Rechenintensive parallele Anwendungen können nicht sinnvoll ohne Kenntnis der zugrundeliegenden Architektur erstellt werden.

Deswegen ist die Wahl einer geeigneten Architektur bzw. die Anpassung eines Algorithmus an eine Architektur von entscheidender Bedeutung für die effiziente Nutzung vorhandener Ressourcen.

Die Aufnahme von Steve Jurvetson (CC-AT-2.0, Wikimedia Commons) zeigt den 1965–1976 entwickelten Parallelrechner ILLIAC 4.

- Die sequentielle Arbeitsweise eines Prozessors kann durch verschiedene Parallelisierungstechniken beschleunigt werden (z.B. durch Pipelining oder die Möglichkeit, mehrere Instruktionen gleichzeitig auszuführen).
- Einzelne Operationen lassen sich auf größere Datenmengen gleichzeitig anwenden (z.B. für eine Vektoraddition).
- Die Anwendung wird aufgeteilt in unabhängig voneinander rechnende Teile, die miteinander kommunizieren (über gemeinsamen Speicher und/oder Nachrichten) und auf Mehrprozessorsystemen, Multicomputern oder mehreren unabhängigen Rechnern verteilt sind.

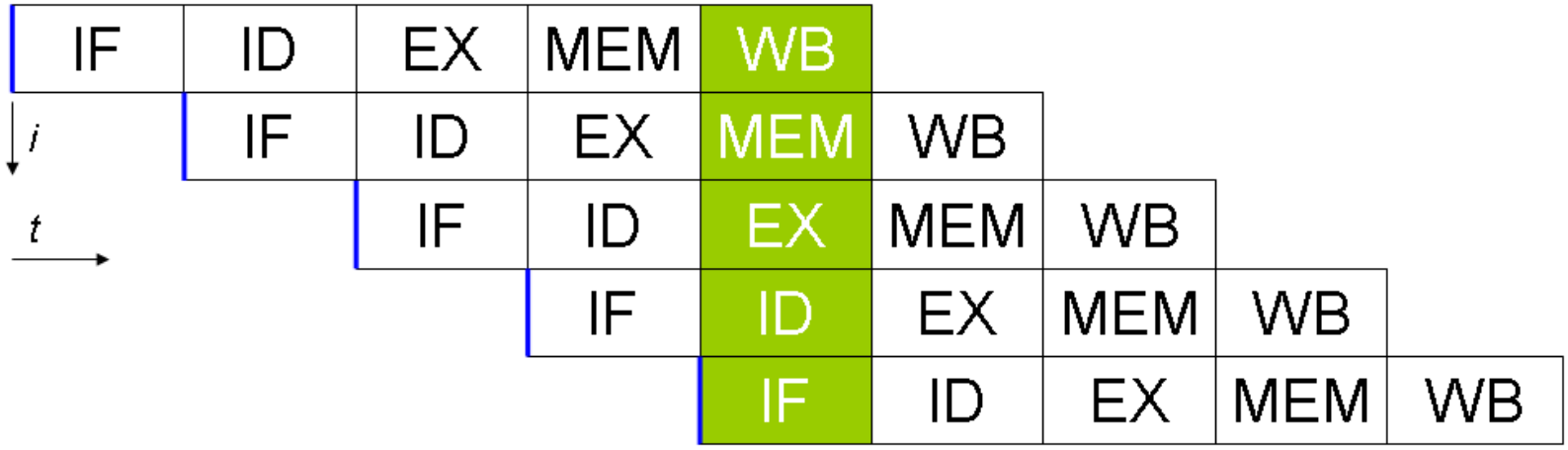
- Eine Anwendung wird durch eine Parallelisierung nicht in jedem Fall schneller.
- Es entstehen Kosten, die sowohl von der verwendeten Architektur als auch dem zum Einsatz kommenden Algorithmus abhängen.
- Dazu gehören:
  - Konfiguration
  - Kommunikation
  - Synchronisierung
  - Terminierung
- Interessant ist auch immer die Frage, wie die Kosten skalieren, wenn der Umfang der zu lösenden Aufgabe und/oder die zur Verfügung stehenden Ressourcen wachsen.

- Moderne Prozessoren arbeiten nach dem Fließbandprinzip: Über das Fließband kommen laufend neue Instruktionen hinzu und jede Instruktion wird nacheinander von verschiedenen Fließbandarbeitern bearbeitet.
- Dies parallelisiert die Ausführung, da unter günstigen Umständen alle Fließbandarbeiter gleichzeitig etwas tun können.
- Eine der ersten Pipelining-Architekturen war die IBM 7094 aus der Mitte der 60er-Jahre mit zwei Stationen am Fließband. Die UltraSPARC-IV-Architektur hat 14 Stationen.
- Die RISC-Architekturen (RISC = *reduced instruction set computer*) wurden speziell entwickelt, um das Potential für Pipelining zu vergrößern.
- Bei der Pentium-Architektur werden im Rahmen des Pipelinings die Instruktionen zuerst intern in RISC-Instruktionen konvertiert, so dass die x86-Architektur ebenfalls von diesem Potential profitieren kann.

Um zu verstehen, was alles innerhalb einer Pipeline zu erledigen ist, hilft ein Blick auf die möglichen Typen von Instruktionen:

- Operationen, die nur auf Registern angewendet werden und die das Ergebnis in einem Register ablegen.
- Instruktionen mit Speicherzugriff. Hier wird eine Speicheradresse berechnet und dann erfolgt entweder eine Lese- oder eine Schreiboperation.
- Sprünge.

Zerlegung in Arbeitsschritte



Eine einfache Aufteilung sieht folgende einzelne Schritte vor:

- Instruktion vom Speicher laden (IF)
- Instruktion dekodieren (ID)
- Instruktion ausführen, beispielsweise eine arithmetische Operation oder die Berechnung einer Speicheradresse (EX)
- Lese- oder Schreibzugriff auf den Speicher (MEM)
- Abspeichern des Ergebnisses in Registern (WB)

Das Diagramm wurde von Stephane Tsacas geklaut.

- Bedingte Sprünge sind ein Problem für das Pipelining, da unklar ist, wie gesprungen wird, bevor es zur Ausführungsphase kommt.
- RISC-Maschinen führen typischerweise die Instruktion unmittelbar nach einem bedingten Sprung immer mit aus, selbst wenn der Sprung genommen wird. Dies mildert etwas den negativen Effekt für die Pipeline.
- Im übrigen gibt es die Technik der *branch prediction*, bei der ein Ergebnis angenommen wird und dann das Fließband auf den Verdacht hin weiterarbeitet, dass die Vorhersage zutrifft. Im Falle eines Misserfolgs muss dann u.U. recht viel rückgängig gemacht werden.
- Das ist machbar, solange nur Register verändert werden. Manche Architekturen verfolgen die Alternativen sogar parallel und haben für jedes abstrakte Register mehrere implementierte Register, die die Werte für die einzelnen Fälle enthalten.
- Die Vorhersage wird vom Übersetzer generiert. Typisch ist beispielsweise, dass bei Schleifen eine Fortsetzung der Schleife vorhergesagt wird.

- Das Pipelining lässt sich dadurch noch weiter verbessern, wenn aus dem Speicher benötigte Werte frühzeitig angefragt werden.
- Moderne Prozessoren besitzen Caches, die einen schnellen Zugriff ermöglichen, deren Kapazität aber sehr begrenzt ist (dazu später mehr).
- Ebenfalls bieten moderne Prozessoren die Möglichkeit, das Laden von Werten aus dem Hauptspeicher frühzeitig zu beantragen – nach Möglichkeit so früh, dass sie rechtzeitig vorliegen, wenn sie dann benötigt werden.

Flynn schlug 1972 folgende Klassifizierung vor in Abhängigkeit der Zahl der Instruktions- und Datenströme:

Instruktionen	Daten	Bezeichnung
1	1	SISD (Single Instruction Single Data)
1	> 1	SIMD (Single Instruction Multiple Data)
> 1	1	MISD (Multiple Instruction Single Data)
> 1	> 1	MIMD (Multiple Instruction Multiple Data)

SISD entspricht der klassischen von-Neumann-Maschine, SIMD sind z.B. vektorisierte Rechner, MISD wurde wohl nie umgesetzt und MIMD entspricht z.B. Mehrprozessormaschinen oder Clustern. Als Klassifizierungsschema ist dies jedoch zu grob.

- Die klassische Variante der SIMD sind die Array-Prozessoren.
- Eine Vielzahl von Prozessoren steht zur Verfügung mit zugehörigem Speicher, die diesen in einer Initialisierungsphase laden.
- Dann wird die gleiche Instruktion an alle Prozessoren verteilt, die jeder Prozessor auf seinen Daten ausführt.
- Die Idee geht auf S. H. Unger 1958 zurück und wurde mit dem ILLIAC IV zum ersten Mal umgesetzt.
- Die heutigen GPUs übernehmen teilweise diesen Ansatz.

Bei Vektor-Prozessoren steht zwar nur ein Prozessor zur Verfügung, aber dieser ist dank dem Pipelining in der Lage, pro Taktzyklus eine Operation auf einem Vektor umzusetzen. Diese Technik wurde zuerst von der Cray-1 im Jahr 1974 umgesetzt und auch bei späteren Cray-Modellen verfolgt.

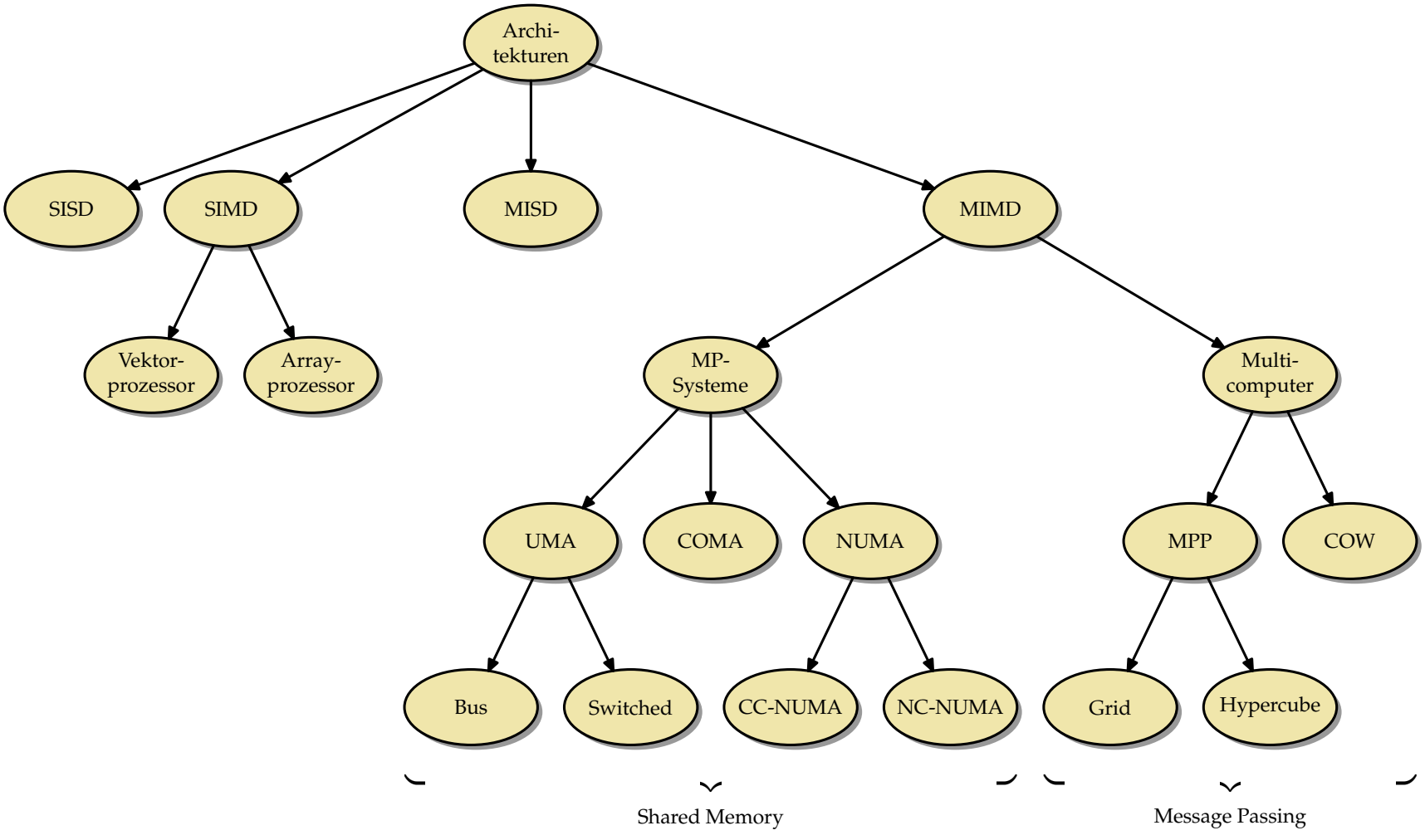
Die MMX- und SSE-Instruktionen des Pentium 4 setzen ebenfalls dieses Modell um.

Die von Rama (Wikimedia Commons, Cc-by-sa-2.0-fr) gefertigte Aufnahme zeigt eine an der EPFL in Lausanne ausgestellte Cray-1.

Hier wird unterschieden, ob die Kommunikation über gemeinsamen Speicher oder ein gemeinsames Netzwerk erfolgt:

- Multiprozessor-Systeme (MP-Systeme) erlauben jedem Prozessor den Zugriff auf den gesamten zur Verfügung stehenden Speicher. Der Speicher kann auf gleichförmige Weise allen Prozessoren zur Verfügung stehen (UMA = *uniform memory access*) oder auf die einzelnen Prozessoren oder Gruppen davon verteilt sein (NUMA = *non-uniform memory access*).
- Multicomputer sind über spezielle Topologien vernetzte Rechnersysteme, bei denen die einzelnen Komponenten ihren eigenen Speicher haben. Üblich ist hier der Zusammenschluss von Standardkomponenten (COW = *cluster of workstations*) oder spezialisierter Architekturen und Bauweisen im großen Maßstab (MPP = *massive parallel processors*).

Architekturen nach Tanenbaum



- Die Theseus gehört mit vier Prozessoren des Typs UltraSPARC IV+ mit jeweils zwei Kernen zu der Familie der Multiprozessorsysteme (MP-Systeme).
- Da der Speicher zentral liegt und alle Prozessoren auf gleiche Weise zugreifen, gehört die Theseus zur Klasse der UMA-Architekturen (*Uniform Memory Access*) und dort zu den Systemen, die Bus-basiert Cache-Kohärenz herstellen (dazu später mehr).
- Die Thales hat zwei Xeon-5650-Prozessoren mit jeweils 6 Kernen, die jeweils zwei Threads unterstützen. Wie bei der Theseus handelt es sich um eine UMA-Architektur, die ebenfalls Bus-basiert Cache-Kohärenz herstellt.

- Die Hochwanner ist eine Intel-Dualcore-Maschine (2,80 GHz) mit einer Nvidia Quadro 600 Grafikkarte.
- Die Grafikkarte hat 1 GB Speicher, zwei Multiprozessoren und insgesamt 96 Recheneinheiten (SPs = *stream processors*).
- Die Grafikkarte ist eine SIMD-Architektur, die sowohl Elemente der Array- als auch der Vektorrechner vereinigt und auch den Bau von Pipelines ermöglicht.