Architektura Systemów Komputerowych

Architektura i elementy składowe systemów komputerowych

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

Katedra Informatyki
Wydział Inżynierii Mechanicznej i Informatyki
Politechnika Częstochowska

Architektura Systemów Komputerowych

Dr hab. inż. Łukasz Szustak
lszustak@icis.pcz.pl

Katedra Informatyki
Wydział Inżynierii Mechanicznej i Informatyki
Politechnika Częstochowska

Wprowadzenie

- Na przestrzeni kilkudziesięciu lat zauważalny jest niesamowity postęp dla systemów komputerowych
- Obecnie telefon komputerowy, którego koszt zakupu nie przekracza $500 oferuje większą wydajność aniżeli najszybszy superkomputer świata w 1993r, którego koszt budowy sięgał $50 million
- Tak szybki rozwój zapewniony jest dzięki:
  - Nieustannemu postępowi dla technologii używanej do budowy komputerów
  - innowacji w projektowaniu komputerów

Wprowadzenie

- W ciągu pierwszych 25 lat istnienia systemy komputerowe zapewniały poprawę wydajności na poziomie ok 25% rocznie
- Zdolność mikroprocesorów do wdrażania kolejnych ulepszeń w procesie tworzenia układów scalonych doprowadziła do zwiększania współczynnika przyrostu wydajności do poziomu 35% na rok

Komputer

- Komputery wywodzą się od mechanicznych maszyn liczących
- Komputer (dawne nazwy: elektroniczna maszyna cyfrowa, maszyna matematyczna) w najszerszym tego słowa znaczeniu to maszyna licząca, służąca do przetwarzania wszelkich informacji, które da się zapisać w formie ciągu cyfr, albo sygnału ciągłego
- Komputer umożliwia:
  - Wykonywania wielokrotnie, automatycznie powtarzanych obliczeń, wg algorytmicznego wzorca zwanego programem
  - Umożliwia wprowadzanie, przechowywanie oraz wyprowadzanie danych/informacji, które da się zapisać w formie ciągu cyfr albo sygnału ciągłego

Komputer = sprzęt + oprogramowanie

Generacje komputerów

- Generacje komputerów – umowny podział komputerów cyfrowych, zależnie od zastosowanej technologii:
  - 0 generacja – przed pojawieniem się uniwersalnych, elektronicznych maszyn cyfrowych, np. przekaźnikowy Z3
  - 1 generacja – budowane na lampach elektronowych
  - 2 generacja – budowane na tranzystorach
  - 3 generacja – budowane na układach scalonych małej i średniej skali integracji
  - **4 generacja – budowane na układach scalonych wielkiej skali integracji, mikroprocesorach (obecne komputery)**
  - 5 generacja – projekty o niekonwencjonalnych rozwiązaniach, np. komputer optyczny, komputer kwantowy

Ogólny podział komputerów

- Ze względu na możliwość przeznaczenia i budowę komputery można podzielić na:
  - *Personal Mobile Device (PMD)*
  - *Desktop Computing*
  - *Servers*
  - *Clusters / Warehouse Scale Computers*
  - *Embedded Computers*

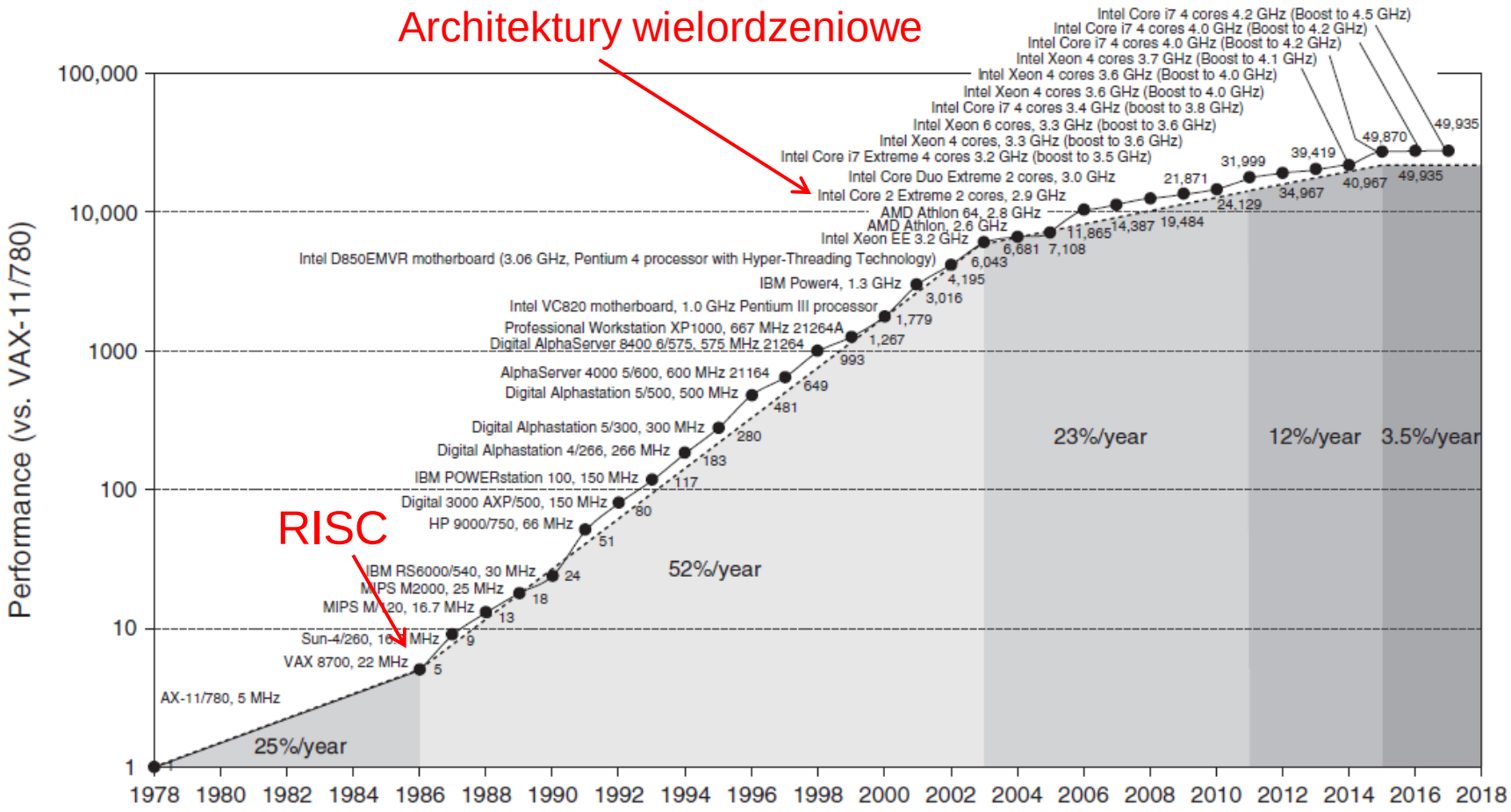
Ogólny podział komputerów: porównanie

Feature	Personal mobile device (PMD)	Desktop	Server	Clusters/warehouse-scale computer	Internet of things/ embedded
Price of system	$100–$1000	$300–$2500	$5000–$10,000,000	$100,000–$200,000,000	$10–$100,000
Price of microprocessor	$10–$100	$50–$500	$200–$2000	$50–$250	$0.01–$100
Critical system design issues	Cost, energy, media performance, responsiveness	Price-performance, energy, graphics performance	Throughput, availability, scalability, energy	Price-performance, throughput, energy proportionality	Price, energy, application-specific performance

Figure 1.2 A summary of the five mainstream computing classes and their system characteristics.

Computer Architecture: A Quantitative Approach Hennessy, John L., Patterson, David A.

Wzrost wydajności systemów komputerowych



Computer Architecture: A Quantitative Approach Hennessy, John L., Patterson, David A.

Wydajności procesorów i pamięci

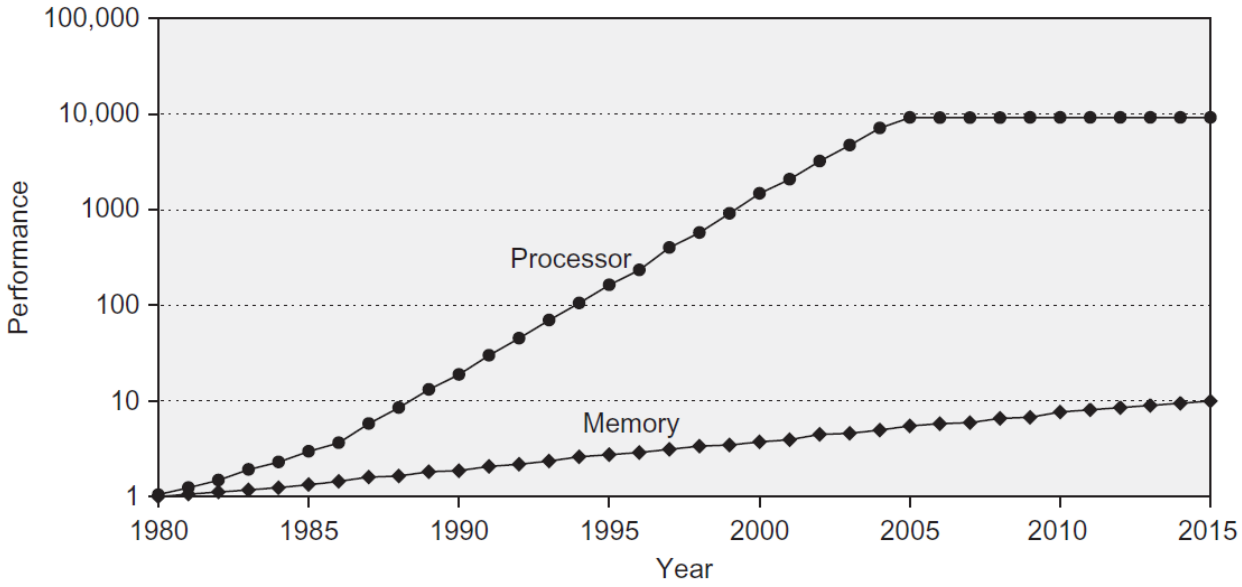


Figure 2.2 Starting with 1980 performance as a baseline, the gap in performance, measured as the difference in the time between processor memory requests (for a single processor or core) and the latency of a DRAM access, is plotted over time. In mid-2017, AMD, Intel and Nvidia all announced chip sets using versions of HBM technology. Note that the vertical axis must be on a logarithmic scale to record the size

Computer Architecture: A Quantitative Approach Hennessy, John L., Patterson, David A.

Wzrost wydajności systemów komputerowych

- 2003/2004 – koniec „ery” procesorów „sekwencyjnych”:
  - *In 1974 Robert Dennard observed that power density was constant for a given area of silicon even as you increased the number of transistors because of smaller dimensions of each transistor. Remarkably, transistors could go faster but use less power*
    Stanowiło to główna koncepcja dla wzrostu wydajności, która napotkała ok 2004 barierę technologiczną:
  - *Dennard scaling ended around 2004 because current and voltage* ***couldn’t keep dropping and still maintain the dependability of integrated circuits***
  - **This change forced the microprocessor industry to use multiple efficient processors or cores instead of a single inefficient processor.**
- Kolejne (obecne) metody zwiększania wydajności bazują na zastosowaniu równoległości na wielu poziomach
- W konsekwencji ich wykorzystanie wiąże się z przebudowaniem aplikacji

Computer Architecture: A Quantitative Approach Hennessy, John L., Patterson, David A.

Wzrost wydajności systemów komputerowych

- Równoległość na wielu poziomach jest obecnie siłą napędową projektowania komputerów we wszystkich czterech klasach komputerów, przy czym głównym ograniczeniem są energia i koszty
- Z punktu widzenia aplikacji/programu/użytkownika możemy wyróżnić dwa rodzaje zrównoleglenia:
  - Data-level parallelism (DLP) – zrównoleglanie na poziomie danych: istnieje wiele elementów danych, na których można operować w tym samym czasie.
  - Task-level parallelism (TLP) – zrównoleglanie na poziomie zadań: realizowane zadania mogą działać niezależnie i w dużej mierze równolegle

Wzrost wydajności systemów komputerowych

- DLP oraz TLP realizowane są w obecnych systemach komputerowych wszystkich grup (Personal Mobile Device, Desktop Computing, Servers, Clusters / Warehouse Scale Computers, Embedded Computers):
  - ***Instruction-level parallelism***: przykład wykorzystanie DLP poprzez zastosowanie np. przetwarzanie potokowego
  - ***Vector architectures, graphic processor units (GPUs), and multimedia instruction sets***: przykład wykorzystanie DLP poprzez przetwarzanie pojedynczej instrukcji na wielu danych
  - **Thread-level parallelism**: umożliwia wykorzystanie zarówno DLP jak i TLP poprzez zastosowanie przetwarzania współbieżnego wielu wątków uruchamianych na pojedynczym zasobie obliczeniowym
  - **Request-level parallelism:** umożliwia realizację TLP poprzez określenie dużej liczby zdań definiowanych przez programistę lub system operacyjny
- A zatem obecne systemy komputerowe w swojej naturze są równoległe

Computer Architecture: A Quantitative Approach Hennessy, John L., Patterson, David A.

Przyszłość komputerów

- Prawa fizyki ograniczają możliwości miniaturyzacji układów scalonych, a zagwarantowanie zasilania dla całego układu staje się coraz do bardziej trudniejsze:

  Dark silicon *- dark silicon is the amount of circuitry of an integrated circuit that cannot be* ***powered-on*** *at the nominal operating voltage for a given thermal design power (TDP) constraint*

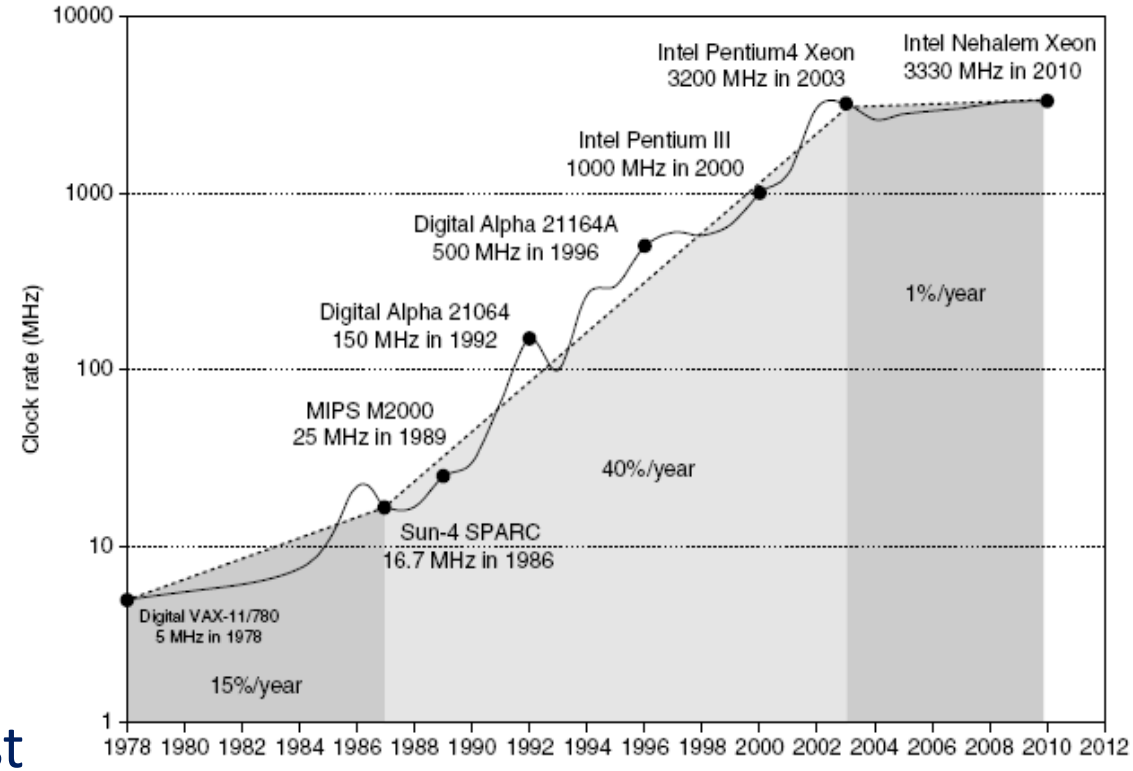
- Wraz ze wzrostem częstotliwości taktowania procesora znacząco rośnie moc wydzielana w postaci ciepła
- Obecnie głównym trendem dla rozwoju komputerów są architektury masywnie równoległe, a równoległość realizowana jest na wielu płaszczyznach
- Niestety, ten trend rozwoju powoli dochodzi do bariery technologicznej

Rozwój technologii w systemach komputerowych

- Technologia układów scalonych:
  - Wzrost gęstości tranzystorów (*Transistor density*): 35% na rok **10 mikrometrów in 1971 to 0.007 mikrometra in 2018**
  - Wzrost rozmiaru układu scalonego: 10-20% na rok
  - Integracja ogólnie: 40-55% na rok
- Zwiększanie rozmiaru DRAM ok 25-40% na rok (ogólnie zwalnia)
- Zwiększanie rozmiaru pamięci Flash: 50-60% na rok
  - Koszt w przeliczeniu na bit jest ok 15-20x taniej niż DRAM
- Zwiększenie pojemności dysków magnetycznych ok 40% na rok
  - Koszt w przeliczeniu na bit jest ok 15-25x taniej niż Flash
  - Koszt w przeliczeniu na bit jest ok 300-500x taniej niż DRAM

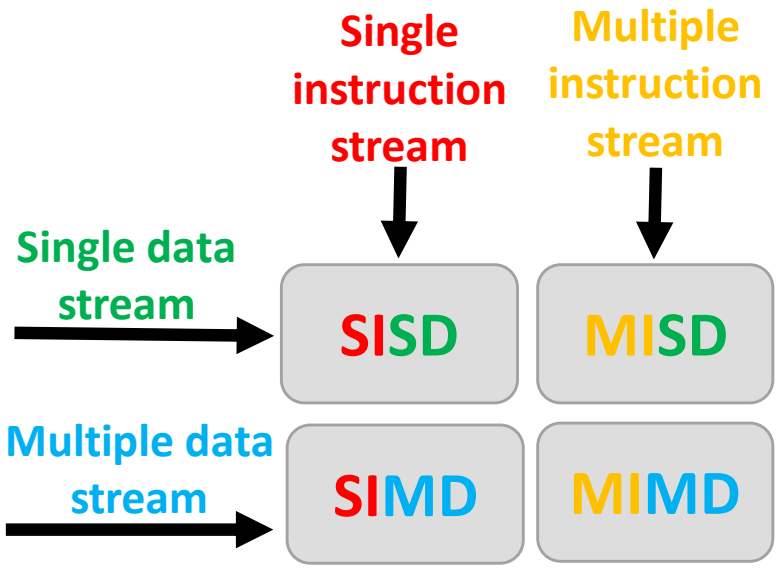
Moc i zużycie energii elektrycznej

- Intel 80386 zużywał ok 2 W, natomiast 3.3 GHz Intel Core i7 zużywa ok 130 W
- Wytworzone ciepło musi być odprowadzane z układów scalonych o niewielkich wymiarach
- W konsekwencji, układy chłodzenia powietrzem są obecnie na granicy swojej możliwości
- Obecnie, częstotliwość taktowanie procesora jest dynamicznie zmieniana, aby ograniczyć pobieraną moc, a w efekcie zmniejszyć ilość wydzielanego ciepła



Taksonomia Flynna

- Zaproponowana przez Michael J. Flynna w 1966r.
- Taksonomia Flynna jest jedną z najwcześniejszych klasyfikacji systemów dla równoległych i sekwencyjnych komputerów i programów
- Klasyfikacja Flynna bazuje na **liczbie przetwarzanych strumieni danych i strumieni rozkazów**
- Taksonomia Flynna wyróżnić cztery następujące grupy: **SISD, SIMD, MISD, MIMD**
- **Taksonomia Flynna opisuje współczesne systemy, chociaż wiele obecnych systemów to połączenie powyższych grup**



Taksonomia Flynna: SISD

- **SISD** *(ang. Single Instruction, Single Data)* jest to architektura klasycznego komputera sekwencyjnego, zawierającego jeden procesor i jeden blok pamięci operacyjnej
- W ramach tej architektury przetwarzany jest jeden strumień danych przez jeden wykonywany program (procesor), a ciąg instrukcji wykonywany jest sekwencyjnie

Taksonomia Flynna: SIMD

- **SIMD** (ang. Single Instruction, Multiple Data) jest to architektura komputerów, w których przetwarzanych jest wiele strumieni danych w oparciu o pojedynczy strumień rozkazów
- Architektura SIMD jest charakterystyczna dla komputerów wektorowych
- Pierwsze komputery wektorowe stosowano w latach 70 np. Cray 1 (1972)
- Obecne komputery również realizują zadania zgodnie z metodologią SIMD

Taksonomia Flynna: MISD

- **MISD** (*ang. multiple instruction, single data)* jest to architektura przetwarzania równoległego, w której wiele równolegle wykonywanych programów przetwarza jednocześnie jeden wspólny strumień danych
- Głównym zastosowaniem są systemy wykorzystujące redundancję (wielokrotne wykonywanie tych samych obliczeń) w celu minimalizacji błędów

Taksonomia Flynna: MIMD

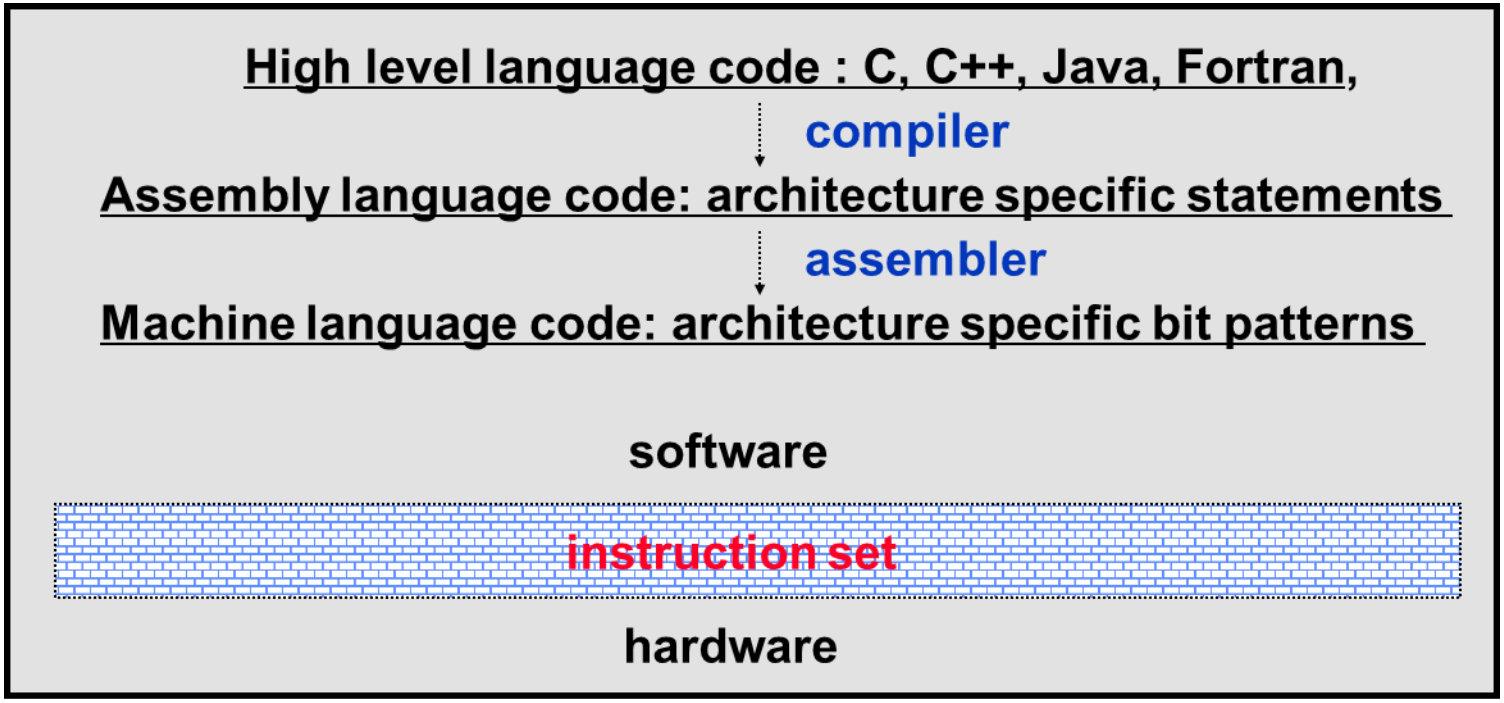
- **MIMD** *(ang. multiple instruction, multiple data)* jest to architektura komputerów, dla której przetwarzanie równoległe zachodzi zarówno na poziomie danych jak i instrukcji
- Przykładem mogą być komputery wieloprocesorowe, a także klastry i gridy
- Komputery zbudowane w architekturze MIMD posiadają wiele procesorów pracujących niezależnie i asynchronicznie umożliwiając tym samym wykonywanie różnych instrukcji na odmiennych danych

Projektowanie systemów komputerowych

- Zadanie, przed którym stoi projektant systemów komputerów, jest „dość” złożone: „*W pierwszej kolejności należy określić jakie atrybuty są ważne dla nowego komputera, a następnie zaprojektować komputer tak, aby zmaksymalizować wydajność i energooszczędność, zachowując jednocześnie ograniczenia związane z kosztami, mocą i dostępnością”*
- To zadanie związane jest z:
  - Instruction set architecture (ISA)
  - Functional organization
  - logic design, and implementation

Instruction set architecture (ISA)

- ISA jest pomostem pomiędzy warstwą oprogramowania a sprzętem



Instruction set architecture (ISA)

- ISA opisuje:
  - Typy danych i operandy
  - Pamięć (rejstry….)
  - Modele adresowania
  - Zbiór instrukcji
  - I/O
- ISA jest opracowywana dla danego rodzaju architektury, np 80x86, ARMv8, and RISC-V:
- ISA dla x86 jest wspierane przez procesory firmy:
  - Intel
  - AMD

Instruction set architecture (ISA)

- ISA opisuje:
  - *Where are operands stored?*
    - *registers, memory, stack, accumulator*
  - *How many explicit operands are there?*
    - *0, 1, 2, or 3*
  - *How is the operand location specified?*
    - *register, immediate, indirect, . . .*
  - *What type & size of operands are supported?*
    - *byte, int, float, double, string, vector. . .*
  - *What operations are supported?*
    - *add, sub, mul, move, compare . . .*

Computer Architecture: A Quantitative Approach Hennessy, John L., Patterson, David A.

Instruction set architecture: przykładowy zestawu instrukcj

Instruction type/opcode	Instruction meaning
Floating point	*FP operations on DP and SP formats*
`fadd.d`, `fadd.s`	Add DP, SP numbers
`fsub.d`, `fsub.s`	Subtract DP, SP numbers
`fmul.d`, `fmul.s`	Multiply DP, SP floating point
`fmadd.d`, `fmadd.s`, `fnmadd.d`, `fnmadd.s`	Multiply-add DP, SP numbers; negative multiply-add DP, SP numbers
`fmsub.d`, `fmsub.s`, `fnmsub.d`, `fnmsub.s`	Multiply-sub DP, SP numbers; negative multiply-sub DP, SP numbers
`fdiv.d`, `fdiv.s`	Divide DP, SP floating point
`fsqrt.d`, `fsqrt.s`	Square root DP, SP floating point
`fmax.d`, `fmax.s`, `fmin.d`, `fmin.s`	Maximum and minimum DP, SP floating point
`fcvt._._`, `fcvt._._u`, `fcvt._u._`	Convert instructions: `FCVT.x.y` converts from type `x` to type `y`, where `x` and `y` are `L` (64-bit integer), `W` (32-bit integer), `D` (DP), or `S` (SP). Integers can be unsigned (`U`)
`feq._`, `flt._`,`fle._`	Floating-point compare between floating-point registers and record the Boolean result in integer register; "__" = `S` for single-precision, `D` for double-precision
`fclass.d`, `fclass.s`	Writes to integer register a 10-bit mask that indicates the class of the floating-point number ($-\infty$, $+\infty$, -0, $+0$, NaN, …)
`fsgnj._`, `fsgnjn._`, `fsgnjx._`	Sign-injection instructions that changes only the sign bit: copy sign bit from other source, the oppositive of sign bit of other source, XOR of the 2 sign bits

Figure 1.6 Floating point instructions for RISC-V.

Computer Architecture: A Quantitative Approach Hennessy, John L., Patterson, David A.

Instruction set architecture: przykład sposobów adresowania

Addressing Mode	Example	Action
1. Register direct	Add R4, R3	R4 <- R4 + R3
2. Immediate	Add R4, #3	R4 <- R4 + 3
3. Displacement	Add R4, 100(R1)	R4 <- R4 + M[100 + R1]
4. Register indirect	Add R4, (R1)	R4 <- R4 + M[R1]
5. Indexed	Add R4, (R1 + R2)	R4 <- R4 + M[R1 + R2]
6. Direct	Add R4, (1000)	R4 <- R4 + M[1000]
7. Memory Indirect	Add R4, @(R3)	R4 <- R4 + M[M[R3]]
8. Autoincrement	Add R4, (R2)+	R4 <- R4 + M[R2] R2 <- R2 + d
9. Autodecrement	Add R4, (R2)-	R4 <- R4 + M[R2] R2 <- R2 - d
10. Scaled	Add R4, 100(R2)[R3]	R4 <- R4 + M[100 + R2 + R3*d]

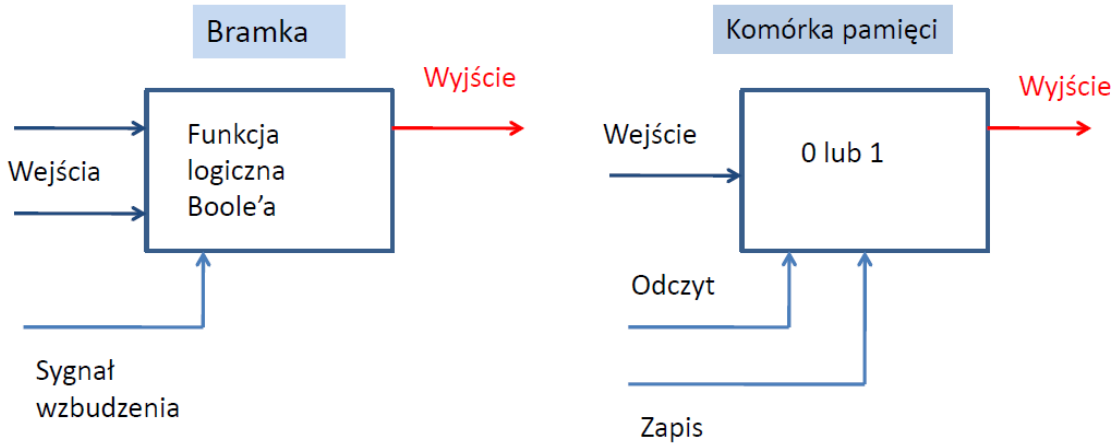
Computer Architecture: A Quantitative Approach Hennessy, John L., Patterson, David A.

Ogólna struktura komputera

- **Jednostka centralna (CPU)** – steruje działaniem komputera i realizuje funkcje przetwarzania danych
- **Pamięć główna** – przechowuje dane wejściowe obliczeń i wyniki obliczeń
- **Wejście/wyjście (I/O)**– przenoszą dane pomiędzy komputerem a jego otoczeniem
- **Połączenia systemowe** – zapewniają łączność pomiędzy procesorem, pamięcią główną a układami wejście-wyjście

Struktura komputera

- Podstawowe elementy komputera:



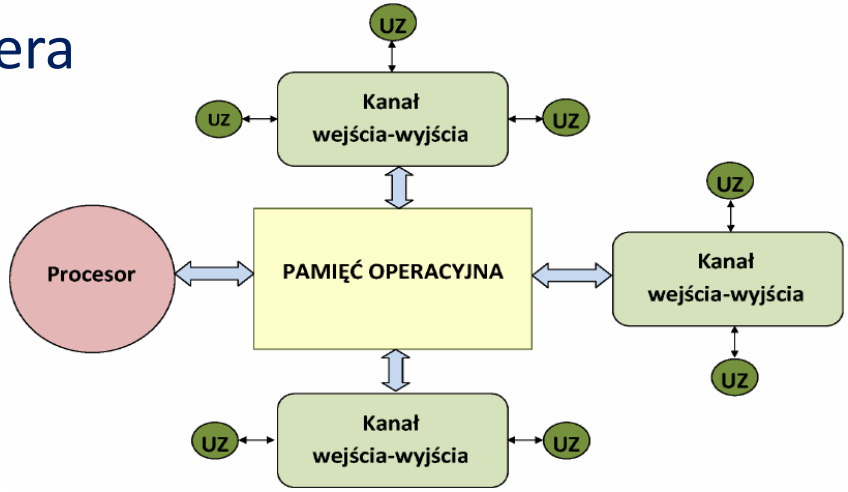
- Podstawowe funkcje realizowane przez komputery:
  - Przetwarzanie danych (bramki)
  - Przechowywanie danych (komórka pamięci)
  - Przenoszenie danych (ścieżki pomiędzy podzespołami)
  - Sterowanie (ścieżki pomiędzy podzespołami)

Organizacja komputera

- Sposób organizacji architektury komputera PC przeszedł długą ewolucję, wraz ze wzrostem wymagań (prędkość przesyłania danych) stawianych komputerom PC szukano nowych rozwiązań umożliwiających jak najwydajniejszą pracę:
  - Architektura pamięciowo-centryczna
  - Architektura szynowa
  - Architektura dwuszynowa
  - Architektura trójszynowa
  - Architektury z połączeniami punkt-punkt

Architektura pamięciowo-centryczna

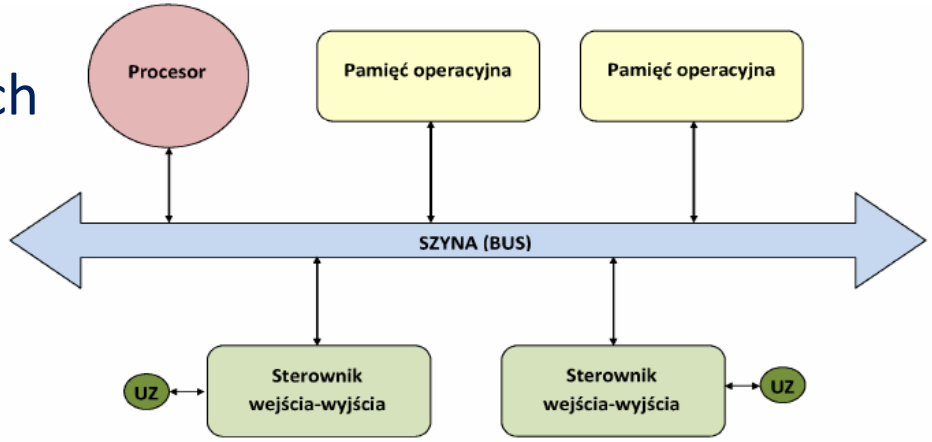
- Pamięć stanowi centrum komputera
- Do pamięci są podłączone:
  - procesor lub procesory
  - kanały wejścia-wyjścia, czyli specjalizowane procesory transmitujące dane pomiędzy urządzeniami zewnętrznymi i pamięcią komputera



- Architektura pamięcio-centryczna charakteryzowała się:
  - szybką wymiana danych z urządzeniami zewnętrznymi
  - małą elastyczność konfiguracji wynikająca z ograniczonej liczby interfejsów pamięci
  - dużym rozmiarem urządzenia

Architektura szynowa

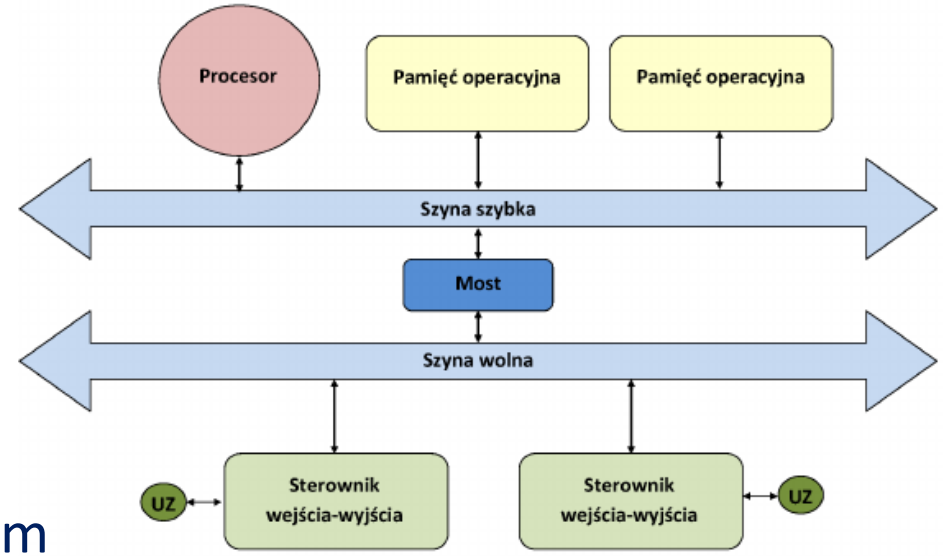
- Struktura szynowa zastosowana w tzw. minikomputerach w latach 70-tych XX wieku szybko stała się podstawową strukturą dla komputerów o różnych zastosowaniach i mocach obliczeniowych



- Głównym elementem komputera jest szyna, czyli zespół przewodów połączony gniazdami
- Komputer składał się z wielu modułów dołączonych do szyny: procesorów, bloków pamięci i sterowników wejścia-wyjścia
- Komputer miał postać kasety lub szafy z wymiennymi modułami/szufladami

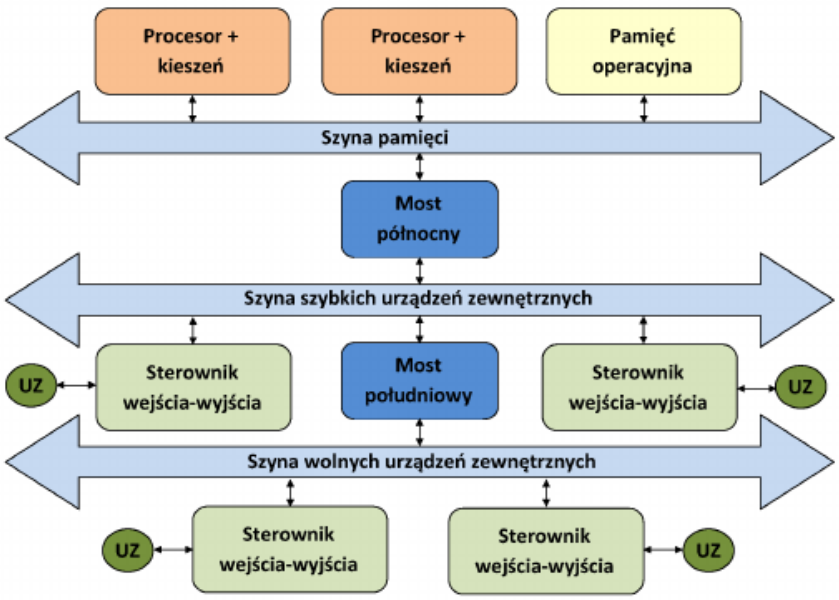
Architektura dwu szynowa

- Szybka, „krótka” szyna o dużej wydajności i przepustowości łączy procesor (lub procesory) z pamięcią
- Do dłuższej, wolniejszej szyny są dołączone sterowniki urządzeń wejścia wyjścia
- Obie szyny są połączone układem zapewniającym wymianę danych tzw. most
- Logicznie obie szyny są widziane przez procesor jak jedna szyna, natomiast główna różnica dotyczy parametrów technicznych czyli kosztów przesyłania danych



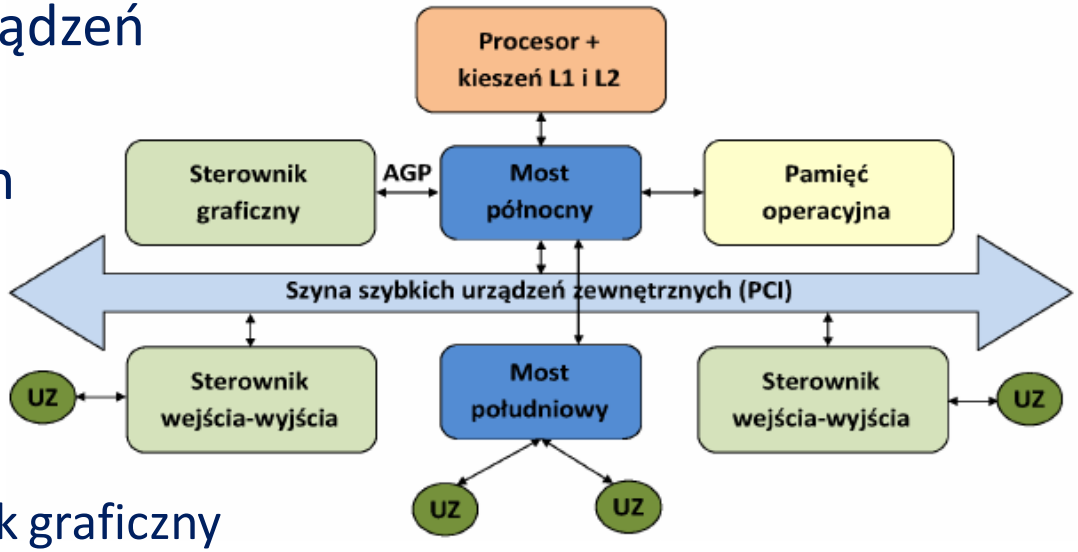
Architektura trój szynowa

- Trzy szyny:
  - procesora i pamięci
  - szybkich urządzeń zewnętrznych (PCI)
  - wolnych urządzeń zewnętrznych (ISA)
- Dwa mosty
  - "północny" łączy szynę procesora z szyną szybkich urządzeń
  - "południowy" łączy szynę szybkich urządzeń z szyną wolnych urządzeń
- Most południowy zawierał sterowniki niektórych urządzeń
- Sterownik pamięci umieszczony w moście północnym



Architektury z połączeniami punkt-punkt

- Nie ma szyny wolnych urządzeń wejścia-wyjścia
- Część połączeń szynowych zastąpiono połączeniem typu punkt-punkt:
  - procesor – most północny
  - most północny – pamięć
  - most północny – sterownik graficzny
  - most północny – most południowy



- Pozostała szyna dedykowana jest dla urządzeń PCI
- Most północny zawiera sterownik pamięci
- Most południowy nie pełni rolę mostu pomiędzy szynami lecz zawiera sterowniki większości niezbędnych w komputerze urządzeń zewnętrznych

Dziękuję za uwagę

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