Nennen Sie die Adressierungsarten die Sie kennen! (S.148)

-Implied Mode
-Register Mode
-Immediate Mode
-Direct Addressing ModeRegister-Indirect Mode
-Program-Counter-Relative-Adressing Mode
-Indirect Addressing Mode

Eräutern Sie die direkte,indirekte,register-indirekte,programm-counter relative Adressierung!

a.) direkt: Die Adresse des Datums ist i Maschinenbefehl gespeichert.
b.) indirekt: Im Maschinenbefehl ist eine Speicheradresse enthalten, an der die Adresse des Datums steht
c.) register indirekt: IM Maschinenbefehl ist angegeben in welchem Register die Adresse des Datums steht
d.) programm counter relative: Im Maschinenbefehl ist ein Displacement angegeben das zum aktuellen Stand des Pcs addiert werden muss um die entsprechende Adresse zu erhalten. Das Displacement ist eine positive oder negative Zahl

Was ist Interleaved Memory und wozu wird er eingesetzt. (s.162)

Sequentielle direkt hintereinander liegende Befehle können schneller erreicht werden, da Adressen, die in verschiedenen Speicherblöcken liegen, schneller auf den Adressbus gelegt werden können, als es sonst – bedingt durch Zugriffszeiten der Bausteine – möglich wäre. Die Daten kommen dadurch ebenfalls schneller beim Prozessor an.

Nenne Sie die 3 Replacement Strategien (s.165)

1. LRU-Methode (Least Recent Used)
2. LFU-Methode (Least Frequently Used)
3. RANDOM
Aus Erfahrung und durch Messungen hat sich gezeigt, daß alle Methoden ungefähr gleich gut sind, wenn die Kapazität des Cache nicht zu klein ist.

Erklären Sie die Buffered-Write-Methode im Zusammenhang mit Caches! Welche Probleme können dabei auftreten?

Bei diesem Verfahren wird der neue Wert gleichzeitig in das Cache und in einen zweiten, schnellen Zwischenspeicher (Buffer) eingetragen. Während der Prozessor schon mit der weiteren Abarbeitung des Programmes fortfahren kann, werden die gepufferten Daten in den Hauptspeicher übertragen. So kann die Datenkohärenz erhalten bleiben. Ein Problem tritt auf, wenn mehrere Schreiboperationen direkt aufeinander folgen und der Puffer daher nicht schnell genug in den Hauptspeicher übertragen werden kann. In diesem Fall muß der Prozessor warten.

DMA (s.168): Was es ist und warum es verwendet wird.

Direct Memory Access (DMA) ermöglicht es, die Kommunikation zwischen dem Prozessor und den meist sehr viel langsameren peripheren Geräten zu beschleunigen. Er dient zur direkten Übertragung großer Datenmengen vom bzw. zum Speicher, ohne die CPU dabei in Anspruch zu nehmen. Um Konflikte zu vermeiden, darf die CPU während des DMAs nicht auf den Bus zugreifen. Somit ist nicht sichergestellt, dass der Prozessor in der Zeitspanne, die zum Transfer nötig ist, Aufgaben durchführt, die ohne externen Buszugriff möglich sind. Auch ist eine zusätzlich Kontrolleinheit nötig – der DMA-Controller.

Vorgang des DMA erklären

1. Der Prozessor teilt dem DMAC die Adresse der Quelle (resource pointer), die des Ziels (destination pointer) und die Größe der zu übertragenden Daten (block length) mit. Hierauf kann die CPU mit der Abarbeitung des Programms fortfahren.
2. Der DMAC fordert nun vom entsprechenden Gerät die Daten an und wartet, bis es zum Transfer bereit ist.
3. Nach dem Ende der Übertragung meldet der DMAC dem Prozessor den erfolgreichen Abschluß der Aktion meistens per Interruptsignal. Die Daten werden also direkt zwischen I/O-Device und Speicher ausgetauscht.

2 Arten von DMA:

Transparent DMA und Cycle-Stealing-DMA

Controller und Co-Prozessoren (s.169)

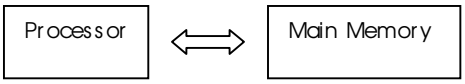
Wozu dient ein virtueller Speicher (s.225)

Bei einer großen Anzahl gleichzeitig speicherresidenter Programme waren vor der Erfindung dieses Konzeptes sehr große Speicher notwendig. In der Zeit der Kernspeicher waren jedoch der Kapazität Grenzen gesetzt. Diesem Problem wurde durch die Erfindung des virtuellen Speichers abgeholfen: Im Prinzip dient dabei ein Externspeicher (eine Disk) als virtueller Hauptspeicher. Da ein Prozessor nur auf den physikalischen Hauptspeicher direkt zugreifen kann, werden die für die Exekution benötigten "Abschnitte" einfach vom Externspeicher geladen. Zu jedem Zeitpunkt ist also nur der gerade benötigte Teil eines Programmes speicherresident.

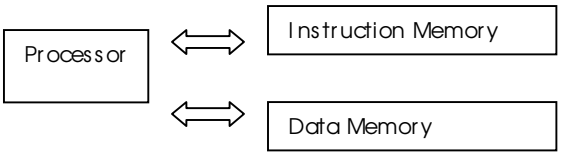
und die Abbildung (Prozess-Zustände) auf der seite 206 anschaun.

Welchen Vorteil besitzt die „von Neumann" und die „Harvard-Architektur"!
Geben Sie auch schematische Darstellungen dser beiden Architekturen an!

Von Neumann-Architektur: Programme und Daten werden in ein und demselben Speicher gehalten und über einen einzigen Datenbus übertragen.



Harvard-Architektur : Programme und Daten werden in getrennte Speichern gehalten und über getrennte Busse übertragen.



Welchen Vorteil besitzt die „Harvard-Architektur" gegenüber „von Neumannschen ", und wo wird dieser genutzt?

Bei der Harcard Architektur ist ein simultaner Zugriff über getrennten Speichern für Daten und Instruktionen möglich. Dieser Vorteil wird bein Instruction-Pipelining genutzt.

Geben Sie einige Quellen für Probleme, die beim Instruction-Pipelining auftreten, an!

1. Sprünge (bedingte und unbedingte)
2. Soubroutine – Calls
3. Interfering instructions: Das Ergebnis der n-ten Operation wird als Operand (n+1)ten benötigt. Operation n zerstört Register, das von Operation (n-1) noch benötigt wird.

Sprünge (bedingte und unbedingte):

Sprünge können sowohl auf Register-Transfer-Ebene als auch auf Maschinen-Code-Ebene erfolgen. Bei dieser Operation wird der MIC (bzw. Programm-Counter) mit einem neuen Wert geladen. Bei bedingten Sprüngen kann zusätzlich eine Bedingung angegeben werden, die erfüllt sein muss, damit ein Sprung erfolgen kann.

Subroutine Calls:
Da es bei umfangreicheren Programmen oft vorkommt, daß eine bestimmte Folge von Befehlen öfters vorkommt, wird dieser nur einmal programmiert und von verschiedenen Stellen aus aufgerufen. Danach wird die Return-from-Subroutine-Instruktion ausgeführt, die an der Stelle nach dem Aufruf der Subroutine im Programm fortsetzt. Es sind auch geschachtelte Aufruffolgen möglich.
Call-Subroutine:
a. Retten des PC, der Registerinhalte und des PSW
b. Laden des PC mit der Startadresse (stellt einen Teil des Befehls dar) der Prozedur
c. Beginn der Abarbeitung der Prozedur durch den Interpreter
Return-from-Interrupt-Subroutine
1.Wiederherstellung des PSW und der Registerinhalte
2.Laden des PC mit dem Wert, der beim Call-Subroutine-Befehl gesichert wurde (der PC zeigt nun auf die erste Instruktion nach dem Call-Subroutine-Befehl)
3. Fortsetzen des Programmablaufes

Interrupts:
Dieses Konzept ermöglicht die Bewältigung von diversen Problemen, die nicht im Programmablauf vorgesehen sind. Dabei wird der Programmablauf unterbrochen und die ISRs (Interrupt Service Routinen) aufgerufen. Danach kann meist das unterbrochene Programm fortgesetzt werden. Die Interrupts sind mit Subroutinen vergleichbar (Unterschied: Werden durch prozessorinterne Ausnahmefälle oder externe Ereignisse ausgelöst).

Was sind Race Conditions?
Das sind Schwierigkeiten, die durch nie zu bestimmende „Exekutionsreihenfolgen" paralleler Prozesse auftauchen können.
Bsp: Druckerwarteschlange
Race Conditions liegen vor, wenn das "Ergebnis" der Ausführung eines Programmsystems von der relativen "Geschwindigkeit" der beteiligten Prozesse abhängt. Das Eingrenzen dieses Problem, das durch die parallele Abarbeitung entsteht, ist durch das relativ seltene und indeterministische Auftreten eine äußerst schwierige und zeitraubende Angelegenheit. Die Eigentliche Ursache ist die Tatsache, daß das P_SIGNAL ignoriert wird, wenn der Empfängerprozeß nicht im Zustand BLOCKED ist ⇒ mittels P_SIGNAL gemeldetes Eintreten eines Ereignisses "aufheben", wenn der empfangene Prozeß (noch) nicht im Blockierungszustand ist. Eine oft verwendete Möglichkeit ist es, im READY-Zustand ankommende Ereignismeldungen in einer Liste im Prozeßdeskriptor zwischenzuspeichern und bei einem späteren P_SLEEP gar keine Blockierung mehr vorzunehmen. In diesem Zusammenhang ist es üblich, von anstehenden Ereignissen zu

sprechen. Eine alternative dazu wäre es, dem Aufrufer von P_SIGNAL mitzuteilen, ob der Empfänger das Ereignis brauchen konnte (BLOCKED war) oder nicht.

Was versteht man unter Trashing?
Ein Phänomen, das beim Paging auftritt, wenn der Speicher zu klein ist um alle Working Sets aufzunehmen: Die Maschine ist praktisch ausschließlich damit beschäftigt, die referenzierten Pages vom Externspeicher zu laden.

Nennen Sie alle Layer des ISO OSI Reference Models!
Layer 1: Physical Layer
Layer 2: Data Link Layer
Layer 3: Network Layer
Layer 4: Transport Layer
Layer 5: Session Layer
Layer 6: Presentation Layer
Layer 7: Application Layer

Welches Problem tritt beim Linken im Zusammenhang mit direkter Adressierung auf und wie kann es behoben werden?
Beim Linken entstehen durch das Zusammenfügen von Modulen Veränderungen in den Adressen der Sprungziele. Bei direkter Adressierung muß der Linker eine Modifikation der Operanden durchführen, da sonst die Adressen der Sprungziele verfehlt werden. Für die Code-Segmente können Probleme dieser Art durch die Berücksichtigung der Konvention für Position Independent Code (PIC) gelöst werden. Dabei werden z.B. statt absoluter Sprungadressen nur die Distanzen (Displacements) von der aktuellen Adresse angegeben. Da sich durch die Verschiebung die Displacements nicht ändern, entfallen die Operandenmodifikationen völlig.

Kann es zwischen Threads zu einem Deadlock kommen?
Es kann dazu kommen, weil die meisten Programme nicht für parallele, sondern für sequentielle Abarbeitung geschrieben sind. Je mehr Programme parallel ausgeführt werden müssen, desto größer ist die Wahrscheinlichkeit eines Fehlers, weil die Verwaltung komplizierter wird.
Möglichkeiten der Behandlung eines Deadlocks:

1. Deadlock Detection and Recovery: im Zuge der ganz gewöhnlichen Zuteilungen und Freigaben von Objekten wird der Resource Allocation Graph aktualisiert und auf Deadlock-Situationen untersucht. Bei Auffinden solcher Zyklen, wird er durch das Terminieren der beteiligten Prozesse aufgelöst. Nachteile: die recht brutale Vorgehensweise kann zu Problemen führen und die Analyse des Graphen verursacht einen nicht unerheblichen System-Overhead beim Type-Management.
2. Deadlock Prevention: basiert darauf, allein durch die Beachtung gewisser Kriterien beim Design eines Betriebssystems eine der notwendigen Bedingungen für einen Deadlock a priori zu "verletzen", wodurch ein solcher gar nicht entstehen kann.
3. Deadlock Avoidance: auch dieser Methode liegt eine sorgfältige Beobachtung der Objekt-Anforderungen zugrunde: es wird versucht vorausschauend zu klären, ob die Zuteilung irgendwelche späteren Deadlocks nach sich ziehen kann. Es gibt dafür zwar Algorithmen, aber diese liefern nur im Falle des Vorhandenseins gewisser zusätzlicher Informationen "sichere" Zuteilungsentscheidungen.

Welche Weiterentwicklungen der RISC Architektur sind im Vorlesungsbuch beschrieben?

1. SPARC-Architektur (Scalable Processor Architecture)
2. MIPS-Architektur (Microprocessor without Interlocking Stages)

Welche löschbaren optischen Platten gibt es und auf welchen physikalischen Prinzipien beruhen sie?

1. rein optische Platten:

Beim **Schreiben** wird ein dünner Film einer besonderen Metallegierung mittels eines leistungsstarken Lasers zum Schmelzen gebracht, was einen Übergang vom kristallinen zum amorphen Zustand zur Folge hat, der weniger Licht reflektiert.
Dies wird dann beim **Abtasten** mit einem schwächeren Laser ausgenutzt, indem man die Stärke des reflektierten Lichtes mißt. Beim **Löschen** wird die gesamte Plattenoberfläche einheitlich erhitzt und alle Teilchen der Metallegierung in den kristallinen zurückversetzt.

2. magneto-optisch Platten:

Beim **Schreiben** wird ein dünner Film einer besonderen Metallegierung mittels eines Lasers erhitzt, was die Umpolung von einzelnen magnetischen Bezirken mittels eines Magnets, der auf dem Schreib-Lesekopf angebracht ist, erleichtert. Das Metall kühlt schnell ab und die Information bleibt erhalten.
Beim **Lesen** nützt man die Eigenschaft aus, daß die Magnetisierungsrichtung der magnetischen Bezirke die Schwingungsebene des polarisierten Lichtstrahles des Lasers an der Plattenoberfläche entweder nach links oder rechts dreht (log. 1 oder 0).
Das **Löschen** erfolgt wiederum durch allgemeine Erhitzung der gesamten Plattenoberfläche, wobei alle magnetischen Bezirke gleichsinnig ausgerichtet werden.

Was ist ein endlicher deterministischer Automat?

endlich: die Anzahl der Zustände des Automaten ist begrenzt
deterministisch: aus der Eingangsinformation und dem Vorzustand läßt sich stets eindeutig vorherbestimmen, in welchen Zustand der Automat wechseln wird.