Constantin Lazari, Marco Wettstein 23. September 2013

1. Beispiele für klassische Prozessoren sind: Intel 4004, Intel 8008, Intel 8088, Intel 8086, Intel 80286, Intel 80386, Motorola 68000, Z80, MOS 6502, PowerPC 970, PDP-11, CDP1802

   (a) Geben Sie das Erscheinungsjahr sowie die intern verwendeten Wortbreite an. Wie viele verschiedene Befehle können damit dargestellt werden? Nennen Sie pro Prozessor ein Computer-Modell bzw. Einsatzgebiet.

**Lösung:**

| Prozessor | Jahr | Wort | Befehle | Einsatzgebiet |
|---|---|---|---|---|
| Intel 4004 | 1971 | 4 Bit | 16 | Rechenmaschinen |
| Intel 8008 | 1972 | 8 Bit | 256 | Terminals |
| Intel 8086 | 1978 | 16 Bit | 65 536 | IBM-PC |
| Intel 8088 | 1979 | 16 Bit | 65 536 | IBM-PC |
| Intel 80286 | 1982 | 16 Bit | 65 536 | PCs |
| Intel 80386 | 1985 | 32 Bit | 4 294 967 296 | PCs |
| Motorola 68000 | 1979 | 32 Bit | 4 294 967 296 | Apple Macintosh |
| Z-80 | 1976 | 8 Bit | 256 | Arcade Spiele |
| MOS 6502 | 1975 | 8 Bit | 256 | HP-Rechner |
| PowerPC 970 | 2002 | 64 Bit | $\approx 1.8 \times 10^{19}$ | Macintosh G5 |
| PDP-11 | 1970 | 16 Bit | 65 536 | Regelungstechnik |
| CDP1802 (RCA 1802) | 1976 | 8 Bit | 256 | Raumfahrt |

Anmerkung zu Intel 4004: Die Befehlsbreite ist 8-Bit, es stehen 46 Befehle zur Verfügung.

2. Der „Pufferüberlauf“ gehört zu den häufigsten Sicherheitslücken in Programmen (mit Computern mit der Von-Neumann-Architektur).

   (a) Beschreiben Sie kurz informell, warum die klassische Harvard Architektur besser gegen diesen schützt (gegenüber der VonNeumann-Architektur).

**Lösung:**

Bei der klassischen Harvard Architektur sind Programmcode (Programmdaten) und Nicht-Programmdaten in physisch getrennten Speichern abgelegt. Bei der Von-Neumann-Architektur sind Code und Daten aber im gleichen Speicher abgelegt. Werden dabei Daten von externen Quellen (Datenträger, Tastatur, oder ähnliches) in den Arbeitspeicher geladen, muss das Programm einen bestimmten Speicherbereich (Puffer) für diese Daten reservieren. Sind die geladenen Daten grösser als dieser Bereich, ist es möglich, dass das Programm Teile des Programmcodes im Speicher überschreibt. So ist es möglich, über eingelesene Daten, den Ablauf des Programmes zu manipulieren. Bei der Harvard-Architektur ist dies nicht möglich, weil eingelesene Daten in einen anderen Speicher geladen werden, als der Programmcode.

(b) Ist Ihre Argumentation auch bei der Super-Harvard-Architektur allgemein korrekt?

**Lösung:**
Nein, denn bei der Super-Harvard-Architektur sind Programmdaten und Nicht-Programmdaten wieder im selben Speicher untergebracht. Es existiert lediglich eine Trennung von Program-Bus (für Programmdaten) und Daten-Bus (für Nicht-Programmdaten).

3. Wortbreiten

(a) Kann ein Prozessor mit geringer Wortbreite auch Werte (bzw. Worte) berechnen, die breiter sind? Zum Beispiel ein Prozessor mit 8-Bit Wortbreite auch 16- oder 32-Bit-Wörter. Falls ja, wie könnte ein solches Verfahren aussehen?

**Lösung:**
Offensichtlich ist dies möglich, sonst könnte nicht mit sehr sehr grossen Werten gerechnet werden können.
Mögliches Verfahren: Man hängt einfach mehrere Speicherregister hintereinander. Beispielsweise besteht dann ein 16-Bit Wort in Wirklichkeit aus 2 8-Bit-Wörtern. Wenn das Programm weiss, womit es es jeweils zu tun hat und ein entsprechendes Handling implementiert ist, besteht theoretisch kein Problem.

4. Architektur

(a) Wieso können der Motorola 68000 und die Intel-Prozessoren 8088, 8086 und 80286 mehr als 65 KB Hauptspeicher adressieren?

**Lösung:**
Der Speicher wird in jeweils 64kb grosse Segmente unterteilt, welche intern mittels 16bit grossen Adressen addressiert werden kann. Einer 16bit Adresse wird dabei eine wiederum 16bit grosse Segmentadresse vorangestellt. Die Segmentadresse im Segmentregister bestimmt dabei das zu wählende Speichersegment, die Adresse im Adressregister die konkrete Speicherzelle.

(b) Was unterscheidet den Motorola 68000 von der Architektur des Intel x86? Welche Vor- und Nachteile ergeben sich daraus?

**Lösung:**
Der Motorola 68k enthält einen grundsätzlich anderen Befehlssatz als die Intel x86er Prozessoren. Operationen und Operanden können frei kombiniert werden. Es gibt also keine eigenen Befehle für „diese Operation“ mit „jenem Operand“. Im Ergebnis ist der Motorola 68k einfacher zu programmieren.