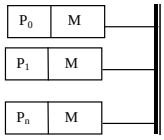
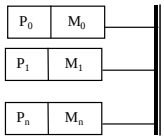
Fragen / Antworten Rechnerarchitektur, sortiert

Nr	Suchwort	Frage, Antwort
38	Aliasing einer Var	Was ist Aliasing einer Variable • Es gibt mehrere Wege, über die aliased Variable adressierungsmäßig erreichbar sind • Aliased Variable können nicht Registern zugeordnet werden
62	Alignment	Welche Vor und Nachteile hat es, jede beliebige Ausrichtung (Alignment) von Daten zuzulassen • Vorteile: Programmierer muß nicht auf das Alignment achten, spart Platz • Nachteil: verlangsamt die Maschine
19	Architektur Akku, Stack, Registersatz	Nennen Sie Vor und Nachteile von Rechnerarchitekturen, die für die CPU interne Operanden-Speicherung einen Akku, eine Stack, oder einen Registersatz vorsehen **Architektur** — **Vorteile** — **Nachteile** Akku — kurze Befehle, einfache Steuerung — hoher Speicherverkehr, da Akku nur temp Speicher Stack — gute Codedichte, einfache Auswertung von Ausdrücken (reverse polish notation) — da der Stack nicht beliebig adressiert werden kann, ist es schwierig effizienten Code zu generieren Register — allgemeines Modell, Scheduling — lange befehle, da jeder Operand genannt wird
8	Architektur Fkt hinzufügen	Auf welcher Architekturebene können einem Rechner neue Fkt hinzu gefügt werden Auf welcher Ebene kann man kostengünstige, und auf welcher leistungsstarke Fkt Erweiterungen vornehmen • Auf allen Ebenen • Kostengünstig: SW Ebene leistungsstark: HW Ebene
53	Architektur Konsistenz	Nennen sie ein Kriterium für die Konsistenz einer Architektur • Aus der teilweisen Kenntnis des Systems, kann man den Rest vorhersagen z.B. kennt man einen Teil der Befehle, weiß man auch, wie die anderen aufgebaut sind (Displacement ist gleich bei ähnlichen Befehlen)
	Architektur orthogonal	Wodurch ist eine orthogonale Struktur gekennzeichnet • Unabhängige Fkt werden unabhängig spezifiziert
32	Assembler Programme	Wie kann man Assemblerprogramme für einen Rechner X auf einen Rechner Y zum Ablauf bringen • Emulieren, falls mikroprogrammierte Maschine(und Architektur ähnlich) • Sonst simulieren
46	Branch Prediction, History, Future File	Ist es zulässig, bei einem Computer mit Branch Prediction Befehlen im vorhergesagten Pfad das Abspeichern von Ergebnissen in Registern zu Erlauben, bevor die Sprungbedingung ausgewertet ist. Was für eine Einrichtung muß als Voraussetzung dafür implementiert sein • Ja(in ein History File sichern, oder von Future File übernehmen) • History oder Future File
56	Branch Pridiction Prinzip	Worauf beruht die Sprungvorhersage(Branch Prediction) • Auf das bisherige Sprungverhalten des Programms (Speicher mit Einträgen von zurückliegenden Sprüngen)

73	Bus & Cache SMMP	Welchen Vorteil und welchen Nachteil hat ein Bus als Verbindung von mit Caches ausgestatteten Prozessoren und den Speichermodulen gegenüber einem mehrstufigen Verbindungs Netzwerk (multistate switching Netzwerk) bei der Realisierung von SMMP Systemen • Vorteil Bus: snooping Protokoll Nachteil Bus: Leistungsfähigkeit durch Busgeschwindigkeit beschränkt, blocking System • Vorteil mehrst. Verb NW: none blocking System • Nachteil mehrst. Verb NW: directory based, teuer, Konsistenz Erhaltung nur mit sehr hohem Aufwand
65	Cache Kohärenz	Wie wird die Cache Kohärenz in Virtually Shared Memory Multi Prozessor Systemen sicher gestellt, bei denen die Prozessoren, die jeweils mit einem Speichermodul des gesamten Shared Memory eine körperliche Einheit bilden, die Ecken eines Hypercubes darstellen, oder wie Perlen auf einer Kette ringförmig angeordnet sind, oder mehrstufiges Verbindungs-Netzwerk • Hypercube: directory Methode • Verbindungs Netzwerk directory Methode • Kette Ring snooping Bus
51	Cache Kohärenz 	Wie wird die **Cache Kohärenz** in einem SMMP System sichergestellt, wenn die Prozessoren und die Speichermodule über: ein Bussystem miteinander verbunden sind(Speichermodule am Prozessorboard sind Teile des Gesamt-ASP) ein mehrstufiges Verbindungs Netzwerk miteinander verbunden sind • Snooping Bus(einer spricht, alle hören zu) • Directory Methode
	Cache Kohärenz bei DMMP 	Spielt die Cache Kohärenz bei DMMP Systemen eine Rolle • Nein, da jeder Prozessor seinen eigenen, lokalen Cache-Speicher hat • Ein Prozessor kann auf den Cache Speicher des anderen Prozessor nicht zugreifen
47	Cache Befehls, Daten Cache	Warum gibt es in einigen Maschinen getrennte Befehls und Daten Cache • Doppelte Anzahl von Schnittstellen zum Speichersystem ⇒ bessere Performance
	Cache Instruction, Data Cache	Aus welchem Grund werden Caches häufig in ein Instruction und ein Data Cache aufgeteilt • Doppelte Anzahl von Schnittstellen zum Speichersystem ⇒ bessere Performance
67	Cache Linesize	Was spricht bei einem Cache für und was gegen eine große Linesize • Vorteil: locality of reference (lokale Zugriffs Strategie, Daten und Befehlszugriffe sind oft in der Nachbarschaft) • Nachteil: häufigeres Ersetzen von Blöcken notwendig
	Cache Second Level Cache	Aus welchem Grund wird häufig ein Second Level Cache eingebaut • Ein First Level Cache ist teurer und in der Größe begrenzt, da zum Erreichen einer max Geschwindigkeit der Cache direkt im Prozessor Chip sitzen muß, aber dort der Platz begrenzt ist
	Cache tag field virtuelle @	Nennen Sie die Vor und Nachteile von Caches, bei denen die tagfields virtuelle Adressen enthalten • Vorteile: virtuelle @ spart die Übersetzung • Nachteile: virtuelle @ ist nicht eindeutig, da alle Programme mit der @ = 0 beginnen • Bei Programmwechsel müssen die Cache geflasht werden • Mehraufwand
17	Cache, virtuell, reell adressieren	Welche Gründe sprechen dafür, das Cache virtuell, oder reell zu adressieren • Virtuell Zeitersparnis durch Wegfall der Adressumsetzung eine virtuelle Adresse ist pro Programm vorhanden, die immer bei = 0 beginnt • Reell Adresse ist eindeutig(kein Löschen des Cache nach Programmwechsel) verschiedene Programme ⇒ verschiedene Adressen
57	Chaining bei Vektorrechner	Was bedeutet Chaining bei Vektorrechner • Überlappung der Befehlsausführung: 2 Vektorbefehle folgen aufeinander. Der erste erzeugt ein Ergebnis, das vom Zweiten benötigt wird Der Zweite muß nicht warten, bis der Erste kpl fertig ist, sondern kann bereits nach der ersten Berechnung des ersten Befehls anfangen
	Compare & swap	Zeigen Sie, wie ein Programm mit Hilfe des Befehls „compare & swap“ eine gemeinsam benützte Speicherzelle verändern kann, ohne daß Änderungen durch andere Programme oder auch die von ihm selbst durchgeführte Änderung in irgendeiner Weise verfälscht werden • Lockbit, Semaphor

77	Compiler Backend Compiler Frontend Compiler	Was ist die Aufgabe des Backends eines Compiler • Frontend: Erzeugt den Code, passend zu der jeweiligen Maschine, auf der der Compiler gerade läuft • Backend: den Sourcecode in eine gemeinsame Zwischensprache übersetzen ⇒ Objektcode
1	Compiler Optimierung	Nennen sie Methoden zur Compiler Optimierung • Code Inlining (Inline Methoden direkt in den Code einbinden) • Loop Unrolling (Schleifen **n** mal in den Code kopieren) • Register Assignment (durch graph coloring) Registern werden Variable zugeordnet • SW Pipelining (nur bei Pipeline Maschinen) verschobenes Verarbeiten von Befehlen
2	Compiler Optimizer	Nennen Sie Methoden, mit denen ein im Compiler enthaltener Optimizer die Laufzeit des Objektprogramms verkürzen kann • Graph Coloring • Code Inlining • Code motion
75	Compiler, Schedular	Erklären Sie, wie ein dem Compiler nach geschaltetes Schedular Programm die Laufzeit des Objekt Programms verkürzen kann • Befehle umstellen, um stall cycle zu vermeiden
78	Coprozessor	Was sind die Gründe bestimmte Fkt in Coprozessoren zu verwirklichen • CPU ohne Coprozessor sind billiger
26	CPI < 1 Takte pro Befehl	Welche Einrichtungen ermöglichen CPI < 1 (Clocks per Instruction = Takte pro Befehl) • Superscalare Maschinen • Super Pipelining
	CPI < 1 Takte pro Befehl	Bei welchen Rechnern ist CPI < 1 möglich • Superscalare Maschinen • Super Pipelining
66	Crossbar System	Welche Vor und Nachteile hat ein Crossbar System (in SMMP Systemen) für die Verbindung von Prozessor Modulen mit Speichermodulen gegenüber einer Busanordnung • Vorteile Crossbar: non blocking System(mehrere parallele Vorgänge können gleichzeitig laufen) • Nachteile Crossbar: directory Protokoll notwendig, teuer, PIN intensiv(höhere Fehlerhäufigkeit) • Vorteile Bus: snooping Protokoll • Nachteil Bus: blocking System
	Datenabhängigkeit	Datenabhängigkeiten stören das Pipelining Welche 3 Arten der Datenabhängigkeit gibt es • Struktur bedingt(nicht genügend funktionelle Einrichtungen vorhanden, z.B. Addierer) • Daten bedingt (ein Befehl braucht als Eingangsgröße Daten, die ein vorausgehender Befehl noch nicht berechnet hat • Ablauf bedingt(Sprung, Verzweigung)

16	Datenabhängigkeit,
	Loop unrolling
		Frage:
		LOOP LD F0, 0(R1)
		ADDD F4, F0, F2
		SD 0(R1), F4
		SUB R1, R1, #8 //Adressberechnung
		BNEZ R1, LOOP //bedingter Rücksprung
		Lösung:
		LOOP LD F0, 0(R1)
		SUB R1, R1, #8
		ADDD F4, F0, F2
		BNEZ R1, LOOP
		SD 8(R1), F4
		In der oben dargestellten Schleife werden doppelt lange Gleitpunktzahlen verarbeitet und die Adresse wird mit dem SUB Befehl jeweils auf die nächste zu lesende Gleitpunktzahl weiter geschaltet
		Aufgrund der Datenabhängigkeit verlangsamt sich der Ablauf durch stall cycle
		Die zugrunde liegende Maschine verfügt über „delayed brachning“(ein delay slot)
		Stellen Sie die Befehlsfolge so um, daß die Anzahl der stall cycle minimisiert wird

15	Datenabhängigkeit,
	History, Future File
	
	WAR
		DIVF F0, F2, F4
		ADDF F6, F0, F8
		SUBF F8, F8, F14
		Bei den folgenden Befehlen sind die Parameter in der Reihenfolge „Ziel“, „Quelle1“, „Quelle2“ angegeben
		Bei der oben stehenden Folge von Gleitpunktbefehlen ist der DIVF Befehl noch nicht abgeschlossen, und der ADDF Befehl kann wegen der Datenabhängigkeit nicht begonnen werden.
		Unter welchen Umständen darf der SUBF Befehl zu dieser Zeit ausgeführt werden
		• SUBF Befehl darf nur ausgeführt werden, wenn ein History oder Future File existiert

5	Delay Slot	Was ist ein delay Slot und wozu dient er • Der Delayslot ist der Befehl nach einem Branch Befehl. Er verhindert den stall-cycle, indem er die Pipeline gefüllt hält
	Delay Slot	Was geschieht mit dem im Delay Slot eines bedingten Sprungbefehls stehenden Befehl • Befehl wird immer ausgeführt • Sollte kein geeigneter Befehl vorhanden sein, so wird im Delay Slot ein NOP Befehl eingesetzt
	Delay Slot	ADD R1, R1, R2 BGEZ R1, Branch Target Darf im oberen Programmstück der unterstrichene Befehl in den Delay Slot des Befehls „Branch On Greather Than Or Equal To Zero“ verlegt werden • Nein, da R1 vom ADD Befehl verändert wird(Berechnung Sprungziel)
29	Delayed Branching	Warum genügt es nicht, bei Maschinen mit „Delayed Branching“ beim INT nur einen Befehlszählerstand zu speichern • beim Delay Slot kann man nicht sagen, ob der nächste Befehl oder der Sprung ausgelöst wird. Darum müssen 2 Befehlszählerstände abgespeichert werden(**aktuelle Adresse und Sprungadresse**)
	Delayed Branching	• egal ob gesprungen wird oder nicht, der darauf folgende Befehl im Delay Slot wird immer ausgeführt (dieser Befehl kann von vor der Verzweigung, vom Zielort, oder vom Fall Through herstammen) • der Mechanismus des Delayed Branching muß von HW und SW unterstützt werden
72	DMMP	Nennen Sie Vor und Nachteile von DMMP Systemen, gegenüber SMMP Systemen • Vorteile DMMP: jeder hat seinen eigenen Speicher, aber Zugang zu jedem, auf Kohärenz muß nicht geachtet werden • Nachteile DMMP: Programm Änderungen sind nötig(Datenaustausch schlecht)
76	DMMP	In Multiprozessor Systemen kann die Laufzeit eines Programms verkürzt werden, wenn sich mehrere Prozessoren an der Verarbeitung diese einen Programmes beteiligen. Wie wird die Zusammenarbeit der Prozessoren in DMMP Systemen bewerkstelligt • Messages von einem Prozessor an einen anderen Prozessor (da kein Prozessor in den Cache eines anderen Prozessor, zugreifen, hinein schauen kann)
34	Dual Mode Computer	Was kennzeichnet Dual Mode Computer • Maschine kann 2 verschiedene Sprachen, die Umschaltung erfolgt mittels eines Mode Bits
33	Emulation	Was bedeutet Emulation • Umsetzung von Mikrocode auf Mikroebene
79	Erweiterung Rechner funktional	Auf welcher Ebene sollte ein Rechner funktionell erweitert werden, wenn höchste Ausführungs Geschwindigkeit erreicht werden soll • Auf der HW Ebene
36	Excess Notation	Was spricht für die Excess Notation des Exponenten bei Gleitpunktzahlen • Damit beim Prüfen auf Null im Zahlenfeld nur Nullen stehen,Überprüfung auf Null wird damit einfacher
	Forwarding (nur Reg beteiligt)	• Das Ergebnis eines Befehles steht nach der MEM Phase fest, und kann an einen anderen Befehl, der um 2 Taktzyklen später dran ist weiter geleitet werden
	Gleitpunktzahlen	Welche Vorteile hat das Rechnen mit unnormalisierten Gleitpunktzahlen • Genauigkeit ist größer
	Graph Coloring	• Daten werden markiert, ob sie voneinander abhängig sind
18	Häufigkeit Zeitanteil	Wann ist der dynamische Zeitanteil eines Befehls größer als seine dynamische Häufigkeit • Ein selten auftretender Befehl, der eine längere Ausführungszeit hat als der Durchschnitt der anderen Befehle

49	Hazard WAW 2 aufeinanderfolgende Befehle beschreiben gleiches Reg	Ein Programm bei dessen Ablauf durch 2 aufeinanderfolgende Befehle das gleiche Register beschrieben wird, ist nicht sinnvoll Bei Maschinen mit einer besonderen, der Beschleunigung des Programmablaufs dienenden Einrichtung kann eine solche Befehlsfolge in einem sinnvollen Programm vorkommen, wie heißt die Einrichtung Erklären sie den dabei auftretenden Write after Write Hazard WAW • Delayed Branching • J versucht in einen Operanden zu schreiben, bevor er von i beschreiben worden ist. Dadurch stünde der von i und nicht der von j geschriebene Operand zur Verfügung

48	Hazard
	RAW
		Geben Sie eine Befehlsfolge an, bei der ein Read after Write Hazard auftritt
		• DIVF F0, F2, F4
		ADDF F8, F0, F6
		SUBF F9, F8, F5
		Zwischen DIVF und ADDF liegt eine Datenabhängigkeit vor
		Außerdem kann SUBF nicht ausgeführt werden, da SUBF auf das Ergebnis(von F8) warten muß
		(RAW wäre, wenn SUBF das Reg F8 liest, bevor das Reg F8 von ADDF beschreiben wurde)

	Hazard
	WAR
		Geben Sie eine Befehlsfolge an, bei der ein Read after Write Hazard auftritt
		• DIVF F0, F2, F4
		ADDF F10, F0, F8
		SUBF F8, F8, F14
		Zwischen DIVF und ADDF liegt für F0 eine Datenabhängigkeit vor
		Außerdem kann SUBF nicht weiterlaufen, da ADDF auf dem Abschluß von DIVF warten muß, und ADDF benötigt das Reg F8 mit dem alten Wert, das von SUBF nicht überschrieben werden darf
		(WAR wäre, wenn SUBF das Reg F8 beschreibt, bevor das Reg F8 von ADDF gelesen wurde)

28	Hazard WAW	Geben Sie ein Bsp für WAW Data Hazard • BNEZ R1, Target • DIVF F0, F8, F10 • ° • ° • ° Target: SUBF F0, F8, F10 **WAW** Der SUBF Befehl soll im Sprungfall das dann nicht mehr gebrauchte DIVF Ergebnis rückgängig machen Wenn keine besondere Vorkehrung getroffen wird, überschreibt der lang dauernde DIVF Befehl das SUBF Ergebnis, was ein Fehler wäre
21	History, Future File	Wozu dient der History File Kann der entsprechende Effekt auch mit einem Future File erreicht werden Welchen Vorteil hat der History File • Ergebnisse werden vom Überschreiben geschützt (Pipelining: Register ist noch nicht vom vorherigen Befehl frei gegeben, darum wird das Ergebnis in einem History File abgespeichert, bis das Register frei gegeben wird) • Ja • Das History File wird im Normalfall gelöscht Nur bei INT müssen die Daten aus dem History File wieder hergestellt werden(beim Future File immer)
7	Interpreter, Compiler	Warum laufen interpretierte Programme normalerweise langsamer als compilierte Programme • Beim Interpreter muß jeder Befehl einzeln zur Laufzeit übersetzt werden(interpretiert) • Der Compiler übersetzt jeden Befehl nur einmal, bei der erstmaligen Übersetzung des Programms • Interpreter: dynamische Übersetzung Compiler: statische Übersetzung

25	Interupt	Nennen Sie INT, die während eines Befehlsablaufs auftreten und nach deren Bearbeitung das unterbrochene Programm fortgesetzt werden kann • Page fault INT • I/O INT • Time Slice INT
68	Interupt	Im Programmablauf können Unterbrechungsanforderungen entstehen Nennen sie mindestens eine Art von Unterbrechungsanforderungen, nach deren Bearbeitung das Programm, das die Unterbrechungsanforderung erzeugt hat, weiter laufen kann Nennen Sie mindestens 3 Arten von Unterbrechungsanforderungen, nach deren Bearbeitung das Programm, das die Unterbrechungsanforderung erzeugt hat, nicht weiter laufen kann • Page Fault INT, I/O INT, Tracing(Programmablaufverfolgung), Aufruf von System-Fkt • Ilegal Opcode, Speicherschutzverletzung, parity Fehler, Überlauf, Unterlauf
13	Interupt **DIVF F0, F2, F4** **ADDF F10, F10, F8** **SUBF F12, F12, F14**	Bei den folgenden Befehlen sind die Parameter in der Reihenfolge „Ziel“, „Quelle1“, „Quelle2“ angegeben Bei der links stehenden Befehlsfolge ist der DIVF Befehl noch nicht abgeschlossen, der ADDF Befehl ist schon fertig, und im SUBF Befehl tritt ein INT auf. Wie kann erreicht werden, daß nach der INT Behandlung wieder richtig aufgesetzt wird • DIVF zu Ende laufen lassen, und bei SUBF wieder aufsetzen(nach der INT Behandlung)
31	Interupt Page Fault INT	B_1 IF ID EX MEM WB B_2 IF ID EX MEM WB B_3 IF ID EX MEM WB B_4 IF ID EX MEM WB B_5 IF ID EX MEM WB B_6 IF ID EX MEM WB Im Bild oben ist der Befehlsablauf in einer 5 stufigen Pipeline Maschine dargestellt. In der unterstrichenen Phase tritt ein Page fault INT auf. Die Befehle, die zu dieser Zeit in Bearbeitung sind, werden abgebrochen und die Interupt Service Routine einschließlich dem Laden der fehlenden Seite wird durchgeführt Mit welchem Befehl wird das unterbrochene Programm wieder gestartet Wie muß verfahren werden, wenn zum Zeitpunkt des Erkennens des INT der Schreibzugriff in der MEM Phase des Befehls B2 schon so weit fort geschritten ist, daß er nicht mehr aufgehalten werden kann • B2(WB von B1 wird noch ausgeführt) • MEM und WB von B2 ausführen lassen und bei B3 neu aufsetzen
45	Interupt Page Fault INT	Was geschieht nach einem Page Fault INT • BS lädt fehlende Seite nach

<table>
<tr><td>14</td><td>Loop unrolling</td><td>Der unten links angeführte Loop Inhalt soll entrollt werden, so daß 4 Loop-Inhalte zu einer entrollten Loop zusammen gefaßt werden. Die Befehle in der entrollten Loop sollen so angeordnet werden, daß min Pipeline Störungen (stall-cycle) auftreten
Frage:
LOOP: LF F0, 0(R1)
ADDF F4, F0.F2
SF 0(R1), F4
SUB R1, R1, #8 //Adressrechnung
BNEZ R1, LOOP //bedingter Rücksprung
Lösung:
LOOP: LD F0, 0(R1)
LD F6, -8(R1)
LD F10, -16(R1)
LD F14, F0, F2
ADDD F4, F0, F2
ADDD F8, F6, F2
ADDD F12, F10, F2
ADDD F16, F14, F2
SD 0(R1), F4
SD -8(R1), F8
SD -16(R1), F12
SUB R1, R1 #32
BNEZ R1, LOOP
SD 8(R1), F16 // 8-32 = -24</td></tr>
<tr><td>14</td><td>Loop unrolling</td><td>Bei den folgenden Befehlen sind die Parameter in der Reihenfolge Ziel, Quelle1, Quelle2 angegeben
n(Rx) bedeutet, daß der so angesprochene Operand unter einer Adresse gefunden wird, die die Summe aus n und dem Inhalt des Register Rx ist
Mit dem Zeichen # wird ein Immediat Operand(Konstante) gekennzeichnet
Der BNEZ Befehl führt einen Sprung aus, wenn das angesprochene Reg einen Inhalt ≠ 0 hat
In der dargestellten Schleife werden doppelt lange Gleitpunktzahlen verarbeitet und die Adresse wird mit dem SUB Befehl jeweils auf die nächste zu lesende Gleitpunktzahl weiter geschaltet
Aufgrund der Datenabhängigkeit verlangsamt sich der Ablauf durch stall cycle
Die zugrunde liegende Maschine verfügt über Delayed Branching(ein Delay Slot)
Stellen Sie die Befehlsfolge so um, daß die Anzahl der stall cycle minimisiert wird
Frage:
LOOP: LF F0, 0(R1)
ADDF F4, F0.F2
SF 0(R1), F4
SUB R1, R1, #8 //Adressrechnung
BNEZ R1, LOOP //bedingter Rücksprung
Lösung:
LOOP: LD F0, 0(R1)
LD F6, -8(R1)
LD F10, -16(R1)
LD F14, F0, F2
ADDD F4, F0, F2
ADDD F8, F6, F2
ADDD F12, F10, F2
ADDD F16, F14, F2
SD 0(R1), F4
SD -8(R1), F8
SD -16(R1), F12
SUB R1, R1 #32
BNEZ R1, LOOP
SD 8(R1), F16 // 8-32 = -24</td></tr>
<tr><td>23</td><td>Loop unrolling</td><td>Was sind die Gründe für Loop unrolling
• Weniger Sprünge
• Weniger Zugriffe
• Mehr Befehle für Scheduling verfügbar
• Mit Loop Unrolling wird versucht, die Befehlsfolge optimal an die Pipelineausführung anzupassen</td></tr>
<tr><td>58</td><td>Lopp Unrollimg</td><td>Erklären Sie warum ein Programm mit Loop Unrolling schneller abläuft als ohne
• Weniger Sprünge
• Keine Schleifenzähler
• Mehr Möglichkeiten für Scheduling
• Weniger Adressberechnungen</td></tr>
<tr><td></td><td>Loop Unrolling
mit Scheduling</td><td>• Unabhängige Datenfelder
• Keine oder nur wenig datenabhängigkeit
• Stalls werden eingeschoben, wenn nicht anders möglich</td></tr>
<tr><td></td><td>Maschinenbefehl</td><td>Makro Opcode greift auf Tabelle zu, dort steht die Anfangsadresse des Mikroprogramms</td></tr>
<tr><td>43</td><td>Maschinensprache</td><td>Können in Maschinensprache vorliegende Programme für einen Rechner X auf einen Rechner Y zum Ablauf gebracht werden, auch wenn der Rechner Y nicht Mikro programmiert ist
• Ja, mit Simulation</td></tr>
</table>

37	Mehrzweckregister	Weshalb hat sich die Rechner Architektur mit Mehrzweckregistern durchgesetzt • Schnellerer Zugriff, da Registerinhalte weniger oft geändert werden müssen
42	Mehrzweckregister	In welcher Weise können die Register einer Maschine mit Mehrzweckregistern für die Beschleunigung von Programmen genutzt werden. Wie heißt die dabei verwendete SW • Variable werden Registern zugeordnet • Graph Coloring
60	Mehrzweckregister	Nennen Sie Vor und Nachteile von Mehrzweckregistern zur CPU internen Speicherung von Operanden • Vorteile: Register sind schnell, gutes Scheduling • Nachteile: lange Befehle, mehr bits für die Register Adressierung vergrößern Context eines Programms
	Mikroprogramm Mikroprogrammschritt	Welche Mikroprogrammschritte werden nach der Interpretation eines jeden Maschinenbefehls ausgeführt IF, ID, EX, MEM, WB
	Monoprozessor Anlagen	Müssen bei Monoprozessoranlagen die von mehreren Programmen gemeinsam benutzten Dateien(shared files) gegen die simultane Benutzung durch mehrere Programme geschützt werden Wie wird das gegebenenfalls gemacht • Ja • Mit Lockbit / Semaphor
	Monoprozessor Systeme	In Monoprozessorsystemen wird der konkurrierende Zugriff auf Datenfelder mit Semaphoren koordiniert Ich der gleiche Mechanismus auch in SMMP Systemen anwendbar Unter welchen Umständen dürfen dabei die Semaphore in prozessor eigenene Caches gepuffert werden • Ja • Multiprozessorsysteme müssen konsistent(kohärent) sein (letzt gültige Info muß geliefert werden))
35	Multiprozessoren, Gattung	Welche Gattung von Mikroprozessor Systemen verfügt über eine einzigen, durchlaufend adressierten Adressraum • SMMP(Shared Memory Multi Prozessor System)
20	Operanden	Bei einigen Maschinen können die Operanden schon aus dem Registersatz ausgelesen werden, bevor der Befehl dekodiert ist. Welche Architektonische Voraussetzung muß bei diesen Maschinen erfüllt sein • Fixed field coding(die Registeradressen stehen immer an der gleichen Stelle im Befehl)
54	Orthogonale Architektur	Wodurch ist eine orthogonale Architektur gekennzeichnet • Unabhängige Fkt werden unabhängig spezifiziert sein(Fkt und Adressangaben sind unabhängig voneinander angebbar)

	Partial Hit	• Erspart die Berechnung des Übersetzungstafelzeiger • Wenn kein Korrespondenzpar gefunden wird, besteht trotzdem eine hohe Chance, daß die neue virtuelle Seite wenigstens in einem schon benutzten Segment liegt • Bei einem Partialhit wird dann wenigstens der Übersetzungstafelzeiger(PTP) schnell gefunden (Korrespondenzpaar = virtuelle Seite – reelle Seite)
27	Pipelining RAW, WAR, WAW	Datenabhängigkeiten stören das Pipelining. Welche drei Abhängigkeiten gibt es • RAW Read after Write • WAR Write after Read • WAW Write after Write
	Pipelining Störungen	Eine häufig auftretende Störung des Pipeline Betriebes tritt auf, wenn ein Befehl Daten weiter verarbeitet, die der vorher gehende Befehl noch nicht in das Zielregister abgespeichert hat(RAW) Wie kann man Auswirkungen solcher Störungen minimisieren • Scheduling, Forwarding, Bypassing, Short circuiting
	Pipelining Störungen	Wieso stören bedingte Sprünge den Pipeline Betrieb. Mit welchen Methoden kann man diese Störungen klein halten • Branch prediction, Delay Slot, (stall cycle)
69	Pipelining Überlappende Befehls-Ausführung	Gegeben ist folgende Situation in einem Rechner mit überlappender Befehlsausführung (Pipelining, Superscalarrechner) $Instruction_1$: lang andauernder Befehl, erzeugt nach Abschluß der Instruction **n** einen INT $Instruction_2$: in Arbeit ..... $Instruction_{n-1}$: in Arbeit $Instruction_n$: abgeschlossen Was kann das System tun, damit nach der Bearbeitung des INT das Programm richtig fort gesetzt wird • Durch BS alle Befehle fertig laufen lassen in ein Future File nicht ausführbare Befehle werden zurückgestellt Falls $Instruction_1$ noch einmal ausgeführt werden muß ⇒ Future File ignorieren (Future File: alle Register liefern ihr Ergebnis im Future File ab)
61	Pipelining Verarbeitung	Nennen Sie einige Einrichtungen, die notwendig sind, um eine Pipeline Verarbeitung zu ermöglichen • Incrementer für die Adressberechnung • Höhere Speicherbandbreite
	Pipelining Verarbeitung	Nennen Sie einige Einrichtungen die notwendig sind, um eine Pipeline Verarbeitung zu ermöglichen • Es sind 2 MAR(memors Adress Reg) notwendig, da in einem Takt sowohl ein Befehl (IF) als auch ein datum (MEM) gelesen werden muß • es ist ein eigener Addierer notwendig, da der PC bei jedem Takt hoch gezählt werden muß • Einrichtung die die Datenabhängigkeit minimiert, und stall Cycle einschiebt • Einrichtung für results Forwarding(Bypass)
	Pipelining Verfünffachung Gesch	Warum führt eine 5 stufige Pipeline nicht zu einer Verfünffachung der Geschwindigkeit • Pipeline stottert oft wegen stall cycle • Wegen Datenabhängigkeit und Hazards, die den Ablauf in der Pipeline stören • Data Hazards vermeidbar mit: results forwarding control Hazards vermeidbar mit: delayed branching, branch prediction
24	Position Independent Code	Bei virtuell adressierten Maschinen können Programme im reellen Speicher an beliebiger Stelle stehend ablaufen. Weshalb braucht man trotzdem noch **Position Independent Code** • Um allgemein nutzbare Unterprogramme(Methoden) gemeinsam nutzen zu können • Verschiedene Programmierer können Teilstrukturen eines Programms schreiben, das später zusammen gefügt wird
3	Register als Speicher	Welche Vorteile bieten Register als Speicher für Variable • Sie sind schneller • Reduzierter Speicherverkehr (Zwischenergebnisse können im Register stehen bleiben)
	Registersatz	Bei einigen Maschinen können die Operanden schon aus dem Registersatz ausgelesen werden, bevor der Befehl dekodiert ist. Welche architektonische Voraussetzung muß bei diesen Maschinen erfüllt sein • Fixed field coding(die Angabe der Register muß immer an derselben Stelle stehen)

	Scheduling	Erklären Sie den Unterschied zwischen statischen und dynamischen Scheduling Für welche Scheduling Methode wird eine besondere HW Einrichtung benötigt und wie heißt diese • Dynamisch (es wird dynamisch zur Laufzeit entschieden) • Statisch(es wird vorher entschieden) Mindestabstände um Stallcycle zu vermeiden FP ALU Op FP ALU Op = 3 FP ALU Op Store double = 2 Load double FP ALU Op = 1
	Scheduling Dynamic Scheduling	In einem Rechner mit Dynamic Scheduling liegt folgende Situation vor $Instruction_1$: lang andauernder Befehl, erzeugt nach Abschluß der Instruction **n** einen INT $Instruction_2$: in Arbeit ..... $Instruction_{n-1}$: in Arbeit $Instruction_n$: abgeschlossen Was kann das System tun, damit nach der Bearbeitung des INT das Programm richtig fort gesetzt wird • Durch BS alle Befehle (simultan) fertig laufen lassen in ein Future File, und danach wieder aufsetzen Nicht ausführbare Befehle werden zurückgestellt Falls $Instruction_1$ noch einmal ausgeführt werden muß ⇒ Future File ignorieren (Future File: alle Register liefern ihr Ergebnis im Future File ab)
44	Scheduling Dynamic Scheduling	Nennen Sie eine Einrichtung zur Steuerung des „Dynamic Scheduling“ • Scoreboard(Tomasulo Algorithmus)
	Scheduling Dynamic Scheduling	• Es wird alles zur Laufzeit entschieden (siehe S27, Fieger) • Einrichtung für Dynamic Scheduling: Scoreboard(entscheidet, wie weiter verfahren wird)
	Scoreboard	• Dient zur Überwachung der komplexen Abhängigkeiten in der Pipeline • Überwacht die verschiedenen Zustände der Funktionseinheiten, um Structur Hazards aufzudecken • Die Verfügbarkeit der Eingangs Operanden zu heben und die Fertigstelluing der Ausgangs Operanden um Data Hazards zu beheben • Überprüft welche Einheiten frei sind • Schaltet die Befehlsausführung weiter, sagt, aus welcher Funktionseinheit das Ergebnis kommt
50	Semaphor	In einem SMMP wird der konkurrierende Zugriff auf ein Datenfeld mit einem Semaphor koordiniert Unter welchen Umständen darf das Semaphor in den Prozessor eigenen Caches gepuffert werden Wie wird sicher gestellt, daß ein Programm das wiederholt(Spin Lock) auf den Semaphor zugreift, das Freiwerden des Semaphor bemerkt • Wenn ein konsistentes System vorliegt • Beim Freiwerden des Semaphores wird das Lock vom zuletzt zugreifenden Programm auf Null gesetzt
63	Semaphor	Von mehreren Programmen bearbeitete Datenbereiche können durch Semaphore gegen gleichzeitige Zugriffe dieser Programme geschützt werden. Welche Möglichkeiten hat ein Programmierer, auf besetzt anzeigende Semaphore zu reagieren • Spin Lock(verbraucht unnütz Rechnerzeit)(gut bei vielen kurzzugreifenden Prozessen) • Warteschlange / Enqueue(gut bei wenig en lang zugreifenden Prozessen)
64	Semaphor	In einem SMMP System wird der konkurierende Zugriff auf Datenfelder mit Semaphore koordiniert Unter welchen Umständen dürfen die Semaphore in Prozessor eigenen Caches gepuffert werden • Es muß sich um ein konsistentes(kohärent) System handeln(nur zuletzt gültige Daten liefern)
30	Semaphor Zugriff auf Daten	Programme, die die selben Daten bearbeiten, müssen den Zugriff auf diese Daten koordinieren. Mit welchen Mitteln geschieht das. Beschreiben sie die Struktur eines Programmes, das in Konkurrenz zu anderen Programmen eine Variable X verändert. Kommt man beim Ablauf dieser Programme in Multiprozessoranlagen mit den gleichen Koordinierungsmaßnahmen aus • mittels Semaphore(entweder Test & Set, Increment & Decrement, oder Compare und Swap) Test & Set◀ Spinlock Branch on Condition Bearbeiten der Dateien Schreibe frei nach Lock • • ja
71	Serialisierte, synchronis. Zugriffe	Geben Sie ein Bsp dafür, daß bei verändernden Zugriffen auf gemeinsam benutzte Daten die Zugriffe von unterschiedlichen Programmen serialisiert(synchronisiert) werden müssen P1 P2 Read Modify +100 Read Modify Write +200 (**geht verloren**) Write +100

	SMMP	Wie kann es in einem SMMP System bei der Benutzung einer Programm eigenen Var (also kein Zugriff von anderen Programmen) zu einer Inkonsistenz dieser Var kommen, wenn keine Konsistenz bewahrenden Einrichtungen vorhanden sind • Ein Programm kann den Prozessor wechseln, der Cache des neuen Prozessor merkt nicht, daß aktuelles datum im Cache des alten Prozessor Cache steht ⇒ snooping Bus notwendig
	SMMP	Um die Konsistenz in einem mit Prozessor eigenen Caches ausgerüsteten SMMP System aufrecht zu erhalten, können Änderungen an gemeinsam benutzten Daten (shared data) mit „Wirte Update“ durchgeführt werden. Nennen Sie einen Fall, in dem Write Update vorteilhaft ist • Wenn Einträge häufig gemeinsam benutzt werden, aber jede CPU nur wenige Änderungen hintereinander auf diesen Einträgen vornimmt
	SMMP	Bei SMMP Systemen, bei denen Prozessoren und Speichermodule über einen Bus verbunden sind, kann mit einem Snooping Protocol die Cache Kohärenz sicher gestellt werden. Welches andere Verbindungssystem für Prozessoren und Speichermodule gibt es, bei dem ebenfalls ein Snooping Protokoll angewandt werden kann • Ringanordnung, Ringstruktur, Interconnection
74	Speicherbedarf Mikroprogramme	Nennen Sie Methoden, den Speicherbedarf für Mikroprogramme zu reduzieren • Unterprogramme(Methoden) • Selten vorkommende Befehle simulieren
6	Sprung Positions unabhängiger Code	Wie werden Zieladressen in Sprungbefehlen angegeben, bei Maschinen mit positions unabhängigen Code • Relativ zum Befehlszählerstand
4	Sprünge relativ zum PC	Wenn Sprünge relativ zum Befehlszähler ausgeführt werden, hat das zwei Vorteile. Bitte benennen sie diese • Programmcode ist „**Position independent**“ • Sprünge führen meist nicht weit vom derzeitigen Befehlszählerstand weg, deshalb werden für die Sprungadresse weniger als 8bit gebraucht
59	Stack, Queue	Was ist der entscheidende Unterschied zwischen einem Stack und einer Queue • Stack: LIFO • Queue: FIFO
22	Strukturierte Datentypen	Nennen Sie strukturierte Datentypen mit gleichartigen und verschiedenen Elementen • Gleichartige Elemente: Array, String, Sets • Verschiedenartige Elemente: Records
9	Superscalar Maschinen	Was kennzeichnet Superscalarmaschinen • Mehrere unabhängige Befehle werden in einem Taktzyklus auf einmal begonnen (mehr als ein Register steht zur Verfügung)
39	SW Pipelining	Was ist SW Pipelining • Schedular Fkt bei der Schleifen umgeordnet werden, um Datenabhängigkeit zu vermeiden
	SW Pipelining	Wie arbeitet das SW Pipelining und wieso ist es so wirksam bei der Beschleunigung von Programmabläufen • Reorganisation von Schleifen, bei der in einem Schleifendurchlauf Befehle aus verschiedenen Durchläufen der Original Codierung zusammen gestellt werden (verhindert Datenabhängigkeit) • Befehle aus verschiedenen Schleifendurchläufen der Originalcodierung werden in einen Schleifendurchlauf zusammen gestellt
41	TLB Partial Hit	Beschreiben Sie den Übersetzungsvorgang, wenn im Translation Look Aside Buffer ein Partial Hit erfolgt • Bei einem Partial hit wurde im TLB nicht die virtuelle page Nummer, sondern nur die virtuelle Segment-Nr gefunden. Die virtuelle page-Nr kann nun durch eine einstufige Übersetzung errechnet werden, da das Segment schon bekannt ist Bei einem Partial Hit stimmen nur die der SegmentNr entsprechenden Adressbits der zu übersetzenden Adresse mit den entsprechenden Adressbits in mind einem Eintrag im TLB überein
	TLB Prinzip, Funktion	• Die durch die Adressumsetzung gefundenen Korrespondenzpaare(virtuelle - reelle SeitenNr) werden in einem schnelle zugreifbaren, prozessornahen Speicher aufgenommen • Die virtuellen SeitenNr werden in einem assoziativen Speicher gebracht, und die zugehörigen reellen SeitenNr sind in den analogen Zeilen eines zugeordneten nicht assoziativen Speicher unter gebracht • Bei angelegter virtueller SeitenNr (sofern vorhanden) wird sofort die zugehörige reelle SeitenNr ausgegeben, (d.h. e entfallen die Speicherzugriffe der normalen Übersetzung) • Jeder Rechner mit virtueller Adressierung braucht einen TLB, da sonst die Adressumsetzung viel zu langsam wäre • Bei einem Programmwechsel muß der Assoziativspeicher des TLB ungültig gemacht werden Rücksetzen genügt nicht, es würde dann die virtuelle Seite 0 zu multiplen Hits führen Statt dessen werden die Validbits gesetzt, die mit den Hitlines des Assoziativspeicher UND verknüpft sind, und beim Eintragen eines Korrespondenzpaares gesetzt werden

	Übersetzung Arbeitsspeicherzyklen	Mit welcher Einrichtung können Arbeitsspeicherzyklen für die Übersetzung von virtuellen Adressen in reelle Adressen vermieden werden. Erklären Sie die Arbeitsweise dieser Einrichtung • TLB • Die virtuellen SeitenNr werden in einen assoziativen Speicher gebracht, und die zugehörigen reellen SeitenNr sind in den analogen Zeilen eines zugeordneten nicht assoziativen Speichers untergebracht Bei angelegter virtueller SeitenNr wird(sofern vorhanden) sofort die zugehörige reelle Seiten Nr ausgegeben ⇒ es entfallen die Speicherzugriffe der normalen Übersetzung • Da jedes Programm dieselben virtuellen Seiten verwenden kann, muß der TLB bei Programmwechseln ungültig gemacht werden
70	Übersetzung einstufig Übersetzungstafel	Warum werden bei virtuell adressierten Maschinen bei einstufiger Übersetzung für jedes Programm unabhängig von der Größe des Programms die vollen Übersetzungstafeln implementiert • falsch generierte Adressen bleiben innerhalb der Übersetzungstafel ⇒ Fehlererkennung von errechneter @ ist Validbit = 0 ⇒ INT ⇒ BS bricht Programm mit Speicherschutzverletzung ab
	Übersetzung einstufig Übersetzungstafel	Braucht man bei der einstufigen Übersetzung virtueller Adressen in reelle Adressen eine Übersetzungstafel für das gesamte System oder braucht man mehrere Begründen Sie die Antwort • man braucht immer eine Übersetzungstafel pro Programm • jedes Programm braucht eine eigene Adresse • falls die lange Übersetzungstafel verkürzt würde, könnte bei einer Fehladressierung, ein Programm in den Adressraum eines anderen Programm gelangen
	Übersetzung einstufig	• Die virtuelle Adresse besteht aus einer SeitenNr und dem Index in dieser Seite • Jedes Programm hat seine eigene Übersetzungstafel, es kann also kein Programm ein anderes stören • 1 Speicherzugriff
52	Übersetzung zwei stufig	Warum der Speicherbedarf für Übersetzungstafeln bei zweistufiger Übersetzung i.a. geringer als bei einstufiger Übersetzung • **Einstufige Übersetzung:** es gibt 1 Übersetzungstafel für das Programm (Anzahl der Einträge = Anzahl der Seiten / page) • **Zweistufige Übersetzung:** es gibt eine Segmenttabelle und nur soviel Page-Tafeln, wie das Programm wirklich benötigt
	Übersetzung zwei stufig	• Die SegmentNr ist der Index in einer Segmenttabelle, die für jedes Programm nur einmal existiert • 2 Speicherzugriffe
55	Unnormalisierte Gleitpunktzahlen	Welche Vorteile hat das Rechnen mit unnormalisierten Gleitpunktzahlen • Zuverlässigkeit des Rechenergebnisses ist höher
12	Vektor Ladebefehl	Was für Parameter braucht ein Vector Ladebefehl • Vecor-Anfangsadresse • Vector-Länge • Stride(Schrittweite) die Anzahl der Elemente in einer Zeile / Spalte
11	Vektor Rechner	Welche Vorteile haben Vectorrechner • Eine Vector-Addition erledigt die Arbeit einer ganzen Schleife • Jede Berechnung ist unabhängig von der vorhergehenden (keine data Hazards) • keine Control Hazards
40	Virtuelle Adressierung	Geben Sie Gründe für die Einführung der virtuellen Adressierung an, und dessen Vorteile • Programme müssen nicht vollständig im ASP stehen, bei Bedarf können benötigte Teile nachgeladen werden ⇒ weniger Ladezeit und ASP Zugriffe • Programm können gegeneinander geschützt werden(ein Adressbereich gehört nur einem Programm) • viele Programme können gleichzeitig im ASP stehen ⇒ große Chance, daß in der Ready Queue ablauf bereite Programme stehen ⇒ bessere Prozessor Auslastung
10	VLIW	Was bedeutet VLIW • Very long instruction word • Viele Befehle werden in einem sehr langem Befehlswort zusammen gefaßt