Abgabe von Bruno Stendal, Martin Baer, Lukas Gewinner und Christian Schäfer
1. Aufgabenblatt zum Kurs

TI2: Rechnerarchitektur

von Bernadette Keßler

bis Freitag, den 11.11.2022, 10:15 Uhr.

Das Von-Neumann-Rechnermodell

1946 wurde das von-Neumann-Rechnermodell vorgestellt, das die Rechnerarchitektur bis heute maßgeblich beeinflusst. Arbeiten Sie die grundlegenden Organisationsprinzipien und Besonderheiten dieses Modells heraus, indem Sie folgende Fragen möglichst prägnant und in eigenen Worten beantworten.

1. Ein general - purpose Computer ist ein Computer, der nur einen Arbeitsspeicher hat, der Operatoren (Programmebefehle) und Daten gleichermaßen speichert und sie nicht unterscheidet. Nur bei der Interpretation des gespeicherten wird entschieden ob er ein Operator ist oder eine Datei. Diese Architektur ist anfällig auf Viren die sich als Operator ausgeben und ausgeführt werden, da sie so interpretiert werden kann.

2. Die Sichtweise ist so zu verstehen, dass Programmebefehle wie auch Daten im selben Arbeitsspeicher gespeichert werden. Sie werden nicht unterschieden und auch nicht in verschiedene Arbeitsspeicher gespeichert. Generell ist es so zu verstehen, dass ein Programmbefehle ein Programm darstellt, die gespeichert werden und von der CPU aufgerufen und danach ausgeführt werden.

3. Die Von-Neumann-Architektur setzt sich aus dem CPU, Arbeitsspeicher, I/O Unit und Bussystem zusammen. Dabei hat das CPU ein Rechnerwerk (ALU) und ein Steuerwerk (Control Unit).

4. Der Datenprozessor führt Befehle aus und speichert die Ergebnisse in Registern die dann über das MBR im Hauptspeicher in bestimmte Adresse gespeichert werden, damit der Befehlsprozessor drauf zugreifen kann, wenn nötig. Der Datenprozessor besteht aus den Komponenten Register, ALU und MBR. Die Register im Prozessor sind zuständig für die Zwischenspeicherung von Ergebnissen nachdem die Befehle im ALU ausgeführt worden sind, diese Zwischengespeicherten. Der ALU ist verantwortlich für die Ausführung(Berechnung) der Befehle, wobei die Befehle erst vom MBR im Hauptspeicher ausgelesen werden und die Daten zum ALU mit dem Datenbus geschickt werden. Der MBR ist verantwortlich für die Kommunikation mit dem Hauptspeicher. Der MBR liest die Daten von der Adresse im Hauptspeicher, wobei die Adresse vom MAR aus dem Befehlsprozessor übermittelt wurde. Das passiert, damit der MBR weiß aus welcher Adresse er die Daten an das ALU übermitteln kann, zur Ausführung(Berechnung).

5. Vermeidung von Datenstau auf Bus -> wenn Bus mit Adressdaten voll ist können keine Daten mehr transportiert werden -> dead end.

6. Die Von-Neumann-Architektur entspricht SISD, *single instruction single data*, da es eine sequenziell arbeitende Architektur ist, wobei immer eine Operation auf ein Datenfeld

ausgeführt wird z.B. x86 Assembler mul 37, dabei wird eine Operation *Multiplikation* mit 37 auf Datenfeld rax ausgeführt. Neben SISD gibt es auch die *multiple* Varianzen davon, also *single instruction multiple data, SIMD*,*MISD* und *MIMD*, welche wie mit einer *Instruction* eine Operation auf mehrere Datenregister ausführt, z.B. Vektoreinheiten bei x86 AVX Einheiten

7. Maschinen Code ist eine Abfolge von binär Zeichen, welche vom Compiler aus z.B. Asssembler generiert werden. Machine Code ist nicht Menschen lesbar, kann jedoch ohne Übersetzung vom Prozessor verstanden und ausgeführt werden, wohingegen Assembler menschenlesbat ist. In Asssemlber entspricht jeder Befehl eine Hardware x86 Instruction, kann aber nicht ohne Übersetzung in Machine Code vom Prozessor verstanden werden. Ein-Adress Operationen beinhalten die durchzuführende Operation z.B. inc (increment) und das Register auf die die Operation ausgeführt werden soll, Multi-Adress Operationen wie z.B. mov beinhalten zwei Adressen, im mov Fall von welchem Register in welches transferiert werden soll.

8. Interpretationsphase: Programmcounter zeigt auf Speicherzelle -> der Inhalt davon wird als Anweisung interpretiert. Ausführungsphase: Anweisung enthält Adresse der Speicherzelle für die Daten -> Verarbeitung. D.h. Anweisung und Daten werden NACHEINANDER ausgeführt, verhindert dead end.

9. Bottleneck ist der Datenfluss zwischen CPU und RAM (getrennt, aber hohe Kommunikationsdichte). Umgehen durch Harvard Architektur -> aufteilen von Daten- und Programmspeicher

Gausssume

Schreiben Sie eine NASM-Funktion, welche die geschlossene Form der Gausssumme imple- mentiert. Machen Sie sich dazu mit Arithmetikbefehlen (ADD, SUB, MUL, DIV, IDIV, IMUL, NEG) in NASM vertraut.

Programm Code wie folgt:

```
1  ;Bruno Stendal, Martin Baer, Lukas Gewinner, Christian Schaefer
2              global gauss
3
4  gauss:      mov rax , rdi        ; rax := rdi
5              mul rax
6              add rax , rdi         ; rdx : rax := rsi * rax
7
8              mov rdx , 0
9              mov rcx , 2          ;rcx := 2
10             div rcx              ;rax := rax /2 , rdx := rax % 2
11             mov rax, 0
12             ret
```