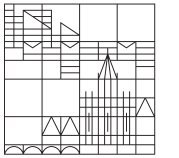
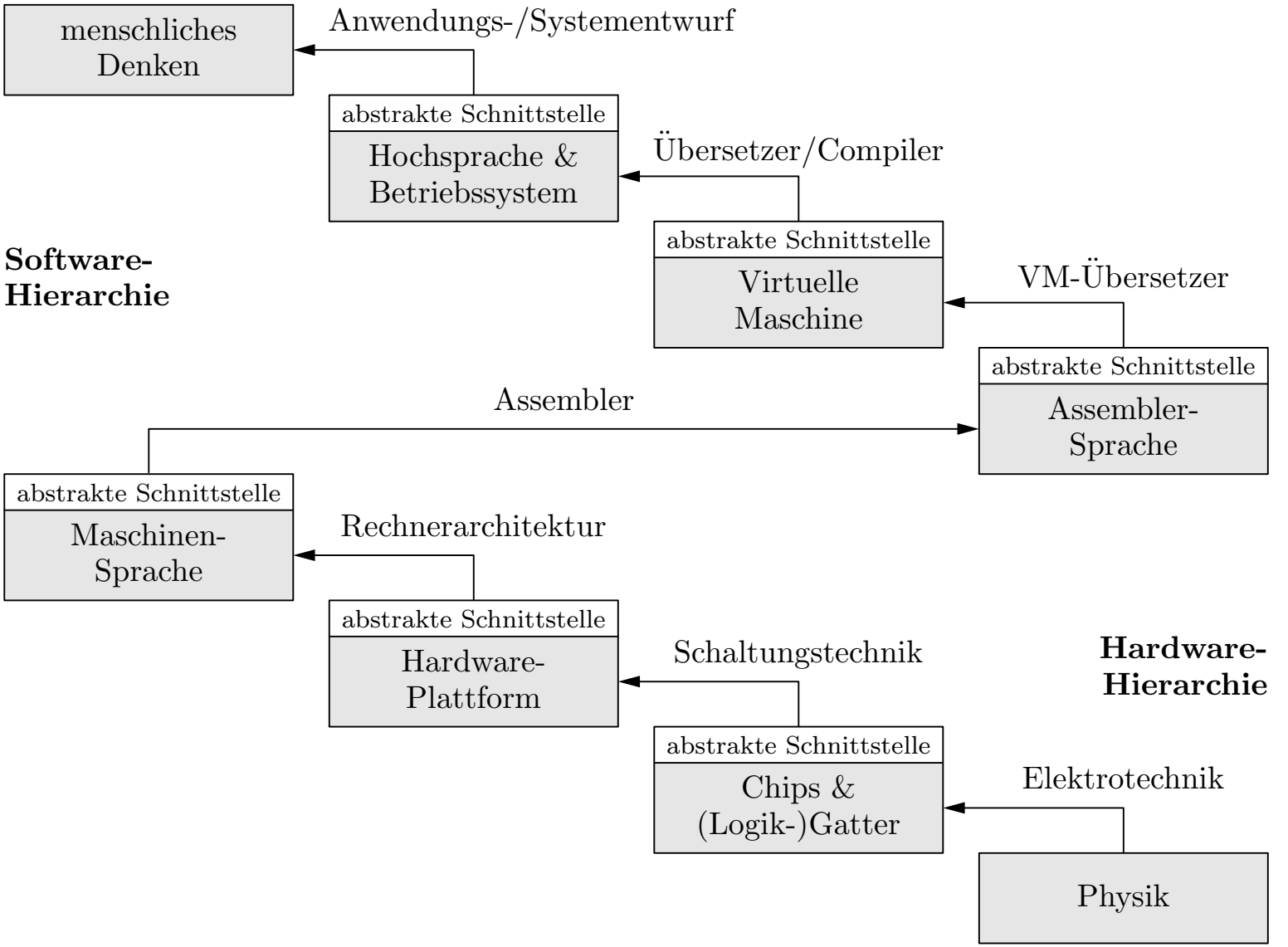
Universität Konstanz



Rechnersysteme und -netze
Kapitel 5 - Rechnerarchitektur

Bastian Goldlücke
Universität Konstanz
WS 2020/21

Rechnersysteme: Plan der Vorlesung



Erinnerung: Schaltungstechnik III

- **Arithmetik** (im wesentlichen Erinnerung an Bekanntes)
  - Zahlensysteme (Basis 10, andere Basen als 10)
  - Addition und Subtraktion, Übertrag
  - Multiplikation und Division, Stellenprodukte
  - Implementierung durch mechanische Rechner

- **Arithmetisch-logische Einheit** (Arithmetic Logic Unit, ALU)
  - Binärarithmetik (Rechnen im Zahlensystem mit Basis 2)
  - Halbaddierer und Volladdierer (1-Bit-Addierer)
  - *n*-Bit-Addierer (mit Übertragskette und Übertragsauswahl)
  - Multiplikation: Bit-Schieben, Standardalgorithmus
  - Multiplikation: negative Zahlen, Booths Algorithmus
  - Hack-Architektur (arithmetisch-logische Einheit, Prozessor)

Erinnerung: Schaltungstechnik IV

- **Sequentielle Logik**
  - Kombinatorische und sequentielle Logik
  - Schaltnetze und Schaltwerke
  - Rückkopplung (feedback) und Taktsignal (clock)
- **Bistabile Kippstufen („Flipflops“)**
  - SR-Riegel (SR latch) als Grundbaustein
  - D-, E-, T- und JK-Riegel (D, E, T, JK latch)
  - Taktpegel- und Taktflankensteuerung (latches vs. flip flops)
  - Master-Slave-Riegel und -Flipflops
- **Register, Zähler und Speicher**
  - Schieberegister, Parallel-Seriell-Wandler, Zähler
  - Speicherregister, Hauptspeicherorganisation

Inhalt

Inhalt

Festverdrahtete Prozessoren: Waschmaschinen

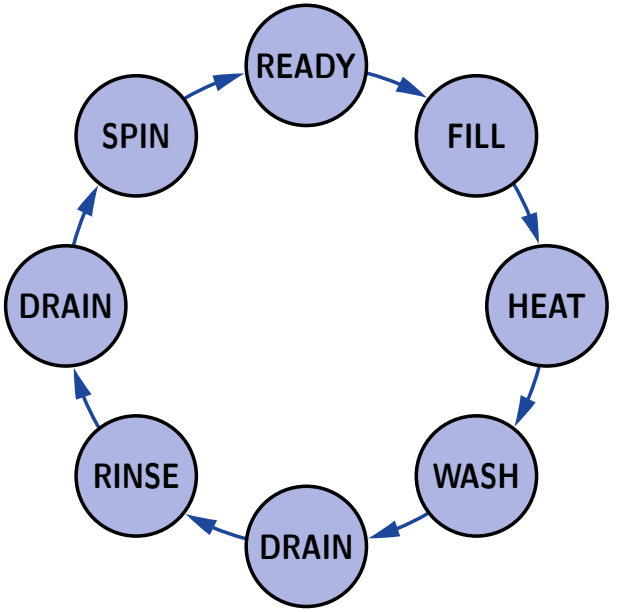
Das Verhalten eines Prozessors kann durch Schaltnetze simuliert werden:

- Im Gegensatz zu einem frei programmierbaren Prozessor ist in einem **festverdrahteten Prozessor** (z.B. einer Ablaufsteuerung) die Bewegung des Befehlszählers (program/instruction counter) festgelegt (durch Schaltkreise fest bestimmt).

- Dies ist nur dann akzeptabel, wenn der Prozessor nur eine einzelne, festgelegte Aufgabe zu erfüllen hat.
- Dies ist in einfachen Automaten der Fall, wie z.B. in Waschmaschinen.

  Diese Durchlaufen eine feste Abfolge von Zuständen, in denen festgelegte Aktionen ausgelöst werden.
- Verzweigungen sind prinzipiell möglich, aber unveränderbar (festes Programm).

Beispiel: Zustandsdiagramm einer Waschmaschine



Festverdrahtete Prozessoren: Waschmaschinen

- Der Waschmaschinen-„Prozessor“ führt wiederholt ein Programm der Länge 8 aus (Programmlänge: Anzahl Anweisungen)
- Der Prozessor kann die Waschmaschinen-„Einheiten“ folgendermaßen steuern:
  - Ventil offen / geschlossen (Wasserzufuhr)
  - Heizung an / aus
  - Motor aus / langsam / schnell
  - Pumpe an / aus (Wasserabfuhr)

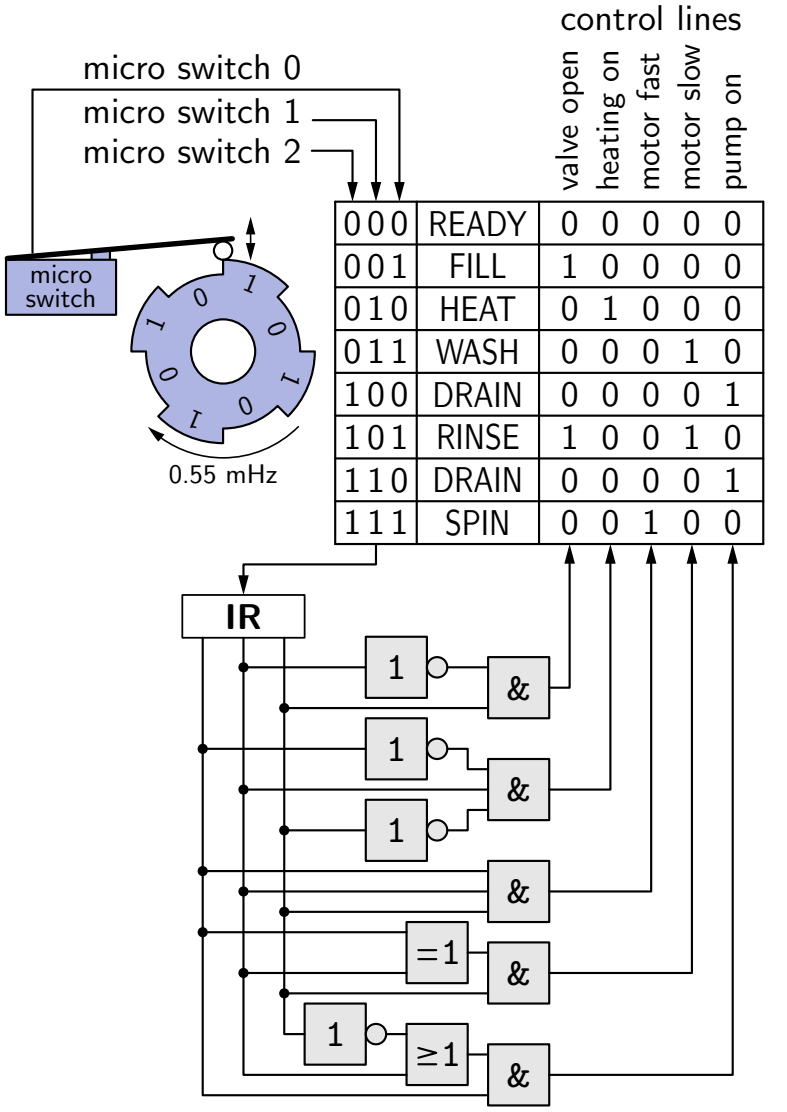
Anweisung	Wirkung			
READY	Ventil geschlossen	Heizung aus	Motor aus	Pumpe aus
FILL	Ventil offen	Heizung aus	Motor aus	Pumpe aus
HEAT	Ventil geschlossen	Heizung an	Motor aus	Pumpe aus
WASH	Ventil geschlossen	Heizung aus	Motor langsam	Pumpe aus
DRAIN	Ventil geschlossen	Heizung aus	Motor aus	Pumpe an
RINSE	Ventil offen	Heizung aus	Motor langsam	Pumpe aus
SPIN	Ventil geschlossen	Heizung aus	Motor schnell	Pumpe aus

Festverdrahtete Prozessoren: Waschmaschinen

Waschmaschinen-„Prozessor“

- Wir nehmen an, daß die Waschmaschine ihr Programm in 30 Minuten ausführt.
- Das Weiterschalten von Anweisung zu Anweisung wird durch Nockenscheiben bewirkt, die sich mit $\frac{1}{1800s} = 0.55\,\text{mHz}$ drehen, und dabei drei Mikroschalter betätigen.
- Ein Schaltnetz implementiert die Dekodierung der Anweisungen und ihre Ausführung.

Bemerkung: Waschmaschinen besitzen heutzutage natürlich eine sehr viel kompliziertere Steuerung als hier dargestellt. Erste Waschmaschinen funktionierten aber durchaus nach einem solchen Prinzip.



Konzept der Speicherprogrammierung

- Eine **Ablaufsteuerung** (wie für eine Waschmaschine) läuft schrittweise ab, wobei von einem Schritt auf den nächsten gemäß vorgegebener Übergangsbedingungen weitergeschaltet wird.

- Gegenüber Rechnern mit freier Programmierbarkeit sind Ablaufsteuerungen meist stark eingeschränkt, oft sogar festverdrahtet.

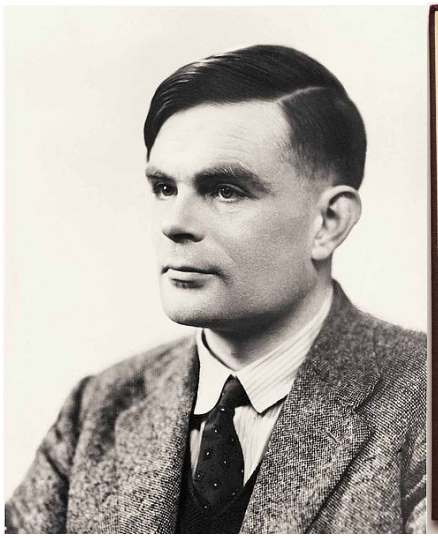
- Für eine freie Programmierbarkeit von Automaten oder Rechnern ist das **Konzept der Speicherprogrammierung** entscheidend:

  Die von einem Rechner auszuführenden Anweisungen werden nicht festverdrahtet, sondern als kodierte Befehle in einem Speicher abgelegt.

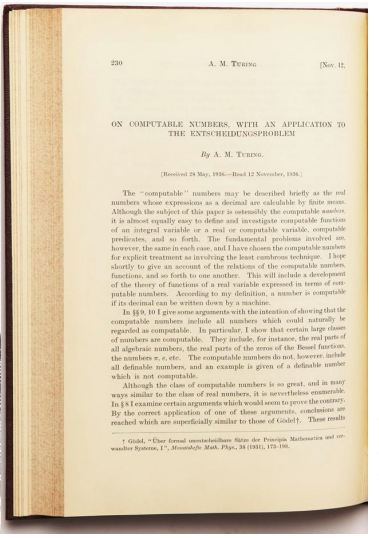
- Programme sind dadurch prinzipiell austauschbar, ein speicherprogrammierbarer Rechner folglich nicht auf eine bestimmte Aufgabe festgelegt.
  Dies ermöglicht **universelle Rechenmaschinen**.

- Die Speicherprogrammierung wurde entscheidend von [John von Neumann 1945] geprägt, der Ideen von [Alan Turing 1936] weiterentwickelte, der wiederum mathematische Ideen von [Kurt Gödel 1930] aufgegriffen hatte.

Konzept der Speicherprogrammierung



www.hnf.de



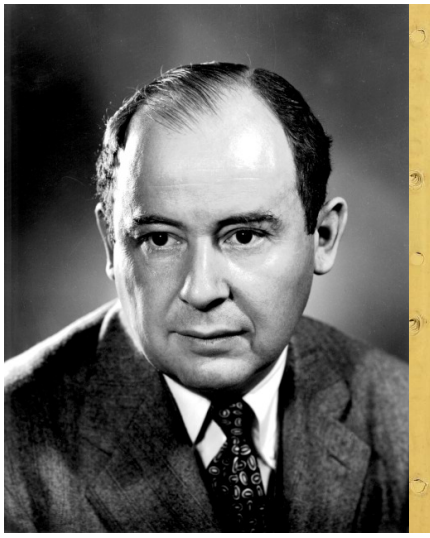
ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM

By A. M. Turing.

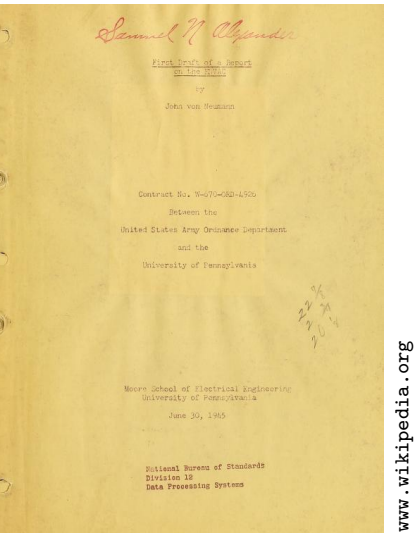
www.abebooks.com

Alan Turing [1912–1954]

“On Computable Numbers,
with an Application to
the Entscheidungsproblem”,
Proc. London Mathematical Society,
2 (1936, published 1937), 42(1)



www.wikipedia.org



First Draft of a Report on the EDVAC

by

John von Neumann

Contract No. W-670-ORD-4926

Between the

United States Army Ordnance Department

and the

University of Pennsylvania

Moore School of Electrical Engineering
University of Pennsylvania

June 30, 1945

National Bureau of Standards
Division 13
Data Processing Systems

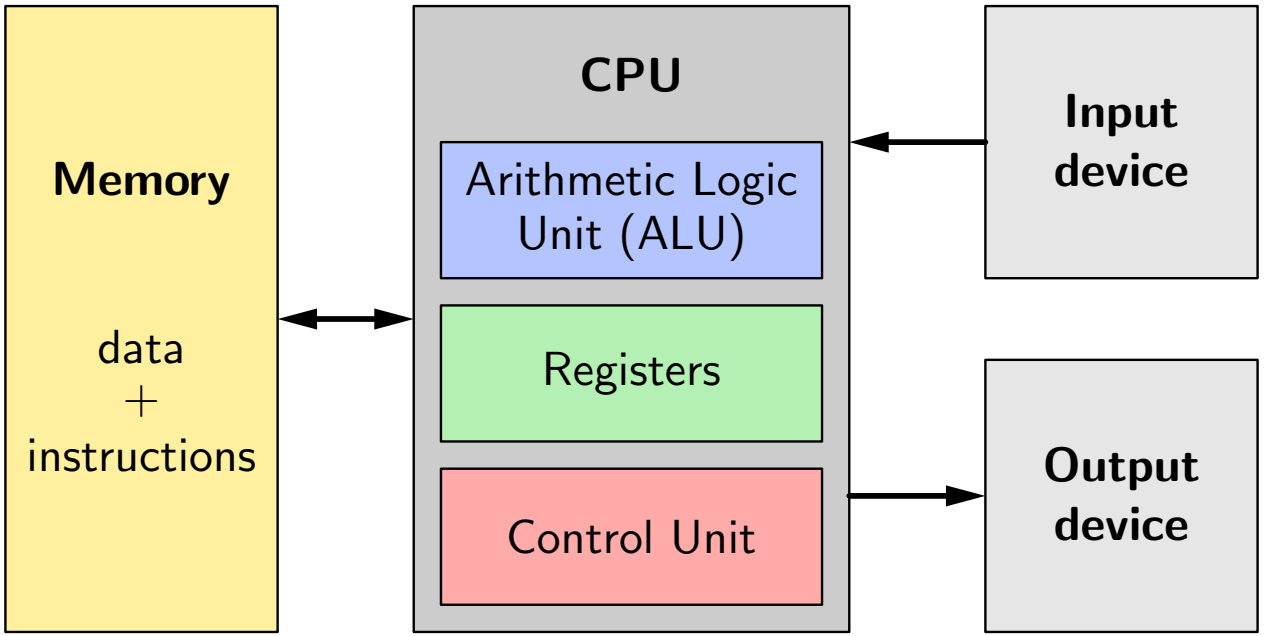
www.wikipedia.org

John von Neumann [1903–1957]

“First Draft of a Report on the EDVAC”
(EDVAC: Electronic Discrete
Variable Automatic Computer),
US Army Ordenance Department
& University of Pennsylvania 1945

- Diese beiden Aufsätze waren entscheidend für die Entwicklung des Konzeptes der Speicherprogrammierung.

Konzept der Speicherprogrammierung



Ausführen einer Anweisung erfordert einen oder mehrere der folgenden Teilschritte:

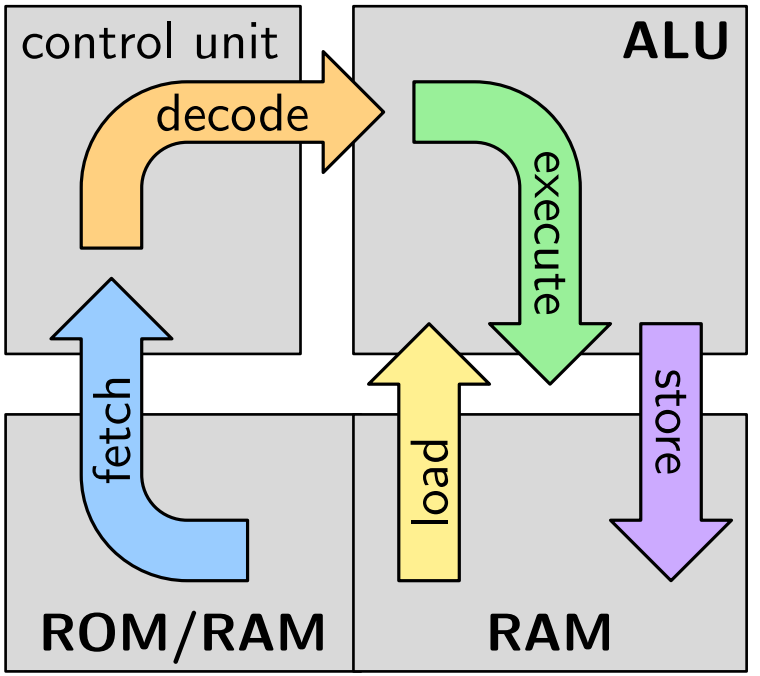
- Die arithmetisch-logische Einheit (ALU) berechnet eine Funktion $f(\text{registers})$.
- Die Ausgabe der arithmetisch-logischen Einheit wird in ein Register geschrieben.
- Weiter muß bestimmt werden, welches die nächste auszuführende Anweisung ist. (Bei einem Verzweigungsbefehl ist dies u.U. nicht die im Speicher folgende.)

Befehlsabruf, -dekodierung und -ausführung

Im Konzept der Speicherprogrammierung besteht das Ausführen eines Befehls aus den folgenden drei Schritten:

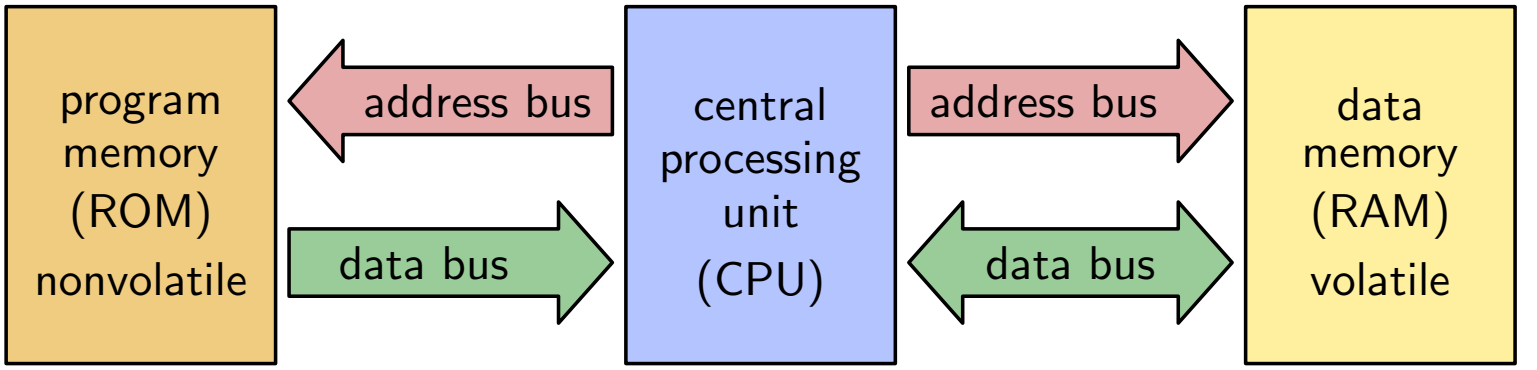
- Befehlsabruf (fetch)
- Befehlsdekodierung (decode)
- Befehlsausführung (execute)

Dies ist der **fetch-decode-execute cycle**, der zur Programmausführung immer wieder durchlaufen wird (für jeden Befehl).

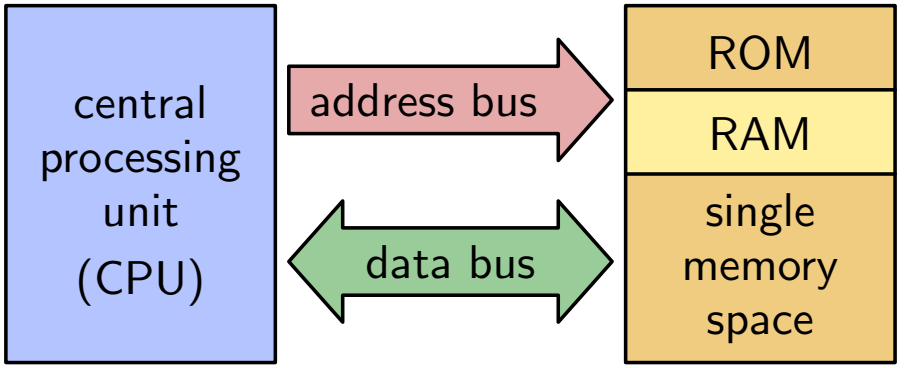


- Der Befehlsabruf (fetch) überträgt den nächsten auszuführenden Befehl in die Steuereinheit (control unit) des Prozessors, wo er dekodiert wird (decode).
- Die Steuereinheit weist dann die arithmetisch-logische Einheit (ALU) an, die im Programmbefehl kodierte Berechnung f auszuführen (execute).
- Die Berechnung kann das Laden (load) von Argumenten aus dem Datenspeicher, das Ablegen (store) von Berechnungsergebnissen in den Datenspeicher erfordern.

Rechnerarchitekturen



Harvard architecture



von Neumann architecture

- Bei der Harvard-Architektur liegen Programm und Daten in zwei verschiedenen Speichern, während sie bei der von-Neumann-Architektur in einem einzigen liegen.
- Meist wird die von-Neumann-Architektur verwendet [John von Neumann 1945], obwohl nur ein Bus für sowohl Daten als auch Programmbefehle Nachteile hat.

Inhalt

Der Hack-Rechner

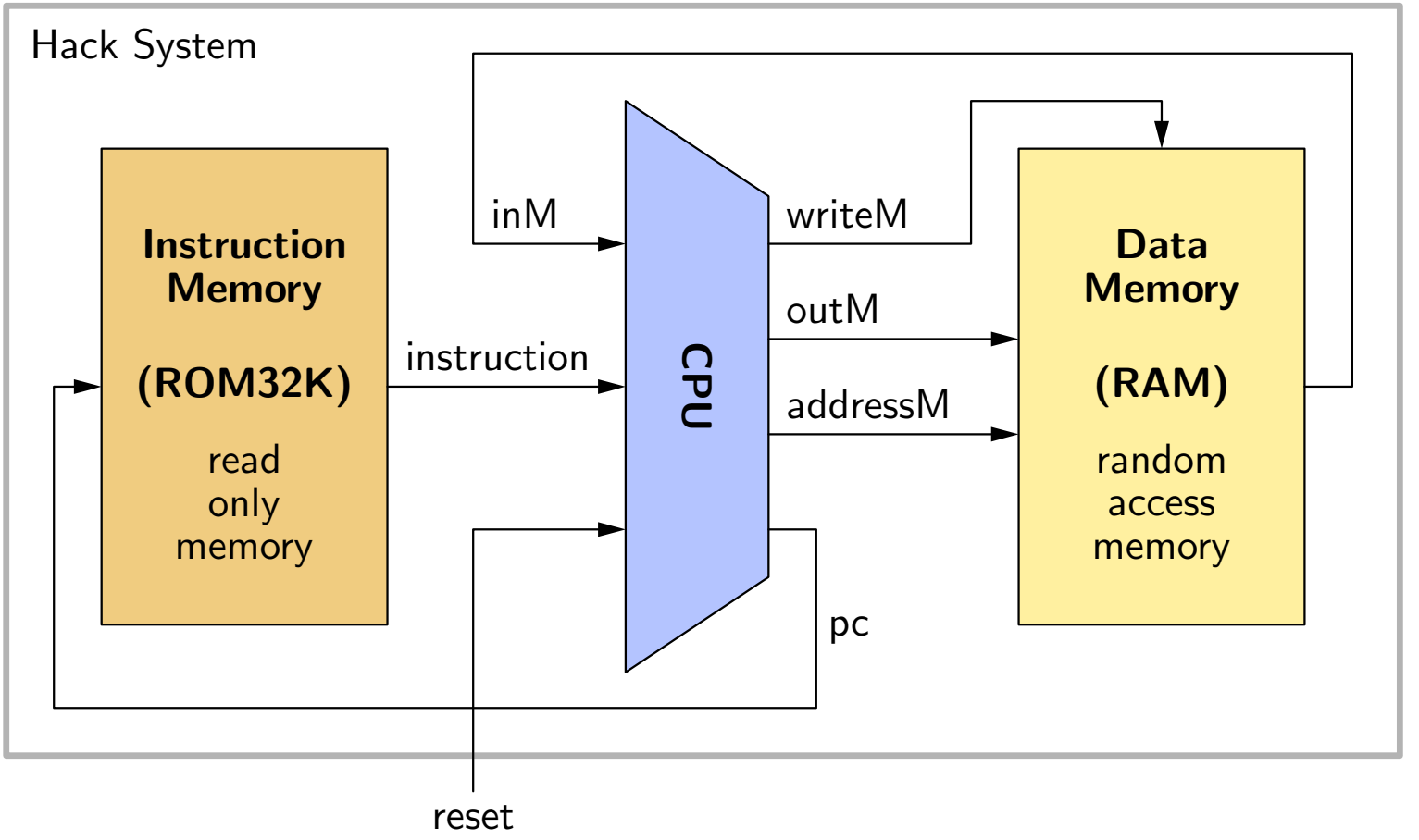
Rahmendaten des Hack-Rechners:

- 16-bit Harvard-Architektur (speicherprogrammierbar):
  Befehlsspeicher und Datenspeicher sind physisch getrennt.
- 512 $\times$ 256 Pixel Schwarz-Weiß-Bildschirm, Standardtastatur
- Kann Programme in der Hack-Maschinensprache ausführen
- Kann leicht aus den Chips aufgebaut werden, die wir bisher betrachtet haben.

Hauptbestandteile des Hack-Rechners:

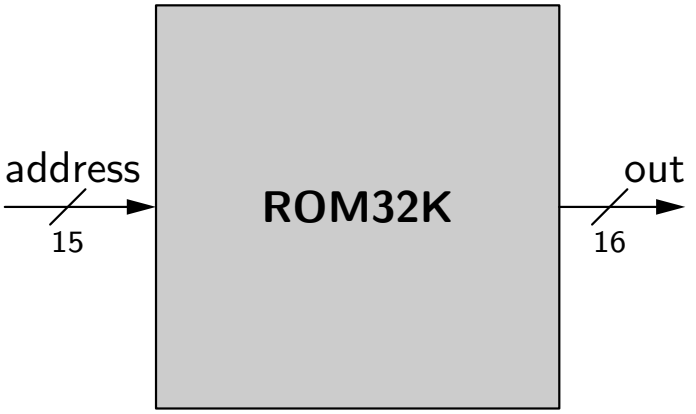
- Prozessor (central processing unit, CPU)
- Befehlsspeicher (32 kB read only memory, ROM)
- Datenspeicher (16 kB random access memory, RAM)
  zzgl. Bildschirm- und Tastaturspeicher
- „Computer“ (übergeordnete Einheit, die die Teile zusammenfügt)

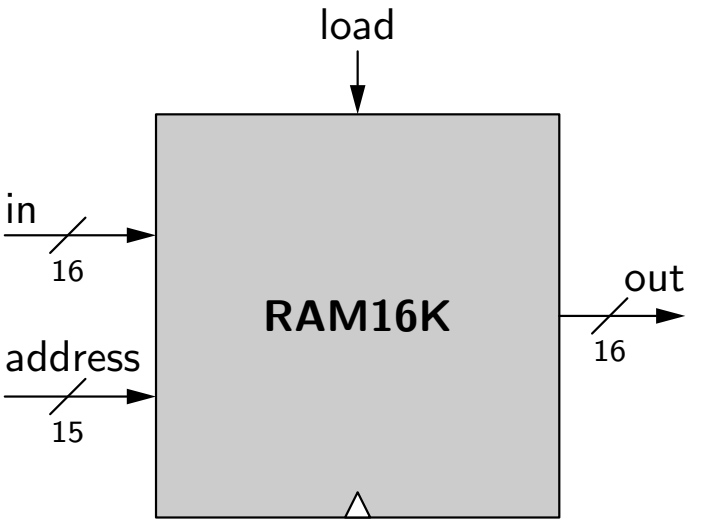
Hack-Architektur: Gesamtsystem



- Rechnerarchitektur aus [Noam Nisan & Shimon Schocken 2008] für die Hack-Plattform (pc: program counter, d.h. Befehlszähler).
- **Harvard-Architektur** (nicht von-Neumann-Architektur!)

Hack-Architektur: Befehls- und Datenspeicher





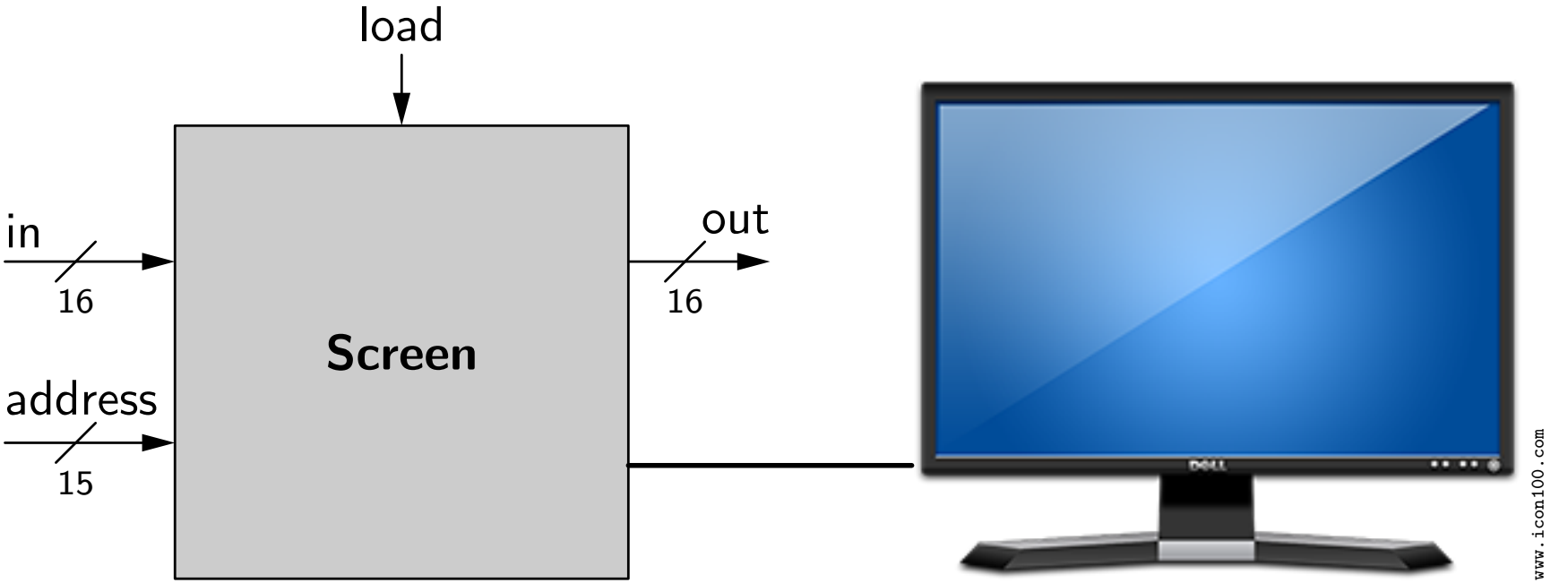
Befehlsspeicher (32k 16-Bit-Zahlen)

- Enthält ein Programm in der Hack-Maschinensprache.
- Liefert stets eine 16-Bit-Zahl: `out = ROM32K[address]`.
- Diese Zahl wird als die nächste Anweisung interpretiert (current instruction).

Datenspeicher (16k 16-Bit-Zahlen)

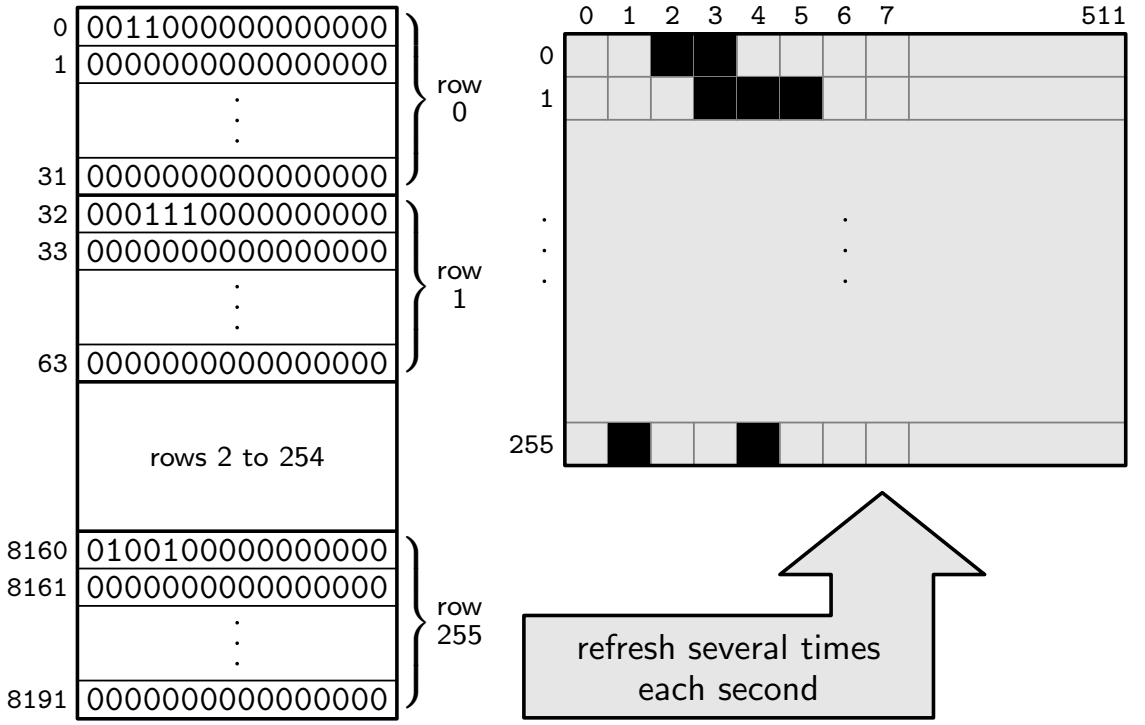
- Maschinenebene: Setze `in`, `address`, `load`, lese `out`.
- Hochsprache: Verwende `peek(address)` (lesen) und `poke(address)` (schreiben). (Betriebssystemfunktionen, später)

Hack-Architektur: Bildschirm



- Der Bildschirm-Chip verhält sich wie normaler 16-Bit-Datenspeicher (random access memory, RAM) mit 8K Speicherplätzen:
  - `out = Screen[address]`
  - `if load then Screen[address] = in`
- Nebeneffekt allerdings: Der Speicherinhalt wird als Bildpunktmatrix aufgefaßt und als Schwarzweißbild auf einem 512×256 Punkte großen Bildschirm angezeigt.

Hack-Architektur: Bildschirmspeicher



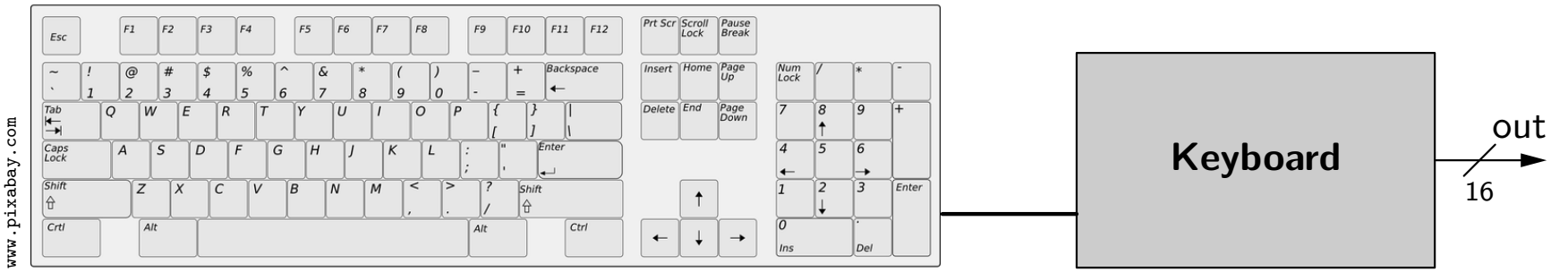
- Lineares Speicherabbild des Bildschirms.
- Jedes einzelne Bit entspricht einem Bildpunkt (Pixel).
- Bildschirmzeilen sind von links nach rechts und oben nach unten hintereinander abgelegt.

Setzen eines Punktes (Pixels) an der Stelle (x, y):

Maschinenebene: Setze Bit `x%16` (d.h., $x \bmod 16$) des Wortes `Screen[y*32+x/16]`.

Hochsprache: Verwende `drawPixel(x,y)`. (Betriebssystemfunktion, später)

Hack-Architektur: Tastatur



- Der Tastatur-Chip stellt ein einzelnes 16-Bit-Register zur Verfügung.

  (Eigentlich wäre ein 8-Bit-Register ausreichend (siehe nächste Folie),
  aber für die Einbindung in das Speicherbild des Hack-Rechners sind 16 Bit einfacher.)

- Eingang: 16-Bit-Zahl, die von einer physischen Tastatur geliefert wird.

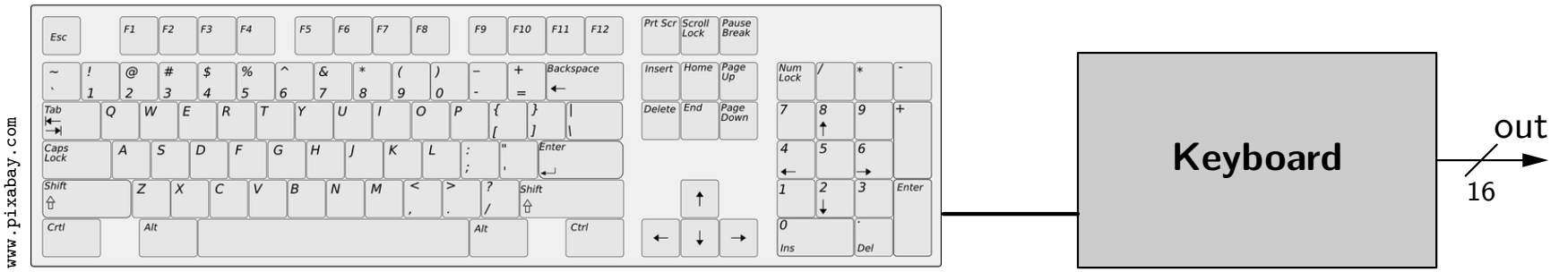
- Ausgang: Der Tastencode (scan code) der gedrückten Taste,
  oder 0, falls keine Taste gedrückt ist.

- Auslesen der Tastatur:

  Maschinenebene: Lese den Inhalt des `Keyboard`-Registers.

  Hochsprache: Verwende `keyPressed()`. (Betriebssystemfunktion, später)

Hack-Architektur: Tastatur



- Gewöhnliche Zeichen (z.B. Buchstaben) werden durch ihren ASCII-Code kodiert (American Standard Code for Information Interchange, Codes 0 bis 127).
- Sondertasten erhalten spezielle Tastencodes zugewiesen:

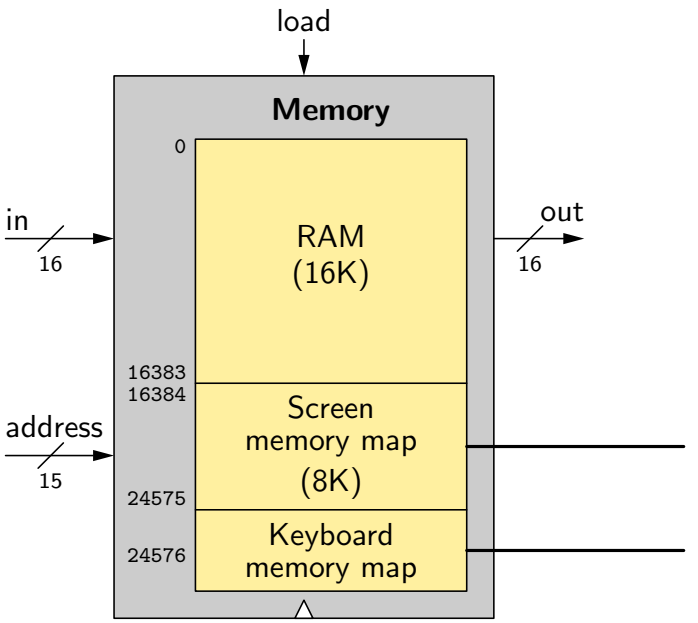
Taste	Code
Zeilenvorschub	128
Zurück	129
Pfeil nach links	130
Pfeil nach rechts	131
Pfeil nach unten	132
Pfeil nach oben	133

Taste	Code
zum Anfang	134
zum Ende	135
Bild hoch	136
Bild runter	137

Taste	Code
Einfügen	138
Löschen	139
Escape	140
F1 – F12	141-152

Bemerkung: Die Zuordnung von Codes zu den Tasten Zeilenvorschub, Zurück, Löschen und Escape ist merkwürdig, da es ASCII-Zeichen für diese Tasten gibt: Code 13 (carriage return), Code 8 (backspace), Code 127 (delete) und Code 27 (escape).

Hack-Architektur: Gesamtdatenspeicher

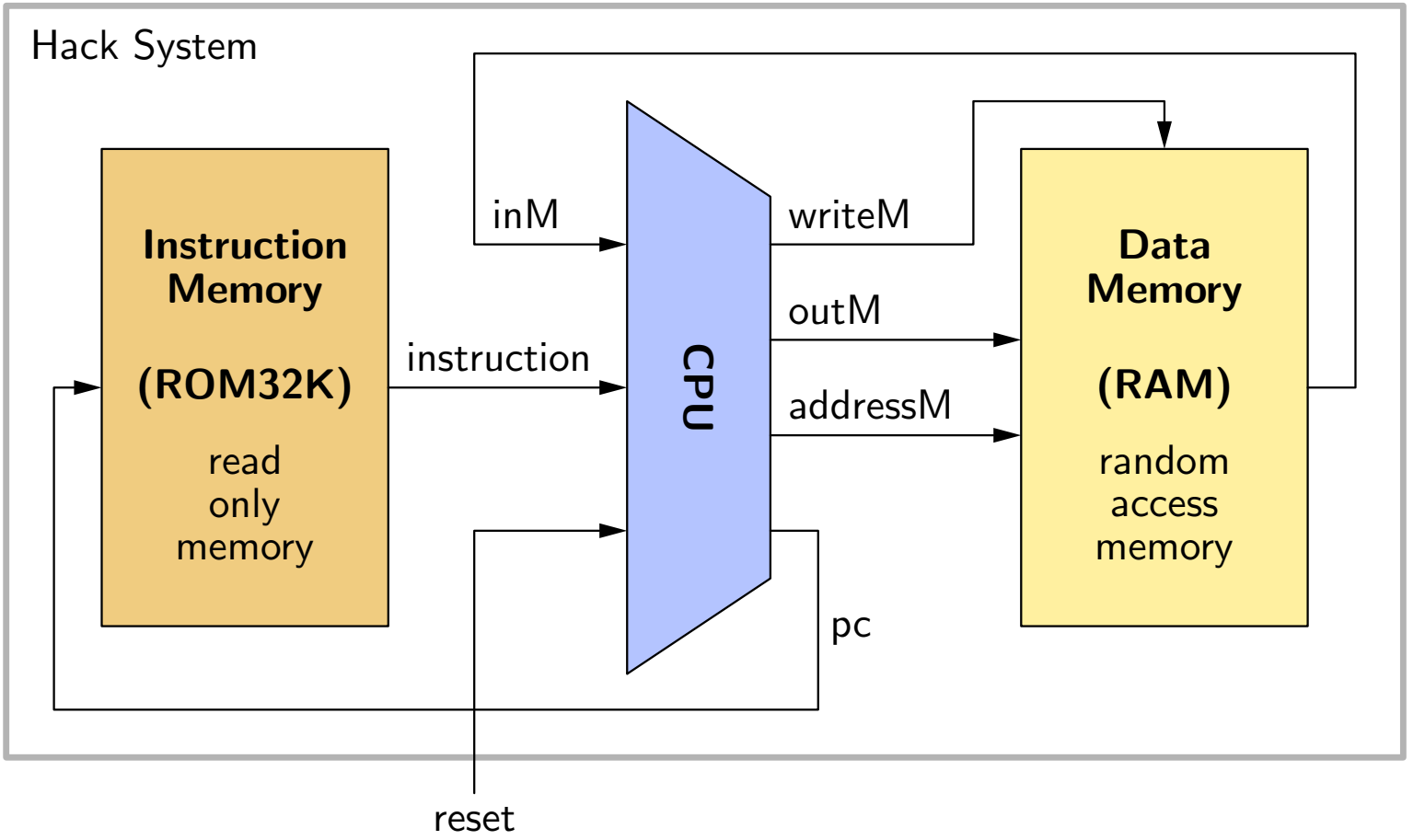


Der Daten-Adreßraum des Hack-Systems umfaßt

- normales RAM (random access memory)
- Bildschirmabbild
- Tastaturregister
- ungenutzten Adreßraum.

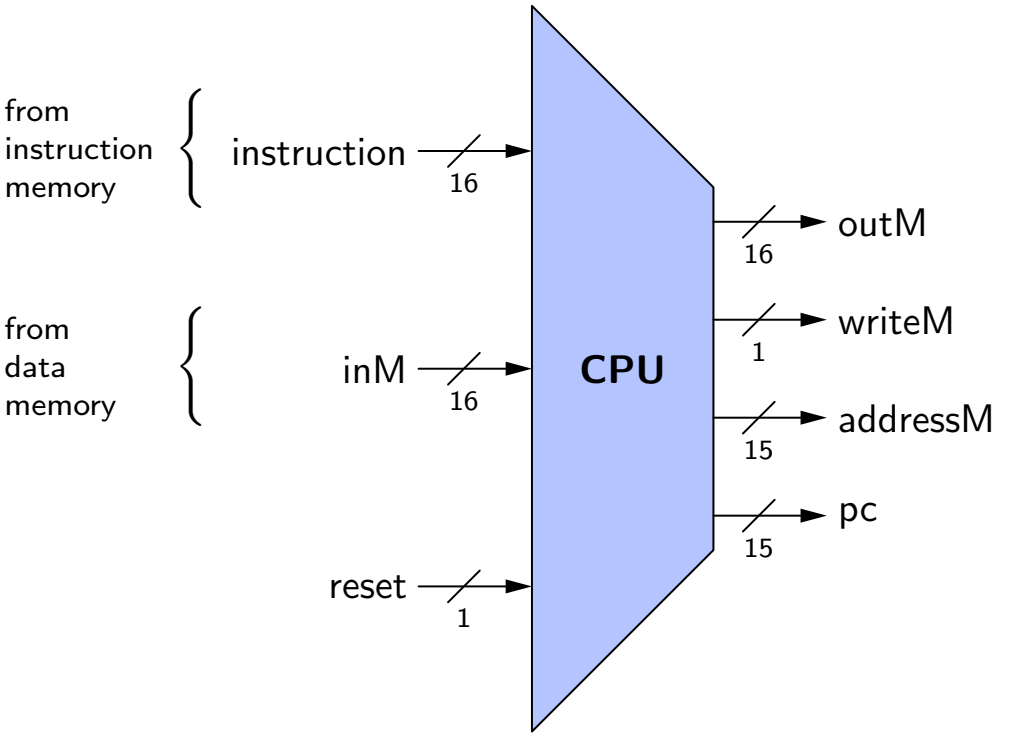
- Im Bereich der Adressen 0 bis 16383 liegt einfacher Datenspeicher (RAM).
- Im Bereich der Adressen 16384 bis 24575 liegt das Bildschirmabbild.
- An der Adresse 24576 liegt das Tastaturregister.
- Ein Zugriff auf Adressen jenseits von 24576 ist ungültig/unzulässig.

Hack-Architektur: Gesamtsystem



- Rechnerarchitektur aus [Noam Nisan & Shimon Schocken 2008] für die Hack-Plattform (pc: program counter, d.h. Befehlszähler).
- **Harvard-Architektur** (nicht von-Neumann-Architektur!)

Hack-Architektur: Prozessor (CPU)




```
CHIP CPU {
    IN instruction[16], inM[16], reset;
    OUT outM[16], writeM, addressM[15], pc[15];
    PARTS:
    // Implementation missing
}
```

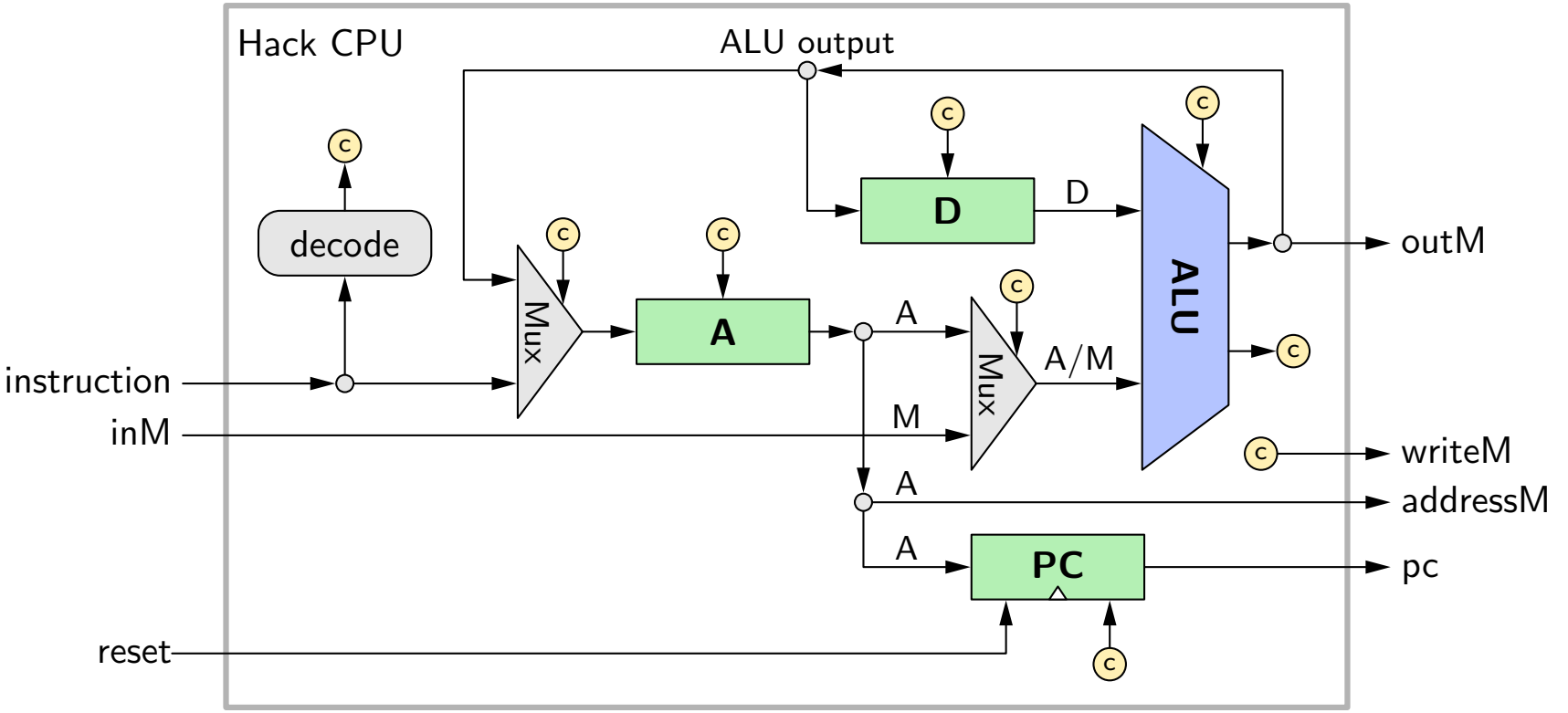
Bestandteile der CPU:

- arithmetisch-logische Einheit (ALU)
- A-Register
- D-Register
- PC-Register (programm counter)
- Steuereinheit (Befehlsdekodierung)

Funktion der CPU:

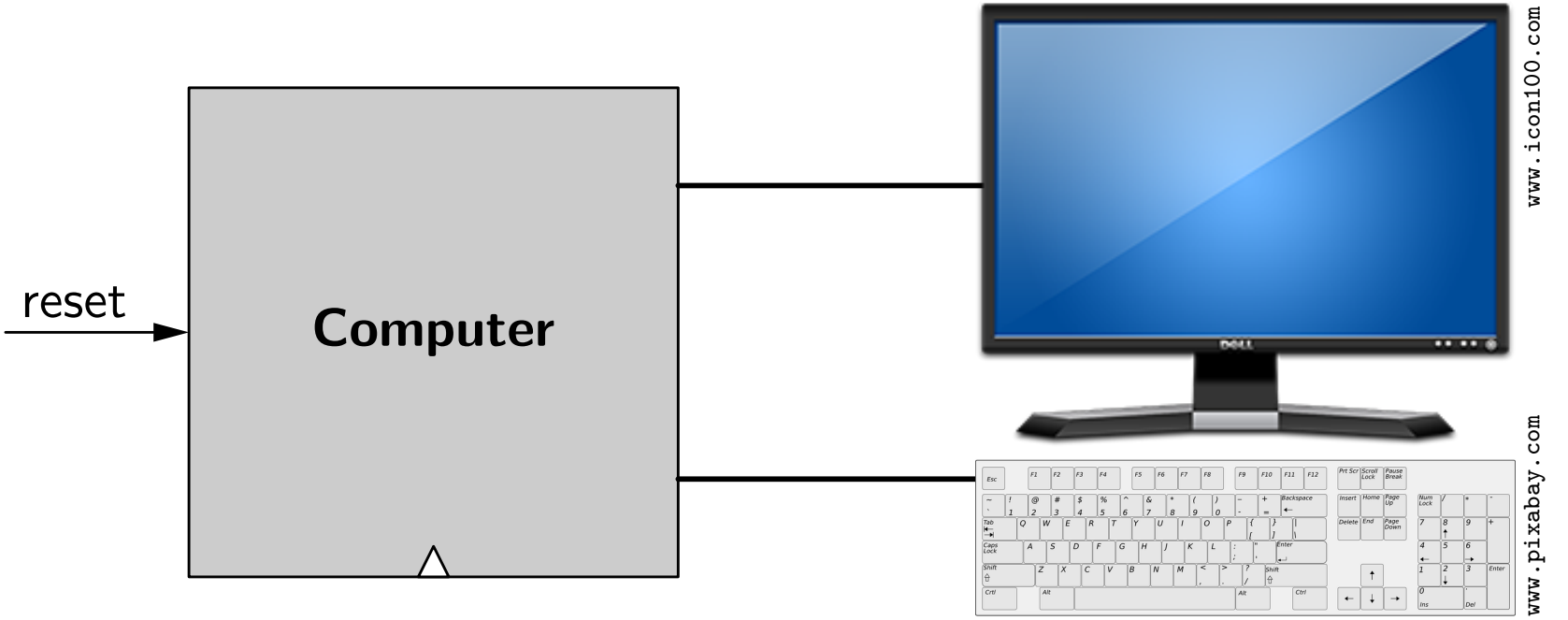
- Ausführen von Befehlen in Hack-Maschinensprache

Hack-Architektur: Prozessor (CPU)



- Nur **Daten- und Adreßpfade** sind gezeigt (d.h., Verbindungen, die Daten und Adressen transportieren), nicht jedoch die **Steuerlogik**, mit Ausnahme der Ein- und Ausgaben von Steuerbits, die durch (c) markiert sind.

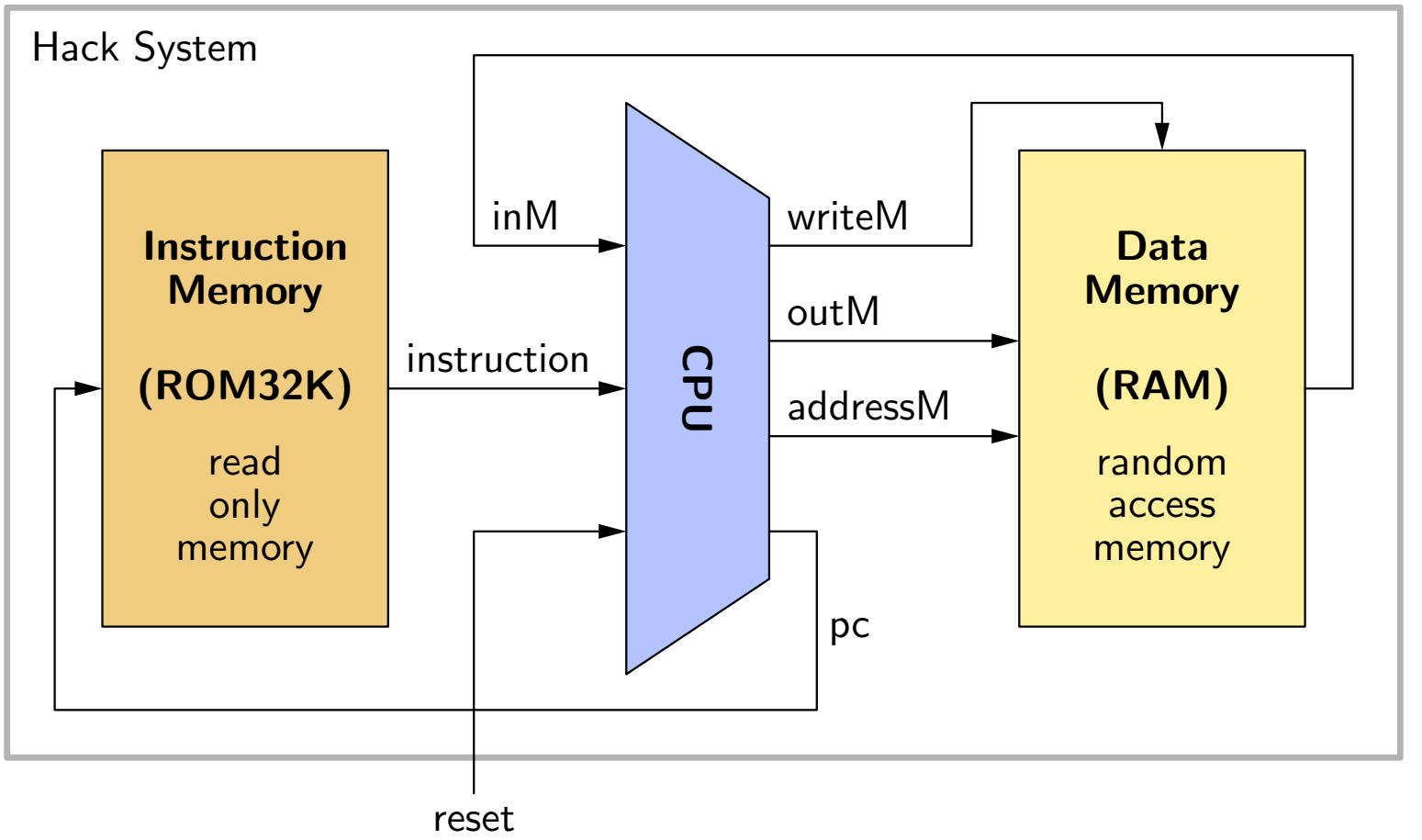
Hack-Architektur: Computer




```
Chip name: Computer // Topmost chip in the Hack platform
Input:     reset
Function:  When reset is 0, the program stored in the computer's ROM executes.
           When reset is 1, the execution of the program restarts.
           Thus to start a program's execution, reset must be pushed
           "up" (1) and "down" (0).

           From this point onward the user is at the mercy of the software.
           In particular, depending on the program's code,
           the screen may show some output and the user may be able
           to interact with the computer via the keyboard.
```

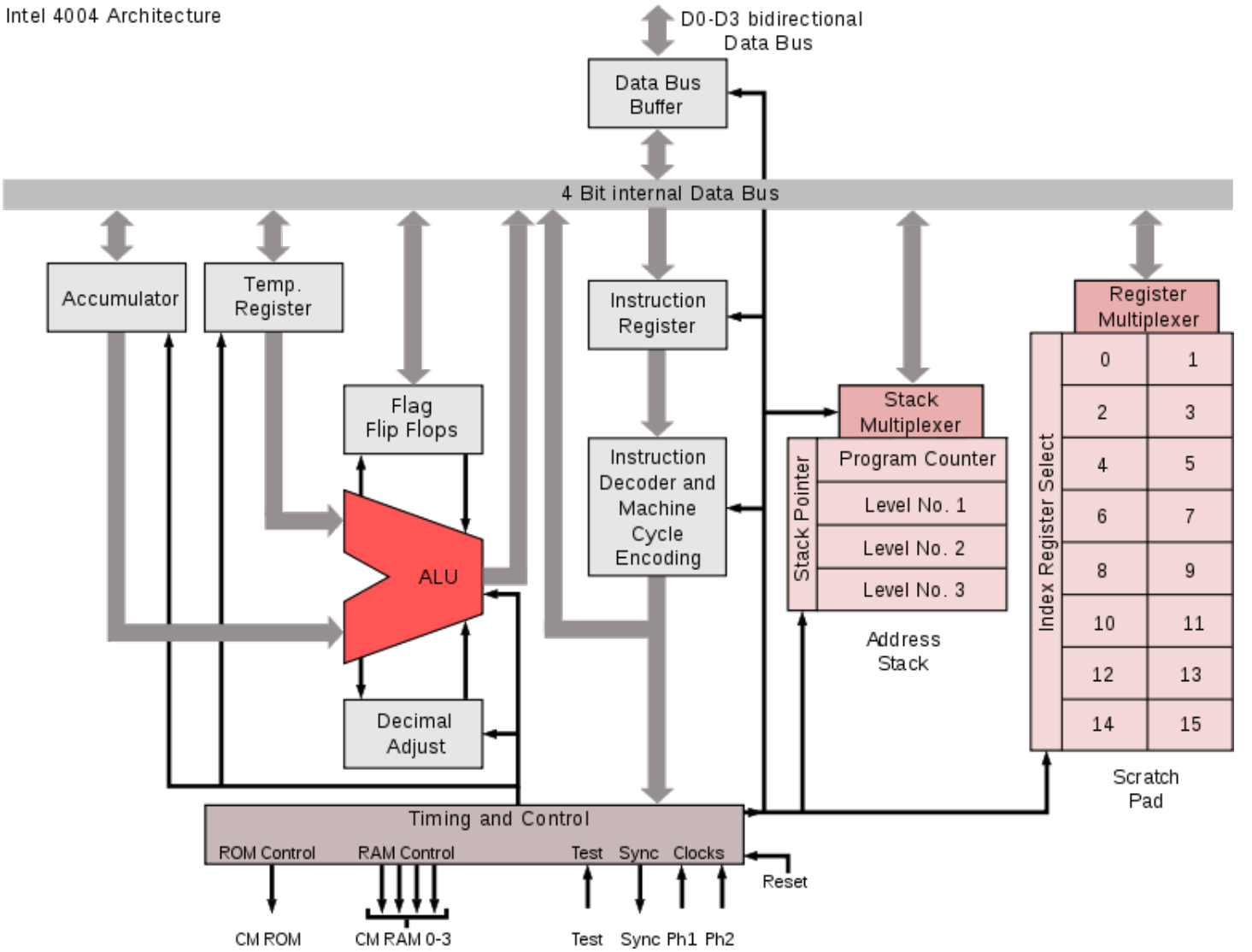
Hack-Architektur: Gesamtsystem



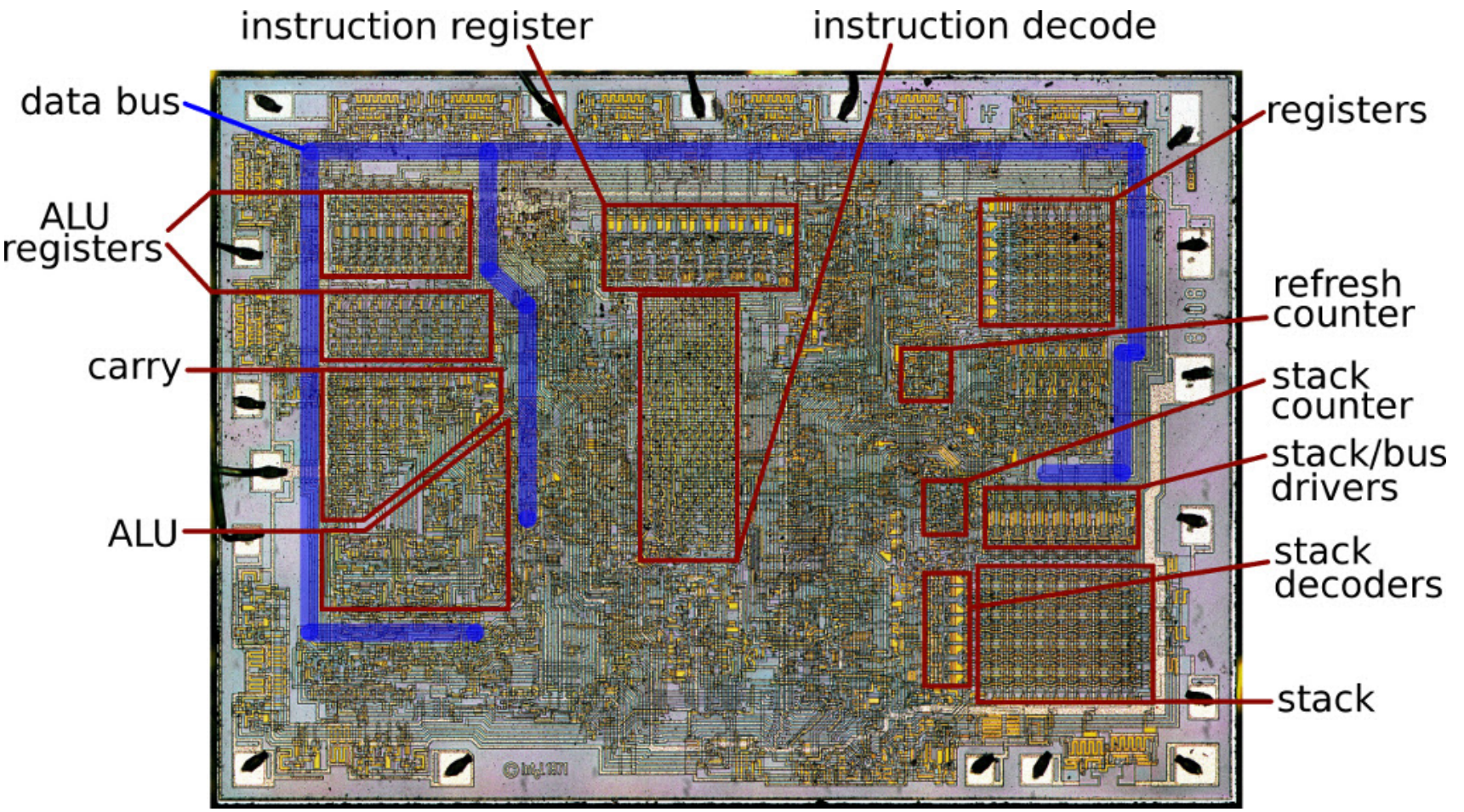
- Rechnerarchitektur aus [Noam Nisan & Shimon Schocken 2008] für die Hack-Plattform (pc: program counter, d.h. Befehlszähler).
- **Harvard-Architektur** (nicht von-Neumann-Architektur!)

Rechnerarchitektur: Prozessor

Intel 4004 [1971]
4/4/12 Bit, 500 kHz, 2300 Transistoren 10μm



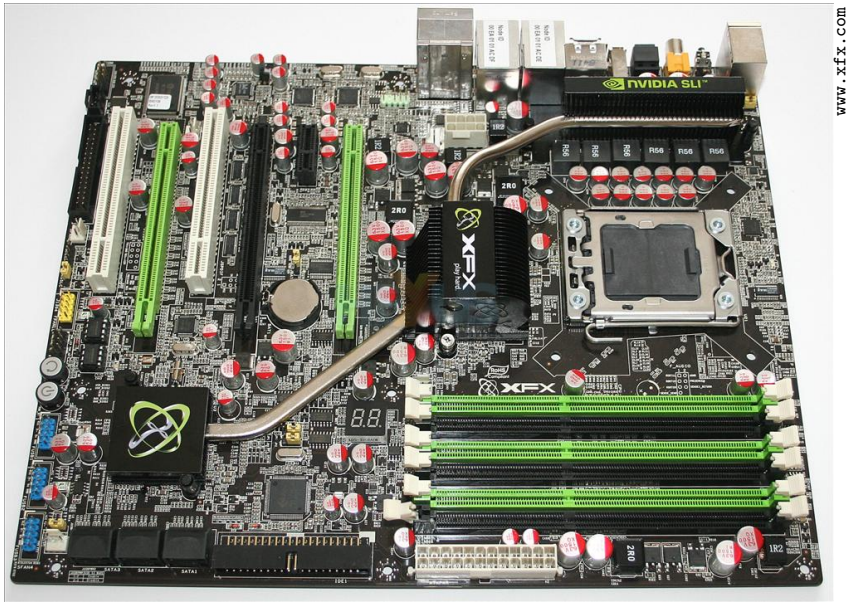
Rechnerarchitektur: Prozessor



Intel 8008 [1972]
8/8/14 Bit, 500 kHz, 3500 Transistoren, 10μm

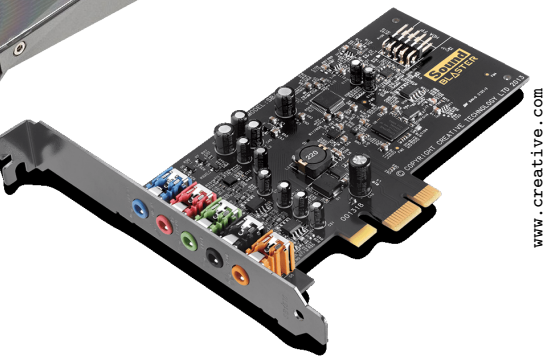
www.righto.com (Ken Shirriff) (modified)

Rechnerarchitektur: Rechnerbauteile



www.xfx.com

www.nvidia.com



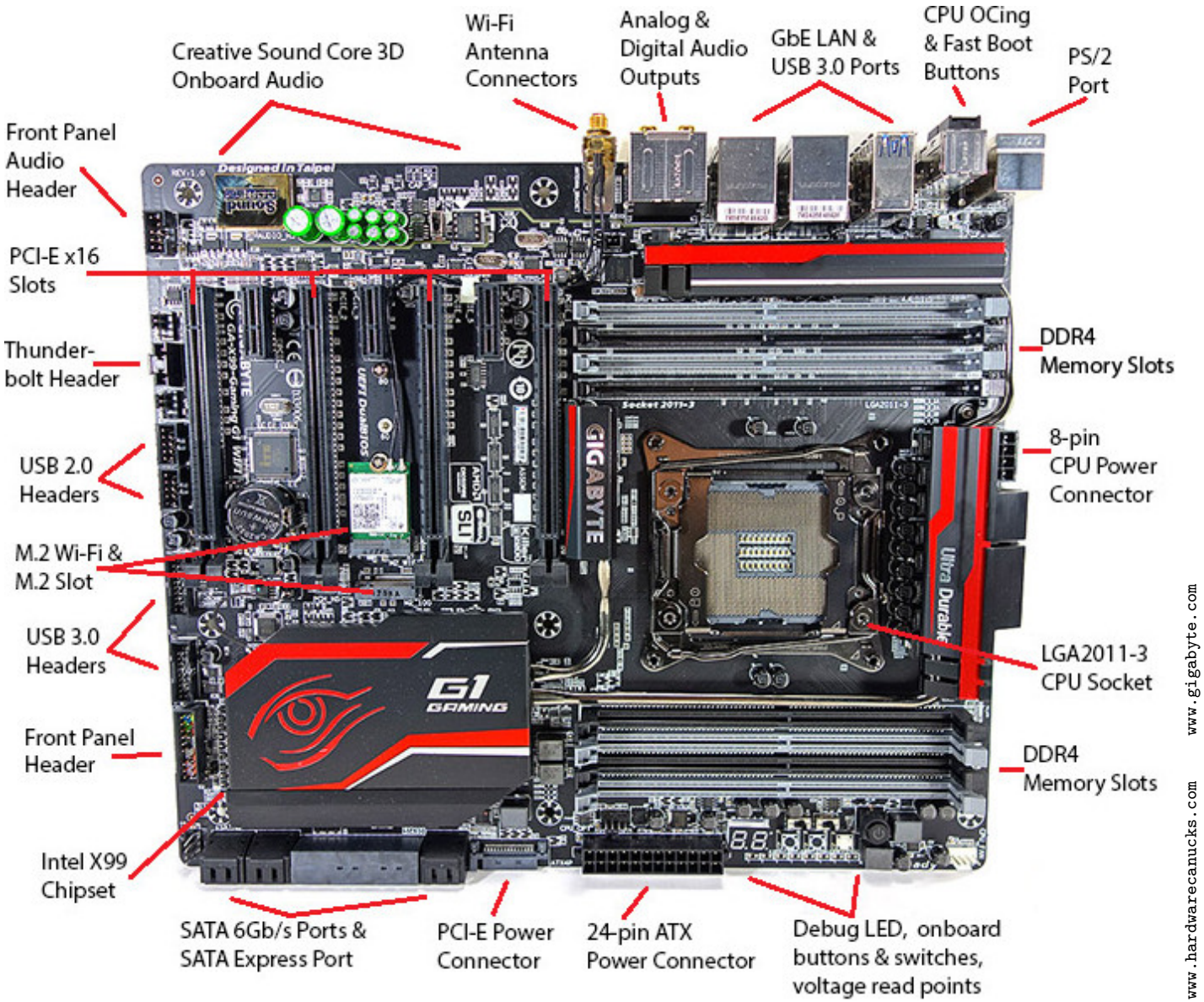
www.creative.com

www.intel.com

www.intel.com

www.crucial.com

Rechnerarchitektur: Hauptplatine



Zusammenfassung: Rechnerarchitektur

- **Speicherprogrammierung**
  - Festverdrahtete „Prozessoren“
  - Konzept der Speicherprogrammierung (stored program concept)
  - Befehlsabruf, -dekodierung und -ausführung (fetch-decode-execute cycle)
  - Rechnerarchitekturen (Harvard und von Neumann)

- **Die Hack-Plattform**
  - Befehls- und Datenspeicher (ROM32K und RAM16K)
  - Bildschirm und Bildschirmspeicher (screen)
  - Tastatur (keyboard)
  - Hauptspeicherorganisation (memory)
  - Prozessor (central processing unit, CPU)
  - Gesamtsystem (computer on a chip)