4.2 PC-Hardware/Aufbau

4.2.1 Struktur der Zentraleinheit

Die Grundbestandteile der Zentraleinheit eines IT-System sind:

a) CPU

b) ROM

c) RAM

d) E/A-Bausteine

Auf welche Weise werden sie nun zusammengeschaltet? Prinzipiell sind mehrere Strukturen (Topologien) möglich (Abbildung 1). Aus Kostengründen wird fast überall die *Bus-Topologie* bevorzugt.

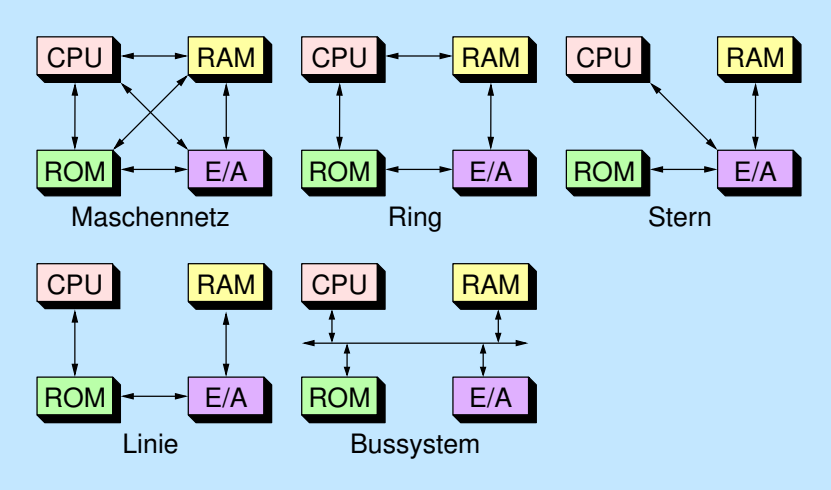

Abbildung 1: Mögliche Topologien von IT-Systemen

Das heißt, es gibt zwischen allen Baugruppen nur einen Informationskanal, den *Systembus* (auch *Bussystem* genannt). Der Aufbau eines PC stellt sich dann wie in Abbildung 2 dar. Die Pfeile stellen (vereinfacht) das Bussystem und damit den Informationsfluss in der Zentraleinheit dar. Elektrisch gesehen besteht ein Bussystem aus einer genau festgelegten Anzahl paralleler Leitungen (Abbildung 3). Sie werden durchnumeriert, beginnend mit Leitung null (die Informatiker beginnen ihre Aufzählungen gerne mit null).

Allerdings sind nicht alle Busleitungen gleich. Man kann jedes Bussystem in drei Teile aufteilen:

a) Steuerbus (hier blau)

b) Datenbus (hier rot)

c) Adressbus (hier gelb)

Genauer sieht man das in Abbildung 4. Natürlich kann man auch den elektrischen Schaltplan so aufteilen und erhält drei Bündel paralleler Busleitungen, die dann sinnvollerweise getrennt numeriert werden (Abbildung 5).

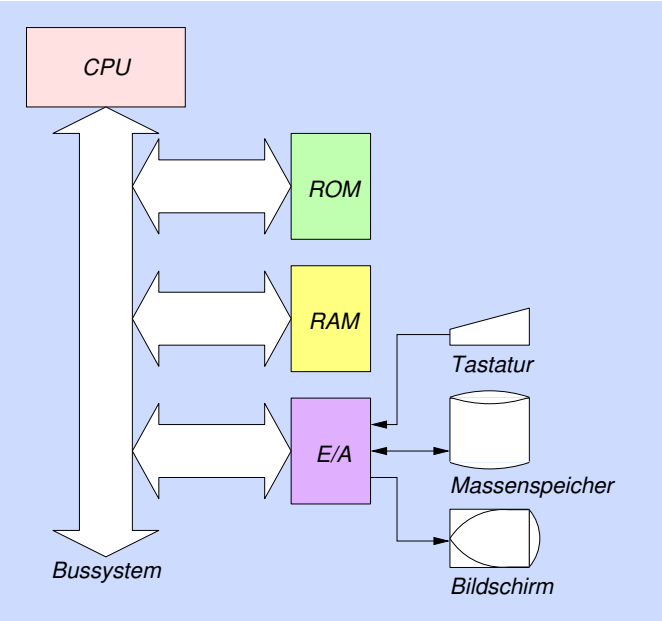

Abbildung 2: PC mit Bussystem

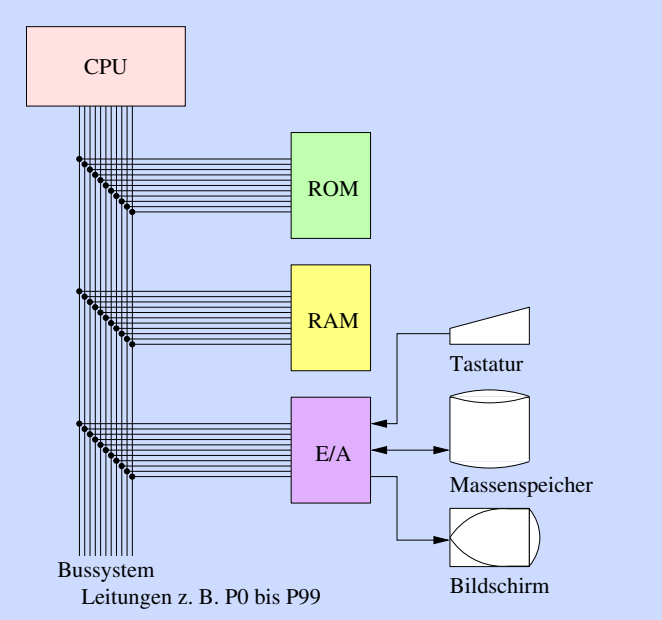

Abbildung 3: Stromlaufplan des Bussystems

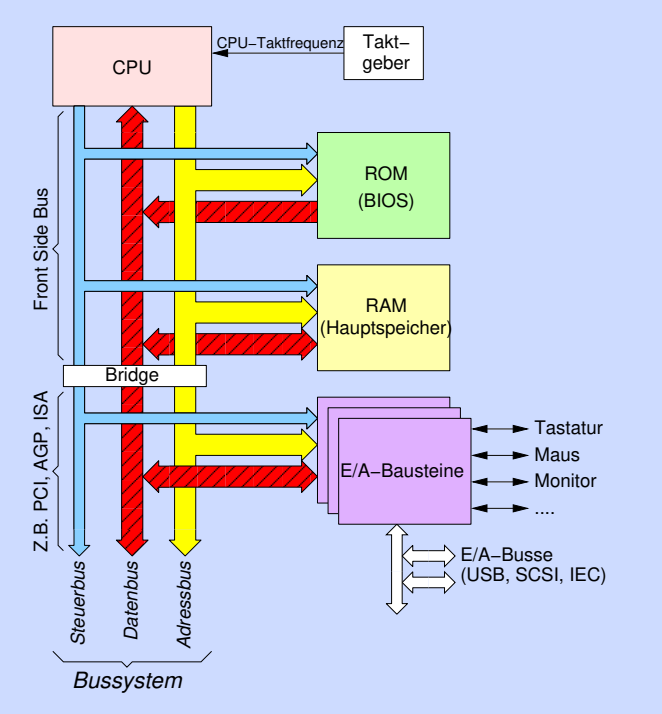

Abbildung 4: PC mit drei Bussen

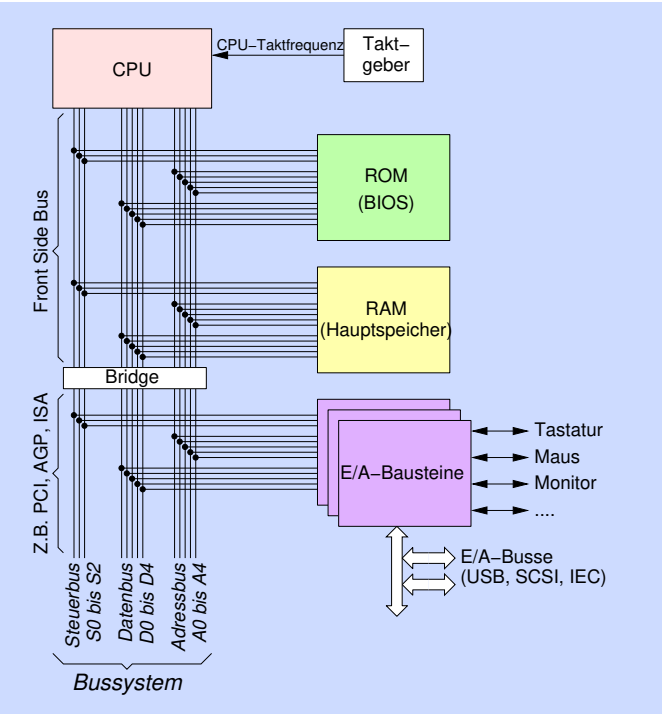

Abbildung 5: Stromlaufplan der drei Busse

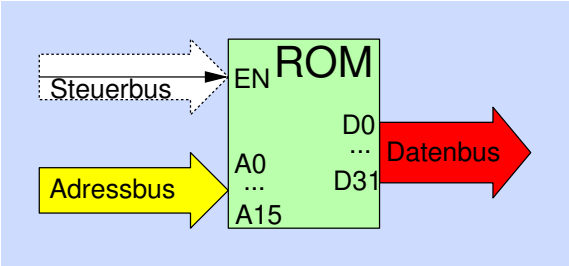

Abbildung 6: Informationsfluss beim ROM

4.2.2 ROM = *read only memory*

Wie sieht nun der Datenverkehr auf dem Bussystem aus? Dazu konzentrieren wir uns zuerst auf das ROM. Denn: Am einfachsten ist der Informationsfluss rund um das ROM zu verstehen. Das ROM (*read only memory*) ist ein Nur-Lese-Speicher (für direkten Zugriff). Abbildung 6 zeigt den entsprechenden Ausschnitt aus der Zentraleinheit. Die Funktion kann man vergleichen mit einem Automaten, der dem Benutzer den Kontostand anzeigt:

- Adresse — Konto-Nummer
- Datenwort — Konto-Stand
- EN (= *enable*) — Startknopf

Der Zugriff auf das ROM erfolgt in drei Schritten:

a) Zuerst wird die Adresse (=der Ort der gesuchten Information) an den Adressbus angelegt

b) Dann wird die Steuerleitung EN (= *enable*) auf den Wert eins gelegt

c) Nach einer *sehr* kurzen Zeit gibt das ROM die gesuchte Information, das sogenannte Datenwort, auf den Datenbus.

4.2.3 RAM = *random access memory*

Der Informationsfluss am RAM ist dem am ROM sehr ähnlich. Das RAM (*random access memory*) ist ein Schreib- und Lesespeicher. *Random access memory* bedeutet „Speicher für direkten (=wahlfreien) Zugriff“. [1] [2] Abbildung 7 zeigt den Ausschnitt aus der Zentraleinheit. Hier ist eine zusätzliche Steuerleitung R/W vorhanden, die die Datenflussrichtung auf dem Datenbus vorgibt.

- R/W = *read / write* — Liegt an dieser Leitung eine Eins, wird aus dem RAM gelesen. Bei einer Null wird in das RAM geschrieben.
- EN = *enable* — Startknopf

Der Lesezugriff auf das RAM erfolgt fast wie beim ROM:

a) Zuerst wird die Adresse (=der Ort der gesuchten Information) an den Adressbus angelegt

[1] RAM und ROM bieten diesen wahlfreien Zugriff; Festplatten-, DVD- und Bandlaufwerke bieten dagegen nur sequentiellen Zugriff.

[2] Bei Speichern mit wahlfreiem Zugriff ist die Zugriffszeit *unabhängig* von der Adresse des aktuellen (und des vorigen) Zugriffs. Bei Speichern mit sequentiellem Zugriff dagegen ist die Zugriffszeit nur dann sehr niedrig, wenn die aktuelle Adresse dicht hinter der vorigen Adresse liegt, ansonsten kann diese Zeit sehr viel höher sein. Kurz gefasst: Bei ungeordneten Zugriffen ist die durchschnittliche Zugriffszeit von Speichern mit sequentiellem Zugriff rund eine Million mal höher als die von Speichern mit wahlfreiem Zugriff. Nur, wenn sehr große Dateien (Videos, Backup, Betriebssysteme) auf einmal nacheinander gelesen oder geschrieben werden, können Speicher mit sequentiellem Zugriff mithalten.

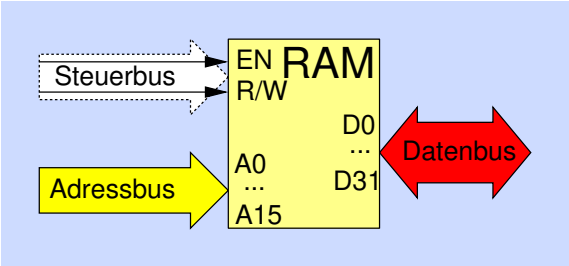

Abbildung 7: Informationsfluss beim RAM

b) Die Steuerleitung R/W wird auf den Wert eins gelegt, so dass aus dem RAM gelesen werden kaann

c) Es wird die Steuerleitung EN (= *enable*) auf den Wert eins gelegt

d) Nach einer sehr kurzen Zeit gibt das RAM die gesuchte Information (=das Datenwort) auf den Datenbus.

Der Schreibzugriff auf das RAM ist sehr ähnlich:

a) Zuerst wird die Adresse (=der Ort, an den Information geschrieben werden soll) an den Adressbus angelegt

b) Das zu schreibende Datenwort (=die Information) wird an den Datenbus angelegt

c) Die Steuerleitung R/W wird auf den Wert null gelegt, so dass in das RAM geschrieben werden kaann

d) Es wird die Steuerleitung EN (= *enable*) auf den Wert eins gelegt

e) Nach einer sehr kurzen Zeit liest das RAM die vom Datenbus erhaltene Information (=das Datenwort) an den gewünschten (=vom Adressbus angegebenen) internen Ort

4.2.4 Bussystem (Aufbau)

Wie bereits beschrieben besteht das Bussystem aus drei Teilen, nämlich dem Steuerbus, dem Datenbus und dem Adressbus. Einer dieser Busse alleine reicht nicht aus: Man braucht stets alle drei Busse. In der Praxis spricht man aus Bequemlichkeit oft von einem Bus, meint aber ein Bussystem. Die drei Busse unterscheiden sich leicht voneinander:

- Adressbus: Unidirektional — eine Einbahnstraße für Adressen; die Information fließt von der CPU hin zu RAM, ROM und Peripherie
- Datenbus: Bidirektional — ebenfalls eine Einbahnstraße; allerdings kann mit Hilfe einer Steuerleitung (R/W) die Fahrrichtung jederzeit neu festgelegt werden.
- Steuerbus: Einzelleitungen — die meisten Steuerleitungen geben Signale von der CPU zu RAM, ROM und Peripherie; Ausnahme sind Interrupt-Anforderungs-Leitungen (*interrupt request*, IRQ), die Signale von Ein- oder Ausgabebausteinen zur CPU führen.

  Das heißt: Die CPU gibt an, ob sie Daten lesen oder schreiben möchte

Die Anzahl der Leitungen (Breite) der einzelnen Busse hat je nach Bus einen ganz unterschiedlichen Effekt:

- Adressbus-Breite: gibt die maximal mögliche Anzahl von Datenworten an, hat damit Einfluss auf die maximale Systemgröße

- Datenbus-Breite: gibt die Größe eines Datenwortes an, hat damit Einfluss auf die Systemgeschwindigkeit
- Steuerbus-Breite: hängt ab vom verwendeten Prozessor, hat im allgemeinen keinen Einfluss auf die Systemgröße oder Systemgeschwindigkeit

4.2.5 Bussystem (Ablauf eines Buszugriffs)

Mit Hilfe von Abbildung 4 kann man sich jetzt überlegen, wie zum Beispiel ein Lesezugriff auf das RAM aussieht:

a) Die CPU legt eine Adresse (=den Ort) an den Adressbus

b) Die CPU legt die Steuerleitung R/W eins (=Lesen)

c) Die CPU legt die Steuerleitung EN (= *enable*) für das RAM auf eins (es gibt also eine EN-Leitung nur für das RAM, ebenso eine andere EN-Leitung nur für E/A!)

d) Nach einer Zeit gibt das RAM das Datenwort auf den Datenbus und die CPU liest es ein

Und noch einmal dasselbe für den Schreibzugriff auf das RAM:

a) Die CPU legt eine Adresse an den Adressbus

b) Die CPU legt das zu schreibende Datenwort an den Datenbus

c) Die CPU legt die Steuerleitung R/W auf null (=Schreiben)

d) Die CPU legt die Steuerleitung EN (= *enable*) für das RAM auf eins

e) Nach einer Zeit liest das RAM das Datenwort vom Datenbus an den gewünschten internen Ort

4.2.6 CPU (Aufbau)

Man sieht, dass ziemlich viel Aktion von der CPU ausgeht. Aber was macht sie genau, und wie macht sie das? Intern besteht die CPU aus vier großen Blöcken (Abbildung 8):

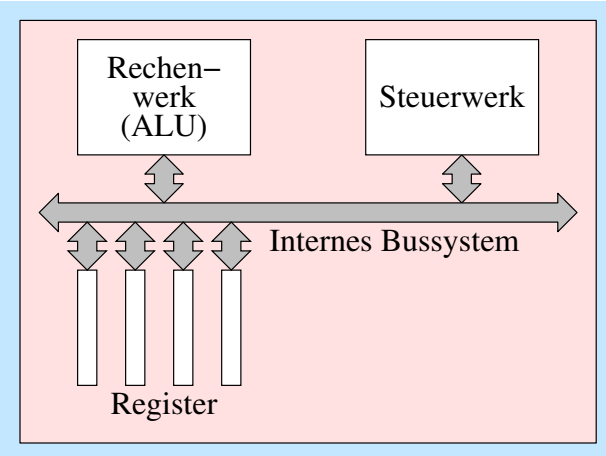

Abbildung 8: Aufbau einer CPU (grob)

- Steuerwerk – ein wichtiger Teil der CPU
- Rechenwerk (ALU) – braucht man für die Rechenfähigkeit

- Internes Bussystem – auch innerhalb der CPU fließen Informationen
- Minimales Speicherwerk (Register) – nur wenige Bytes als winziger Zwischenspeicher

Das hilft noch nicht viel weiter.

4.2.7 CPU (Ablauf eines Befehls I)

Schaut man sich ein Handbuch einer CPU an, so findet man oft – neben sehr groben Schaubildern – eine Liste von *Befehlen*, die die CPU beherrscht. Einen Befehl kann man vergleichen mit einem Kunststück, das ein Haustier (Hamster, Hund, Katze) oder Zirkustier beherrscht: Sobald der Besitzer eine bestimmte Handbewegung macht, beginnt das Haustier mit dem Kunststück. Wenn es fertig ist, wartet es (im Idealfall) auf die nächste besondere Handbewegung. Genauso ist es mit der CPU: Sie erwartet von einer bestimmten Stelle eine Befehlsnummer (dargestellt durch ein Muster von Nullen und Einsen) und arbeitet dann den Befehl ab.

Nacheinander passieren immer wieder die folgenden zwei Schritte:

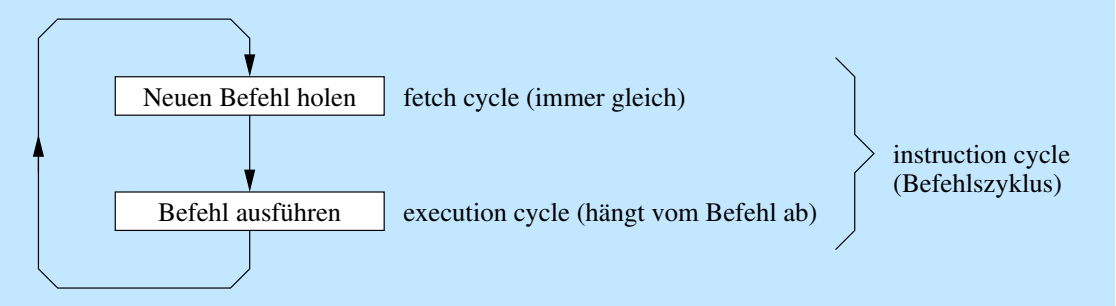

Abbildung 9: Befehlszyklus

a) Befehl holen: *fetch cycle* (immer gleich)

b) Befehl ausführen: *execution cycle* (natürlich abhängig vom auszuführenden Befehl)

Beide zusammen nennt man den Befehlszyklus. Entscheidend für die Systemgeschwindigkeit ist die Dauer eines Befehlszyklus bei den wichtigsten Befehlen.

4.2.8 CPU (Ablauf eines Befehls II)

Nur woher bekommt die CPU ihren Befehl? Dazu existieren zwei verschiedene Möglichkeiten, die Harvard- und die Von-Neumann-Struktur. Sie unterscheiden sich durch die Anzahl der Bussysteme:

- Harvard-Struktur (Abbildung 10)
  - Zwei Bussysteme, daher höherer Aufwand.
  - Wird bei sehr schnellen Signalprozessoren benutzt.

- Von-Neumann-Struktur (Abbildung 11)
  - Nur ein einziges Bussystem, daher geringer Aufwand.
  - Wird derzeit in PCs benutzt.

Bei der Harvard-Struktur erhält die CPU ihre Befehle von einem speziellen Bussystem, das nur für Befehle zuständig ist (in der Abbildung 10 oben dargestellt). Das andere Bussystem ist für die Abarbeitung der Befehle zuständig. Bei der Von-Neumann-Struktur dagegen erhält die CPU ihre Befehle über das gleiche Bussystem, über das sie ihre Befehle auch verarbeitet.

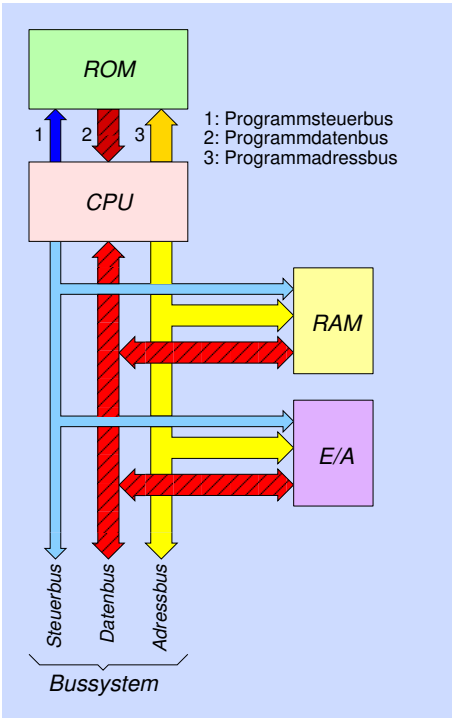

Abbildung 10: CPU und Zentraleinheit mit Harvard-Struktur

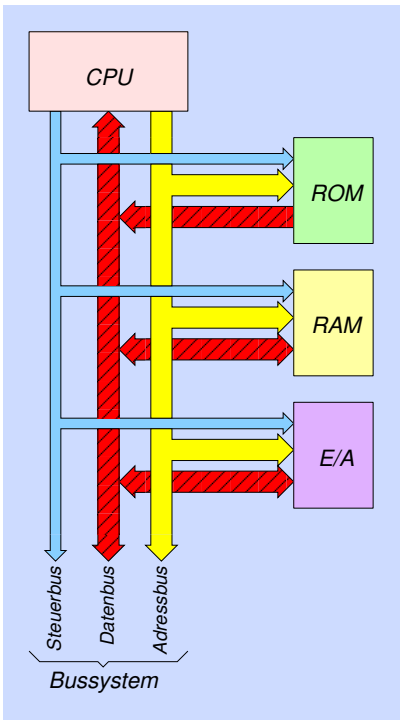

Abbildung 11: CPU und Zentraleinheit mit Von-Neumann-Struktur

4.2.9 Exkurs: Vergleich Rechnerstruktur – Bibliothek

Man kann die Arbeit einer CPU im Rechnersystem vergleichen mit der Arbeit eines Bibliothekars. Abbildung 12 zeigt das Schema an der Harvard-Struktur:

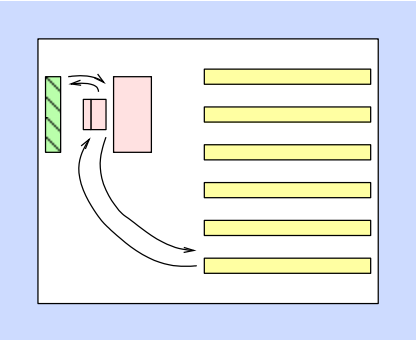

Abbildung 12: Harvard-Struktur als Bibliothek

- Die Regalreihen (rechts) entsprechen dem RAM
- Der Bibliothekar (Mitte) entspricht der CPU
- Das Handregal mit Aufträgen (links) entspricht dem ROM

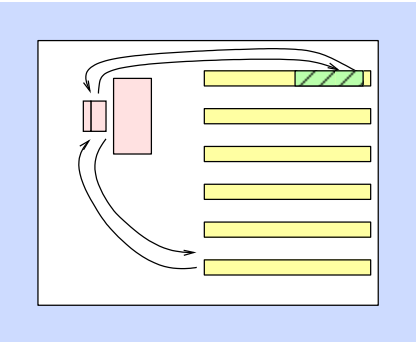

Abbildung 13: Von-Neumann-Struktur als Bibliothek

Abbildung 13 zeigt das gleiche Schema an der Von-Neumann-Struktur:

- Die Regalreihen (rechts) entsprechen dem RAM
- Der Bibliothekar (Mitte) entspricht der CPU
- Das Handregal mit Aufträgen (links) fehlt jetzt. Stattdessen wird dem Bibliothekar ein Teil eines Bücherregals zugewiesen, aus dem er jetzt seine Aufträge holen muss (rechts oben, schraffiert).

Bei der Harvard-Struktur liefen der *fetch cycle* auf dem Programm-Bussystem ab, der *execution cycle* dagegen auf dem zum RAM führenden Bussystem. Bei der Von-Neumann-Struktur dagegen laufen beide Teile des Befehlszyklus nacheinander auf dem einzigen Bussystem ab.

4.2.10 CPU (Ablauf eines Befehls III)

Abschließen soll ein Befehlszyklus dargestellt werden. Zur besseren Übersicht nutzen wir dazu die Harvard-Struktur. In unserem Fall soll der Befehl lauten: Hole den Inhalt des RAM-Datenwortes mit der Adresse 0111 in die CPU (genauer: in das Register A):
`http://www.gyrator.de/material/I42S/harvard_anim.gif`

a) Beginn des *fetch cycle*

b) Die CPU setzt die Adresse auf dem oberen Bussystem (also die Befehlsadresse) auf den Wert 0000 (direkt nach dem Einschalten beginnt man bei 0000)

c) Die CPU setzt die Steuerleitung EN (auf dem oberen Bussystem) auf eins

d) Das ROM antwortet auf dem oberen Datenbus mit dem Datenwort 10000111. Die CPU interpretiert dieses Datenwort als Befehl. Der Inhalt des Befehls bedeutet: Hole den Inhalt des RAM-Datenwortes mit der Adresse 0111 in die CPU

e) Ende des *fetch cycle*

f) Beginn des *execution cycle*

g) Die CPU legt die Adresse 0111 auf den unteren Adressbus

h) Die CPU setzt die Steuerleitung EN (auf dem unteren Bussystem) für das RAM auf eins

i) Das RAM antwortet auf dem unteren Datenbus mit dem Datenwort 01010101 (war offenbar der Inhalt des gesuchten Ortes)

j) Die CPU liest das Datenwort vom unteren Datenbus ein

k) Ende des *execution cycle*

l) Beginn des **nächsten** *fetch cycle*

m) Die CPU setzt die Befehlsadresse auf den Wert 0001 (eins höher als der letzte Befehl)

n) weiter mit c) – vermutlich liegt im ROM an der Befehls-Adresse 0001 ein anderer Befehl; auch der wird geholt und abgearbeitet ...

Man sieht, dass eine CPU mit jedem Befehl immer nur einen kleinen Schritt in einem Programm abarbeiten kann. Die hohe Rechengeschwindigkeit ergibt sich schließlich daraus, dass sehr viele Befehle pro Sekunde abgearbeitet werden.