Grundkonzepte der Rechnerarchitektur

Vorlesung Mikroprozessortechnik

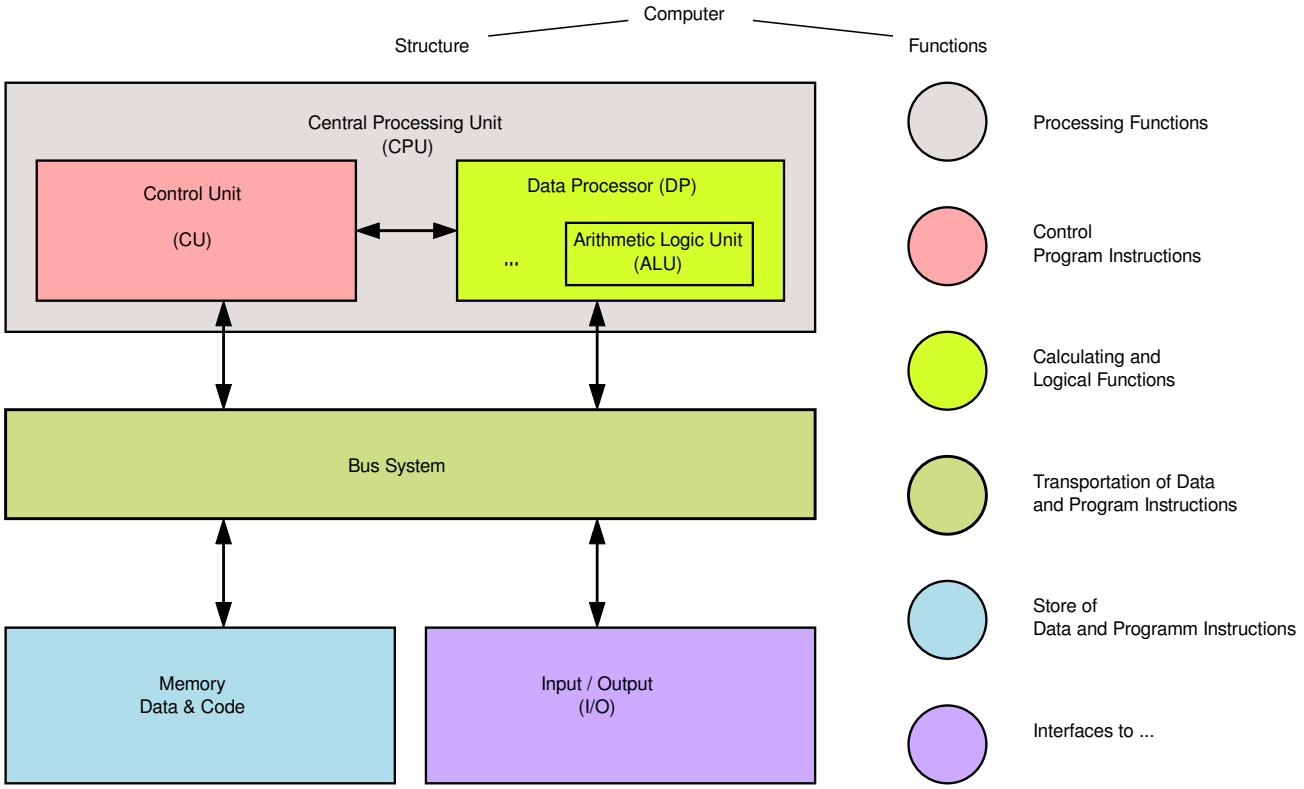
HAW Hamburg

29. Dezember 2017

John von Neumann (1903-1957)

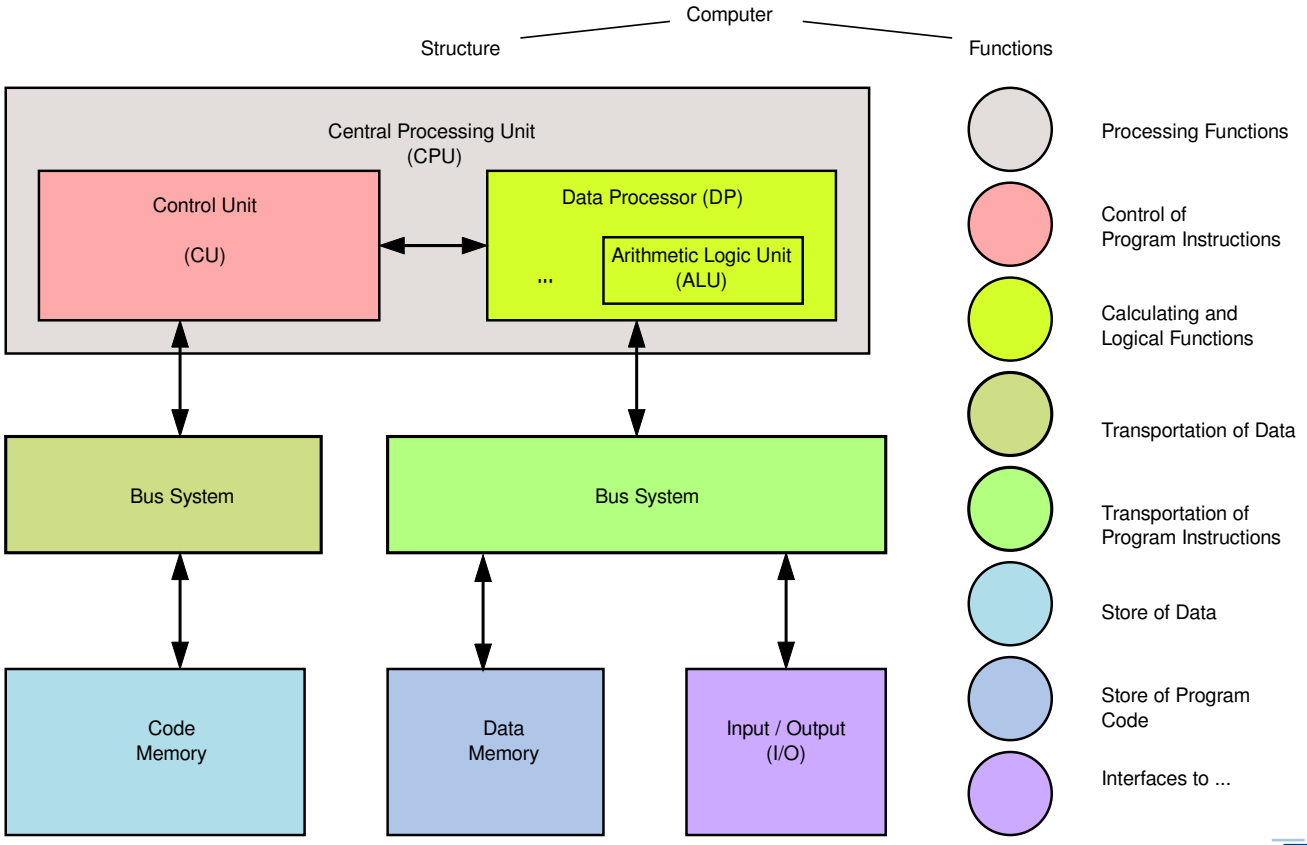
John von Neumann ca. 1940 (Quelle: Heinz-Nixdorf-Institut)

von-Neumann-Architektur 1945



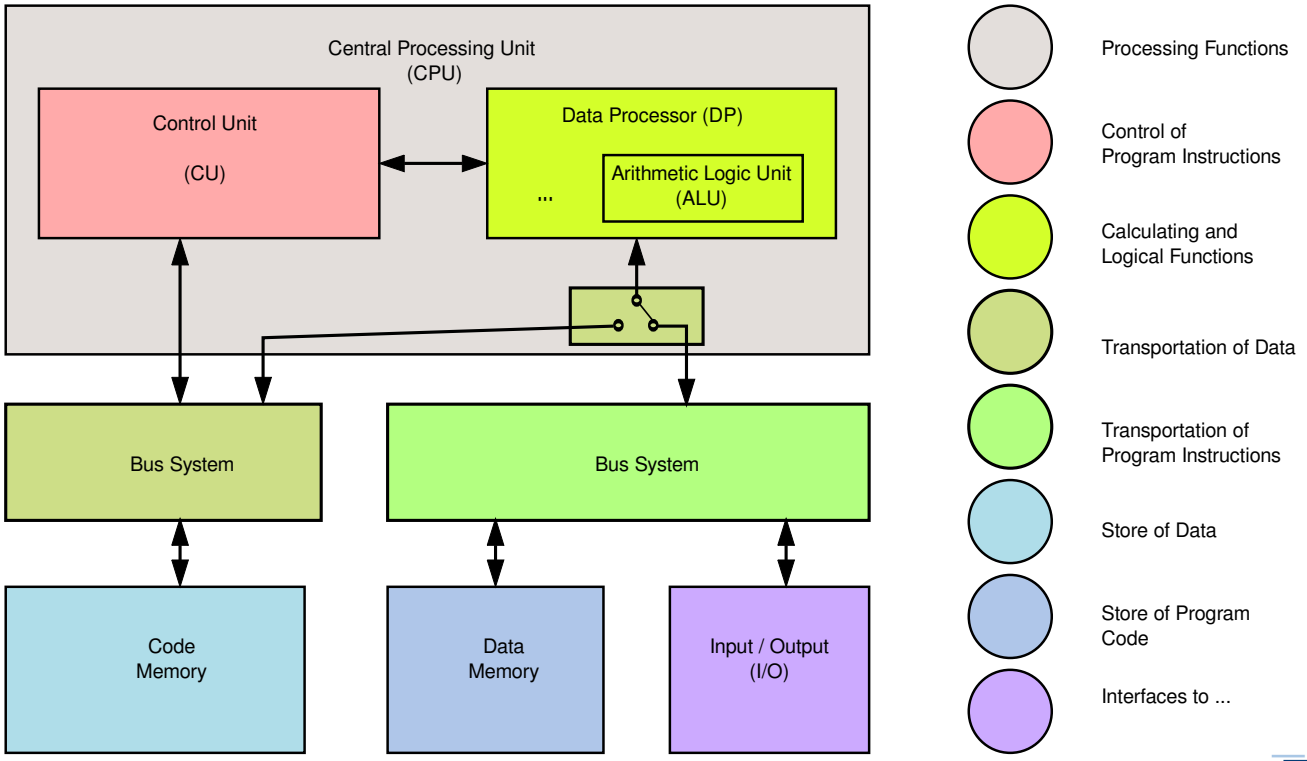
John von Neumann: First Draft of a Report on the EDVAC (auch Princeton Architectur)

Harvard-Architektur



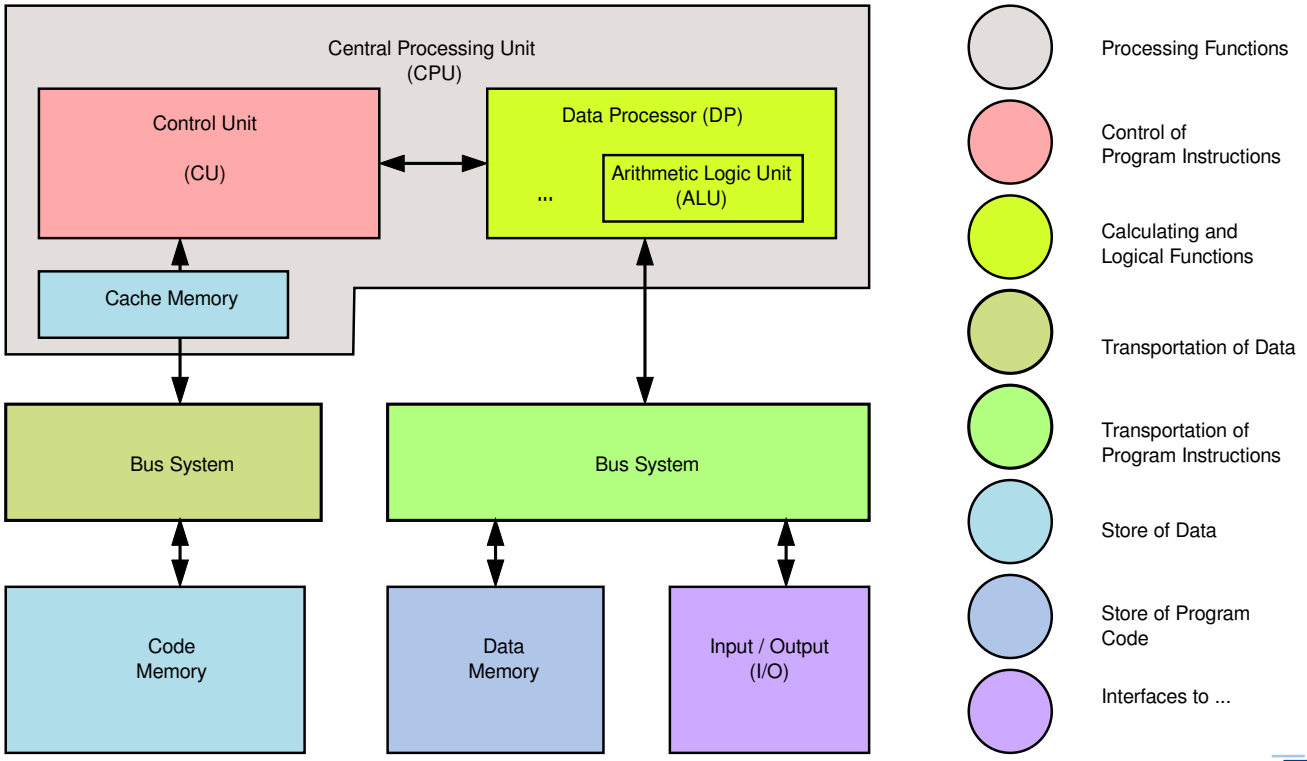
An der Harvard University entstand ebenfalls 1945-1946 ein anderes Konzept)

Erweiterte Harvard-Architektur



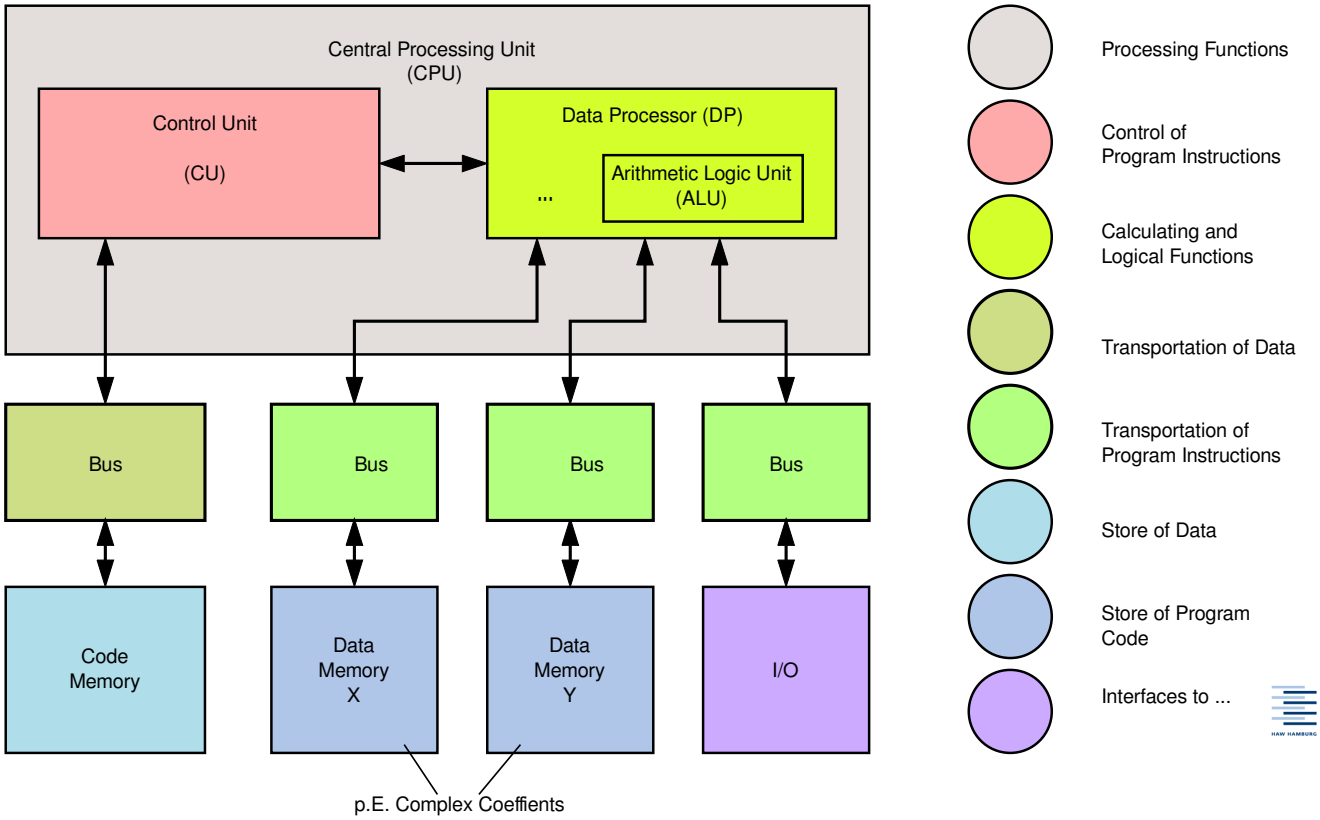
Die Harvard-Architektur wurde für ladbaren Programcode modifiziert.

Moderne Harvard-Architektur

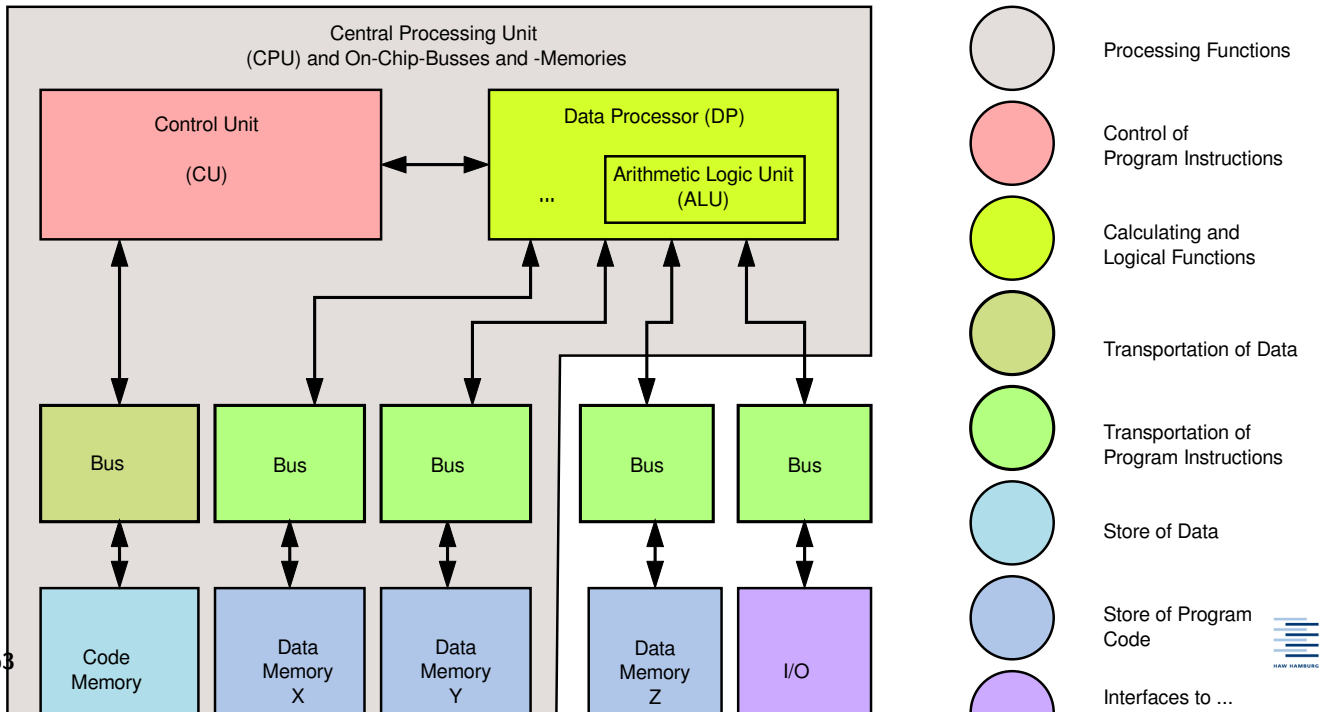


Die Trennung von Code und Datenspeicher lässt besonders vorteilhaft schnelle Zwischenspeicher zu (Caches).

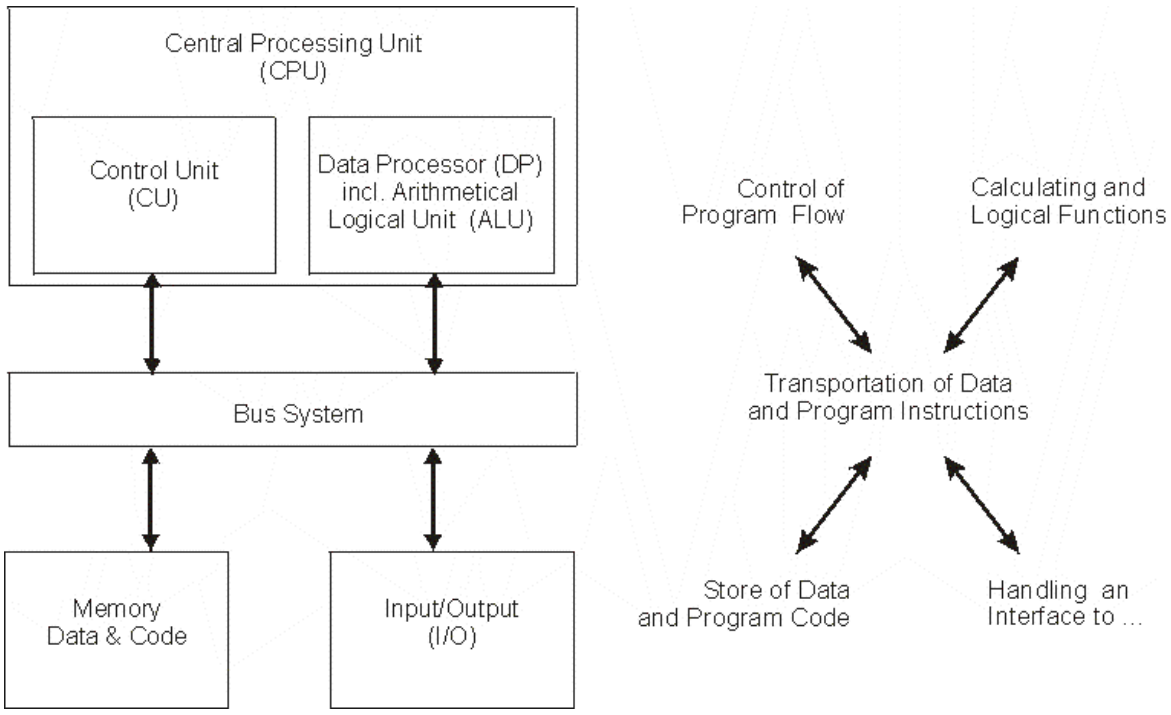
Harvard-Architektur in digitalen Signalprozessoren (DSP)



Harvard-Architektur in DSP-Systemen mit separaten Speichern und Bussystemen auf dem Chip

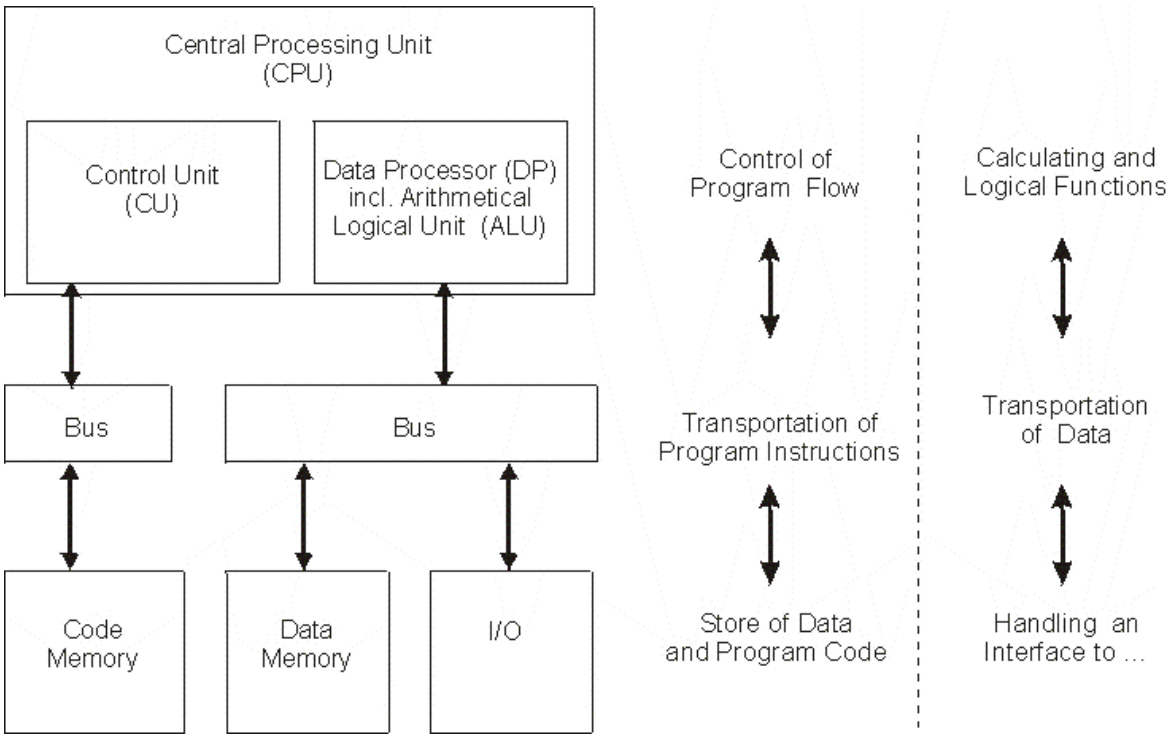


Die Von-Neumann-Architektur hat ein Bus-System



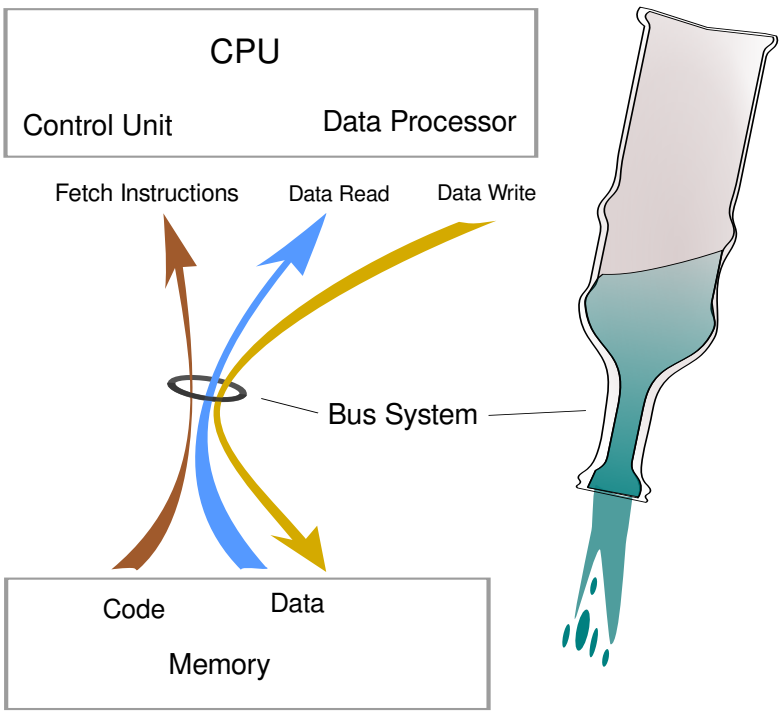
Die von-Neumann-Architektur nutzt ein Bus-System im Multiplex-Betrieb für Code und Daten. Programmcode und Daten sind in einem gemeinsamen Speicher abgelegt.

Die Harvard-Architektur hat zwei Bus-Systeme



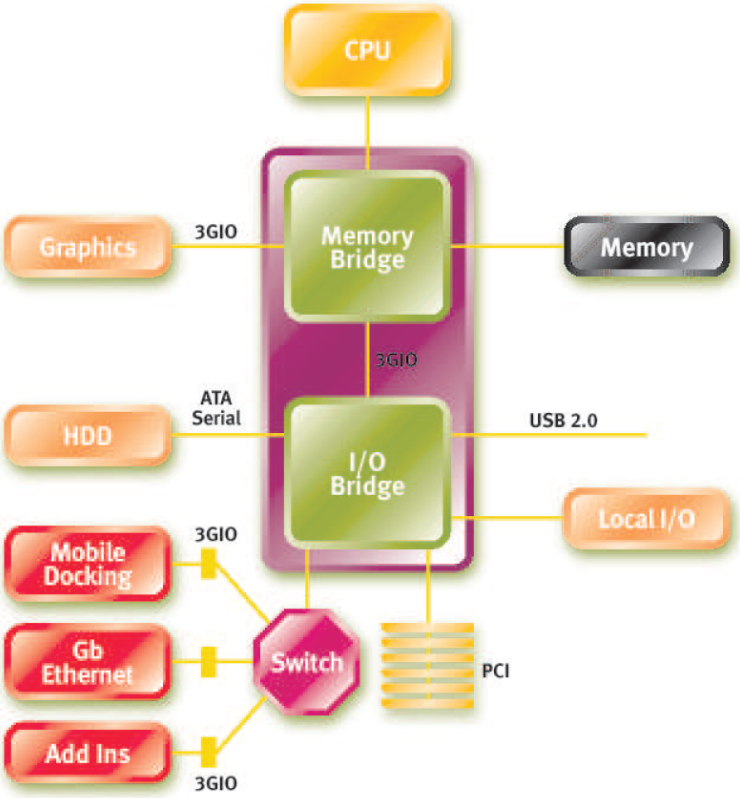
Die Harvard-Architektur nutzt getrennte Bus-Systeme für Code und Daten. Programmcode und Daten sind in getrennten Speichern abgelegt.

Von-Neumann-Bottleneck



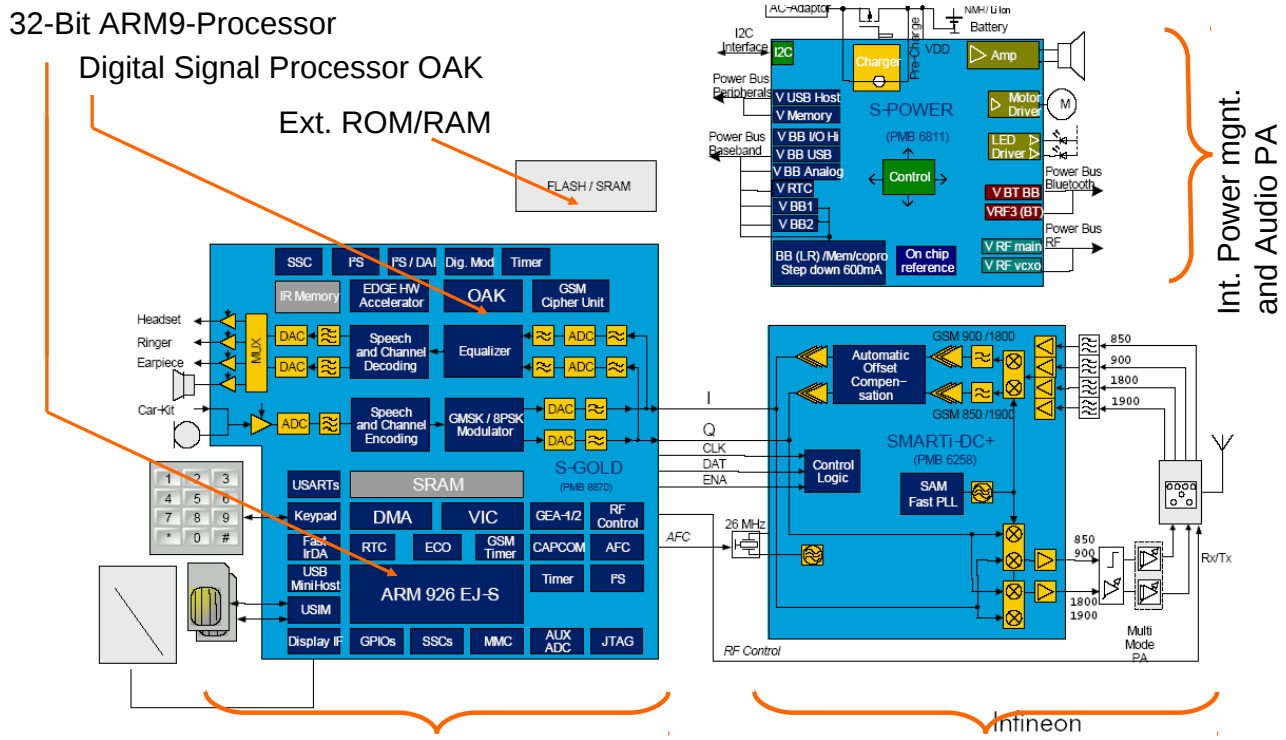
Der Von-Neumann-Bottleneck entsteht durch den Transfer von Daten und Code nacheinander über den Bus (Engpass).

Moderne PC-Architektur

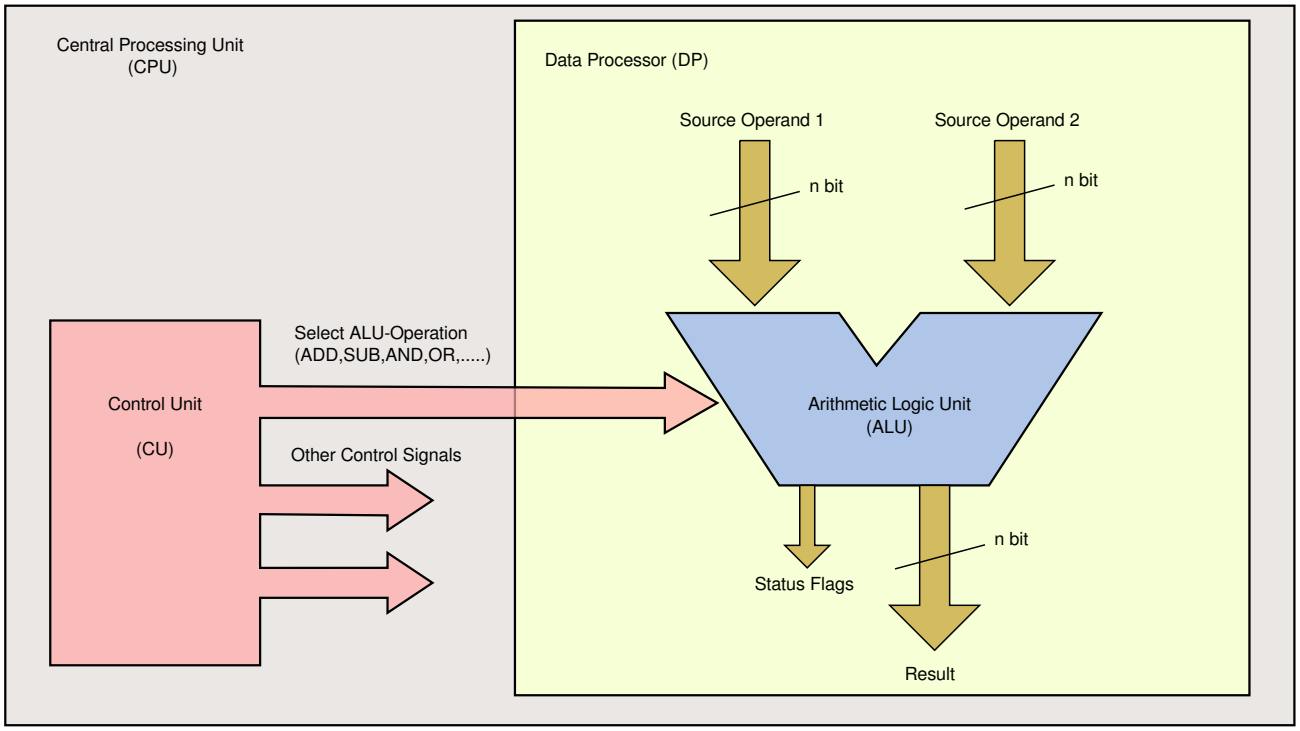


Die Firma Intel prägt die Konzepte heutiger PC-Systeme: x86-Architekturen

Chipsatz eines Mobiltelefons

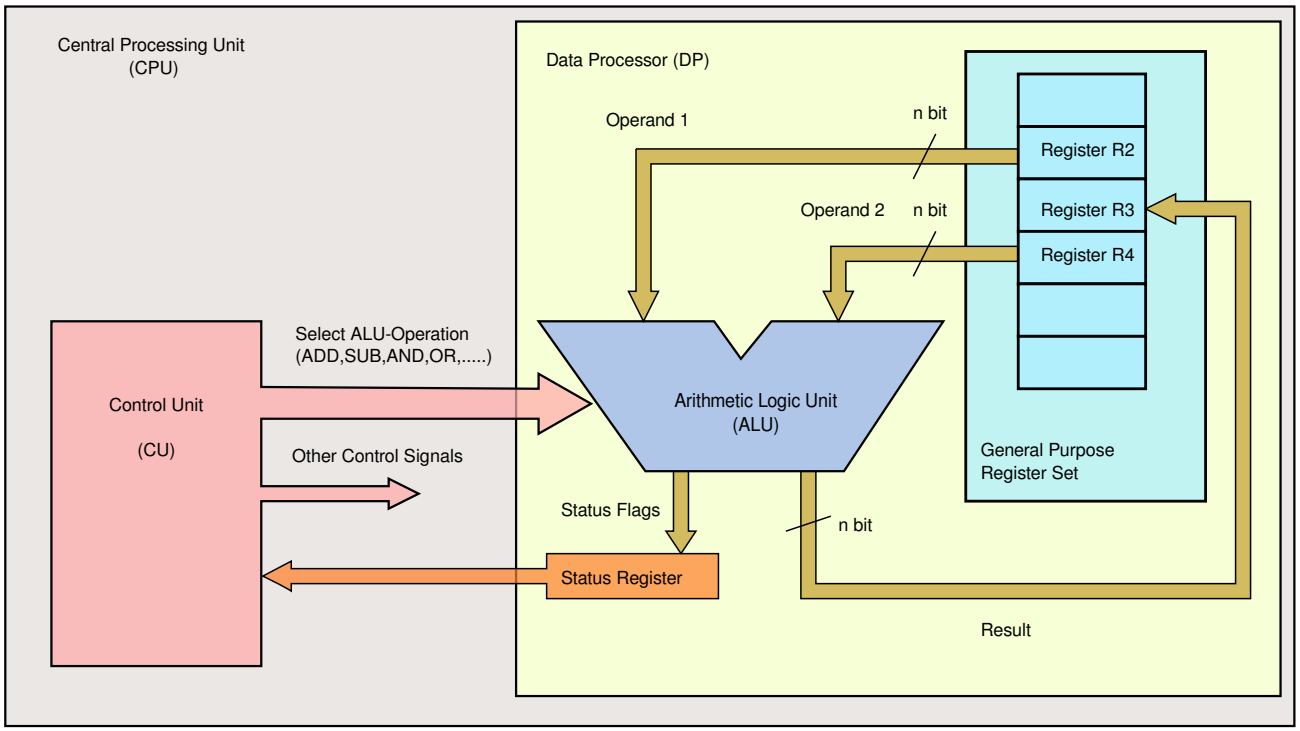


Control Unit und Data Processor



Die Control Unit wählt die Funktion der Arithmetic Logic Unit (ALU) aus = Operation. Die Operanden werden in der ALU verarbeitet.

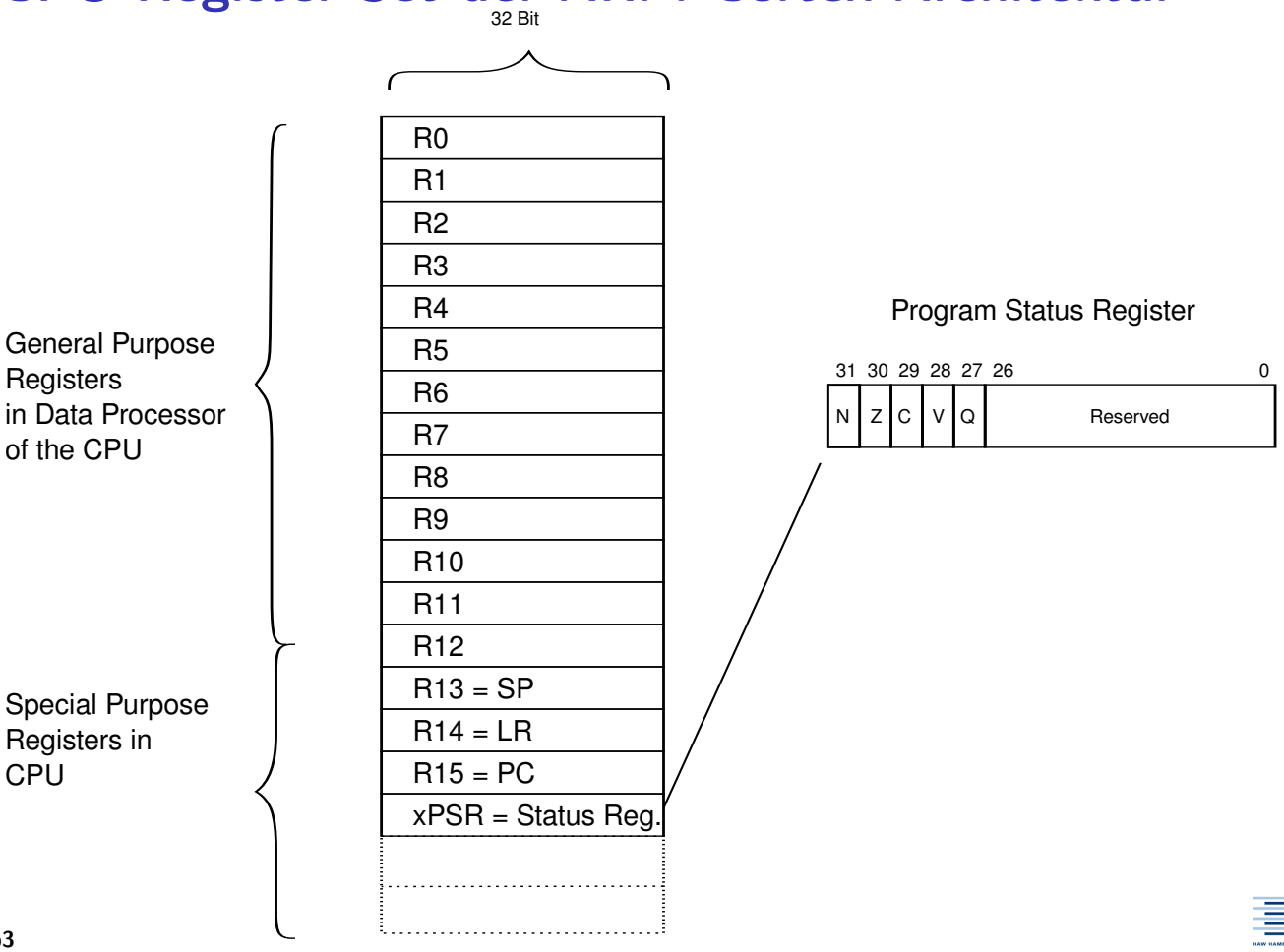
Control Unit und Data Processor



Die Operanden und das Resultat werden im General Purpose Registern des DP gespeichert.

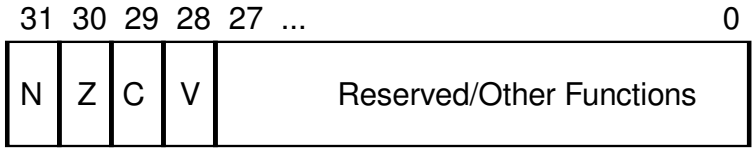
Die Statusinformation steht der CU zur Verfügung.

CPU Register Set der ARM Cortex Architektur



Statusregister mit einigen Flags

PSR= Program Status Register



Nach jedem Maschinen Befehl (Machine-Level-Instruction) wird das Statusregister aktualisiert

Bit Flag

31 Negative-Flag (Negative, negatives Ergebnis eines Vergleiches)

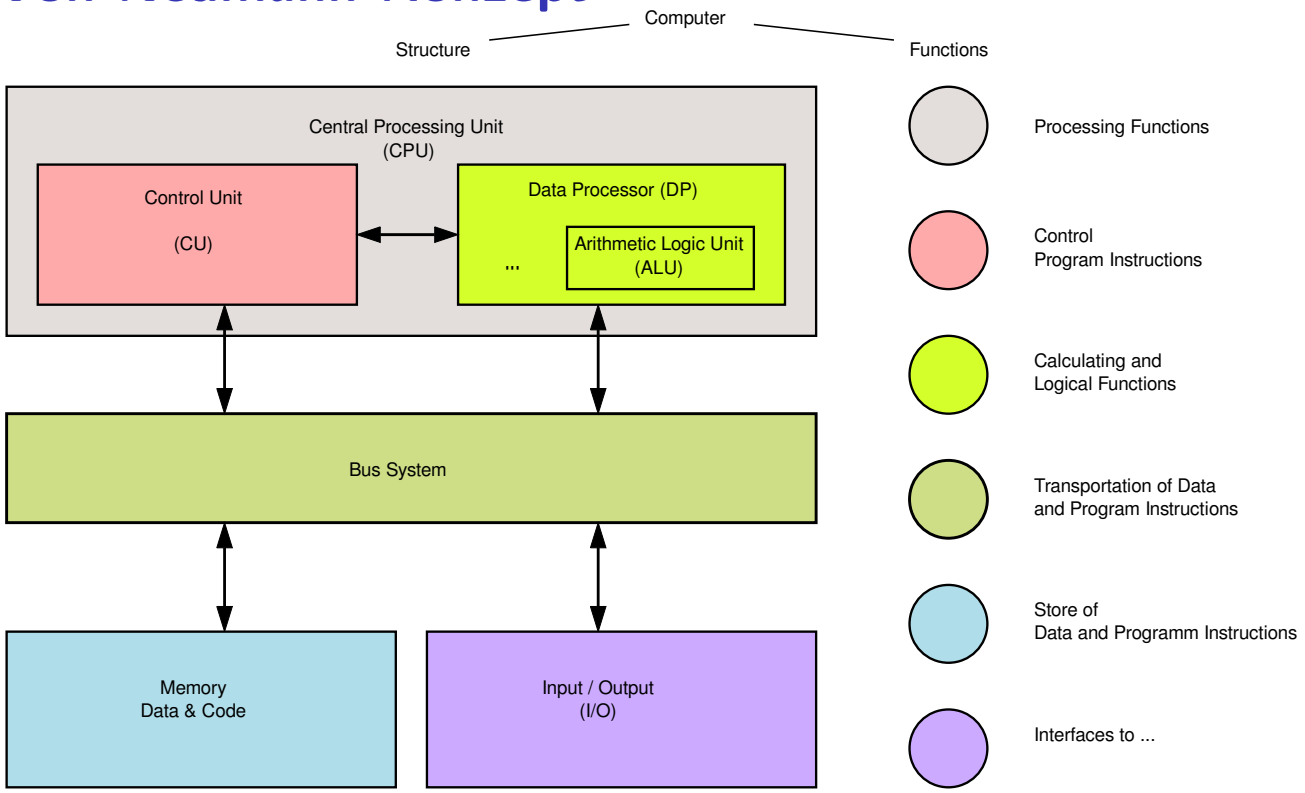
30 Zero-Flag (Zero, Ergebnis 0 einer arth./log. Operation)

29 Carry-Flag (Carry, vorzeichenloser Übertrag bei Addition,
kein Übertrag bei Subtraktion)

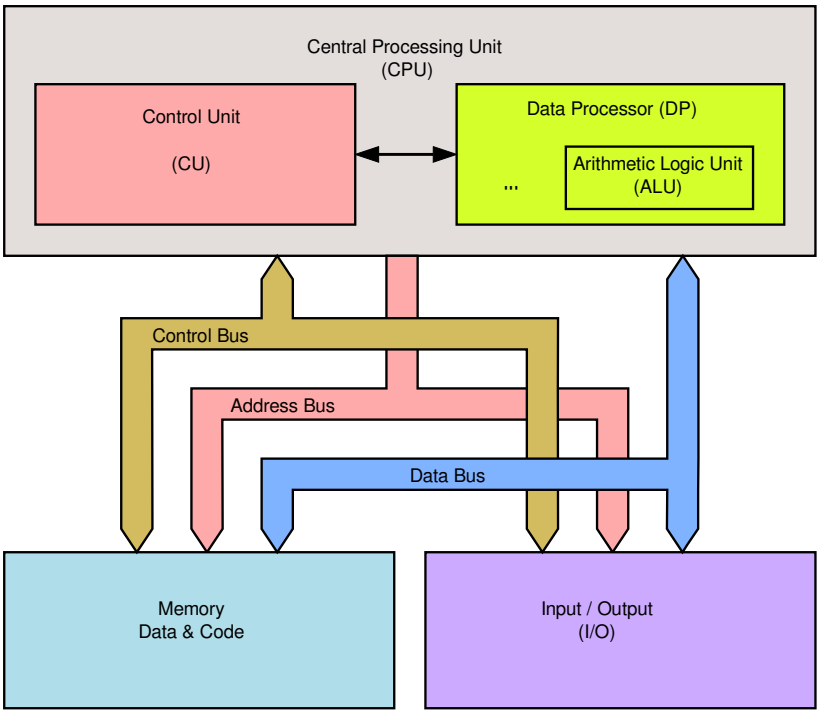
28 V-Flag (Overflow, vorzeichenbehafteter Überlauf

Hinweis: Carry/Übertrag kann zum "Weiterrechnen" (mit besonderen, langen Rechenfunktionen in SW) genutzt werden, Overflow/Überlauf nicht !

Bus-System als zentrale Schnittstelle im Von-Neumann-Konzept

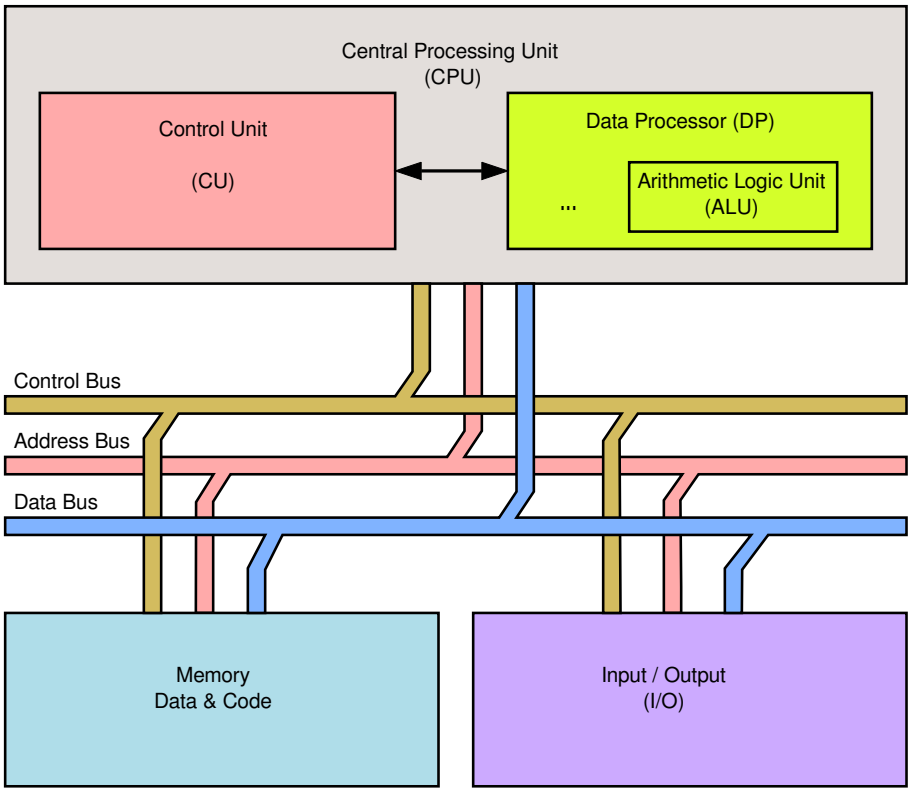


Informationsflüsse



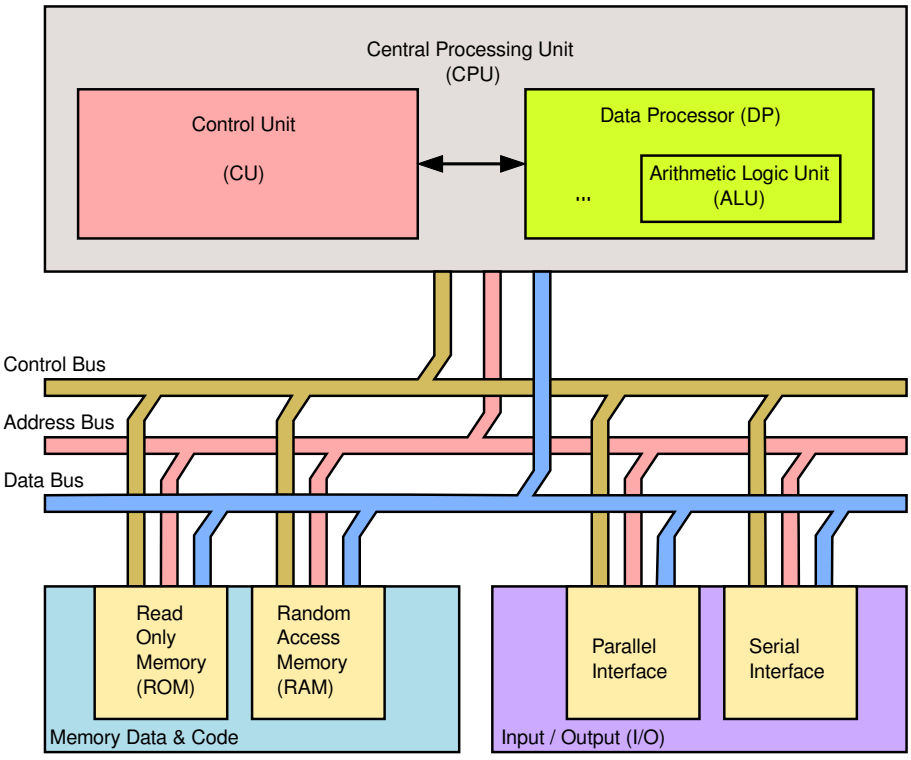
Data Bus: Informationen von der CPU ausgehend und zur CPU laufend
Address Bus: Addresinformationen unidirektional von CPU ausgehend
Control Bus: Zusammenfassung von Leitungen mit unabhängigen Aufgaben

Bus-Strukturen



Konzept des Busses: Alle Module sind an die Leitungen angeschlossen
Informationen werden von der Quelle an alle Module des System verteilt
jedoch dort nur genutzt, wenn ein Modul als Ziel aktiviert wird

Speicher-Module und I/O-Einheiten

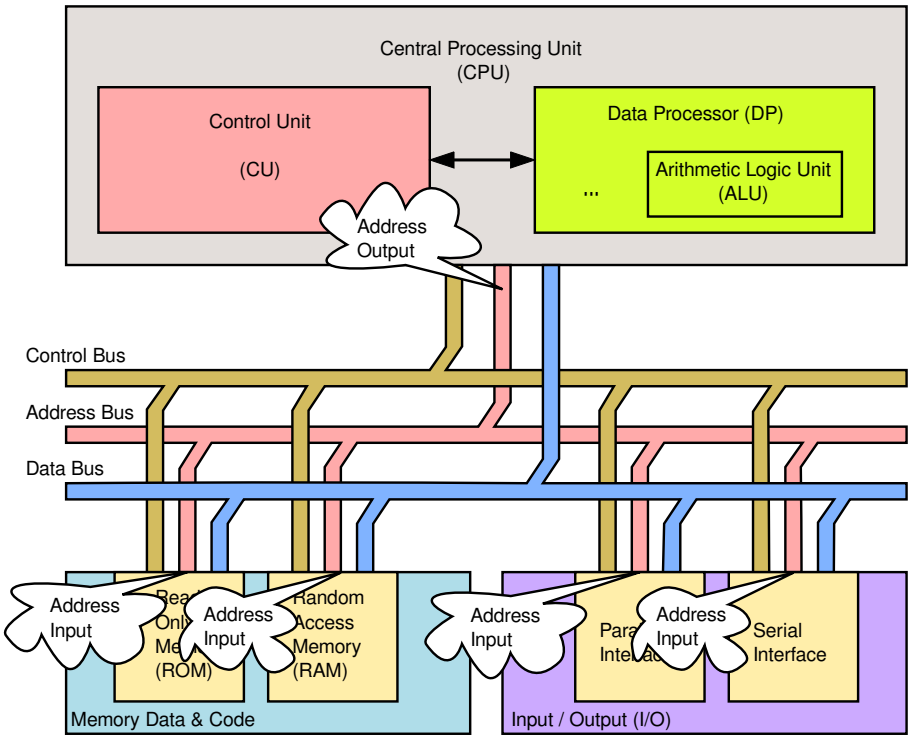


Adressbus

- Eine Adresse besteht aus 32 Bit und adressiert ein Byte. [1]
- Namen der Adressleitungen A31 bis A0
- Der Adressbus wird in nur einer Richtung genutzt:
- Richtung: von der CPU
  zum Speichermodul x
  oder
  zum I/O-Modul
- Adressiert werden Speicherzellen oder Peripherieregister
- Nach einer gesendeten Adresse werden vom Datenbus 32 Bit von dieser Adresse geliefert (Memory-Read).
- Nach einer gesendeten Adresse werden an den Datenbus 32 Bit geschrieben, die an dieser Adresse gespeichert werden (Memory-Write).

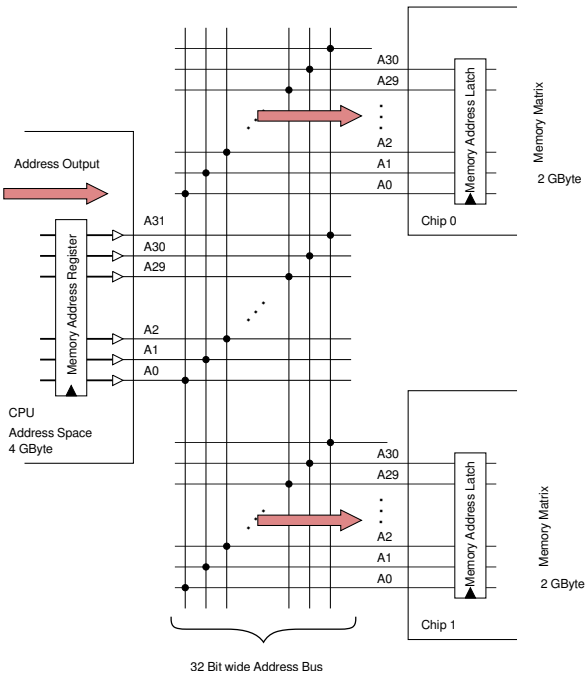
[1] Alle Angaben beziehen sich hier auf ARM-Cortex-Systeme

Adressen werden an alle Module gesendet



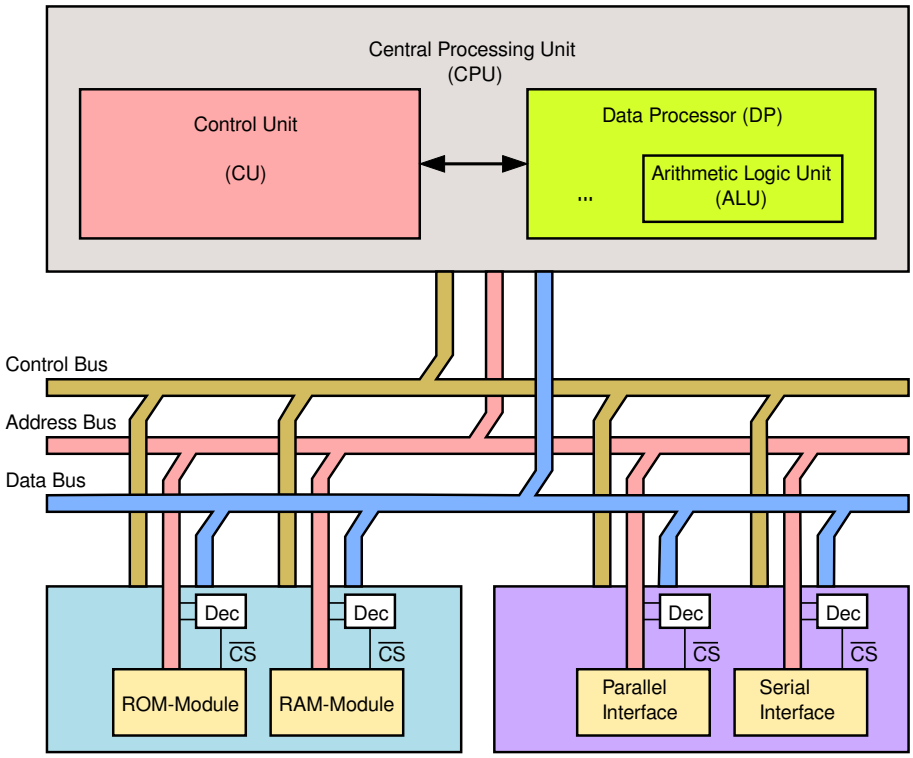
Eine Adresse wird im Broadcast von der CPU an den Bus gesendet. Alle Module empfangen die Adresse, aber stets nur ein Modul (bzw. eine Speicherstelle oder Register) wird adressier

Der Adressbus verbindet CPU und Speicherchips



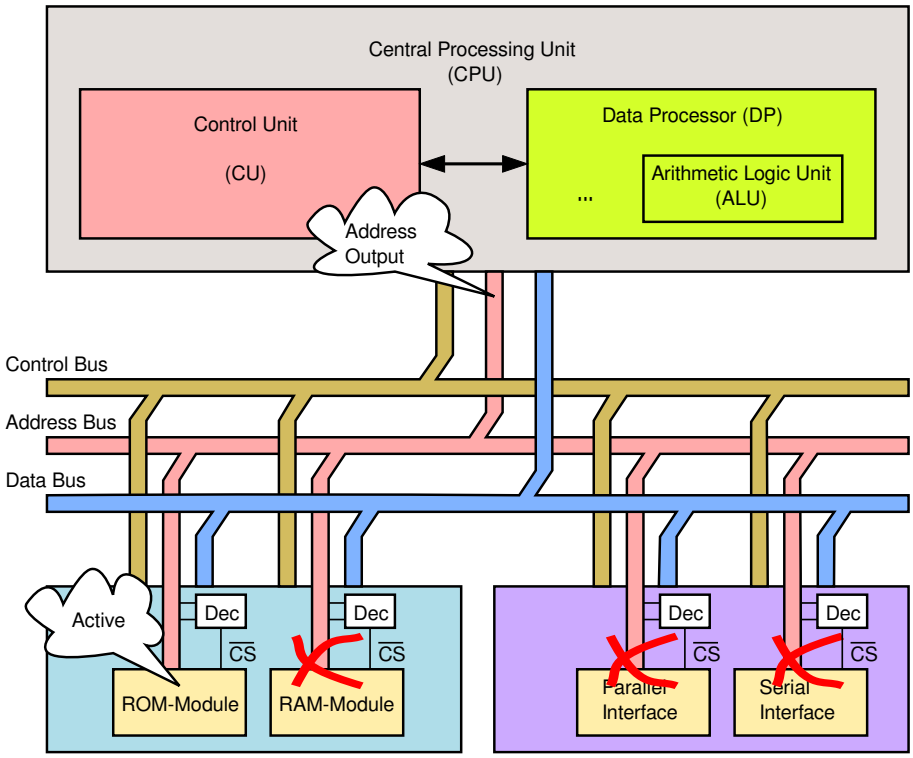
Der Adressbus besteht aus parallelen Leitungen von der CPU zu den Speicherchips

Decodieren der Adressen aktiviert Module

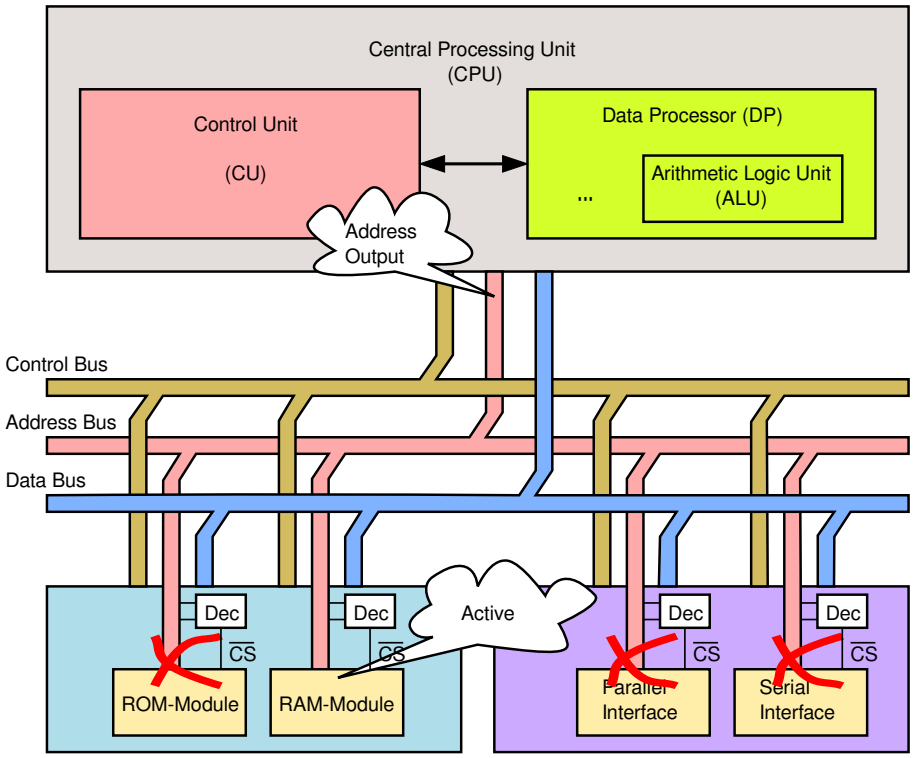


Adressdecoder entscheiden über die Aktivierung der Module anhand der Adresse oder eines Teil des Adresse davon.

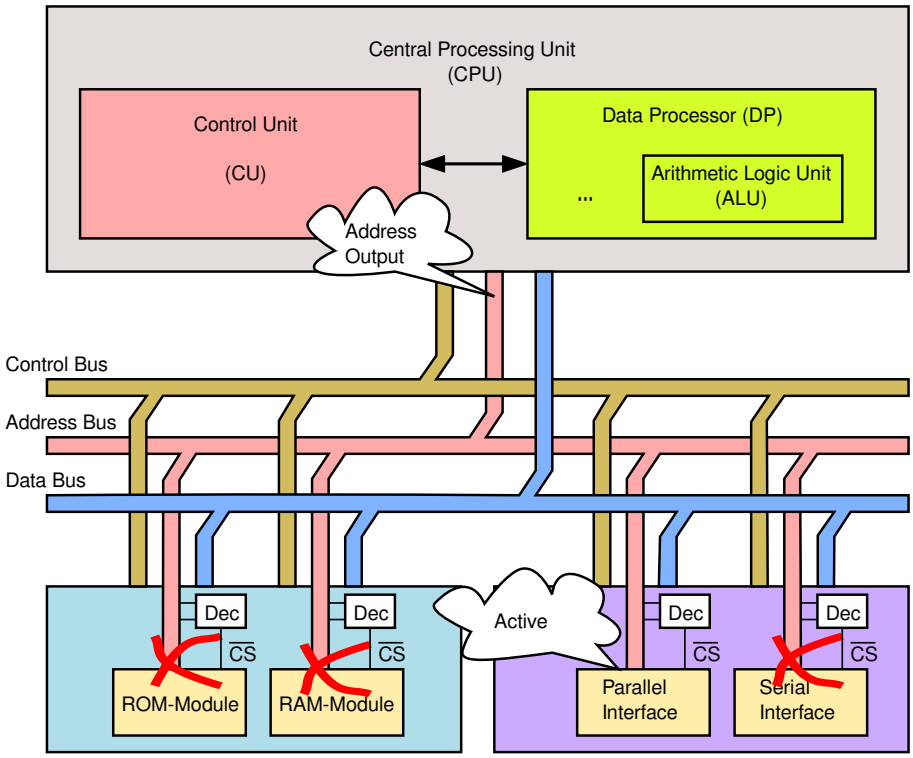
Adressdecoder entscheiden über die Auswahl (1)



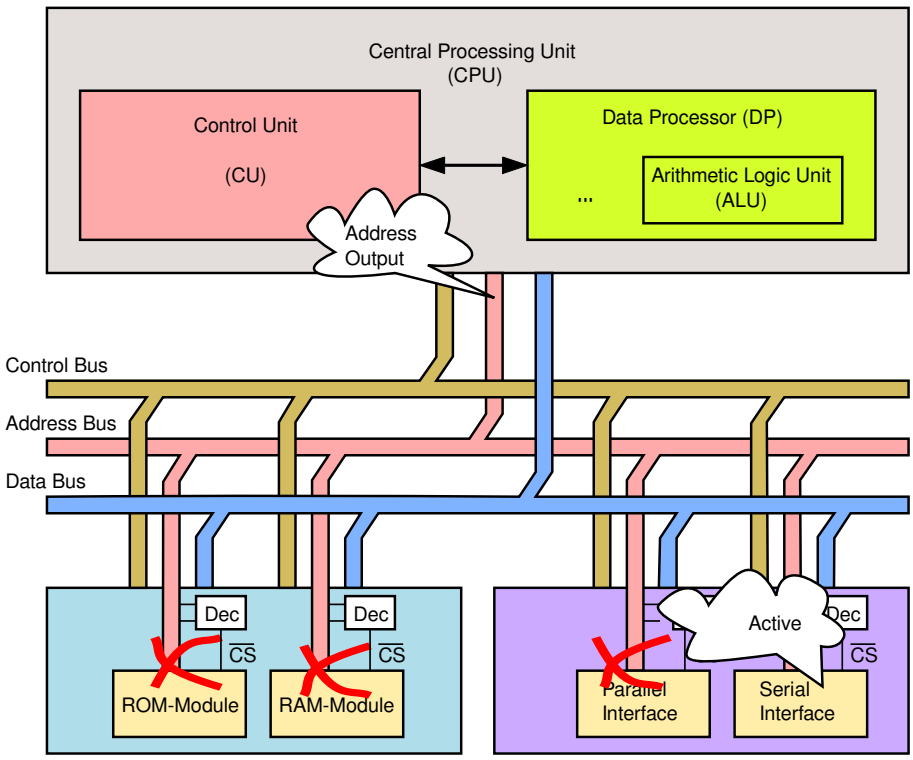
Adressdecoder entscheiden über die Auswahl (2)



Adressdecoder entscheiden über die Auswahl (3)



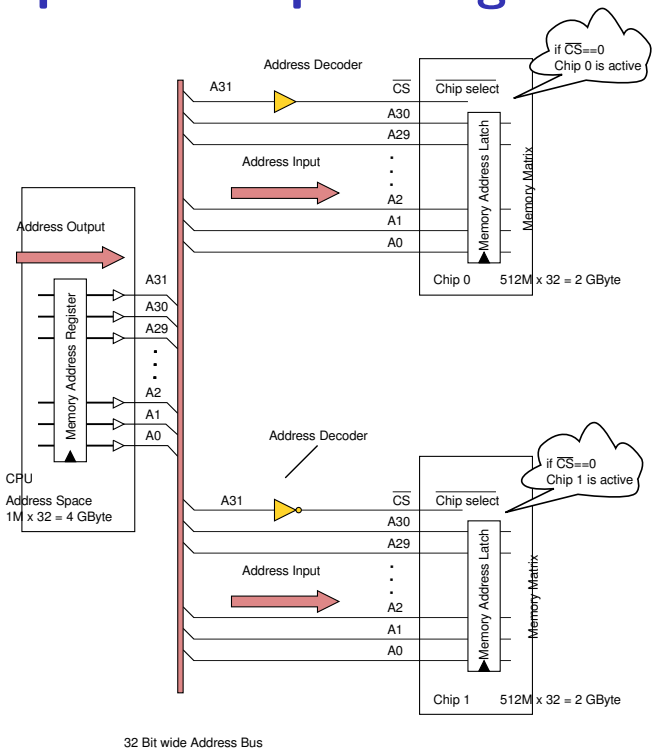
Adressdecoder entscheiden über die Auswahl (4)



Beispiel 1: CPU und zwei Speicherchips

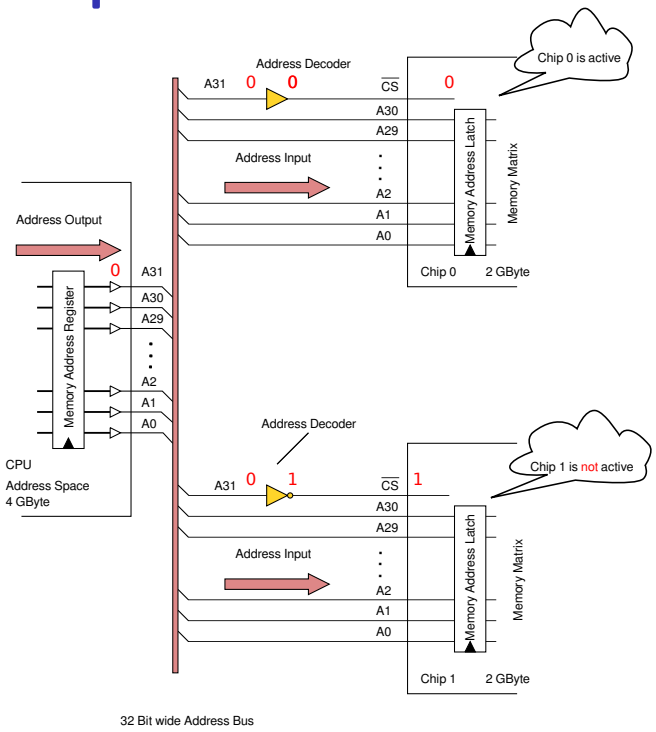
- CPU gibt Adressen mit 32 Adressbits aus:
  - $\rightsquigarrow 2^{32}$ Speicherplätze $\curvearrowright$ 4 G Address Space
  - $\rightsquigarrow 2^{32}$ Speicherplätze je 1 Byte $\curvearrowright$ 4 GB max. Kapazität möglich
- Speicherchip 0 hat Adressleitungen mit 31 Adressbits:
  - $\rightsquigarrow 2^{31}$ Speicherplätzee$\curvearrowright$ 2 G Address Space
  - $\rightsquigarrow 2^{31}$ Speicherplätze je 1 Byte $\curvearrowright$ 2 GB Kapazität
- Speicherchip 1 hat Adressleitungen für 31 Adressbits:
  - $\rightsquigarrow 2^{31}$ Speicherplätze $\curvearrowright$ 2 G Address Space
  - $\rightsquigarrow 2^{31}$ Speicherplätze je 1 Byte $\curvearrowright$ 2 GB Kapazität

Beispiel 1:
Auswahl der Speicherchips erfolgt mit A31



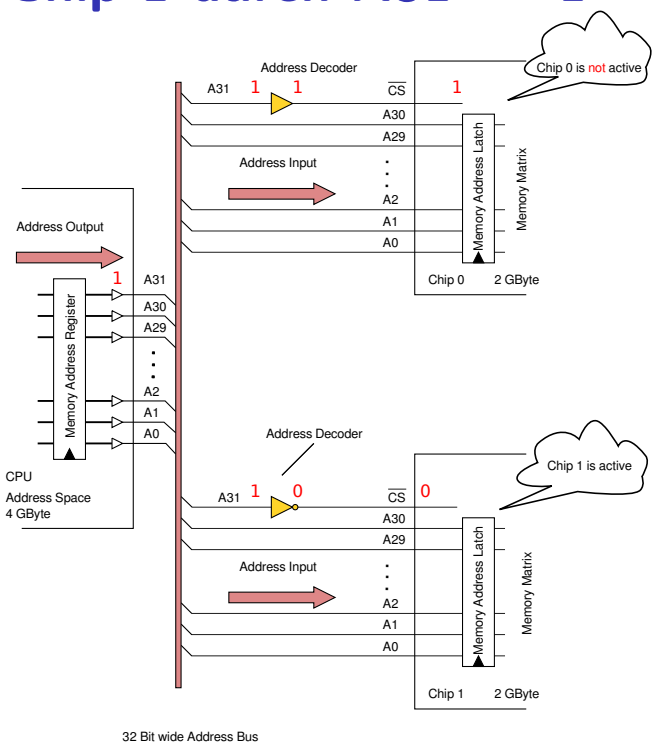
Die Auswahl erfolgt mit der "freien"Leitung A31, welche nicht an die Adress-Eingänge der Chips angeschlossen ist, weil die Chips jeweils nur 31 Adress-Eingänge besitzen.

Beispiel 1:
Auswahl des Chip 0 durch A31 = '0'



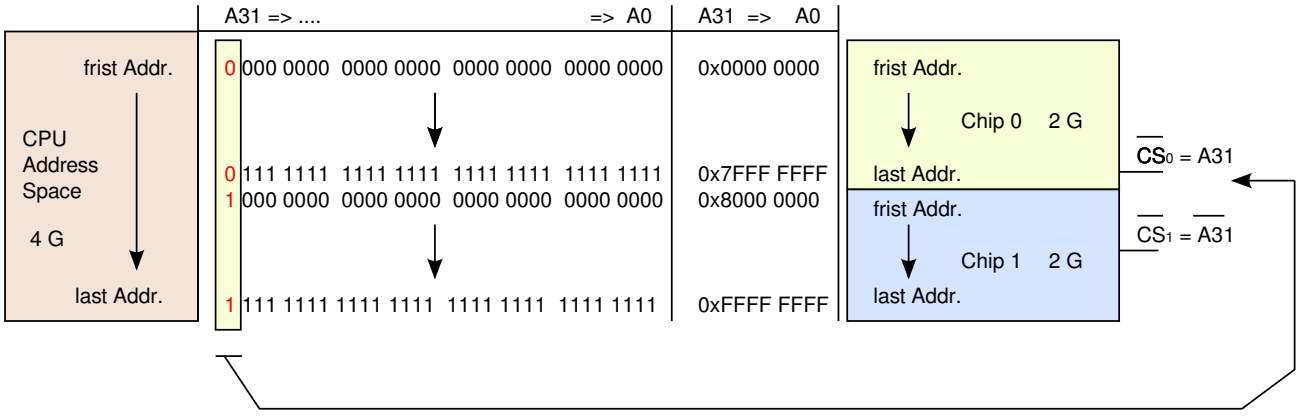
Die Auswahl erfolgt Adressleitung A31. Hat die Stelle A31 den Wert 0 in der Adresse, dann wird $\overline{CS_0}$ auf 0 gesetzt und der Chip 0 selektiert.

Beispiel 1:
Auswahl des Chip 1 durch A31 = '1'



Die Auswahl erfolgt Adressleitung A31. Hat die Stelle A31 den Wert 1 in der Adresse, dann wird $\overline{CS_1}$ auf 0 gesetzt und der Chip 1 selektiert.

Adressräume CPU und Speicherchips

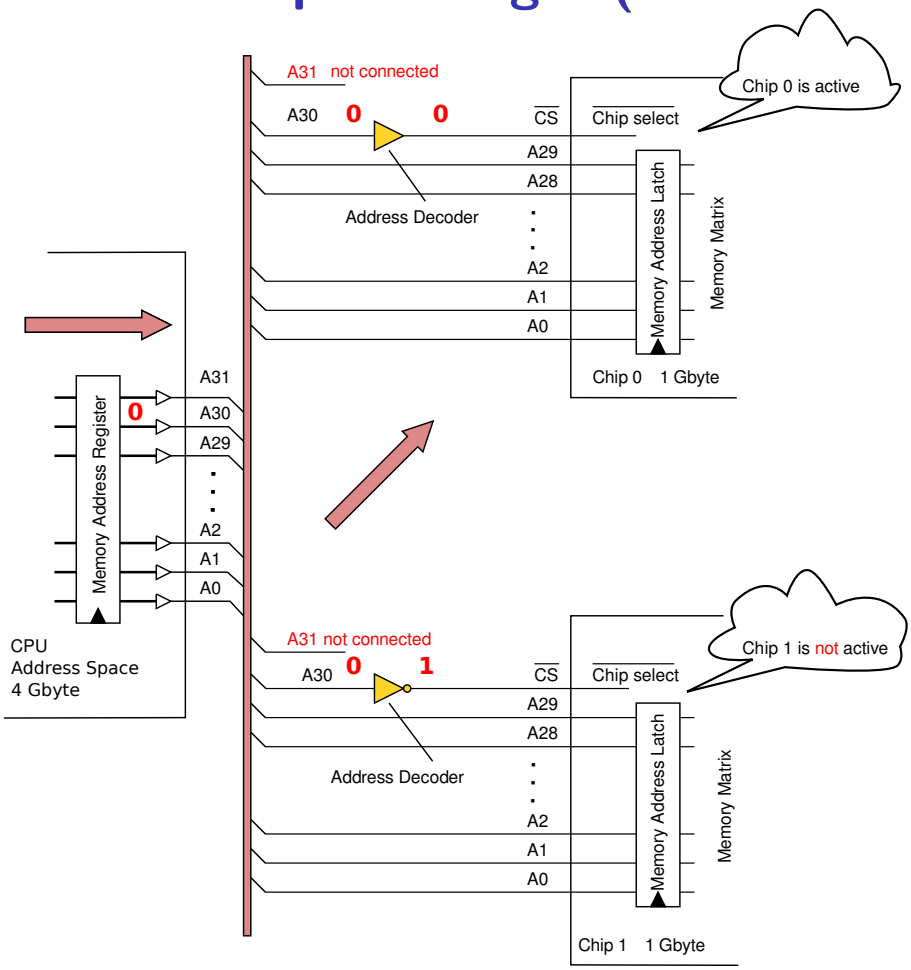


Die Adressdecoder entscheiden über die Aktivierung der Module anhand des "freien Teils" der Adresse. Meist wird "negative Logik" für das Chip-Select Signal verwendet, dies Chips werden mit dem Wert 0 selektiert. Man schreibt dann $\overline{CS_0}$, $\overline{CS_1}$ usw. .

Beispiel 2: CPU und zwei Speicherchips

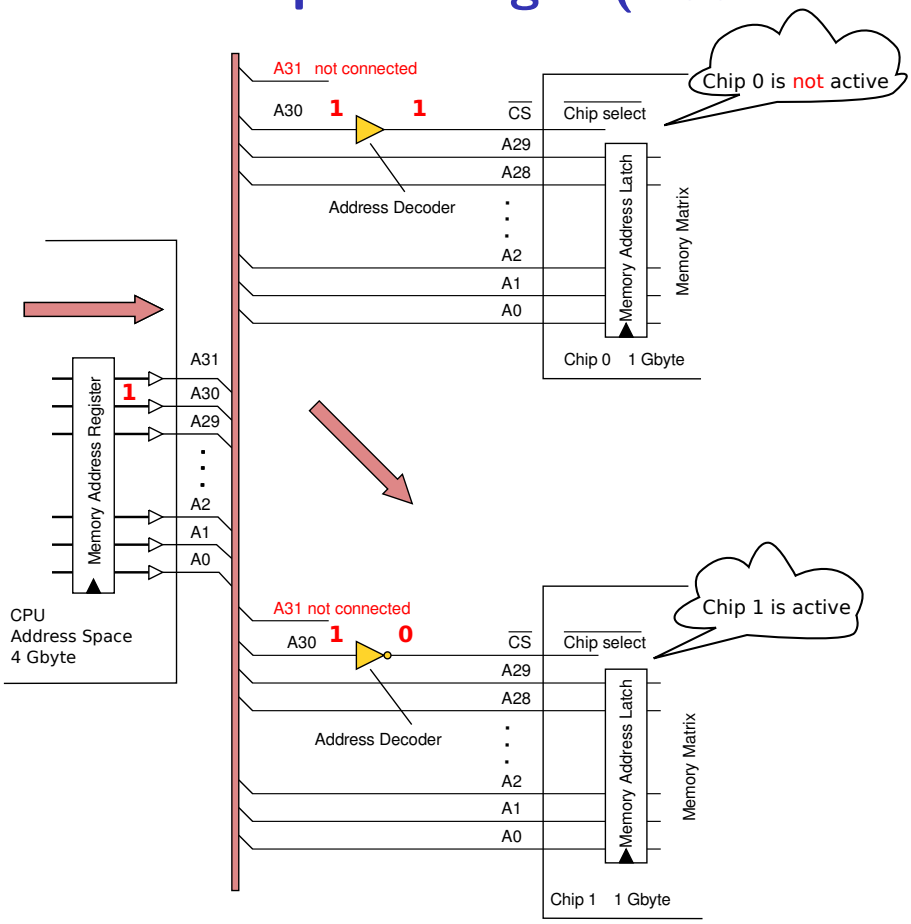
- CPU gibt Adressen mit 32 Adressbits aus:
  $\rightsquigarrow 2^{32}$ Speicherplätze $\curvearrowright$ 4 G Address Space
- Speicherchip 0 hat Adressleitungen mit 30 Adressbits:
  $\rightsquigarrow 2^{30}$ Speicherplätze $\curvearrowright$ 1 G Address Space
- Speicherchip 1 hat Adressleitungen für 30 Adressbits:
  $\rightsquigarrow 2^{30}$ Speicherplätze $\curvearrowright$ 1 G Address Space

Adressdecoder Beispiellösung 1 (A30 selektiert)



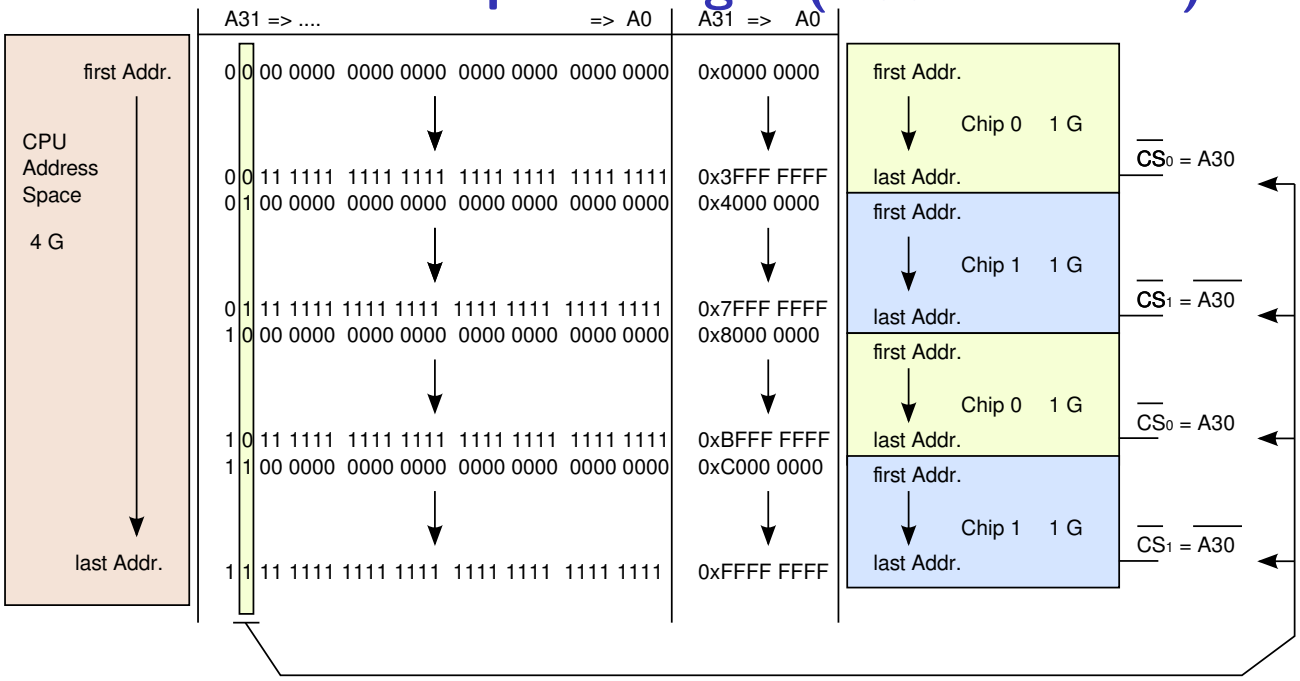
32 Bit wide Address Bus

Adressdecoder Beispiellösung 1 (A30 selektiert)



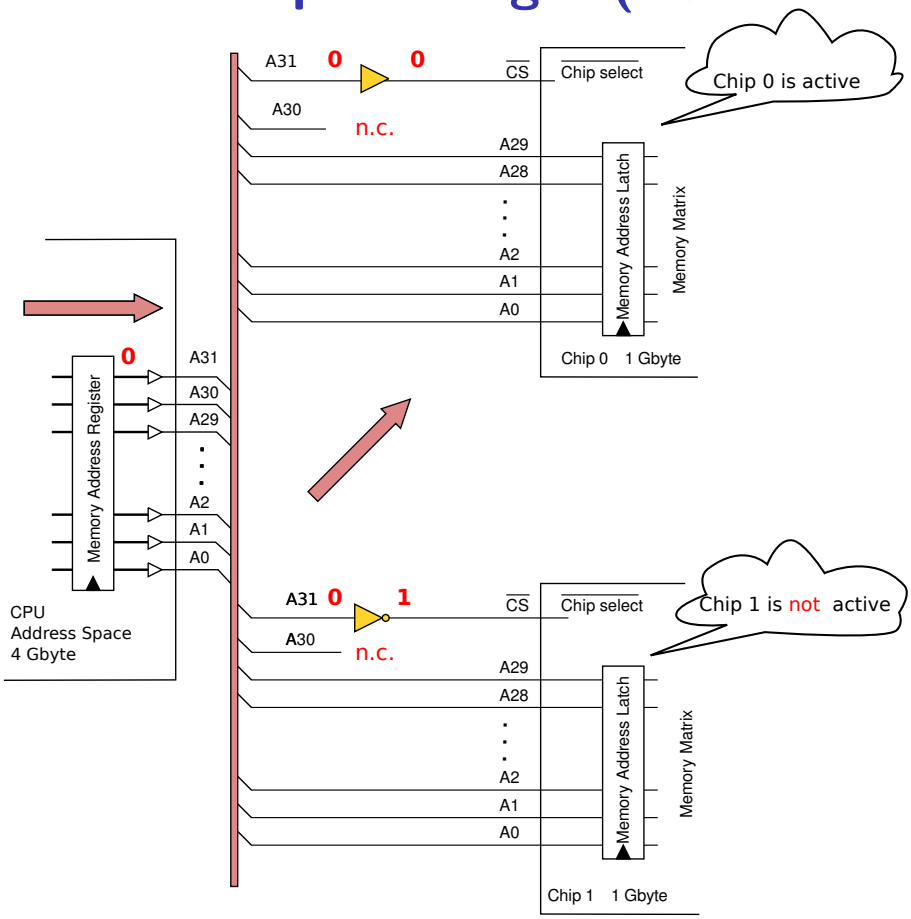
32 Bit wide Address Bus

Adressdecoder Beispiellösung 1 (A30 selektiert)

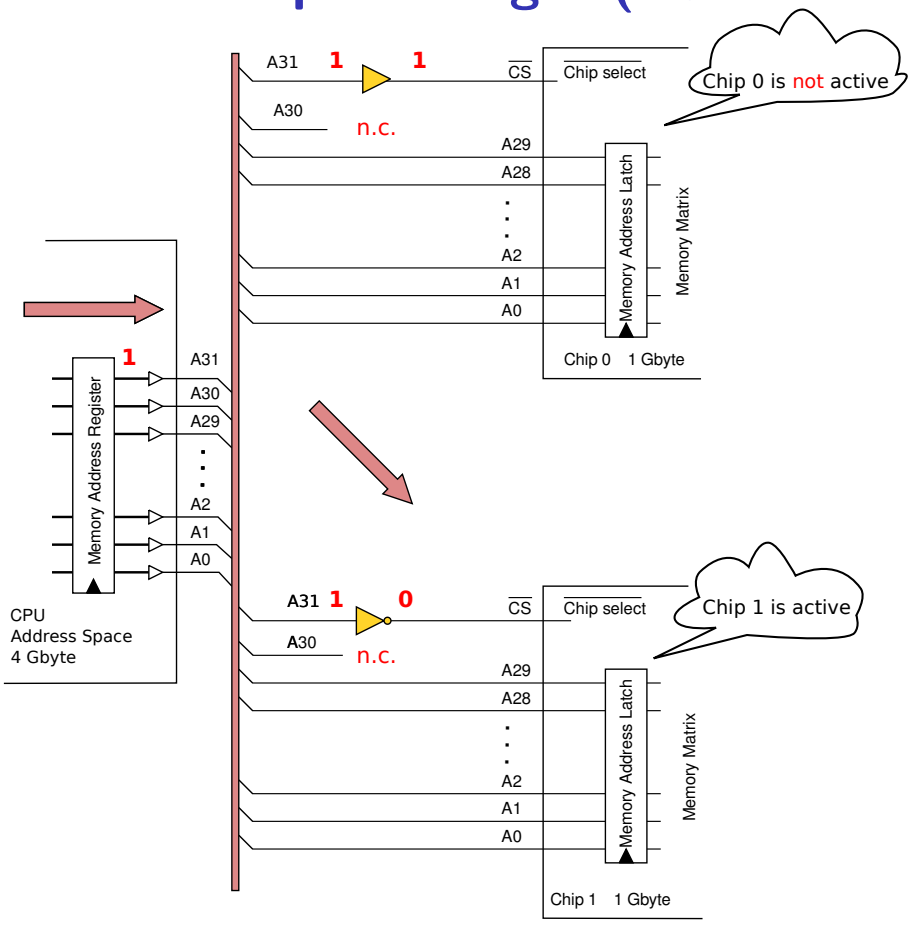


Durch die unvollständige Decodierung (A31 wird nicht einbezogen) entstehen 'Spiegel' der Chips im Adressraum. Der Effekt wird Mirroring genannt. Damit wird eine Speicherstelle unter mehreren Adressen erreichbar.

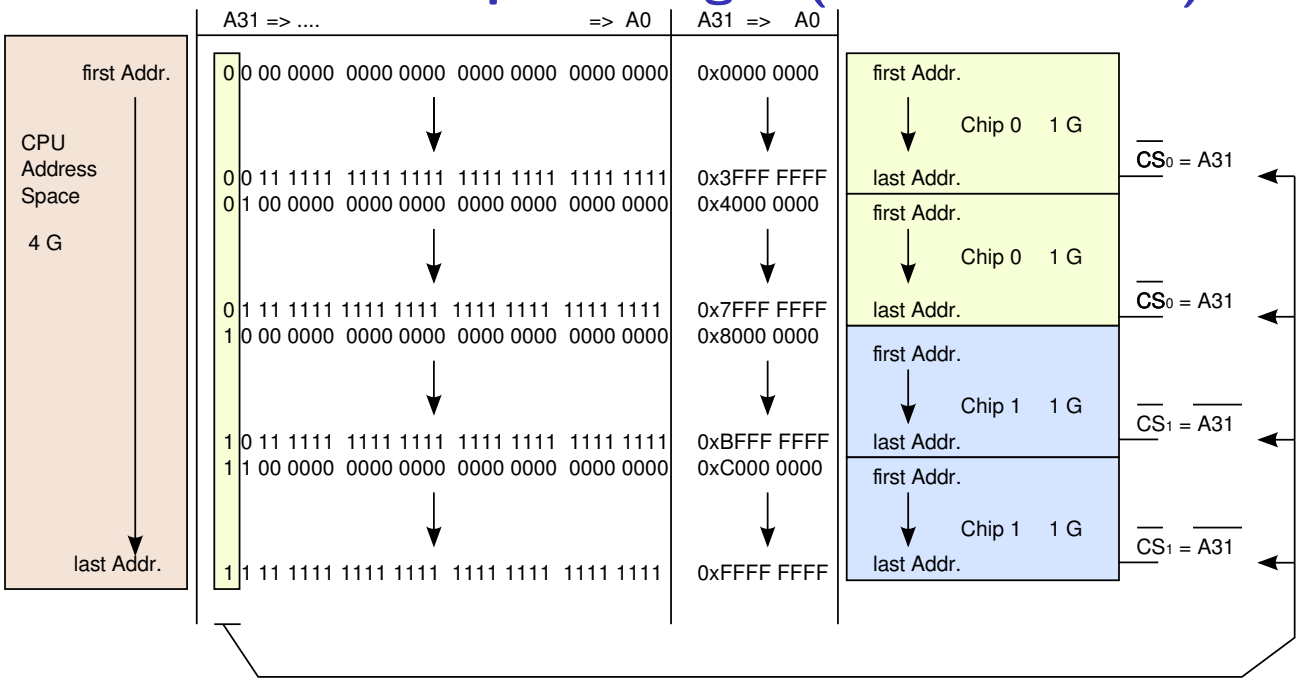
Adressdecoder Beispiellösung 2 (A31 selektiert)



Adressdecoder Beispiellösung 2 (A31 selektiert)

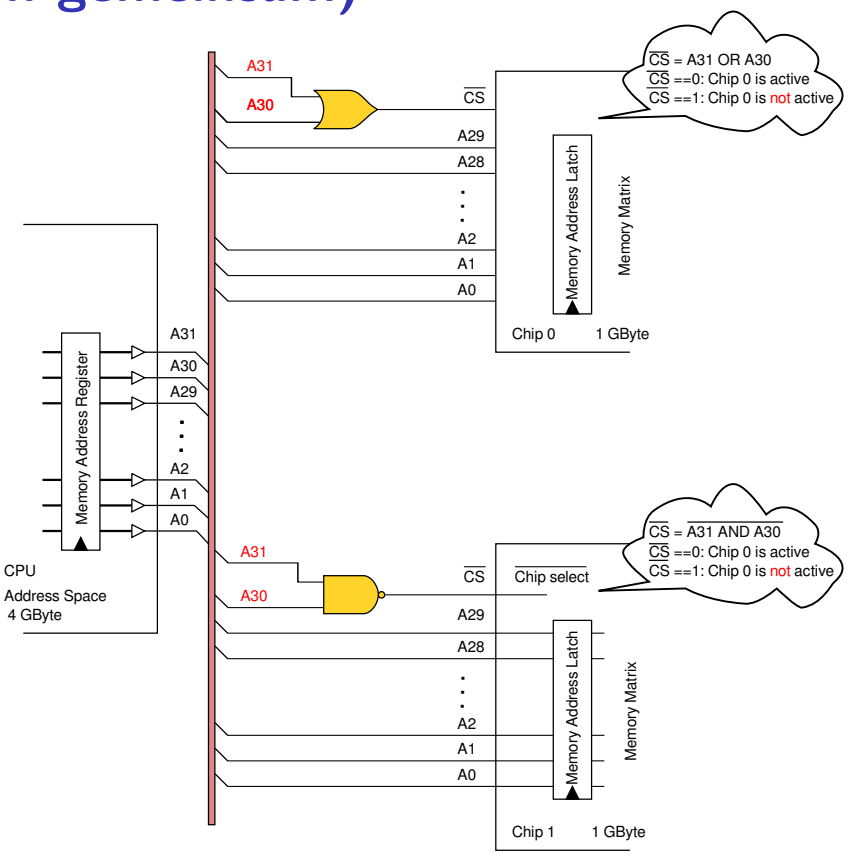


Adressdecoder Beispiellösung 2 (A31 selektiert)

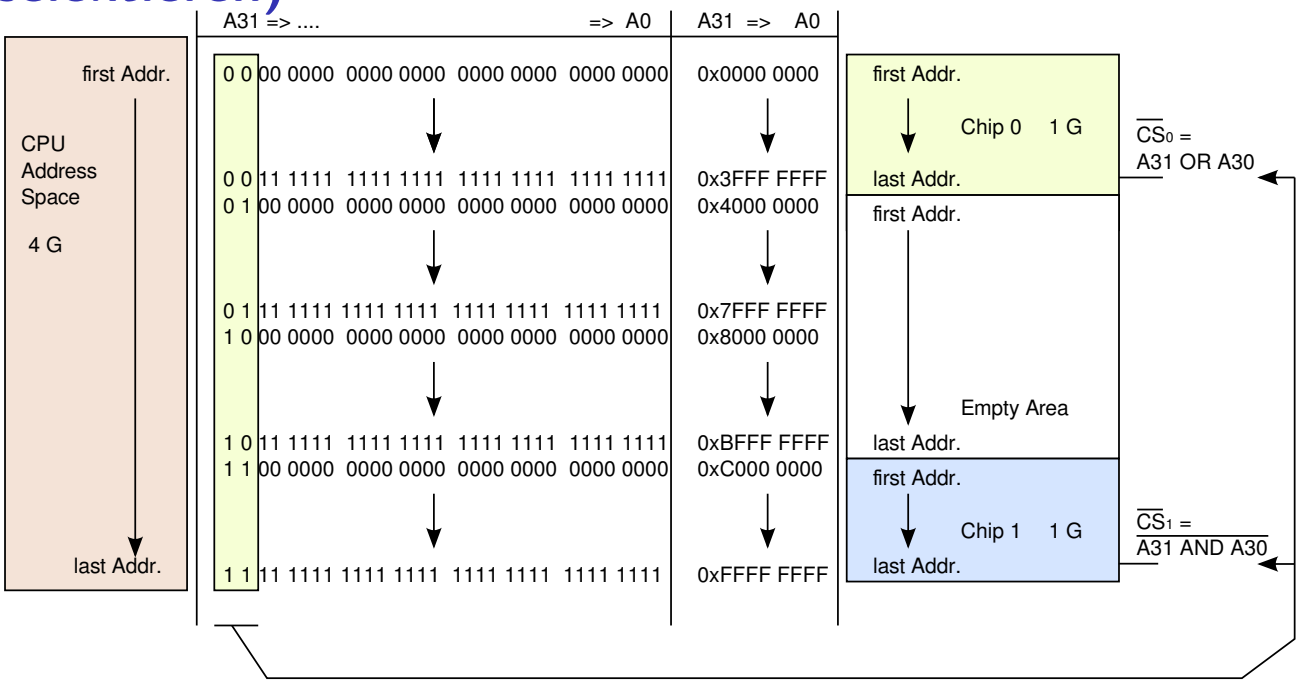


Auch durch diese unvollständige Decodierung (A30 wird nicht einbezogen) entstehen wieder 'Spiegel' der Chips im Adressraum.

Adressdecoder Beispiellösung 3 (A31 und A30 selektieren gemeinsam)

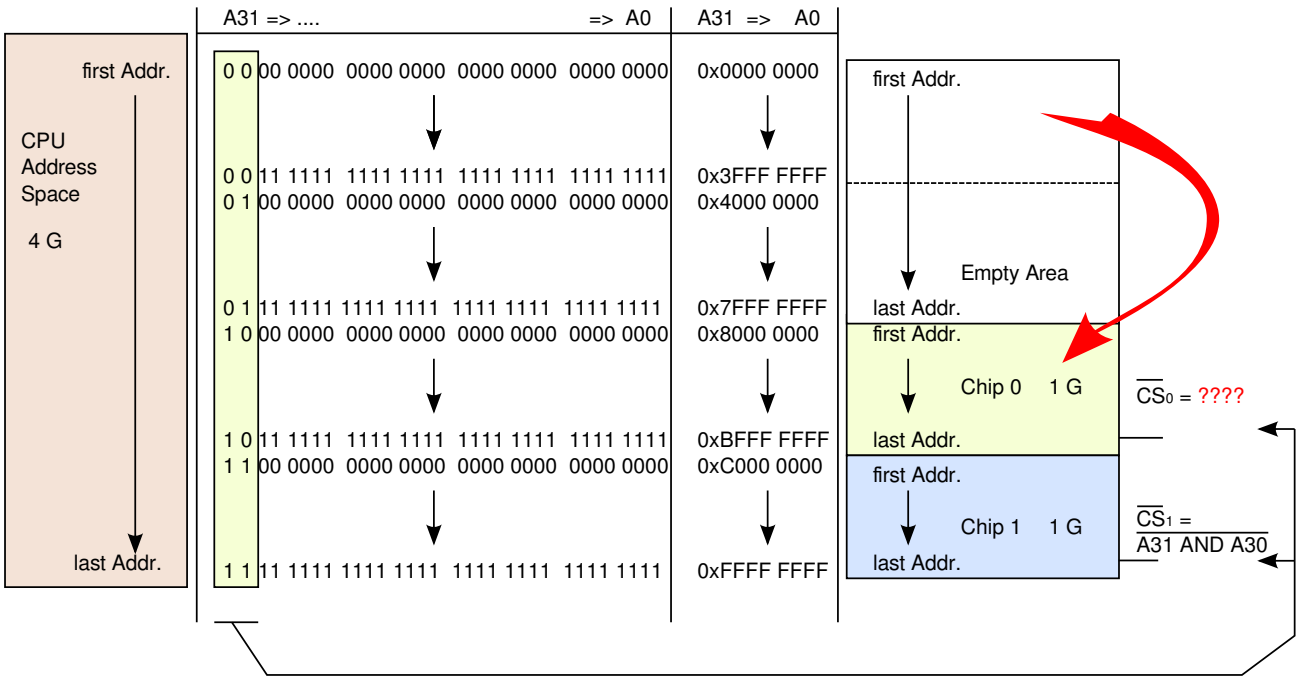


Adressdecoder Beispiellösung 1 (A31 und A30 selektieren)



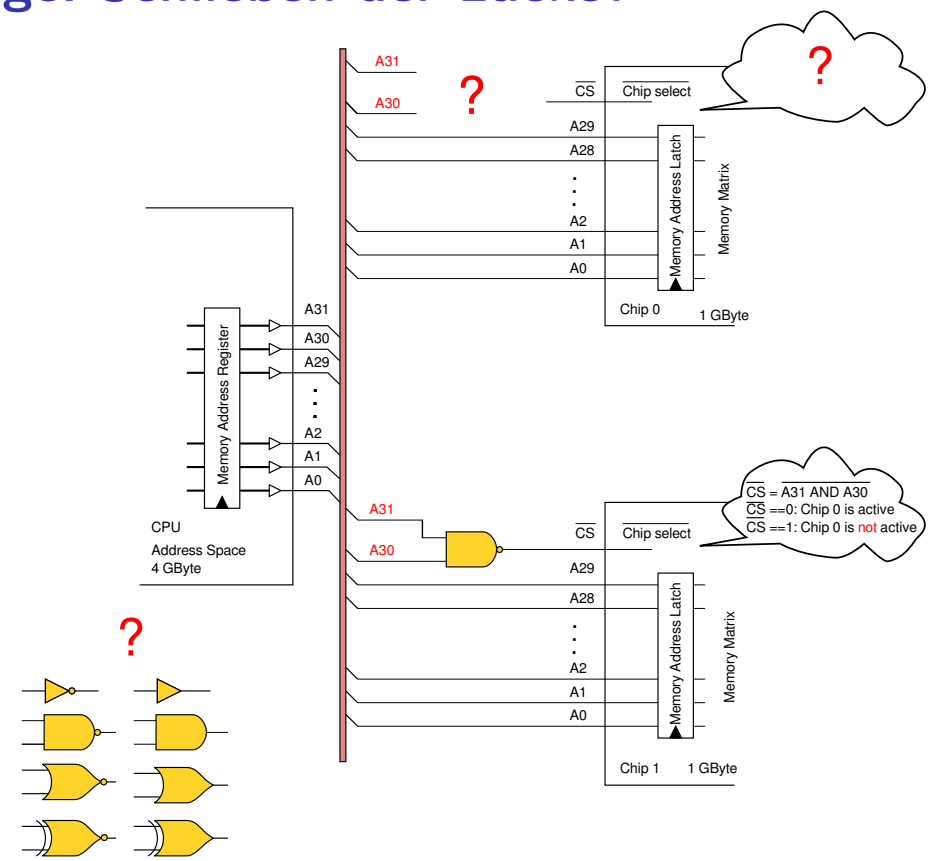
Alle Adressbits werden genutzt: A31 und A30 für die Decoder und der Rest für Adresseingänge die Chips. Durch diese vollständige Decodierung entstehen keine 'Spiegel' der Chips im Adressraum. Jede Speicherstelle hat nur eine Adresse, aber es gibt unbelegte Lücken.

Quizfrage: Schließen der Lücke in der Addressmap ?



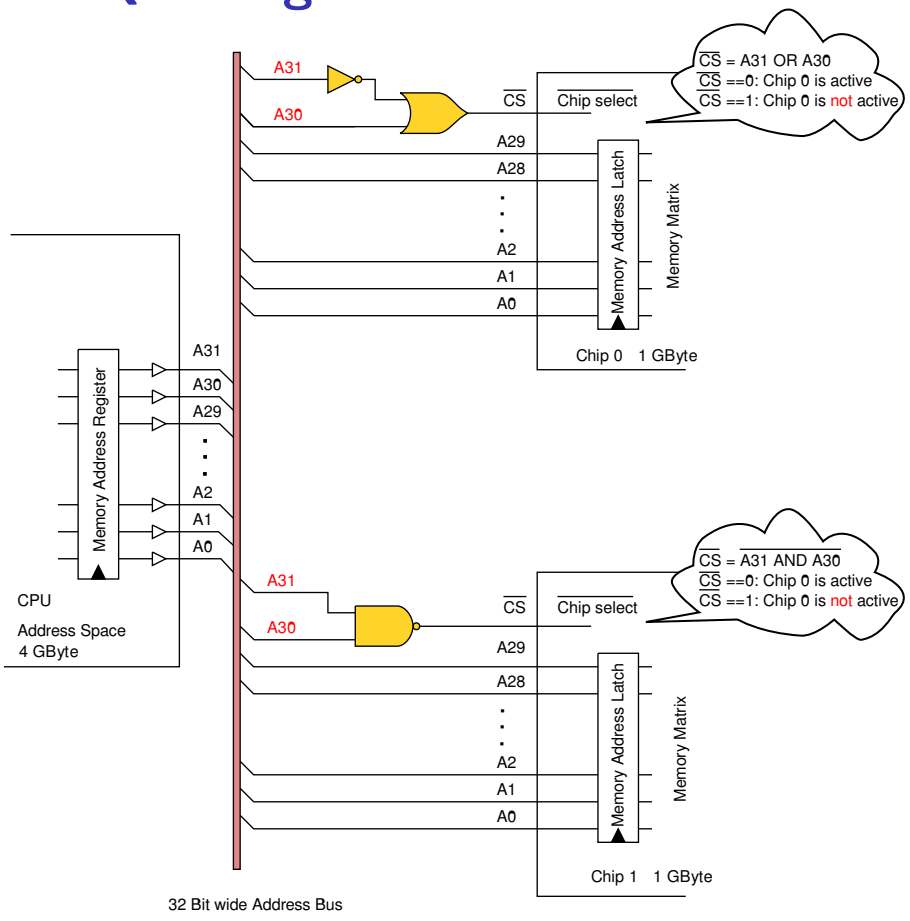
Aufgabe: "Verschieben"der Position des Chip 0 bis an den Chip 1, damit die Lücke in der Adressbelegung geschlossen wird.

Quizfrage: Schließen der Lücke?

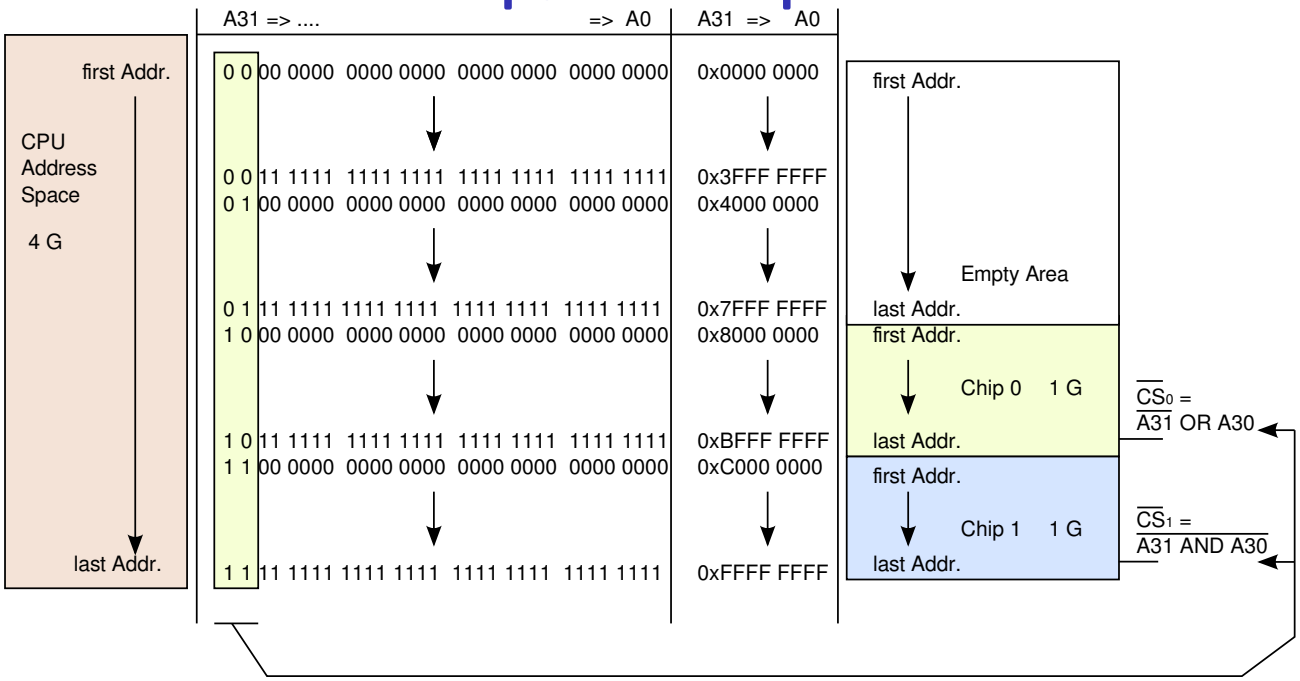


Nutzen Sie einige der links unten 'bereitgestellten' Gatter für einen Adressdecoder.

Lösung der Quizfrage



Adressdecoder: Chip0 und Chip2 ohne Lücke



Die Lücke ist nun an Anfang des Adressraums verschoben. Die Chips befinden sich am Ende. Das ist ungünstig für die Software, weil die meisten Systeme dort bestimmte Programmteile erwarten. Überlegen Sie bitte selbstständig, wie Sie die Chips an den Anfang und Lücke an das Ende des Adressraums bringen können,

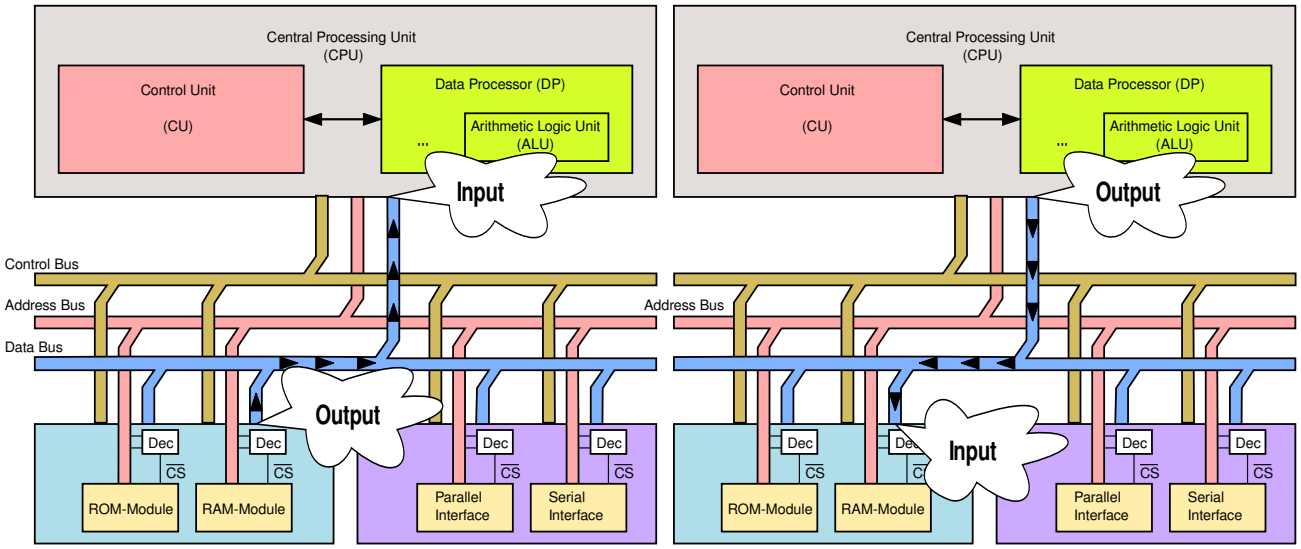
Datenbus

- Namen der Datenleitungen D31 bis D0 [2]
- Der Datenbus wird in zwei Richtungen genutzt (bidirektional):
  Diese Richtung wird umgeschaltet:

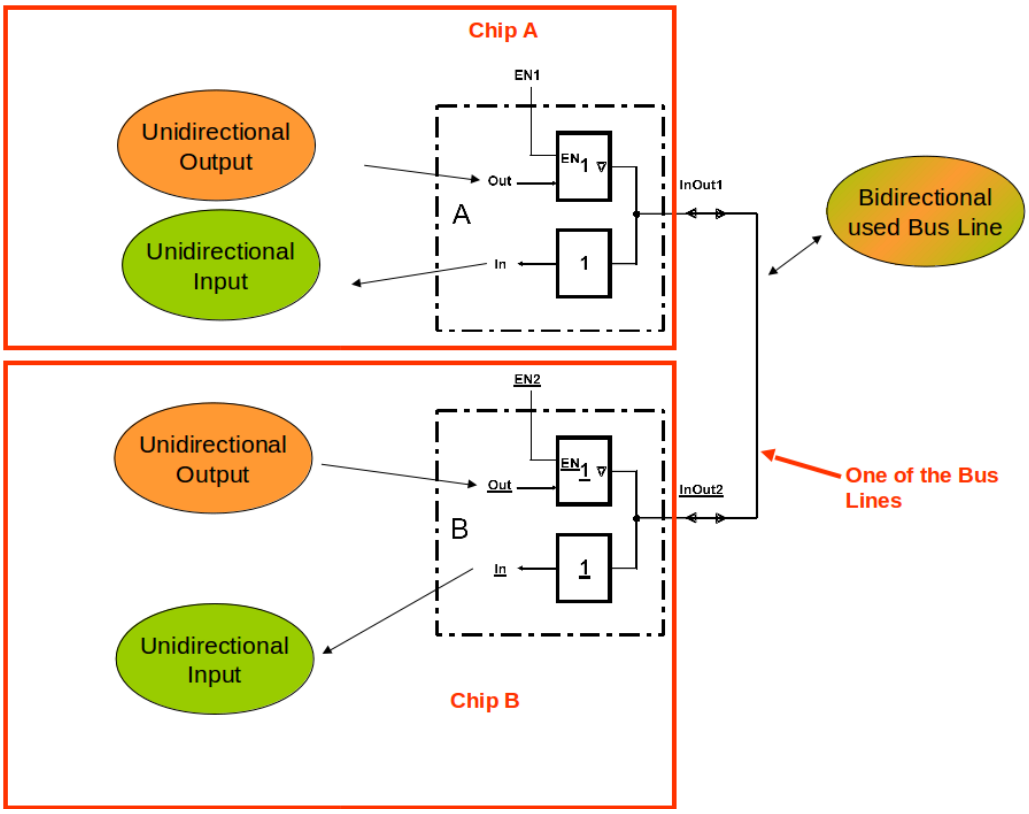
1. Richtung "Write" von der CPU
   $\Longrightarrow$ zum Speichermodul x
   oder
   $\Longrightarrow$ zum I/O-Modul
2. Richtung "Read"
   vom Speichermodul x
   oder
   vom I/O-Modul
   $\Longrightarrow$ zur CPU

[2] Alle Angaben beziehen sich hier auf ARM-Cortex-Systeme mit einer Datenbusbreite von 32 Bit.

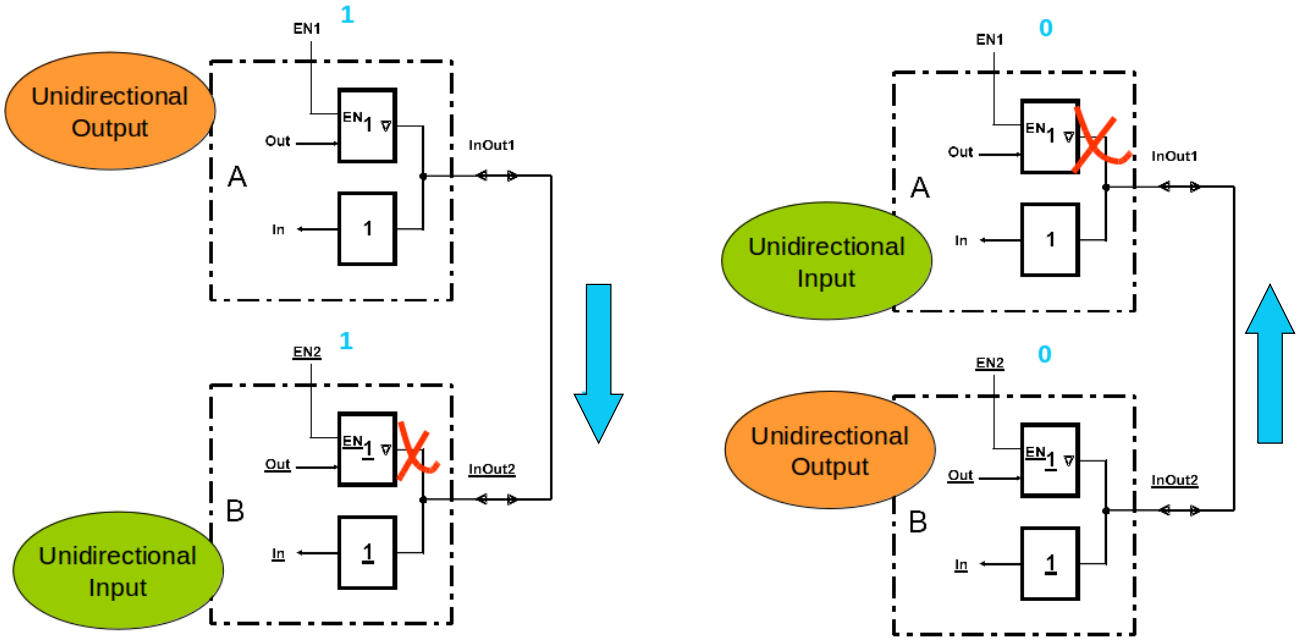
Der Datenbus wird bidirektional benutzt



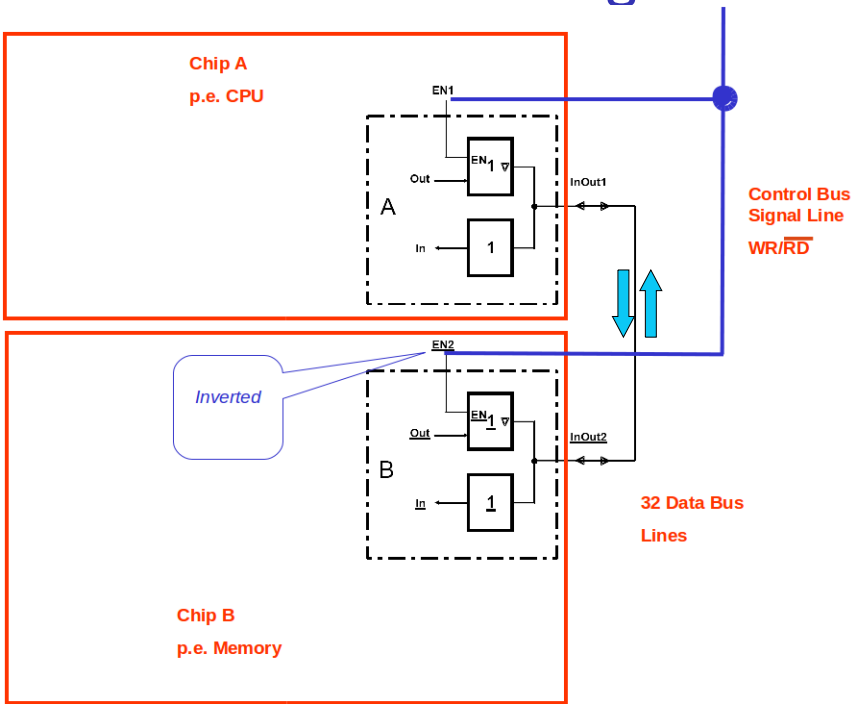
Signalrichtungen auf Leitungen und Bussen: unidirektional und bidirektional



Tristate Bustreiber am Interface zwischen Chip/Module und dem Datenbus



Die Richtung des Datenbus wird durch genau eine Leitung auf dem Controlbus umgeschaltet



Lese/Schreib-Umschaltung durch die CPU auf der Leitung des Controlbus $WR/\overline{RD}$ (oder bei anderen Typen $\overline{WR}/RD$)