

Mit diesem Fragenkatalog möchte ich keinem die Arbeit wegnehmen. Ich habe die Fragen und Antworten vom alten Fragenkatalog **1:1** übernommen. Ein paar neue Fragen gibt es, die hab ich vom Forum rausgesucht. Die Kapitel beziehen sich auf die Vorlesungsfolien.

➔ keine Garantie auf Richtigkeit

µComputer – Zusammenfassung

-

Vorlesungsprüfung

Kapitel 1 – Einleitung:

1.) Schritte der IC-Produktion chronologisch auflisten: (Belichten, Entwickeln, Ätzen, Dotieren)

- (1) Entwurf und Simulation
- (2) Masken werden erstellt mit den Mustern des Schaltkreises
- (3) Ein Siliziumeinkristall wird gezüchtet und gezogen
- (4) Die Wafer werden gesägt
- (5) Der Wafer wird mit einer isolierenden Schicht aus Siliziumoxid ummantelt
- (6) Photoempfindliche Schutzschicht wird aufgebracht
- (7) Mit UV Licht wird belichtet. Wo Maske nicht abdeckt, wird Schutzschicht zerstört
- (8) Der Wafer wird entwickelt, gewaschen und gebacken zerstörte Teile der Schutzschicht werden entfernt.
- (9) Der Wafer wird chemisch geätzt. An den Stellen ohne SI wird die Schutzschicht entfernt
- (10) Der Wafer wird dotiert. Die ungeschützten SI- bahnen verändern ihre el. Eigenschaft. Schritt 5 – 10 wird beliebig wiederholt bis gewünschtes Muster erreicht wird. Man kann auch weitere Schichten aufbringen (Isolator, Cu, Al)
- (11) Für Metallisierung wird Metall (Cu) aufgebracht
- (12) Wieder eine photoempfindliche Schicht
- (13) Belichtung entsprechend der gewünschten Metallisierung
- (14) Belichtete Stellen werden chemisch entfernt
- (15) Die Metallschicht wird geätzt
- (16) Die einzelnen Funktionsgruppen des Chips sind jetzt mit leitendem Metall verbunden
- (17) Weitere Metallisierungsschichten werden aufgebracht, getrennt durch Isolierungsschichten
- (18) Die Chips werden am Wafer getestet. Fehlerhafte Chips werden markiert
- (19) Die Wafer werden gesägt
- (20) Die Chips werden paketiert und weiter getestet

2.) 4 Vorteile eines IC's gegenüber einer diskreten Transistor – Schaltung

Kleiner, Billiger, höhere Schaltgeschwindigkeit, sparsamer beim Stromverbrauch

3.) Erklären Sie wie man die Funktion eines IC's von außen bestimmen kann (Spionage) bzw. wie man sich als Hersteller schützt:

- Chipindividuell gescrambelte Busleitungen und Speicherzellen
- Dummy-Strukturen
- Stromführende Metallisierungsebenen, die den Chip umgeben (werden sie durch ätzen entfernt, ist der Chip nicht mehr funktionsfähig)
- Überwachung der Passivierungsschicht (eine Schutzschicht des Siliziumchips, dessen Entfernung einen Interrupt auslösen kann, der z.B. den Schlüssel löscht)

4.) Nenne 4 Vorteile modularisierten Designs

- Überblick: Die einzelnen Teile können getrennt voneinander betrachtet werden
- Teams: Man kann die einzelnen Teile getrennten Entwicklungsteams übergeben und damit die Entwicklungszeit verkürzen.
- Testen: Die einzelnen Teile können getrennt voneinander getestet werden. Die Fehlersuche ist damit bedeutend effizienter.
- Widerverwendbarkeit: eine Software-Library, ein VHDL-Block, ein Design kann vielleicht in einem anderen Problem wieder verwendet werden.

5.) Nenne 4 Vorteile eines Schichtenmodells.

- Die einzelnen Schichten können getrennt implementiert und getestet werden
- Man kann Schichten austauschen
- Protokolle werden vergleichbarer
- Strukturierter Aufbau, besser für Debuggen, Erweiterungen, Updates,...

6.) Nenne 3 Vorgänge, bei denen man Leistungsbewertung benötigt

- Auswahl neuer Computer
- Entwurf neuer Systeme
- Speicherzugriffszeit

7.) 4 Möglichkeiten der Leistungsbeurteilung

- Auswertung von Hardwaremaßzahlen (Taktrate, MIPS, MOPS/ FLOPS, CPI)
- Messungen der Laufzeit von Programmen (Benchmarks)
- Messungen während des Betriebs (Software-, Hardware-, Hybridmonitore)
- Modelltheoretische Verfahren (Simulation, Analytische Verfahren)

8.) Nenne 4 Beispiele für skalare Hardwaremaßzahlen der Leistungsbewertung

- Taktfrequenz
- Befehlsausführungszeit
- Speicherzugriffszeit
- Speicherzugriffsbandbreite

9.) Was gibt die MIPS Zahl an und warum ist sie relativ nutzlos?

$$MIPS = \frac{\text{Anzahl der Instuktionen}}{\text{Ausführungszeit} \cdot 10^6}$$

- ➔ Problem: unterschiedliche Befehlssätze zweier Prozessoren machen Vergleich unmöglich
- ➔ MIPS eher Marketing Werkzeug

10.) Warum sind Kernprogramme und Mixe heutzutage nicht mehr sehr aussagekräftig?

Mixe:

Es wird versucht, mittlere Operationszeit T zu bestimmen. Dazu werden die unterschiedlichen Befehlstypen nach der erwarteten relativen Häufigkeit ihres Auftretens bewertet und summiert.

$$T = \sum_{i=1}^n t_i \cdot p_i$$

n ...Anzahl der gesamt betrachteten Befehle
 t_i ...Operationszeit des Befehls i
 p_i ...relative Häufigkeit des Auftretens von i

Kernprogramme:

Sind Typische Anwenderprogramme, die für den zu bewertenden Rechner geschrieben werden. Sie werden jedoch nicht zur Ausführung gebracht.

Problem: Komplex, moderne (parallele) Abläufe nicht berücksichtigt

Darum sind Kernprogramme, aber auch Mixe bei modernen Prozessorarchitekturen nur bedingt sinnvoll anwendbar.

11.) Was ist der Unterschied zwischen Kernprogrammen und synthetischen Programmen bei der Leistungsbeurteilung?

Kernprogramme:

Sind Typische Anwenderprogramme, die für den zu bewertenden Rechner geschrieben werden. Sie werden jedoch nicht zur Ausführung gebracht.

Problem: Komplex, moderne (parallele) Abläufe nicht berücksichtigt

Synthetische Programme:

Synthetische Programme sind eine willkürliche Mischung von Operationen, die nicht unbedingt in funktionalem Zusammenhang stehen. Sie werden nur zum Zweck der Rechnerbewertung erstellt und zum Laufen gebracht. SP bieten aber auch einen Vorteil. Da sie keine produktiven Ergebnisse liefern müssen, eignen sie sich gut zur Erzeugung von Lasten beliebiger Ausprägung und damit zum Beleuchten ganz bestimmter Eigenschaften der getesteten Maschine.

12.) Was sind Benchmarks?

Ein Benchmark ist ein Programm, das ein Anforderungsprofil repräsentiert und zum Vergleich und Analyse der Leistungsfähigkeit von Rechensystem verwendet wird.

Beispiele für Anforderungsprofile: Datenbanken, wissenschaftliche Berechnungen, Server Dienste. Man unterscheidet:

➤ Klassische Benchmarks

- *) Dhrystone für Integer
- *) Whetstones für Floating Point

➤ Spezialbenchmarks

- *) CPU-Leistung
- *) Ein/Ausgabe-Verhalten
- *) Speicherzugriffsverhalten
- *) Grafikfähigkeit

➤ Synthetische Benchmarks

- *) Künstliche Programmfragmente um dediziert Performance zu messen (ohne dass das Programm einen Sinn ergibt)

13.) Was ist neben der Performance des Computersystems in der Maßzahl eines Benchmarktests noch enthalten?

Benchmarks messen nicht nur die Hardware-Leistung eines Computers. Sie messen implizit auch Compiler, Softwarestruktur, Codeoptimierung, etc.

14.) Unterschied zwischen Software Monitor und Hardware Monitor?

- Softwaremonitore: Sind spezielle Messprogramme, die in die Software des zu beobachtenden Systems eingebettet sind.
- Hardwaremonitore: Sind elektronische Messinstrumente, die mit Messfühlern an internen Kontaktpunkten Zustände des Systems untersuchen.

15.) Unterschied zwischen Sampling und Eventing?

- Sampling: Beim sampling wird eine Messung jeweils nach einem zeitlich festgelegtem Abtastintervall D gestartet.
- Eventing: Beim eventing wird eine Messung durch das Auftreten eines ganz bestimmten Ereignisses ausgelöst. Also wenn ein Programm in eine festgelegte Stelle im Code läuft.

16.) Was versteht man unter Modelltheoretische Verfahren?

Das experimentelle Bestimmen der Leistungsfähigkeit (Benchmarks) ist sehr aufwändig. Es ist nur bei existierenden Maschinen machbar → Ausweg: Modellierung Simulation/Analyse

Anforderungen:

- Klarheit und Übersichtlichkeit
- (Einfache) Anpassungs- und Erweiterungsfähigkeit des Modells
- Gültigkeit des Modells
- Formale und empirische Richtigkeit

Vorteil: Leichte Handhabung, billig, beliebig grob oder fein machbar, schnell modifizierbar, alles beobachtbar

Werkzeuge: Verkehrstheorie, Statistik, etc.

Ziel:

- Aufdecken von Beziehung zwischen Systemparametern
- Erkennen von Engpässen
- Ermitteln von Leistungsgrößen
- Formale und empirische Richtigkeit

Kapitel 2 – Prozessorarchitektur:

17.) Charakteristik der „von-Neumann“ Architektur

- Der Rechner besteht aus den Komponenten: Zentraleinheit CPU, Speicher und Ein/Ausgabe
- Der Aufbau eines Rechners ist unabhängig vom zu bearbeitenden Problem
Programme und Daten befinden sich im gleichen Speicher
- Der Speicher ist in gleich große Zellen aufgeteilt, die fortlaufend nummeriert sind
- Programme sind lineare Sequenzen von einzelnen Befehlen
- Im Steuerwerk gibt es einen Befehlszähler, der immer auf den nächsten auszuführenden Befehl zeigt
- Aufeinander folgende Befehle befinden sich in aufeinander folgenden Speicherzellen (Ausnahme: Sprungbefehle)

18.) Was ist eine „semantische Lücke“, wie hängt sie mit der „von-Neumann“ Architektur zusammen und welche beiden Aspekte hat diese Lücke?

Der „von-Neumann“-Rechner ist nicht in der Lage, die von der auf ihm implementierten höheren Programmiersprache verwendeten Typen-Attribute für Datenobjekte zu erkennen und zu verarbeiten. Dieser Sachverhalt wird als Semantische Lücke bezeichnet. Im Speicher an sich nicht mehr zu unterscheiden. Ein falscher Sprung, und Daten werden als Programm ausgeführt. Keine Unterstützung für Hochsprachen. Jede C-Variable, jedes Java Objekt wird unkenntlich gemacht → Bitketten. Ein von Neumann-Rechner speichert Daten und Befehle in ein und derselben Art ab: als Bitketten. Wenn hier Dinge durcheinander geraten, kann ein Prozessor mitunter Daten als Befehle missverstehen und umgekehrt.

19.) Was ist das Minimalitätsprinzip in der Mikrocomputerarchitektur?

Ein „von-Neumann“-Rechner zeichnet sich in seiner Hardware-Struktur durch einen minimalen Hardware-Aufwand aus. Die Vorgehensweise, nur das absolut Nötigste einzubauen (sodass das Weglassen auch nur eines Bauteils oder einer Funktionseinheit die Gesamtfunktion zerstört) wird MP genannt. Eine nicht-minimale von-Neumann-CPU ist schneller. Komplexität wird in die Zeit verlagert.

20.) Was bedeutet „von Neumann“ Flaschenhals

Der Von-Neumann-Flaschenhals der Von-Neumann-Architektur bezeichnet den Sachverhalt, dass das Verbindungssystem (Daten- und Befehls-Bus) zum Engpass zwischen dem Prozessor und dem Speicher wird. Da im Gegensatz zur Harvard-Architektur nur ein gemeinsamer Bus für Daten und Befehle genutzt wird, müssen sich diese die maximal übertragbare Datenmenge aufteilen.

Hauptspeicher und Bussystem bestimmen somit die Arbeitsgeschwindigkeit des gesamten Systems und bilden den Von-Neumann-Flaschenhals. In der Praxis versucht man, diesen Effekt durch die Nutzung von Datencaches abzuschwächen.

21.) Grundkomponenten von Neumann, Unterschied zu Harvard

Folgende Komponenten sind in der von Neumann Architektur enthalten:

- Prozessor (CPU): Besteht aus Steuer- und Rechenwerk und übernimmt die Ablaufsteuerung der Befehle.
- Hauptspeicher: Dient zur Ablage von Daten und Programmen. Er besteht aus Speicherzellen fester Wortlänge, die über Adressen angesprochen werden.
- I/O System: Schnittstelle nach außen
- Bus: Über den Bus werden die Speicherwörter zwischen Prozessor und Speicher, als auch Prozessor und I/O Einheit ausgetauscht.
- Bei der Harvard-Architektur stehen Code und Daten nicht im selben Speicher, sondern in zwei getrennten Speichern. Damit vermeidet man den „Flaschenhals“.

22.) Unterschied zwischen Rechenwerk und Steuerwerk

Rechenwerk:

Das Rechenwerk übernimmt alle Berechnungen, auch Adressberechnungen. Die klassische ALU kann: - addieren, subtrahieren, AND, OR, XOR. Die moderne CPU besitzt üblicherweise diese Rechenwerke:

- Integer Unit (IU)
- Floating Point Unit (FPU)

Die FPU kann auch weggelassen werden (bei Microcontrollern der Fall), dann müssen Fließkommaoperationen mit der Integer Unit mühsam emuliert werden. Es gibt auch einige Spezialrechenwerke wie z.B.:

- Checksummenberechnung
- Matrizenberechnung
- FFT, Z-Transformationen

Steuerwerk:

Das Steuerwerk ist die Schaltzentrale der CPU. Decodierte Befehle geben dem Steuerwerk die Anweisung, was es zu tun hat. Folgende Aufgaben hat das Steuerwerk:

- Lesezugriff auf den externen Speicher
- Schreibzugriff auf den externen Speicher
- Aktivierung von ALU-Funktionen
- Kanalisieren und Transportieren von Daten von A nach B

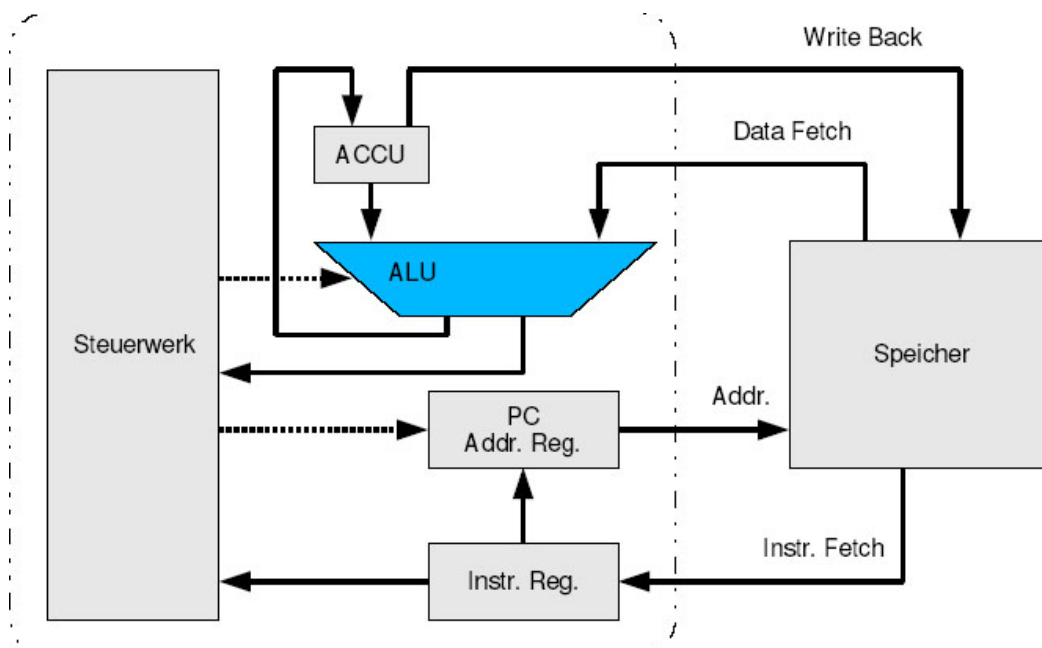
Diese Aufgaben werden in 2 Bereiche gruppiert:

- Abarbeiten des Programms: FETCH, DECODE, LOAD/STORE etc.
- Abarbeiten der EXE Phasen der Befehle (Rechenwerke steuern, etc.)

Die Abläufe der Steuerung können auf zwei Arten implementiert werden:

- Hard wired: Übergangsgraphen des Automaten werden hardwaremäßig in Gatter und Flip-Flops umgerechnet. Sie ist die schnellere Methode.
- Mikroprogramm: Es werden ein digitaler Zähler, Speicher und Logikmatrizen verwendet und der Automat somit strukturiert.

23.) Aufzeichnen eines klassischen CPU-Blockschaltbildes



24.) Was kann mit dem Program Counter gemacht werden?

Das Adressregister bzw. Program Counter kann rechnen. Um den sequenziellen Programmfluss im Speicher auszulesen muss es nach oben zählen können, für Sprungbefehle und Verzweigungen muss es addieren und subtrahieren können.

25.) Nenne 4 CPU Architekturen nach dem Registerzugriff

Stack Maschine

Accu Maschine

Register-Memory Maschine

Register Maschine (LOAD/STORE Architektur)

26.) Erklären sie die Befehle MOV R4,R6 ; ADD R4,R2,R4 ; BREQ label

MOVE Kopieren eines Registerinhaltes in ein anderes. Es wird der Inhalt des Registers R6 in das Register R4 kopiert. ADD R4, R2, R4: BRANCH Sprung zu einer Programmstelle, wenn eine gewisse Bedingung erfüllt ist z.B.: branch if equal BREQ

27.) Was ist Register-Indirekte-Adressierung und nennen sie ein Beispiel dafür.

Bei der Register-Indirekten-Adressierung steht die Zieladresse in einem Register, welches bekannt gegeben wird. Dies wird bei der einstufigen Adressierung verwendet.

28.) Unterschied Register Memory-Maschine / Register-Maschine.

Register Memory Machine: Es kann flexibel auf den Speicher zugegriffen werden.

Register Machine: Nur LOAD und STORE – Befehle können auf den Speicher zugreifen.

Die Begriffe Register-Memory-Machine und Register-Machine beziehen sich auf den Umgang der Register und werden zur Klassifizierung der Architektur herangezogen. Andere Möglichkeiten sind Stack Machine, Accu Machine und Memory-Memory-Machine.

29.) K4. Unterschied Mikrobefehl, Makrobefehl und Beziehung der beiden zueinander

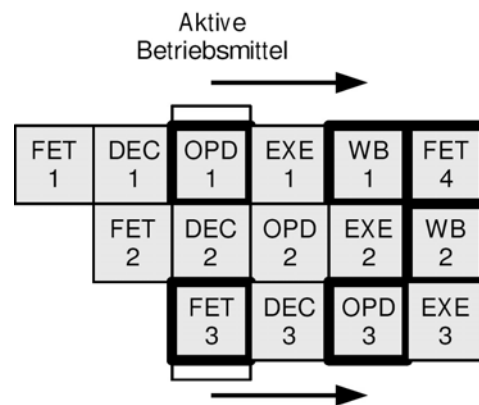
Ein Makrobefehl (d.h. normaler Maschinenbefehl) besteht aus einer Sequenz von 4 Mikrobefehlen, die wiederum aus einer Reihe von Pikobefehlen bestehen. Mikrobefehle können ihre Funktion auch in vektorisierter Form, d.h. Nanobefehlen tragen. Wenn ein Makrobefehl ausgeführt werden soll, wird das zugehörige Mikroprogramm im Mikroprogramm Speicher gesucht und ausgeführt.

30.) Was ist ein Pikobefehl?

Bei horizontalen Mikroprogrammen ist das Mikro-Codewort sehr breit und besteht aus einzelnen Bits, von denen jedes einzelne direkt einer Steuerleitung des Steuerwerks entspricht. Diese einzelnen Bits werden auch als Pikoprogramm-Befehle oder kurz Pikobefehle bezeichnet.

31.) Was ist horizontale Programmierung bzw. was ist vertikale Programmierung?

Mikroprogramme werden klassischerweise in der CISC Architektur verwendet, fest verdrahtete Steuerwerke in RISC. Sehr breiter Mikrocode (auch als „horizontale Verlagerung“ bezeichnet) sieht aus wie VLIW, sehr schmaler (auch als „vertikale Verlagerung“ bezeichnet) sieht eher aus wie RISC. Bei horizontalen Mikroprogrammen ist das Mikro-Codewort sehr breit und besteht aus einzelnen Bits, von denen jedes einzelne direkt einer Steuerleitung des Steuerwerks entspricht. Diese einzelnen Bits werden auch als Pikoprogramm - Befehle oder kurz Pikobefehle bezeichnet. Bei vertikalen Mikroprogrammen beinhaltet jedes Mikroprogrammwort lediglich einen Pikobefehl, ein Makroprogrammwort besteht in diesem Fall also aus einer mitunter sehr langen Mikroprogrammwortkette. Der Aufbau ist einfach, die Worte sind schmal, aber die Abarbeitung dauert länger.

32.) Zeichne eine 3 Stage Pipeline mit Fetch, Decode, Execute, Writeback (bis F5)**33.) 3 Stufige Pipeline mit 4 Zyklen (FE , DE , EX, WB) → wann DE 5 ?**

In der siebten Phase

	F1	D1	E1	W1	F4	D4	E4	W4		
		F2	D2	E2	W2	F5	D5	E5	W5	
			F3	D3	E3	W3				

F...Fetch D...Decode E...Execute W...Write Back

34.) Unterschied zwischen static und dynamic scheduling beim Auflösen von Pipeline Steuerkonflikten

- static scheduling: Software-Lösung → Der Compiler sortiert Befehle um.
- dynamic scheduling: Hardware-Lösung → Der Prozessor lädt immer Blöcke von Befehlen auf einmal und sortiert diese um.

35.) Was versteht man unter Datenabhängigkeit bei Pipeline Konflikten?

Ein Befehl benötigt das Ergebnis eines anderen Befehls, dessen Abarbeitung noch nicht abgeschlossen ist.

36.) Was versteht man unter Befehlsabhängigkeit bei Pipeline Konflikten?

Die Befehlsabhängigkeit kann bei Pipelines zu einem Datenkonflikt bzw. Steuerkonflikt führen. Dabei benötigt ein Befehl noch ein Ergebnis, welches aber noch nicht zur Verfügung steht.

37.) Wann muss eine Pipeline geflusht werden?

- Sprungbefehle, deren Ziel nicht bekannt war,
- Verzweigungsbefehle, deren Verzweigungsargument nicht bekannt war, und
- Interrupts, welche naturgemäß unvorhersehbar sind.

38.) Vorteile und Nachteile von Pipelining

- + viele Abläufe können unterteilt werden -> schneller
- + Ausnutzung aller Ressourcen
- + Achten auf Datenkonsistenz (Data, Ressource, Control hazards)
- NOPs müssen eingefügt werden -> es kann zu einer nicht effektiven Nützung kommen.

39.) Unterschied zwischen Datenkonflikt und Steuerflusskonflikt bei Pipeline

Datenkonflikt: Ein Befehl benötigt das Ergebnis eines anderen Befehls, dessen Abarbeitung noch nicht abgeschlossen ist. Diese Konflikte werden in 3 Gruppen geteilt:

1. Read after write: Dieser Fall ist der häufigste.
2. Write after read: Ein Ergebnis wird in ein Register geschrieben, obwohl der alte Registerwert noch benötigt wird.
3. Write after write: Tritt auf, wenn mehrere Pipeline-Stufen schreiben können.

Der Datenkonflikt kann mit Software, als auch mit Hardware gelöst werden. Bei der Softwarelösung muss der Compiler basierend auf dem gegebenen Pipelinesystem die Befehle so anordnen, dass wenige Konflikte entstehen (instruction scheduling). Hardwaremäßig kann das Problem durch

- interlocking (Pipeline anhalten, NOP-Befehle einfügen)
- forwarding (das erwartete Resultat wird direkt zum Befehl geschickt und nicht vorher in Registern gespeichert) gelöst werden.

Steuerflusskonflikt: Ein Sprung kann nicht durchgeführt werden, weil die Zieladresse noch in der Pipeline ist. In solchen Fällen muss der Prozessor durch interlocking warten, bis die Berechnungen durchgeführt wurden. Lösungen für diese Konflikte sind:

Softwaremäßig:

- static scheduling: Compiler ordnet Befehle so an, dass kein Konflikt entsteht.
- delayed branch: Leerbefehle werden eingefügt.

Hardwaremäßig:

- Pipeline-Leerlauf: HW-Version des delayed-branch
- dynamic scheduling: Prozessor sortiert Befehle, nicht der Compiler
- branch prediction: Daten für nächsten wahrscheinlichen Befehl laden
- Spekulative Ausführung: Alle möglichen Zweige werden ausgeführt

40.) Lösungen für Pipeline-Ressourcenkonflikte

Ressourcenkonflikt: Zwei Pipeline-Stufen benötigen dieselbe Ressource (den Bus, einen Multiplizierer, etc.)

- Interlocking (kurze Pause)
- Schnellere Ressource (hat z.B. höheren Takt, sodass sie schon fertig ist, wenn die zweite Pipelinestufe zugreift bzw. Jobs [hintereinander] in einer Phase ausführt).
- Ressourcenduplizierung. Zweifellos die bereits beste Methode, wenn auch die aufwändigste.

41.) Wie wirkt sich ein "cache miss" auf die Pipelineverarbeitung aus?

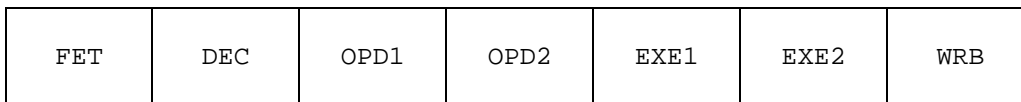
Hat keine Wirkung, da der CPU den Cache gar nicht bemerkt, von ihm aus sieht es bei einem Cache-hit, dass der Speicher manchmal schneller arbeitet, sonst ist der Cache völlig transparent.

42.) Geben sie die verschiedenen Möglichkeiten der Pipelineverarbeitung bei Signalprozessoren an!

- Pipeline Control by Interlocking
Pipeline-Verarbeitung der Befehle bzw. Konflikte sind für den Assemblerprogrammieren nicht sichtbar. Rechenergebnisse und Speicheroperationen sind in einem einzigen Befehl abgeschlossen. Dafür werden die Befehle automatisch hardwaremäßig verzögert.
- Time-Stationary Pipeline Control
Programmierer hat Zugriff auf alle Phasen des Pipelining und muss angeben welche Operationen überlappend durchgeführt werden sollen. Im Befehlsword ist dabei für jede Operation, sie simultan in einem Befehlszyklus durchgeführt wird, ein eigenes Feld vorgesehen.
- Data-Stationary Pipeline Control
Ein Befehl gibt an, was im Zuge der Befehlsverarbeitung in der Pipeline mit den Daten passiert. Daten und Befehle bilden Paare die gemeinsam durch die Pipeline transportiert werden.

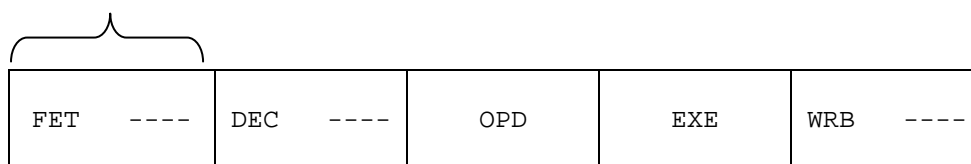
43.) Pipeline mit FETch, DECode, OPerandenDecode, EXEcute und WriteBack, wobei FET DEC und WB doppelt so schnell fertig sind wie die anderen, sprich nur die Hälfte der Zeit benötigen. Was kann man tun um die Pipeline voll auszunutzen und wieviele Phasen hat man dann?

Takt verdoppeln, OPD und EXE in OPD1/OPD2 und EXE1/EXE2 aufspalten, man hat dann 7 Phasen AMD und damit deutlich weniger Strom benötigt beziehungsweise deutlich weniger Abwärme produziert.



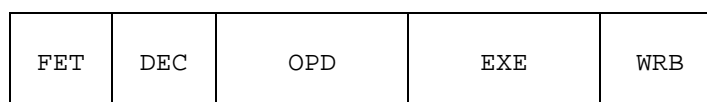
44.) Es gibt FETch, DECode, OPerandenDecode, EXEcute und WriteBack. OPD und EXE sind doppelt so lang wie die anderen Phasen. Wie kann man den Taktzyklus Ändern, dass man keinen Leerlauf hat und wieviele Phasen hat man dann?

1 Takt



---- ... Hälfte der Zeit ist ungenützt!

→ Takt halbieren → Zeit in FET, DEC und WRB einsparen (1 1/2 Takte)



→ man hat erneut 5 Phasen

45.) Branch prediction, spekulative Ausführung definieren, Unterschiede erklären

Branch prediction: Der Prozessor sagt voraus, welcher Befehl als nächstes ausgeführt wird. Dies kann einfach durch Vorgabe des Programmierers geschehen, oder der Prozessor besitzt eine Statistik, aus der er die Wahrscheinlichkeit, dass ein bestimmter Zweig genommen wird, auslesen kann (dynamic branch prediction).

Spekulative Ausführung: Im Gegensatz zur Branch prediction werden alle Zweige ausgeführt. Erfährt der Prozessor, welcher Zweig der Richtige war, verwirft er alle anderen Berechnungen (für die andern Zweige) und gibt die verwendeten Register wieder frei.

46.) Warum haben neuronale Netze keinen „von Neumannschen“ Flaschenhals?

Wie der Name schon vermuten lässt sind die einzelnen „Funktionseinheiten“ neuronaler Netze netzförmig miteinander verbunden und müssen sich nicht wie bei einer Neumannschen Architektur einen Bus teilen.

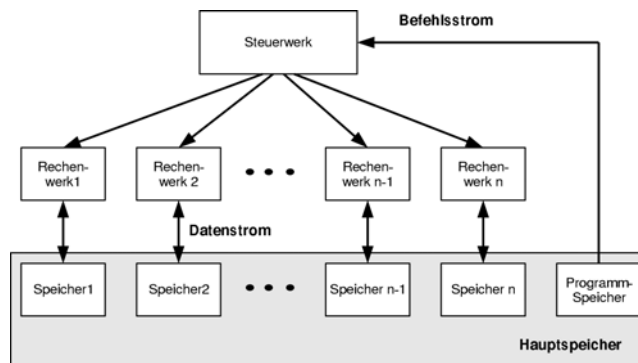
47.) Was ist SIMD?

Rechnerklassifikation nach Flynn. Die Rechner werden in 4 Klassen, die sich durch die Anzahl der gleichzeitig vorhandenen Instruktions- und Datenströme unterscheiden, unterteilt. SIMD: single instruction stream – multiple data stream

48.) Zeichnen sie den Informationsfluss bei SIMD und bei MIMD

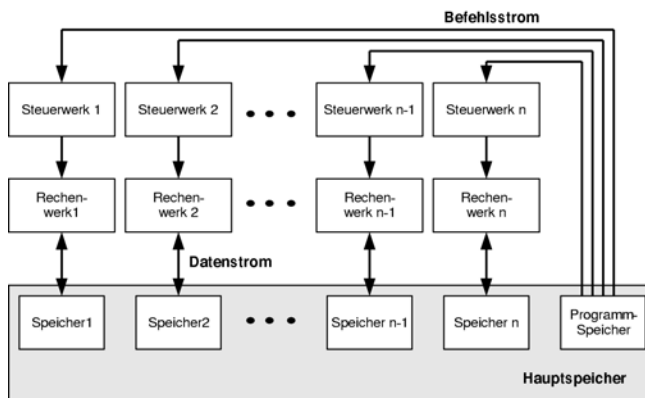
SIMD: Single instruction stream, multiple data stream

Bei SIMD gibt es nur ein Steuerwerk, wobei der entschlüsselte Befehl gleichzeitig auf mehrere Operanden in mehreren Rechenwerken angewandt werden kann.



MIMD: Multiple instruction stream, multiple data stream

Bei MIMD gibt es keine zentrale Steuerungsinstanz. Es gibt mehrere Steuerwerke und Rechenwerke.



49.) Was versteht man unter SIMD-Erweiterungen (& Anwendungsbeispiele)?

Eine sehr verbreitete Erweiterung der Rechenfähigkeit einer CPU sind Betriebsmittel für SIMD Befehle. Besonders Multimedia-Anwendungen (und hier im speziellen Video - Rendering) hat es erforderlich gemacht, dass „normale“ Universal-CPU's und Desktop-CPU's spezialisierte Betriebsmittel bekommen:

- MMX (Multi-Media Extension, 64-Bit Integer SIMD, 57 neue Befehle)
- 3DNow! (von AMD, 32-Bit Integer/Float SIMD a.k.a. single precision, 21 neue Befehle)
- SSE (Streaming SIMD Extensions im Katmai Kern von Intel [Pentium III], ebenso 32- Bit Integer/Float SIMD, 70 neue Befehle)
- SSE2 (64-Bit double precision float SIMD)
- SSE3 (im Intel Prescott Core sowie im AMD Athlon64, 13 neue Befehle im Vergleich zu SSE2)
- AltiVec (die Integer/Float SIMD Erweiterung im Power Chip, so genannt von Motorola. Um die Verwirrung zu steigern, nennt IBM diese Technologie VMX, Apple hingegen nennt sie Velocity Engine)

50.) Jeweils zwei Beispiele für SIMD und MIMD → nur jeweilige Architektur angeben

SIMD → VLIW, Vektorrechner, MMX-Einheit

MIMD → Transputer, Sega Spielekonsole, Cray T3D, Blue Gene

51.) Was macht der Compiler bei VLIW?

Bei VLIW (Very Large Instruction Word)- Prozessoren fügt der Compiler die einzelnen Befehle zu langen Befehlswörtern zusammen und optimiert diese. Dabei achtet der Compiler darauf, dass die Befehle so angeordnet werden, dass der Prozessor parallel ausführbare Pakete erhält.

52.) Unterschiede zwischen Superskalar und VLIW

Bei superskalaren Prozessoren übernimmt der Prozessor selbst das Umsortieren, Anordnen und Optimieren der Befehlskette. Bei superskalaren DSPs erfolgt parallele Verarbeitung mit vorgegeben Mehrfach-Arithmetik- Befehlen. Bei VLIW übernimmt der Compiler diese Aufgaben.

53.) Vergleiche Vektor- und Superskalarprinzip

<u>Vektorprinzip</u>	<u>Superskalarprinzip</u>
<ul style="list-style-type: none"> → Implementieren lediglich SIMD → Mehrere Rechenwerke, aber nur ein Steuerwerk. → Entsprechender Befehl kann gleichzeitig auf mehrere Operanden in mehreren Rechenwerken angewendet werden. 	<ul style="list-style-type: none"> - mehrere Befehle parallel ausgeführt - Ressourcen mehrfach ausgeführt (MIMD) - Prozessor entscheidet während Betrieb, was parallel ausgeführt wird und was nicht. - Hardware komplexer

55.) Was ist "out-of-order-execution" und wann kann man sie nicht benutzen?

Out-of-order-execution ist die Möglichkeit, Befehle in den Ausführungseinheiten eines superskalaren Prozessors außerhalb der Programmreihenfolge auszuführen, mit dem Ziel, die Pipelines möglichst gut auszulasten. Das wird mittels eines Scoreboards durchgeführt, welches sich merkt, welche Register fertig sind und welche Register noch warten. Man kann out-of-order- execution nur bei Befehlsfolgen verwenden, die nicht voneinander abhängig sind.

56.) Wie kann der Transmeta Crusoe x86 Befehle ausführen, welche Schritte sind dazu nötig?

Eine Software-Lösung namens Code-Morphing emuliert dabei in Echtzeit eine x86-CPU. Da Emulationen normalerweise recht ineffizient sind, optimiert die Transmeta-Software während der Laufzeit ständig die Emulation der ablaufenden Programme. Der Prozessor selber ist ein 128-Bit-VLIW-Prozessor, der deutlich einfacher aufgebaut ist als eine gewöhnliche x86-CPU von Intel.

57.) Nenne 4 Charakteristika von RISC?

- „Wenige“, aber dafür schnelle Befehle
- Die häufigsten Befehle sind implementiert, weniger häufige nicht
- Steuerwerk in Hardware
- nur LOAD und der STORE Befehl greift auf den Speicher zu. Alle anderen Befehle müssen mit Registern auskommen
- „Große“ Anzahl an Registern, bzw. ganze Registerbänke
- Verwendung von Pipelines
- Einheitliche Länge der Befehle
- Schlaue Cache-Verwaltung um Speicherzugriffe zu minimieren
- Parallelisierung diverser Ressourcen

58.) Hauptvorteile eines Intel 80x86 gegenüber dem Motorola 68060?

Intel 80x86 ist nach außen hin ein CISC Prozessor und intern werden die Befehle in RISC ähnliche Befehle zerlegt und im RISC Kern abgearbeitet. Motorola ist ein CISC Prozessor (CISC Prozessoren haben typischerweise ein µProgramm und benötigen viele Taktzyklen zum Abarbeiten eines Befehls. Intel setzt auf Kompatibilität der Prozessoren (DOS läuft auch noch auf aktuellen Prozessoren), Motorola nicht.

59.) Wie funktioniert das Register-Windowing bei RISC Prozessoren?

Registerfenster sollen das Lokalhalten von Daten unterstützen. Typische RISC Prozessoren wie die Berkeley RISC besitzen weit über 100 Register, von denen aber immer nur 32 für sichtbar sind:

R0...R9 globale Register
 R10...R15 Ausgaberegister
 R16...R25 lokale Register
 R26...R31 Eingaberegister

Die Idee ist nun, dass die ersten 10 Register von allen Prozeduren gesehen werden. Die Restlichen von R10 bis R31 sind jeweils nur einer Prozedur zugeordnet. Falls nun eine Prozedur eine andere aufruft, wird nur das "Fenster" auf einen freien Registerbereich umgeschaltet. So müssen die Register nicht neu aus dem Speicher geladen werden und es wird dadurch viel Zeit gespart. Normalerweise überlappen sich die einzelnen Fenster um einige Register, um somit gleich eine effiziente Möglichkeit der Parameterweitergabe zu bieten. Wenn alle Registerfenster voll sind → wird das Register als Ringregister organisiert. Sind alle Registerfenster voll, wird das Ältteste in den Speicher ausgelagert, was von so genannten Trap-Routinen erledigt wird.

60.) Formel für die Angabe des Geschwindigkeitsvorteils bei Parallelisierung (speedup)

Das Gesetz von Andahl gibt an, wie schnell sich ein Gesamtsystem verbessert, wenn man nur einen Teil dieses Systems verbessert. Wird die Parallelisierung eines Code-Teils als Verbesserung angesehen, so lautet die Formel:

$$S = \frac{N}{B \cdot N + (1 - B)}$$

S gibt hierbei den Speedup an, N die Anzahl der Prozessoren und B ist ein Faktor der den nicht parallelisierbaren Teil des Codes repräsentieren soll. Wäre alles vollständig parallelisierbar (B = 0) wäre S = N, d.h. der Geschwindigkeitsgewinn wäre ausschließlich von der Anzahl der Prozessoren abhängig.

61.) Wie funktioniert eine μProgrammsteuerung?

Ein μProgramm (= „softwaremäßige Implementierung des Steuerwerks“) kann hier mit mehr Möglichkeiten dienen. Ein flexibler, universeller, einfacher Zähler (Mikroprogramm Counter) spielt ein Programm (das Mikroprogramm), welches in einem Speicher (dem Mikroprogrammspeicher, on-chip) steht und das aus kleinsten Kommandos (den Mikrobefehlen) besteht, ab.

62.) Ein Vorteil und ein Nachteil einer μProgrammsteuerung?

- + Ein Mikroprogramm kann man leicht ändern, um Fehler auszubessern oder neue, notwendige Befehle hinzuzufügen.
- + Mikroprogramm ist übersichtlicher, flexibler und bietet mehr Möglichkeiten
- Mikroprogramm → geringe Geschwindigkeit

63.) Bei welchen Speicherarten ist Interleaving möglich?

DRAM
 Harddisk
 optische Laufwerke (CD, DVD)

64.) Wie viele Zeiger braucht FIFO und LIFO?

FIFO benötigt zwei Zeiger, LIFO einen.

65.) Welcher Prozessor ist für Server besser geeignet, einer mit feiner oder grober Parallelisierung?

Weil bei Serveranwendungen hauptsächlich load-balancing und Latenzzeit wichtig und die Anfragen der Clients meist dieselben sind, ist grobe Parallelisierung besser für Server geeignet (mehrere komplette Rechnersysteme)

66.) Ein Big Endian Rechner speichert 0xABCD1234. Was liest ein Little Endian an dieser Stelle?

0x3412CDAB

67.) Vier typische externe Steuerepins eines m-P, plus ihre Bedeutung und Verwendung ? (außer HALT, RESET, CLK)

- AS Address Stable
- DS Data Stable
- BERR Bus Error
- IL Interruptlevel

68.) Vorteile und Nachteile von Festkommazahlendarstellung

- + Einfache zu implementieren
- Nur kleine Zahl (2^{Bit} Zahl)

69.) 4 Architekturelle / Technologische Methoden, um die Verlustleistung von μChips zu verringern?

- Serienschaltungen von Transistoren reduzieren den Leckstrom um den Faktor 5-10
- Funktionsblöcke können mit Transistoren schnell deaktiviert und isoliert werden, wenn sie nicht benötigt werden.
- Man kann den gesamten Transistor negativ aufladen. Er wird langsamer, aber auch sparsamer. Bei Bedarf kann er wieder in den „fast mode“ gebracht werden.
- Bessere Isolierungen können die Tunnelströme verringern.
- Niedrigere Spannungen (auch nur für einzelne teile des Chips und auch abhängig von der Last der CPU) reduzieren die Verlustleistung und die Leckströme.
- Multithreading Prozessoren und intelligentes Caching sparen Energie, weil keine Größeren unproduktiven Wartezeiten entstehen.

70.) Gleitkomma Single Precision Zahl IEEE 754, Aufbau

Single Precision Float ist eine 32 Bit Zahl. Nach dem IEEE-Standard werden Gleitkommazahlen einfacher Genauigkeit (Floats) als 1 Bit Vorzeichen, 8 Bit Exponent und 23 Bit Mantisse gespeichert.

$$(-1)^s \cdot 1. \text{fffffffffffffffffffffff} \cdot 2^{(\text{exp}-127)}$$

Vorzeichen ist einfach: es ist negativ, also gesetzt(1)

32 Bit floats entsprechen 6-7 stelliger Genauigkeit in Dezimalsystem

Double precision floats entsprechen 15-16 Stellen

71.) Gleitkomma Double Precision Zahl IEEE 754, Aufbau

Double Precision Float ist eine 64 Bit Zahl. Nach dem IEEE-Standard werden Gleitkommazahlen einfacher Genauigkeit (Floats) als 1 Bit Vorzeichen, 11 Bit Exponent und 52 Bit Mantisse gespeichert.

Vorzeichen ist einfach: es ist negativ, also gesetzt(1)

Double precision floats entsprechen 15-16 Stellen Genauigkeit in Dezimalsystem

72.) Nenne 4 Vorteile und Nachteile von Gleitkommadarstellung

- + Größere Zahlendynamik und Präzision
- + Besser für den Einsatz von C-Compilern
- + Genauigkeit konstant
- + Kaum mehr Skalierungsoperationen
- Nicht jede beliebige Bitkombination ist eine gültige Zahl
- Variablen müssen immer initialisiert werden
- kein einheitliches Zahlenformat
- höherer Hardwareaufwand im Gegensatz zur Festkommadarstellung

73.) Welche Eigenschaften hat die Zweierkomplementdarstellung

MSB: Vorzeichen (1: neg. , 0: pos.)

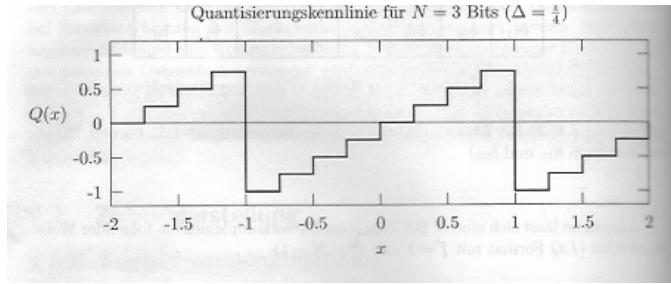
Das Vorzeichen einer Zahl wird geändert, indem alle n Bit bitweise invertiert werden und anschließend 1 addiert wird

Nur eine Darstellung für die Zahl 0

Wertebereich mit n-Bits - $(2^{(n-1)} \dots (2^{(n-1)} - 1)$

74.) Darstellung des Überlaufverhaltens von Festkommazahlen im Zweierkomplement!

Bei Überlauf auf die größte positive Zahl erfolgt die betragsmäßig größte negative Zahl und umgekehrt.



75.) Nenne 4 typische ALU-Flags

- C- (Carry-) Bit: Übertrag, die (n+1)-te Stelle des Ergebniswortes
- V- (Overflow-) Bit: Die Berechnung ist aus dem Zahlenbereich geglitten
- Z- (Zero-) Bit: das Ergebnis ist null
- N- (Negative-) Bit: das Vorzeichen des Ergebnisses
- H- (Half Carry-) Bit: Übertrag zwischen den beiden Nibbles, wird für BCD (binary coded decimal) Rechnungen benötigt
- P- (Parity-) Bit: Parität, ein einfachen Fehlererkennungscode

76.) Erklären sie Carry- und Zero-Flag.

C-Flag: Das Übertragsflag wird gesetzt z.B. bei Addition von Binärzahlen:

$$\begin{array}{r}
 1101 \\
 + 0001 \\
 \hline
 1110
 \end{array}$$

1 – Übertrag = Carry

Z-Flag: Ist das Ergebnis einer Berechnung 0, so wird das Zero-Flag Gesetzt.

Kapitel 3 – Prozessortypen:

77.) Unterschied zwischen µP und µC?

µProzessor: Ein µP besteht aus Steuerwerk, Rechenwerk, Schnittstelle zum Systembus, on-Chip Cache Speicher und der Speicherverwaltungseinheit.

µComputer: hat alles „on-board“ und besteht aus einem µProzessor (CPU), Speicher und zusätzlicher Peripherie (I/O Controller, AD/DA Wandler, Timer, PWM Units, Watchdog, ...)

78.) Welche hierarchischen Ebenen gibt es beim Speicher eines µC und warum?

- Register (sehr schneller Speicher)
- interner Arbeitsspeicher (RAM) zur Programmausführung (meist als SRAM realisiert)
- interner Festspeicher in welchem das Arbeitsprogramm (Firmware) residiert (meist als ROM oder FLASH-Speicher realisiert, wobei herstellerseitig programmiertes ROM nur bei sehr großen Stückzahlen interessant ist). Heutige µC haben zwischen einigen kByte und 1 bis 2 Mbyte Speicher.
- interner nichtflüchtiger Datenspeicher zur Ablage von Konfigurationen oder aktuellen Prozessparametern (meist als EEPROM mit einigen Bytes bis einigen Dutzend kByte realisiert)

79.) Kriterien, die ein universeller digitaler Signalprozessor erfüllen sollte

- Effiziente Arithmetik, (z.B. Multiplizieren + Speichern)
- Effiziente Schleifen für Signalflüsse
- Mehrere Busse
- Spezielle Betriebsmittel (Digitale Filter..)
- Harvard-Architektur
- Breite Busse, gute interne Caches
- evt. zwei Datenspeicher für Zufuhr von zwei Datenströmen.

80.) 4 Adressierungsarten für DSP's

- Register – indirekte Adressierung (mit nachträglicher Inkrementierung)
- Register – direkte Adressierung
- Modulo – Adressierung
- Bit – Reverse Adressierung (für FFT)

81.) Welche Speicherarchitektur wird gerne in DSP's verwendet und warum?

Es wird Harvard Architektur verwendet, d.h. Daten und Befehle befinden sich im Unterschied zum von Neumann Architektur in getrenntem Speicher. In einem Taktzyklus können ein Befehlsword und ein Datenword transferiert werden, was eine Beschleunigung um den Faktor 2 gegenüber einem Rechner mit einem einzigen Speicher ergibt. Da im allgemeinen bei Rechenoperationen 2 Operanden benötigt werden, würde ein paralleler Transfer dieser Daten zur ALU einen deutlichen Geschwindigkeitszuwachs bringen.

82.) Kann bei Verwendung eines DSPs in einem öffentlichen Netz eine „Buffer-Overflow Attacke“ durchgeführt werden? Ja oder Nein. Begründen Sie.

Kann es nicht, weil Daten und Code im getrennten Speicher stehen (Harvard-Architektur)

83.) Warum ist 500MHz DSP bei graphischen Aufgaben schneller als ein 2GHz P4?

Ein DSP besitzt im Gegensatz zu normalen Prozessoren eine MAC (Multiply and Accumulate)- Einheit, die es erlaubt in sehr kurzer Zeit arithmetische Operationen zu verarbeiten. Dem DSP liegt eine erweiterte Harvard-Architektur zugrunde. Die für graphische Anwendungen notwendige Rechenleistung kann durch Vervielfachung der MAC-Einheiten extrem gesteigert werden (schneller als 2Ghz P4)

85.) Welche Funktionseinheiten sollte ein moderner integrierter Signalprozessor besitzen?

- Jeder Befehl soll nur einen Taktzyklus dauern
- Befehlsatz soll gut an DSP Aufgaben angepasst sein
- MAC mit großer Genauigkeit
- Adress- und Datenverarbeitungsoperationen parallel
- Getrennte Speicher und Bussysteme zumindest für Befehle und Daten
- Register ohne Einschränkung bei jedem Befehl verwendbar
- Bei Schleifen → Schleifenzähler und Stack automatisch verwaltet
- Bei indirekter Adressierung → bit reversed und Ringpuffer möglich
- SIMD und eingeschränkte MIMD Befehle sollen vorgesehen sein

86.) Welche verschiedenen Wortlängeneffekte bei Signalprozessoren kennen Sie? Geben Sie die Ursachen für die einzelnen Effekte an!

- Toleranzproblem durch die Koeffizientengenauigkeit
- Filterrauschen und Untersteuerung durch die Quantisierung bei der Multiplikation
- Übersteuerung bei der Addition/Subtraktion
- nichtlineare parasitäre Schwingungen (Grenzzyklen) durch Quantisierung der Signale in Rückkopplungsschleifen

87.) Sie wollen mit einem Signalprozessor einen FFT-Algorithmus implementieren. Welche Anforderungen stellen Sie an die folgenden Funktionseinheiten?

Arithmetikeinheit: Kaskadenanordnung, Blockgleitkommazahlen, Blockverarbeitung;

Adressrechner: Bit-reversed

Programm-Sequencer: ???

Speicher: In-Place zur Speicherung der Signalwerte

88.) Welche Aufgaben hat die Programmsteuereinheit eines Signalprozessors?

- Automatische Verwaltung von Unterprogrammaufruf und Rückkehr zum übergeordnetem Programm
- Interrupt-Verwaltung (Maskierung, Verschachtelung)
- Automatische Schleifenverwaltung (Schleifenzähler, Abbruchbedingung, Stack)
- Retten und Wiederherstellen des ALU-Status bei Interrupts
- Verwaltung bedingter Sprünge (Prüfen der Bedingungen)
- Bedingte Befehlsausführung (Conditional Execution)
- Verwaltung eines event. vorhandenen Instruction Cache

89.) Was ist Rapid Prototyping?

Softwaresystem mit Eingabe des DSP-Algorithmus und Ausgabe einer Maske für einen VLSI-Chip (Wunschtraum). „Bescheidenere“ Lösungen: Simulationswerkzeuge MATLAB und PTOLEMY.

90.) Programmspeichergröße und Taktrate einer typischen Smart Card?

- 66 MHz Taktrate
- 32-Bit Prozessor
- 400 kByte RAM
- 1 MByte EEPROM

91.) Sicherheitsmaßnahmen bei SmartCard? 3Szenarien?

- stromführende Metallisierungsebenen, die den Chip umgeben (Werden diese z.B. durch Ätzen entfernt, ist der Chip nicht mehr funktionsfähig.),
- die kryptographische Sicherung von Daten in nicht-flüchtigen Speicherbereichen.
- Ionen-implantierte ROM (verhindern Auslesen mittels IR- oder UV-Licht),

92.) Was ist ein FPGA und wie programmiert man ihn?

Ein Field Programmable Gate Array (FPGA) ist ein frei programmierbarer Logikbaustein. Er besteht aus Tausenden Logikblöcken, denen verschiedenste logische Funktionen zugewiesen werden können. Mit einem FPGA ist es möglich, komplette Mikroprozessoren nachzubauen (RISC und CISC). Programmiert wird er durch spezielle Synthesesoftware. Es gibt zwei Arten von FPGA:

- FPGA optimierte Prozessoren:
Die Implementierung eines „klassischen“ Prozessors mithilfe eines FPGA ist meist ineffizient, allerdings gibt es einige spezielle Mikroprozessorarchitekturen (RISC ähnlich) die den Vorteil haben, direkt mit den Registern oder der ALU einer CPU verdrahtet werden zu können (ALTERA: NIOS II)
- Selbstrekonfigurierende FPGA-Prozessoren:
Moderne FPGAs können sich während des Betriebes selbst rekonfigurieren (Rekonfiguration) oder müssen angehalten werden (statische Rekonfiguration).

93.) Unterschied zwischen Emulation-Chip-ICE und "Bond out"-Chip-ICE.**Emulator-Chip-ICE (In circuit emulator):**

Es wird eine andere leistungsstärkere Prozessortype oder ein anderes Prozessorsystem mit CPU und externen Komponenten als Ersatz für den ursprünglichen Prozessor eingesetzt. Dadurch ist es möglich eine ganze Prozessorfamilie mit dem ICE zu debuggen, allerdings können die elektrischen Eigenschaften und Timing unterschiedlich sein. Die Emulation eines komplexen Prozessors ist außerdem sehr kostspielig.

Bond out chip ICE:

Es wird die gleiche Prozessortype verwendet, die vom Hersteller mit zusätzlichen Kontakten versehen wurde. Dadurch ist es möglich CPU-interne Datenleitungen nach außen zu führen (Bond out). Der Vorteil liegt darin, dass im Gegensatz zum Emulator-Chip-ICE ein „echter“ Prozessor in der Schaltung eingebaut wurde, allerdings muss man für jede Prozessortype einen neuen Bond-out-Chip kaufen.

94.) Welchen Anteil der Verlustleistung erhöht asynchrones Mikroprozessordesign, welchen verringert es?

Asynchrones Mikroprozessordesign verringert die dynamische Verlustleistung. Da es keinen gemeinsamen Takt gibt, schalten die Transistoren nur wenn sie wirklich gebraucht werden, was zu einem bis zu 70% (laut Wikipedia) niedrigeren Energieverbrauch führt.

95.) Vorteile von asymmetrischen Multiprozessoren?

Asymmetrische Dual Core: unterschiedliche cores

Mobilgerät: Betriebssystem, GUI, Webdienste & DSP Funktionen notwendig

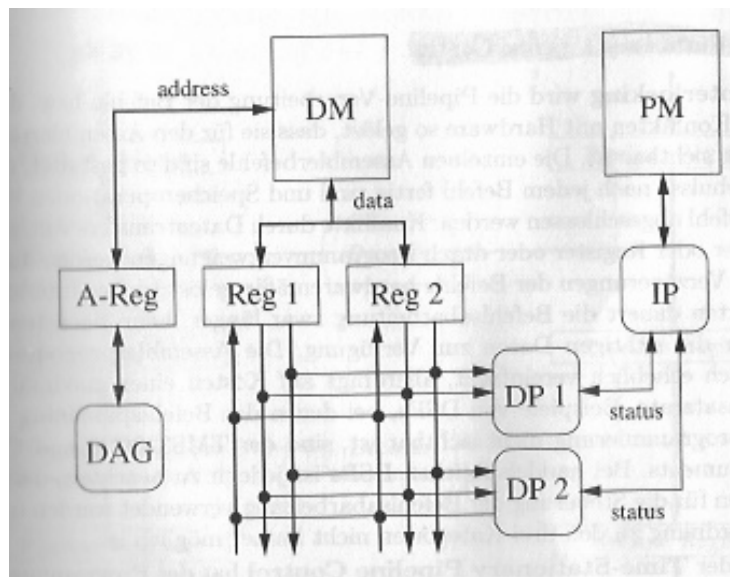
Mit asymmetrischem dual core machbar.

Bei asymmetrischen Implementierungen werden CPU Cores verschiedener Architekturen mit meist ganz verschiedenartigen Befehlssatz, Speichermodell und Taktfrequenz verwendet.

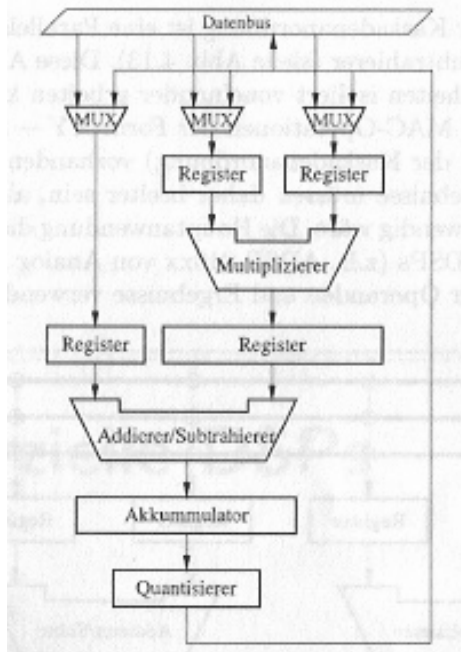
96.) Ein Vorteil einer asymmetrischen Multiprozessorarchitektur

Ein Asymmetrisches Multiprozessorsystem ist eine Multiprozessor-Architektur, bei dem ein bestimmter Prozessor das Betriebssystem ausführt und Prozesse an sich und die anderen Prozessoren verteilt. Es können allerdings auch mehrere Prozessoren bestimmte Teile des Betriebssystems ausführen, um die Last zu verringern

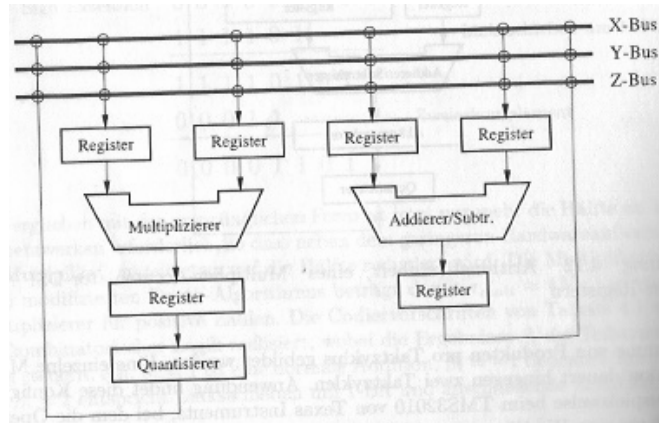
97.) Skizzieren Sie eine Signalprozessorarithmetikeinheit in Kaskadenstruktur. Welche maximale Durchsatzrate (Addition + Multiplikation pro Zeiteinheit) weist ihre Recheneinheit auf? Superskalare registerbasierte Rechenarchitektur aufzeichnen!



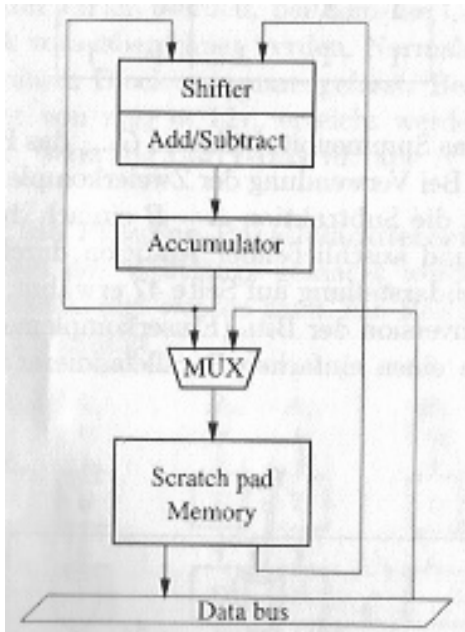
98.) Architekturen von MAC-Einheiten!



Arithmetikeinheit Multiplier-Akkumulator-DSP mit Pipeline Registern



DSP-Arithmetikeinheit für $X \cdot Y + Z$ Operationen mit parallelschaltetem Multiplizierer und Addierer / Subtrahierer



Arithmetikeinheit eines Shift-Add-DSP für Multiplikation mit dem CSD-Code.

Kapitel 4 – Computerarchitektur:

99.) SRAM vs. DRAM – 4 wichtige Unterschiede:

SRAM

- Bits in Flip-Flops gespeichert
- nicht zerstörendes Lesen
- schneller als DRAM
- kein Refresh-Zyklus benötigt

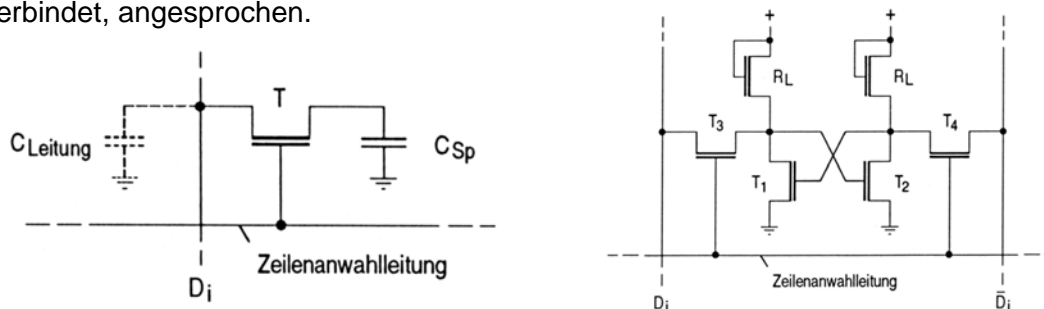
DRAM

- Bits in Kondensatorladungen gespeichert
- zerstörendes Lesen
- langsamer als SRAM
- Refresh-Zyklus unvermeidbar

100.) Wie sind DRAM und SDRAM aufgebaut?

DRAM (Dynamic random access memory):

Bits werden durch eine Kondensatorladung gespeichert. Die Speicherzellen werden über eine Zeilenleitung mit einem Schalttransistor, der den Kondensator mit der Spaltenleitung verbindet, angesprochen.



Bits werden zerstörend gelesen, weil beim Lesevorgang die Speicherzelle an die Masse gehängt wird und der auftretende Entladestrom gemessen wird. Dabei verliert der Kondensator die Ladung und muss neu beschrieben werden (write after read). Die Zykluszeit des Lesens eines DRAMs ist also mindestens doppelt so lang, wie das Lesen eines SRAMs. Der Vorteil des DRAMs liegt in der hohen Speicherkapazität.

SDRAM (Synchronous Dynamic RAM):

Der SDRAM ist eine getaktete DRAM-Technologie, wodurch der Steuerungsaufwand für asynchrone Kommunikation wegfällt. Eine State Machine ist ebenfalls integriert, die über ein programmierbares Register weiß, wie viele Wörter übertragen werden sollen und automatisch die Adressfortschaltung übernimmt. Erweiterungen sind DDRSDRAM (Double Data Rate SDRAM), welche nicht nur die positive Taktflanke, sondern beide Taktflanken zur Datenübertragung nutzen.

101.) Wann werden bei DDRSDRAM Daten übergeben?

(Double data rate) bei steigender und fallender Flanke. QDR (quad data rate) gibt's auch.

102.) Welches Problem löst Interleaving bei DRAM?

Wenn von 2 nacheinander folgenden Adressen gelesen wird, muss nicht gewartet werden bis der Refresh-Zyklus abgeschlossen ist sondern es wird von einer anderen Speicherbank gelesen während bei der anderen noch geschrieben wird.

103.) Was ist ein refresh-Zyklus und wozu dient er, wie läuft er ab?

Ein Nachteil von DRAMs. Da die Speicherkondensatoren durch unvermeidliche Leckströme ihre Ladungen (und damit auch die in ihnen gespeicherte Information) im Laufe der Zeit verlieren, müssen bei einem DRAM diese Kondensatoren in regelmäßigen Abständen nachgeladen werden (Refresh). Dies geschieht in zwei Abläufen: Zuerst muss zerstörend gelesen und gleich darauf wieder geschrieben werden. Dabei gibt es zwei Arten des Refreshens: Der Speicher mit n Zeilen wird in m Teile aufgeteilt. Ist $m > 1$ spricht man vom **distributed refresh**, ist $m=1$, von **burst refresh**. In Echtzeitsystemen kommt eher distributed refresh zum Einsatz, da die Antwortzeiten gering sein müssen. Dieser Vorgang kostet Zeit, der Bus wird unter Umständen belegt und verursacht natürlich auch einen höheren Energieverbrauch.

104.) Phänomene, die zu dynamischen Verlustleistungen bei CMOS führen?

Dynamischer Verbrauch geht auf Schaltvorgänge zurück, bei denen – unvermeidbare – Kapazitäten umgeladen werden und in den Schaltübergängen Schaltströme fließen. Bei DRAMs sind diese Kapazitäten nicht nur unvermeidbar, sondern sogar die Basis der Technologie. Zusätzlich zu den Schaltkapazitäten kommen Kurzschlüsse, die auftreten, wenn CMOS-Schalter umschalten und die beiden dafür notwendigen Transistoren für einen kurzen Augenblick beide leitend sind.

84.) Welche Aufgabe hat ein Cache bei Signalprozessoren?

Zwischenspeicher der Programmbefehle. Speziell bei Schleifen Programmspeicher entlasten

113.) Wie funktioniert assoziativer Speicher?

Assoziative Speicher werden im Cache verwendet. Dabei wird das CAM (Content Addressable Memory) verwendet. Werden dem CAM Daten geschickt, vergleicht der Encoder zuerst, ob die angeforderten Daten schon im CAM liegen. Ist das der Fall, liefert er die Position dieser Daten zurück. Befinden sich die gewünschten Daten nicht im CAM, aktiviert der Cache Controller die Ausgangspuffer und kopiert die aus dem Hauptspeicher geholten Daten nun in das CAM. Die Daten werden an irgendeine Stelle im Cache Datenspeicher geschrieben, aber die Adresse muss im CAM abgespeichert werden. Den Umstand, dass beliebige Adressen auf beliebige Cachepositionen abgebildet werden, bezeichnet man als voll assoziativ.

105.) Welche Implementierung von Caches gibt es und welche hat den geringsten Hardwareaufwand?

- Voll assoziativer Cache
- Set assoziativer Cache

Beim set assoziativen Cache ist der Hardwareaufwand am geringsten, weil beim voll assoziativen Cache ein CAM benötigt wird, der sehr komplex ist und vom Prinzip her schneller sein muss als das Cache - Data - Memory um einen Geschwindigkeitsvorteil ausnützen zu können, das ist aber in der Realität kaum zu realisieren, da das Cache – Data – Memory ohnehin schon ein sehr schneller Speicher ist.

107.) Was macht der Cache-Controller?

Die Steuerung des Caches übernimmt der Cache Controller. Er steuert das eigentliche Cache-Ram (Cache Data Memory) und die Puffer zum Systembus.

108.) Was steht im Cache außer eigentlichen Nutzerdaten?

Der Cache Speicher ist typischerweise zeile-orientiert, die einzelnen Zeilen enthalten:

- Frame (die Kopie der Hauptspeicher Inhalte)
- Tag (die Hauptspeicher-Adresse von der der Frame-Inhalt kam)
- Flags (dirty, used, etc.)

Beim Neustarten des Computersystems ist der Cache leer.

109.) Unterschied zwischen Write-Through und Copy-Back

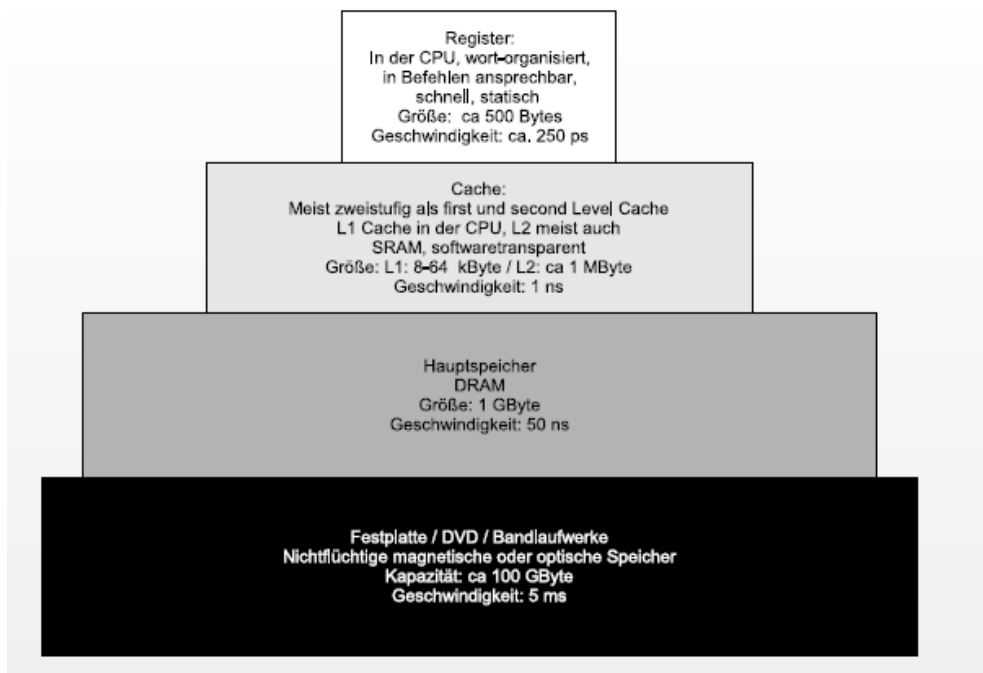
Copy-Back-Verfahren: Die vom Prozessor veranlassten Schreiboperationen werden zunächst nur im Cache ausgeführt und gesammelt; die Aktualisierung im Hauptspeicher erfolgt erst dann, wenn die Cache-Einträge wegen eines cache miss vom Cache an den Hauptspeicher retourniert werden (d. h. wenn für weitere Einträge „kein Platz“ mehr im Cache ist). Hier wird offensichtlich bewusst die Inkonsistenz der Speicher-Einträge in Kauf genommen, solange der Cache-Speicher eine Kopie davon besitzt. Genauer gesagt ist im Cache eigentlich das Original der fraglichen Daten enthalten, während die entsprechenden Bereiche des Hauptspeichers besser als ausgelagert markiert werden sollten.

Write Trough-Verfahren: Die oberste Maxime beim Write-through-Verfahren ist es, die Inhalte im Hauptspeicher und im Cache jederzeit auf dem aktuellen Stand zu halten. hier wird also genau jenes vermieden, welches vorhin beim Copy-Back-Verfahren als problematisch hingestellt wurde. Jede Schreiboperation vom Prozessor in den Cache wird grundsätzlich in den Hauptspeicher weitergereicht, also „durchgeschrieben“. parallel dazu werden die Schreibaufträge natürlich auch im Cache-Speicher wahrgenommen. Da der Hauptspeicher bekanntermaßen viel langsamer ist als der Cache-Speicher, ist es wohl hinderlich, jedes Mal auf den Abschluss eines vollen Hauptspeicher-Schreibzyklus zu warten. Eine probate Abhilfe bieten hier zu Beispiel Pufferspeicher (vergleiche buffered write through), wobei aber sicher noch nicht alle Probleme dieses Verfahrens aus der Welt geschafft sind.

110.) Welches Problem löst Buffered write through zum normalem write through Cache?

Beim normalen Write Trough werden die Daten an den Cache gerichtet und werden in einem Durch gleich zum Hauptspeicher weitergereicht. Problem hierbei ist allerdings die Geschwindigkeit des Hauptspeichers der ja langsamer ist als der Cache. Dieses Problem wird mit Buffered Write Trough behoben: Es wird ein Pufferspeicher verwendet der die Daten zwischenspeichert bis sie in den Hauptspeicher geladen sind d.h. die CPU muss nicht warten.

111.) Speicherhierarchie: Zeichne und erkläre wozu?



112.) Unterschied zwischen Pages und Segments ?

	Page	Segment
Wörter pro Adresse	Eines	Segment und Offset
Transparenz	Unsichtbar für Anwendungsprogramme	Kann für Anwendungsprogramme sichtbar sein
Ersetzen eines Blocks	Trivial, da alle Blöcke gleich groß	Komplex, freier Bereich muss gefunden werden
Speicherineffizienz	Interne Fragmentierung durch unausgefüllte Frames	Externe Fragmentierung durch unterschiedlich große Segmente
Effizienter Bus-Traffic	Ja: Page Size kann auf optimale Zugriffs- und Transferzeit abgestimmt werden	Nicht immer, kleine Segmente sind ineffizient

114.) Was ist virtueller Speicher?

- Wenn der Arbeitsspeicher nicht ausreichend für eine Anwendung ist, so müssen Daten aus dem Arbeitsspeicher z.B. auf die Festplatte geschrieben werden
- Problem ist dabei, dass ein größerer logischer Adressbereich auf einen kleineren physikalischen Adressbereich abgebildet werden muss.
- Für einen virtuellen Speicher wird meistens einen MMU verwendet. Dort werden die logischen Adressen in Realadressen mit Hilfe von Tabellen umgesetzt die vom Betriebssystem verwaltet werden.

115.) Welche wesentliche Eigenschaft würde verletzt, wenn man nur mit physikalischen Adressen arbeiten würde?

Der logische Adressraum beschreibt die Organisation des Hauptspeichers aus Programmsicht. Der physische Adressraum ist durch den tatsächlich verfügbaren Arbeitsspeicher gegeben. Logische Adressen ermöglichen das Erstellen von multitasking-fähigen Programmen, da nicht explizit untersucht werden muss, ob eine bestimmte Adresse mit Speicheradressen aus anderen Programmen kollidiert.

116.) Wozu braucht man logische Adressen?

Es gibt logische und physikalische Adressen. Physikalische sind direkt die Zellen im Speicher in denen die Daten stehen. Mit den logischen arbeitet man, ansonsten wäre es schlecht, weil man genau einen Speicherbaustein verwenden könnte, Kaskadierung wäre eigentlich nicht möglich da ja jeder der Bausteine die gleichen physikalischen Adressen hat

117.) Was ist der Unterschied zwischen logischen und physikalischen Adressen und welche Einheit im Computer rechnet diese um?

Die MMU (Memory Management Unit rechnet logische in physikalische Adressen um). Logische Adressen sind Adressen in Programmabläufen. Physikalische Adressen sind Adressen einer realen Speicherzelle, wo Informationen abgespeichert werden.

54.) Welche 2 Arten der Vektoradressierung gibt es?

- Tabellarische Zuordnung zwischen den logischen und den physikalischen Adressen (indirekte Adressierung).
- Abbildung nach linearer Funktion $y = kx + d$
 y.....physikalische Adresse x
 d.....Teile der logischen Adresse
 k.....Parameter (meist eine Zweierpotenz)

118.) Welche Auswirkungen hat eine große Clustergröße einer Festplatte auf die FAT (File Allocation Table) Zugriffszeit

Ein Cluster ist die aus einem oder mehreren Sektoren bestehende Zuordnungseinheit, die von einer Datei belegt werden kann. Der Datenbereich ist in eine feste Anzahl von Clustern eingeteilt. Zu jedem dieser Cluster existiert ein Eintrag in der FAT. Dadurch wird die Zugriffszeit vermindert.

119.) 4 Aufgaben der MMU ?

- Übersetzen der logischen Adressen in physikalische Adressen
- Übertragung der physikalischen Adressen in den Cache oder auf den externen Adress-Bus.
- Hardwareunterstützung für Virtual Memory und normales Paging in der Paging Unit
- Hardwareunterstützung für Segmente
- Unterstützung für Memory Protection

120.) Was macht die GPU?

Berechnung von Grafiken

spezialisiert auf hohen Durchsatz

PC: auf Grafikkarte, via Northbridge mit CPU verbunden

Hardware für 3D-Rendering, Polygone, etc.

121.) Was macht die FPU?

Floating point Berechnungen

Grundrechnungsarten, Sinus, Wurzelziehen, etc.

Mehr oder weniger gesteuert von CPU

122.) Erklären Sie die Northbridge!

CPU und GPU sind über die so genannte Northbridge miteinander verbunden. Bei der Northbridge handelt es sich um einen sehr schnellen Controllerbaustein, an den verschiedene Busse angreifen, der aber auch den Hauptspeicher mit dem Prozessor verbindet.

123.) Wozu dient der Translation Lookaside Buffer, was ist er und was leistet er?

- Er ist eine Funktionseinheit der MMU (Memory Management Unit)
- Beim Umrechnen von virtuellen Speicher in physikalischen Speicher muss mit Übersetzungstabellen gearbeitet werden → sehr zeitintensiv deshalb wird im TLB die Pagetabelle gepuffert
- TLB kann begrenzte Menge dieser Daten halten (32 – 64 Einträge) und beschleunigt den Vorgang damit
- Erlaubt durch assoziative Ordnungsregister parallele Zugriffe und ist dadurch sehr schnell aber auch teuer.

124.) Fehlererkennung bzw. Korrekturverfahren?

Fehlererkennung: Paritätsbit, Reed-Solomon Code, CRC, Cross-Interleaved Code (CDs).

Paritätsbit: zeigt ob die Anzahl der Einser im Code gerade/ungerade ist, kann nur Einzelfehler entdecken. CRC basiert auf Polynomdivisionen, kann leichter in Hardware implementiert werden (Festplatten, Disketten).

125.) Wie werden Speicherblöcke mittels Parity-Bit auf Fehler überprüft?

Paritätsbit: zeigt ob die Anzahl der Einser im Code gerade/ungerade ist, kann nur Einzelfehler entdecken.

126.) Vorteil der Thumb-Erweiterung des ARM?

- Thumb Befehle halb so lang wie ARM Befehle
- Ausführung in 32 Bit Kernen mittels Dekompressor (wandelt 16 Bit Thumbcode in 32 Bit ARM ohne Performanceverlust um)
- Thumb ist Submenge des ARM-Befehlssatz
- 30 – 40 % kürzerer Code
- Thumb und ARM-Code ist mischbar → beim Umschalten wird Pipeline geleert
- weniger Speicher → Einsatz für mobile Geräte

127.) Was ist eine Daisy Chain?

Eine Daisy-Chain ist ein Beispiel für dezentrale Busarbitration. Die Master erkennen selbstständig ob sie den Bus benutzen dürfen. Es gibt verschiedene Strukturen einer Daisy-Chain. Es kann z.B. einen Central Bus Arbiter geben, das ist ein ausgezeichnete Master. Je näher ein Master am Central Bus Arbiter steht, desto höher ist seine Priorität. Will ein Master schreiben, zieht er die Bus Request Leitung (BRQ) auf LOW und erhält vom Central Bus Arbiter eine Erlaubnis in Form von einem BGT (Bus Grant Signal). An eine Daisy Chain können beliebig viele Master angeschlossen werden. Allerdings sind die hinteren Master benachteiligt, weil ihre Priorität mit der Entfernung abnimmt. Um die Daisy Chain gerechter zu machen, kann die BUS GRANT Leitung des ersten Masters mit dem letzten Master verbunden werden und der Central Bus Arbiter weggelassen werden. Diese Methode heißt „zirkulärer Arbiter“.

128.) Wer darf bei einer Daisy Chain unterbrechen?

Daisy-Chain ist ein Beispiel für dezentrale Busarbitration, die einzelnen Master erkennen selbstständig, ob sie den Bus benutzen dürfen. Dabei sind die Master nach ihrer Priorität geordnet, d.h. das eine Einheit erhält die Kontrolle über den Bus, falls keine von den anderen Einheiten mit höherer Priorität den Bus braucht.

129.) Wie kann zwischen Speicher und E/A Controller Daten ausgetauscht werden?

Über DMA (direct memory access) oder über den Prozessor

130.) Vollständiger Interruptzyklus -> was macht CPU (Register, etc.)

Befehlszählerstand u. Programmstatuswort auf den Stack retten

Register und AKKU auf dem Stack ablegen

Befehlszähler auf ISR- Einsprung-Adresse setzen, Programmstatus zurücksetzen

ISR abarbeiten

Befehlszählerstand und Programmstatus vom Stack zurückschreiben

Register, AKKU vom Stack holen

„normale“ Programmabarbeitung fortsetzen

131.) Wozu dient ein Interrupt Controller

Zum Auflösen mehrerer Interrupts. Von jeder Interruptquelle wird eine individuelle Interrupt - Request-Leitung zum Interruptcontroller geführt und je nach Ausstattung des Prozessors mit codiertem od. uncodierten Interrupteingängen hat der Baustein entweder nur einen Interrupt-Level zuzuordnen oder gegebenenfalls auch noch die Prioritätenfrage zu entscheiden.

132.) 2 Hauptaufgaben von Semaphoren?

Synchronisieren von parallelen Prozessen (z.B. Prozesse sollen unter einer bestimmten Bedingung (gegeben durch Semaphor) auf einander warten)

gegenseitiger Ausschluss (mutal exclusion) beim Zugriff mehrerer Prozesse / Prozessoren auf gemeinsame Betriebsmittel. Mit anderen Worten: Das Ermöglichen eines exklusiven Zugriffs mehrerer Prozesse auf eine gemeinsam genützte Ressource

Kapitel 5 – Systemsoftware:

133.) Phasen des Kompilierens und Compilerausführung?

1. lexikalische Analyse
2. Syntaxanalyse
3. Semantische Analyse
4. Zwischencodeerzeugung
5. Codeoptimierung
6. Codeerzeugung

134.) 4 Aufgaben eines Betriebssystems

- Rechteverwaltung, Sicherheit
- Virtueller Speicher, Massenspeicher
- Verwalten von I/O Geräten
- Speicherverwaltung

135.) Was bezeichnet man als Firmware?

Unter Firmware bzw. hardwarenaher Software versteht man Software, die in verschiedene elektronische Geräte in einem programmierbaren Chip, und zwar heute fast ausschließlich in Microcontroller, eingebettet ist. Sie ist zumeist in einem Flash-Speicher, einem EPROM oder einem EEPROM gespeichert, der heute zumeist in den Microcontrollerbaustein integriert ist. Bei PC-Peripherie wird manchmal aus Kostengründen (Entfall des Flash-Speichers) oder wegen leichter Anpassbarkeit (an z.B. unterschiedliche Absatzmärkte) auch die Firmware beim Laden des Treibers in das RAM des Peripheriegerätes übertragen. Der Hersteller eines elektronischen Gerätes ist für die Gesamtfunktion verantwortlich. Soweit hierfür ein Betriebssystem oder diverse Parameter festgelegt werden müssen, tut er dies als Firmware. Der Hersteller versucht Manipulationen durch den Anwender zu unterbinden. Insofern kann man von einer vom Hersteller „festgelegten“ Software sprechen.

136.) Defragmentierung von Festplatten

Speicherverwaltung in Blöcken (oder Segmenten?) dadurch werden einzelne Dateien die größer sind als diese Blöcke evtl. nicht zusammenhängend gespeichert Fragmentierung da man dann diese einzelnen teile suchen und zusammensetzen muss zusätzlicher Aufwand -> langsamer...Defragmentieren sucht die teile zusammen und speichert die Dateien als schön zusammenhängend ab (ich vermute einen Zusammenhang mit interleaving) dadurch erspart man sich den Overhead... Defragmentieren kann man alle Dateien die gerade nicht in Benutzung, Systemdateien z.B. werden gerade verwendet, bleiben also evtl. fragmentiert

137.) Was ist ein Segmentation-Fault? Wann tritt ein Segmentation-Fault auf und wie kann man das verhindern?

Bei Computern tritt eine Segmentation fault auf, wenn ein Computerprogramm auf eine Ressource (insbesondere auf Speicher) zuzugreifen versucht, die vor einem solchen Zugriff geschützt bzw. nicht erreichbar ist. Hierbei sendet der Betriebssystemkern (Kernel) ein Signal zum Prozess, der den Fehler ausgelöst hat. Dies führt im Normalfall zur Beendigung des Prozesses. Durch den Speicherschutz moderner Betriebssysteme und Memory Protection Units der Mikrocontroller werden dadurch andere Prozesse und das Betriebssystem geschützt.

138.) Was ist JTAG und wofür nutzt man es?

Die JTAG-Schnittstelle (Joint Test Action Group) ist die wichtigste Methode zum Testen und Verifizieren von Software und Hardware in Mikroprozessorsystemen. Die zum Debuggen notwendige Hardware ist serienmäßig in dem Prozessor integriert und über einen seriellen Datenbus nach außen geführt. Dieser JTAG-Port besteht aus 5 Leitungen, an die Bausteine verschiedener Hersteller gemeinsam in einer Kette verbunden werden können. Vorteile des JTAG sind, dass mit wenigen Leitungen viele Prozessoren angesprochen werden können und auch Prozessoren die im fertigen Produkt installiert sind, debuggt werden können. Außerdem ist das Manipulieren aller Register und der I/O-Zugriff möglich. Nachteile sind die proprietäre JTAG-Hardware und die Notwendigkeit, den Programmablauf stoppen zu müssen um Debugg-Informationen zu übertragen.

139.) Was ist ein Präzedenzgraph & wozu wird er beim Design verwendet?

Zur besseren Darstellung der Reihenfolge der Berechnung wird der SFG in den Präzedenzgraphen (PG) umgezeichnet, bei dem die Verzögerungselemente dadurch eliminiert werden, dass sie durch eingeprägte Signale (gespeicherte und daher zum Zeitpunkt n bekannte Signalwerte) ersetzt werden.

- bestimmt notwendige Hardware mit optimalen Zeitplänen mit Auslastung der Recheneinheiten
- Ermöglicht Auslesen des zeitkritischen Pfades
- Zeigt inhärente Parallelität des Algorithmus
- Wird verwendet, um verschiedene Hardwarekonfigurationen durchzuspielen
- Sichere Methode zum Feststellen der Berechenbarkeit
- Bessere Darstellung der Berechnung
- Erstellung suboptimaler Zeitpläne

140.) Was ist ein erweiterter Datenflussgraph?

Bei dem benachbarte Abtastintervalle zusammengefasst werden!

Kapitel 6 – Netzwerke:

141.) Was ist Cluster Middleware?

Ein Cluster besteht aus einer Gruppe von PCs welche alle über eigene Ressourcen (Prozessor, Speicher, Netzwerk..) verfügen. Stellt sich eine Gruppe, durch ein Netzwerk verbundener Computer, für den Benutzer als ein einziges System dar, so sagt man es besitzt ein Single System Image (SSI). Dieses SSI wird durch einen Middleware-Layer zwischen Betriebssystem und Benutzerumgebung geschaffen.

Vorteile:

- Der Benutzer braucht nicht zu wissen wo seine Applikation laufen wird
- Der Benutzer muss nicht über die Beschaffenheit des Netzes Bescheid wissen.
- Der Administrator muss nicht wissen wo eine Ressource liegt.
- Das Risiko von Administrationsfehlern wird gesenkt wodurch das System für den Endbenutzer zuverlässiger erscheint.
- Die Zuteilung von Ressourcen an Benutzer wird vereinfacht.

Ziele beim Design der Middleware für Cluster Systeme sind in erster Linie das bereitstellen einer transparenten Schicht auf das System, einer gut skalierbaren Leistung und dem Erreichen einer hohen Verfügbarkeit des Systems.

142.) Was ist ein High-Availability-Cluster?

High-Availability-Cluster werden verwendet um eine hohe Ausfallsicherheit zu garantieren. Dabei werden mehrere Rechner benutzt; fällt einer aus, übernehmen die anderen Rechner.

143.) Welche Aufgabe hat die Mittelschicht bei Clustern?

Die Mittelschicht (Middleware Layer) zwischen Betriebssystem und Benutzerumgebung schafft ein Single System Image (SSI) für den Benutzer. Dieser muss nicht wissen, wo seine Applikation läuft, wie die Beschaffenheit des Systems aussieht und die Zuteilung von Ressourcen an den Benutzer wird vereinfacht.

144.) Was ist der Unterschied zwischen Client-Server und Peer-to-Peer?

Im Client-Server-System stellt der Server bestimmte Netzwerkdienste bereit; der Client kann diese in Anspruch nehmen. Das bedeutet, dass der Client meist Daten vom Server anfordert und dieser die Anforderung erfüllt. Bei Peer-to-Peer hingegen sind alle Teilnehmer sowohl Client als auch Server. Die gleichberechtigten Teilnehmer stellen unter Verzicht auf zentrale Koordinationsinstanzen wechselseitig Ressourcen zur Verfügung. Es gibt allerdings auch kombinierte Konzepte wie Hybrid P2P, ein teilzentrales Netzwerk.

145.) Wann und warum wird bei Client-Server synchrone bzw. asynchrone Kommunikation verwendet

Synchrone Kommunikation: Werden Anfragen und Antworten jeweils vollständig und nacheinander abgearbeitet. Sender und Empfänger von Daten synchronisieren, d.h. sie warten, bis die Kommunikation vollständig abgeschlossen ist. In einer synchronen Client-Server-Kommunikation ist also der Client solange blockiert, bis der Server den Request angenommen, bearbeitet und der Response beim Client angekommen ist. Die Vorteile dieses Kommunikationskonzepts sind offensichtlich bei Anwendungen wie dem Chat oder bei der Nutzung gemeinsamer Ressourcen, z.B. eines im Team bearbeiteten Dokuments. Denn bei der synchronen Kommunikation muss zuerst die Rückmeldung vorliegen, bis weitere Aktionen erfolgen können. Außerdem wird so jede Anfrage immer hinsichtlich ihrer Korrektheit verifiziert.

Asynchrone Kommunikation: Versteht man einen Modus der Kommunikation, bei dem das Senden und Empfangen von Daten zeitlich versetzt und ohne Blockieren des Prozesses durch bspw. Warten auf die Antwort des Empfängers (wie bei synchroner Kommunikation der Fall) stattfindet. Beispiel: E-Mail.

Kapitel 7 – Powermanagement:

146.) Unterschied zwischen dynamischer und statischer Verlustleistung und wie wirkt sich die Taktzahl aus?

Statische Verlustleistung:

Ursache: Leckströme heute dominierend durch immer feinere Strukturen

$$P_{\text{static}} \sim V_{\text{cc}} * I_{\text{leak}} * N * k_{\text{design}}$$

V_{cc} = Kernspannung des Prozessors

I_{leak} = Leckstrom eines Transistors

N = Anzahl der Transistoren

k_{design} = Faktor, der Integrationsdichte widerspiegelt

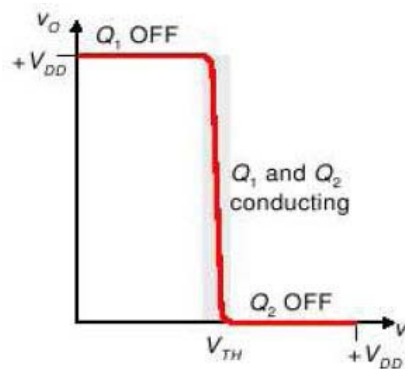
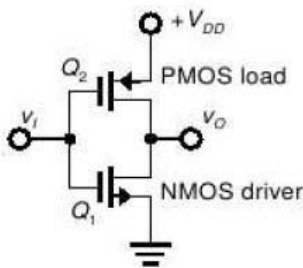
C = Kapazität, die geschaltet wird (Gatter, Leitungen)

Dynamische Verlustleistung:

$$P_{\text{dynamic}} \sim V_{\text{cc}}^2 * C * f + P_{\text{short}}$$

f = Frequenz des Prozessors

P_{short} = Verlustleistung durch temporären Kurzschluss beim Schalten



→ Je größer der Takt desto größer wird die Verlustleistung.

147.) Wie erfolgt bei statischen und dynamischen die Reduktion der Leistungsaufnahme?

Statische Verlustleistung:

$$P_{\text{static}} \sim V_{\text{cc}} * I_{\text{leak}} * N * k_{\text{design}}$$

→ Spannung wohl bester Ansatzpunkt (5V -> 3,3V -> 1,2V) -> Statische Verlustleistung fällt linear ab

Statische Verlustleistung:

$$P_{\text{dynamic}} \sim V_{\text{cc}}^2 * C * f + P_{\text{short}}$$

→ Spannung wohl bester Ansatzpunkt (5V -> 3,3V -> 1,2V) -> Dynamische Verlustleistung fällt sogar quadratisch ab. Wenn die Spannung halbiert wird, wird max. Frequenz halbiert (=halbe Rechengeschwindigkeit), dynamische Rechengeschwindigkeit fällt auf ¼

→ Nachteil: Maximale Taktfrequenz fällt (grob) linear mit Spannung ab.

148.) Energieverbrauch CompilerMöglichkeit zur Energieeinsparung

- Optimierung von Software – Vermeiden von „teuren“ Aktionen
- Compiler muss RAM / Buszugriffe vermeiden
- Register statt RAM
- i.d.R. spart Optimierung Delay / Speicher auch Energie!