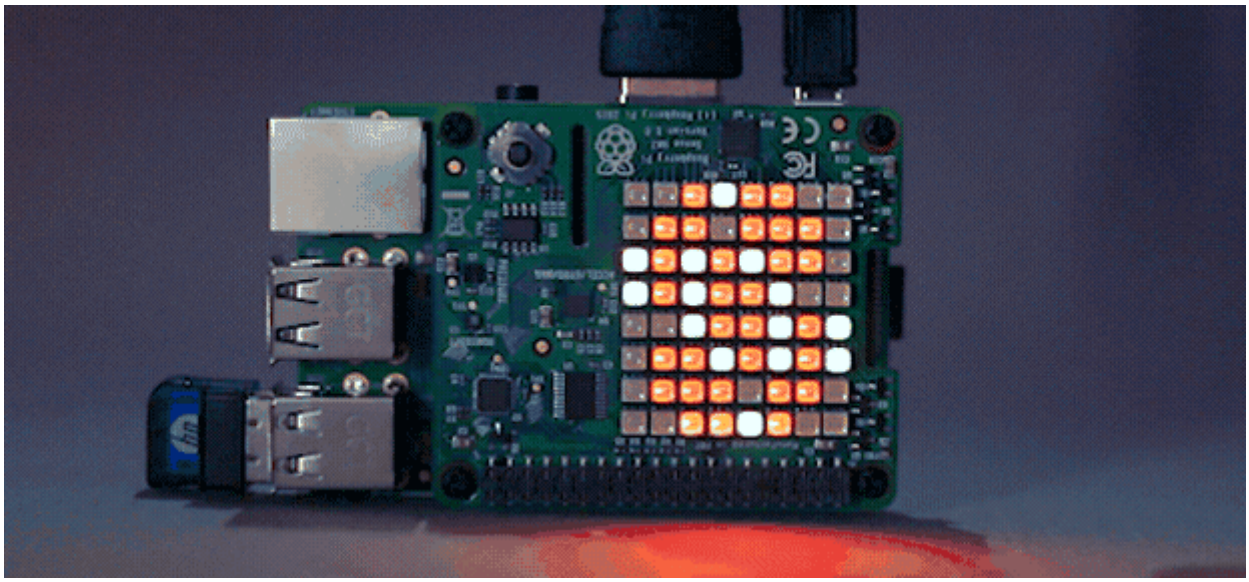
Einführung und Motivation

Parameter	Kursinformationen
Veranstaltung:	`Vorlesung Digitale Systeme`
Semester	`Sommersemester 2022`
Hochschule:	`Technische Universität Freiberg`
Inhalte:	`Wiederholung Grundbegriffe`
Link auf den GitHub:	https://github.com/TUBAF-IfI-LiaScript/VL_DigitaleSysteme/blob/main/lectures/01_Grundbegriffe.md
Autoren	Sebastian Zug, Karl Fessel & Andrè Dietrich



Grundbegriffe des Rechneraufbaus

Die interaktive Version des Kurses ist unter diesem Link zu finden.

Lernziele

- Wiederholung der Grundbegriffe der Rechnerarchitektur
- Diskussion von generellen Unterscheidungsmerkmalen bei der Auswahl eines Prozessors
- Ablauf des Compiliervorganges

Fragen an die heutige Veranstaltung ...

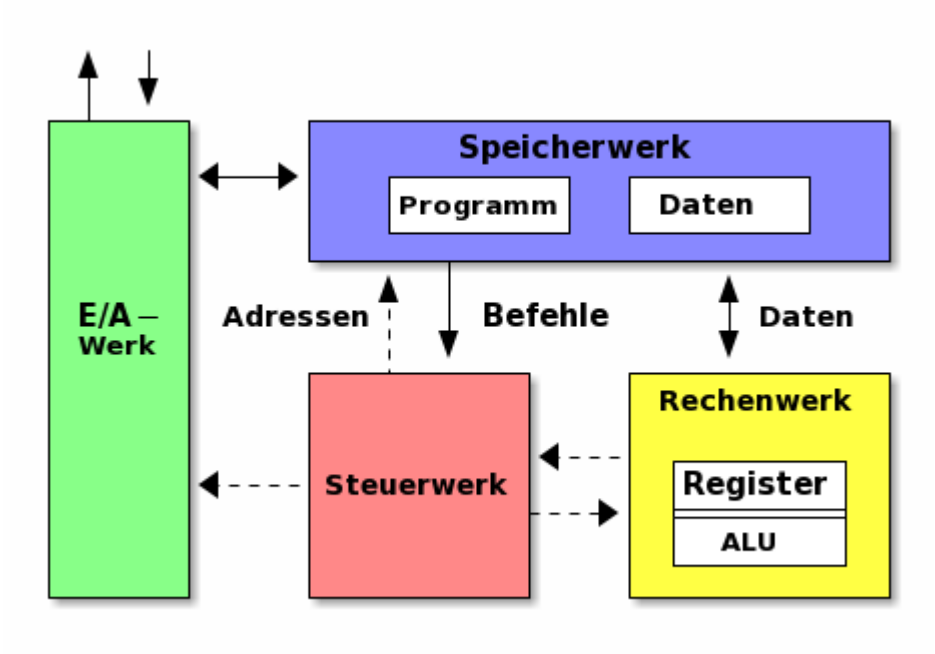
- Zählen Sie die Basiskomponenten eines Rechners auf.
- Erläutern Sie den Unterschied zwischen dem RISC und CISC Ansatz.
- Erklären Sie das Grundkonzept des Pipelining.
- Beschreiben Sie die Abläufe bei der Abarbeitung von Befehlen.
- Was bedeutet die Angabe *8Bit-Controller*?

Was sind die Grundkomponenten eines Rechners?

Der von Neumann-Rechner arbeitet sequentiell. Befehl für Befehl wird abgeholt, interpretiert, ausgeführt und das Resultat abgespeichert. Seine Komponenten definieren sich zu

- Steuerwerk (Taktgeber und Befehlszähler)
- Speicherwerk
- Rechenwerk (CPU)
- E/A-Einheit.

Die Datenbreite, Adressierungsbreite, Registeranzahl und Befehlssatz können als "Gestaltungsparameter" verstanden werden.



Das Steuerwerk

Das Steuerwerk ist für die Ausführung der Befehlsfolgen, die aus dem Speicher gelanden werden verantwortlich.

- Befehlszähler (Programm Counter)
- Einheit zur Befehlsentschlüsselung
- Einheit zur Steuerung der Befehlsausführung

Merke: Steuerwerke koordinieren das Laden und die Ausführung der Maschinenbefehle. Dafür muss das Steuerwerk selbst und die anderen Elemente konfiguriert werden.

Frage: Was passiert von seiten des Steuerwerkes bei der Ausführung von `i=i+1;`?

Die Umsetzung der Befehle im Steuerwerke kann *hart verdrahtet* oder auf der Basis von Mikroprogrammen realisiert werden.

Aspekt	**Kombinatorische Logik**	**Mikroprogramm**
Grundlegende Repräsentation	Endlicher Automat	Programm
Fortschaltung der Kontrolle	Expliziter Folgezustand	Programmzähler
Logische Repräsentation	Boolsche Gleichungen	Wahrheitstabelle
Implementierungstechnik	Gatter, Programmierbare Logikbausteine	R/W-Speicher, ROM

Zur Wiederholung sei ggf. auf die Vorlesung eingebettete Systeme verwiesen Model CPU.

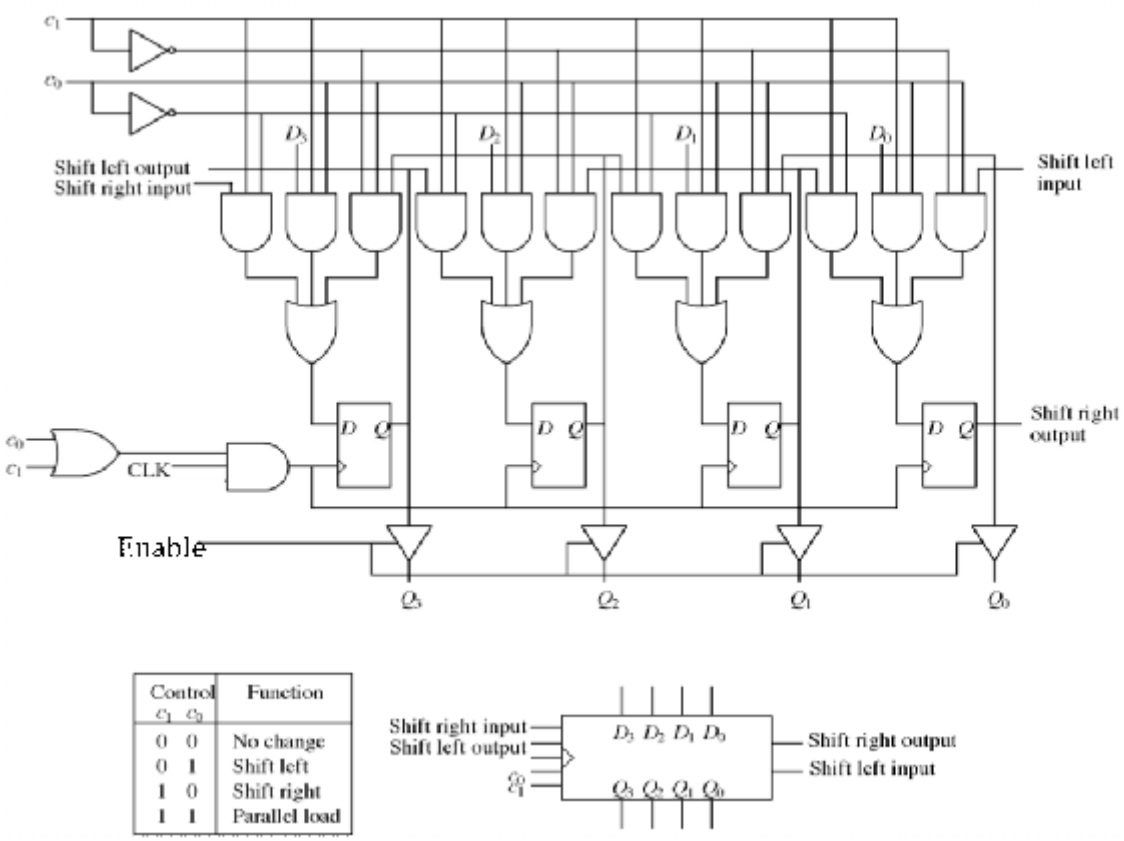
Das Rechenwerk (ALU)

Der Begriff Rechenwerk wird häufig synonym mit arithmetisch-logische Einheit (ALU) gebraucht, genau genommen stellt eine ALU jedoch lediglich eine (von oft vielen) Funktionseinheiten eines Rechenwerks dar, das zusätzlich aus einer Reihe von Hilfs- und Statusregistern besteht.

Folgende Klassen von Verknüpfungen sind in der ALU integriert:

- Arithmetische Operationen, wie Addition, Subtraktion, Vergleiche, Multiplikation und Division die
  - sich für einfache Controller auf Integerwerte beschränkt,
  - für leistungsfähigere Systeme aber Fließkomma-Operationen abdecken
- Logische Verknüpfungen, wie Disjunktion (ODER), Konjunktion (UND), Negation (NOT), Einer-Komplement (XOR)
- Verschiebeoperationen, wie Rotation, arithmetischer und logischer Shift

Wie lässt sich ein solches System praktisch realsieren? Schauen wir uns eine 4-Bit PIPO Schieberegister genauer an.



4-Bit PIPO Schieberegister [1]

Die spezifische ALUs integriert neben üblichen Hardware-Addierern/Multiplizieren auch spezielle Einheiten Multiply-Accumulate-Einheiten (MAC) oder Barrel-Shifter.

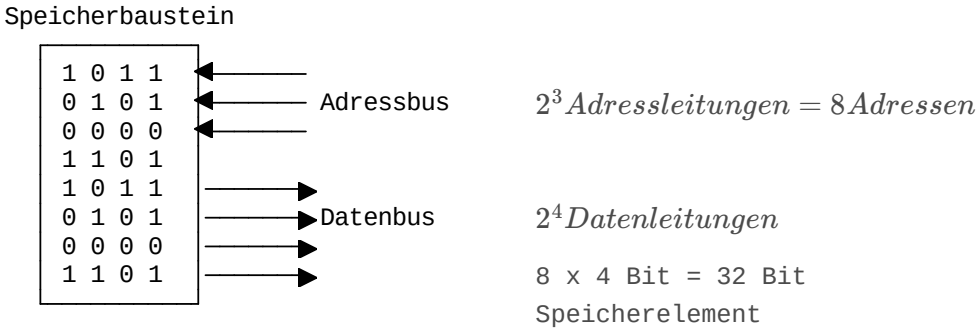
[1] Ali Alnoaman, Foreneintrag Electrical Engineering, Link

Speicherwerk

Das Speicherwerk umfasst das Programm und den Arbeitsspeicher. Im kommenden Abschnitt wird auf die Kombination beider Inhalte in ein und demselben Speicherbereich diskutiert.

Grundlegende Designvorgaben:

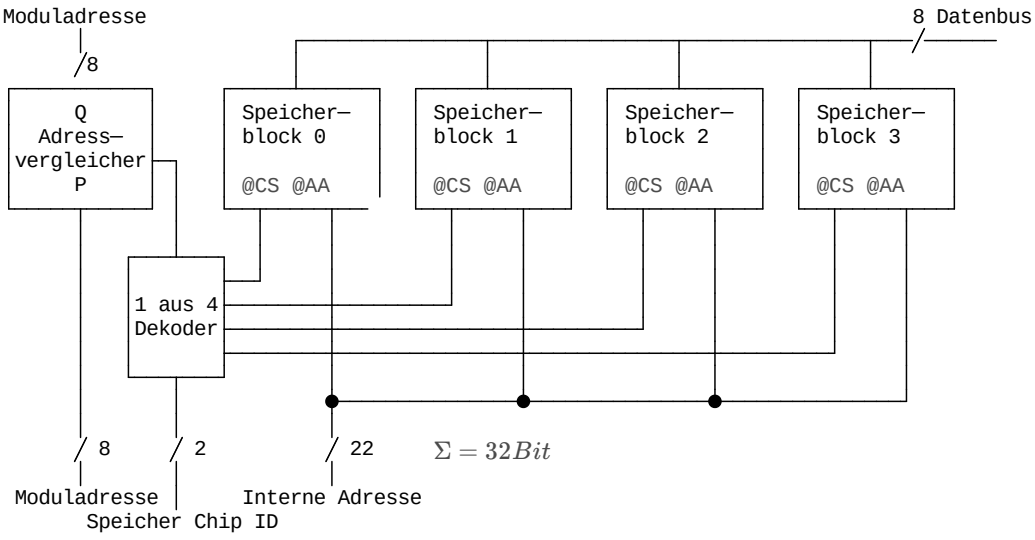
- Hinter jeder Adresse verbirgt sich nur ein Speicherbereich / Bauteil
- Der Adressraum wird maximal ausgenutzt, jedes Bauteil sollte nur unter einer Adresse zu erreichen sein.
- Der Speicher sollte kontinuierlich besetzt sein und keine Lücken aufweisen.



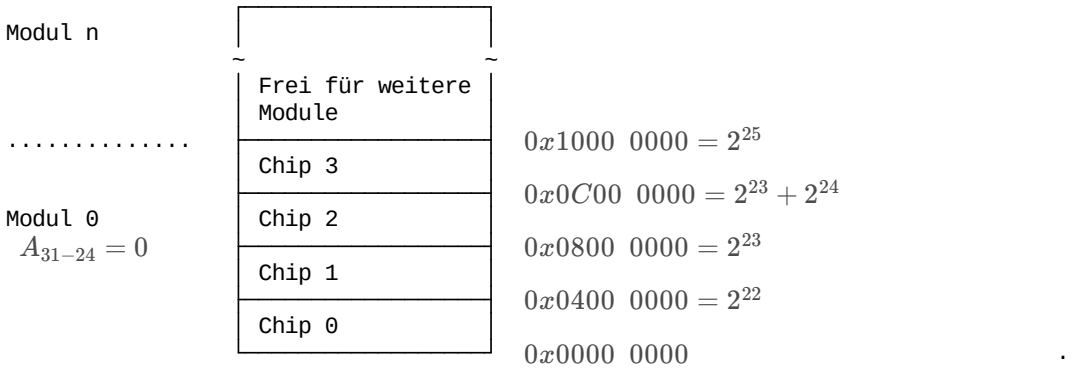
Wie verzahnen wir die Speicherkomponenten, wenn wir einen größeren Speicher in unseren Controller integrieren wollen?

Versuchen wir also einen zusammenhängenden Speicher von 16MByte zu entwerfen, der aus 4MByte Speicherelementen besteht. Daraus ergeben sich folgende Grundüberlegungen:

- 1 MegaByte entspricht $2^{10} KB$ $(1024 KB)$. Für die Adressierung von 4MByte benötigen wir also $2^{20} \cdot 4$ Adressen $2^{22} = 4194304$ referenzieren.
- Wir benötigen 4 der 4MByte Chips, die neben den eigentlichen Adressleitungen individuell angesprochen werden müssen
- Die Implmentierung soll erweiterbar sein, so dass wir als Chiphersteller unterschiedliche Speicherdimensionen anbieten können.



Daraus ergibt sich dann folgende Speicheraufteilung:

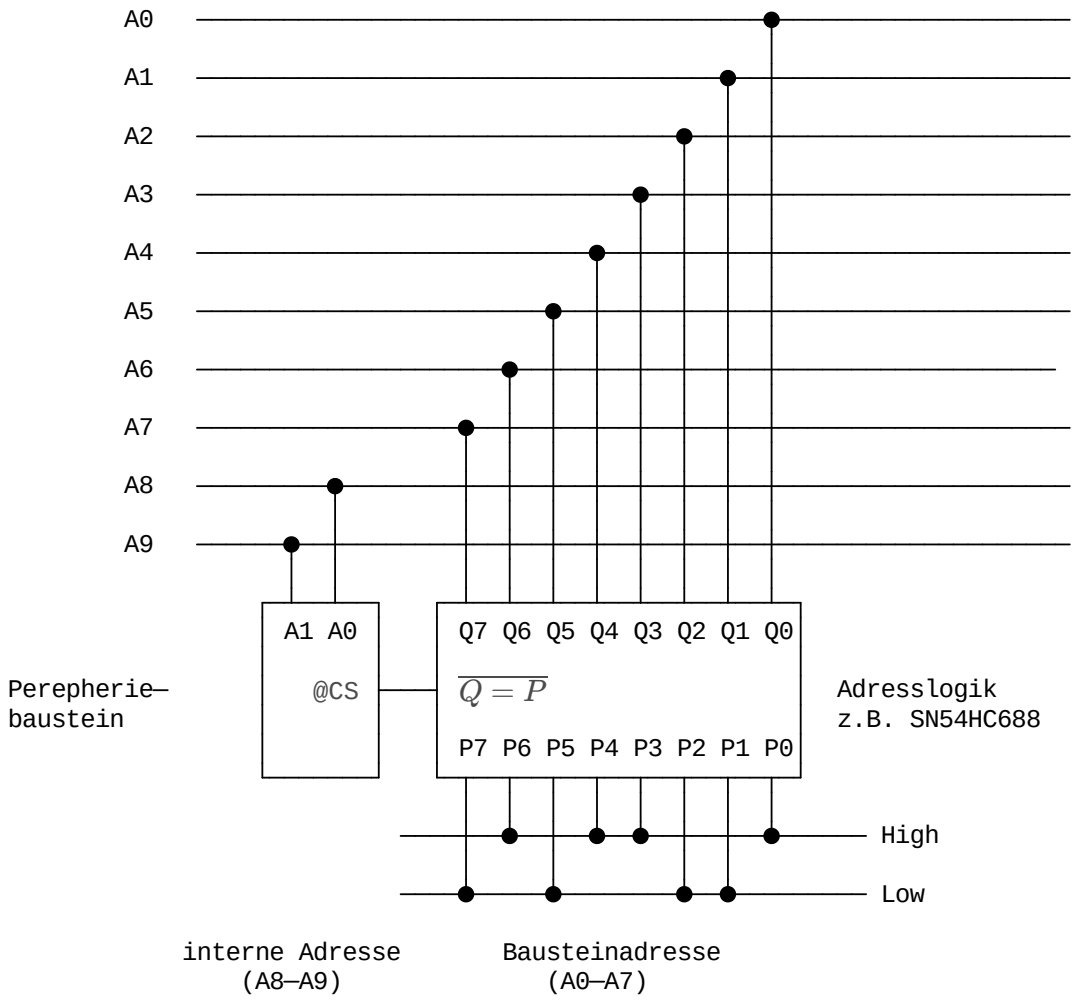


Frage: Was bedeutet das sogenannte Alignment innerhalb des Speichers?

Ein- & Ausgabe

Diese Komponente übernimmt die unmittelbare Erfassung von externen Informationen bzw. deren Bereitstellung. Im einfachsten Fall können dies einzelne Anzeigeelemente / Schalter sein, darüber hinaus dient es als Interface für weitere technische Bausteine. Alternative Bezeichnungen sind *I/O Ports* oder *I/O Kanäle*.

Wie kann die Adresskodierung der I/O Komponenten vorgenommen werden?



Frage: Was verbirgt sich hinter dem Begriff Memory-Mapped-IO?

Bus

Busse bilden für Daten- oder Adressen ein gemeinsames physikalischen Verbindungsnetzes aus mehreren Leitungen, an dem die relevanten Komponenten angeschlossen sind.

Vorteile:

- Flexibilität bei der Integration neuer Geräte
- Niedrige Kosten durch Mehrfachnutzung eines Kommunikationsmediums

Nachteile:

- Mehrfachnutzung durch verschiedene Geräte bremst die Bandbreite aus
- Kommunikationsflaschenhals (Maximale Bus-Datenrate kann Systemdurchsatz begrenzen)
- Größtes Anforderungsprofil bestimmt die Konfiguration / Kompromisslösung

… und wie wird daraus eine Architektur?

> Rechnerarchitektur ist ein Teilgebiet der Technischen Informatik, das sich mit dem Design von Rechnern (Computern) und speziell mit deren Organisation sowie deren externem und internem Aufbau (was ebenfalls mit 'Rechnerarchitektur' bezeichnet wird) beschäftigt.

> Die **Architektur** eines Rechners beschreibt dessen grundsätzlichen Aufbau (Hardwarestruktur) und das Zusammenspiel der Komponenten (Organisationsstruktur).

Unterschied Harvard und die von Neumann-Architektur?**

Umsetzung

Und wie sieht das Ganze in einem realen System aus? Der Intel 4004 ist ein 4-Bit-Mikroprozessor des Mikrochipherstellers Intel, der am 15. November 1971 auf den Markt kam. Er gilt als der erste Ein-Chip-Mikroprozessor, der in Serie produziert und am freien Markt vertrieben wurde. Der "Rechner" an sich setzte sich aus zunächst vier einzelne Bausteinen zusammen:

- 4001: ein 2048-Bit-ROM (adressiert in 256 8-Bit-Adressen) mit einem 4-Bit-Ausgabeport
- 4002: 80×4-Bit-RAM-Datenspeicher mit einem 4-Bit-I/O-Port
- 4003: I/O-Erweiterungs-Chip, bestehend aus einem statischen Schieberegister
- 4004: die eigentliche CPU

Architektur des 4004 [1]

Element	**Bestandteile**
Steuerwerk	*Instruction Register*, *Instruction Decoder*, *Timing and Controll*
Rechenwerk	*ALU*, *Flag Flip-Flops*, *Decimal Adjust*, *Temp Register*, *Accumulator[-register]*
Eingabe-Ausgabe	nicht dargestellt
Speicherwerk	*Data Base Buffer*

Speicherauszug den Intel 4004:

Adresse	Speicherinhalt	OpCode	Mnemonik
0010	1101 0101	1101 DDDD	LD $5
0012	1111 0010	1111 0010	IAC

Unterstützung für die Interpretation aus dem Nutzerhandbuch, dass das Instruction-Set beschreibt:

Auszug aus dem Handbuch des 4004 [2]

[1] Intel 4004 (Autor Appaloosa)

[2] Intel 4004 Assembler

Befehlsabarbeitung und Instruktions-Sets

Einschub: Die Abarbeitung eines Befehls kann in mehrere Teilschritte zerlegt werden:

- Befehl holen (fetch)
- Befehl entschlüsseln (decode)
- Befehl ausführen (execute)
- Ergebnis speichern (Write back)

Frage: Welchen Befehlssatz sollte uns Prozessor umfassen?

RISC (Reduced Instruction Set) stellt tatsächlich eine “Entwurfs-Philosophie“ dar, in der das Ziel höchste Leistung ist, die im Zusammenspiel von Hardware und einem optimierenden Compiler erreicht wird. Im RISC-Ansatz wird in der der Instruktionssatz in Hinblick auf Einfachheit und Regularität entworfen, so daß die Verwendung der Instruktionen durch den Compiler einfach und überschaubar ist.

Nachteile	Vorteile
Mehr Speicherplatz für Programme	Einfachheit der Hardwarerealisierung
Aufwändigere Implementierung IM ASSEMBLER	höhere Taktraten
	Ausnutzung von Pipelining Techniken

	CISC	RISC
Ausführungszeit einer Instruktion	$>= 1$	meist 1
Instruktionssatz	groß	klein
Instruktionsformat	variabel	strikt
unterstützte Adressierungsschemata	komplex	einfach (Load/Store)
CPU-Abarbeitungslogik	Mikroprogramm	Hardware
Komplexität	Hardware	Compiler

Merke: Die Abgrenzung zwischen Complex Instruction Set (CISC) vs Reduced Instruction Set (RISC) ist heute kaum noch präsent. Vielmehr verschmelzen die Konzepte in aktuellen Prozessoren.

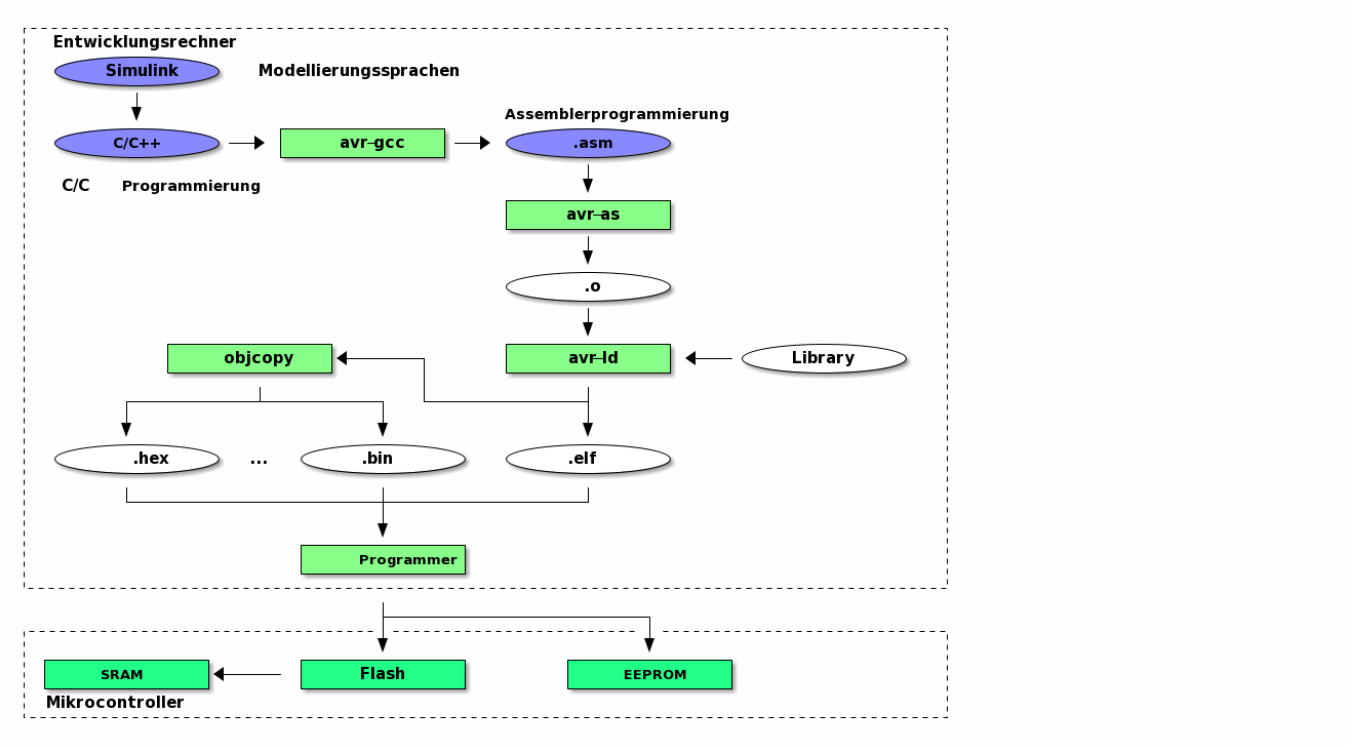
Umsetzung in verschiedenen Controllern

Aspekt	Intel 4004	Microchip Atmega	Texas Instruments MSP 430	STM 32
Architektur	Harvard	Harvard	von-Neumann	Harvard
Busbreite	4	8	16	32
Registeranzahl	16	32	16	16
Befehlssatz	45	131	51	
Geschwindigkeit	740 kHz	16 MHz	25 MHz	480 MHz

Vom C Programm zum ausführbaren Code

Aus den vorangegangen Aussagen ist klar geworden, wie die Komponenten des Rechners während der Abarbeitung eines Programmes zusammenarbeiten. Wie aber entsteht der **prozessorspezifische** Maschinencode?

Die Verarbeitungskette in folgender Grafik adressiert zwar explizit die AVR Tools, lässt sich aber auf jede Plattform übertragen.



Hausaufgaben

Sofern Sie noch keine Erfahrung in der Entwicklung mit AVR-Controllern haben, sollten Sie sich mit den avr-gcc Tutorials auseinander setzen.