Rechnerorganisation

Zusammenfassung

Fabian Damken

8. November 2023

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Inhaltsverzeichnis

1 Zahlendarstellung

1.1 Kommazahlen

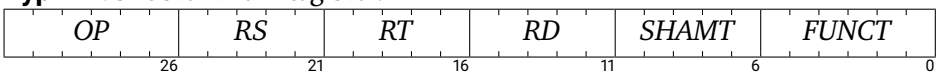
Gleitkommazahlen nach IEEE 754

Typ	Format
Single	

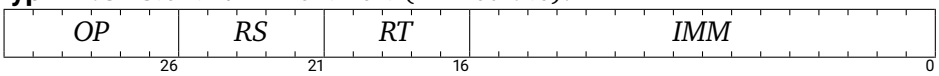
2 MIPS

2.1 Befehlstypen

R-Typ Das R steht für Register.



I-Typ Das I steht für Direktwert (Immediate).



J-Typ Das J steht für Sprung (Jump).

3 Theoriefragen

3.1 Zahlendarstellung

Der IEEE 754 Standard kennt Sonderfälle. Benennen Sie die in der Vorlesung behandelten Sonderfälle und geben Sie die Kodierung an.

normalisiert	$\pm$	$0 < C < max$	*Beliebiges Bitmuster*
denormalisiert	$\pm$	0	*Beliebiges Bitmuster* $\neq 0$
Null	$\pm$	0	0
Unendlich	$\pm$	max	0
Keine Zahl, NaN	$\pm$	max	*Beliebiges Bitmuster* $\neq 0$

Läßt sich jede reelle Zahl als Gleitkommazahl nach IEEE 754 darstellen? Nein, da:

1. nicht jede reelle Zahl als Bruch darstellbar ist ($\mathbb{Q} \subset \mathbb{R}$).
2. nicht jeder Bruch dargestellt werden kann (bspw. $\frac{1}{3}$).

Was ist die größe/kleinste darstellbare Zahl einer n-Bit Zahl in der Festkommanotation $k.l$ mit Vorzeichen-Bit? Die größte darstellbare Zahl ist $2^k - 1 + \sum_{i=1}^{l} \frac{1}{2^i}$.
Die kleinste darstellbare Zahl ist $-(2^k - 1 + \sum_{i=1}^{l} \frac{1}{2^i})$.

In welche (mathematischen) Form werden Gleitkommazahlen dargestellt? Eine Gleitkommazahl z wird in der Form $z = \pm m \cdot b^{\pm e}$ dargestellt.

3.2 Maschinennahe Programmierung

Was ist der Unterschied zwischen einem byte- und wortadressiertem Speicher? Bei einem byteadressiertem Speicher hat jedes Byte eine eigene Adresse, bei einem wortadressiertem Speicher nur jedes Wort.

Erklären Sie den Unterschied zwischen Big-Endian und Little-Endian. Bei Big-Endian wird das höchstwertige Byte (MSB) in der niedrigsten Speicheradresse abgelegt.
Bei Little-Endian wird das niederwertigste Byte (LSB) in der niedrigsten Speicheradresse abgelegt.

Was sind Immediates? Immediates sind Direktwerte, welche direkt im Assemblercode stehen (konstante Werte, Literale).

Wann sollten Daten in Register gespeichert werden? Daten sollten in Registerns gespeichert werden, wenn diese erneut verwendet werden.

Gegeben sei ein wortadressierter Speicher. Wie viele Wörter mit einer Länge von n Byte lassen sich maximal adressieren, wenn die Adresse m Byte groß ist? Es lassen sich maximal 2^m Worte adressieren.

Gegeben sei ein byteadressierter Speicher. Wie viele Wörter mit einer Länge von n Byte lassen sich maximal adressieren, wenn die Adresse m Byte groß ist? Es lassen sich maximal $\frac{2^n}{m}$ Worte adressieren.

Was ist der Unterschied zwischen einer Funktion und einer Prozedur? Eine Prozedur hat keine Rückgabe (`void`), eine Funktion hat eine Rückgabe.

Wann ist es sinnvoll, Programme direkt in Assembler (statt in C oder Java) zu schreiben? Es ist sinnvoll, Programme direkt in Assembler zu schreiben, wenn:

- der Speicherplatz begrenzt ist.
- eine schnelle Reaktionszeit nötig ist (Echtzeitanwendungen).
- manuelle Performanzopimierung nötig ist.
- direkter Zugriff auf den Prozessor nötig ist.

Nennen Sie einige Vorteile einer Hochschprache wie C gegenüber Assembler.

- Vorhandene Kontrollstrukturen.
- Bessere Verständlichkeit des Codes.
- Vorhandene Datenstrukturen.
- Typprüfung.
- Besse Wartbarkeit des Codes.

3.2.1 MIPS

Ist MIPS wort- oder byteadressiert? MIPS ist byteadressiert.

Wo wird die Rücksprungadresse bei einem Aufruf von `jal` gespeichert? Die Rücksprungadresse wird in dem Register `$ra` gespeichert.

Welche Register müssen von dem Aufrufer, welche von dem Aufgerufenen gesichert werden? Der Aufrufer sichert die Register `$t0 - $t9`, `$a0 - $a3`, `$v0 - $v1` und den Stack unterhalb von `$sp`.
Der Aufgerufene sichert die Register `$s0 - $s7`, `$ra`, `$sp` und den Stack oberhalb von `$sp`.

Wie alloziert man Speicher auf dem Stack und wie gibt man diesen wiedr frei? Alloziert wird Speicher auf dem Stack, in dem die gewünschte Anzahl Bytes von `$sp` subtrahiert wird.

Freigegeben wird der Speicher auf dem Stack analog, in dem die freuzugebene Anzahl Bytes zu `$sp` addiert wird.

Warum muss bei mehrfachem Aufruf von Unterfunktionen/-prozeduren immer `$ra` auf dem Stack gesichert werden? Bei einem erneuten Aufruf von `jal` wird das Register `$ra` überschrieben, wodurch die bisherigen Rücksprungadressen verloren gehen würden.

Nennen Sie alle Adressierungsarten, die MIPS spezifiziert. MIPS unterstüzt Adressierung durch Register, Adressierung durch Direktwerte aus Instruktionen, relative Adressierung zu der Basisadresse und Pseudodirekte Adressierung.

3.2.2 IA32

Beschreiben Sie kurz den Registersatz, welcher bei IA32 zur Verfügung steht. Registersatz:

ID	Name	Spezielle Bedeutung
EAX	Accumulator	Artithmetik, Ein- und Ausgabe, Interrupts
EBX	Base Register	-
ECC	Count Register	Schleifen
EDX	Data Register	Multiplikation, Division, Portadressen für `IN/OUT`
EBP	Base Pointer	Zeiger auf tempärare Speicheradressen im Stack (bspw. Stackframe)
ESP	Stack Pointer	Zeiger auf aktullen Stackkopf
ESI	Source Index	Quelle für String-Operationen, Nutzung für indirekter Adressierung
EDI	Destination Index	Ziel für String-Operationen, Nutzing für indirekte Adressierung

3.3 Mikroarchitektur

Erklären Sie die Begriffe RISC und CISC. RISC steht für *Reduced Instruction Set Computer*. Diese Computer haben einen einfach gehaltenen Befehlssatz, in dem ausschließlich Register (und Direktwerte) als Parameter genutzt werden können. Somit muss bei einem Zugriff auf den Speicher zuerst der Wert in ein Register geladen werden, damit mit diesem gerechnet werden kann (MIPS: `lw/sw` Befehle).

CISC steht für *Complex Instruction Set Computer*. Diese Computer haben einen sehr komplexen Befehlssatz, in dem auch erlaubt wird, direkt mit Daten aus dem Speicher zu rechnen.

Da Speicherzugriff (im Vergleich zu Registerzugriffen) sehr lange dauern, ist die Abarbeitungszeit von Befehlen in einem CISC sehr unterschiedlich, wobei diese in einem RISC relativ konstant bleibt. Dadurch sind Beschleunigungsmethoden wie Pipelining in einem RISC deutlich einfacher umzusetzen.

Das *reduced/complex* bezieht sich nicht auf die Anzahl der vorhanden Befehle im Befehlssatz.

Was sind die Unterschiede zwischen einem Eintakt-, Mehrtakt- und Pileline-Prozessor? Bei einem Eintakt-Prozessor wird jeder Befehl in einem Takt abgearbeitet, bei Mehrtakt-Prozessoren kann eine Abarbeitung mehre Takte benötigen, wobei jede Ausführung in mehrere Teilschritte zerlegt wird. Bei einem Pipeline-Prozessor werden die Teilschreitte eines Befehls zeitgleich mit den Teilschritten des nächsten Befehls ausgeführt.

Warum benötigt das Registerfeld des vorgestellten Prozessors ein Write-Enable-Signal? Das Schreiben muss aktiviert werden, da die Daten nicht bei jedem Befehl modifiziert werden (dürfen), beispielsweise bei Sprungbefehlen.

Erläutern Sie anhand des Datenpfades, wie bei einem `lw`/`sw`-Befehl die Adresse berechnet wird. Zur Berechnung der Speicheradresse wird die Basisadresse des Registers an den Eingang `A` der ALU angelegt und der Direktwert an den Eingang `B`. Dann wird eine Addition ausgeführt, das Ergebnis ist die Speicheradresse.

Welcher Teil des Prozessors bestimmt, welche Rechenoperation die ALU ausführt? Das Steuerwerk bestimmt `ALUControl`.

Worin unterscheiden sich eine Harvard- und eine Von-Neumann-Architektur? Bei der Harvard-Architektur liegen die Instruktionen in einem von den Daten getrennten Speicher, bei der Von-Neumann-Architektur wird ein gemeinsamer Speicher für Instruktionen und Daten genutzt.

3.3.1 Performanz

Durch was wir die Takfrequenz eines Eintakt-Prozessors bestimmt? Die langsamste Instruktion bestimmt die Taktfrequenz. Die Dauer einer Instruktion ist auf die Verzögerungszeiten im Datenpfad zurückzuführen.

Erklären Sie die Begriffe Latenz und Durchsatz im Bezug auf einen Prozessor.

Latenz Die Latenz beschreibt die Ausführungszeit einer Instruktion.

Durchsatz Der Durchsatz beschreibt die Anzahl der Instruktionen, die in einem bestimmen Zeitfenster bearbeitet werden können.

In welche Phasen wird die Ausführung einer Instruktion bei dem vorgestellten Pipeline-Prozessor unterteilt? Die Ausführung einer Instruktion wird in Fetch-, Decode-, Execute-, Memory-, und Writeback-Phase unterteilt.

Was versteht man unter einem Hazard? Hazard sind Abhängigkeiten zwischen Pipeline-Stufen. Es wird zwischen *Data-Hazard* und *Control-Hazard* unterschieden.

Data-Hazard Ein Data-Hazard trinnt dann auf, wenn Daten gelesen werden, welche in einem vorherigen Befehl geschrieben werden (sollten), dieser aber noch nicht komplett ausgeführt wurde.

Control-Hazard Ein Control-Hazard trinnt dann auf, wenn Instruktionen ausgeführt werden, welche von einem vorherigen Befehl unterbrochen werden (beispielsweise durch einen Sprung). Da dieser Sprung erst relativ spät entschieden wird, wurde die Ausführung von späteren Instruktionen bereits begonnen.

Bei der Entwicklung moderner Prozessoren werden viele Resourcen in die Verbesserung der Sprungvorhersage gesteckt. Warum wirkt sich ein falsch vorhergesagter Sprung negativ auf die Performanz aus? Wurde ein Sprung falsch vorhergesagt, müssen bereits berechnete Instruktionen verworfen werden (Flush). Diese Takte hätten für die Berechnung von den korrekten Instruktionen genutzt werden können.

Bei Prozessoren unterscheidet man zwischen in-order und out-of-order Ausführung von Instruktionen. Welcher Ansatz ist schneller? Können Sie sich Einsatzszenarien für den anderen Ansatz vorstellen? Bei out-of-order Ausführung werden Instruktionen umsortiert, um Hazards zu vermeiden. Bei in-order Ausführung werden alle Instruktionen in der eigentlichen Reihenfolge ausgeführt und im Falle von Hazards die CPU angehalten (`HALT`) oder NoOps eingeführt.

Da für ot-of-order Ausführung mehr Hardware zur Umsortierung vorhanden sein muss, wird bei billigen Prozessoren noch in-order Ausführung eingesetzt.