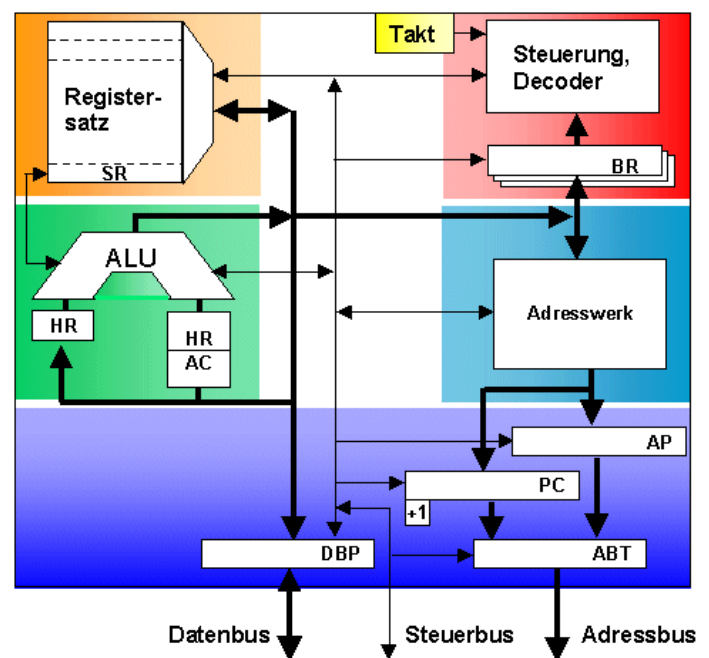


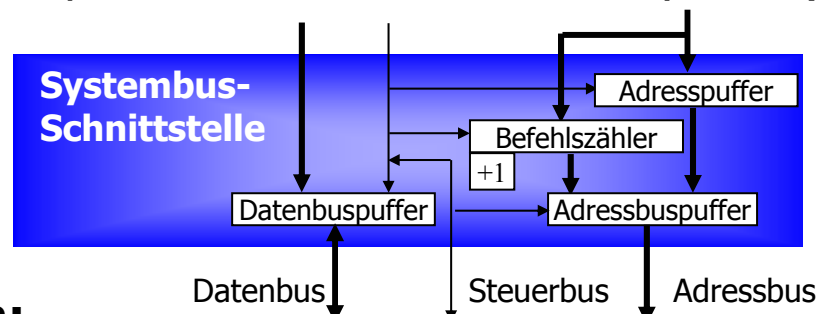
Interner Aufbau eines einfachen mP

- ❑ Steuerwerk
- ❑ Rechenwerk
- ❑ Adresswerk
- ❑ Registersatz
- ❑ Interne Busse
- ❑ **Systembusschnittstelle**



Die Systembus-Schnittstelle

Die Systembus-Schnittstelle (Bus Interface Unit, BIU) stellt die Verbindung des Mikroprozessors zu seiner Umwelt (Komponenten des Mikrorechnersystems) dar.



Aufgaben:

- kurzfristiges Zwischenspeichern (Puffern) von Adressen und Daten
- elektrische Anpassung der Signalpegel



Register der Systembus-Schnittstelle

□ **Datenbuspuffer (DBP)**

- speichert alle Daten vom und zum Prozessor
- verlangsamt ggf. den Zugriff auf externen Speicher
- schaltet Ausgänge ggf. hochohmig (Tristate).

□ **Befehlszähler (PC / IP)**

- enthält die Adresse der Speicherzelle des nächsten auszuführenden Befehls
- wird nach jedem Befehl inkrementiert (außer bei Sprüngen und Unterprogrammaufrufen, dort wird der Befehlszähler mit der Sprungadresse geladen)



Register der Systembus-Schnittstelle

□ **Adresspuffer (AP)**

speichert die vom Adresswerk erzeugten Operandenadressen.

□ **Adressbustreiber:**

- schaltet wahlweise den Befehlszähler (PC) oder der Adresspuffer (AP) auf den Adressbus
- schaltet Adressen ggf. hochohmig (Tristate)



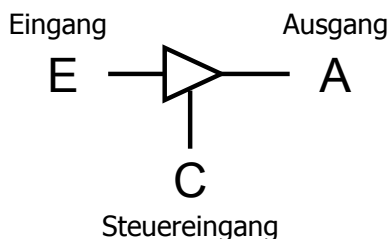
Tri-State-Treiber

Gatter, die neben den Pegelzuständen **H** (high) und **L** (low) einen **dritten hochohmigen Zustand** besitzen.

- ❑ In diesem Zustand ist der Ausgang hochohmig gegen Betriebsspannungen beider Polaritäten.
- ❑ Diese Gatter ermöglichen es, mehrere Gatterausgänge auf eine gemeinsame Leitung zusammenzuschalten (Bussystem)
- ❑ Sie dienen auch durch Ausgangstreiber zur elektrischen Anpassung der Prozessorsignale an die Signalspezifikationen, die von anderen Systemkomponenten verlangt werden.

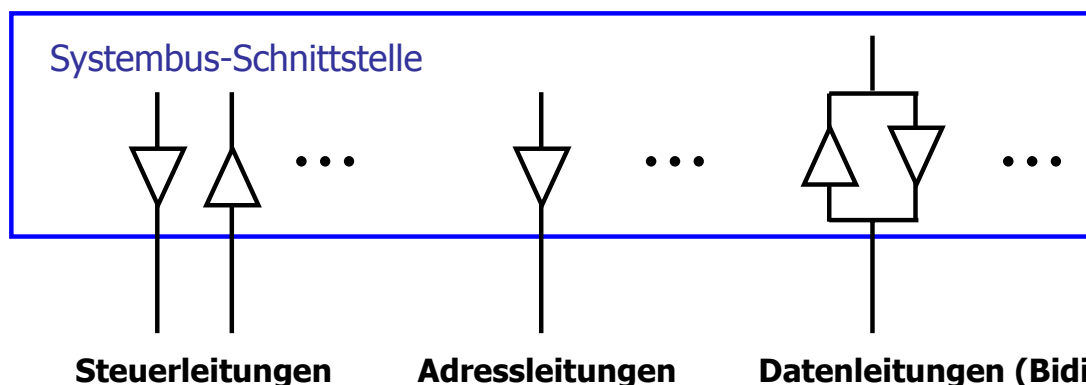


Tri-State-Treiber



Funktionstabelle:

C	E	A
H	L	L
H	H	H
L	-	Z hochohmig



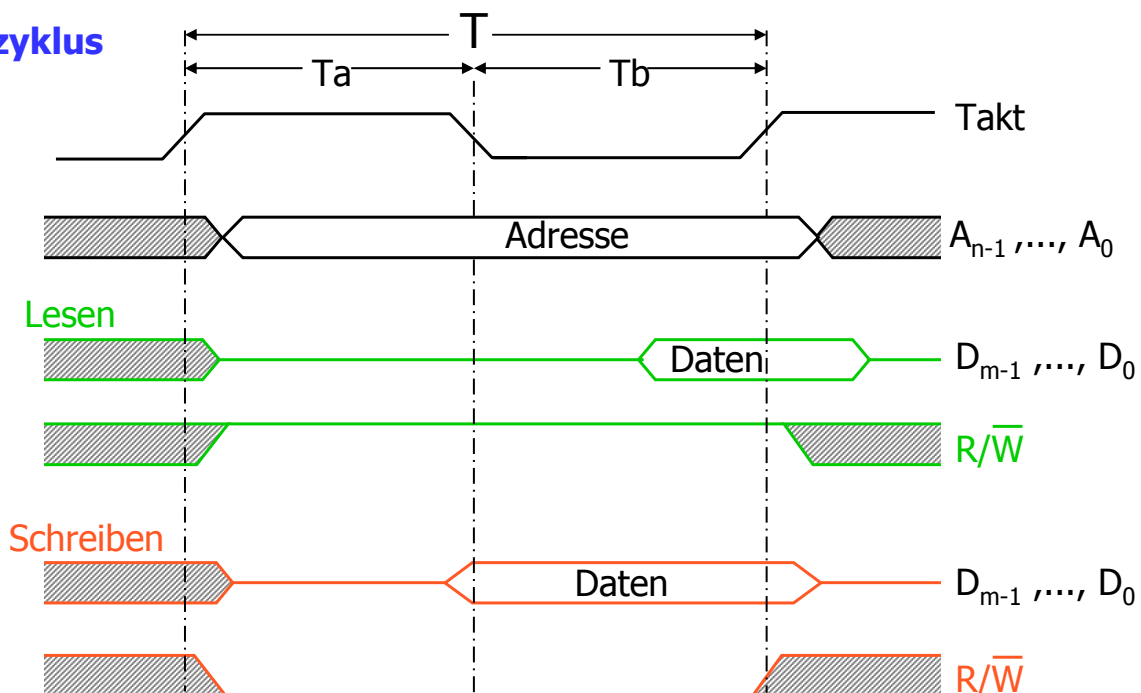
Zeitverhalten der Systembussignale

- ❑ Synchroner Systembus
- ❑ Semi-Synchroner Systembus
- ❑ Asynchroner Systembus



Zeitverhalten eines synchronen Systembus

T: Buszyklus



Timing

Adresse wird zu Beginn des Buszyklus (T_a) auf den Adressbus gelegt

Auswahl der Übertragungsrichtung durch R/\overline{W}

Lesen ($R/\overline{W} = 1$):

- Speicher (oder andere Systemkomponente) liefert ihre Daten gegen Ende des Buszyklus (T_b)
- Übernahme der Daten in den Prozessor mit der steigenden Flanke des Systemtakts

Schreiben ($R/\overline{W} = 0$):

- Prozessor legt die Daten zu Beginn der zweiten Takthälfte (T_b) auf den Systembus
- Übernahme der Daten in den Speicher durch die steigende Flanke von R/\overline{W} oder des Systemtakts

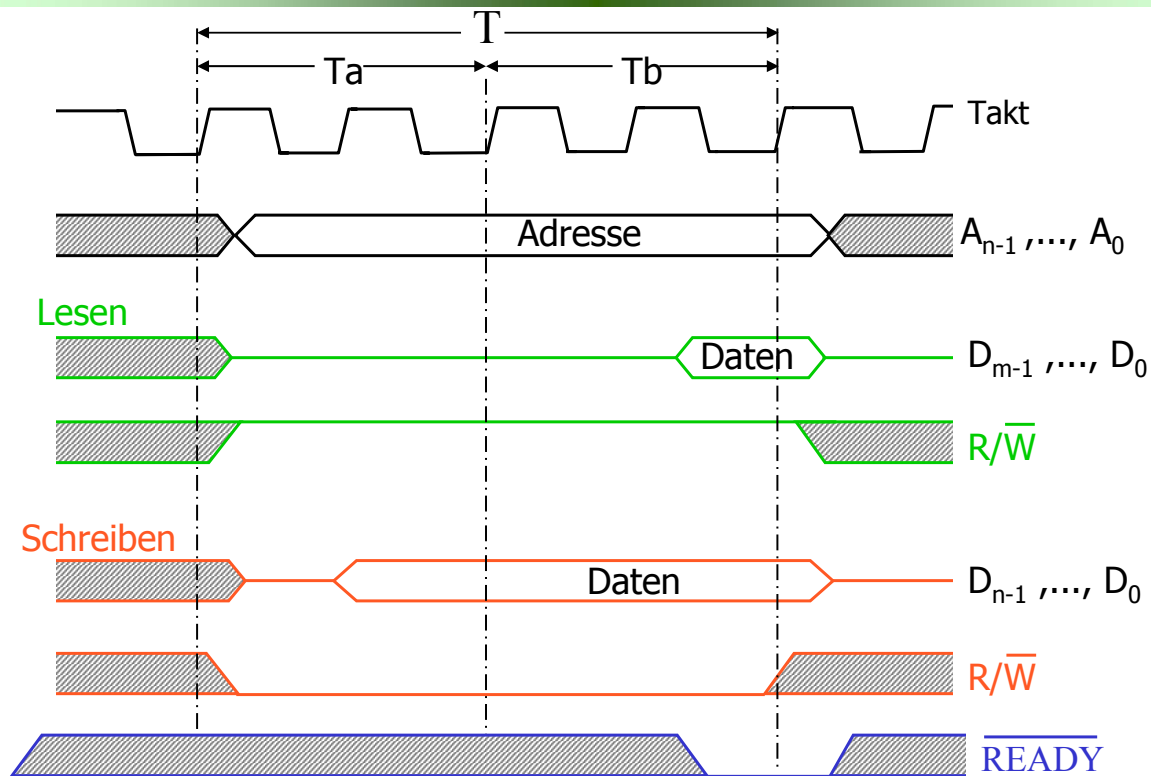


Synchroner Systembus

- Alle Vorgänge synchron zum Takt nach einem starren Muster ablaufen → **synchroner Systembus**
- Übergabe und Übernahme der Daten geschieht zu festgelegten Taktflanken
- Synchrone Busse in den Anfangsjahren der μ Pen
- **Nachteil:** Alle am Bus angeschlossenen Komponenten müssen strenge Zeitvorgaben erfüllen
 - ➔ Langsamste Komponente bestimmt die zulässige Geschwindigkeit des Busses, oder aber der Bus schließt den Einsatz von „schnellen“ Komponenten aus (z. B. keine schnelle Speicher ☹)



Semi-synchroner Systembus



Timing

Lesen ($R/\bar{W} = 1$):

- hinreichend schneller Speicher liefert seine Daten im letzten Taktzyklus von (T_b)
- Übernahme der Daten in den Prozessor durch die steigende Taktflanke des nächsten Buszyklus

Schreiben ($R/\bar{W} = 0$):

- Prozessor legt die Daten zu Beginn der zweiten Takthälfte von (T_a) auf den Systembus
- Übernahme der Daten in den Speicher durch die steigende Flanke von R/\bar{W}



Semi-synchroner Systembus

Um auch langsamere Speicher benutzen zu können:

Steuereingang $\overline{\text{READY}}$

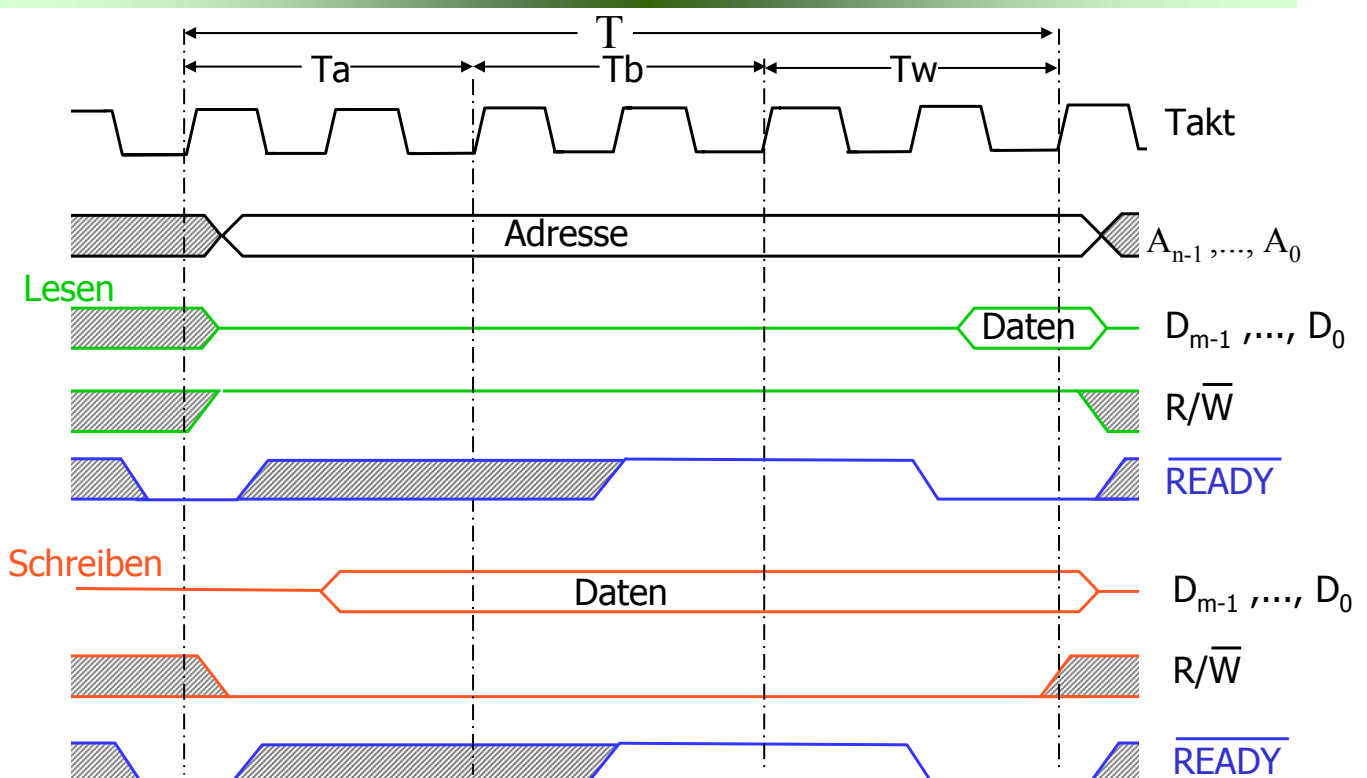
Buszyklus wird nur abgeschlossen, wenn rechtzeitig vor Ende von der Takthälfte (T_b) $\overline{\text{READY}} = 0$ ist

Ist $\overline{\text{READY}}$ am Ende des Buszyklus nicht 0

→ Es werden solange **Wartezyklen** (T_w , Dauer: z. B. halbe Buszykluslänge) eingefügt, bis $\overline{\text{READY}} = 0$ wird



Einfügen eines Wartezyklus



Semi-synchroner Systembus

- Moderne Prozessoren besitzen höhere Taktfrequenzen
- Schreibe- bzw. Lesezugriffe benötigt mehrere Taktzyklen
- **Steuereingänge (READY)** zur Synchronisation der Buszugriffe durch Einführung von Wartezyklen (*wait states*) → unterschiedlich schnelle Speicher und Geräte können individuell bedient werden
- Bezeichnung: Semi-synchrone Busse (Synchrone Busse mit Wartezyklen)
- **Timing:**
 - Adresse zu Beginn des Buszyklus (T_a) auf den Adressbus
 - Auswahl der Übertragungsrichtung wieder durch R/\overline{W}

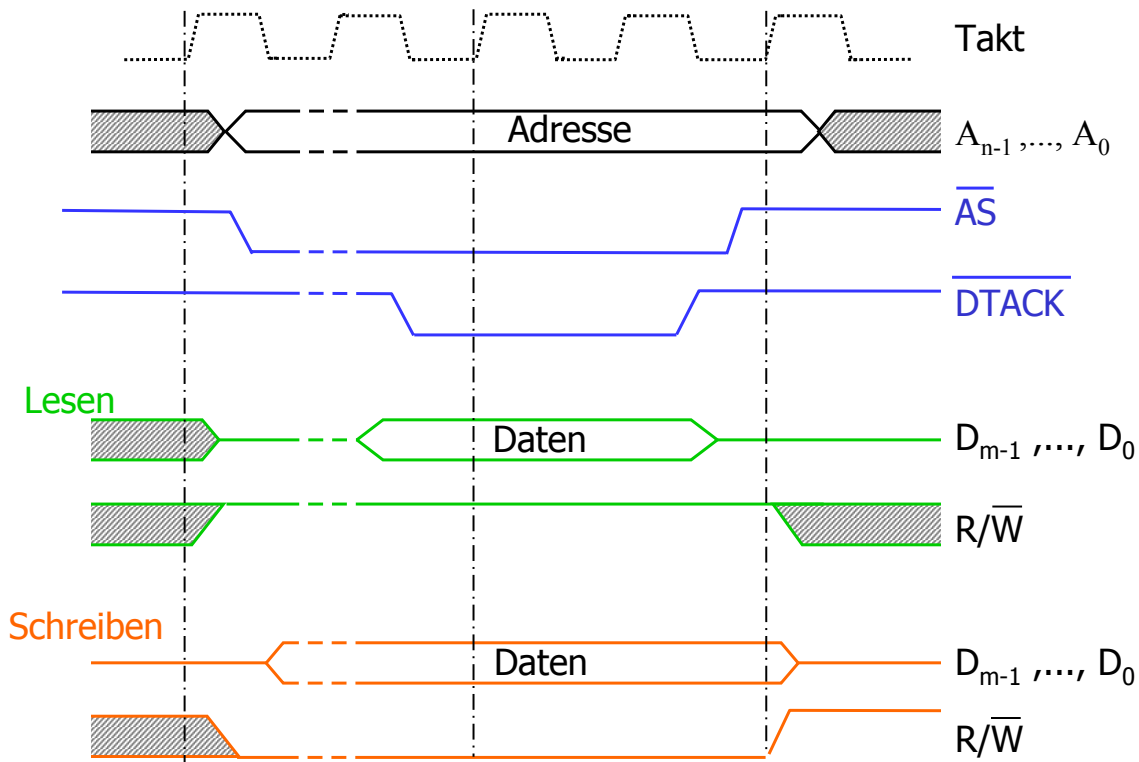


Semi-synchroner Systembus

- Bus ist immer noch synchron (streng am Takt orientiert), die Dauer eines Buszyklus ist jedoch nicht mehr fest, sondern in Vielfachen von Taktzyklen variierbar
 - ➔ **semi-synchroner Systembus**
- höherer Steueraufwand als synchroner Systembus
- Zeitverhalten ist auf verschieden schnelle Bausteine anpassbar
- Sind nur ausreichend schnelle Bausteine im System, kann **READY** z. B. fest auf 0 gelegt werden. Anderenfalls kann dieses Signal durch eine geeignete Verzögerungsschaltung (z. B. Monoflop) erzeugt werden



Asynchroner Systembus



Timing

- Auswahl der Übertragungsrichtung wieder durch R/\overline{W}
 - Mit $\overline{AS} = 0$ zeigt die CPU an, dass sie eine gültige Adresse auf den Adressbus gelegt hat
 - Mit $\overline{DTACK} = 0$ zeigt der Speicher (Komponente) an, dass er die Daten zur Verfügung gestellt (Lesen) oder übernommen (Schreiben) hat
 - Zwischen $\overline{AS} = 0$ und $\overline{DTACK} = 0$ kann eine beliebige Zeitspanne liegen
 - Wird $\overline{DTACK} = 0$, so nimmt die CPU die Adresse wieder vom Adressbus und setzt \overline{AS} wieder zu 1
 - Daraufhin nimmt der Speicher (Komponente) das Datum vom Datenbus und setzt \overline{DTACK} wieder zu 1
- ➔ vollständig asynchroner Übertragungsablauf, Anschluss von Komponenten mit fast beliebiger Zugriffszeit möglich



Asynchroner Systembus

- Zeitliche Abläufe werden durch Handshake-Signale gesteuert

Handshake-Signale:

- \overline{AS} (Address Strobe) von CPU
 - \overline{DTACK} (Data Transfer Acknowledge) von Speicher/Komponente
-
- Systemtakt spielt keine Rolle mehr für die Synchronisation der Übertragung (nur noch für die Synchronisation der Signale : synchrones Steuerwerk)



Beispiele

- Beispiele für asynchronen Systembus:
Motorola 68000 - 68030
- Beispiele für semi-synchronen Systembus:
Intel-Prozessoren, Motorola 68040



Multiplex-Bus

Nötig, falls bestimmte Gruppen von Signalen zeitlich hintereinander über dieselbe Busleitungen geschickt werden müssen (z. B. zur Einsparung von Busleitungen)

Beispiel:

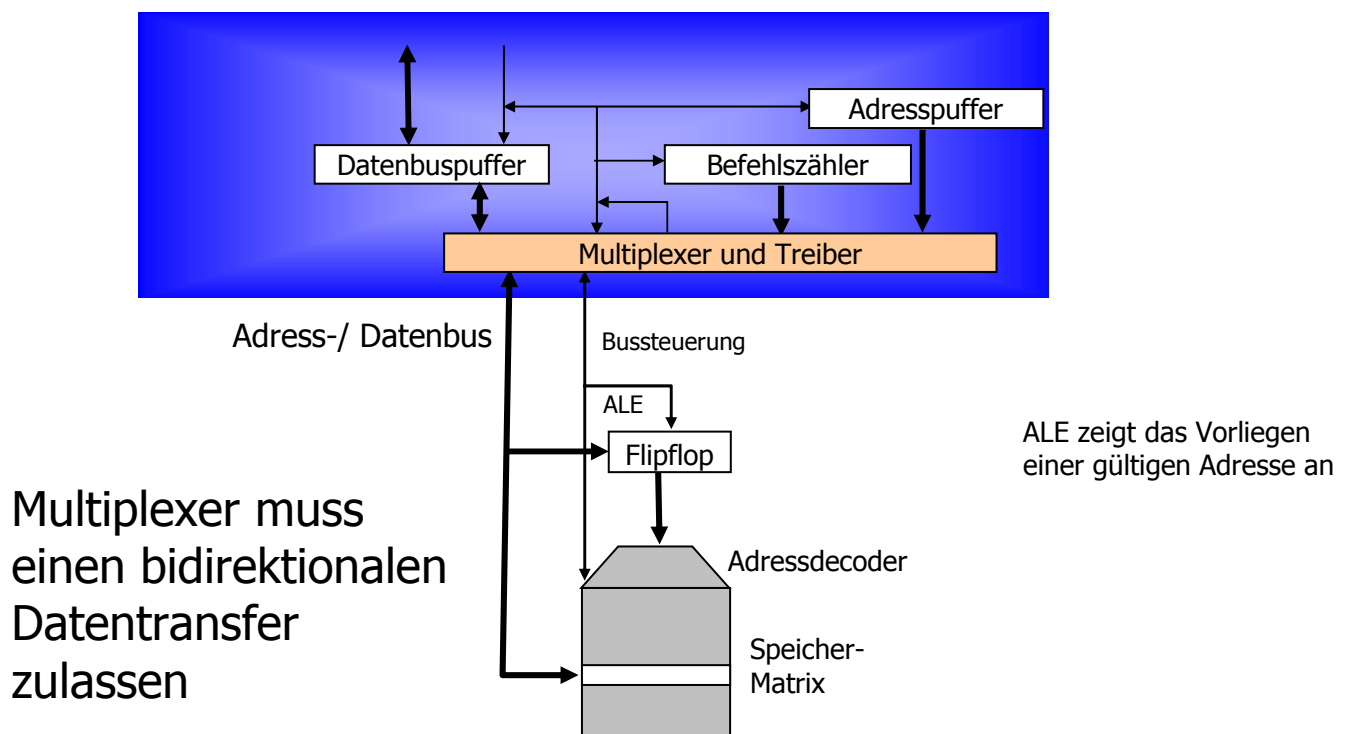
gemeinsamer Daten- und Adreßbus (Bsp.: PCI-Bus)

Die Adresse zur Übertragung eines Datums vom bzw. in den Speicher muss während der gesamten Zugriffszeit vorliegen

➔ Speichern der Adresse in Flipflops (Latches)

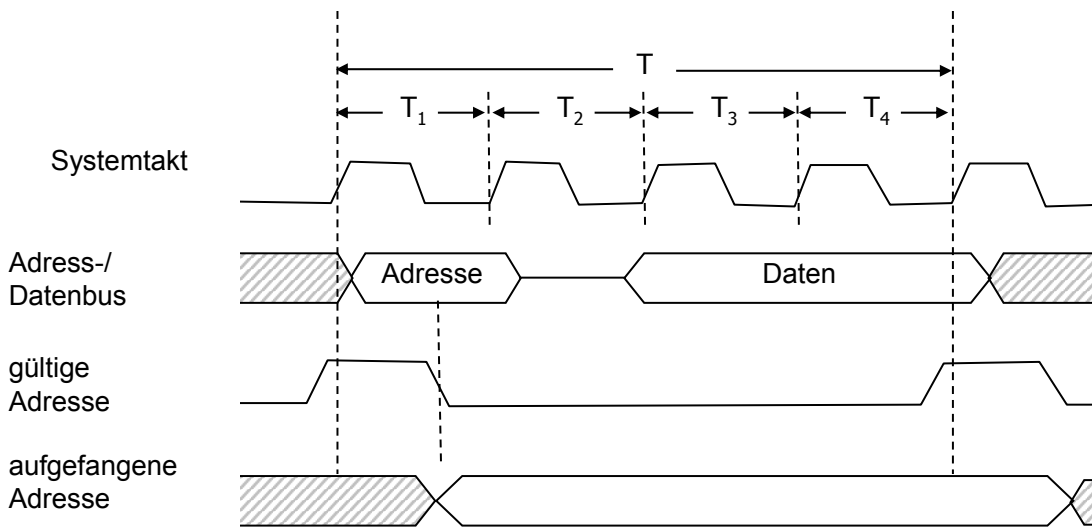


Multiplex-Busschnittstelle



Zeitverhalten des Multiplexbusses

Daten/Adressen-Multiplex-Betrieb



Daten/Adress-Multiplex-Betrieb

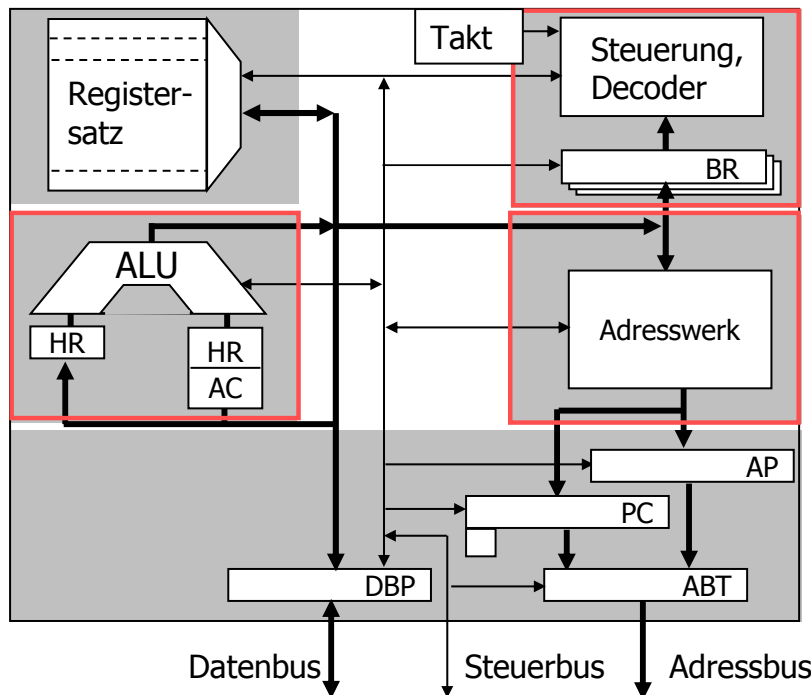
- gültige Adresse auf dem gemeinsamen Bus wird durch ein Signal angezeigt, z. B. \overline{AS} (Address Strobe) oder VMA (Valid Memory Address) oder \overline{ALE} (Address Latch Enable) oder ...
- mit der aktiven Flanke dieses Signals (im Beispiel fallende Flanke) wird die Adresse in ein vor den Speicher geschaltetes Adress-Flipflop übernommen → stabile Adresse am Speicher
- Danach kann der Bus umgeschaltet werden und nun die Daten über die gleichen Leitungen geschickt werden

Weitere Multiplex-Möglichkeiten:

- höher- und niederwertige Adressbits (z. B. bei dynamischen Speicherbausteinen)
- höher- und niederwertige Datenbits
- gemischte Formen (z. B. niederwertige Adress- und Datenbits, ...)



1.7 Befehlsabarbeitung



„Pipelining“



Befehlsabarbeitung

□ Holphase (Opcode fetch)

- Befehlszähler auf den Adressbus schalten
- OpCode in Datenbuspuffer übertragen und den Befehlszähler um 1 erhöhen
- OpCode im Datenbuspuffer ins Befehlsregister übertragen und Befehlszähler auf den Adressbus schalten

□ Decodierphase

- Befehl dekodieren
- Operand in Datenbuspuffer bringen
- Befehlszähler um 1 erhöhen



Befehlsabarbeitung

□ Ausführungsphase:

- Es werden keine weiteren Operanden/Adressen benötigt:
 - Befehl durch die ALU ausführen
 - Beispiele: Befehle, wie Erhöhen eines Registerinhaltes oder Austauschen zweier Registerinhalte
- Es werden weitere Operanden/Adressen benötigt:
 - benötigte Operanden holen und evtl. Berechnung der Operanden-Adressen
 - Befehl ausführen



Beispiel zur Befehlsabarbeitung (fiktiver Prozessor)

Assembler

Maschinencode

ADCA \$A7,X

high low
(6D A7)

addiere zum Akkumulator **A** den Inhalt der Speicherzelle, deren Adresse sich aus der Summe des Offsets **\$A7** und des Inhalts des **X**-Registers ergibt.

STA \$3F

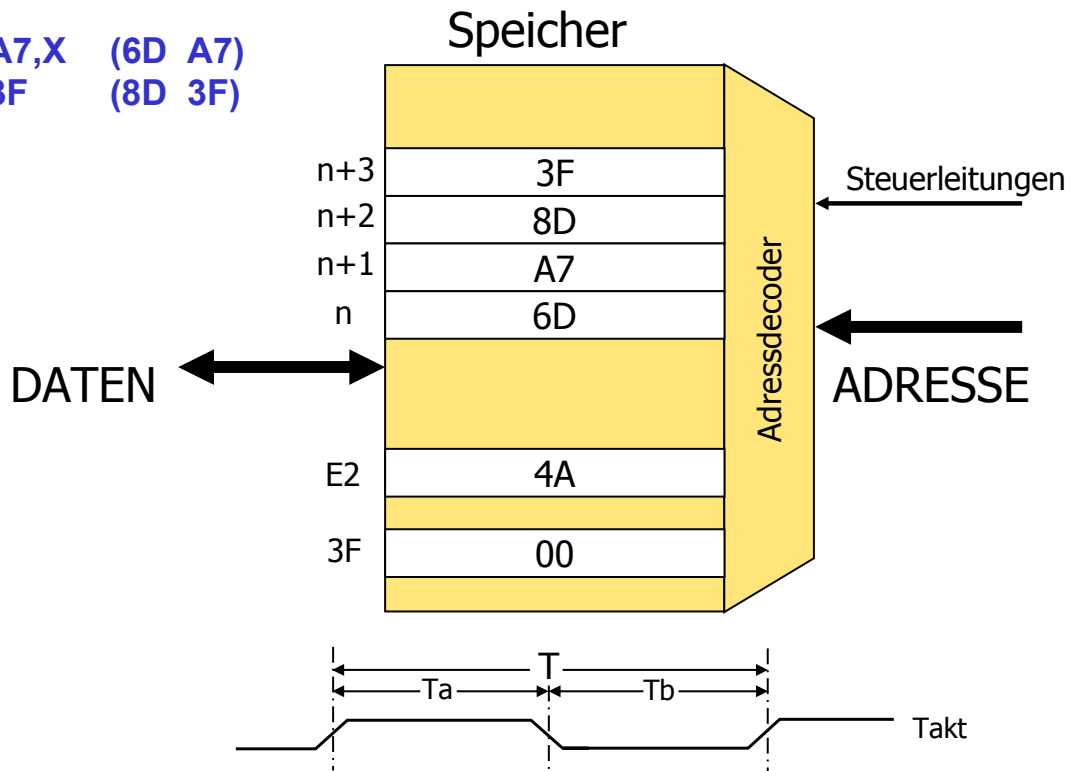
(8D 3F)

speichere **A** in der Speicherzelle mit der Adresse **\$3F**



Speicherbelegung vor der Befehlsausführung

ADCA \$A7,X (6D A7)
STA \$3F (8D 3F)



Anmerkung

Speicherformate eines zusammengesetzten Datums:

□ Big Endian:

Adresse eines zusammengesetzten Datums ist die Adresse des "most significant" Bytes (High Byte) (wie im Beispiel) verwendet z. B. bei Motorola 680XX, SUN Sparc, ...

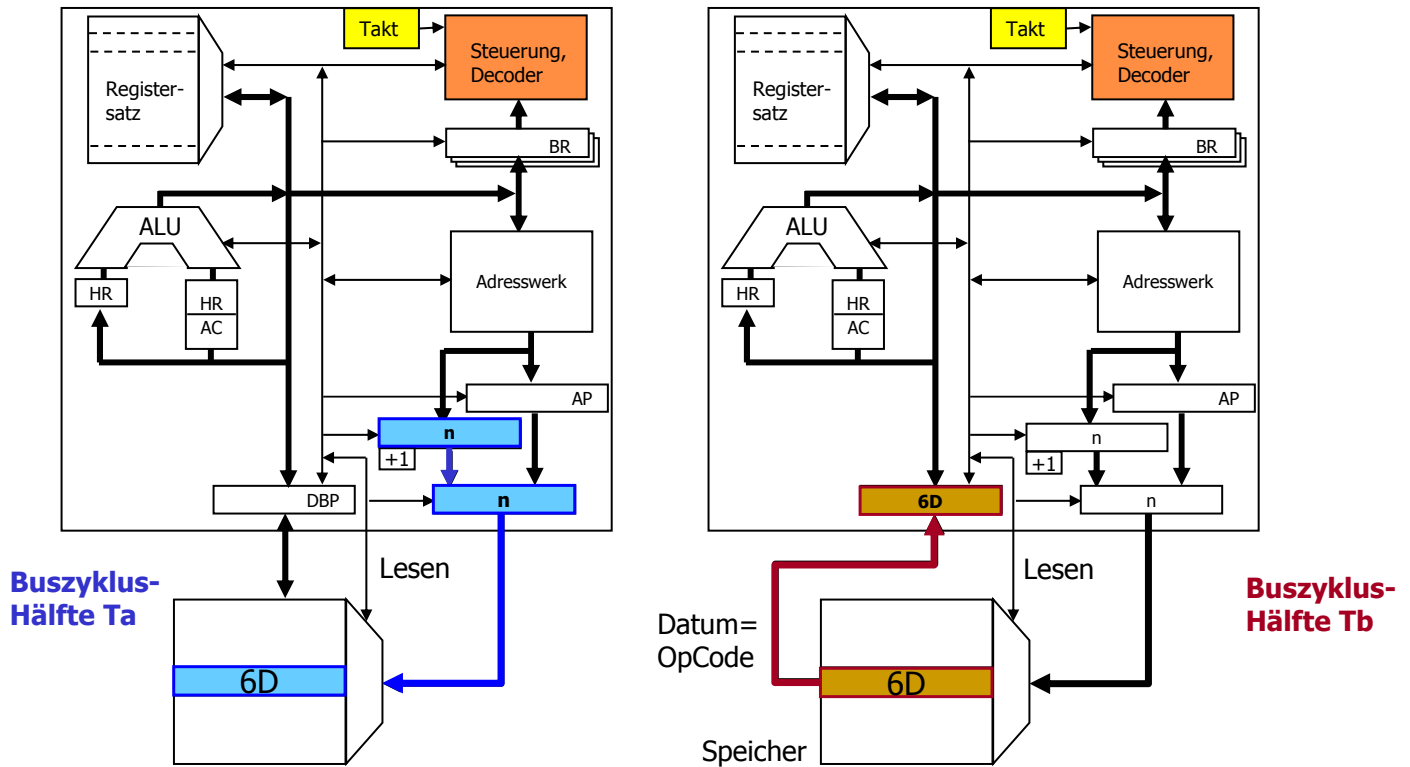
□ Little Endian:

Adresse eines zusammengesetzten Datums ist die Adresse des "least significant" Bytes (Low Byte) verwendet z. B. bei Intel 80XXX, Pentium, ...

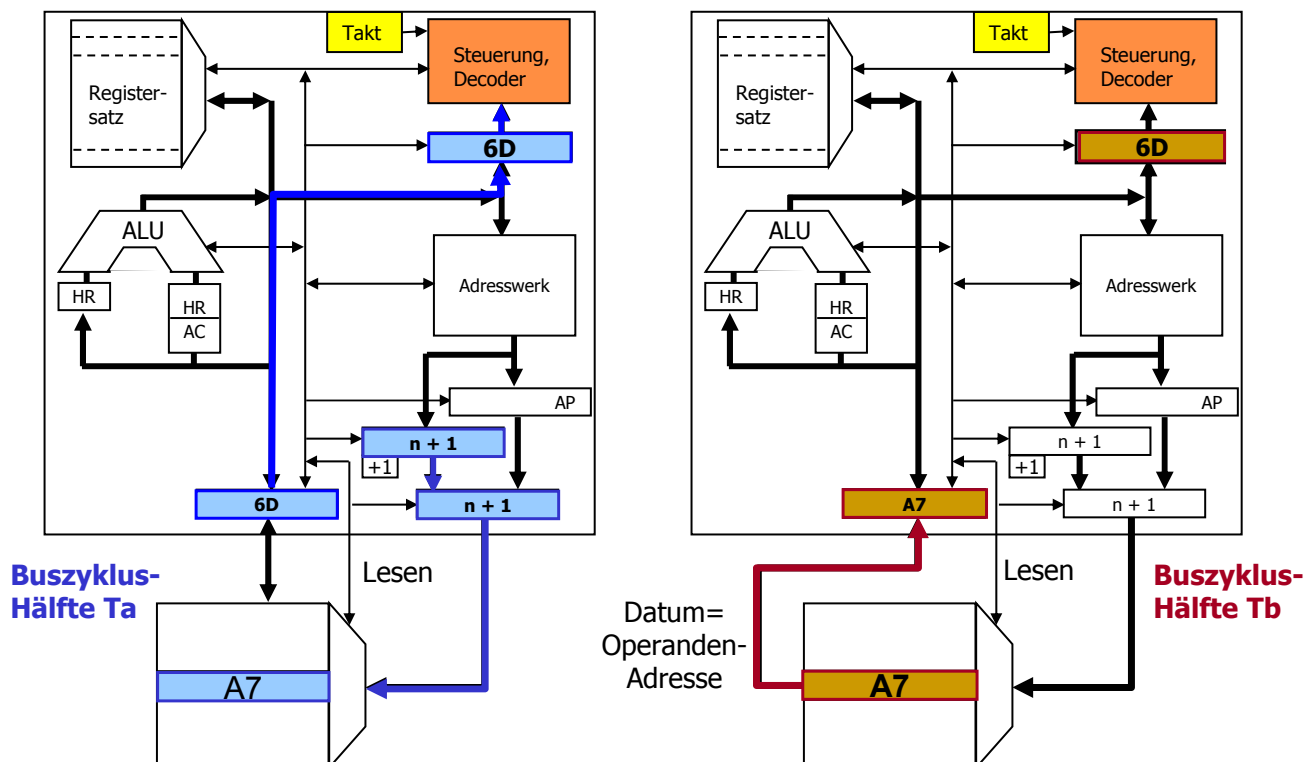


Die Holphase

(Prefetch, Befehlsregister BR noch durch vorigen Befehl belegt)

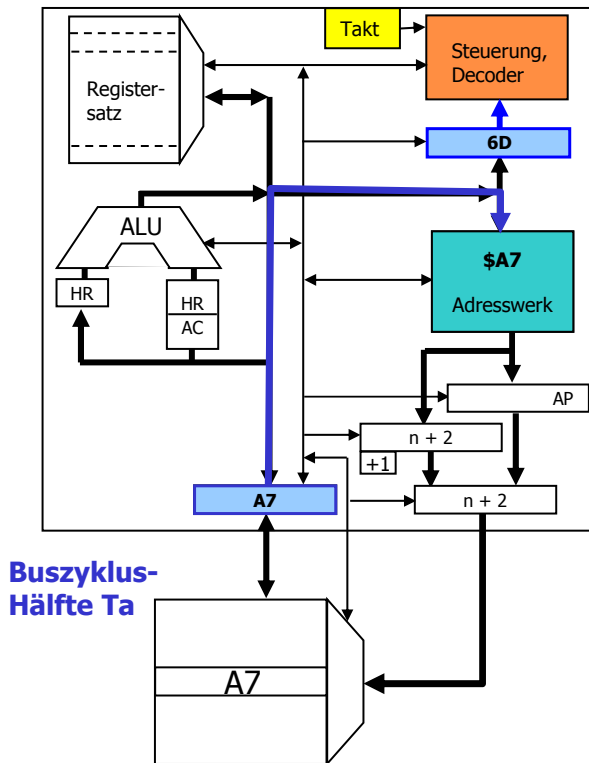


Ende der Holphase, Decodierphase

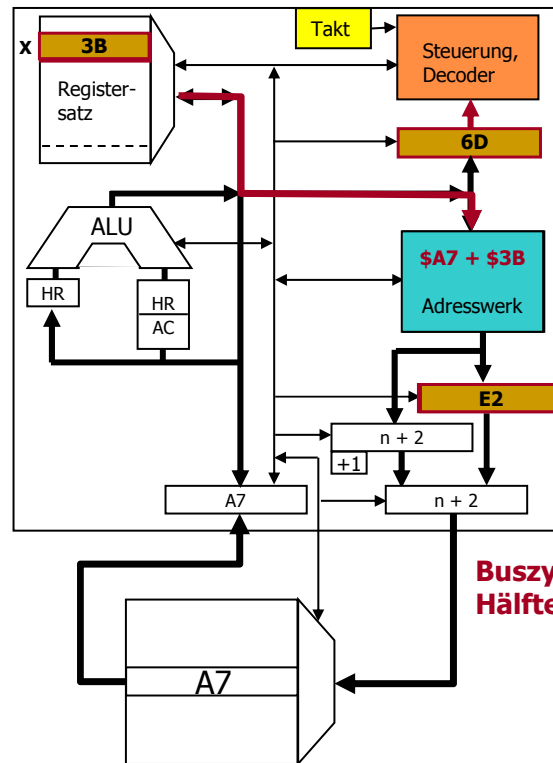


Ausführungsphase

Berechnung der Operandenadresse



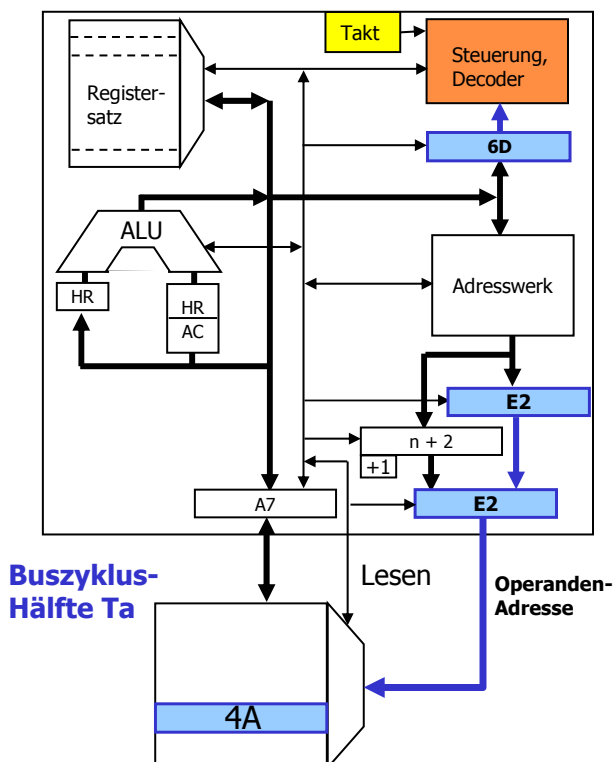
Buszyklus-Hälfte Ta



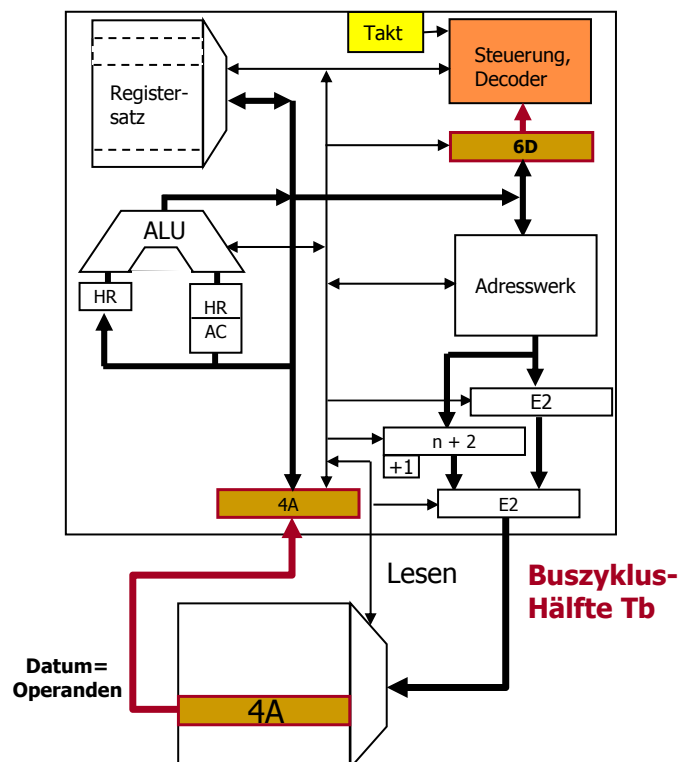
Buszyklus-Hälfte Tb



Ausführungsphase, Übertragung des Operanden



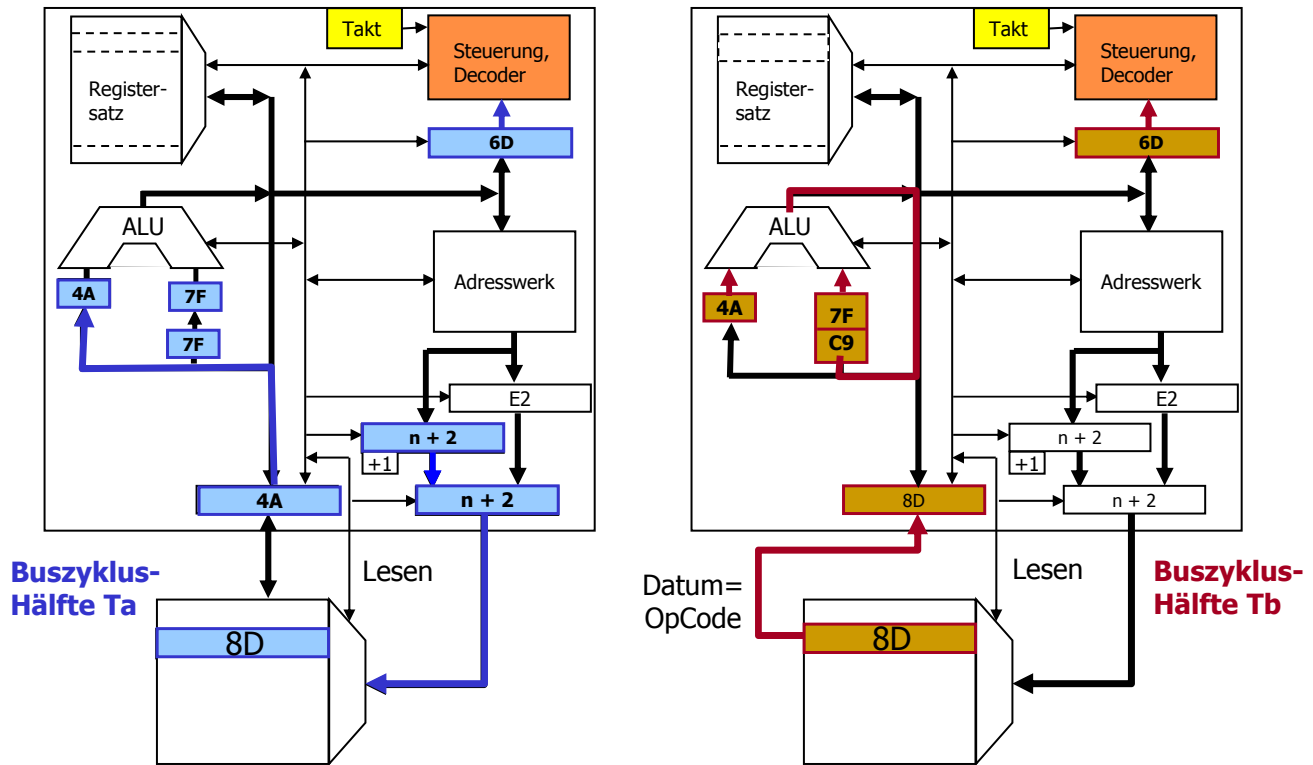
Buszyklus-Hälfte Ta



Buszyklus-Hälfte Tb



Ausführungsphase, Berechnung des Ergebnisses



Aufgabe

Führen Sie das Beispiel mit dem Befehl
STA \$3F (8D 3F) zu Ende

