Übung 1

Armin Kleinert und Ruth Höner zu Siederdissen

Aufgabe 1: Das Von-Neumann-Rechnermodell

Mit dem von-Neumann-Rechnermodell wurde erstmalig das Konzept für einen echten „general-purpose Computer“ vorgeschlagen. Was ist darunter zu verstehen?

Ein “general-purpose Computer” ist nicht für die Lösung eines spezifischen Problems gedacht, sondern allgemein für die Lösung von Problemen. Welches Problem der Computer letztendlich lösen kann ist von den Programmen auf dem Rechner bzw. von der Programmierung abhängig.

„Programme sind auch nur Daten“ ist eine grundlegende und eng mit dem von-Neumann-Rechnermodell verbundene Sichtweise. Was ist darunter zu verstehen?

Das von-Neumann-Rechnermodell kennt nur einen Speicher. Das Programm, das aus den Instruktionen/Befehlen besteht, wird im selben Speicher abgelegt wie die Daten, mit denen gerechnet wird. Somit können Code und Daten nicht unterschieden werden. Was am Ende ein Befehl ist und was Daten, mit denen man rechnet, hängt von der Interpretation ab. Dies kann allerdings zu Problemen führen, da die Daten als Befehle interpretiert werden können und andersherum.

Das von-Neumann-Rechnermodell setzt sich aus vier Hauptbestandteilen zusammen. Welche Bestandteile sind dies und welchem Zweck dienen sie?

Die vier Hauptbestandteile des von-Neumann-Rechnermodells sind der Prozessor (CPU), der Hauptspeicher, das Ein- und Ausgabesystem und ein Hauptverbindungsnetzwerk (BUS). Der Prozessor ist die zentrale Steuerung des Systems. Er kontrolliert den Datenfluss und die Ausführung aller Befehle. Der Prozessor ist über das Verbindungsnetzwerk mit dem Hauptspeicher und dem Ein- und Ausgabesystem verbunden. Über diese Verbindung werden sämtliche Daten übertragen. Im Hauptspeicher werden Daten und Programme in Sequenzen abgespeichert. Das Ein- und Ausgabesystem ist der Zugang zum System. Hier werden Programme und Daten eingegeben, beispielsweise über Maus, Tastatur, Mikrofon beziehungsweise ausgegeben, zum Beispiel über Monitor oder Lautsprecher.

Im von-Neumann-Rechnermodell gibt es einen Daten- und einen Befehlsprozessor als Bestandteil der CPU. Welche Aufgaben werden von welchen Komponenten dieser Prozessoren erfüllt?

Der Befehlsprozessor, auch Steuereinheit genannt, interpretiert die CPU-Befehle und formt daraus Steuersignale. Diese werden dem Datenprozessor, auch Recheneinheit genannt, übergeben, damit dieser die Befehle ausführen kann. Ein- und Ausgabebefehle sowie Speicherbefehle werden vom Datenprozessor mit Hilfe des Ein- und Ausgabesystems und des Hauptspeichers ausgeführt. Der Befehlsprozessor sagt demnach was gemacht werden muss und der Datenprozessor ist für die Ausführung zuständig.

Das von-Neumann-Rechnermodell unterscheidet zwischen Daten- und Adressbus. Warum macht das Sinn?

Würde man einen Bus für sowohl Daten als auch Adressen verwenden, müssten diese seriell hintereinander abgeschickt werden, also erst die Adresse und dann die Daten. Durch die Trennung in Daten- und Adressbus können Adresse und Daten parallel übertragen werden. Dies erhöht die Bearbeitungsgeschwindigkeit.

Die Arbeitsweise eines von-Neumann-Rechners wird durch die Bezeichnung SISD allgemein charakterisiert. Welches Prinzip verbirgt sich hinter dieser Abkürzung? Was für andere Kategorien gibt es nach Flynn?

SISD steht für Single Instruction, Single Data und meint, dass der Prozessor zu jeder Zeit nur genau einen einzelnen Befehl ausführt und damit nur einen Wert manipulieren kann. Zum Beispiel den Wert einer Speicherstelle überschreiben.

Weitere Kategorien nach Flynn sind:
SIMD: Steht für Single Instruction, Multiple Data und meint, dass eine Instruktion auf mehreren Daten parallel angewendet wird.
MISD: Steht für Multiple Instruction, Single Data. Hier werden mehrere Instruktionen gleichzeitig auf den gleichen Daten ausgeführt.
MIMD: Kurz für Multiple Instruction, Multiple Data. Wie bei MISD können mehrere Instruktionen gleichzeitig laufen und wie bei SIMD können diese Instruktionen mehrere Daten gleichzeitig bearbeiten.

Bahnbrechend neu am von-Neumann-Rechnermodell war das Konzept einer quasi universellen Programmierbarkeit. Erörtern Sie in diesem Zusammenhang die Begriffe Maschinencode, Assemblersprachen sowie Ein- und Mehr-Adress-Befehle.

Universelle Programmierbarkeit bedeutet, dass Programmiersprachen sämtliche Funktionen berechnen können, die auch eine universelle Turingmaschine ausführen kann. Korrekt
Maschinencode (auch Maschinen-lesbarer Code genannt) ist eine Abfolge von Befehlen, die von der CPU ausgeführt werden können. Die Befehle können die Funktionalitäten des Computers und somit die Berechnung der Funktionen steuern. Assemblersprachen bieten eine “Menschenlesbare” Form des Maschinencodes. Manche bieten auch zusätzliche Features zur Vereinfachung an. Der Code in Assemblersprache wird vor der Ausführung in Maschinencode umgewandelt.

Bei Ein-Adress-Befehlen und Mehr-Adress-Befehlen handelt es sich um Formate von Befehlen in Maschinen- und Assemblercode. Ein-Adress-Befehle bestehen im Code nur aus 2 Teilen: Dem Mnemonik (OP-Code bzw. menschenlesbarer Befehl) und einem Wert (Literal, Register oder Position im Speicher). Dazu gehören die Befehle MUL, DIV und die Varianten von JMP in NASM.
Mehr-Adress-Befehle können mehrere Argumente (Literal, Register oder Position im Speicher) akzeptieren. Klassische Beispiele sind MOV, ADD, IDIV (1-3 Argumente) oder SUB in NASM.

Charakteristisch für das von-Neumann-Rechnermodell ist ein Zwei-Phasen-Konzept der Befehlsverarbeitung. Welches Problem wird damit auf welche Weise gelöst?

Beim Zwei-Phasen-Konzept wird in der ersten Phase (Interpretationsphase) der Inhalt einer Speicherstelle, auf die der Program Counter zeigt, gelesen und als Befehl interpretiert. In der zweiten Phase (Ausführungsphase) wird der Befehl ausgeführt. Dafür wird der Inhalt aus der Speicherzelle geholt, deren Adresse in dem Befehl steht, und als Daten/ Operanden interpretiert.

Mit dem Zwei-Phasen-Konzept wird das Problem gelöst, dass sämtliche Daten, also Befehle und Operanden, in einem gemeinsamen Speicher liegen und nicht voneinander zu unterscheiden sind. Der Program Counter zeigt auf eine bestimmt Speicherstelle und zeigt somit, wo man sich im Programm befindet. Der Inhalt der Zelle, auf die der Program Counter zeigt, wird immer als Befehl interpretiert. Die Adressen für die Daten, mit denen der Befehl ausgeführt werden soll, sind im Befehl selbst enthalten.

Die Architektur eines klassischen von-Neumann-Rechners führte schon bald zu einem gewichtigen Problem, dem von-Neumannschen „Flaschenhals". Was ist darunter zu verstehen und wie versuchte man später dieses Problem zunächst zu umgehen?

Die Bezeichnung von-Neumannscher „Flaschenhals" bezieht sich auf zentrale Verbindung zwischen CPU und Hauptspeicher. Über diese Verbindung fließen in dieser Architektur sämtliche Daten und Befehle hin und her. Dies führt dazu, dass das System langsamer wird, denn die Daten kommen eventuell nicht schnell genug von A nach B, um dort bearbeitet zu werden.

Um dieses Problem zu umgehen, wird eine Mischform der Harvard-* und der von-Neumann-Architektur verwendet. Es gibt weiterhin nur einen Hauptspeicher und eine Hauptverbindung zwischen diesem Speicher und der CPU. Innerhalb des Prozessors gibt es aber zwei Caches, in denen Befehle und Daten getrennt voneinander zwischengespeichert werden. Das führt dazu, dass die Befehle und Daten nicht für jede Ausführung über diese Hauptverbindung transportiert werden müssen.

*In der Harvard-Architektur sind die Speicher von Befehlen und Daten getrennt. Dadurch würde der Flaschenhals, wie oben erwähnt, kein Problem mehr darstellen. Diese Architektur wurde aber aus verschiedenen Gründen (unter anderem die Kosten) nicht komplett umgesetzt.

Quellen

Eure Korrektur macht besonders viel Spaß, da es hauptsächlich abhacken ist xD

3/3

- Vorlesungsfolien
- Assembler-Cheatsheet aus dem Whiteboard
- NASM-Manual
- SISD: https://www.itwissen.info/SISD-single-instruction-single-data-SISD-Rechnerarchitektur.html
- SIMD: https://www.itwissen.info/SIMD-single-instruction-multiple-data-SIMD-Rechnerarchitektur.html
- MISD: https://www.itwissen.info/MISD-multiple-instruction-single-data-MISD-Rechnerarchitektur.html
- MIDM: https://www.itwissen.info/MIMD-multiple-instruction-multiple-data-MIMD-Rechnerarchitektur.html
- C-- Spezifikation: https://www.microsoft.com/en-us/research/wp-content/uploads/2016/07/ppdp.pdf
- Universelle Programmierbarkeit: https://de.wikipedia.org/wiki/Turing-Vollst%C3%A4ndigkeit
- Ein-Adress-Befehle & Mehr-Adress-Befehle: https://www.geeksforgeeks.org/difference-between-2-address-instruction-and-1-address-instructions/