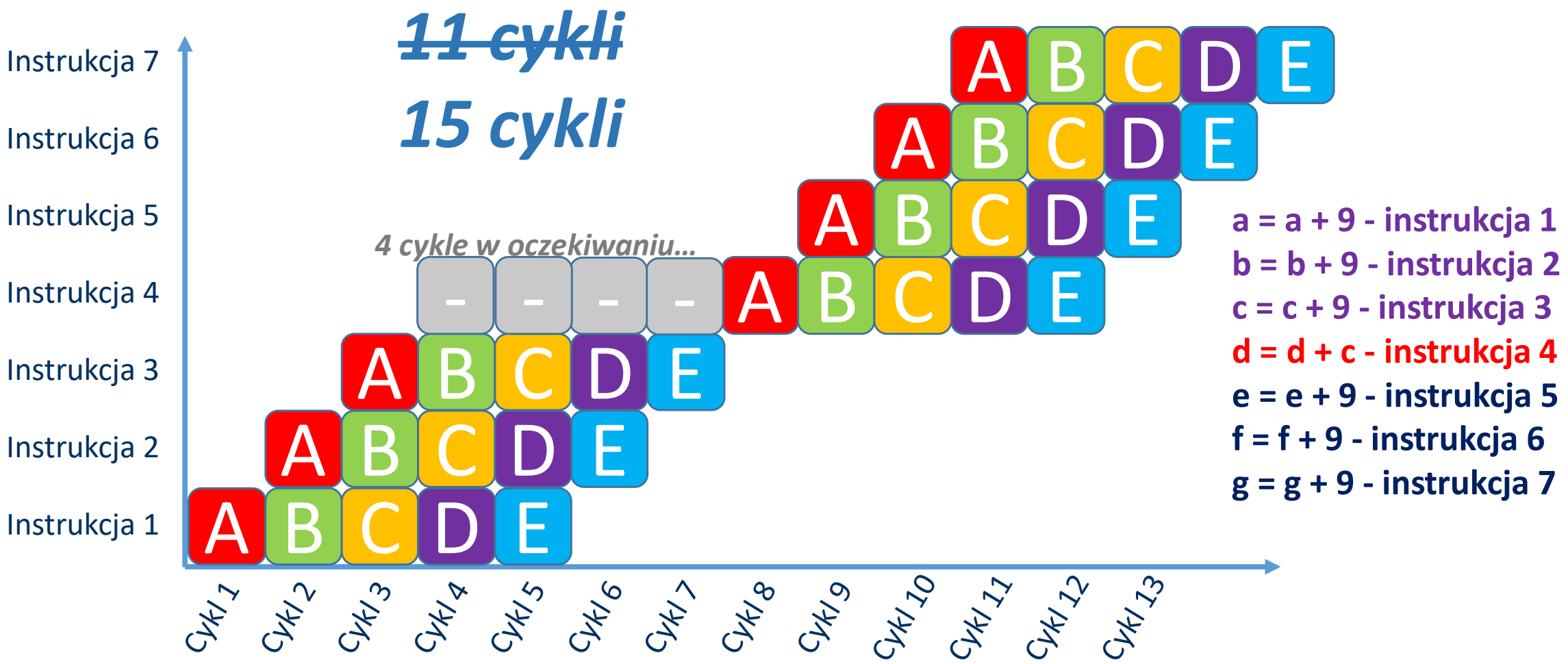
Jednostka centralna CPU: przetwarzanie potokowe - koncepcja



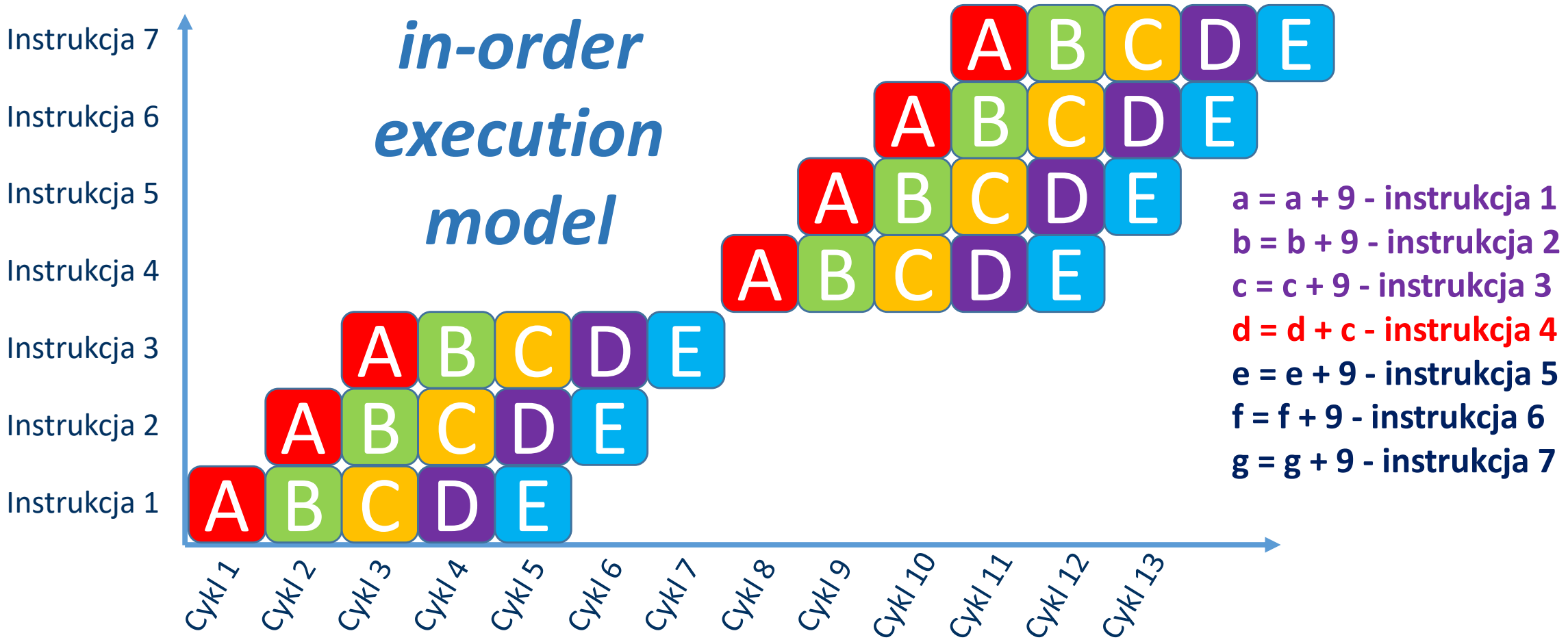
Jednostka centralna CPU: przetwarzanie potokowe - koncepcja



Jednostka centralna CPU: przetwarzanie potokowe - koncepcja

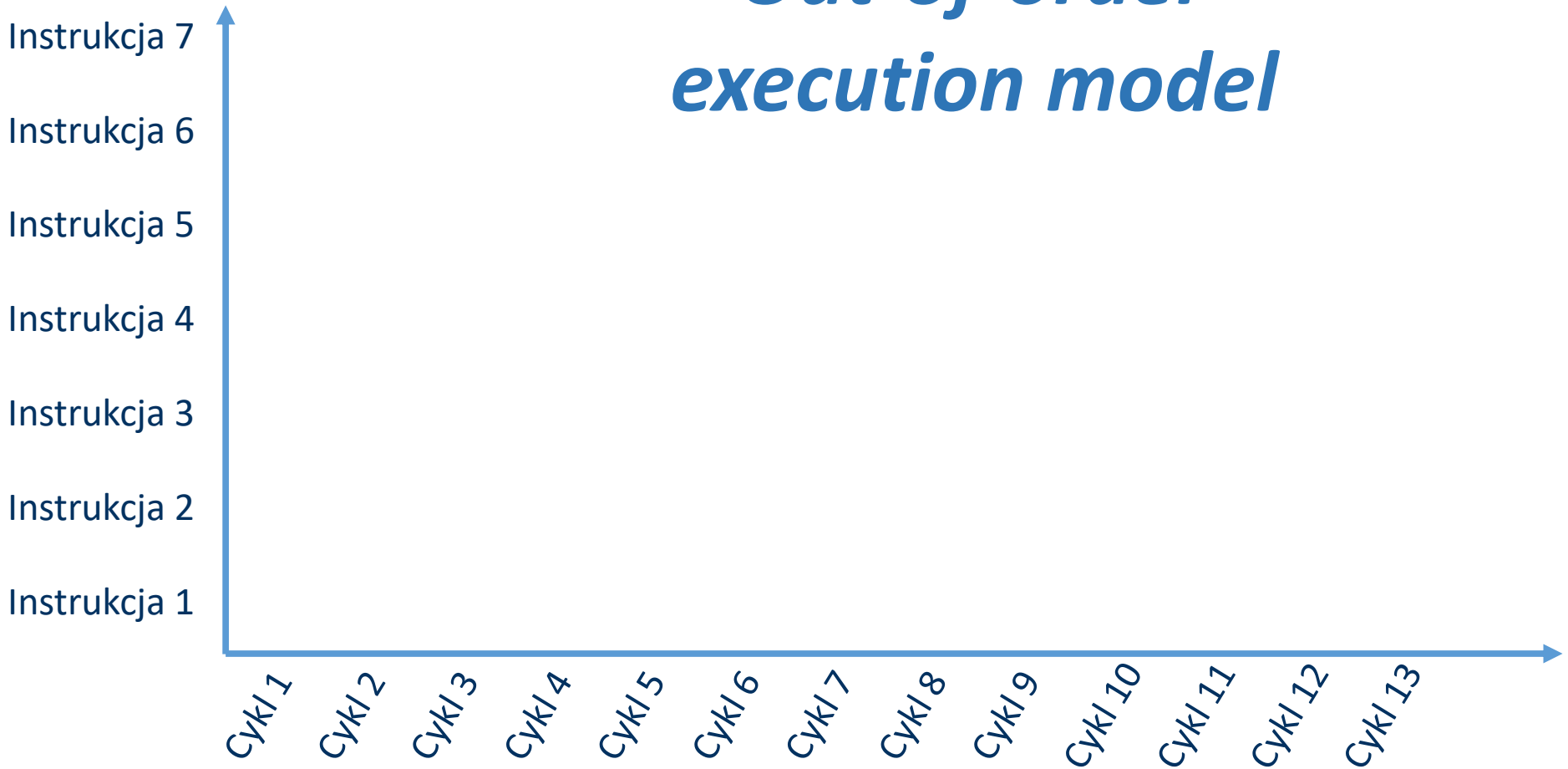


Jednostka centralna CPU: przetwarzanie potokowe - koncepcja



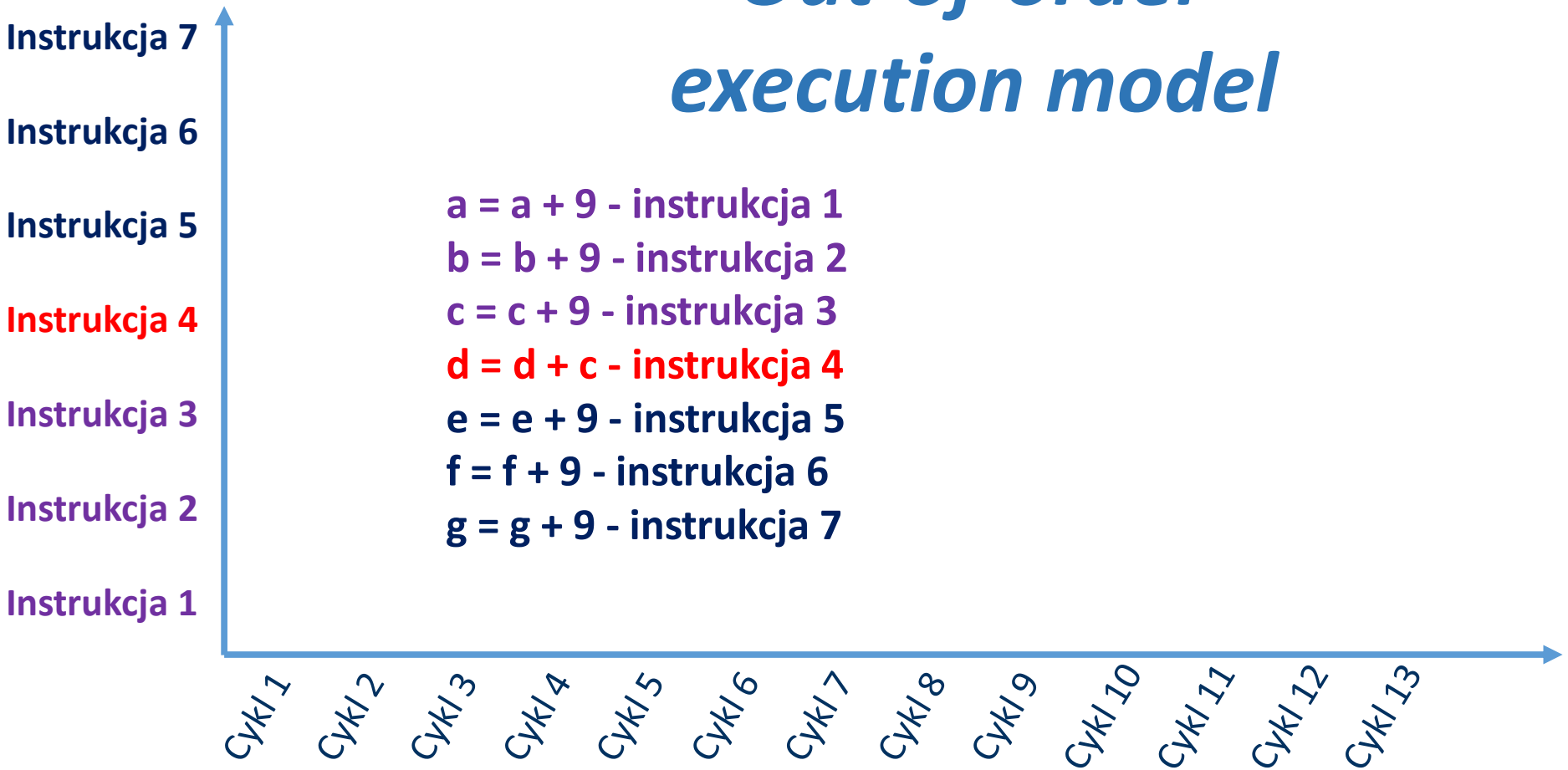
Jednostka centralna CPU: przetwarzanie potokowe - koncepcja

Out-of-order execution model



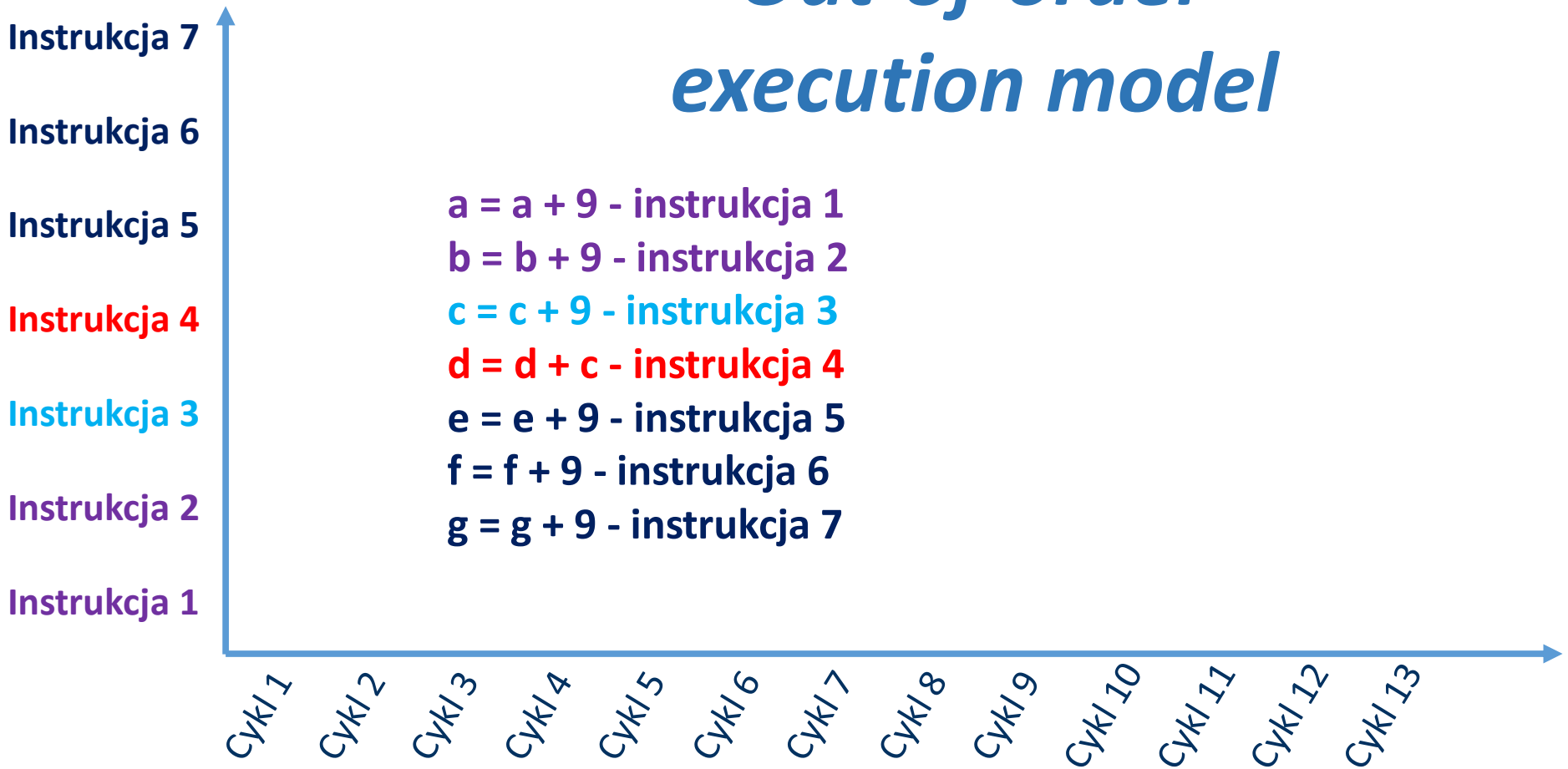
Jednostka centralna CPU: przetwarzanie potokowe - koncepcja

Out-of-order execution model

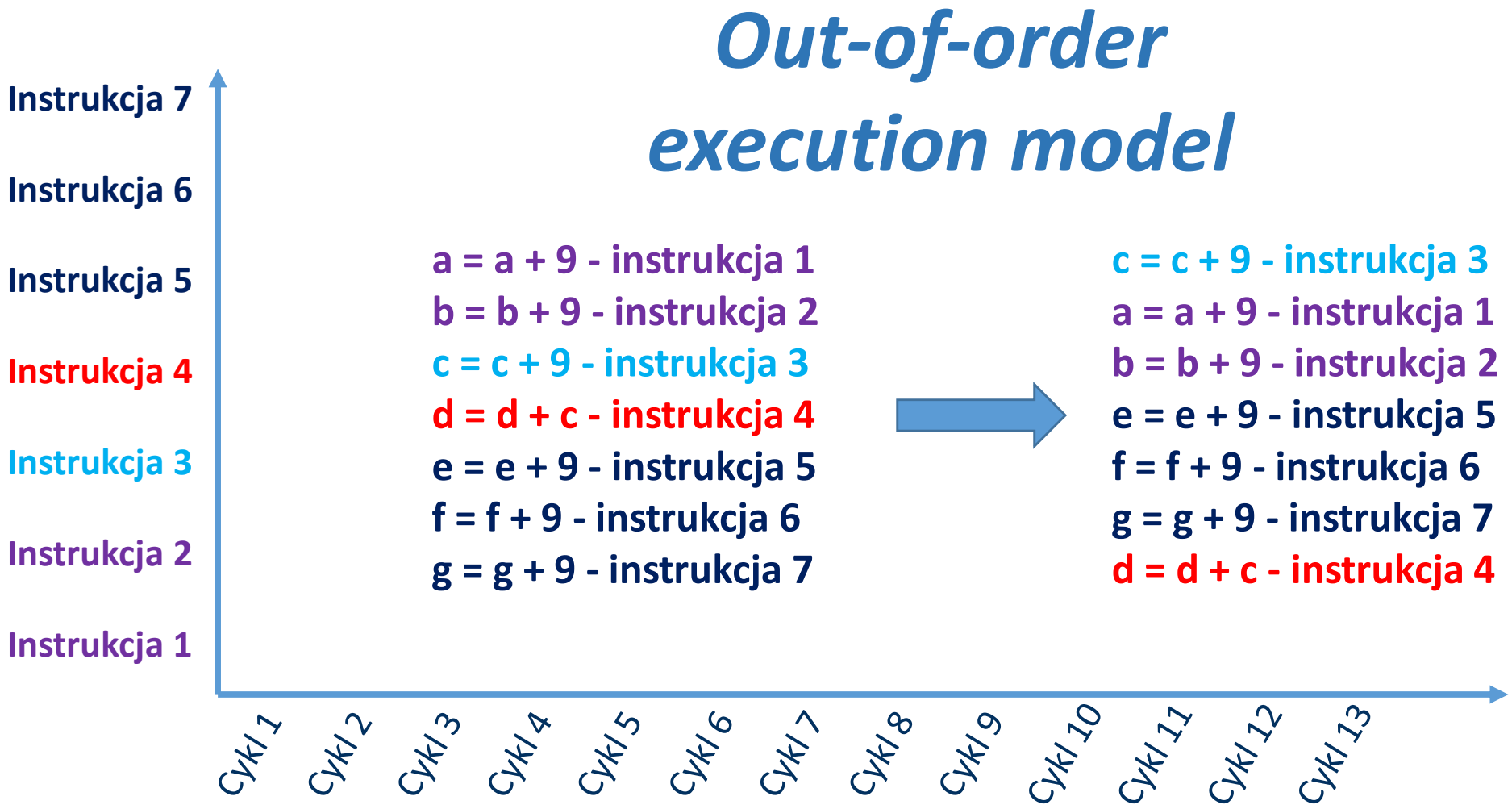


Jednostka centralna CPU: przetwarzanie potokowe - koncepcja

Out-of-order execution model

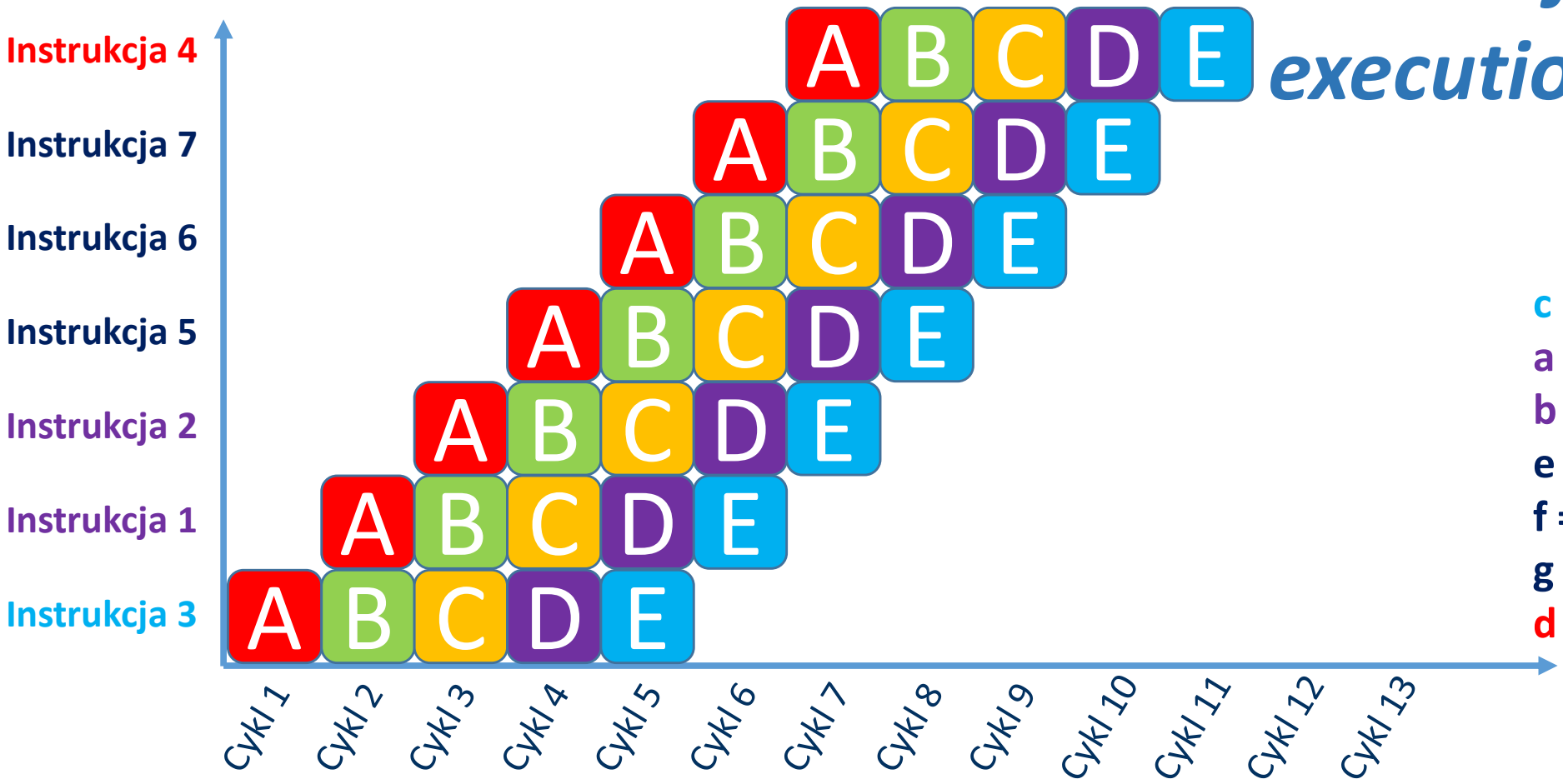


Jednostka centralna CPU: przetwarzanie potokowe - koncepcja



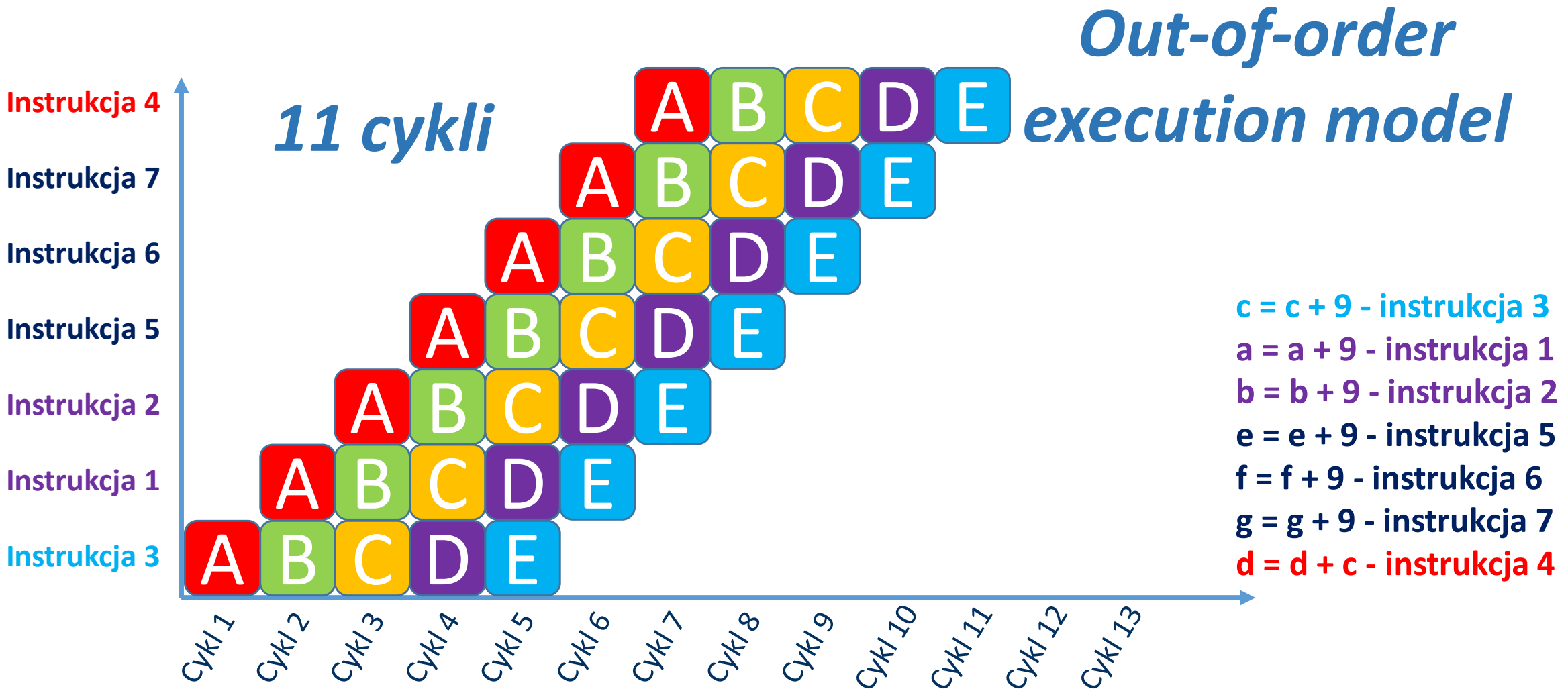
Jednostka centralna CPU: przetwarzanie potokowe - koncepcja

Out-of-order execution model



c = c + 9 - instrukcja 3
a = a + 9 - instrukcja 1
b = b + 9 - instrukcja 2
e = e + 9 - instrukcja 5
f = f + 9 - instrukcja 6
g = g + 9 - instrukcja 7
d = d + c - instrukcja 4

Jednostka centralna CPU: przetwarzanie potokowe - koncepcja

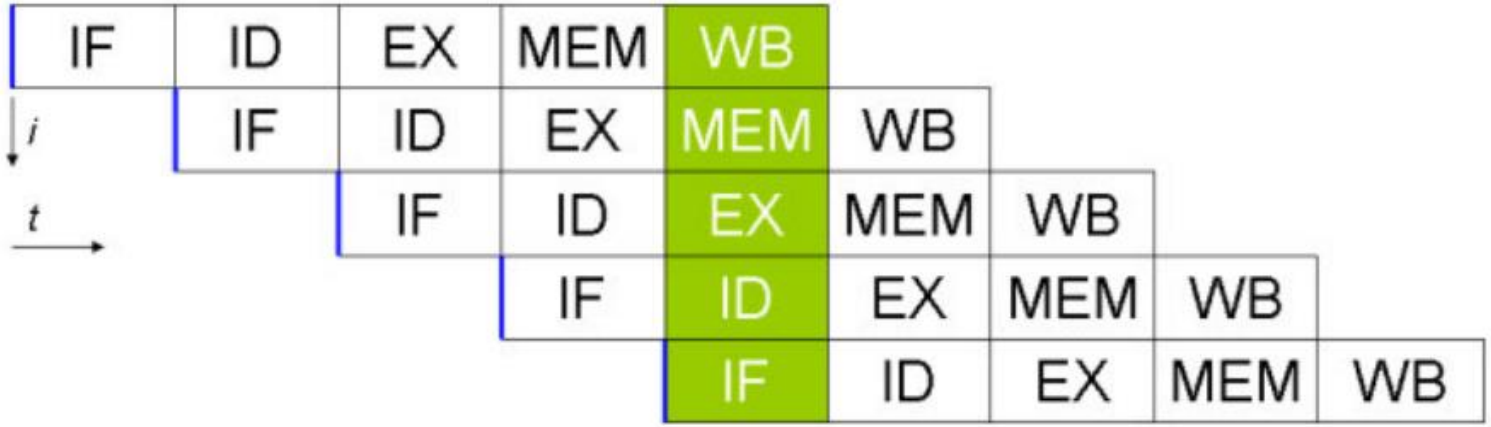


Jednostka centralna CPU: przetwarzanie potokowe

- Mimo, że wykonanie jednej instrukcji może wymagać wielu etapów, to w każdym cyklu rozpoczyna się i kończy wykonywanie jednej instrukcji
- "W praktyce taki wyidealizowany model nie może być zrealizowany"
- Powodem jest występowanie zależności między danymi, zależności sterowania i zależności zasobów
- Ponadto poszczególne instrukcje mogą wymagać różnej liczby etapów
- Trzeba też uwzględnić obsługę sytuacji wyjątkowych
- Czas wykonania instrukcji przy przetwarzaniu potokowym jest dłuższy niż przy jej wykonaniu w sposób sekwencyjny, ale średni czas wykonania jednej instrukcji przy wykonywaniu dostatecznie długiego ciągu instrukcji jest krótszy

Jednostka centralna CPU: przetwarzanie potokowe

- Poniższy rysunek przedstawia uproszczony przebieg wykonania instrukcji procesora



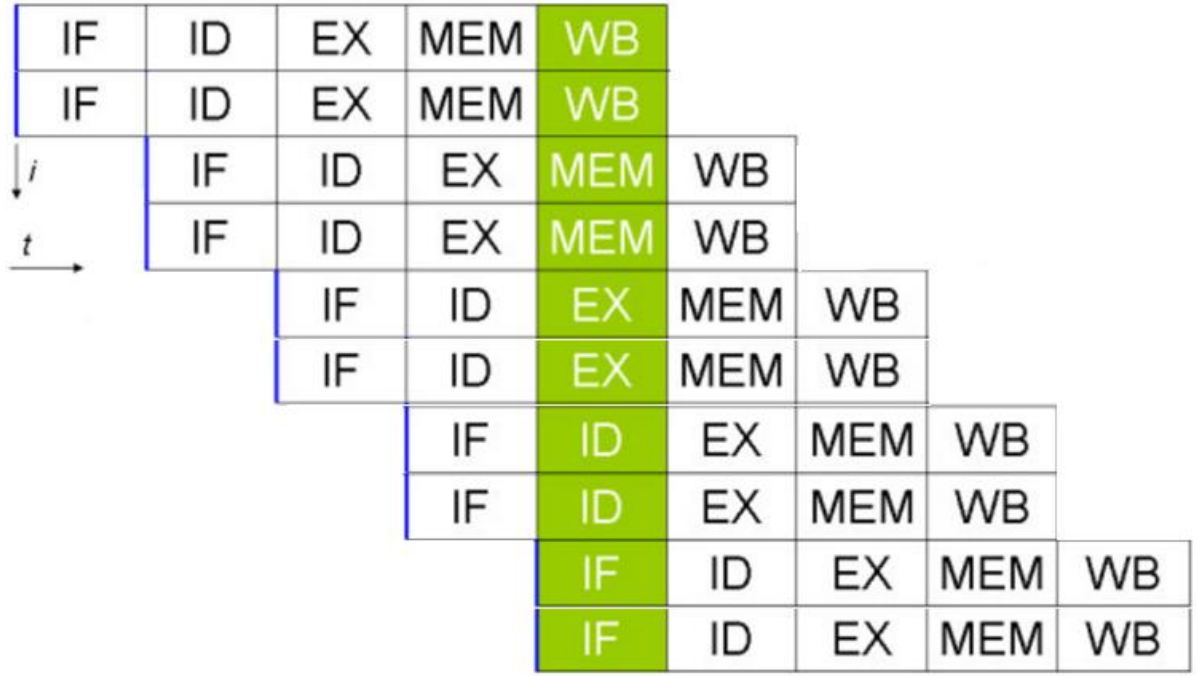
1. Pobranie instrukcji z pamięci - instruction fetch (IF)
2. Zdekodowanie instrukcji - instruction decode (ID)
3. Wykonanie instrukcji - execute (EX)
4. Dostęp do pamięci - memory access (MEM)
5. Zapisanie wyników działania instrukcji - store; write back (WB)

Jednostka centralna CPU: przetwarzanie potokowe

- W powyższym 5-stopniowym potoku, przejście przez wszystkie stopnie potoku (wykonanie jednej instrukcji) zabiera co najmniej 5 cykli zegarowych
- Jednak ze względu na jednoczesną pracę wszystkich stopni potoku, jednocześnie wykonywanych jest 5 rozkazów procesora, każdy w innym stadium wykonania
- Oznacza to, że taki procesor w każdym cyklu zegara rozpoczyna i kończy wykonanie jednej instrukcji i statystycznie wykonuje rozkaz w jednym cyklu zegara
- Każdy ze stopni potoku wykonuje mniej pracy w porównaniu do pojedynczej logiki, dzięki czemu może wykonać ją szybciej – z większą częstotliwością - tak więc dodatkowe zwiększenie liczby stopni umożliwia osiągnięcie coraz wyższych częstotliwości pracy

Jednostka centralna CPU: przetwarzanie potokowe - superskalarność

- Obecne systemy oferują w swojej budowie kilka jednostek wykonawczych, np. Intel Skylake ma 4 jednostki ALU (slajd 31- 32)
- Jest to cecha mikroprocesorów oznaczająca możliwość jednoczesnego rozpoczęcia/ukończenia kilku instrukcji w pojedynczym cyklu zegara



Jednostka centralna CPU: przetwarzanie potokowe - superskalarność

- Pełne wykorzystanie wszystkich jednostek wykonawczych zależy od tego, czy w programie nie występują zależności między kolejnymi instrukcjami (np. czy kolejna instrukcja jako argumentu nie potrzebuje wyników poprzedniego rozkazu):

```
a = b + 5
c = a + 10
```

Instrukcje nie będą mogły zostać wykonane równolegle, ponieważ wartość c zależy od wyliczanego wcześniej a

- Natomiast

```
a = b + 5
c = b + 15
```

Ponieważ instrukcje są niezależne, wykonywanie superskalarne będzie możliwe

- Ponadto współczesne procesory mogą zmieniać kolejność wykonywania instrukcji (out-of-order)

Dziękuję za uwagę

Kontakt:
dr hab. inż. Łukasz Szustak, prof. PCz
lszustak@icis.pcz.pl
Katedra Informatyki
Wydział Inżynierii Mechanicznej i Informatyki
Politechnika Częstochowska