Rechnerstrukturen I

Zusammenfassung

Aless Lasaruk

9. November 2005

Zusammenfassung

Dieses Dokument enthält Fragen und Antworten zum Fach Rechnerstrukturen I. Die Fragen sind durch systematisches Durchgehen des gleichnamigen Skriptums entstanden, decken aber den Stoff nicht vollständig. Man beachte auch, dass die Antworten nicht exakt formuliert sind. Das Ziel der Formulierung ist das Minimieren der schriftlichen Antwort und Maximieren der mündlichen.

Inhaltsverzeichnis

1 Grundlegende Methoden bei der Gestaltung von Prozessoren

- **Was ist ein *Rechensystem*?**
  Funktionseinheit zur Verarbeitung von Daten.
- **Was ist eine *Rechenanlage*?**
  Materielle Teil eines Rechensystems.
- **Wodurch ist eine *Benutzerschnittstelle* festgelegt?**
  Durch eine Sprache, die der Benutzer verwenden soll.
- **Was bedeutet *berechnen*?**
  Durchführung einer Abfolge von Transformationen um von einer Quell- zu einer Zieldarstellung zu gelangen.

1.1 Begriffe

- **Was ist ein Interpretationssystem?**
  Eine Sprache, ein Prozessor und ein Transformierer.
- **Welche grundlegenden Unterschiede gibt es zwischen Transformierern?**
  Übersetzer $(L_i \rightarrow L_{i+1})$ oder *Interpreter* $\begin{pmatrix} L_{i-1} \\ L_i \end{pmatrix}$.
- **Was ist *Prozessorverhalten*?**
  Ergebnis der Durchführung von Maschinenbefehlen.
- **Wie kann man die *Struktur einer Rechenanlage* definieren?**
  Durch Komposition von Teilanlagen und Verbindungen zwischen diesen.
- **Was ist ein *Operationsprinzip*?**
  Rechnerinterne Darstellung, Basisfunktionseinheiten und Interpretation der Transformationen. Das Operationsprinzip ist eine Invariante in der Menge aller gleichartigen Systemen.
- **Können Sie das Operationsprinzip eines *von Neumann-Rechners* beschreiben?**
  Bus an dem Befehlsprozessor, Datenprozessor und I/O-Prozessor mit Speicher verbunden sind. Im Speicher sind Daten und Programme. Speicher ist linear. Zugriff ist zufällig. Datenstruktur sind Binärcodes. Programme sind geordnete Mengen von Befehlen. Befehle enthalten Addressen von Zellen. Aufeinanderfolgede Befehle stehen aufeinanderfolgend im Speicher.
- **Welche Kritikpunkte an von Neumann-Systeme kennen Sie?**
  Befehle und Daten können falsch interpretiert werden, Bus/Speicher stellen einen Flaschenhals dar, Es gibt nur eine rechnende Einheit.
- **Können Sie aktuelle Modifikationen der Prozessoren nennen?**
  Relative Addressierung, Parallelverarbeitung von Befehlen, Sepeicherhierarchie, Mikroprogrammierung, Unterschiedliche Wortlängen der Befehle, Unterbrechungen.
- **Was ist eine *Rechenorganisation*?**
  Rechenstruktur und das Verhalten der Funktionseinheiten.
- **Was ist eine *Rechnerarchitektur*?**
  Rechnerstruktur und Operationsprinzip.

1.2 Rechnerarchitekturen

- **Welche funktionelle Anforderungen kennen Sie?**
  Einsatzbereich, Softwarekompatibilität, Betriebsanforderungen und Standards
- **Welche aktuellen Prozessoren kennen Sie?**
  Nachschauen.

- **Welche Parameter kann man zur Leistungsbewertung heranziehen?**
  Taktzeit, Ausführungszeit von einem Programm, Anzahl der Taktzyklen für ein Programm, Mittlere Anzahl von Maschinenbefehlen pro Taktzyklus, Anzahl der Maschinenbefehle pro Anwendung, Anzahl der Taktzyklen zur Ausführung eines Programms.
- **Welche Zusammenhänge kennen Sie zwischen diesen Größen?**
  Ausführungszeit = Taktzeit * Anzahl der Taktzyklen. Anzahl der Taktzyklen = Anzahl der Maschinenbefehle * Mittlere Anzahl der Takten pro Befehl.
- **Welche Kriterien spielen also für die Ausführungszeit eine Rolle?**
  Hardware-Technologie (Taktzeit), Hardware-Organisation (Mittlere Anzahl von Taktzyklen pro Befehl, z.B. Cache) und Befehlssatz und Compilertechnik (anzahl Befehle).
- **Wie kann man *Prozessorleistung* definieren?**
  Durch die *MIPS* Zahl. Diese ist das Reziproke der mittleren Ausführungszeit eines Maschinenbefehls (Mittlere Zahl der Taktzyklen pro Befehl mal die Taktlänge).
- **Welche Kritikpunkte gibt es gegen die MIPS-Bewertung?**
  Abhängigkeit vom Instruktionssatz und vom Programm/Daten. Kann sich gegenläufig zur Leistung entwickeln. Z.B. Einführung einer Fließkommaarithmetik führt zu langsameren Befehlen steigert aber die Leistung des Rechners.
- **Ist also die MIPS-Zahl eine ausreichende Charakterisierung der Leistung eines Rechensystems?**
  Nein.
- **Wie verändert sich die MIPS-Zahl?**
  Nach dem *Joy-Gesetz*. MIPS = 2 hoch (Jahr-1984).
- **Welche Alternativen dazu kennen Sie?**
  Die *FLOP Zahl*. Das ist die Anzahl der Gleitpunktoperationen geteilt durch die Ausführungszeit.
- **Von welcher Annahme geht die FLOP-Zahl aus?**
  Die Gleitpunktoperationen sind standartisiert.
- **Welche Kritikpunkte gibt es bei FLOP**
  Die selben, wie bei MIPS.
- **Um welche Zahlen geht es bei FLOPs?**
  Megat, Tera usw.-Flops.
- **Mit welchen nicht analytischen Mitteln kann man die Systemleistung noch messen?**
  Mit Testprogrammen.
- **Welche Typen von Testprogrammen kennen Sie?**
  Synthetische Programme (Benchmarks) und reale Programme.
- **Welche Darstellung der Leistung erweist sich als günstig?**
  Sterndarstellung.
- **Welche Kritikpunkt gibt es bei der Herrstellerangaben der Kennzahlen?**
  Diese werden durch mittelung bestimmt. Das verschleiert die Leistungseinbrüche. Prozessoren sind oft auf die Benchmarkprogramme getrimmt.
- **Welche Mittelung sollte man benutzen?**
  SPECint2000. Wurzel aus den Produkten der Quotienten der Benchmarkwerte.
- **Welchen Vorteil hat der *SPECint2000*-Wert gegenüber der arithmetischen Mittelung?**
  Das Verhältnis zweier SPEC-Werte von einer Maschine zu zwei Verschiedenen Maschinen X und Y liefert den SPEC-Wert für die beiden Maschinen X zu Y.
- **Was ist aus der Sicht des Anwenders und des Administrators für die Leistung wichtig?**
  Anwender ist an Antwortzeiten interessiert und der Administrator am Durchsatz und hoher Auslastung.

- **Welche Leistungsgrößen kennen Sie?**
  Antwortzeit (des Systems), Bedienzeit, Durchsatz, Auslastung.
- **Können bei gleichem Durchsatz unterschiedliche Antwortzeiten im System entstehen?**
  Ja. Z.B. durch Veränderung der Reihenfolge.
- **Warum ist die Maximalauslastung von Einzelkomponenten i.A. nicht möglich?**
  Engpässe erzeugen Wartezeiten, Konkurrierende Aufträge, Zu geringe Arbeitslast.
- **Welche Leistungsgrößen kennen Sie bei Speichern?**
  Kapazität, Zugriffszeit, Zykluszeit, Übertragungsrate, Preis.
- **Welche Leistungsgrößen kennen Sie bei Prozessoren?**
  Taktrate, MIPS/FLOPS, Übertragungsraten.
- **Welche Leistungsgrößen kennen Sie bei Programmen?**
  Anzahl der auzuführenden Befehle, Lokalität, Datenlokalität.
- **Welche Leistungsgrößen kennen Sie bei Betribssystem?**
  Seitenersetzungs-Strategie, Speicherverwaltung, Prozessverwaltung, I/O-Verwaltung.

1.3 Warteschlangennetze

- **Aus welchen Bestandteilen setzt sich ein *Warteschlangennetz* zusammen?**
  Bedieneinheit, Warteschlange, Verbindung.
- **Welche Eigenschaften können Warteschlangennetze nachbilden?**
  Unabhängige Bedieneinheiten, sequentielle Belegung der Einheiten durch Prozesse, gleichzeitige Belegung (Parallelität).
- **Was ist ein Verzögerungssystem?**
  Warteschlange und eine Bedieneinheit.
- **Durch weilche Daten ist eine Bedienstation charakterisiert?**
  Maximaler Durchsatz und mittlere Bedienzeit.
- **Wie berechnet sich der maximale Durchsatz?**
  Reziprokes der mittleren Bedienzeit.
- **Welche Aussagen erhält erwartet man durch eine Systemanalyse?**
  Wie lange bleibt ein Auftrag im mittel im System. Wie lang sind seine Wartezeiten im Mittel. Wie viele Aufträge sammeln sich im Mittel in den Warteschlangen.

1.4 Verteilungen

- **Welche wichtige Eigenschaft hat die Exponentialverteilung?**
  Die Markov-Eigenschaft. Auf Aufträge bezogen die Wartezeit zu einem bestimmten Zeitpunkt hängt nicht davon ab, wie lange man schon gewartet hat.
- **Welche Eigenschaften besitzen Bedienstationen in einem Warteschlangenmodell?**
  Diese sind aus mehreren Bedieneinheiten zusammengebaut. Sie sind entweder frei oder belegt.
- **Was ist die *Zwischenankunftszeit*?**
  Zeitraum zwischen den Ankünften zweier aufeinanderfolgender Aufträge.
- **Was ist die *Ankunftsrate*?**
  Reziproke der Zwischenankunftszeit (Mittlere Anzahl von ankommenden Aufträgen).
- **Was ist die *Bedienrate*?**
  Reziproke der mittleren Bedienzeit (Mittlere Anzahl von bedienten Aufträgen).

- **Was ist die *Auslastung* eines Systems?**
  Mittlere Bedienzeit durch die mittlere Zwischenankunftszeit. Oder auch Ankunftsrate durch die Bedienrate.
- **Wie ändert sich die Auslastung bei m Bedienstationen?**
  Durch m teilen, wenn die Bedienraten der Teilstationen gleich sind.
- **Wie kann man den Durchsatz bei einer Exponentialverteulung definieren?**
  Der Durchsatz ist gleich der Zugangsrate (Ankunftsrate).
- **Was ist die *Verweilzeit*?**
  Gesamtzeit, die ein Auftrag in der Station verbringt.
- **Was ist die *Wartezeit*?**
  Die Zeit gibt an, wie lange der Auftrag in der Warteschlange liegt.
- **Was ist die *Füllung*?**
  Anzahl der Aufträge in einer Bedienstation (unabhängig davon, ob diese in der Warteschlange sind oder nicht). *Warteschlangenlänge* ist die Anzahl der Aufträge in der Warteschlange.
- **Was besagt das Gesetz von Little?**
  Die Mittlere Füllung ist das Produkt der Ankunftsrate und der mittleren Verweilzeit. Mittlere Warteschlangenlänge ist das Produkt der Ankunftsrate und der Wartezeit.
- **Was ist ein offenes Warteschlangennetz?**
  Ein netz in dem die Anzahl der Aufträge nicht konstant ist.
- **Welche Syntax benutzt man zur Beschreibung von elementaren Bedienstationen?**
  Verteilung der Zwischenankunftszeiten/Verteilung der Bedienzeiten/Anzahl der Bedienstationen.
- **Was ist der Systemzustand bei Warteschlangennetzen?**
  Verteilung der Aufträge auf Bedienstationen.
- **Was ist ein Markov-Prozess?**
  Ein Prozess in dem der nächste Zustand nur vom aktuellen Zustand abhängt.
- **Was beschreibt das Einzelschrittverfahren?**
  Man betrachtet nur solche Änderungen, bei denen nur ein Auftrag die Stationen wechselt.
- **Wie ist bei geschlossenen Warteschlangensystemen die Übergangsrate?**
  Gleich der Bedienrate, weil das ist die Wahrscheinlichkeit dafür, dass ein Auftrag abgeschlossen ist.
- **Was besagt der *Satz vom Flußgleichgewicht*?**
  Die Wahrscheinlichkeit für das Auftreten eines Zustandes ist gleich der Wahrscheinlichkeit für sein Verschwinden. Summe der Produkte der Wahrscheinlichkeiten der erzeugenden Zuständen mit den entsprechenden Übergangsraten ist gleich der Wahrscheinlichkeit des Zustandes mal die Summe der ausgehenden Raten.
- **Was entsteht bei der Anwendung des Satzes vom Flußgleichgewicht?**
  Ein lineares Gleichungssystem (Zusätzlich summe aller Wahrscheinlichkeiten ist 1).
- **Was ist eine *Randwahrscheinlichkeit*?**
  Wahrscheinlichkeit dass in einer gegebenen Bedienstation eine bestimmte Anzahl von Aufträgen vorliegt.
- **Wie kann man in einem geschlossenen Warteschlangennetz die Auslastung beschreiben?**
  Auslastung einer Station ist die Summe der Randwahrscheinlichkeiten für mindestens ein Auftrag.
- **Wie kann man in einem geschlossenen Warteschlangennetz den Durchsatz beschreiben?**
  Durchsatz einer Station ist das Produkt der Auslastung mit der Bedienrate.

- **Wie kann man in einem geschlossenen Warteschlangennetz die mittlere Füllung beschreiben?**
  Die mittlere Füllung ist die Summe der Wahrscheinlichkeiten multipliziert mit der jeweiligen Anzahl der Aufträge.
- **Wie kann man in einem geschlossenen Warteschlangennetz die mittleren Verweilzeiten beschreiben?**
  Die mittleren Verweilzeiten für eine Station ist die mittlere Füllung durch den Durchsatz.
- **Ist das beschriebene Modell effizient?**
  Nein. Bei komplexen systemen wird der Aufwand zu hoch. Daher simuliert man das Warteschlangennetz meist.
- **Wie geht man bei der Analyse von offenen M/M/1 Warteschlangennetzen?**
  Man bestimmt zunächst die Wahrscheinlichkeiten der Zustände in Abhängigkeit von der Wahrscheinlichkeit des 0-en Zustandes. Es gilt $P(i) = P(0) * A^i$ mit A Auslastung. Dann bestimmt man $P(0) = 1 - A$.
- **Wie kann man bei M/M/1 Systemen den Durchsatz bestimmen?**
  Produkt der Auslastung und der Übergangsrate.
- **Wie kann man bei M/M/1 Systemen die mittlere Füllung bestimmen?**
  Die ist $A/(1 - A)$ für A-Auslastung.
- **Wie kann man bei M/M/1 Systemen die mittlere Verweilzeit bestimmen?**
  Die ist $1/m(1 - A)$ mit m ist die Übergangsrate.
- **Wie kann man bei M/M/1 Systemen die mittlere Warteschlangenlänge bestimmen?**
  Die ist $A^2/(1 - A)$.
- **Kennen Sie ein praktisches Beispiel der Analyse?**
  Ja muss ich noch anschauen. Seite 77.

1.5 Verfügbarkeit und Zuverlässigkeit von Rechensystemen

- **Was ist *Verfügbarkeit*?**
  Verfügbarkeit ist der Anteil der Zeit im Betrieb eines Systems in der Gesamtzwit.
- **Was ist *Zuverlässigkeit?***
  Wahrscheinlichkeit, dass in einer gegebenen Zeitspanne kein Ausfall entsteht.
- **Welche Fehlertypen kennen Sie?**
  Hardwarefehler, Softwarefehler, Bedienfehler.
- **Welche Auswirkungen von Fehlern kennen Sie?**
  Signalfehler (unzulässige Abweichung), Modulfehler (fehlerhaftes Verhalten eines Moduls), Systemfehler (Gesamtsystem).
- **Wie kann man Fehler nach ihrem Auftreten charakterisieren?**
  Permanente Fehler, intermittierende Fehler (sporadisches Auftreten), transiente Fehler (vorübergehend).
- **Wie kann man Fehler behandeln?**
  Auftragsverzögerung, Operationswiederholung, Fehlerkorrektur, Fehlermaskierung, Rekonfigurierung, Fehlergesichertes und Fehlerstoppverhalten.
- **Was ist *MTTF*?**
  Mean Time to Failure. Es gibt eine empirische und stochastische Zeit.
- **Wie ist die *Ausfallrate* Definiert?**
  Ausfallrate ist die Ausfallwahrscheinlichkeit während der Lebensdauer. Man darf die Ausfallrate nicht mit der Lebensdauer verwechseln. $\frac{R'(t)}{R(t)}$ mit $R(t)$ der Zeitverlauf noch nicht ausgefallener Systeme ist.

- **In welchem Zusammenhang steht die Verteilungsfunktion der Systemlebzeit und die Ausfallrate?**
  Die Ausfallrate charakterisiert die Verteilungsfunktion.
- **Was ist der Unterschied zwischen der Ausfallrate und *Überlebenswahrscheinlichkeit*?**
  Ausfallrate ist Verteilungsfunktion der Überlebenswahrscheinlichkeit.
- **Wie wird im Folgenden die Ausfallrate angenommen?**
  Konstant.
- **Was folgt die die Lebzeit des Systems bei konstanter Ausfallrate?**
  Diese ist exponentiell verteilt.
- **Was besagt dann der Mittelwert der Exponentialverteilung?**
  Die mittlere Zeit zwischen aufeinanderfolgenden Ausfällen. Und auch die mittlere Zeit bis zum ersten Fehler. Mittlere Lebenserwartung eines Systems.
- **Warum kann man die Ausfallrate als konstant ansehen?**
  Weil die über einen langen Zeitbereich in der Mitte der Lebzeit des Systems nahezu konstant ist. Badewannenkurve.
- **Wie kann man die mittlere Anzahl von Ausfällen in der Zeit t berechnen?**
  Ausfallrate mal die Zeit (wie auch erwartet).
- **Warum ist die Bestimmung der mittleren Ausfallzeit wichtig?**
  Ein gesichertes Kriterium zur Bestimmung von Ausfallzeiten in zusammengesetzten Systemen.
- **Wie bestimmt man die mittlere Lebzeit ohne der Dichtefunktion?**
  Partielle Integration der negativen Zuverlässigkeitsfunktion (Anteil funktionierender Systeme in der Zeit).
- **Wie bestimmt man die Wahrscheinlichkeit für den Ausfall einer Teilmenge von Systemen?**
  Poisson-Verteilung ($\frac{(-\lambda t)^k}{k!}e^{-\lambda t}$)
- **Was ist der Erwartungswert bei der Poisson-Verteilung?**
  λt (Exponentialentwicklung benutzen)
- **Wie begegnet man der höheren Ausfallrate bei zusammengesetzten Systemen?**
  Man führt Redundanz ein.
- **Was ist der Unterschied zwischen heißer und *kalter Redundanz*?**
  Bei heißer Redundanz sind alle redundanten Systeme eingeschaltet.
- **Was ist ein *Serien-Parallelgraph*?**
  Ein serien-parallelgraph setzt sich aus serein oder parallelkompositionen rekursiv zusammen. Verbindung zwischen zwei ausgezeichneten Knoten ist ein SP-Graph.
- **Was ist ein *Überlebensgraph*?**
  Ein Serien-Parallelgraph mit zwei Knotentypen, wobei Kanten jeweils nur unterschiedliche Knotentypen verbinden. Dem einen Typ von Knoten wird ein boolesches Signal (Lebenssignal) zugeordnet und die anderen sing Konstruktionsknoten.
- **Was beschreibt ein Überlebensgraph?**
  Bedingung für die Lebzeit des Systems. Keine Systemstruktur.
- **Welchen Vorteil hat die Darstellung in SP-Graphen?**
  Eine einfache hierarchische Dekomposition von Systemen.
- **Was ist ein *Überlebenspfad*?**
  Ein Pfad im Überlebensgraphen vom Anfangs bis zum Endpunkt, sodass alle Signale "Wahr" sind.

- **Wie verfährt man, wenn man die Überlebenswahrscheinlichkeit berechnen will?**

  Man konstruiert aus einem SP-Graphen einen booleschen und dann einen pseudo-booleschen Term, der dann in Wahrscheinlichkeiten umgesetzt wird.
- **Wie werden Disjunktionen umgesetzt?**

  Indem man De-Morgan benutzt und auf Komplemente und Konjunktionen zurückgeht.
- **Wodurch ersetzt man die Überlebenssignale?**

  Durch Verteilungsfunktionen für deren Lebzeit.
- **Was kommt als Ergebnis der Konstruktion heraus?**

  Eine Verteilungsfunktion für die Lebzeit des Systems.
- **Wie ist die Lebenserwartung bei seriellen und parallelen Systemen?**

  Einzelsystem durch die Anzahl der Systeme bzw. nicht mehr Exponentiell verteilt: Reziprokes der Ausfallrate plus die abgebrochene harmonische.
- **Wie ist die Lebenserwartung von dem zwei aus drei System?**

  Fünf sechstel der Lebenserwartung eines enzelnen Systems.
- **Wieso lohnt es sich trotzdem 2aus3-Systeme zu verwenden?**

  Wenn die Verwendungszeit deutlich kleiner ist als die mittlere Lebenerwartung, so ist die Ausfallwahrscheinlichkeit kleiner als bei einem Einzelsystem.
- **Wie nennt man ein 2aus3 System noch?**

  Ein *TMR*-System (triple modular redundancy).
- **Wann sind Fehlerbehebungen von Interesse?**

  Wenn die Reparatur sich loht (Taschenrechner) oder nicht all zu schwer ist (Raumfähre). Wenn Kurzausfälle tolerierbar sind.
- **Was ist *mittlere Verfügbarkeit*?**

  Verhältnis zwischen ausfallfreier Zeit und Reparaturzeit.
- **Welches einfaches Modell kennen Sie zur Beschreibung von Systemen mit Reparatur?**

  Zwei Zustände. Fehlerfrei und in Reparatur. Diese gehen unter gewissen Wahrscheinlichkeiten ineineander über (Mittlere Zeiten der Exponentialverteilungen).
- **Wodurch wird dieses Modell gelöst?**

  Satz vom Flussgleichgewicht.
- **Kennen Sie ein komplexeres Modell?**

  Fünf Zustände. Zwei Prozessoren, Rekonfiguration mit einem Prozessor, Neustart mit einem Prozessor, Ein Prozessor arbeitet und Ausfall.

1.6 Elementares Berechnungsschema

- **Was ist eine *Berechnung*?**

  Das Überführen des Maschinen-Zustandes von einem Anfangszustand in ein Endzustand.
- **Was ist ein *elementares Berechnungsschema*?**

  Eine Menge von Eingabezellen, Eine Menge von Ausgabezellen, Gerichteter Berechnungsgraph und Präzedenzgraph.
- **Was ist ein *Berechnungsgraph*?**

  Ein Berechnungsgraph ist ein Graph mit Speicherzellen als Knoten und Operatioren als Kanten, sodass stets immer verschiedenartige Knoten durch Kanten verbunden sind. Bedeutung: Welche Operatoren welche Speicherzellen verändern können.
- **Was ist ein *Präzedenzgraph*?**

  Ein Präzedenzgraph ist ein gerichteter azyklischer Graph mit mit Instanzen der Operatormenge als Knoten und einem Anfangs und einem Endzustand. Bedeutung: Ausführungsreihenfolge der Operatorinstanzen.

- **Wann ist ein Präzedenzgraph *deterministisch*?**
  Wenn mindestens ein Knoten mehr als einen Nachfolgerknoten hat.
- **Wann ist ein Präzedenzgraph *determiniert*?**
  Die Werte des Endzustandes sind von der Ausführungsreihenfolge unabhängig.
- **Wann sind Operatorinstanzen *direkt datenabhängig*?**
  Wenn eine Operatorinstanz unmittlbar den Wert der anderen nutzt.
- **Wann sind Operatorinstanzen *direkt ausgabeabhängig*?**
  Wenn beider Operatorinstanzen die selbe Speicherzelle beschreiben.
- **Wann ist ein elementares Berechnungsschema determiniert?**
  Wenn je zwei Operatorinstanzen weder direkt datenabhängig noch direkt ausgabeabhängig sind oder die Reihenfolge der Instanzenausführung ist durch den Präzedenzgraphen festgelegt (dann kommt immer das selbe raus).
- **Was besagt die *Bernstein Bedingung*?**
  Eigentlich ist das genau das, was oben steht.
- **Wann heißen zwei Instanzen *antidatenabhängig*?**
  Wenn im Präzedenzgraphen die Nachfolger und Vorgänger einen nicht leeren Schnitt haben, aber die Rechenergebnisse nicht genutzt werden.
- **Wie beseitigt man Antidatenabhängigkeiten?**
  Indem man jeder Zelle in einem bestimmten Ablauf genau einmal einen Wert zuweist.
- **Was versteht man unter dem Begriff *topologisches Sortieren*?**
  Ermittlung einer linearen Ordnung, die Elemente einer vorgegebenen Halbordnung benutzt.
- **Was kann man im Bezug auf Präzedenzgraphen und toplogisches Sortieren aussagen?**
  Jede topologische Sortierung stellt ein korrektes Programm dar.
- **Was ist der Unterschied zwischen *ASAP* und *ALAP*-Schedules?**
  Beim ASAP wird ein Befehl so schnell, wie möglich (as soon as possible) ausgeführt, d.h. wenn seine Daten vorliegen. Bei ALAP so spät, wie möglich.
- **Welche Sortierung ist vorzuziehen?**
  Eine die ein Kompromiss zwischen Hardware und Softwareaufwand liefert.
- **Wie kann die Berechnung von nicht datenabhängigen Instanzen durchgeführt werden?**
  Seriell von einer Einheit und parallel durch mehrere.
- **Wie kann die Berechnung von datenabhängigen Instanzen durchgeführt werden?**
  Seriell durch eine Dateneinheit und orts-seriell von mehreren Einheiten.
- **Wie können die Implementierungen verglichen werden?**
  Auslastung = Bedienzeit * Durchsatz.
- **Wie kann man die Auslastung bei zeitsequentiellen Implementierungen bewerten?**
  Minimaler Aufwand. Daher hohe Auslastung.
- **Wie kann man ortsseqentielle Implementierungen bewerten?**
  Auslastung geht gegen den Anteil der Arbeit einer Einheit.
- **Was benötigt man beim Kontrollfluß, wenn man parallele Verarbeitung zuläßt?**
  Synchronisationsbefehle.
- **Wie kann man Datenfluss-Graphen beschreiben?**
  Kombination aus Berechnungsgraph und Reihenfolgegraph. Das geht, wenn man zu jeder Operation genau ein Operationsexemplar zuordnen kann.

- **Was fehlt einem solchen Datenfluß-Graphen?**
  Schleifen.
- **Wann ist ein Operationsknoten aktiviert?**
  Wenn seine Eingabedaten verfügbar sind.
- **Was versteht man unter Feuern?**
  Verbrauchen der Werte und das Liefern der Ergebnisse an Nachfoleknoten.
- **Welche Typen von Operationsknoten kennen Sie?**
  Merger, Switch, Decider, Operator, F-Gate, T-Gate.
- **Was ist ein Datenflußrechner?**
  Ein Modell bei dem der Datenfluß durch Aktivitäts-Templates geregelt wird. Der besteht aus einer Aktualisierungseinheit, einem Speicher für Templates, einem Zusteller und einer Warteschlange für aktivierte Templates.
- **Was ist der Unterschied zwischen einer statischen und einer dynamischen Datenflußarchitektur?**
  Bei einer statischen Architektur wird ein Template aktiviert, wenn seine Daten verfügbar sind und ein anderes Template auf die Ergebnisse wartet. Bei dynamischen Architekturen entfällt die zweite Bedingung.
- **Warum können bei statischen Systemen die Kanten der Templates nur ein Token tragen?**
  Das Feuern kann (und wird) durchgeführt, wenn das Ergebnis angefordert wird (Rückwärtsgerichtete Knoten).
- **Was muss zur Realisierung von dynamischen Datenflußarchitekturen erfolgen?**
  FIFO-Warteschlangen an den Ausgaben.
- **Wo findet man das Prinzip der Datenflußarchitekturen?**
  Bibliotheken von komplexen Funktionen.

1.7 Analyse der Fließbandverarbeitung

- **Was ist der Unterschied zwischen räumlicher Parallelität und *Fließbandverarbeitung*?**
  Bei räumlicher Parallelität bearbeiten verschiedene Bearbeitungsstationen unterschiedliche Aufgaben. Bei der Fließbandverarbeitung bearbeiten Bearbeitungeinheiten immer die gleiche Bruchteile von Aufgabentypen.
- **Wie ist die Hardwarestruktur einer Fließbandverarbeitung aufgebaut?**
  Durch Register getrennte Stufen.
- **Welche Bedingung muß man an den Takt setzen?**
  Der muß länger sein, als die maximale Zeit in einer Stufe.
- **Welche Optimierungsziele verfolgt man bei der Erstellung von Fließbändern?**
  Die Verarbeitunszeit in den Stufen soll gleich sein und die Anzahl der Speicherplätze zwischen den Stufen soll klein sein.
- **Welche Fließbandanwendungen kennen Sie?**
  Arithmetik und Fließband.
- **Was ist der Unterschied zwischen einem statischen und einem dynamischen Fließband?**
  Bei statischen Fließbändern muß das Band freigemacht werden, damit eine Umschaltung erfolgt.
- **Welche Nachteile haben also statische Fließbänder?**
  Geringer Nutzfaktor, wenn zwischen Aufgaben umgeschaltet werden muß.

- **In welche Befehlsphasen kann man bei einem von Neumann-Rechner den Zyklus aufteilen?**
  Befehl holen, Befehl entschlüsseln, Operanden holen, Operationsausführung, Ergebnis abspeichern.
- **Was muß man beachten, wenn man beliebige Stufenbelegungen für die Verarbeitungsstufen betrachtet?**
  Man muß planen damit keine Kollisionen kommen.
- **Welche Annahmen trifft man für die Analyse?**
  Jede Verarbeitungszeit ist eine Periode und für jeden Auftrag gibt es einen Muster der Belegungen.
- **Was ist eine *Reservierungstabelle*?**
  Eine Tabelle, in der genau dann an einer Stelle i, j ein Eintrag a steht, wenn die Verarbeitungsstufe i zur relativen j mit dem Auftrag a beschäftigt ist.
- **Was ist eine *Ablauftabelle*?**
  Eine Tabelle in der beim Eintrag i, j der Wert ka genau dann steht, wenn zur absoluten Taktzeit j die Verarbeitungsstufe i mit dem Teilauftrag k des Auftrages a beschäftigt ist.
- **Was ist die Latenzzeit?**
  Konfliktfreier Abstand zwischen zwei aufeinanderfolgenden Aufträgen.
- **Was ist eine *Latenzfolge*?**
  Eine Latenzfolge ist eine periodische Folge von minimal zulässigen Latenzzeiten zwischen Aufträgen.
- **Was ist ein *Latenzzyklus*?**
  Die kürzeste sich wiederholende Teilfolge in einer Latenzfolge.
- **Was ist die *mittlere Latenzzeit*?**
  Durchschnitt der Werte in einem Latenzzyklus.
- **Welcher Zusammenhang gilt zwischen der mittleren Latenzzeit und der Anzahl der Markierungen je Stufe?**
  Mittlere Latenzzeit ist größer als maximum der Markierungen.
- **Wie analysiert man ein statisches Fließband?**
  Ermittlung des Kollisionsvektors, Erstellung des Übergangsgraphen, Ablesen der miminalen mittleren Latenzzeit.
- **Was ist ein Kollisionsvektor?**
  Vektor mit 1-Eionträgen dann, wenn die Initialisierungsabstand zum Konflikt führt.
- **Wie bestimmt man den Kollisionsvektor?**
  Man Initialisiert alles auf 0 bis auf den Eintrag 0. Dann bekommt die Stelle $j_2 - j_1$ den Wert 1 falls ein i existiert, sodass der i, j_1 Eintrag und i, j_2 Eintrag sind markiert.
- **Warum steht beim 0-Eintrag immer 1?**
  In einem statischen Fließband können zwei Aufträge nicht gleichzeitig initialisiert werden.
- **Wodurch sind die Einträge im Kollisionsvektor korrekt?**
  Wenn zur Zeit j_1 und zu j_2 zwei solche Aufträge mit dem Abstand $j_2 - j_1$ starten, dann ist die Stufe i von beiden Aufträgen belegt.
- **Welchen Zweck erfüllt der Übergangsgraph?**
  Er verhilft die Latenzzeiten zu erkennen.
- **Womit startet man die Konstruktion des Übergangsgraphen?**
  Mit dem Kollisionsvektor als ersen Zustand.
- **Wie erweitert man den Übergangsgraphen?**
  Für jeden 0-Eintrag bei i wird ein Nachfolgezustand erzeugt und die entsprechende Kante mit i markiert. D.h. nach i-Schritten kann wieder ein Auftrag initialisiert werden. Der neue Zustand ist wieder ein Kollisionsvektor.

- **Was drückt der Nachfolgezustand aus?**
  Kollisionsvektor relativ zur Startzeit des vorausgehenden Auftrages.
- **Wie bestimmt man diesen Vektor effizient?**
  Verschieben des alten Kollisionsvektors um Kantenmarkierung Stellen und bitweises Oder mit dem alten Vektor.
- **Wie verfährt man allgemein?**
  Man führt eine Liste nicht bearbeiteter Knoten zu der man alle Einträge hinzufügt, die 0-Wert haben. Für alle Knoten in dieser Liste bildet man die Nachfolgerknoten (Oder mit Wurzelknoten). Hat dieser 0-Einträge, so kommt er in die Liste nichtbearbeiteter Aufträge. Am Ende verbindet man jeden Knoten des erzeugen Graphen mit dem ersten Knoten durch eine Kante mit einer Markierung $>= m$.
- **Wie Analysiert man den Graphen?**
  Mittlere Latenzzeiten einfacher Zyklen bestimmen. Einfach sind Zyklen, in denen kein Knoten mehrfach vorkommt.
- **Wie kann man die Suche vereinfachen?**
  Indem man nach reflexiven Kanten im Graphen mit dem kleinsten Wert sucht.
- **Wie findet man solche?**
  Man sucht das kleinste k, sodass es keine natürliche Zahl i gibt mit $i * k$ ist verboten.
- **Welche Arten von Aufträgen können bei dynamischen Fließbändern kollidieren?**
  Aufträge des gleichen Typs und Aufträge verschiedener Typen.
- **Weclhe Semantik haben Kollisionsvektoren für dynamische Fließbänder?**
  Ein Kollisionsvektor $X -> Y$ sagt aus, wann Aufträge vom Typ X mit denen vom Typ Y in Konflikt kommen können.
- **Wie bestimmt man solche Kollisionsvektoren?**
  Analog zum statischen Fall nur für alle $X -> Y$.
- **Was ist der Unterschied bei der Auswertung des Graphen?**
  Man muß zusätzliche Annahmen über die Auftragsreihenfolge machen.
- **Wie kann man Reservierungsmuster optimieren?**
  Indem man Zwischenstufen einfügt.

1.8 Parallelverarbeitung

- **Auf welchen Ebenen kann die Parallelverarbeitung eingeführt werden?**
  Einprozessorsysteme und Mehrprozessorsysteme.
- **Welche Mittel zur Leistungssteigerung gibt es?**
  Fließbandbetrieb, Anzahl von Registern erhöhen, Cache, I/O-Prozessor, Parallelität.
- **Welche Begriffe charakterisieren die Parallelverarbeitung in Einprozessorsystemen?**
  Fließbandverarbeitung, Super-Skalar-Prozessor, Superpipeline-Prozessor und Very-Long-Instruction-Word-Prozesser.
- **Wie kann man Parallelverarbeitung mit Fließband verknüpfen?**
  Parallele Fließbänder. Parallele Ausführung von datenunabhängigen Aufträgen.
- **Wie kann man Parallelverarbeitung mit einem *Super-Skalar-Prozessor* realisieren?**
  Mehrere Verarbeitungseinheiten z.B. zwei ALU's usw. Überholen der Befehle möglich.
- **Wie kann man Parallelverarbeitung mit einem *Superpipeline-Prozessor* realisieren?**
  Höhere interne Taktfrequenz. Dadurch kann man Teilaufträge vereinfachen und mehr Fließbandstufen einführen.

- **Wie kann man Parallelverarbeitung mit einem *VLIWA* realisieren?**
  Verwendung von Befehlswörern sehr großer Länge. Unabhängige Befehle werden in jeweils ein Wort verpackt.
- **Welche Einheit übernimmt die Zusammenfassung von Wörtern in VLIWA?**
  Der Compiler.
- **Was versteht man unter *Trace-Scheduling*?**
  Parallelisierung entlang eines Programmpfades, von dem man hofft, dass dieser gewählt wird.
- **Welche Parallelitätsarten auf Prozessorebene kennen Sie?**
  SISD, *SIMD*, *MISD*, *MIMD*.
- **Wie kann man einen SISD Rechner beschreiben?**
  Das sind von Neumann-Rechner.
- **Was ist für die SIMD-Architektur charakteristisch?**
  Ein Befehlsprozessor beauftragt ein Feld von Datenprozessoren. Jeder der Datenprozessoren besitzt eigenen lokalen Speicher und arbeitet streng synchron mit anderen. Einzelne Prozessoren können ausgeschlossen werden.
- **Was ist bei SIMD-Architekturen kritisch?**
  I/O bzw. Verbindungsnetzwerk.
- **Wie wird Kommunikation bei SIMD-Architekturen durchgeführt?**
  send und recieve Befehle mit Ausschluss bestimmter Rechner.
- **Wie viele Prozessoren enthalten typische SIMD-Rechner?**
  Sehr viele 65536 1 Stellige z.B.
- **Was ist bei SIMD-Rechnern Probelmmatisch?**
  Systeme überleben nicht die technologischen Sprünge.
- **Welche MIMD-Rechner-Typen kennen Sie?**
  Nachrichtengekoppelt oder *Speichergekoppelt*.
- **Wie sind nachrichtengekoppelte MIMD-Rechner aufgebaut?**
  Eine über ein Netz verbundenen Ansammlung von von Neumann Rechnern.
- **Was ist bei nachrichtengekoppelten MIMD problemmatisch?**
  Netzwerk ist ein entscheidendes Merkmal. Programmierung muß Kommunikation umfassen.
- **Wie wird Synchronisation meistens durchgeführt?**
  Indem man eine blockierende recieve Funktion einbaut.
- **Welchen Kompromiss muß man bei MIMD-Rechnern schließen?**
  Parallelität und Synchronisationsaufwand.
- **Wie sind speichergekoppelte MIMD-Rechner aufgebaut?**
  Mehrere Rechner greifen auf einen gemeinsamen Speicher zu.
- **Was ist bei speichergekoppelten MIMD-Rechnern der Flaschenhals?**
  Der gemeinsame Speicher.
- **Wie erfolgt bei speichergekoppelten MIMD-Rechnern die Kommunikation?**
  Über den Speicher.
- **Muß man synchronisieren?**
  Ja z.B. über Barrieren-Synchronisation mit Hilfe einer geeigneten Hardware.
- **Wie kann man bei speichergekoppelten MIMD-Rechnern die Leistung aufbessern?**
  Durch einführung lokaler Speicher. Dabei muß man aber Konsistenz bewahren.

2 Maschinenbefehlssatz

- **Aus welchen Teilen besteht ein Maschinenbefehl?**
  Befehlcode, Operandenaddressen und Addresse des Folgebefehls.

2.1 Maschinenbefehlssätze

- **Wie kommt mann zum *Dreiaddressformat*?**
  Durch Weglassen des Nachfolgerbefehls.
- **Wie kommt man zum *Zweiaddressformat*?**
  Durch Überdeckung des Ergebnisses mit einem der Operanden.
- **Welche Folgen hat das *Zweiaddressformat*?**
  Man muß möglicherweise den Operanden an die Stelle transportieren, wo das Ergebnis abgespeichert wird. Neuer Befehlstyp MOV.
- **Wie kommt man zum *Einaddressformat*?**
  Durch das Prinzip der Implizierung der Daten. Einführung des Akkumulators. Neuer Befehlstyp LOAD und STORE.
- **Wie kommt man zum *Nulladdressformat*?**
  Durch spezielle Speicherorganisation mit einem Stack.
- **Welche Unterschiede gibt es zwischen Befehlsformaten?**
  Weniger Operanden bedeutet kürzere schnellere Befehle aber mehr davon und mehr Operanden bedeutet kürzeren Code, aber unterschiedliche längere Ausführungszeiten.
- **Welches System benutzt Intel 80x86?**
  Zweiaddresssystem mit einer Speicheraddresse.
- **Welche Vorteile und Nachteile haben Registermaschinen mit auschließlicher Registeraddressierung?**
  Gleiche Wortlänge, Einfache Compilerarbeit, Gleiche Befehlsausfürhungszeiten. Aber auch längerer Code mit umständlicheren Befehlskodierungen.
- **Welche Vorteile und Nachteile haben Zweiaddressmaschinen mit einer Speicheraddresse?**
  Zugriff auf Daten ohne Laden, Befehlsformat ist einfach zu kodieren, Kurze Befehle. Aber : Operanden sind nicht gleichberechtigt, Anzahl der Register wird durch den Bedarf an Addressstellen Begrenzt, Befehlsausführungszeit hängt von der Lage der Operanden.
- **Welche Vorteile und Nachteile haben Dreiadressmaschinen mit drei Speicherzellen?**
  Kompakter Code, Kein Registerverbrauch für Zwischenwerte. Unterschiede in der Instuktionslänge und große Variation der Ausführungszeit.
- **Was ist eine *Lade-Speichere Architektur*?**
  ALU Befehle benutzen keine Werte aus dem Speicher.
- **Wie beurteilen sie die *VAX*-Architektur?**
  Diese Architektur hat sich nicht durchgesetzt.

2.2 Sepeicheraddressen

- **Welche Besonderheiten bei der Addressierung des Speichers kennen Sie?**
  Alligned-Addressierung, *Big-Endian* und *Little-Endian* Darstellungen (Array-Datenobjekte und Felder haben aber gleiche Addressen).

2.3 Wortformate

- **Sind die Befehlswörter immer gleich lang?**
  Nein.
- **Wie begegnet man dem Problem, dass unterschiedliche Befehle unterschiedlich viele Stellen im Befehlswort benötigen?**
  Man führt spezielle Erweiterungsfelder ein, die die Interpretation eines weiteren Feldes zur Befehlsinterpretation hinzuziehen.

2.4 Maschinenbefehle

- **In welche Gruppen werden Maschinenbefehle unterteilt?**
  ALU, Tranfer, Kontrollfluß, System, Gleitkomma, String, I/O.
- **Welche Anforderungen kann man an Befehle stellen?**
  Orthogonale Befehle (Alle Kombinationen möglich), Elementar (nicht komplex).
- **Welche Befehle sind für Effizienz von Bedeutung?**
  Sprungbefehle und Prozeduraufrufe.
- **Welche Sprungdistanzen sind häufig in Programmen?**
  4-7 Speicherwörtern und meist nach vorne.
- **Wie realisiert man bedingte Sprünge aus der Programmierersicht?**
  Durch setzen von Bedingungscodes, Durch Ausschreiben in spezielle Register und durch Vergleich im Befehl.
- **Aus welchen Bestandteilen setzen sich Prozeduraufrufe zusammen?**
  Prozessorstatus retten. Parameterübergabe.
- **Wie behandelt man Prozeduraufrufe?**
  Entweder Oberprogram rettet die Daten oder Unterprogramm. Beim Unterpgramm bestet die Möglichkeit zu optimieren, indem man nur das nötige rettet.
- **Welche Probleme entstehen, wenn Unterprogramm selbst die Rettung der Daten übernimmt?**
  Bei verschachtelten Prozeduraufrufen bei getrennte Kompilierung kann es dazu kommen, dass falsch auf globale Variablen zugegriffen wird (wenn Oberprogramm die Daten im Sepeicher ändert und diese nicht rettet).

2.5 Einflüsse von Hochsprachen und Compilern

- **Welchen Einfluß haben Compiler auf die Rechnersturkturen?**
  Sie bestimmen, welche Struktur und Befehlsformen sich durchsetzen.
- **Was ist ein Basisblock, Maximaler Basisblock?**
  (Maximaler Block, sodass) Jeder Befehl hat genau einen Nachfolger und Vorgänger-Befehle.
- **Welche Eigenschaften haben solche Basisblöcke?**
  Sie besitzen keine Schleifen.
- **Welche Optimierungen können bei maximalen Basisblöchken vorghenommen werden?**
  Mehrfachberechnung von Teilausdrücken, Propagation von Konstanten, Reduktion der Höhe des Ausdrucksbaumes.
- **Welche globale Optimierungen kann man bei Programmen machen?**
  Elimination von Mehrfachberechnungen, Propagation von Identitäten, Rausziehen von schleifeninvarianten Berechnungen.
- **Welche maschinenabhängige Optimierungen kennen Sie?**
  Ersetzen z.B. von Multiplikationen mit 2 durch Schiften. Befehlsreihenfolge für Fließkomma. Sprungdistanzen optimieren.

- **Kann man (naheliegend) globalen Optimum erreichen?**
  Optimum nicht. Das Problem ist NP-Vollständig und multimodal. Mit guten Heuristiken kann man aber sehr gute Ergebnisse erreichen.
- **Welches zentrale Problem spielt bei RISC-Maschinen eine Schlüsselrolle bei der Optimierung?**
  Registerzuweisung.
- **Wie erfolgt Registerzuweisung?**
  Bestimmung eines Interferenzgraphen über live-in und live-out Mengen. Auswahl der Variablen für den Arbeitsspeicher. Vereinfachung und Einfärben des Graphen durch Optimierung.
- **Wie ist eine Definitionsstelle und eine Verwendungsstelle definiert?**
  Definitionsstelle ist ein Knoten in einem Ausführungsgraphen, in der eine Variable auf der linken und Verwendungsstellen auf der rechten Seite auftritt.
- **Was ist *Lebendigkeit*?**
  Eine Variable v ist auf einer Kante lebendig, wenn die Kante auf einem Pfad zu einer Verwendungsstelle führt, welcher nicht über eine Definitionsstelle von v führt.
- **Was bedeutet *live-in* und *live-out*?**
  Eine Variable ist in einem Knoten *live-in* bzw. *live-out*, wenn sie auf jeder Eingangs- bzw. Ausgangskante des Graphen lebendig ist.
- **Wie kann man die Lebendigkeitsinformation aus Definitionsstellen und Verwendungsstellen ableiten?**
  In einer Verwendungsstelle einer Variable v ist v live-in. Falls eine Variable in einem Knoten live in ist, so ist sie live-out in allen Vorgängerknoten. Falls eine Variable in einem Knoten live-out ist aber der Knoten ist keine Definitionsstelle der Variable, dann ist die Variable auf in dem Knoten live-in.
- **Welche Bedingungen kann man aus Lebendigkeitsdefinietionen ableiten?**
  Die Menge der in einem Knoten lebendigen Variablen ist die Menge der verwendeten Variablen vereinigt mit allen solchen live-out Variablen, die in diesem Knoten nicht definiert werden. Die Menge der live-out Variablen ist die Vereinigung aller live-in Variablen der Nachfolgerknoten.
- **Wie kann man obige Bedingungen lösen?**
  Fixpunktiteration indem man mit leerer Menge für jeden Knoten anfängt.
- **Was ist eine sinnvolle Vorgehensweise bei der Berechnung der live-in und live-out Mengen?**
  Fixpunktiteration mit Knoten teif im Ausführungsgraphen anfangen.
- **Welche Grundlage liegt dem *Interferenzgraphen* zugrunde?**
  Zwei Variablen, die zu einem Zeitpunkt (Knoten) beide lebendig sind können nicht dem selben Register zugewiesen werden.
- **Wie ist die Bildungsvorschrift des Interferenzgraphen?**
  Falls ein Knoten die Definitionsstelle einer Variablen ist füge eine Kante zu allen Variablen in der live-out Menge.
- **Welche Optimierungsaufgabe liegt der Vereinfachung zugrunde?**
  Einfärbung durch möglichst wenige Farben (bei einer gegeben Anzahl von Farben = Registern ist das Problem i.A. unlösbar).
- **Wie verfährt man bei der Optimierung von Interferenzgraphen?**
  Ein Knoten, der weniger als die Anzahl verfügbarer Register besitzt kann weggelassen werden. So kann man so lange Knoten entfernen, bis alle mehr als die Anzahl verfügbarer Register Nachbarn haben.
- **Wie kann man noch den Interferenzgraphen optimieren?**
  Durch Zuweisen vom Speicherzellen zu Knoten. Dann entfällt jeweis ein solcher Knoten.

- **Wie speichert man dabei die entfernten Variablenknoten?**
  Aud einem Stack.
- **Wie erfolgt die Farbenzuweisung?**
  Für die Variablen auf dem Stack wird jeweils eine neue Farbe zugewiesen. Dabei wird der Graph wiederaufgebaut.
- **Was kann man machen, wenn die Neuzuweisung nicht möglich ist?**
  Man muß das Programm umschreiben, indem man die Variable sofort nach der Definition in den Speicher schreibt.
- **Wie kann man Fragestellungen nach Compilerstategien und Maschinenbefehlsauswahl beschreiben?**
  Registerzahl, Variablenposition, Addressierung, Einfluß von Optimierungstechniken auf die Verfügbarkeit, Knotrollstrukturen und Häufigkeit.
- **Wo sind Verbesserungen und Optimierungen bei Compilern zu erwarten?**
  Transport und Rechenaufgaben.

2.6 Spezielle Maschinenbefehle

- **Welche Sonderbefehlstechnologien kennen Sie?**
  MMX bei Intel, 3Dnow bei AMD.
- **Welche Aufgabe haben diese Techniken?**
  Neue Befehle zur Verarbeitung von Multimedia. Z.B. die selbe Operation parallel auf verschiedenen Daten.
- **Was ist eine Besonderheit des *MMX*-Befehlsatzes?**
  Sättigungs-Arithmetik.
- **Kennen Sie ein typische Anwendung für MMX-Befehlssatz?**
  Überblendung von Bildern in Filmen.

3 Zentralprozessoren

- **Wie ist ein *Zentralprozessor* aufgeteilt?**
  In *Befehlsprozessor*, in *Datenprozessor* und in zusätzliche Speichereinheiten.
- **Welche Hierarchie besteht in Prozessorsystemen?**
  Zentralprozessor (Oben), Befehlsprozessor (Entschlüsselung des Befehls, Operandenfeststellen), Mirkobefehlsprozessor (Mikroprogramm ausführen), Nanobefehlsprozessor (Logische Schritte des Datenprozessors), Picobefehlsprozessor (Physikalische Schritte) und Datenbefehlsprozessor.

3.1 Datenprozessoren

- **Sind alle Register im Datenprozessor direkt durch Maschinenbefehle addressierbar?**
  Nein. Z.B. der Interruptaddressenregister oder Speicheraddressenregister oder der Befehlszähler.
- **Wie ist ein einfacher Datenprozessor aufgebaut?**
  Aus zwei Eingabebussleitungen, einer Ausgabebusleitung, einem Registerspeicher und einer ALU (die dazwischen liegen) und Kontrollregistern (PC,SAR,SDR).
- **Welche zusätzliche logische Einheit besitzt die ALU?**
  Eine Schiebeeinheit.
- **Was bewirkt unter anderem lange Signallaufzeiten in ALU?**
  Stellenüberträge bei Additionen, wenn man diese Ortssequentiell implementiert.

- **Wann entsteht naiv beim Addieren ein Übertrag?**
  Wenn mindestens zwei der Eingaben a,b oder c (c ist der vorhergehende Übertrag) mit 1 belegt sind.
- **Welche Idee liegt hinter dem vereinfachen des Netzes?**
  Bildung der disjunkitven Normalform durch Einsetzen der Bedingungen für den Übertrag ineinander und Zerlegung der Additionsfunktion in einen ortsparallelen und ortssequentiellen Anteil.
- **Wann entsteht ein Ausgangsübertrag nach dieser Vorgehensweise?**
  Wenn entweder einer der Eingänge 1 und der andere 0 ist, so kann man den Übertrag propagieren. Wenn aber beide Eingänge 1 sind, so ist der Übertrag immer auch 1.
- **Welche Darstellung für den Übertrag erhält man?**
  Generieren oder propagieren, falls alter Übertrag vorhanden $g_i \vee c_i p_i$.
- **Wie kann man eine Darstellung der Formel für den Übertrag interpretieren?**
  Man propagiert den Übertrag von einer der Stufen weiter, wenn alle Stufen davor einen Übertrag propagiert haben oder generiert einen neuen in aktueller Stufe.
- **Wie nennt man dieses Prinzip?**
  Carry-look-ahead.
- **Wie kann man den carry-look-ahead Addierer für große Stellenzahlen implementieren?**
  Indem man immer Terme mit dem Eingangsübertrag ausklammert und eine hierarchische Baumstruktur erzeugt.
- **Aus welchen Teilen besteht als die Bildung eines Carry-Lookahead Addierers?**
  Darstellung des Terms durch Generate- und Propagate-Funktionen. Hierarchisierung durch Betrachten von Anteilen unabhängig vom Eingangsübertrag und abhängig.
- **Welchen Komplexitätsvorteil hat ein carry-look-ahead Addierer?**
  $\log_2 n$ statt $2n$. $O(\log_2 n)$ und Raumkomplexität $O(n \log_2 n)$.
- **Können Sie einen carry-look-ahead Addierer skizzieren?**
  Ja. Übung!
- **Wie funktioniert ein *carry-select* Addierer?**
  Man berechnet in den höherwertigen Stufen den Wert des Übertrages für beide Fälle des Eingangsübertrages. Daher muß das Netz auf die Berechnung des Übertrages von niederen Stufen nicht warten.
- **Welche Komplexität hat der carry-select Addierer?**
  Halbe Laufzeit beim doppelten Flächenbedarf. $O(\sqrt{n})$ Zeit bei $O(n)$ kosten.
- **Wie ist die Fließkommaaddition generell aufgebaut?**
  Angleich der Exponenten durch Verschiebung, Addition, Normalisierung und Zusammenfügen. Diese Befehle können parallel durchgeführt werden.
- **Welche Stufen würde man in einer Fließkommaaddition einbauen?**
  Vergleich und Addition der Exponenten, Addition der Signifikaten, Normierung der Exponenten, Normierung der Signifikanten.
- **Wie realisiert man die Register bei CISC und RISC?**
  Bei CISC sind wenige (8) Register und zusätzlich Steuerregiseter in einem Registersatz. Bei RISC gibt es mehrere Möglichkeiten: *Großer Registersatz* und *Registerfenster*.
- **Wann kann man Registerfenster verwenden?**
  Wenn die Anzahl der benötigten Register gut abschätzbar ist.
- **Welches Problem entsteht bei Registerfenstern bei rekursiven Aufrufen?**
  Die Unterprozedure-Tiefe kann sehr hoch sein. Dazu macht man die Registerfenster zirkulär.
- **Wie ist das Problem bei einem Sparc-Prozessor gelöst?**
  Die Registerfenster überlappen sich genau in den Eingabe/Ausgabeparametern.
- **Wie kann man den Registerplatzmangel beseitigen?**
  Man kann Registerfenster variabler Größe einführen.

3.2 Befehlsprozessor

- **Wie wird die Maschinensprache interpretiert?**
  Überführung in eine Mikroprogrammsprache (Mikrobefehlsprozessor) und dann in die Steuersprache der Logikebene(*Picoprogrammsprache*, steuerung des Datenprozessors).
- **Welche Addresstypen stehen im PC?**
  Prozessaddressen meistens (allerdings nicht notwendig).
- **Welche Arten von Informationen enthalten die Befehle auf den Ebenen?**
  Was getan werden soll, Mit welchen Werten gearbeitet werden soll (*Addressberechnungswerk*) und Information über die Addresse des nächsten Befehls.
- **Was ist ein *Mikrprogramm*?**
  Ein Mikroprogramm beschreibt die Interpretation eines Maschinenbefehls. Die Menge aller Mikroprogramme heißt *Emulator*.
- **Wie werden in verschiedenen Architekturen die Emulatoren beschrieben?**
  Bei RISC durch Hardwarerealisierung und bei CISC durch Firmware.
- **Was ist eine *SWING*-Archikektur?**
  Oft verwendete Befehle durch Hardware und selten verwendete durch Firmware.
- **Welche Aufgaben erfüllt der *Mikrobefehlsprozessor*, wenn die Picobefehlsebene fehlt?**
  Er holt sich den namen des Befehls und erzeugt parallel mehrere Folgen von Picobefehlen. Danach eine Rückmeldung "Befehl holen".
- **Wodurch werden Mikrobefehlsprozessoren realisiert?**
  Durch endliche Automaten.
- **Was ist ein *Mykrozyklus*?**
  Takt des Mikorbefehlsprozessors.
- **Welche Ziele verfolgt man bei der Realisierung von Mikrobefehlsprozessoren?**
  Verkleinerung des Speichers für Mikroprogrammspeicher.
- **Wie kann man die Transformation von Programmnamen in Startaddresse realisieren?**
  Über eine Festwertetabelle mit einem Multiplexer für Fehlercode.
- **Wovon hängt die Zulässigkeit eines Befehls?**
  Vom Systemmodus.
- **Welche Rückmeldungen bekommt der Mikrobefehlsprozessor?**
  Statusanzeige des Rechenwerks, Arbeissrpeichers, I/O.
- **Was ist der unterschied zwischen *Programmaddressen* und *Maschinenaddressen*?**
  Erstere sind relativ, die anderen sind physikalisch.
- **Wie kann man die Problemmatik der Addressierung im Mikrobefehlsprozessor lösen?**
  Reduktion der Anzahl der Addressen, über die Daten in einem Mikroprogrammwort zu speichern sind. Einführungs des Befehlszählers und Relativaddressierung.
- **Wie funktioniert *seitenrelative Addressierung*?**
  Man zerlegt den Addressraum in disjunkte Seiten.
- **Wie Funktioniert *direkte Steuerung*?**
  Jeder Befehl hat eine eigene Stelle im Befehlswort. Nicht durchführbar, da zu viele Befehle.
- **Wie kann man Entschlüsselung vornehmen?**
  Sich ausschließende Befehle können kompakt kodiert werden (*horizontale Steuerung*). Dekodierung erfolgt über Multiplexer. Oder durch kodieren aller möglichen Anweisungen dual und Interpretation mit einer Tabelle (*quazi horizontale Steuerung*). Ein Befehl pro Auftrag (*vertikale Steuerung*).

- **Welche Nachteile können Interpretationstabellen haben?**
  Große Hardwareumwandlungen im Falle von Designänderungen.
- **Welche Ziel verfolgt man bei der Entschlüsselung grundsätzlich?**
  Kurze Codewörter und geringer Hardwareaufwand ohne die Parallelität aufzugeben.
- **Welche Bedeutung haben Firmware-Emulatoren?**
  Man kann zwischen den Befehlssätzen unterschiedlicher Maschinen wechseln. Kompatibilität.
- **Warum ist Änderung der Mikroprogramme seitens des Benutzers problematisch?**
  Mikroprogramme sind schwer zu schreiben, Speicherkapazität ist beschränkt, Mehrprogrammbetrieb macht das Nachladen von Mikroprogrammen nicht attraktiv.

3.3 Maschinenbefehlsprozessoren

- **Welche Aufgaben löst der *Maschinenbefehlsprozessor*?**
  Befehlsaddressbestimmung, Operatorbestimmung und Operandenbestimmung. Die wichtigste Aufgabe ist: Addressübersetzung aus Programmaddressen in physikalische Addressen.
- **Welches Berechnungsmodell entspricht einem Maschinenbefehlsprozessor?**
  Endlicher Automat.
- **Welche Forderungen stellt man beim Entwickeln von Maschinenbefehlsprozessoren?**
  Modifikation der Prozessaddressen ohne Änderung der Programmaddressen (Addressierungen), Kurze Programmaddressen (Lokalität), Unterstützung des Ladeprogramms (Basis-Addressierung), Lageunabhängigkeit von Programmen (Verdeckte Basisregister), Verschiebung Überflüssig (Seitenaddressierung), Variable Abbildung von Prozessaddressen (Virtueller Speicher), Unabhängige Prozessaddressräume (Segmentierung).
- **Wieviele Addressebenen gibt es normal?**
  Physikalische, Prozess- und Programmaddresse.
- **Welchen Zweck erfüllt die Zweistuffigkeit?**
  Multiprogramm-Prozessoren.
- **Welche *Programmaddressen* gibt es?**
  Symbolische und numerische.
- **Welche zentralen Forderungen stellt man an Prozessaddressen?**
  Möglichkeit der Modifikation von Prozessaddressen ohne der Programmänderung (*Ablaufinvarianz*) und *Zeitlokalität* und *Programmlokalität*.
- **Welche Beispiele für Ablaufinvarianz kennen Sie?**
  Unterprogrammparameter, Mehrfachinkarnationen, Datenstrukturen unbekannten Platzbedarfes, Berechnete Indizes in Feldern.
- **Wie kann man diese Probleme lösen?**
  Registerindirekte Addressierung (Bearbeitung von Feldern, Interpretation von Registerinhalten als Addressen), Basisaddressierung(Basis+Offset), Speicherindirekte Addressierung (Interpretation von Speichern als Addresse) und indizierte Addressierung (dual zu Basisaddressierung).
- **Wie kann man *Raumlokalität* erzwingen?**
  Indem man Addressierung relativ zum aktuellen Befehlszählerabstand macht oder zu einem offenen Basisregister.
- **Welche Aufgaben hat die *MMU*?**
  Transformation von Prozess- in Maschinenaddressen.
- **Wie und wann müssen die Addressen übersetzt werden, wenn die Prozessaddresse gleich der Maschinenaddresse ist?**
  Beim Ladenn des Programms spätestens. Das bildet ein Problem, denn die Übersetzung beim laden ist Aufwendig und erschwert den Betrieb.

- **Wie kann man das Ladeprogramm unterstützen?**
  Indem man befehlszähler-relative Addressierung und offene Basisregisteraddressierung benutzt.
- **Welches Problem besteht, wenn ein Programm in den Speicher geladen ist?**
  Das Programm ist nicht verschiebbar. Das führt dazu, dass der Speicher fragmentiert wird.
- **Wie kann man Programme lageunabhängig machen?**
  Verdeckte Basisregister im Addressberechnungswerk. Alle Prozesse verwenden den Speicher von 0 und aufsteigend. Besisregister wird beim Prozesswechsel umgesetzt (Transparenz)
- **Welche Vorteile hat Addressumsetzung?**
  Setzen von Addressen im Ladeprogrmm entfällt. Man spart die Angaben über lageunabhängige Addressen.
- **Wie kann man die Fragmentierung des Speichers aufheben?**
  Durch *Seitenaddressierung*. Einteilung des Speichers in Seiten. Prozessaddresse besteht aus einem *Seitenindex* und einem *Wortindex*.
- **Wie regelt man bei Seitenaddressierung die Zugriffsrechte?**
  Man orndet den Seiten die Rechte zu.
- **Wie organisiert man Seitenaddressierung effizient?**
  Man ordnet die Seitentabelle in Registern, Ein zurletzt benutzter Ausschnitt wird in einem Pufferspeicher *TLB* (*Assoziativspeicher*) gelagert.
- **Wie ist *virtueller Speicher* organisiert?**
  Der Prozessraum ist virtuell größer als der physikalische Raum. Will ein Prozess auf eine Addresse zugreifen, die nicht im Speicher ist übernimmt das Betriebssystem die Addressierung und lädt den Speicher nach.
- **Was ist der Unterschied zwischen *Demand-Paging* und *Prepaging*?**
  Bei Prepaging werde vorausschauend Seiten mehr geladen, von denen man glaubt, dass sie verwendet werden könnten.
- **Welche grundlegenden Möglichkeiten bestehen beim Addressieren von Auslagerungspeichern?**
  Eine Seiten-Seitenrahmentabelle pro Prozess, Zweistufige Verfahren, Assoziativspeicher.
- **Was spricht für große und was für kleine Seitentabellen?**
  Verwatungsaufwand beim Zugriff auf Festplatte und kleinerer Assoziativspeicher sprechen für große Seiten. Aktualität und Verschnitt (unbenutzte Regionen) sprechen für kleine Seiten.
- **Welche Problemme entstehen beim *statischen Binden*?**
  Sich verändernde Datenbereiche, Korrktur von Teilprogrammen erfordert neue Bindung, zusammenhängender Programmaddressraum führt zu einem solchen Prozessaddressraum.
- **Wie kann man unabhängige *Prozessaddressräume* realisieren?**
  Durch *Segmentieren*. Man unterteilt den Speicher in Segmente unterschiedlicher Größen.
- **Welche Probleme haben Segmentaddressierungen?**
  Vollständiges Laden von Segmenten. Speicher muß eine geeignete Lücke aufweisen damit das Segment reinpasst.
- **Welche Ziele verfolgt also die Segmentierung?**
  Mehrere lineare Addressräume, Trennung und Schutz von Programmen und Daten, dynamisch wachsende und schrumpfende Bereiche, gemeinsame Nutzung vom Programmcode, Übersichtliche Gliederung des Prozessaddressraumes.
- **Welche Nachteile hat die Segmentaddressierung?**
  Zusätzlicher Platz, großer Verwaltungsaufwand.

- **Aus welchen Teilen besteht die Addresse?**
  Segmentindex, Seitenindex, Wortindex.
- **Welche Aufgaben erfüllen Unterbrechungen?**
  Fehler (z.B. I/O), Anforderung eines Dienstes, Schrittweise Programmausführung, Zeitscheibenzuteilung, Seitenfehler, Fehlerhafte Speicheraddresse, Spannungsversorgung, Zugriffsrecht.
- **Welche Lösungen für Unterbrechungen kennen Sie?**
  Sodtware-Basis und Hardware-Basis.
- **Was wird bei der Unterbrechungsbehandlung gemacht?**
  Retten des Prozessorstatus, Ausführen des Interrupt-Handlers.
- **Nach welchen Kriterien kann man Unterbrechungen unterscheiden?**
  Synchronität (vom aktuellen Prozesszustand abgeleitet), Vom Benutzerprozess angefordert, Maskierbar, Ausführung am Anfang oder am Ende des befehls, Unterbrechung oder Beendigung des Prozesses.
- **Was ist zu beachten bei Unterbrechungen?**
  Bei langen Instruktionen muß die Addressfortschaltung (mit Seitenfehlern) des Befehlszählers/Stackpointers rückgängig gemacht werden, Fließbandverarbeitung, Unterteilung eines langen Befehls in Phasen.
- **Welche typischen Unterbrechungen kennen Sie beim Speicher?**
  Seitenfehler beim Speicherzugriff.

4 Fließbandverarbeitung

4.1 Einführung

- **In welcher Form kommt Parallelität in Rechnerstrukturen vor?**
  Mehrrechnersysteme, Mehrprozessorsysteme, Fließbandverarbeitung.
- **Steht bei der Parallelisierung von Maschinenbefehlen nur die Fließbandverarbeitung zur Verfügung?**
  Nein. Man kann auch auf Mehrrechnern parallelisieren oder in parallelen Einheiten.
- **Wie ist aus der Sicht des Programmierers die Fließbandverarbeitung zu erfolgen?**
  Transparent. Höchstens der Compiler sollte Teile übernehmen.
- **Welche Grundlage bietet das von Neumann-Prinzip für die Fließbandverarbeitung?**
  Der nächste Befehl ist mit hoher Wahrscheinlichkeit bekannt.
- **In welche Phasen kann man eine Befehlsverarbeitung gliedern?**
  Befehl holen, Befehl entschlüsseln und Operanden aus Registerspeicher holen, Befehlsausführung, Speicherzugriff, Zurückschreiben. Man beachte die komische Reihenfolge.
- **Welche Modifikationen an der Beispielbusstruktur müssen vorgenommen werden?**
  Prozessorinternen busse werden durch direkte Leitungen ersetzt. Registerspeicher erhält Register für die beiden Ausgelesenen Operanden. Befehls- und Datenspeicher werden getrennt (und auch die zugehörigen Register). Befehlszähler wird extern realisiert. Zusätzliches Netz für einen Vergleich mit 0. Kopien der Register PC, IR usw. müssen für folgende Phasen bereitgestellt werden. Erweiterung des Direktoperanden wird in einem Schaltnetz realisiert.
- **Welche Funktion erfüllt in der Skizze das Zeichen *SGN-EXT*?**
  Die kurzen lokalen Addressen sind Vorzeichenbehaftet. SGN-EXT ist ein Schaltnetz, welches die Erweiterung auf vollständige Addresse durchführt.

4.2 Hazards bei der Fließbandverarbeitung

- **Was ist ein *Hazard*?**
  Situation in der die Ausführung des nächsten Befehls in einer Fließbandverarbeitung verhindert ist.
- **Welche Hazards kennen Sie?**
  Ressourcenhazards (*Kollision*), Datenhazards und Kontrollhazards?
- **Warum kann man nicht in den Registerspeicher bei den einzelnen Stufen schreiben?**
  Das würde eine sehr genaue Taktung erfordern, mit der Cache z.B. schlecht synchronisiert werden kann.
- **Wie kann man Fließbandverarbeitung bewerten?**
  Indem man die mit der selben Struktur vergleicht mit der Ausnahme, dass jeder Befehl komplett abgearbeitet werden muß bis der nächste kommt.
- **Wie kann man den Idealzustand bei einem Fließband beschreiben?**
  Das Band ist zu keiner Zeit unbeschäftigt.
- **Ist ein Idealzustand realisierbar?**
  Nein. Es gibt Initialisierungszeiten z.B.
- **Durch welche Größen wird beim Fließband Gewinn gemessen?**
  Durch Verhältnis der mittleren Ausfürhungszeiten und durch Durchsatz.
- **Wie kann man bei sequentieller/paralleler Ausführung die Ausführungszeit von n Befehlen berechnen?**
  Anzahl der Befehle mal die Anzahl der Stufen mal die Taktzeit (seriell). Einmaliges Durchlaufen plus $n-1$ Schritte, weil der Erste Befehl komplett durchgeht und dann in jeweils Stufen-Zeit-Abständen weitere folgen (Analog zu einer Leitung).
- **In welcher Größenordnung befindet sich der Gewinn?**
  In etwa die Anzahl der Stufen (mal verhältnis der Taktzeiten).
- **Warum ist die Betrachtung der Taktzeit wichtig?**
  Weil sowohl die Vergrößerung der Anzahl der Stufen als auch Einbau von Registern sich auf die Taktzeit auswirkt.
- **Verstehen sie das wie die Taktperiode berechnet wird?**
  Nein. Muß ich später nachschauen.
- **Wie berechnet man den Durchsatz?**
  Als Verhältnis der Anzahl der Befehle zu Verarbeitunszeit dieser Befehle.
- **Welche Größenordnung hat der Durchsatz im sequantiellen/Fließbandbetrieb?**
  Reziproke der Verarbeitungszeit eines Befehls im sqeuentiellen. Im Fließbandbetrieb das n-*Speedup*-fache (Verhältnis der mittleren Ausführungszeiten in einer Foge von n Befehlen).
- **Wie kann man die Anzahl der Fließbandstufen bewerten?**
  Z.B. mit dem Verhältnis des maximalen Durchsatzes durch die Fließbandkosten.
- **Was erreicht man mit *Bubbles*?**
  Verzögerung der Befehle um Kollisionen zu vermeiden.
- **Was sind *Ressourcen-Hazards*?**
  Diese entstehen, wenn z.B. Datenleitungen von mehreren Stufen benötigt werden.
- **Was kann man gegen Ressourcen Hazards machen?**
  Bubbles einfügen.
- **Was sind *Datenhazards*?**
  Diese entstehen, wenn ein Zugriff auf Daten erfolgt, die noch nicht verfügbar sind.

- **Welche Bedingung muss eingehalten werden um Datenhazards zu eliminieren?**
  Die Summen aus der Schreibphase des Befehls i und seiner Fetchzeit kleiner sein als die Summe der Fetchzeit des nächsten Befehls und der Lesephase des Befehls i.
- **Wie kann man die Bedingung anschaulich interpretieren?**
  In dem Zeitdiagramm muss der Lesebefehl rechts (später) als der Schreibbefehl liegen.
- **Was ist der Unterschied zwischen Einführen von NOP Befehlen und Blasen?**
  NOP-Befehler müssen auch vom Speicher geholt werden. Blasen führen zu aufwendiger Hardware.
- **Wie können Datenhazards gelöst werden?**
  Umordnung von Befehlen durch den Compiler, Einfügen von NOP-Knoten, Einsatz einer Hardwareeinheit, die Bubbles einfügt, Verwendung einer Einheit, die die Daten früher bereitsstellt.
- **Was ist eine *Bypass*-Schaltung?**
  Eine Schaltung, die bereits berechnete Werte den anderen Schaltungen bereitstellt.
- **Welche Typen von Datenhazards kennen Sie?**
  RAW, *WAW* (nur wenn Werte in unterschiedlichen Stufen verändert werden), *WAR* (nur wenn geschrieben werden kann, bevor gelesen wird).
- **Können alle Hazards ohne Einführung einer Blase gelöst werden?**
  Nein. Bei Speicherzugriffen müssen Blasen eingeführt werden. Man muss Blasen einfügen.
- **Wie erfolgt die Beseitigung von Hazards auf Compiler-Ebene?**
  Duruch Umordnen der Befehle (LP durch Linearisierung) und einfügen von NOP-Befehlen.
- **Welche Befehle verursachen *Kontrollflußhazard*s?**
  Verzweigungen.
- **Welchen Anteil von Verzweigungen weist man Programmen nach?**
  In etwa 30%.
- **Warum sind Sprungbefehle im Modell aus der Vorlesung besonders Problemmatisch?**
  Die Addressenberechnung für den Sprung erfolgt erst in dritter Phase.
- **Was muß man machen, wenn der Verzweigung gefolgt wird?**
  Falls man das Fließband weiter laufen ließ, so muss man die Ergebnisse der bereits ausgeführten Phasen rückgängig machen.
- **Wie kann man Kontrollhazards noch begegnen?**
  Blasen einfügen, bis klar ist ob der Verzweigung gefolgt wird.
- **Wie kann man die mittlere Ausführungszeit berechnen?**
  Erste Fallunterscheidung nach Wahrscheinlichkeit b, ob ein Verzweigungsbefehl kommt. Wenn einer kommt noch eine Falunterscheidung nach Wahrscheinlichkeit p, ob gefolgt wird oder nicht. Somit $(1-b)*A+b*(p*B+(1-p)*C)$.
- **Was kann man gegen Kontrollhazards machen?**
  Änderungen in der Fließbandstruktur, Compileroptimierung, Sprungvorhersage.
- **Wie kann man die Fließbandstruktur verändern?**
  Man zieht die Berechnung der Sprungaddresse in die zweite Stufe vor. Dazu wird ein zusätzliches einfaches Addiernetz implementiert.
- **Wie kann man mit Compilermittlen Kontrollhazards verhindern?**
  Man verwirft Befehle, die nach dem Sprungbefehl aufs Band kommen nicht und führt sie aus. Dadurch kann man das Band mit sinnvollen Befehlen laden. Sonst kann man auch per Compiler der Hardware in einem Speziallbefehl mitteilen, welche der Verzweigungen meist genommen wird.
- **Wie kann man *Branch-Prediction* betreiben?**
  Man führt einen assoziativen Sprungzielpuffer, der nach mindestens einmaliger Ausführung die Sprünge vorhersagt.

- **Welche Predictor-Typen gibt es insgesamt?**
  Einstellige, zweistellige und Correlations-Basierende.
- **Wie sollte man bei Branch-Prediction die Informationen speichern?**
  Man kann entweder mit einer oder mit zwei Stellen kodieren für jeden Sprungbefehl. Als Kodierungsgrundlage kann man einen endlichen Automaten verwenden.
- **Welche Arten von zweistelligen Vorhersagen kennen Sie?**
  Sättigungszähler und Verlasen des schwachen zustandes, sobald er falsch vorhersagt.
- **Was ist an einfachen Vorhersagen schlecht?**
  Es gibt Beispiele, wo die Vorhersage immer falsch ist.
- **Wie kann man abhängige Sprungvorhersagen realisieren?**
  Mit *correlation-based predictors*. Man geht hier von der Annahme, dass das Folgen dem Vorgängerbefehl und dem aktuellen zusammenhängen. DerZustand A/B sagt aus: Wurde dem letzten Befehl gefolgt, so steht links die Vorhersage, sonst rechts. Im Zustand negiert sich die Vorhersage.
- **Welche Eigenschaft erfüllt obige Einheit?**
  Man kann die Zustände getrennt voneinander ändern.
- **Welches Problem ergibt sich bei Unterbrechungen?**
  Ein später gestarteter Befehl kann in einer Stufe eine Unterbrechung auslösen bevor es ein früher gestarteter Befehl tut.
- **Welche Unterbrechungen können in welchen Stufen passieren?**
  0: Seitenfehler, Speicherschutzverletzung, Fehlerhafte Addresse. 1: Nicht erlaubter Befehlscode. 2: Arithmetikfehler. 3: wie in 1 mit Werten.
- **Welche Probleme sind bei der Unterbrechungsbehandlung zu lösen?**
  Zeitpunkt zum Einspeisen der Unterbrechungsbehandlung, Festlegung der auszuführenden Befehle, Unterdrückung von Schreibaufträgen von nicht mehr auszuführenden Befehlen.
- **Wie arbeitet die Methode *precise exception*?**
  Eine Fehlerbehandlung heißt präzise, wenn alle Befehle vor dem unterbrechenden Befehl ausgeführt wurden, alle danach nicht ausgeführt wurden und der Befehlszähler weist auf den Verursachenden Befehl hin.
- **Welche Idee steckt hinter precise exceptions?**
  Man schiebt die Unterbrechungsausführung auf das Ausführungsende der Befehle. Man sammelt die Unterbrechungsvektoren dabei in einem Speicher ab um sie gerecht auszuführen.
- **Wie kann man Fließbandsysteme für komplexe Operationen erweitern?**
  Man führt mehrere Einheiten, die spezialisiert sind auf bestimmte Operationen.
- **Welche Probleme entstehen dabei?**
  Längere Hazardzeiten, Mehr als ein Schreibbefehl auf den Speicher, WRW sind möglich, Schwierigere Fehlerbehandlungen, RAW Hazards häufiger.
- **Welche Idee steht hinter *dynamischen Scheduling*?**
  Unterschiedliche Ausführung der Befehle.
- **Wie kann man dynamisches Scheduling realisieren?**
  Man benötigt Tabellen, die Datenabhängigkeiten untersuchen, Ressourcenzuteilung vornehmen und den Status der Ausführung der Befehle enthalten.
- **Welche Aufgaben erfüllt das *scoreboard*?**
  Genau die obigen.
- **Kann in jedem Befehl der Fließbandverarbeitung eine Unterbrechung aufkommen?**
  Nein. Dazu muss was gemacht werden, z.B. ein Speicherzugriff.

5 Speicherhierarchie

5.1 Technische Grundlagen

- **In welche Gruppen kann man Speicher unterteilen?**
  Interne Speicher, Pufferspeicher, Arbeitsspeicher, Hintergrundsspeicher (Festplatte).
- **Wie unterscheidet man aus physikalsicher Sicht die Speicher?**
  Strukturspeicher (Magnetspeicher, Optische Speicher, Halbleiter), Speicher aus rückgekoppelten nichtlinearen Vierpolen (Flipflops), Ladungsspeicher (Gate-Substrat-Kapazität eines MOS-Transistors).
- **Welche Eigenschaften haben Strukturspeicher?**
  Zum Halten der Daten ist keine Energie erforderlich. Zum Schreiben und Lesen ist allerdings Energie erforderlich.
- **Was muß zusätzlich bei Ladungsspeichern durchgeführt werden?**
  Auffrischen der Daten.
- **Welche charakteristischen Größen eines Speichers kennen Sie?**
  Kapazität, Übertragungsblockgröße, Zugriffszeit, Zykluszeit, Blockübertragungsrate, Bandbreite, Zugriffsart (Sequentiell, Ditekt, Wahlfrei, Assoziativ).
- **Wie ist ein d-dimenensionaler Halbleiterspeicher organisiert?**
  Als ein Würfel mit d-Dimensionen, wobei für je eine Dimension wird ein Addressenkodierer eingesetzt.
- **Was benötigt man an den Enden des Addressdekodierers?**
  Verstärkerschaltungen.
- **Warum führt man eine mehrdimensionale Anordnung ein?**
  Weil man dadurch sehr wenige Verstärker braucht.
- **Wie kann man Refresh-Zyklen verlängern?**
  Ausgesonderte langhaltende Speicher, Geringe Spannung.
- **Welches zentrale Problem ergibt sich bei Halbleiterspeichern?**
  Zuverlässigkeit.
- **Wodurch entstehen Fehler?**
  Durch ionisierende Teilchen, Verstärkerfehler.
- **Welches einfaches Verfahren zur Fehlererkenung nutzt man?**
  Paritätssummen abspeichern.
- **Welche Fehler lassen sich dadurch erkennen?**
  Geradfache (Wahrscheinlichkeit über binomialkoeffizienten).
- **Was sind *Hamming-Codes*?**
  Jedes gültige Wort aus 2^n Wörtern darf eindeutig? zu jedem ungültigen Wort aus 2^{n+k} die Hammingdistanz 1 besitzen.
- **Kennen Sie einen Algorithmus zur Konstruktion von Hammingcodes?**
  Erstelle eine Matrix mit $n+k$ Spalten für die Nutzstellen und Prüfstellen und k Zeilen für die Prüfstellen. Die Prüfstellen bekommen die Spalten 2^{i-1}. Jede Spalte q_i enthält genau ein Kreuz in der Zeile i. Für jede Nutzspalte trage zwei Kreuze ein, sodass jede Nutzspalte andere Belegungen besitzt.
- **Wie kann man die Korrektur-Spalten dual interpretieren?**
  Als Spaltennummern.
- **Was bezeichnen die Kreuze in der so gebildeten Tabelle?**
  Nutzwerte, für die die Quersumme bestimmt wird.
- **Wie bestimmt man den Fehlerwektor?**
  Dur komponentenweise XOR-Addition des alten und neuen Prüfvektors.

- **Was besagt der Fehlervektor?**
  Die Spalte mit dem Fehler.
- **Wie viele Fehler kann man mit *Hamming-Zahlen* korrigiern?**
  Einen.
- **Welche *Hammingdistanz* ist für's Erkennen zweier Fehler erforderlich?**
  4.
- **Welche Bedingung gilt für zwei Fehlererkennenden Code?**
  $n \leq 2^{k-1} - k$. (Eine Stelle mehr ist erforderlich wegen $2^k \geq 1 + n + k$)
- **Wodurch wird Speicherhierarchie ermöglicht?**
  Lokalität eines Programms und geeignete Verwaltung des Datentransfers.
- **Welche Werte nutzt man um Speicherhierarchien zu charakterisieren?**
  Mittlere Verfügbarkeit auf einer Ebene.
- **Wie kann man die Verfügbarkeit ermitteln?**
  Mit Hilfe von Programmen (Abhängigkeit von Eigenschaften der Progamme, Größe der Blöcke, Speicherkapazität und Transferstrategie).
- **Was ist bei Speicherhierarchien eine Strategie?**
  Kosten in die Nähe der billigesten/schnellsten Einheit zu bringen.
- **Wie errechnet man die mittlere Zugriffszeit eines Prozessors auf ein Datim in einer bestimmten Ebene in der Speicherhierarchie?**
  Summe der Ebenenzeiten.
- **Wie kann man vollständige Zeit errechnen?**
  Summe aller mittleren Zugriffszeiten über alle Ebenen mal die Wahrscheinlichkeit, dass das Datum auf der Ebene vorhanden ist aber nicht auf der Ebene davor.
- **Wie kann man die mittlere Zugriffszeit nach Umformungen beschreiben?**
  $\sum_{i=1}^{n}(1 - H(i-1))t_j$. Also mit der Fehlerwahrscheinlichkeit gewichtete Summe der mittleren Zugriffszeiten über alle Hierarchien.

5.2 Pufferspeicherverwaltung (Cache)

- **Welche Größen sind für die Organisation von Pufferspeichern von Bedeutung?**
  Cachegröße, Blockgröße, Anzahl der Caches, Trefferraten, Zugriffszeiten, Verzögerungszeiten beim Fehler, Ersetzungsstrategien.
- **Wie wird ein Speicher und Pufferspeicher meist organisiert?**
  Speicher ist in Blöcke fester Größe W eingeteilt. Pufferspeicher ist in Blockrahmen unterteilt mit ebenfalls Größe W.
- **Welche Abbildungsstrategien zwischen Speicher und Puffersspeicher unterscheidt man grundsätzlich?**
  Direkte Abbildung, vollständig assoziative und mehrfach assoziative.
- **Wie geht man bei direkter Abbildung vor?**
  Jeder Block hat eine feste Nummer modulo Blockrahmenzahl.
- **Wie sehen die Addressen bei fester Addressierung aus?**
  Addresse div Rahmenzahl, Address mod Rahmenzahl, Bytenummer.
- **Welche Funktion hat das *dirty*-Flag?**
  Entscheidung ob man in den Speicher zurückschreiben muss oder nicht.
- **Welchen Nachteil hat die direkte Strategie?**
  Unnötiges Überschreiben der Blöcke beim sequentiellen Auslesen von mehreren Wörtern mit gleicher Blockrahmenzahl.
- **Wie geht man bei der vollständig assoziativen Lösung vor?**
  Jeder Block kann in jedem Rahmen untergrebracht werden.

- **Welche Nachteile haben vollassoziative Puffer?**
  Sowohl suchen als auch schreiben eines Blocks ist zu aufwändig.
- **Wie geht man bei einfach assoziativen Pufferspeichern vor?**
  Man teilt den Pufferspeicher in kleinere Rahmen ein innerhalb welcher assoziativ zugegriffen wird.
- **Wie wird ein einfach assoziativer Pufferspeicher addressiert?**
  Tag ist die Nummer des Rahmenblocks. Innerhalb des Blocks wird assoziativ verglichen.
- **Welche Strategien unterscheidet man beim Zusrückschreiben von Werten aus dem Pufferspeicher?**
  FIFO (z.B. Zyklisch), LRU, und zufällig.
- **Welche möglichkeiten ergeben sich beim Schreiben in den Speicher?**
  Durchschreibemethode (ineffizient), Rückschreibemethode (Dirty-Bit),
- **Wie kann man Chacheprobleme bei Mehrprozessorsystemen lösen?**
  Mit Durchschreibetechnik, Rückschreiben nur bei Veränderung im Cache oder Tabellen von gemeinsamen Ressourcen.
- **Von welchen Architekturmerkmalen hängen die Zugriffszeiten bei Fehlern ab?**
  Anzahl der Taktzyklen pro Befehl und von der Zugriffszeit auf den Arbeitsspeicher.
- **Wie kann man Konflikte bei Pufferzugriffen unterscheiden?**
  Erstzugriffe, Kapazitätsbeschränkung, Zugriffskonflikte.
- **Wie kann man eine n-fach assoziative Speicherung mit direktzugriff vergleichen?**
  Ein Sprung von direkt zu 2-fach assoziativ. Dann nur geringer Abfall.
- **Wie kann man Fehlerraten auf Pufferspeicher reduzieren?**
  Veränderung der Blockgröße, Zusatzhardware (Überlaufcache parallel), Softwareverbesserungen.
- **Welche Beobachtungen kann man machen bei unterschiedlichen Blockgrößen?**
  Es gibt ein Optimum der Trefferrate in abhängigkeit von der Blockgröße in einem mittleren Bereich. Danach steigt die Fehlerrate weiter.
- **Was muss man außer der Fehlerrate noch beachten bei Vergrösserung der Blöcke?**
  Zugriffszeit steigt bei höheren Blockgrößen.
- **Welche Hardwaremöglichkeiten existieren zur Verbesserung von Puffern?**
  Überlaufcache für Direktpuffer, Pseudoassoziativer Puffer (irgendwo zusätzlich ablegen),
- **Welche Softwaremöglichkeiten zur Verbesserung der Pufferzugriffszeit existieren?**
  Veschmelzungen von Feldern (Verschränkter Zugriff), Schleifenoptimierung (Ordnung), Zusammenfassung von Schleifen, Blockbildung.
- **Wie kann man die Zugriffszeiten bei Cachefehlern reduzieren?**
  Einführung eines Schreibpuffers zwischen Cache und Speicher (Konsistenzprobleme: wo liegt ein Datum im Puffer?), Second-Level-Cache.
- **Welche Bedingung sollte für die Caches beim second-level Cache gelten?**
  Langsameres enthält alle Blöcke des Schnelleren.
- **Wie kann man allgemeine Zugriffszeiten bei Caches einsparen?**
  Indem man Prozessaddressen statt Maschinenaddressen verwendet (*virtuelle Caches*).
- **Was spricht gegen die Verwendung von virtuellen Caches?**
  Prozesswechsel erfordert das Ungültigsetzen alle Blöcke, Mehrere Kopien einer Speicherzelle (Konsistenz), Ein/Ausgabe werden erschwert.
- **Was verwendet man beim Pentium II?**
  Zwei vierfach assoziative Caches für Instruktionen und zwei fach assoziativ für Daten.

5.3 Speicherverwaltung

- **Welche Erfordernisse stellt man an den Arbeitsspeicher?**
  Einfache Peripheriezugriffe, Einfache Prozessorzugriffe (mehrere Prozessoren) und Pufferspeicher.
- **Welche Modifikationen des einfachen Speichermodells kennen Sie?**
  Ausleseparallelität (Bus), Parallelisierung innerhalb des Speichers und weitere Übertragung über Bus und Cache zum Prozessor.
- **Was versteht man unter Speicherverschränkung?**
  Ordnung der Speicherbausteine zur optimalen Ausleserate.
- **Welche Möglichkeiten der Zuordnung kennen Sie?**
  Nebenliegende Addressen (nacheinander/nebeneinander Auslesen) und verteilte Addressen (parallel Auslesen).
- **Wovon hängt die Fehlerrate bei Seitenzugriffen ab?**
  Größen der Speicher und Strategie der Seitenauswahl.

Index