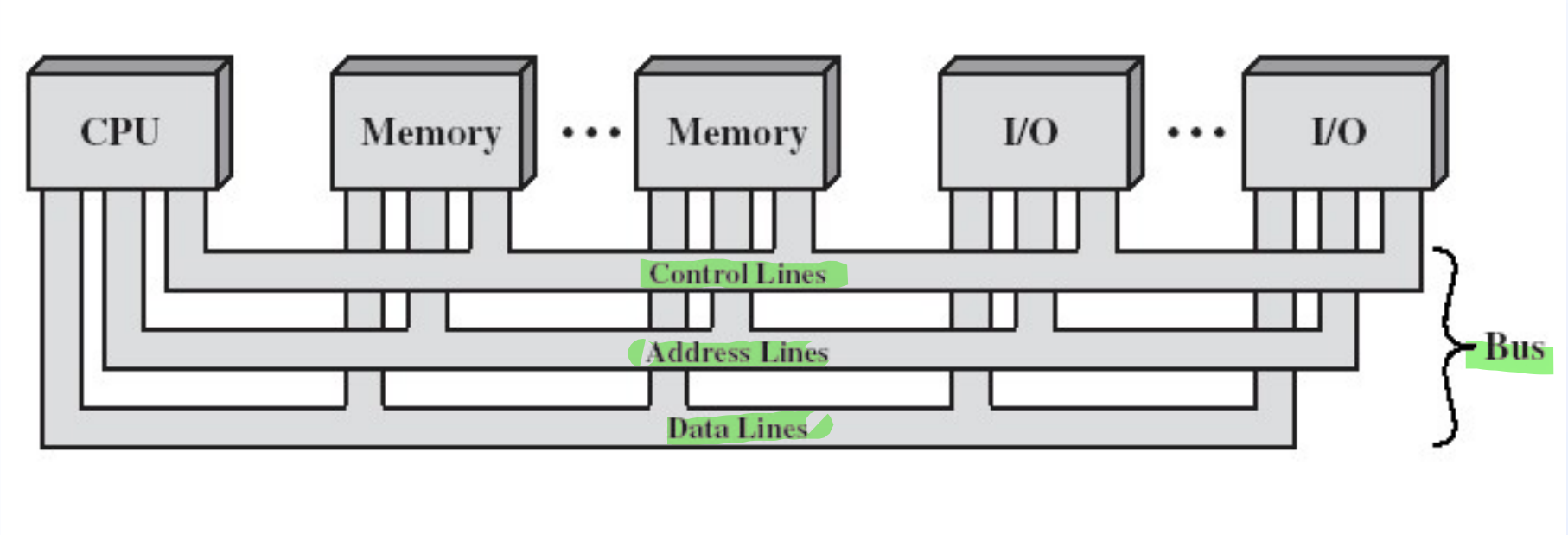
Architettura degli Elaboratori
a.a. 2021/2022

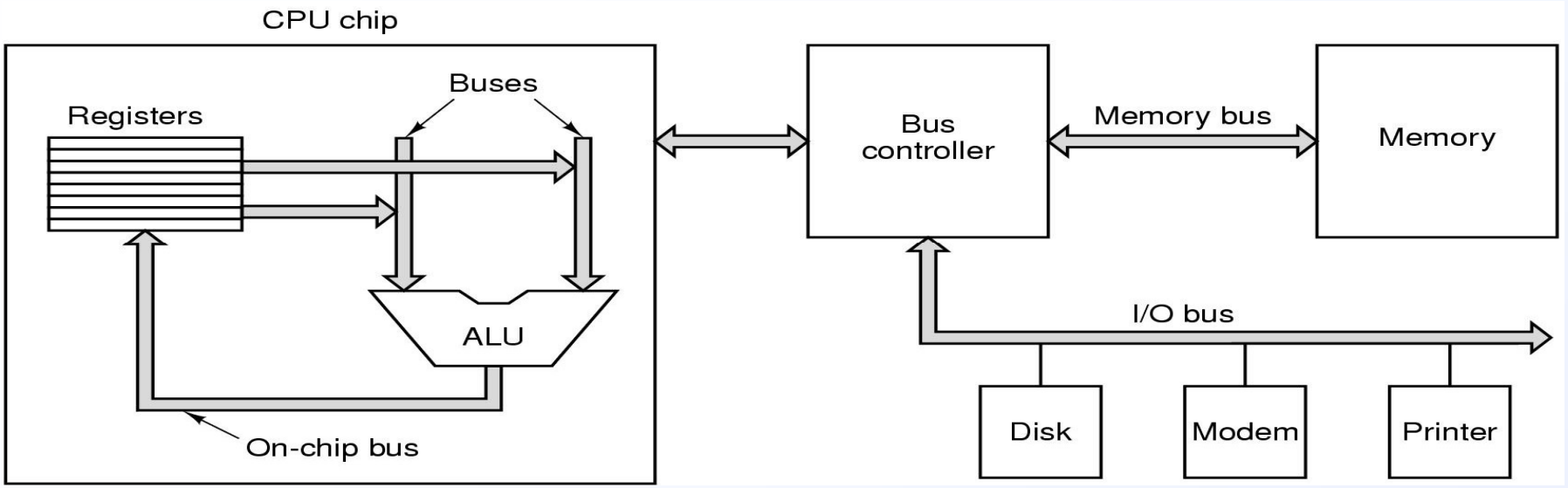
Il Bus

Connessioni di una CPU



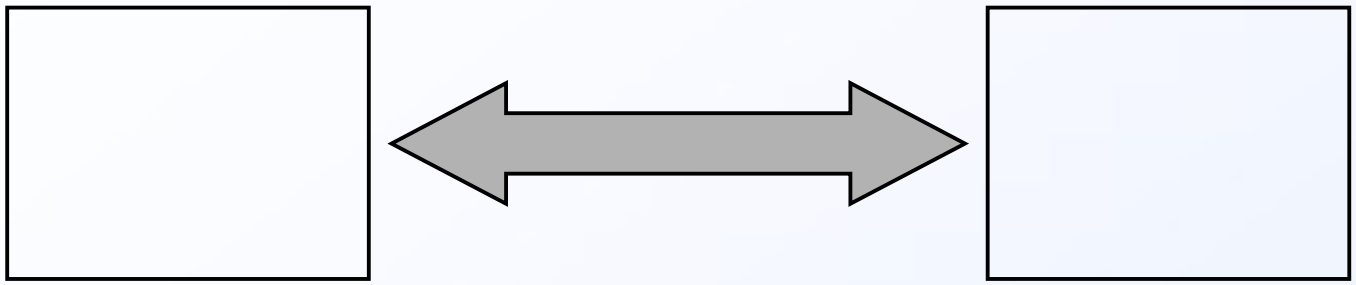
- Linee di controllo
- Linee dati
- Linee indirizzi

Bus



- Il bus è l'insieme di linee elettriche che collegano i moduli di un elaboratore
- Vari tipi di bus (con dimensioni fisiche molto diverse):
  - *Bus di sistema*: interconnette la CPU, le schede di I/O e la memoria
  - *Bus interni al chip*: mettono in comunicazione i moduli della CPU
  - *Bus SCSI*: connette le periferiche può avere una estensione di qualche metro
  - *Portante fisico di una rete ETHERNET*: connette calcolatori
- Possono essere considerati **tutti bus** in quanto le funzionalità ed i problemi da affrontare sono gli stessi

Bus



Un bus è costituito da un fascio di collegamenti elettrici

In genere viene rappresentato mediante una freccia larga ad indicare che le linee in esso contenute hanno funzionalità distinte (controllo, indirizzo, dati)

Affinché i moduli connessi dal bus siano in grado di comunicare è necessario che essi interagiscano con il bus secondo un insieme di regole ben definite.

L'insieme delle regole viene definito come il **protocollo del bus**

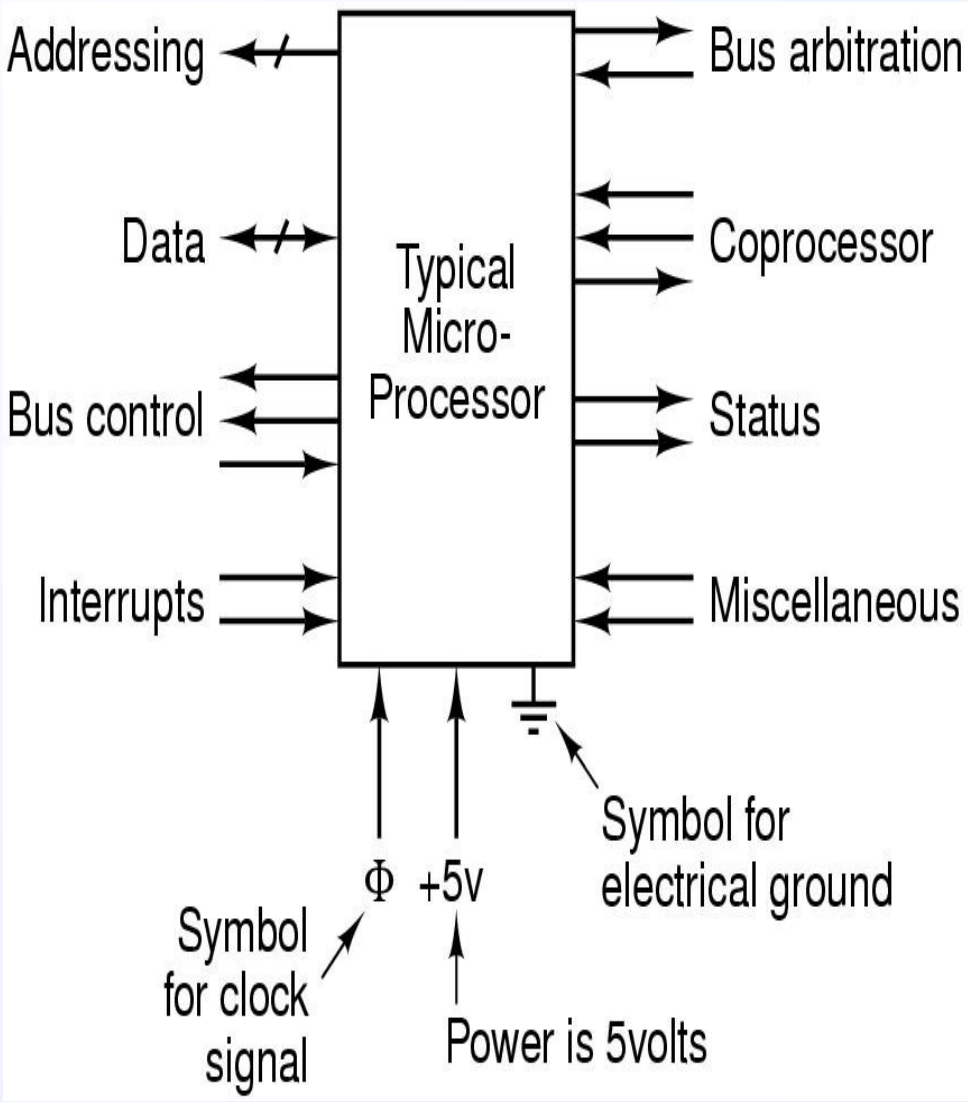
Bus

Le linee del bus possono essere linee di dati, di indirizzo e di controllo

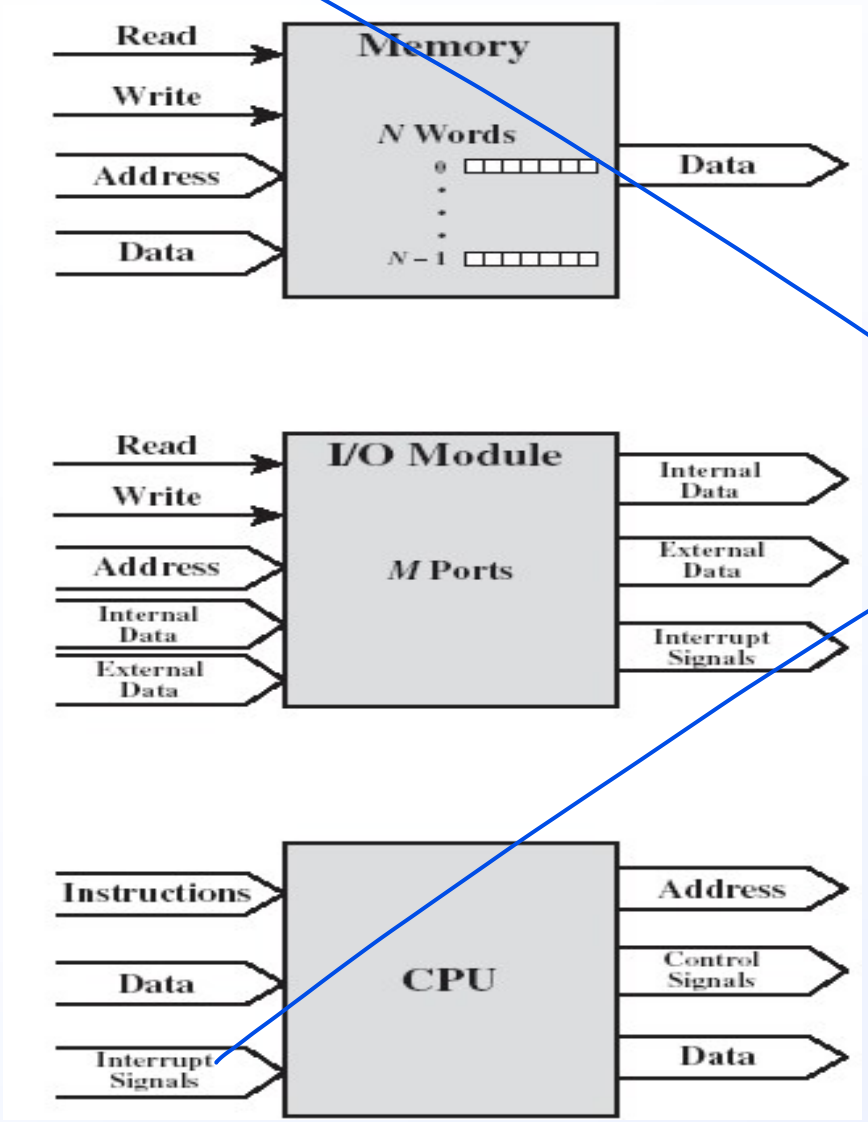
- **Linee di dati**: è detto *data bus.* Il numero di linee (larghezza del data bus) determina il numero di bit che possono essere trasmessi alla volta: ha un impatto sulle prestazioni del sistema.
- **Linee di indirizzo**: permettono di individuare la sorgente/ destinazione dei dati trasmessi sul data bus.
- **Linee di controllo**: controllano l'accesso e l'utilizzo delle linee di dati e di indirizzo

Connessioni di una CPU

- ***Indirizzo***: *n* piedini corrispondono a 2^n locazioni di memoria indirizzabili (valori tipici: *n* = 16, 20, 32, 64)
- ***Dati***: *n* piedini permettono di leggere/scrivere una parola di *n* bit con una sola operazione (valori tipici *n* = 8, 16, 32, 36, 64)
- ***Controllo***: regolano il flusso e la scansione dei dati verso e dal chip. Categorie principali:
  - *controllo del bus*
  - *interrupt*
  - *arbitraggio del bus*
  - *segnali del coprocessore*
  - *stato*
  - *varie*



Connessioni al bus



I dispositivi hanno connessioni diverse a seconda della loro natura

Dispositivi attivi e passivi

I dispositivi collegati ad un bus si dividono in

- ***attivi*** (***master***): possono decidere di iniziare un trasferimento, in genere sono collegati al bus per mezzo di un particolare chip detto *bus driver*
- ***passivi*** (***slave***): rimangono in attesa di richieste, in genere sono collegati per mezzo di un chip detto *bus receiver*
- i dispositivi che si comportano sia come master che come slave (es. la CPU) sono collegati attraverso un chip combinato: il *bus transceiver*

Wired-OR: OR booleano dei segnali contemporanei di richiesta di utilizzo del bus su una singola linea

Master	Slave	Example
CPU	Memory	Fetching instructions and data
CPU	I/O device	Initiating data transfer
CPU	Coprocessor	CPU handing instruction off to coprocessor
I/O	Memory	DMA (Direct Memory Access)
Coprocessor	CPU	Coprocessor fetching operands from CPU

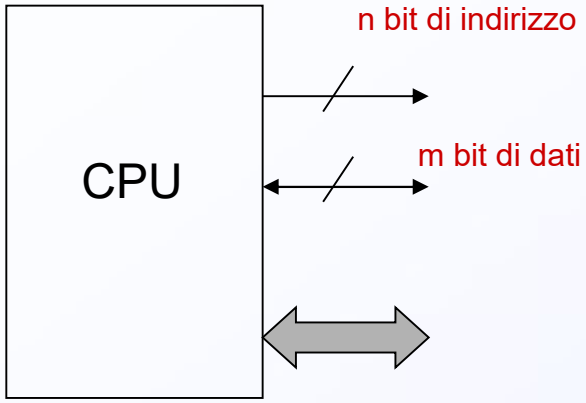
Problemi bus

Progettazione di un bus

I problemi principali nella progettazione di un bus riguardano

- ***Larghezza*** del bus: (numero di linee)
- ***Arbitraggio***: come scegliere tra due dispositivi che vogliono diventare contemporaneamente arbitri dello stesso bus
- ***Funzionamento*** del bus:come avviene il trasferimento dei bit

Larghezza del bus



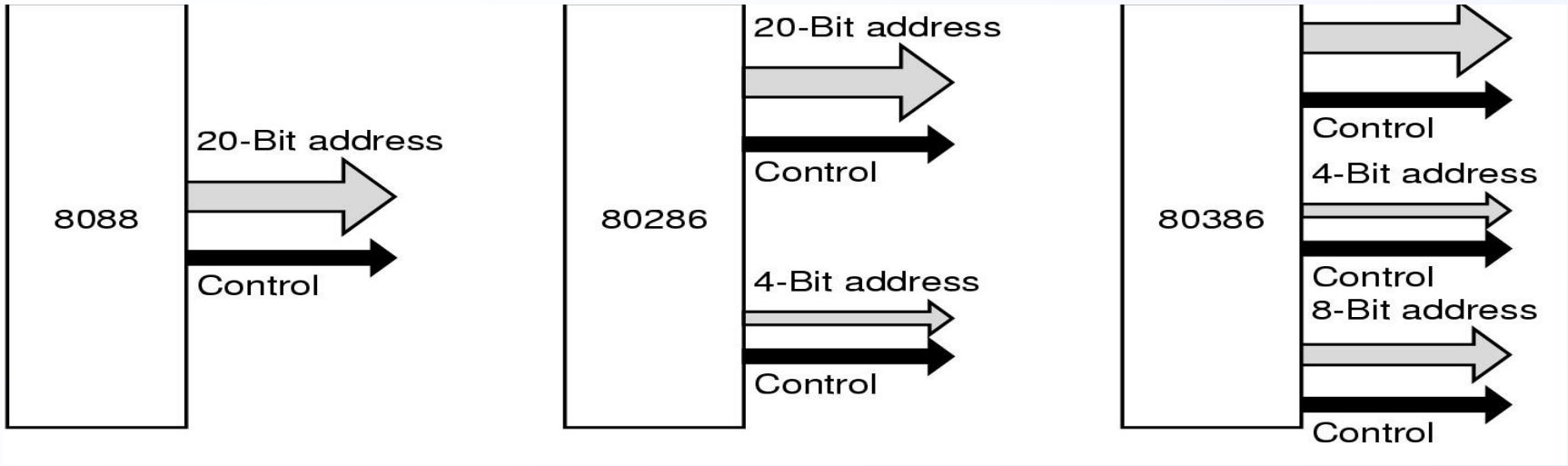
E' possibile indirizzare 2^n celle

e trasferire m bit alla volta

- Il numero delle linee utilizzate per trasferire gli indirizzi determinano la massima quantità della memoria indirizzabile

- Il numero delle linee utilizzate per il trasferimento di dati determinano la quantità di informazioni che è possibile trasferire con una singola operazione

Larghezza del bus



- Bus "larghi" sono più costosi di quelli stretti ma offrono una *banda* più larga e quindi maggiore velocità di trasferimento
- Maggiore velocità si ottiene aumentando la larghezza del bus (più bit/trasferimento) o diminuendo il ciclo di bus (più trasferimenti/secondo)
- Per ovviare ai problemi dati da bus molto larghi talvolta si opta per un *multiplexed bus*: le linee utilizzate per il trasferimento dei dati e degli indirizzi sono le stesse; prima si trasmettono gli indirizzi e poi i dati. Ovviamente il bus multiplexato è più lento.

Bus clocking

I bus si possono dividere in due categorie ben distinte:

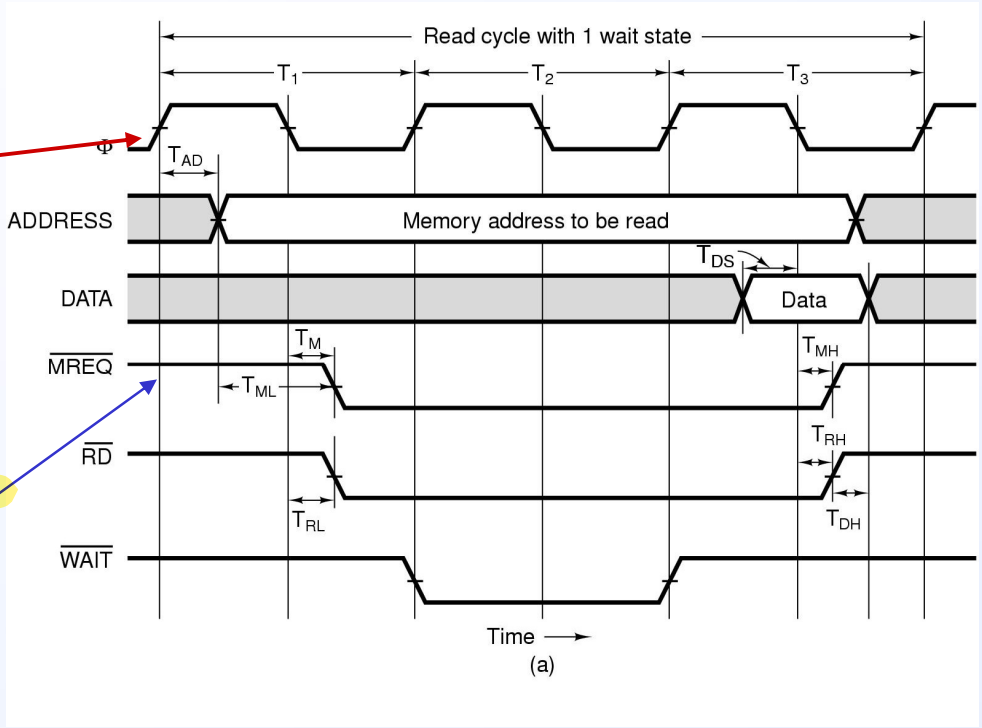
- ***Bus sincroni***: hanno una linea pilotata da un oscillatore, che ha una *determinata frequenza* (fino a qualche centinaia di MHz). Tutte le attività del bus richiedono un numero intero di questi cicli
- ***Bus asincroni***: non hanno un clock principale; i cicli del bus possono essere *della lunghezza necessaria* e non devono essere uguali per tutti i dispositivi.

Bus sincrono

Il bus sincrono ha una linea pilotata da un **oscillatore**, con una frequenza fino ad alcune centinaia di Mhz; ogni attività richiede un numero intero di cicli.

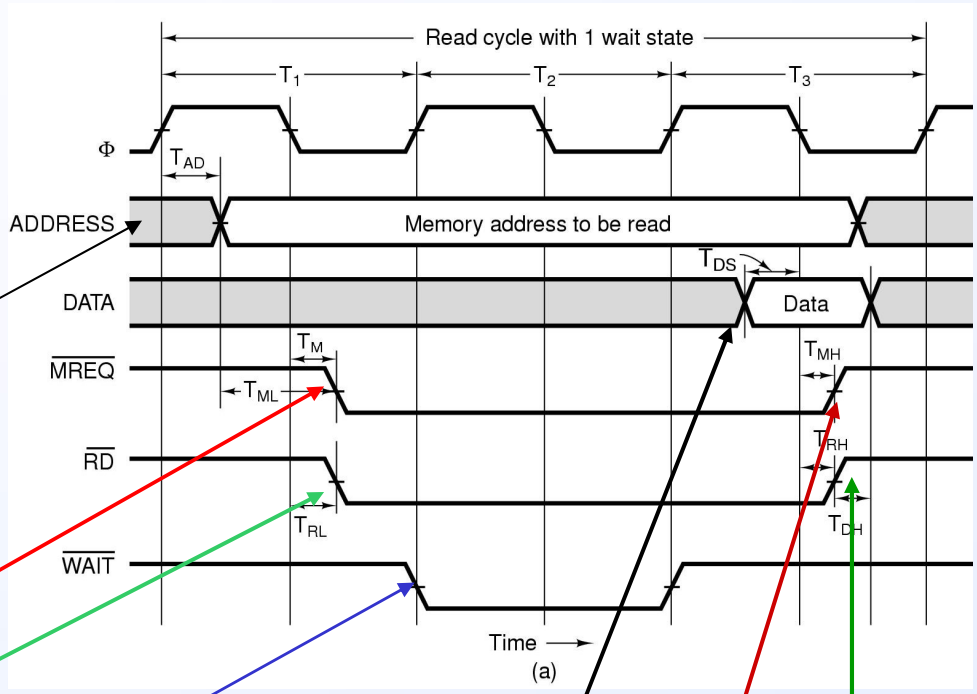
La durata delle fasi è nota ad entrambi i partecipanti (master e slave) e l'unica incognita è l'inizio della comunicazione.

La linea **MREQ** indica l'inizio della comunicazione



Bus sincrono – Ciclo di lettura

Supponiamo di avere un clock di 100 Mhz (ciclo di 10 nsec.) e che la lettura dalla memoria richieda 15 nsec da quando l'indirizzo è stabile



La **CPU** mette l'indirizzo sul bus dopo un certo tempo necessario per stabilizzarsi (T_{AD}), l'indirizzo rimane disponibile

A linee di indirizzo stabili la **CPU** attiva i segnali di richiesta di inizio comunicazione e il segnale di lettura

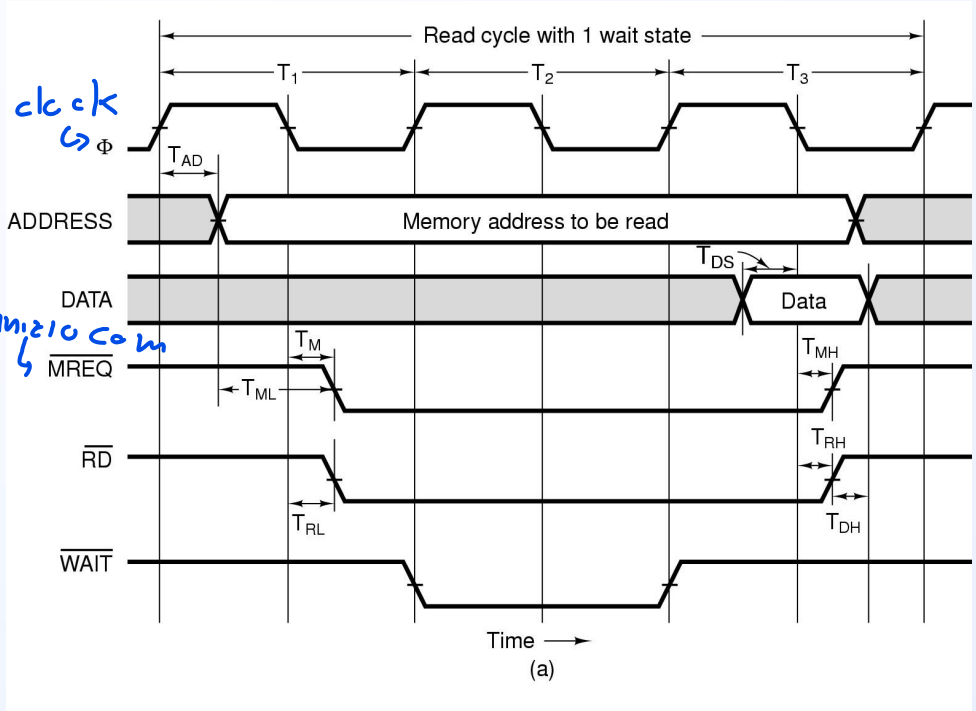
Per avvertire la CPU di non aspettare i dati, all'inizio del ciclo T_2, la **memoria** attiva il segnale di attesa dati

La **memoria** mette i dati sulla linea dati

Dopo aver letto i dati la **CPU** disattiva i segnali di richiesta e di lettura

Bus sincrono – Ciclo di lettura

Nella specifica di temporizzazione occorre tenere conto di alcuni parametri temporali



T_{AD}: intervallo di tempo tra il fronte di salita del clock e l'istante in cui sono valide le linee degli indirizzi (max)

T_{ML}: intervallo di tempo dall'istante in cui sono valide le linee degli indirizzi prima dell' effettiva segnalazione di MREQ (min)

T_M/T_{RL}: intervallo di tempo tra il fronte di discesa del clock e il fronte di discesa di MREQ/RD (max)

T_{DS}: tempo di setup per le linee dati prima del fronte di discesa del clock (min)

T_{DH}: hold time tra il fronte di salita di MREQ e la rimozione da parte dello slave dei dati (min)

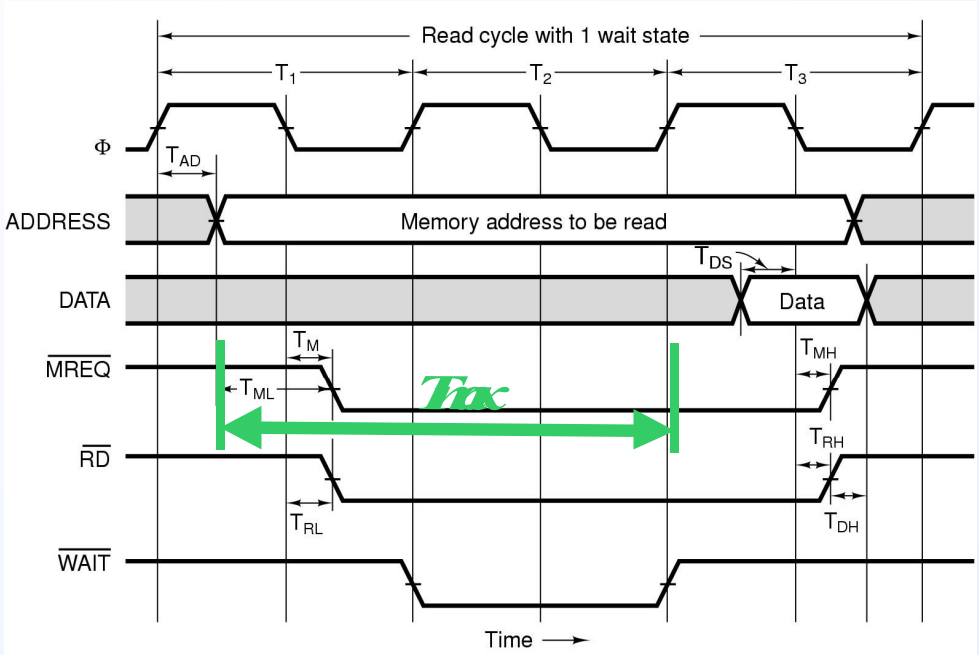
T_{MH}/T_{RH}: ritardo di MREQ/RD dal fronte di discesa del clock (max)

Bus sincrono – Ciclo di lettura

Qual è il tempo massimo che la memoria ha a disposizione per produrre i dati dal momento in cui appare l'indirizzo?

da address a data

$$T_{max} = T_1 + T_2 + T_3/2 - T_{DS} - T_{AD}$$



Simbolo	Parametro	Min	Max	Unità
T_{AD}	Ritardo dell'indirizzo rispetto al clock		4	n sec.
T_{ML}	Indirizzo stabile prima di MREQ	2		n sec.
T_M	Ritardo di MREQ dal fronte di discesa di ϕ		3	n sec.
T_{RL}	Ritardo di RD dal fronte di discesa di		3	n sec.
T_{DS}	Tempo di assestamento dati.	2		n sec.
T_{MH}	Ritardo di MREQ dal fronte di discesa di ϕ		3	n sec.
T_{RH}	Ritardo di RD dal fronte di discesa di ϕ in T3		3	n sec.
T_{DH}	Tempo di hold dei dati dalla Negazione di RD	0		n sec.

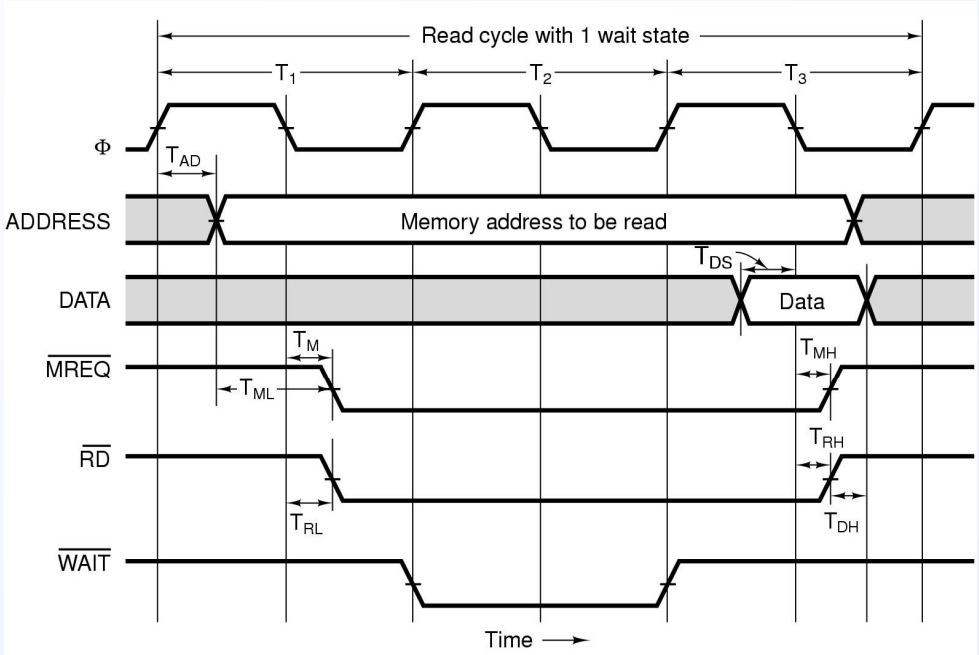
Bus sincrono – Ciclo di lettura

Le specifiche del bus impongono dei vincoli sul valore di alcuni dei parametri temporali, ad esempio:

T_{AD} < 4 nsec T_{ML} > 2 nsec

T_M < 3 nsec T_{DS} > 2 nsec

T_{MH} < 3 nsec T_{DH} > 0 nsec

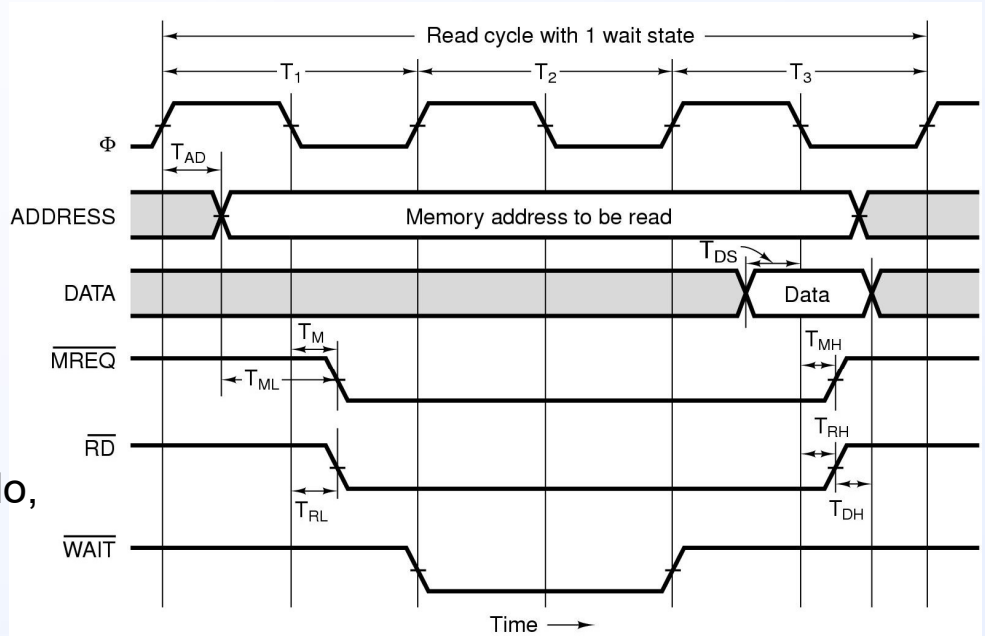


Simbolo	Parametro	Min	Max	Unità
T_{AD}	Ritardo dell'indirizzo rispetto al clock		4	n sec.
T_{ML}	Indirizzo stabile prima di MREQ	2		n sec.
T_M	Ritardo di MREQ dal fronte di discesa di ϕ		3	n sec.
T_{RL}	Ritardo di RD dal fronte di discesa di		3	n sec.
T_{DS}	Tempo di assestamento dati.	2		n sec.
T_{MH}	Ritardo di MREQ dal fronte di discesa di ϕ		3	n sec.
T_{RH}	Ritardo di RD dal fronte di discesa di ϕ in T3		3	n sec.
T_{DH}	Tempo di hold dei dati dalla Negazione di RD	0		n sec.

Con un ciclo di clock di 10 nsec
$T_{max} = 10 + 10 + 5 - T_{AD} - T_{DS} =$
$= (25 - 4 - 2)$ nsec = 19 nsec

Bus sincrono – Ciclo di lettura

Con un ciclo di bus di 30 nsec. , quanto tempo ha la RAM per inserire il dato sul bus *dopo che il segnale MREQ ha assunto il valore 0*?



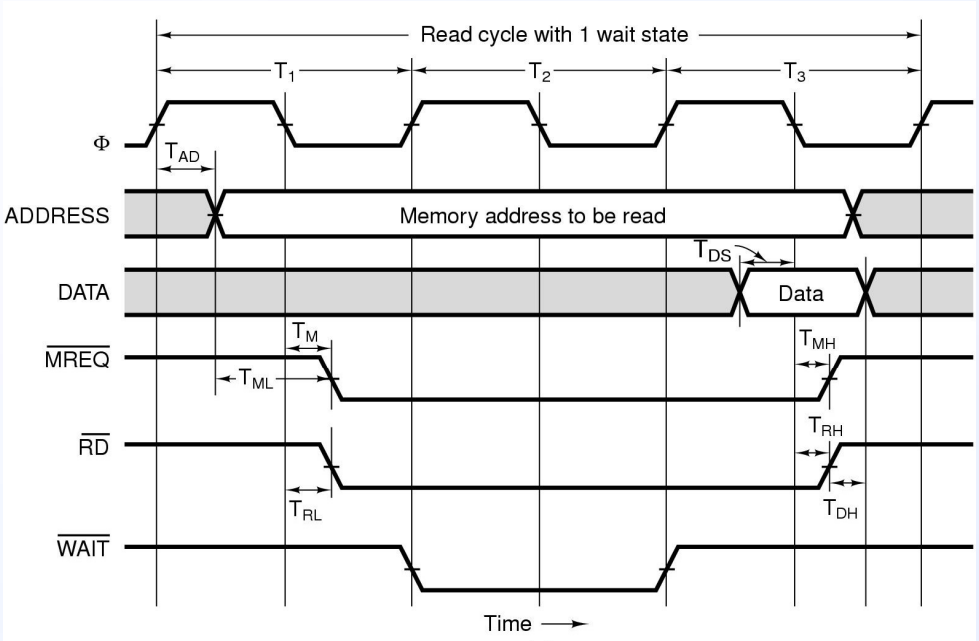
MREQ assume il valore 0 al più tardi dopo 18 nsec ($T_1/2+T_M$=15 + 3) dall'inizio del primo ciclo di clock.
I dati devono essere presenti sul bus almeno 2 nsec (T_{DS}) prima del fronte di discesa del terzo ciclo,
Quindi il tempo a disposizione della RAM e':
75 – (18 + 2) nsec. = 55 nsec.

Simbolo	Parametro	Min	Max	Unità
T_{AD}	Ritardo dell'indirizzo rispetto al clock		4	n sec.
T_{ML}	Indirizzo stabile prima di MREQ	2		n sec.
T_M	Ritardo di MREQ dal fronte di discesa di ϕ		3	n sec.
T_{RL}	Ritardo di RD dal fronte di discesa di		3	n sec.
T_{DS}	Tempo di assestamento dati.	2		n sec.
T_{MH}	Ritardo di MREQ dal fronte di discesa di ϕ		3	n sec.
T_{RH}	Ritardo di RD dal fronte di discesa di ϕ in T3		3	n sec.
T_{DH}	Tempo di hold dei dati dalla Negazione di RD	0		n sec.

Bus sincrono – Ciclo di lettura

e se il ciclo di bus fosse di 10 nsec., con max T_M = 3 nsec., min T_{DS} = 4 nsec? Si potrebbe usare un chip di RAM di 10 nsec?

La RAM avrebbe ora 25 – (8+4) = 13 nsec. per inserire il dato, se il chip di RAM risponde sempre in 10 nsec., il sistema può funzionare



Simbolo	Parametro	Min	Max	Unità
T_{AD}	Ritardo dell'indirizzo rispetto al clock		4	n sec.
T_{ML}	Indirizzo stabile prima di MREQ	2		n sec.
T_M	Ritardo di MREQ dal fronte di discesa di ϕ		3	n sec.
T_{RL}	Ritardo di RD dal fronte di discesa di		3	n sec.
T_{DS}	Tempo di assestamento dati.	2		n sec.
T_{MH}	Ritardo di MREQ dal fronte di discesa di ϕ		3	n sec.
T_{RH}	Ritardo di RD dal fronte di discesa di ϕ in T3		3	n sec.
T_{DH}	Tempo di hold dei dati dalla Negazione di RD	0		n sec.

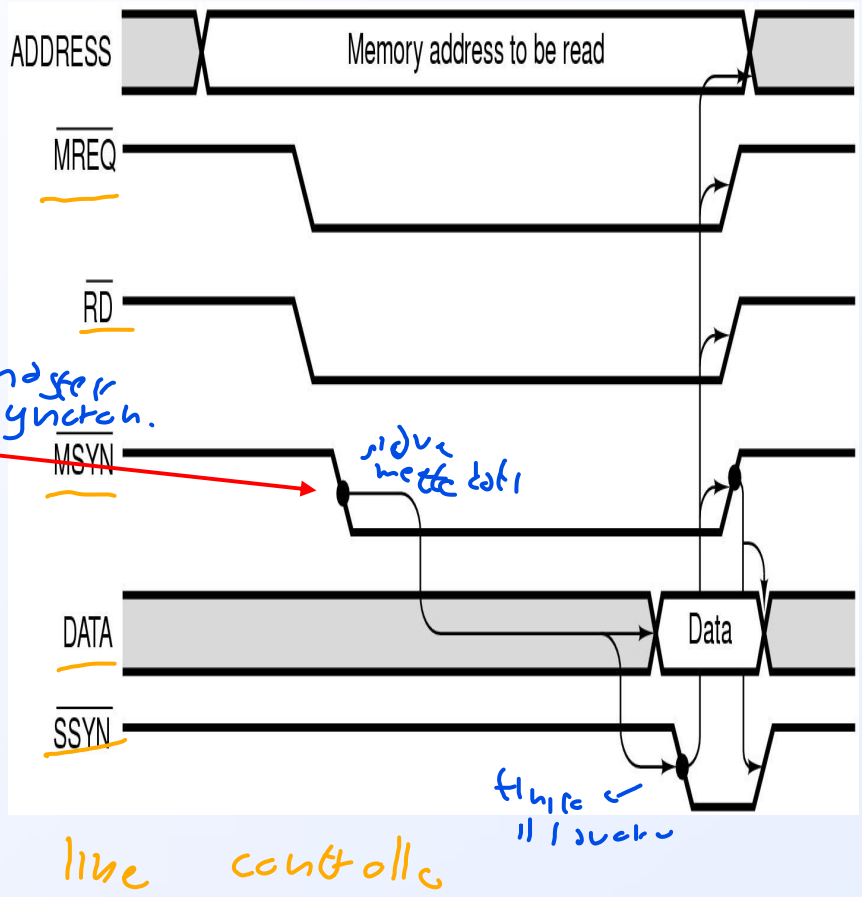
vedi su notion

Bus asincrono

Il bus sincrono deve adattarsi al dispositivo più lento

In questi casi funziona meglio un bus asincrono, cioè senza master clock

Attivati i segnali di accesso alla memoria e lettura, il master del bus attiva il segnale Master SYNchronization



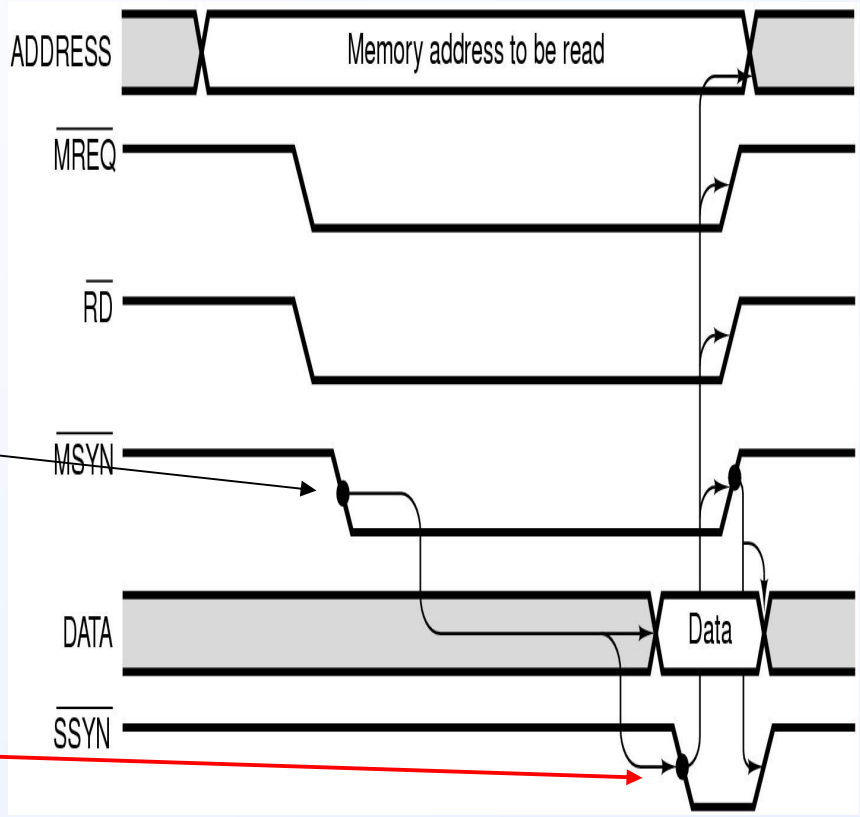
Bus asincrono

Il bus sincrono deve adattarsi al dispositivo più lento

In questi casi funziona meglio un **bus asincrono**, cioè senza master clock

Attivati i segnali di accesso alla memoria e lettura, attiva il segnale Master SYNchronization

Lo slave esegue il suo lavoro all'attivazione del segnale e quindi terminatolo attiva il segnale di Slave SYNchronization



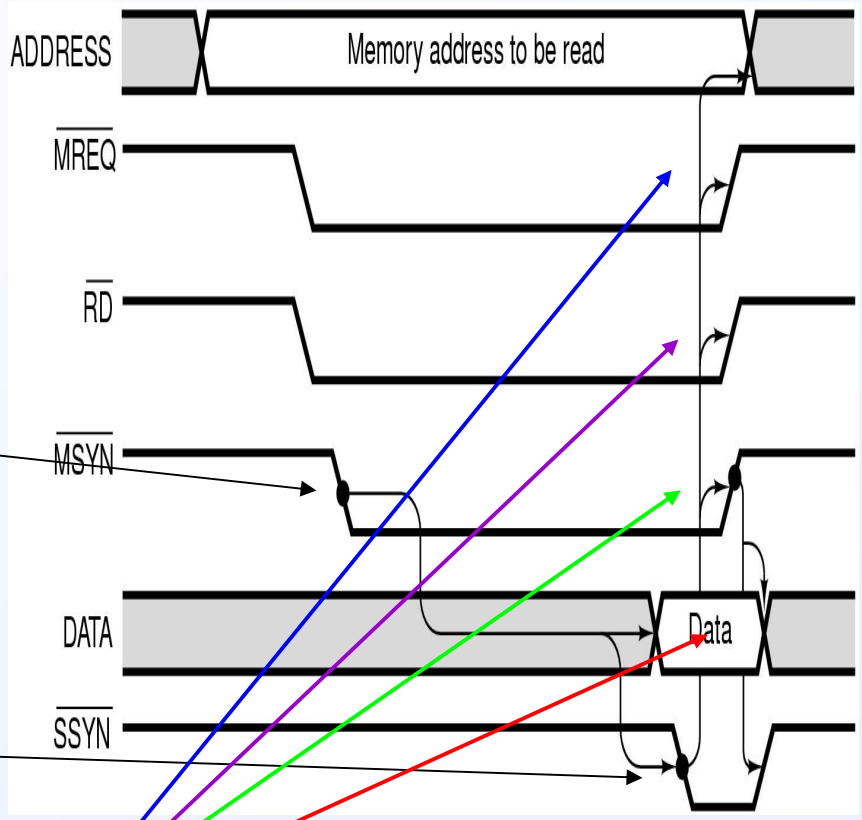
Bus asincrono

Il bus sincrono deve adattarsi al dispositivo più lento

In questi casi funziona meglio un **bus asincrono**, cioè senza master clock

Attivati i segnali di accesso alla memoria e lettura, attiva il segnale Master SYNchronization

Lo slave esegue il suo lavoro all'attivazione del segnale e quindi terminatolo attiva il segnale di Slave SYNchronization



I dati vengono memorizzati dal master che nega i segnali di accesso alla memoria, lettura e Master SYNchronization

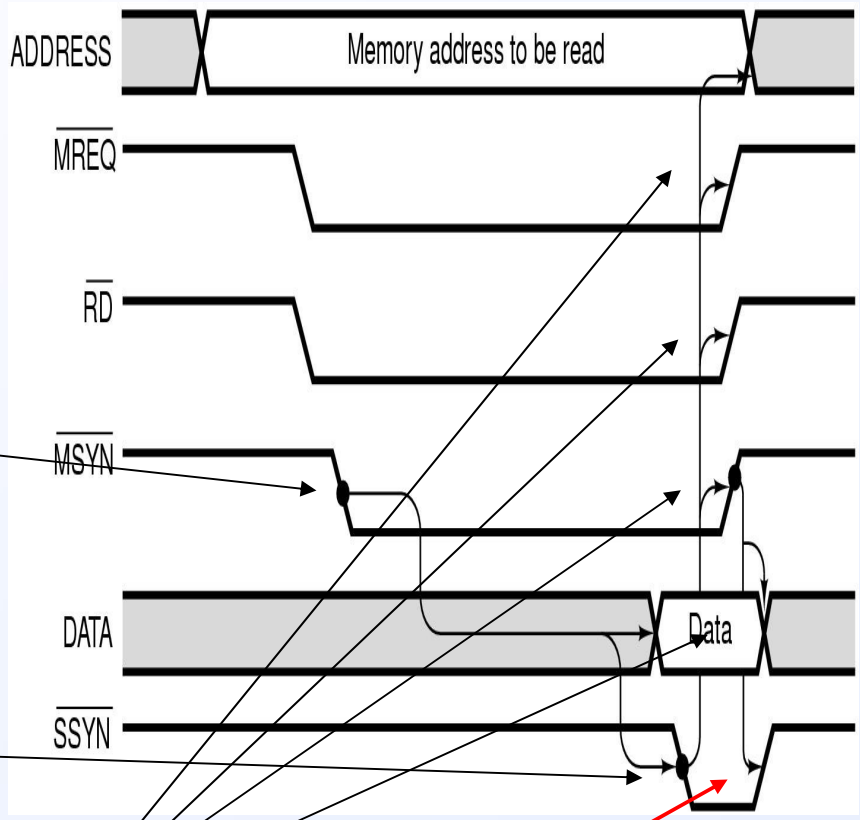
Bus asincrono

Il bus sincrono deve adattarsi al dispositivo più lento

In questi casi funziona meglio un **bus asincrono**, cioè senza master clock

Attivati i segnali di accesso alla memoria e lettura, attiva il segnale Master SYNchronization

Lo slave esegue il suo lavoro all'attivazione del segnale e quindi terminatolo attiva il segnale di Slave SYNchronization



I dati vengono memorizzati dal master che nega i segnali di accesso alla memoria, lettura e Master SYNchronization

Full handshake

Lo slave dopo aver visto la negazione del segnale di master synchronization, nega a sua volta il segnale di Slave SYNnchronization

Sincrono vs Asincrono

Bus sincrono

- Vantaggi
  - Realizzazione slave semplice
  - Se la durata di un' operazione è fissa non occorre una linea di wait
- Svantaggi
  - Durata di una operazione di comunicazione deve necessariamente avere una durata pari ad un numero intero di cicli

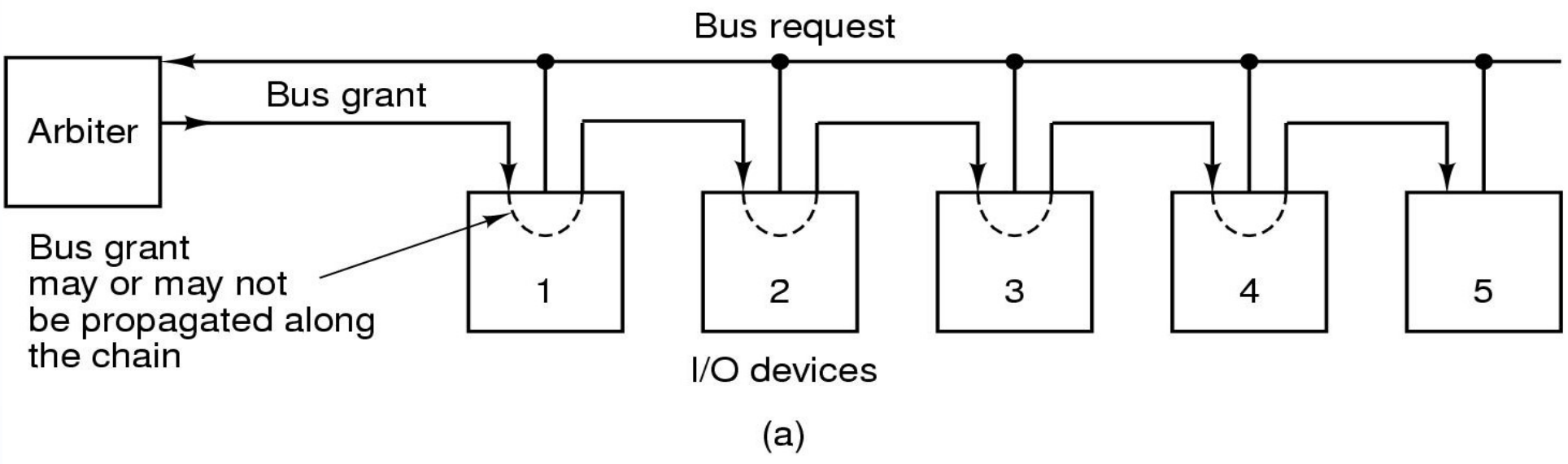
Bus asincrono

- Vantaggi
  - Flessibilità: la durata di una operazione è determinata unicamente dalla velocità della coppia master/slave
- Svantaggi
  - Per completare una operazione di comunicazione sono sempre necessarie 4 azioni
  - Occorre inserire negli slave i circuiti necessari a rispondere opportunamente al protocollo

Arbitraggio del bus

- Cosa succede se più di un dispositivo richiede l'utilizzo del bus contemporaneamente?
- È necessario un meccanismo di ***arbitraggio del bus*** per evitare ambiguità delle informazioni immesse sul bus stesso, sono possibili due strade:
  - arbitraggio centralizzato
  - arbitraggio decentralizzato
- Meccanismo del ***grant***

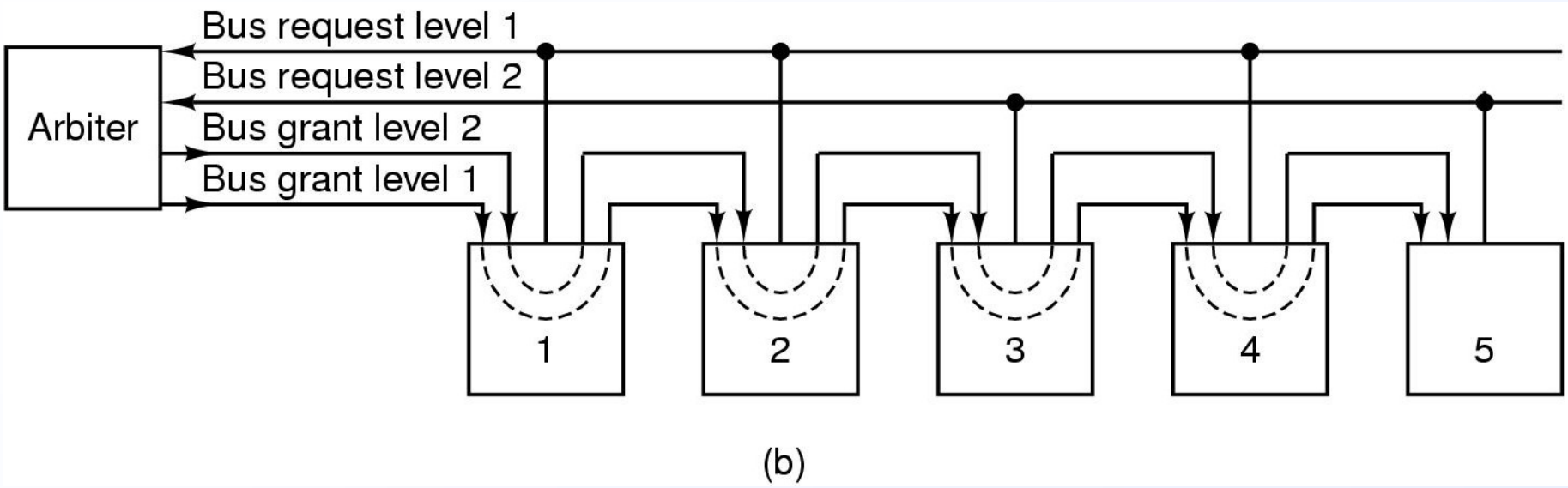
Arbitraggio del bus



Arbitro centralizzato quando l'arbitro "vede" una richiesta attiva la linea di *grant* del bus

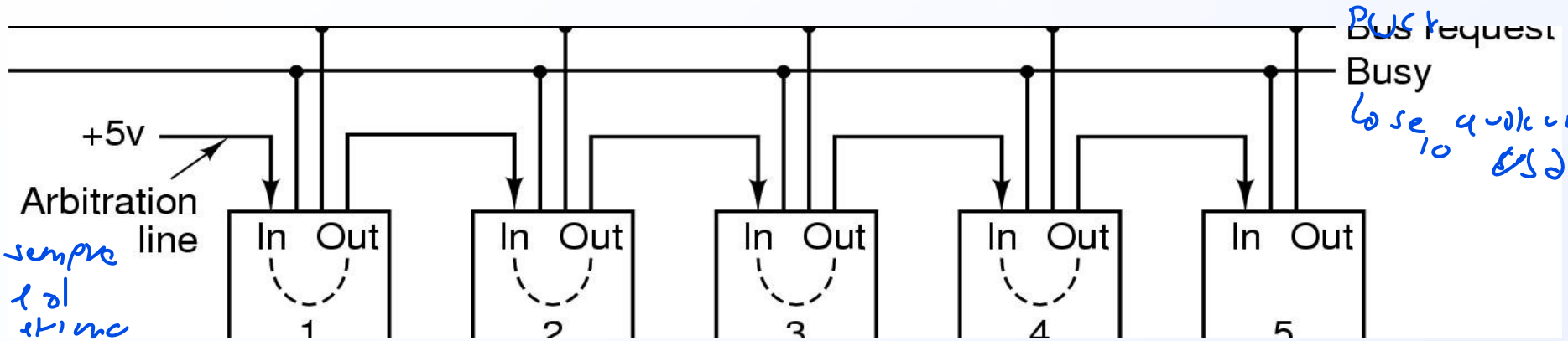
- ***Daisy chaining***: il *grant* viene trasmesso lungo la linea bus grant finché un dispositivo non accetta l'assegnamento
- Vince il dispositivo più vicino

Arbitraggio del bus



- Due (o più) livelli di priorità
- L'assegnamento segue lo stesso meccanismo del daisy chaining ma di dispositivi con priorità più alta hanno precedenza nell'assegnamento del grant da parte dell'arbitro
- Linea di ***acknowledge***
- ***Scelta attenta delle priorità tra i vari dispositivi***

Arbitraggio del bus

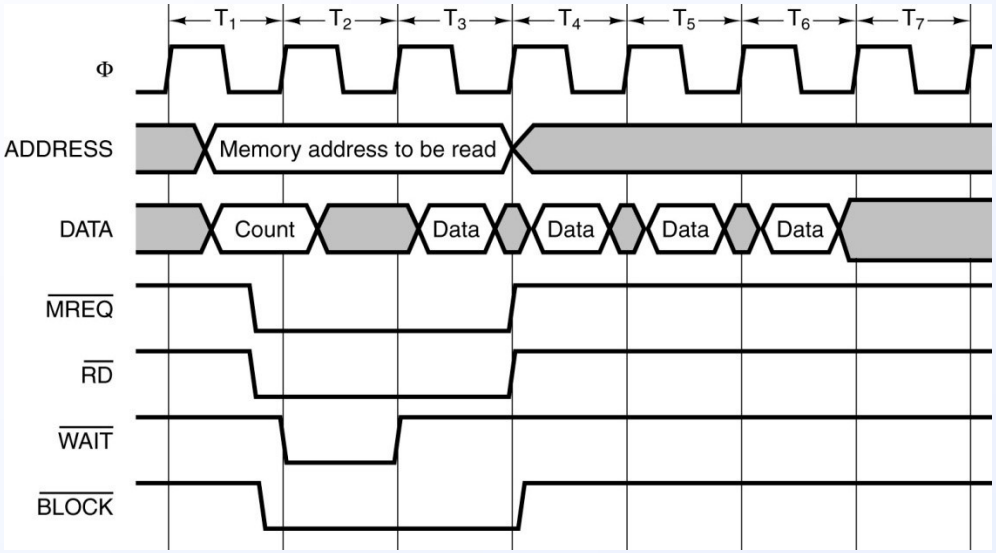


- Arbitro decentralizzato
  - Tante linee quanti dispositivi, ogni dispositivo osserva le linee prima di effettuare la richiesta (soluzione poco flessibile ma risparmia il costo dell'arbitro)
- Soluzione con tre linee, *bus request*, *busy* e *linea di arbitraggio*
  - Più economico e più veloce del daisy chaining centralizzato

Altri tipi di cicli di bus

Trasferimento a blocchi

- Il master invia allo slave, su una linea dati il numero di parole richieste
- Il master setta un segnale BLOCK sull'apposita linea
- Lo slave trasmette una parola in ogni ciclo fino alla fine del blocco
- Pratica comune in caso di lettura dalla cache
- Il master resetta BLOCK una volta ricevuto il numero di parole richieste

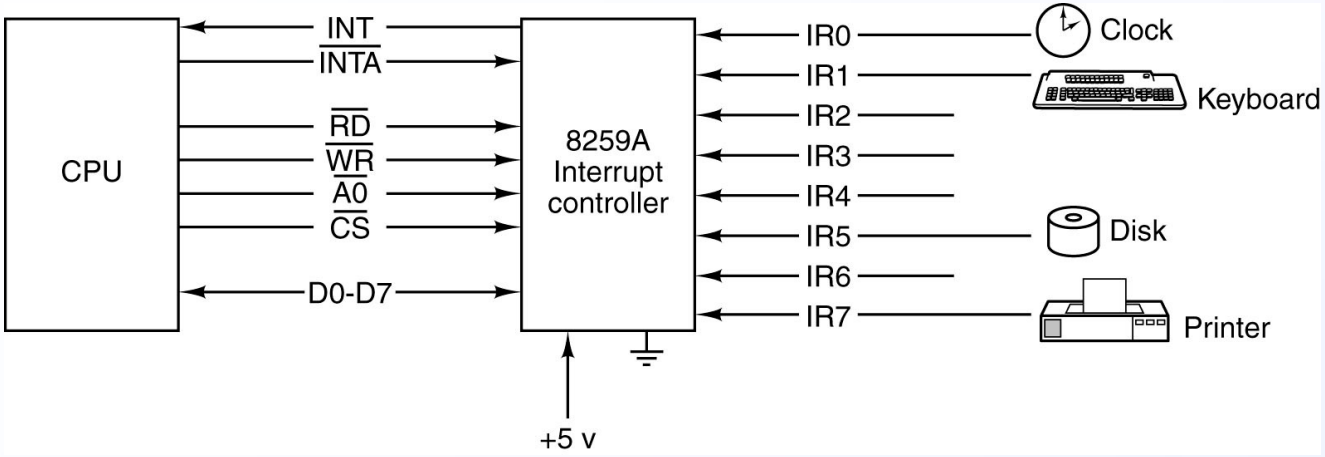


Altri tipi di cicli di bus

- Ciclo di bus **read-modify-write** (*test-and-set*). Si usa nel caso multiprocessore: più CPU collegate alla stessa memoria: questo ciclo impedisce ad altre CPU di usare il bus

- Ciclo di bus per il segnale di **interrupt**. Segnale di interrupt viene inviato dai periferici di I/O quando hanno terminato un lavoro. Se più periferici devono segnalare contemporaneamente esiste un problema analogo all'arbitraggio del bus. Priorità negli interrupt.

Controllore di interrupt



8259A interrupt controller.

- Quando un device vuole chiedere un interrupt attiva la sua linea di input
- Il controller attiva la linea INT quando riceve uno o più segnali dai device
- Quando la CPU è in grado di servire l'interrupt attiva la linea INTA
- Il controller specifica quale device ha mandato il segnale sulla linea D0-D7
- La CPU usa questo numero come indirizzo in una tabella (**interrupt vector**) per trovare l'indirizzo della procedura per gestire quel tipo di interrupt