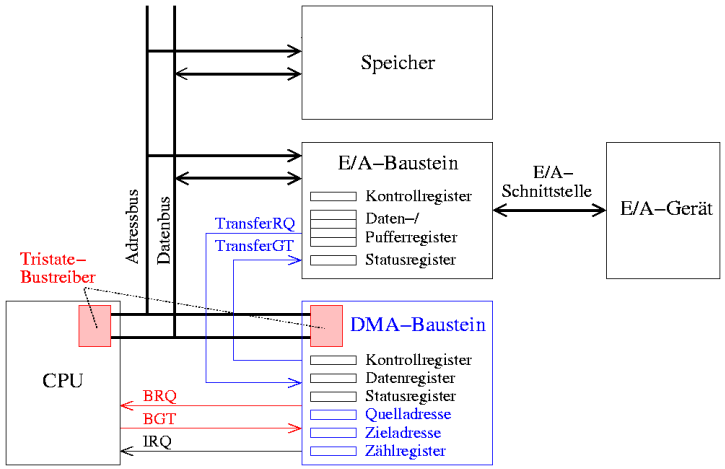
5 Direct Memory Access

- oft werden lange Datenströme aus dem Speicher zur Peripherie ausgegeben, bzw. von der Peripherie in den Speicher eingelesen
  (⇒ unnötige Belastung der CPU mit trivialen Aufgaben: Inkrementieren der Adresse, Zählen der Datenworte, Statusabfrage des E/A-Bausteins)
- Idee: ein zusätzlicher **DMA-Baustein** (*Direct Memory Access*) führt nach Initialisierung durch die CPU den Speichertransfer selbständig durch
  (⇒ CPU kann sich anspruchsvolleren Tätigkeiten widmen!)
- ein DMA-Baustein enthält
  - ein **Quelladressregister** und ein **Zieladressregister**, in dem die Start- und Zieladresse des zu transferierenden Datenblocks eingetragen werden
  - ein **Zählregister**, das mit der Anzahl der zu transferierenden Bytes bzw. Datenworten initialisiert werden muss
  - ein **Kontrollregister**, um z.B. Richtung oder Arbeitsmodus festzulegen

5 Direct Memory Access (2)

- prinzipieller Aufbau eines Systems mit DMA-Bausteins:



5 Direct Memory Access (3)

- prinzipieller Ablauf eines DMA-Transfers vom E/A-Baustein in den Speicher:
  1) CPU initialisiert E/A-Baustein (Kontrollregister) und DMA-Baustein (mit Startadresse, Zieladresse und Anzahl an Datenworten)
  2) E/A-Baustein setzt das Signal **TransferRQ** (*Transfer Request*), sobald Daten vorhanden sind
  3) DMA-Baustein fordert mit dem Signal **BRQ** (*Bus Request*) den Bus von der CPU an
  4) CPU deaktiviert eigene Bustreiber und setzt Signal **BGT** (*Bus Grant*)
  5) DMA-Baustein zeigt dem E/A-Baustein den Beginn des Datentransfers durch das Signal **TransferGT** (*Transfer Grant*) an
  6) DMA-Baustein transferiert Daten vom E/A-Baustein in den Speicher
  7) DMA-Baustein gibt durch Rücknahme von **BRQ** den Bus wieder frei
  8) DMA-Baustein kann der CPU durch Interrupt das Ende des Transfers signalisieren

5 Direct Memory Access (4)

- Arbeitsmodi eines DMA-Bausteins:
  - **Burst Modus**: DMA-Baustein erhält den Systembus für den kompletten Transfer eines Blocks aus vielen Datenworten
    - E/A-Baustein benötigt internen Pufferspeicher
    - Vorteil: sehr hohe Transferrate
    - Nachteil: CPU wird für lange Zeit am Speicher-/Buszugriff gehindert
  - **Cycle Stealing**: DMA-Buszyklen und CPU-Buszyklen werden gemischt
    - ein fester Anteil an Buszyklen (z.B. jeder zweite Buszyklus) wird vom DMA-Baustein der CPU „gestohlen“
    - DMA kann zusätzlich alle von der CPU nicht benötigten Buszyklen nutzen (z.B. während der Ausführungsphase von Register/Register-Befehlen)
    - Vorteil: CPU wird nur geringfügig behindert
    - Nachteil: geringere Transferrate

5 Direct Memory Access (5)

- Anmerkungen:
  - im Kontrollregister des DMA-Bausteins kann die **Übertragungs-art** gewählt werden:
    - beim **direkten** DMA-Transfer erfolgt die Datenübertragung direkt zwischen E/A-Baustein und Speicher (⇒ 1 Buszyklus je Wort)
    - beim **indirekten** DMA-Transfer wird jedes Wort im Datenregister des DMA-Bausteins zwischengespeichert (⇒ 2 Buszyklen je Wort)
  - Statusregister des DMA-Bausteins enthält Informationen über Zustand (frei/belegt), Blockende und Fehler
  - auch der Datentransfer zwischen zwei E/A-Bausteinen oder von Speicher zu Speicher kann über DMA realisiert werden
  - oft mehrere **DMA-Kanäle** je DMA-Baustein (⇒ mehrere Transfers gleichzeitig in Bearbeitung)
  - DMA-Baustein ist bei heutigen PCs im **Chipset** integriert

6 Bussysteme

- Verbindung von **mehreren** Komponenten über identische Transport- und Steuerleitungen
  (Vorteil: bedeutend kleinerer Verdrahtungsaufwand als bei Punkt-zu-Punkt Verbindungen)
- Bussysteme waren früher hersteller- und systemspezifisch, sind heute jedoch weitgehend **standardisiert**
  - E/A-Geräte bzw. E/A-Karten können unabhängig von Hersteller und Rechnersystem entwickelt werden:
    - größerer Absatzmarkt, geringere Kosten
    - Beispiele: Steckkarten für PCI-Bus, Festplatten
  - Standardisierung umfasst
    - Signale und Spannungspegel
    - zeitliches uns elektrisches Verhalten der Bussignale
    - Steckverbinder und Pinbelegungen

6 Bussysteme (2)

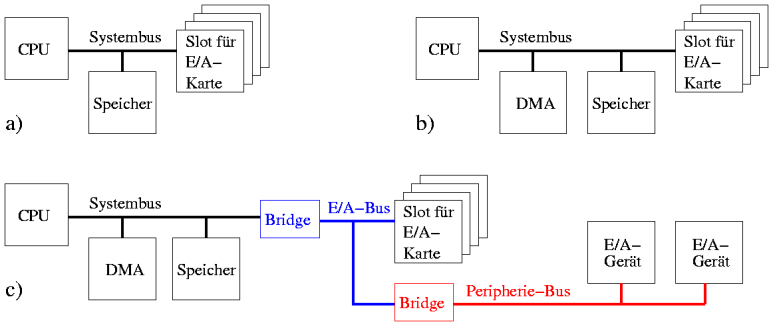
- verschiedene Arten von Bussystemen in einem Rechner:
  - **prozessorinterne Busse**
    - arbeiten i.a. mit CPU-Taktfrequenz
    - verbinden z.B. Registersatz, Arithmetik-Einheiten und L1-Datencache
  - **Systembus**
    - verbindet CPU mit schnellen Systemkomponenten
    - auch als *Front-Side-Bus* bezeichnet
    - Taktfrequenz typisch 100 bis 200 MHz
  - **Ein-Ausgabebus** (intern, auch *Local Bus*)
    - Bus für E/A-Erweiterungen der Hauptplatine, z.B. PCI-Bus
  - **Peripherie-Bus** (extern)
    - zum Anschluss mehrerer Peripheriegeräte an eine Busschnittstelle, z.B. USB, SCSI-Bus

6 Bussysteme (3)

- unterscheidende Merkmale von Bussystemen:
  - **Breite** von **Datenbus** und **Adressbus**
  - **Multiplexing** von Adress- und Datenleitungen
  - maximale **Datentransferrate** (in MByte/s)
  - **Taktung** (synchron/asynchron) und ggf. **Bustaktfrequenz**
  - **maximale Anzahl** von möglichen Buskomponenten bzw. Bussteckplätzen (*Slots*)
  - Art der **Busarbitrierung**
  - Art der möglichen **Buszyklen**
  - Realisierung von **Interrupts**
  - Unterstützung von **DMA**
  - **physikalische Größen**: Spannungspegel, Steckverbinder

6 Bussysteme (4)

- zwei verschiedene Bussysteme können mittels einer **Bridge** gekoppelt werden:
  - Anpassung z.B. von Datenbusbreite (z.B. 32 → 16 Bit), Taktfrequenz, Buszyklen
- einige mögliche **Systembus-Architekturen**:



6 Bussysteme (5)

- Bus besteht aus Datenleitungen (**Datenbus**), Adressleitungen (**Adressbus**), Steuerleitungen (**Steuerbus**) sowie weiteren Leitungen zur Stromversorgung
- oft **Multiplexing** von Daten und Adressen zur Einsparung von Busleitungen
- ein **Master** ist eine aktive Buskomponente, die einen Buszyklus auslösen kann (z.B. CPU, DMA-Baustein)
- ein **Slave** ist stets passiv (z.B. Speichermodul, E/A-Baustein)
- Busleitungen sind
  - zumeist *Tristate*-Leitungen, die je Buszyklus von einem ausgewählten Master getrieben werden
  - ggf. auch *Open-Collector*-Leitungen zur Realisierung einer globalen Oder-Verknüpfung (*Wired-OR*), z.B. für Interrupt- oder Busanforderung
  - i.a. „*low active*“ (aktiv, wenn das Bussignal = 0 ist)

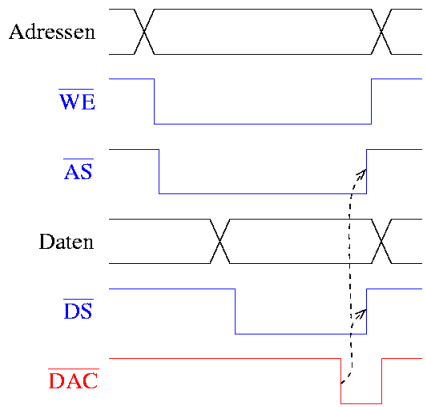
6.1 Busprotokolle

- standardisiertes **Busprotokoll** legt Zeitverhalten für verschiedene Arten von Buszyklen fest, z.B.:
  - Einzelwort-Transfer (Adresse, 1 Datenwort)
  - Block- oder Burst-Transfer (Wortanzahl *n*, Startadresse, *n* Datenworte)
- Busprotokoll auf einem **asynchronen Bus**:
  - Master adressiert den Slave und wählt die Richtung (Lesen/Schreiben)
  - Synchronisation über Handshaking z.B. mittels der Signale **/AS** (*Address Strobe*), **/DS** (*Data Strobe*) und **/DAC** (*Data Acknowledge*)
  - Ausbleiben der Bestätigung **/DAC** wird durch einen Timer überwacht; ggf. wird ein *Bus Error* ausgelöst
- Busprotokoll auf einem **synchronen Bus**:
  - Synchronisation erfolgt über ein gemeinsames Bustaktsignal (**BClk**)
  - es gibt feste Zeitpunkte für die Gültigkeit von Adressen und Daten und für die Übernahme von Daten

6.1 Busprotokolle (2)

- Beispiel 1: Schreiben eines Datenworts auf **asynchronem** Bus

1) Master gibt Adresse aus und setzt Signal **/WE** (*Write Enable)*
2) Master zeigt Gültigkeit der Adresse durch Signal **/AS**
3) Master gibt Datenwort aus
4) Master zeigt Gültigkeit des Datenworts durch **/DS** an
5) Slave quittiert Datenübernahme durch Signal **/DAC**
6) Master nimmt **/AS** und **/DS** zurück
7) Slave nimmt **/DAC** zurück



- Slave bestimmt die Länge eines Buszyklus durch Signal **/DAC** !

6.1 Busprotokolle (3)

- Beispiel 2: Schreiben eines Datenworts auf **synchronem** Bus

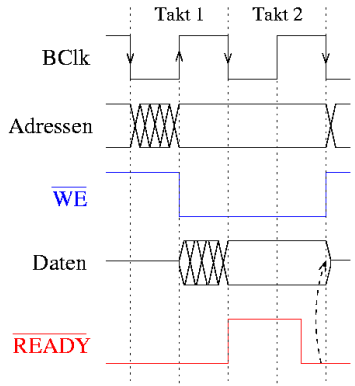
  1) Master beginnt bei ↓ einen neuen Buszyklus und gibt eine Adresse aus

  2) bei folgender Flanke ↑ ist Adresse gültig; gleichzeitig setzt Master das Signal **/WE** und gibt Datenwort aus

  3) bei folgender Flanke ↓ sind Daten gültig; Slave setzt **/READY=1**

  4) ist bei ↓ am Ende des zweiten Taktes wieder **/READY=0**, so hat Slave die Daten übernommen

  5) Master beginnt einen neuen Buszyklus



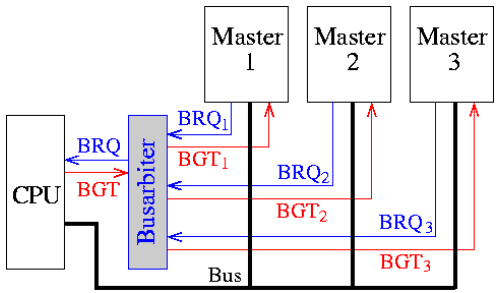
- Slave kann über Signal **/READY=1** den Buszyklus um einen oder mehrere Bustakte (*Wait Cycles*) verlängern!

6.2 Busarbitrierung

- können **mehrere** Buskomponenten **Master** werden, so ist eine eine **Busarbitrierung** erforderlich:
  - wenn ein Master den Bus benötigt, setzt er Signal **/BRQ** (*Bus Request*)
  - Zuteilung des Busses durch Signal **/BGT** (*Bus Grant*) an Master
  - nach Beendigung des Buszyklus deaktiviert Master das Signal **/BRQ**
  - **/BGT** wird zurückgesetzt
- eine einfache Busarbitrierung für *einen* zusätzlichen Master für den Systembus ist i.a. im Mikroprozessor integriert
- bei mehreren Mastern ist eine weitere externe Busarbitrierung erforderlich; zwei Varianten:
  - **zentrale** Busarbitrierung
    (zentrale Logik, implementiert in einem zusätzlichen Bussteuerbaustein)
  - **dezentrale** Busarbitrierung
    (Logik ist über alle Buskomponenten verteilt)

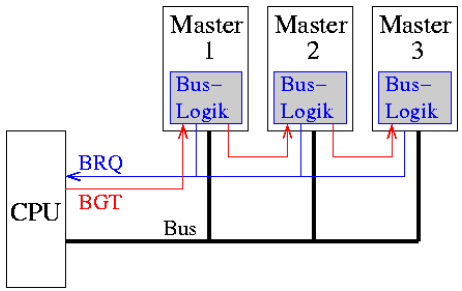
6.2 Busarbitrierung (2)

- Realisierung einer **zentralen** Busarbitrierung:
  - separate Anforderungsleitungen **/BRQ**$_i$ und Zuteilungsleitungen **/BGT**$_i$ für jeden Master i
  - beliebige Algorithmen zur Busarbitrierung in Hardware implementierbar
  - unterschiedliche Prioritäten können ggf. berücksichtigt werden
  - **schnelle** Buszuteilung
  - Beispiel: PCI-Bus



6.2 Busarbitrierung (3)

- Realisierung einer **dezentralen** Busarbitrierung:
  - eine gemeinsame **/BRQ**-Leitung, realisiert in *Wired-OR* Technik
  - Signal **/BGT** wird in einer Kette (*Daisy Chain*) weitergegeben: Master i gibt Signal **/BGT** an Master i + 1 weiter, wenn er Bus nicht angefordert hat
  - Master mit kleinstem Index in der Kette hat höchste Priorität
  - **unfaire** Busvergabe
    (⇒ Aushungerungsproblem!)
  - **langsame** Buszuteilung

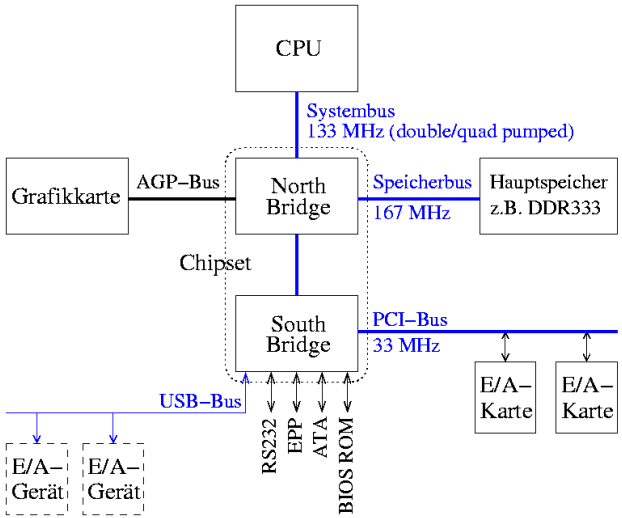


6.4 Fallstudie: PC-Bussysteme

- Bussysteme im PC:
  - Hauptplatine enthält x86 CPU und Chipset, gekoppelt über **Systembus** (64 Bit Daten, 32 Bit Adressen, synchron, typisch 100 bis 200 MHz Taktfrequenz)
  - Datenübertragungsrate höher, da je Taktzyklus oft mehrere Datenworte übertragen werden („*double pumped*“ oder „*quad pumped*“)
  - Chipset steuert **Speicherbus** (i.a mit einer vom Systembus abweichenden Taktfrequenz)
  - Chipset enthält serielle und parallele Schnittstelle, DMA-Baustein, Plattenkontroller, Bridge für PCI-Bus, Bridge für USB, ...
  - Chipset kann mehrere CPU-Zugriffe puffern und ggf. zusammenfassen
  - schneller **AGP-Bus** (*Accelerated Graphics Port*) für Grafikkarte
  - **PCI-Bus** mit mehreren PCI-Steckplätze (PCI-*Slots*) für Soundkarte, 10/100 MBit-Netzwerkkarte, ...

6.4 Fallstudie: PC-Bussysteme (2)

- Bus-Architektur eines PC: (Stand 2003)



6.4 Fallstudie: PC-Bussysteme (3)

- früher: **ISA-Bus** (*Industry Standard Architecture*, 1984)
  - zuerst 8-Bit Daten, später **16-Bit Daten** und **24 Bit Adressen**
  - synchroner/asynchroner Bus mit **8 MHz** Taktfrequenz, max. 8 MByte/s
  - konzipiert als prozessornaher Systembus für 286-basierte AT PCs, Bussignale überwiegend identisch zum Prozessorbus
  - ISA-Steckverbinder: (62+36 Pins)

  - E/A-Adressen und Interrupts auf ISA-Buskarte über Jumper einzustellen
  - bis vor kurzem als weiterer Ein-/Ausgabebus für langsame, preiswerte E/A-Karten in PCs eingesetzt
  - die Übertragungsrate war Ende der 80er Jahre für PCs nicht mehr ausreichend; viele Alternativen wurden entwickelt:
    - **MCA** (*Microchannel Architecture*, IBM, 1987): 32-Bit Daten, 10 MHz
    - **EISA** (*Enhanced* ISA, 1989): 32-Bit Daten und Adressen, 8 MHz
    - **VLB** (VESA *Local Bus*, 1992): 32-Bit Daten, 40 MHz

6.4 Fallstudie: PC-Bussysteme (4)

- heute: **PCI-Bus** (*Peripheral Component Interconnect*, 1993)
  - synchroner Bus, von Intel entwickelt
  - 12 Arten von Buszyklen, u.a. auch Einzelwort- und Burst-Transfer mit beliebiger Blocklänge
  - PCI 1.0 (heute typisch): **32-Bit Daten**, bis zu **33 MHz** Bustaktfrequenz
  - PCI 2.1: auch **64-Bit Daten**, bis zu **66 MHz** Bustaktfrequenz möglich
  - theor. max. Übertragungsraten von **133 MByte/s** (32-Bit Bus, 33 MHz) bis zu **533 MByte/s** (64-Bit Bus, 66 MHz) bei Burst-Transfer
  - **Multiplexing** von Daten und **32-Bit Adressen**
  - bis zu **4 masterfähige Slots** mit zentraler Bus-Arbitrierung
  - PCI-Steckverbinder: (124 Pins bei 32-Bit Daten)

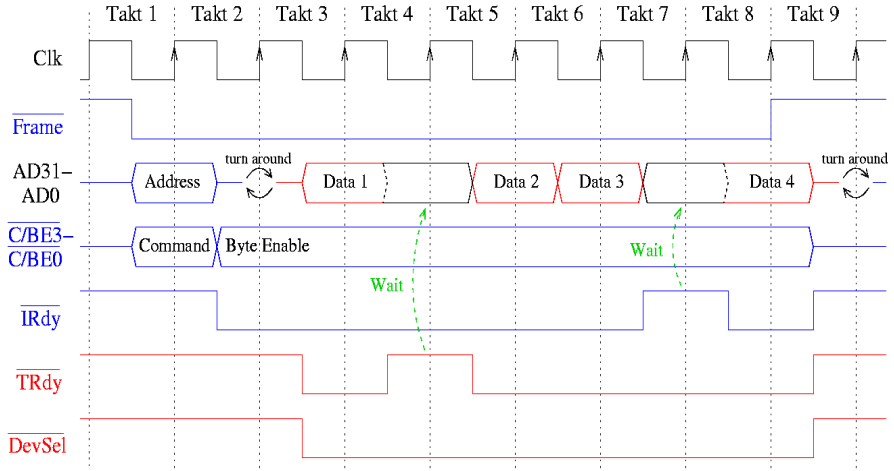
  - prozessorunabhängiger Bus (nicht als Systembus einsetzbar!), auch in anderen Architekturen (z.B. Ultra-Sparc, PowerPC, ...) verbreitet
  - Weiterentwicklung: **PCI-X** mit bis zu 133 MHz Taktfrequenz

6.4 Fallstudie: PC-Bussysteme (5)

- **PCI-Busleitungen** (Auswahl) für **Master** und **Slave**:
  - **Clk**: Bustakt, auf dem alle Signale synchronisiert sind
  - **AD31** bis **AD0**: gemultiplexter 32-Bit Adress-/Datenbus
  - **C/BE3** bis **C/BE0** (*Command / Byte Enable*): enthält entweder ein Bus-Kommando (zur Auswahl einer Buszyklusart) oder eine Byteauswahl (aus 32-Bit Datenwort)
  - **/REQ$_i$**: Busanforderung von Karte in Slot i
  - **/GNT$_i$**: Buszuteilung an Karte in Slot i
  - **/INTA** bis **/INTD**: vier Unterbrechungs-Leitungen
  - **/Frame**: signalisiert Beginn und Ende eines Buszyklus
  - **/IRdy** (*Initiator Ready*): Master ist bereit zum Datentransfer
  - **/TRdy** (*Target Ready*): Slave ist bereit zum Datentransfer
  - **/DevSel**: Target bestätigt die Dekodierung seiner Adresse
  - **/IdSel$_i$**: Auswahl von Karte in Slot i zur Konfiguration

6.4 Fallstudie: PC-Bussysteme (6)

- Beispiel eines PCI-Buszyklus (Lesen eines Blocks aus 4 Worten):



6.4 Fallstudie: PC-Bussysteme (7)

- **bidirektionale Flußkontrolle**:
  - sowohl **Master** (*Initiator*) als auch **Slave** (*Target*) können den Transfer eines Datenworts durch Aktivierung von **/IRDY** bzw. **/TRDY** um einen oder mehrere Bustakte verzögern
- max. **Transferrate**: ein 32-Bit Wort je Bustakt
  - kann nur bei einem sehr langen Burst-Transfer erreicht werden
- **Richtungsumschaltung** der Bustreiber (*turn around*)
  - für gemultiplexte Adress-/Datenleitungen
  - benötigt einen zusätzlichen Bustakt
  - nur beim Lesen erforderlich ($\Rightarrow$ langsamer als Schreiben!)
- über PCI-PCI-Bridges **hierarchisch** erweiterbares Bus-System mit maximal 255 PCI-Bussen

6.4 Fallstudie: PC-Bussysteme (8)

- **3 Adressräume:**
  - 32 Bit **Speicher-Adressraum** (max. 4GByte)
  - 32 Bit **E/A-Adressraum** (I/O-Ports)
  - **Konfigurations-Adressraum** (256 Byte je PCI-Karte)
  - Buskommando legt für jeden Buszyklus den Adressraum fest, z.B.:

    0010 *I/O Read* 0110 *Memory Read* 1010 *Configuration Read*
    0011 *I/O Write* 0111 *Memory Write* 1011 *Configuration Write*
  - bei Zugriff auf den Speicher-Adressraum sind **Burst-Transfers** möglich, z.B. mit den Buskommandos

    1110 *Memory Read Line* (Lesen einer kompletten Cachezeile)
    1100 *Memory Read Multiple* (Lesen mehrerer Cachezeilen in einem Buszyklus)