Thomas Merkel

Rechnertechnologie Probeklausur

von-Neumann-Architektur

- Befehlshol- und Interpretationsphase (jeweils 1 Befehl wird geladen)
- Befehlsausführungsphase (jeweils ein Ergebnis wird bestimmt)
    - Prinzip: SISD (single instruction, single data)

Vorteile:

- Gemeinsamer Speicher für Daten und Programme
- Während dem Programm sind Bibliotheken nachladbar
- Laden der Programme an x-beliebigen Stellen im Arbeitsspeicher

Nachteile

- Langsamer Speicherzugriff
- Daten können nur hintereinander gelesen werden => nicht gleichzeitig

Havard-Architektur

- Verwendung von getrennter Speicher für Daten und Programme
- Schneller Zugriff auf Prozessorexterne Daten
- In einem Zyklus können 2 Daten gleichzeitig gelesen werden.

Zuordnung Prozessorgattung zu Architektur

- von-Neumann-Architektur:
    - alle Mikroprozessoren, ein Teil der Mikrocontroller
- Harvard Architektur:
    - bei einigen Mikrocontrollern und allen Signalprozessoren

Da Signalprozessoren eine höhere Rechenleistung bereitstellen sollen, muss die Harvard Architektur verwendet werden. Da bei dieser mehrere Daten parallel verarbeitet werden können.

Bei Mikrocontrollern können beide Architekturen vorkommen. Falls der Mikrocontroller 2 Daten in einem Zyklus verarbeiten muss, dann wird die Harvard Architektur verwendet. Falls es jedoch ausreicht, dass Daten hintereinander verarbeitet werden, ist die von-Neumann-Architektur angebracht.

Mikroprozessoren entsprechen der von-Neumann-Architektur, weil ein Mikroprozessor z.B. keinen kontrollinternen Speicher hat, in dem er mehre Daten parallel verarbeiten muss.

Thomas Merkel

Speichertypen im Vergleich

Statischer Speicher (SRAM)

Stelle:

- Prozessorinterner Speicher
  - liegt im Adressraum des Prozessors
- Prozessorexterner Speicher
  - liegt im Standardarbeitsspeicher eines Prozessors (insbesondere bei Signalprozessoren oder Controllern)

Begründung für Einsetzung:

- Kurze Zugriffszeiten
- Als Prozessorinterne oder Prozessorexterne Speicher realisiert
- Lese- und Schreibgeschwindigkeit ist gleich
- Wiederholtes Lesen möglich
- Der Speicherinhalt bleibt solange erhalten, solange Versorgungsspannung bereitsteht

Dynamischer Speicher (DRAM)

Stelle:

- Standardspeicher des Motherboards jedes Rechners
  - Kondensator

Begründung für Einsetzung:

- Zur Aufrechterhaltung ist ein „Refresh“ notwendig (auf Dauer automatisch Spannungsverlust. Erneuerung der Spannung, damit diese vorhanden bleibt)
- Geringer Speicherbereich
- Geringe Verlustleistung
- Längere Zugriffszeiten

EPROM

Stelle:

- Befindet sich auf dem Motherboard, Ladung bleibt auf dem „Gate“ auch wenn die Versorgungsspannung wegfällt.

Begründung für Einsetzung:

- Prozessorinterner Speicher bei Signalprozessoren oder Controller
- Prozessorexterner Speicher z.B. BIOS des PC
- EPROM (ist elektrisch schreibbar, löschbar mit UV-Licht)
- EEPROM (ist elektrisch schreib- und löschbar)
- FLASH-EPROM (ist elektrisch schreib- und löschbar in Speicherzellen)

Vor- und Nachteile von Prozessoren mit einer Vielzahl von Registern

Vorteile	*Nachteile*
• kürzeste Zugriffzeiten • direkt Adressierbar • verfügbar als Adress-, Datenregister, Akkumulatoren • verwendet als Registerfile	• kleine Speicherkapazität • muss oft adressiert werden

Thomas Merkel

Speicherverwaltung, dynamische Erweiterbarkeit des Befehlssatzes

Auswirkungen der virtuellen Speicherverwaltung

- Physikalische Begrenztheit verbergen z.B. Betrieb eines virtuellen Speichersystems in einem Rechner mit einer MMU (Memory Management Unit)
- Die aktiven Prozesse dürfen selbst niemals mehr Speicherplatz belegen als physikalisch zur Verfügung steht.
- Virtuelle und nicht physikalische Adressen werden verwendet. Diese Umwandlung übernimmt die MMU.
- Diese virtuelle Speicherverwaltung ist heute beinahe in jedem modernen Betriebssystem.

Auswirkungen dynamische Erweiterbarkeit (die einem Befehlssatz zugrunde liegt)

- Erweiterung des Umfeldes des Maschinenbefehlssatzes z.B. Mikroprogrammierung des Steuerwerks des Mikroprozessors.

Datenformate

Datenformate zur Darstellung von vorzeichenlosen Zahlen

8 Bit	0 bis 255	unsigned char	0 bis 0FFH	1 Byte
16 Bit	0 bis 65535	unsigned short	0 bis 0FFFFH	2 Byte
32 Bit	0 bis 2^{32}-1	unsigned int unsigned long	0 bis 0FFFFFFFH	4 Byte

Verwendung und Implementierung:

- Zeiger in Hochsprachen C
- Adressen und Adressrechenoperationen in Assembler
- Programmierung von Schnittstellen in Assembler

Datenformate zur Darstellung vorzeichenbehafteten Zahlen

8 Bit	-128 bis 127	char	80FH .. 0 .. 7FH	1 Byte
16 Bit	-32768 bis 32767	short	8000FH .. 0 .. 7FFFH	2 Byte
32 Bit	-2^{31} bis 2^{31}-1	int / long	80000000 .. 0 .. 7FFFFFFFH	4 Byte

Datenformate zur Darstellung gebrauchten rationalen Zahlen

float	$-3{,}4 \cdot 10^{38}$ bis $3{,}4 \cdot 10^{38}$
double	$-1{,}7 \cdot 10^{308}$ bis $1{,}7 \cdot 10^{308}$
long double	$-1{,}1 \cdot 10^{4932}$ bis $1{,}1 \cdot 10^{4932}$

Verwendung und Implementierung

- Zusätzliche Darstellung des Wertebereichs 0<=x<1
- Gebrochene rationale Zahlen werden immer vorzeichenbehaftet dargestellt
- Darstellung als
  - Floating Point Zahl
  - Fix Point Zahl
- Floating Point Zahl, Definition in „C“ als float oder double
- Zahl ist grundsätzlich normiert enstprechend

  Zahl = Vorzeichen (+/-) • 1. Mantisse • 2^{E}

Problem: die Zahl 0 ist nicht darstellbar
Lösung: Verwendung einer Kombination aus dem Exponenten (negative Zahlen im Exponenten haben einen Wert mehr)

- Vorzeichen, Gesamtvorzeichen der Zahl (1 Bit)
- Mantisse, Grundsätzlich positiv (23 Bit float, 52 Bit double)
- Exponenten, Vorzeichenbehaftet (8 Bit float, 11 Bit double)

Pentium, Addition / Subtraktion; Multiplikation / Division

Beim Befehlssatz des Pentium wird für die Addition und Subtraktion nicht zwischen vorzeichenbehafteten und vorzeichenlosen Zahlen unterschieden, dies geschieht nur bei der Multiplikation und Division.

Ist es sinnvoll in Hochsprachen (z.B. C) einen Unterschied zwischen vorzeichenlosen und vorzeichenbehafteten Zahlen bei der Definition von Zahlen zu machen?

- Ja, es ist sinnvoll, denn der Compilier muss wissen ob das MSB (Most Significant Bit) der Vorzeichenregelung entspricht.

Ist bei Multiplikation und Division eine Mischung aus vorzeichenlosen und vorzeichenbehafteten Zahlen in einem Befehl möglich?

- Nein, eine Mischung ist nicht Möglich, da sonst keine Unterscheidung mehr stattfinden kann.
- Beispiel (gilt nur bei einem Byte):
  - eine Zahl >=128 multipliziert mir einer anderen
  - das MSB ist immer gesetzt
  - Multiplikation mit einer negativen Zahl
  - Ergebnis stimmt nicht mehr

Assembler Adressierung

Unterschied zwischen direkter und indirekter Adressierung

Direkte Adressierung

Bei dieser Adressierung wird die Adresse im Programmsegment platziert. Die direkte Adressierung kann auf Quell- und Zieloperanden angewendet werden. Bedingt durch die Platzierung der Adresse im Programmsegment ist diese während des Programmablaufs nicht mehr änderbar.

- Adresse des Operanden steht im Befehl
- In Hochsprache „C“ für globale Variablen eingesetzt
- z.B. MOV AX, [2034H]; Inhalt der Speicherstelle 2034H in AX
- Anzahl der Bytes hängt vom Registernamen ab

Indirekte Adressierung

Bei der indirekten Adressierung wird die Adresse in einem Adressregister platziert und kann dadurch während der Laufzeit des Programms abgeändert werden. Die Adressierung ist auf Quell- und Zieloperanden anwendbar.

- Adresse des Operanden steht im Adressregister
- In der Hochsprache „C“ für berechnende Variablen einsetzbar
- z.B. MOV AX [BX]; Speicherstelle auf BX in AX
- Anzahl der Bytes hängt vom Registernamen ab

Lassen sich in einem Befehl 2 direkte Adressen verwenden?

Keine Unterstützung des Befehlssatzes. *mov* verlangt mindestens ein Register oder Segment. Der Adressbus ist beschränkt und kann nicht zwei Speicherstellen gleichzeitig holen, deswegen muss immer ein Wert im Register liegen.