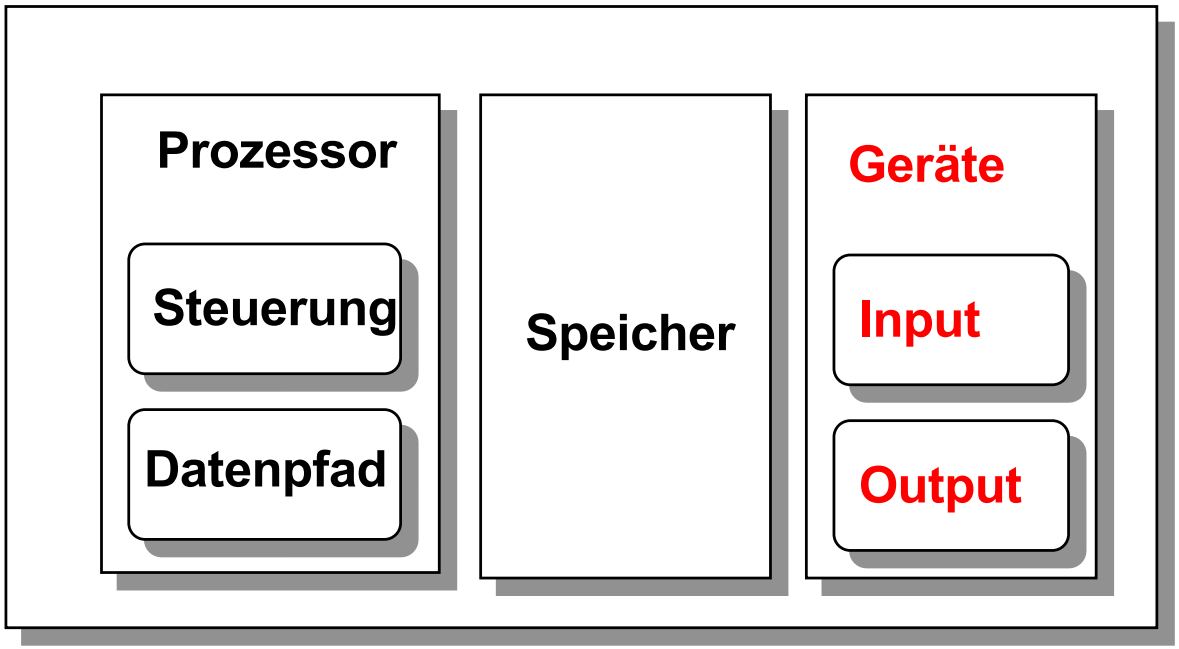
I/O Systems

[Adapted from Mary Jane Irwin for
Computer Organization and Design,
Patterson & Hennessy, © 2005, UCB]

Review: Hauptkomponenten eines Computer



- Wichtige Metriken für ein I/O System
  - Performance
  - Erweiterbarkeit
  - Zuverlässigkeit
  - Kosten, Grösse, Gewicht

Input und Output Geräte

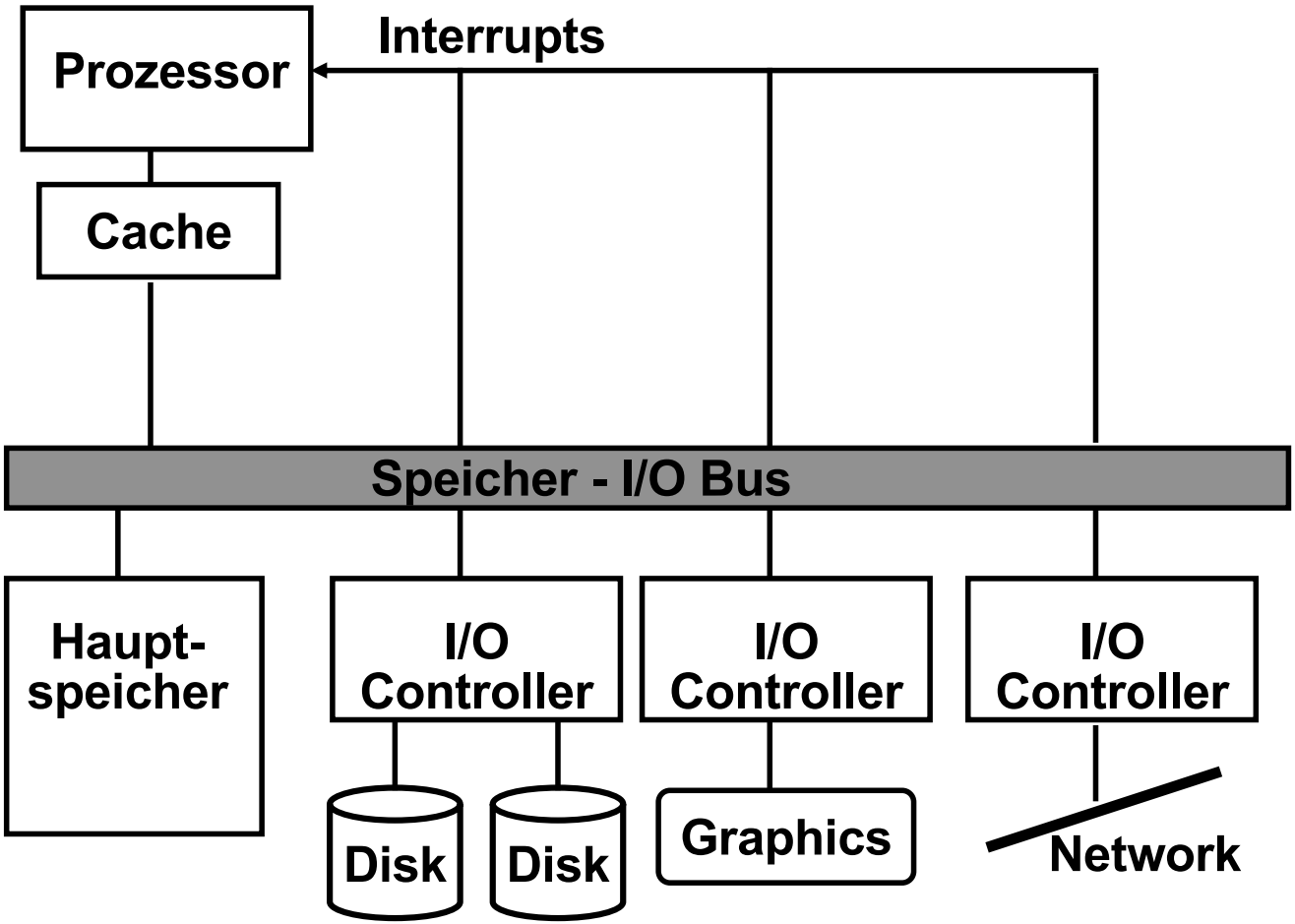
- I/O Geräte sind unglaublich vielfältig in Bezug auf
  - Verhalten – input, output oder speichern
  - Partner – Mensch oder Maschine
  - Datenrate – Der Höchstdurchsatz welcher zwischen den I/O Geräten, Hauptspeicher und Hauptspeicher übertragen werden kann

Gerät	Verhalten	Partner	Datenrate (Mb/s)
Keyboard	input	Mensch	0.0001
Mouse	input	Mensch	0.0038
Laser printer	output	Mensch	3.2
Graphics display	output	Mensch	800-8000
Network/LAN	input or output	Maschine	100-1000
Magnetic disk	storage	Maschine	240-2560

I/O Performance Messungen

- I/O Bandbreite (Durchsatz) – Menge an Informationen die pro Zeiteinheit über eine Verbindung (e.g. Bus) zwischen Prozessor/Speicher (I/O Geräten) ein- und ausgegeben werden kann
  1. Wie viele Daten können wir durch das System in einer bestimmten Zeit verschieben?
  2. Wie viele I/O Operationen können wir pro Zeiteinheit durchführen?
- I/O Antwortzeit (Latenz) – Die total vergangene Zeit um eine input oder output Operation abzuschliessen
  - Eine speziell wichtige Performancemetrik in Echtzeitsystemen
- Viele Applikationen benötigen beides, hohen Durchsatz und kurze Antwortzeiten

Ein typisches I/O System



I/O System Performance

- Designen eines I/O System welches den Anforderungen an die Bandbreite und/oder Latenz entspricht heisst:

1. Finden des schwächsten Glied im I/O System – Finde die Komponente welche das Design limitiert
   - Das Prozessor- und Speichersystem?
   - Die darunterliegenden Verbindungen (e.g. Bus) ?
   - Die I/O Steuerung?
   - Ein angeschlossenes I/O Geräte?
2. (Re)konfigurieren des schwächsten Glieds um die Anforderungen an Bandbreite und/oder Latenz zu erfüllen
3. Bestimmen der Anforderungen an die restlichen Komponenten um diese Anforderungen an Bandbreite und/oder Latenz zu erfüllen

I/O System Performance Beispiel

- Eine Arbeitslast einer Disk besteht aus 64KB reads und writes wobei das (User) Programm 200'000 Befehle pro Disk I/O ausführt und
  - Ein Prozessor der 3 Milliarden Befehle/s unterstützt und durchschnittlich 100'000 BS Befehle für eine Disk I/O Operation benötigt

Die maximale Disk I/O Rate (# I/O's/s) des Prozessors ist

$$\frac{\text{Befehl Ausführungsrate}}{\text{Befehle pro I/O}} = \frac{3 \times 10^9}{(200 + 100) \times 10^3} = 10'000 \text{ I/O's/s}$$

  - Ein Speicher-I/O Bus mit einer möglichen Datenraten von 1000 MB/s

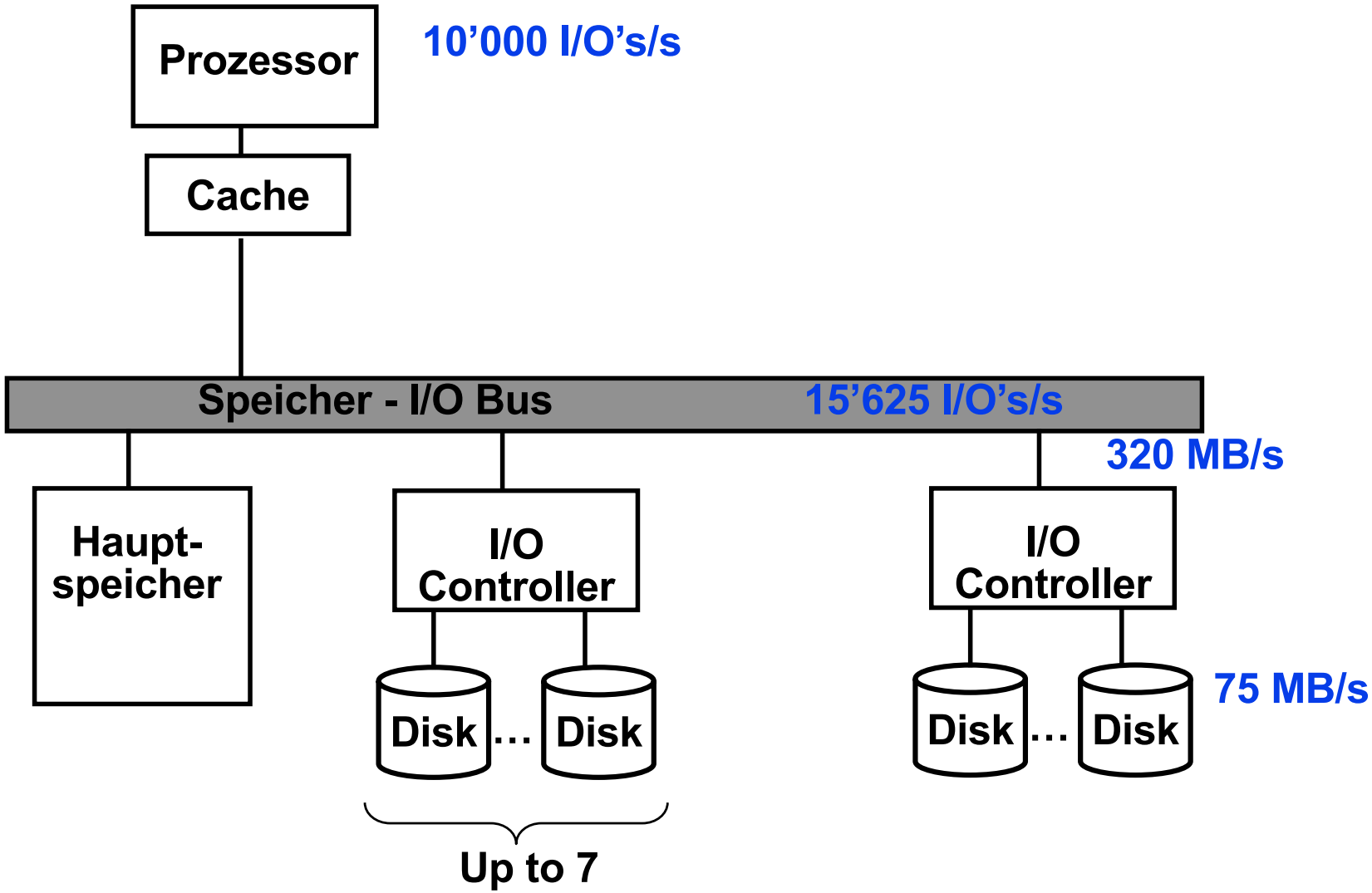
Jede Disk I/O liest/schreibt 64 KB daher die maximale I/O Rate des Busses

$$\frac{\text{Bus Bandbreite}}{\text{Bytes pro I/O}} = \frac{1000 \times 10^6}{64 \times 10^3} = 15'625 \text{ I/O's/s}$$

  - SCSI Disk I/O Steuerungen mit einer DMA Übertragungsrate von 320 MB/s können bis 7 Disks pro Steuerung verwalten
  - Jede Disk hat eine read/write Bandbreite von 75 MB/s und eine durchschnittliche Latenz 6 ms (suchen plus rotieren)

Frage: Was ist die maximal mögliche I/O Rate und was ist die Anzahl Disks und SCSI Steuerungen um diese Rate zu erreichen?

Disk I/O System Beispiel



I/O System Performance Beispiel

Wir haben gesehen, der Prozessor nicht der Bus ist der Flaschenhals

- Disks mit einer read/write Bandbreite von 75 MB/s und einer durchschnittlichen Latenz von 6 ms (für suchen plus rotieren)

Disk I/O read/write Zeit = Such + Rotations + Übertragungs Zeit =
6ms + 64KB/(75MB/s) = 6.9ms

Somit kann jede Disk 1000ms/6.9ms bzw. 146 I/O's pro Sekunde vollenden.
Um den Prozessor auszulasten benötigen wir 10,000 I/O's per second or
10,000/146 = 69 disks

Um die Anzahl SCSI Disk Steuerungen zu berechnen, benötigen wir die durchschnittliche Übertragungsrate pro Disk um sicherzustellen, dass wir die maximalen 7 Disks pro SCSI Steuerungen nutzen können und eine Disk Steuerungen nicht den Speicher-I/O Bus während der DMA Übertragung überlastet

Disk Übertragungsrate = (transfer size)/(transfer time) = 64KB/6.9ms = 9.56 MB/s

Daher würden 7 Disks weder die SCSI Steuerung (mit max. Übertragungsrate von 320 MB/s) noch den Speicher-I/O Bus (1000 MB/s) überlasten. Das heisst wir benötigen 69/7 oder 10 SCSI Steuerungen.

I/O System Verbindung

- Ein Bus ist eine geteilte Kommunikationsverbindung (Set an Kabel welches verschiedene Subsysteme verbindet) welche eine ganze Bandbreite an Geräten mit verschiedensten Latenzen und Datenraten verbindet
  - Vorteile
    - Vielseitig – Neue Geräte können einfach angehängt und zwischen Computersystemen mit dem selben Busstandard ausgetauscht werden
    - Tiefe Kosten – Ein Set einzelner Kabel wird geteilt
  - Nachteile
    - Erzeugt einen Flaschenhals – Die Bus Bandbreite limitiert den maximalen I/O Durchsatz
- Die max. Busgeschwindigkeit ist üblicherweise limitiert durch
  - Die Länge des Bus
  - Die Anzahl der Geräte am Bus

Bus Eigenschaften

- Steuerleitungen
  - Signalanfragen und Acknowledgments
  - Gibt Typ der Information auf der Datenleitung an
- Datenleitungen
  - Daten, Adresse, und evtl. komplexere Kommandos
- Bus Transaktion besteht aus
  - Master erteilt das Kommando (und Adresse) – request
  - Slave empfängt (oder sendet) die Daten – action
  - Die Transaktion ist definiert durch dem Bezug zum Speicher
    - Input – Daten vom I/O Gerät in den Speicher (eingeben)
    - Output – Daten von Speicher zu den I/O Geräten (ausgeben)

Arten von Bussen

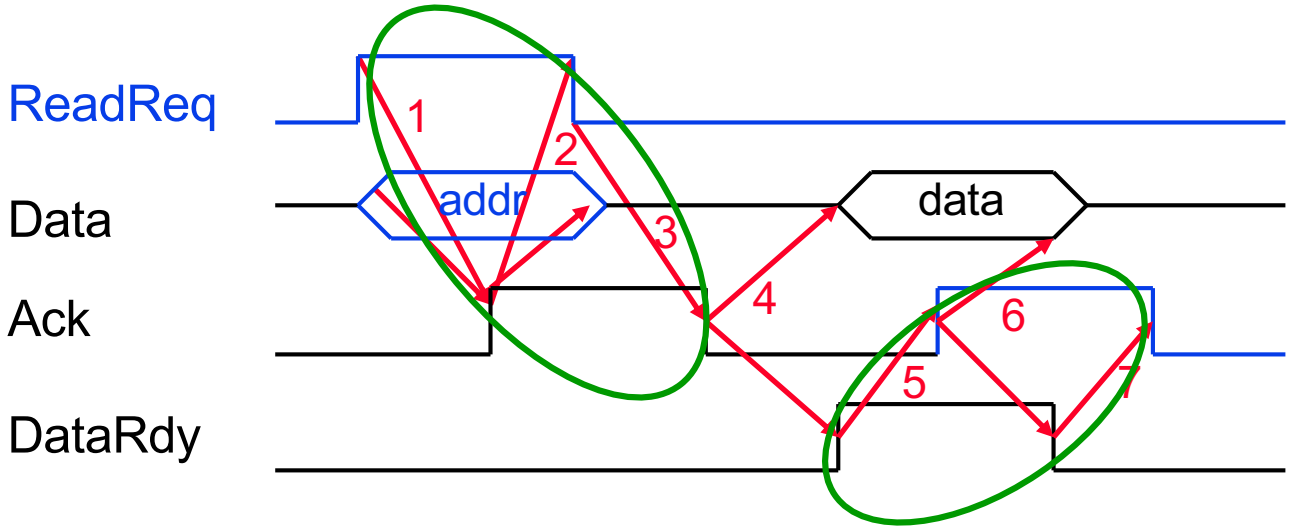
- Prozessor-Speicher Bus (proprietär)
  - Kurz mit hoher Geschwindigkeit
  - Angepasst an das Speichersystem um die Speicher-Prozessor Bandbreite zu maximieren
  - Optimiert für das Übermitteln von Cacheblöcken
- I/O Bus (Industriestandard, e.g., SCSI, USB, Firewire)
  - Normalerweise länger und langsamer
  - Muss ein breites Spektrum an I/O Geräten unterstützen
  - Verbunden mit dem Prozessor-Speicher Bus oder Backplane Bus
- Backplane Bus (Industriestandard, e.g., ATA, PCIexpress)
  - Befand sich oft auf der Rückwand des Gehäuses
  - Ein Bus der I/O Geräte mit dem Prozessor-Speicher Bus verbindet

Synchrone und Asynchrone Busse

- Synchroner Bus (e.g., Prozessor-Speicher Busse)
  - Beinhalten eine Uhr in den Steuerleitungen und hat ein fixes Protokoll zur Kommunikation relativ zur Uhr
  - Vorteile: Benötigt wenig Logik und kann sehr schnell sein
  - Nachteile:
    - Jedes auf dem Bus kommunizierende Gerät muss die selbe Taktrate haben
    - Um Taktversatz zu vermeiden, muss der Bus, wenn er sehr schnell sein soll gleichzeitig auch kurz sein
- Asynchroner Bus (e.g., I/O Busse)
  - Ohne Uhr, benötigt daher ein ein Handshaking Protokoll und zusätzliche Steuerleitungen (ReadReq, Ack, DataRdy)
  - Vorteile :
    - Kann weiten Bereich an Geräten und Geschwindigkeiten abdecken
    - Kann ohne Sorge von Taktversatz oder Synchronisationsproblemen verlängert werden
  - Nachteile: langsamer

Asynchrones Bus Handshaking Protokol

- Output (lese) Daten vom Speicher zu I/O Gerät



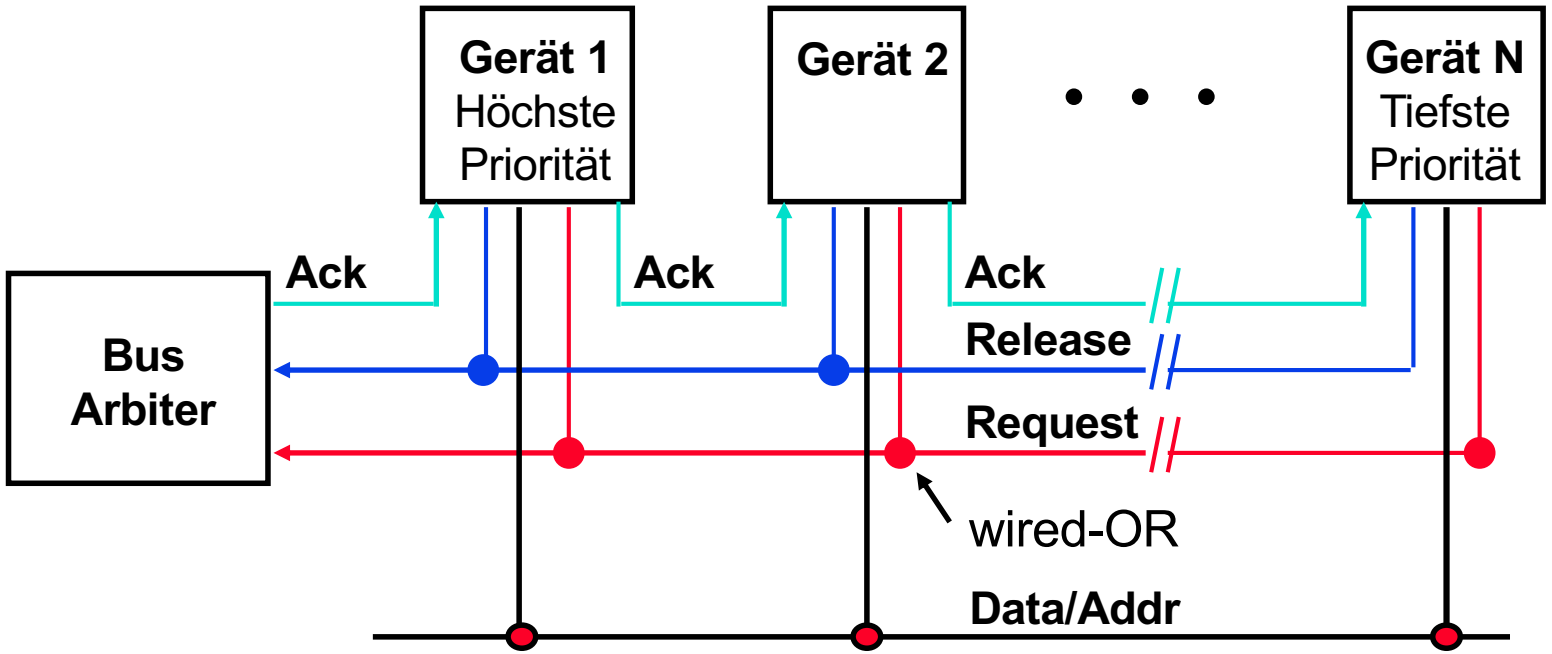
I/O Gerät signalisiert eine Anfrage durch ein steigendes ReadReq und anlegen der addr auf die Datenleitung

1. Speicher sieht ReadReq, liest addr von der Datenleitung, und setzt Ack
2. I/O Gerät sieht Ack und senkt ReadReq und gibt Datenleitung frei
3. Speicher sieht ReadReq ging nach unten und lässt Ack fallen
4. Ist der Speicher bereit, legt er Daten an Datenleitung an, setzt DataRdy
5. I/O Gerät sieht DataRdy, liest Daten Datenleitung, und setzt Ack
6. Speicher sieht Ack, gibt Datenleitung frei, und lässt DataRdy fallen
7. I/O Gerät sieht DataRdy geht nach unten und lässt Ack fallen

Notwendigkeit einer Bus Arbitrierung/Zuteilung

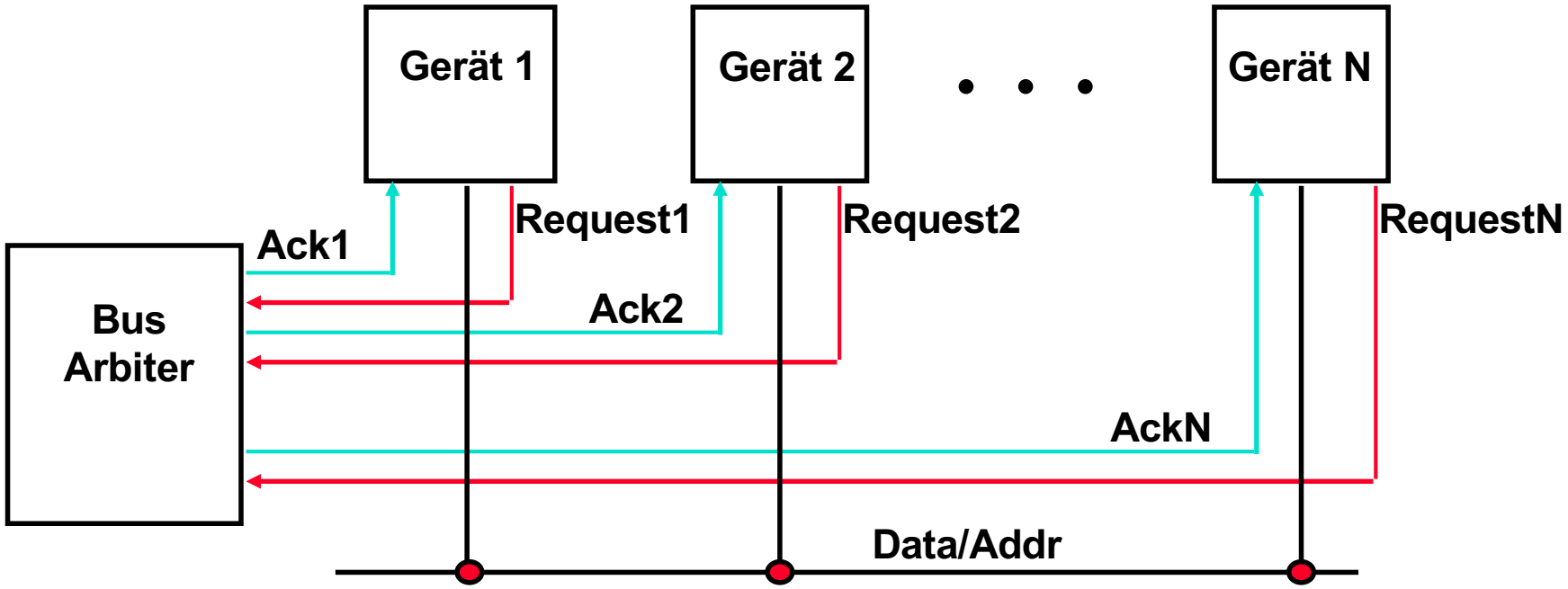
- Mehrere Geräte könnten den Bus gleichzeitig wollen. Daher brauchen wir einen Weg um mehrfache Anfragen zu behandeln
- Bus Arbitration Verfahren versuchen folgendes ausgleichen:
  - Bus Priorität – Gerät mit höchster Priorität zuerst
  - Fairness – Auch Geräte mit tiefster Priorität sollen nie komplett vom Bus ausgeschlossen werden
- Bus Arbitration Verfahren kann man in 4 Klassen aufteilen:
  - Daisy chain arbitration – Nächste Folie
  - Zentralisierte, parallele arbitration – Übernächste Folie
  - Distributed arbitration by self-selection – Jedes Gerät welches den Bus belegen möchte, platziert einen Code der dessen Identität dem Bus anzeigt
  - Distributed arbitration by collision detection – Gerät nutzt den Bus wenn er frei ist oder wartet nach einer Kollision mit anderem Gerät eine bestimmte Zeit. (Ethernet)

Daisy Chain Bus Arbitration



- Vorteile: Einfach
- Nachteile:
  - Kann keine Fairness garantieren – ein Gerät mit tiefer Priorität könnte niemals Zugriff bekommen
  - Langsamer – Das Daisy-Chain Signal limitiert die Busgeschwindigkeit

Zentralisierte Parallele Arbitration

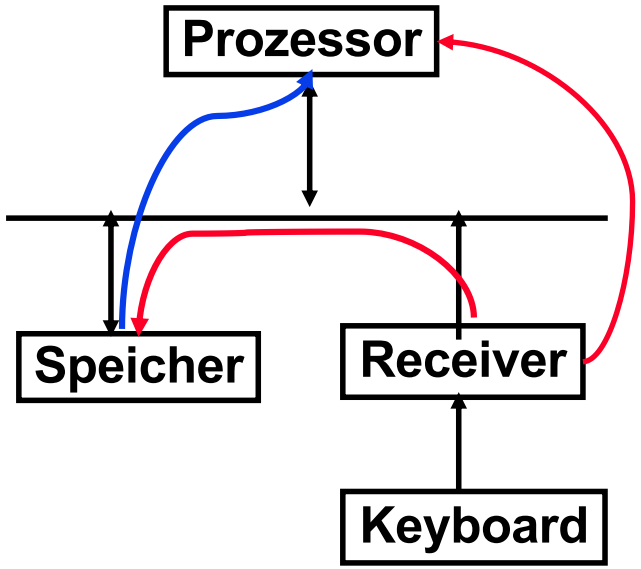


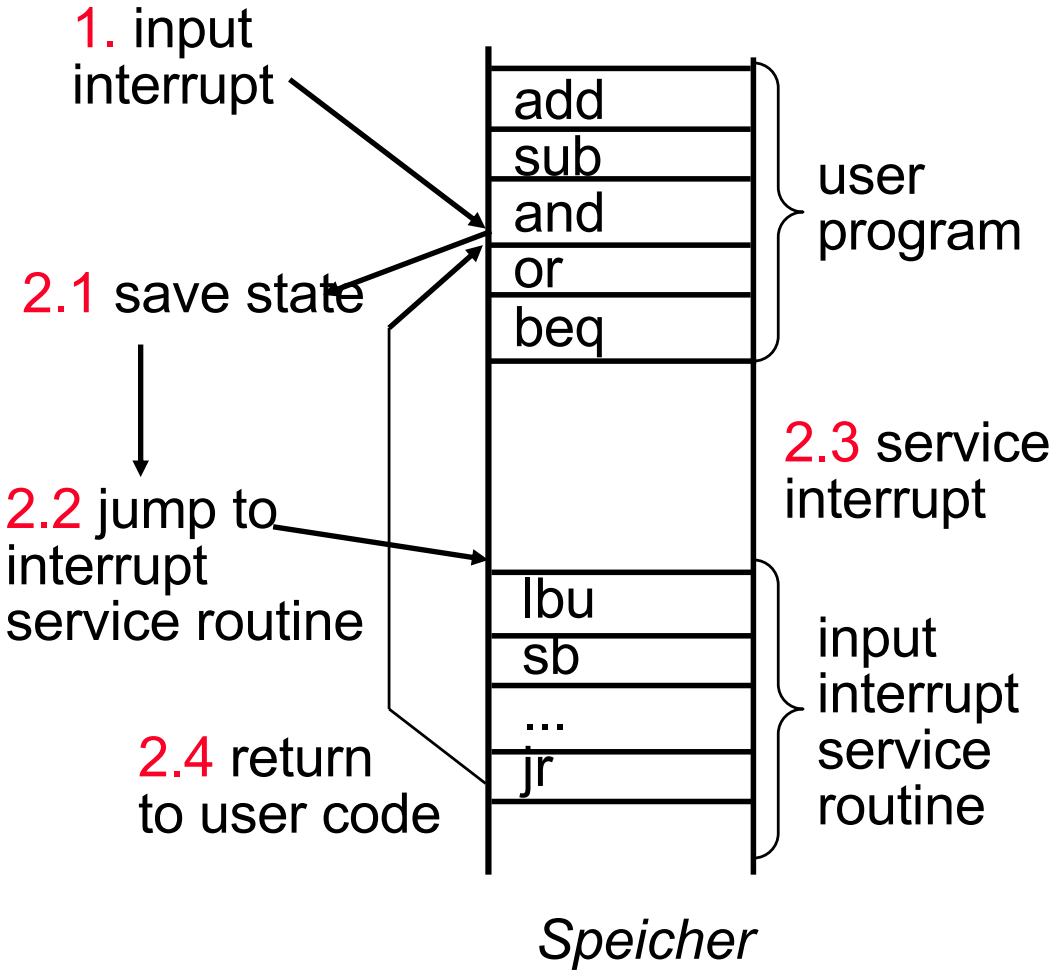
- Vorteile: Flexibel, kann Fairness sicherstellen
- Nachteile: Komplexere Arbiter Hardware
- Genutzt in eigentlich allen Prozessor-Speicher Bussen und in high-speed I/O Bussen

Kommunikation von I/O Geräten und Prozessor

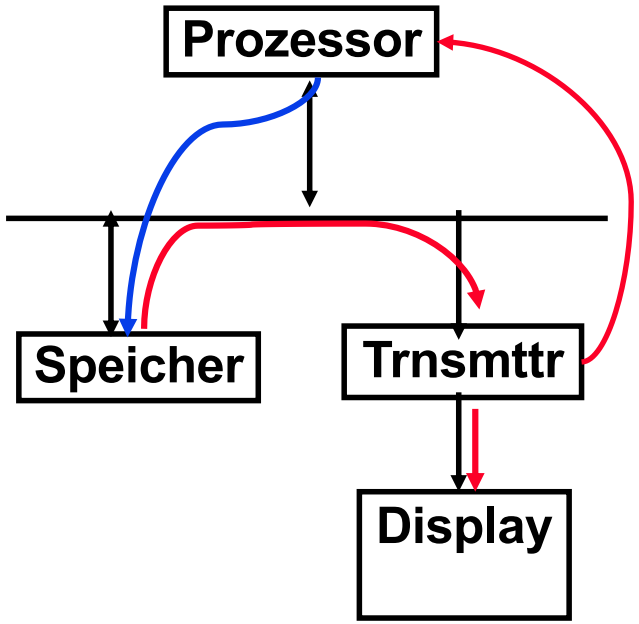
- Wie der Prozessor die I/O Geräte steuert
  - Spezielle I/O Befehle
    - Diese müssen Gerät und Kommando spezifizieren
  - Speicher-Mapped I/O
    - I/O Geräten wird Teil des oberen Speicheradressraum zugewiesen
    - Read und writes dieser Speicheradressen werden als Kommando für diese I/O Geräte interpretiert
    - Load/stores in den I/O Adressraum können nur durch das BS ausgeführt werden
- Wie die I/O Geräte mit dem Prozessor kommunizieren
  - Polling – der Prozessor prüft periodisch des Status der I/O Geräte um herauszufinden ob diese etwas tun müssen
    - Prozessor hat komplette Kontrolle – macht aber die ganze Arbeit
    - Kann durch Geschwindigkeitsunterschiede viel Prozessorzeit verschwenden
  - Interrupt-driven I/O – Die I/O Geräte senden einen Interrupt an den Prozessor um Anzuzeigen das sie etwas benötigen

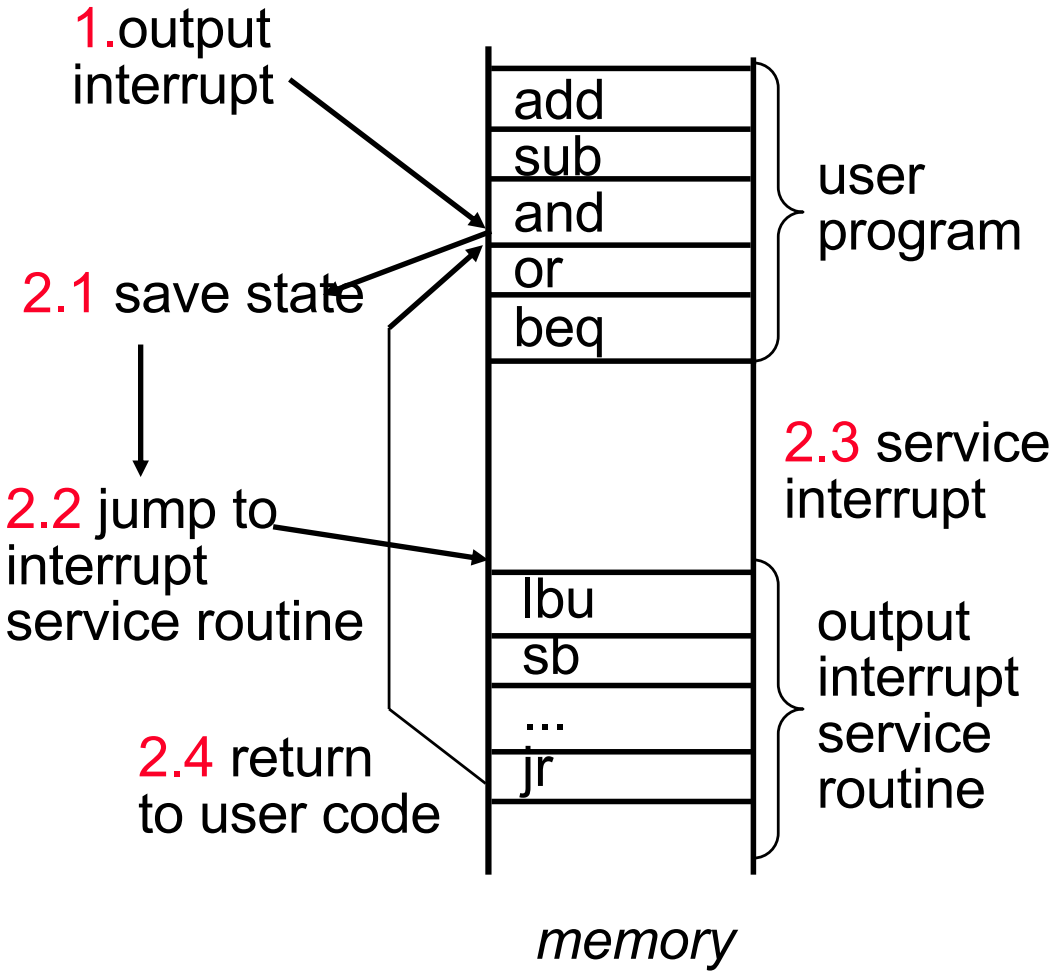
Interrupt-Driven Input





Interrupt-Driven Output





Interrupt-Driven I/O

- Ein I/O Interrupt ist bezüglich Befehlsausführung asynchron
  - Er gehört zu keinem Befehl und kann so keinen Befehl an der Komplettierung hindern
    - Man kann einen geeigneten Punkt zur Behandlung des Interrupt wählen
- Mit I/O Interrupts
  - Benötigt Möglichkeit Interrupt produzierendes Gerät zu identifizieren
  - Kann verschiedene Dringlichkeit haben (benötigt evtl. Priorisierung)
- Vorteile eines Interrupt
  - Befreit den Prozessor vom regelmässigen Pollen eines I/O Event.
- Nachteil – Spezielle Hardware wird benötigt um
  - Einen Interrupt auszulösen (I/O Gerät), ihn zu detektieren und zum speichern nötiger Informationen um die normale Verarbeitung nachdem Behandeln des Interrupts weiterführen zu können (Prozessor)

Direct Memory Access (DMA)

- Für Geräte mit hoher Bandbreite (wie Disks) würde interrupt-driven I/O viele Prozessorzyklen belegen
- DMA – die I/O Steuerung hat die Fähigkeit Daten ohne Prozessor direkt zum/vom Speicher zu verschieben
  1. Der Prozessor beginnt den DMA Transfer durch das Liefern der I/O Geräte Adresse, die auszuführenden Operation, der Ziel und Source Speicheradressen und die Anzahl zu übertragenden Bytes
  2. Die I/O DMA Steuerung regelt den gesamten Transfer (möglicherweise tausende von Bytes), welche den Bus benötigen
  3. Ist der DMA Transfer komplett, unterbricht die I/O Steuerung den Prozessor um ihm mitzuteilen der Transfer ist beendet
- Eventuell sind mehrere DMA Geräte in einem System
  - Prozessor und I/O Steuerungen konkurrieren um Buszyklen und für Speicher

Das DMA Stale Data Problem

- In Systemen mit Caches, kann es zwei Kopien von Datenwerten geben, eine im Cache, eine im Hauptspeicher
  - Für ein **DMA read** (Disk zu Speicher) – würde der Prozessor «stale data» verwenden wenn bereits eine Kopie im Cache liegt
  - Für ein **DMA write** (Speicher zu Disk) und ein write-back Cache – würde das I/O Gerät «stale data» erhalten wenn eine Kopie welche noch nicht in den Hauptspeicher kopiert wurde noch im Cache liegt
- Das Kohärenz Problem wird gelöst durch
  1. Leiten aller I/O Aktivitäten durch den Cache – teuer und massiver negativer Einfluss auf die Performance
  2. Das BS soll selektiv den Cache für ein I/O **read** als ungültig markieren oder write-backs für ein I/O **write** (flushing) erzwingen
  3. Bereitstellen von Hardware um den Cache selektiv als ungültig zu markieren oder den Cache zu leeren (flush) – benötigt ein Hardware ”Schnüffler” (snooper)