Università degli Studi dell'Insubria
Dipartimento di Scienze Teoriche e Applicate

Architettura degli elaboratori

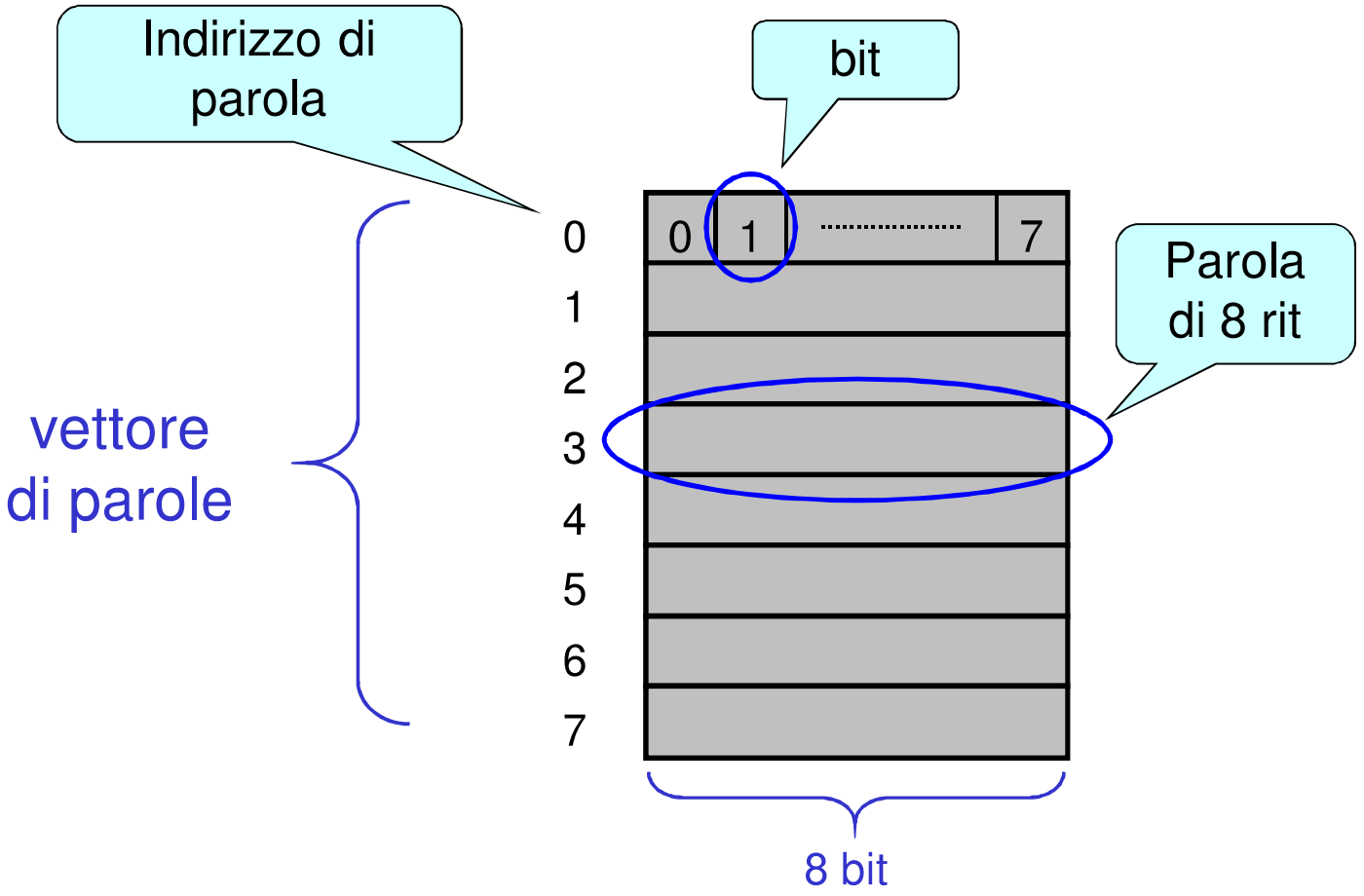
Memoria

Banco di Memoria

- Un blocco funzionale capace di memorizzare un certo numero *m* di parole di un certo numero *n* di bit ciascuna
  - quindi nessuna differenza con un banco di registri… ma, *m* molto più grande!

- Le due operazioni eseguibili sul banco sono:
  - **lettura**, ovvero si prelevano gli *n* bit memorizzati in una parola
  - **scrittura**, ovvero si memorizzano in una parola *n* bit

Organizzazione a parole (es. 8*8)



- Capacità: 8 parole × 8 bit = 64 bit

Componente di memoria

- Un banco di memoria viene definito per composizione di componenti integrati di memoria
- Un componente integrato (chip) di memoria si caratterizza specificandone:
  - la capacità, misurata in numero totale di bit memorizzabili: di solito si esprime come prodotto del numero di parole per il numero di bit contenuti nella parola
  - le funzioni: lettura e scrittura / solo lettura
  - il numero di porte di accesso

Interfaccia di memoria

- Il contenuto della memoria viene letto o scritto una parola per volta, in un ciclo di clock (o in più cicli in memorie lente)
- Si accede a una parola di memoria tramite la porta di accesso alla memoria
  - può funzionare in lettura e scrittura (è il caso più frequente), o solo lettura.

- Il banco di registri visto aveva tre «bus»:
  - un componente integrato di memoria ha tipicamente un solo bus dati

Il Bus

- Connette la CPU a un banco di memoria
- Costituito da una serie di connessioni su cui viaggiano segnali suddivisi in:
  - **dati**: la parola da leggere o da scrivere
    (nel nostro caso: su cavi bidirezionali)
  - segnali di **indirizzo** (l'indirizzo del dato dal leggere o da scrivere)
    (nel nostro caso: monodirezionali, verso il banco di memoria)
  - alcuni segnali di **controllo** per ogni banco di memoria
    (nel nostro caso: monodirezionali, verso il banco di memoria)

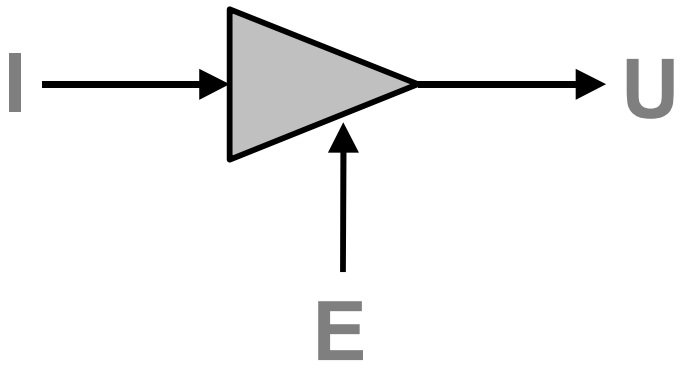
Bus dati bidirezionale

- Il tipico banco di memoria ha **uno solo bus**, *usato sia in lettura che in scrittura*.
  - Quindi si può eseguire una sola operazione alla volta, lettura *oppure* scrittura
  - Come implementare un bus di questo tipo (detto «bidirezionale»)?
  - E' necessario un dispositivo che consenta la NON interferenza fra le uscite e le entrate nel bus: il **tri-state buffer**

- Altro uso: unire le uscite di 2 o più componenti in un unico bus, per costruire banchi di memoria grandi
  - Per garantire la non interferenza fra le uscite dei vari componenti uso il tri-state buffer

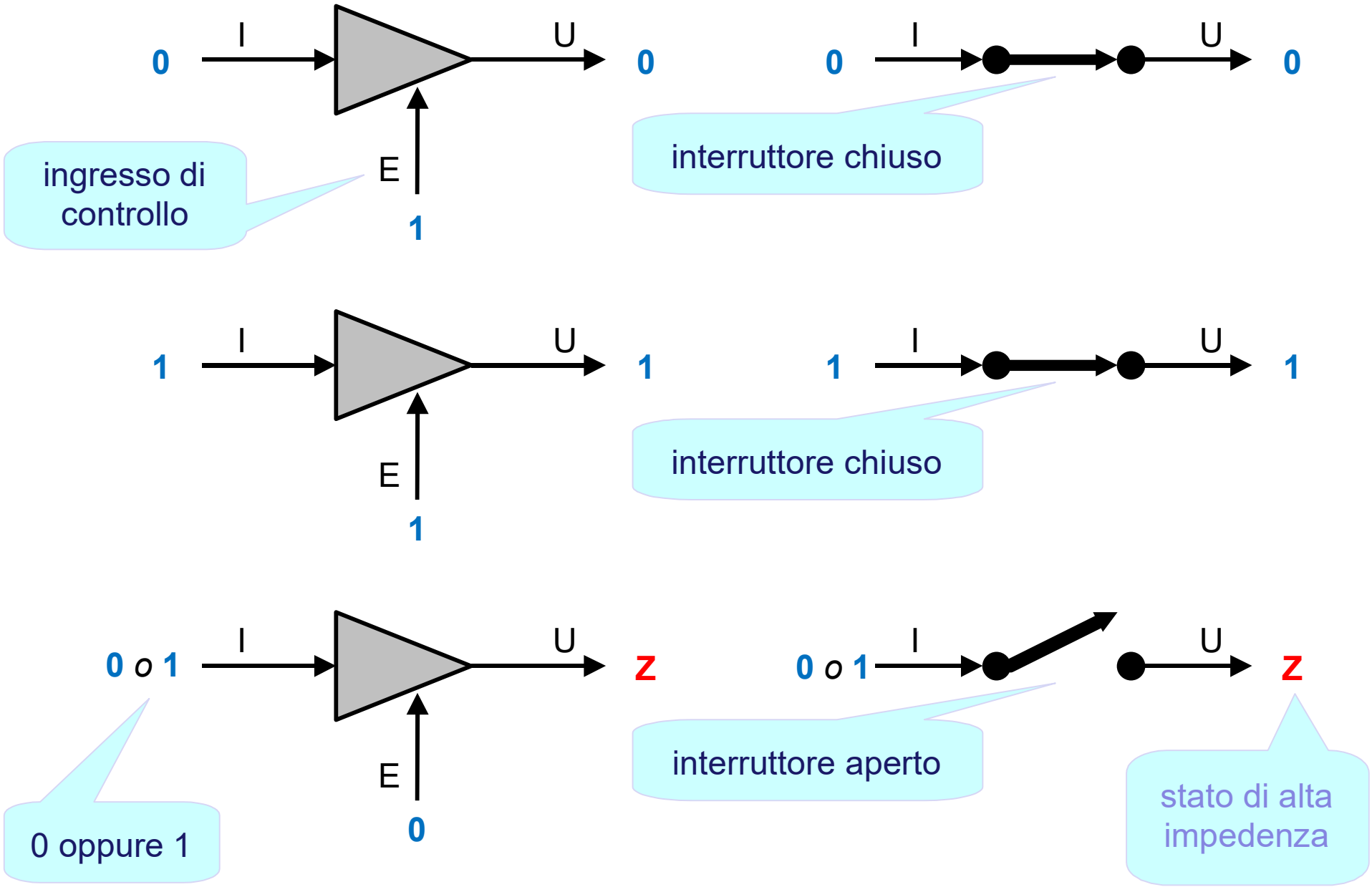
Tri-State Buffer (buffer a tre stati)

- È un dispositivo elementare dotato di un ingresso dati I, un ingresso di controllo E, e un'uscita U. Il dispositivo è modellabile come un contatto:
  - quando E (Enable) vale 0, il dispositivo forza lo stato fluttuante Z, o di alta impedenza, che isola elettricamente l'uscita
  - in stato di *bassa impedenza* consente di avere in uscita o il livello alto (**1**) o il livello basso (**0**)



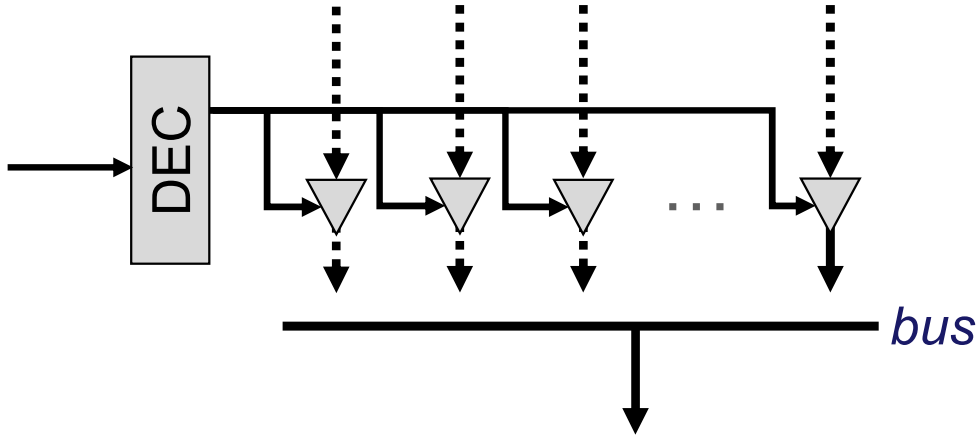
E	I	U
0	**0**	**Z**
0	**1**	**Z**
1	**0**	**0**
1	**1**	**1**

Funzionamento

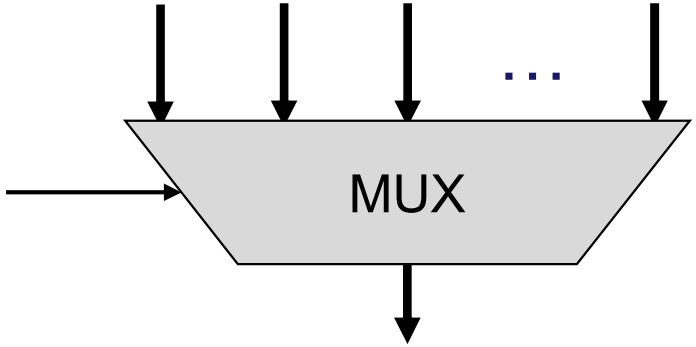


Usi di un Tri-state buffer: per un MUTEX

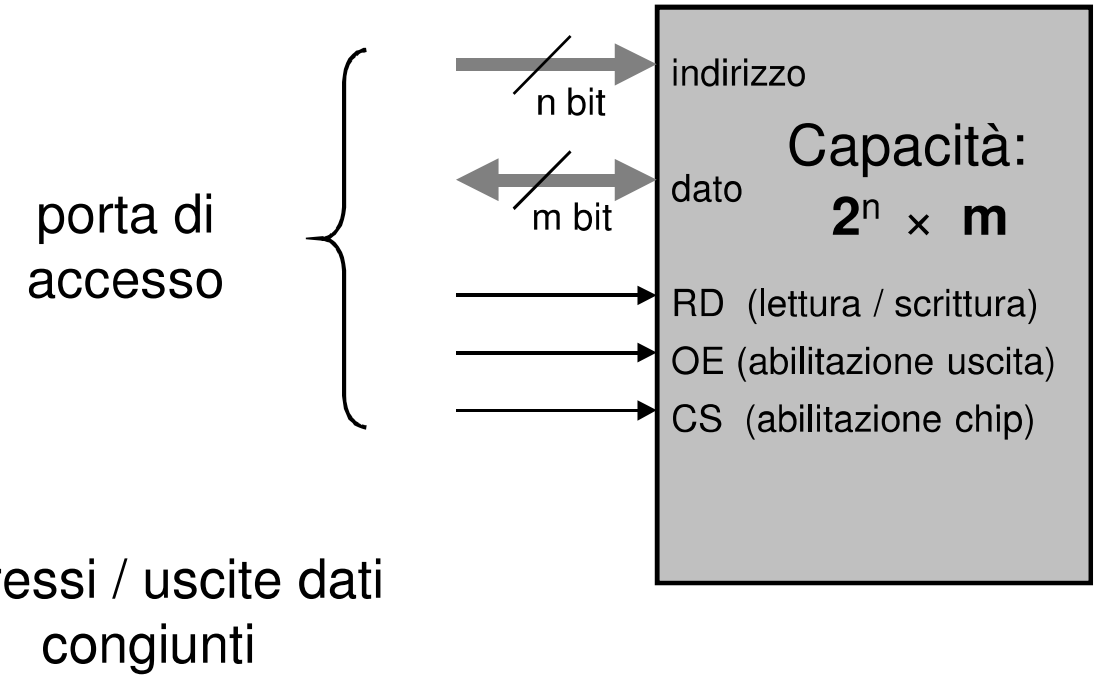
- Un'implementazione di **MUTEX** molto **scalabile**
  - ma non necessariamente veloce
- Questo:



- E' un modo di implementare questo:



Componente di memoria



- L'interfaccia è formata dai seguenti segnali:
  - Gli ingressi di indirizzo, che codificano l'indirizzo della parola su cui si deve operare
  - Le uscite / ingressi di dato, che servono per leggere / scrivere una parola
  - Il comando di lettura / scrittura, RD (read): RD = 1 lettura; RD = 0 scrittura
  - Comandi di abilitazione del componente CS e delle uscite OE

CS e OE

- CS e OE permettono di usare lo stesso bus sia in uscita che in entrata
  - e gestire tanti dispositivi interfacciati su uno stesso *bus*

- **CS - Chip Select**:
  - isola elettricamente il dispositivo (quando non è selezionato)
  - quando si manda il segnale sul bus, solo il dispositivo con CS=1 ascolta, gli altri è come se non ci fossero
  - per il dispositivo con CS a 1 è come essere l'unico sul bus
  - per i dispositivi con CS a 0 è come essere isolati dal bus
- **OE - Output Enable**:
  - isola elettricamente *le uscite* del dispositivo (quando non è selezionato), ma non le entrate
  - quando OE = 1, il dispositivo riversa il suo output sul bus
  - quando OE = 0, l'output del dispositivo è ignorata

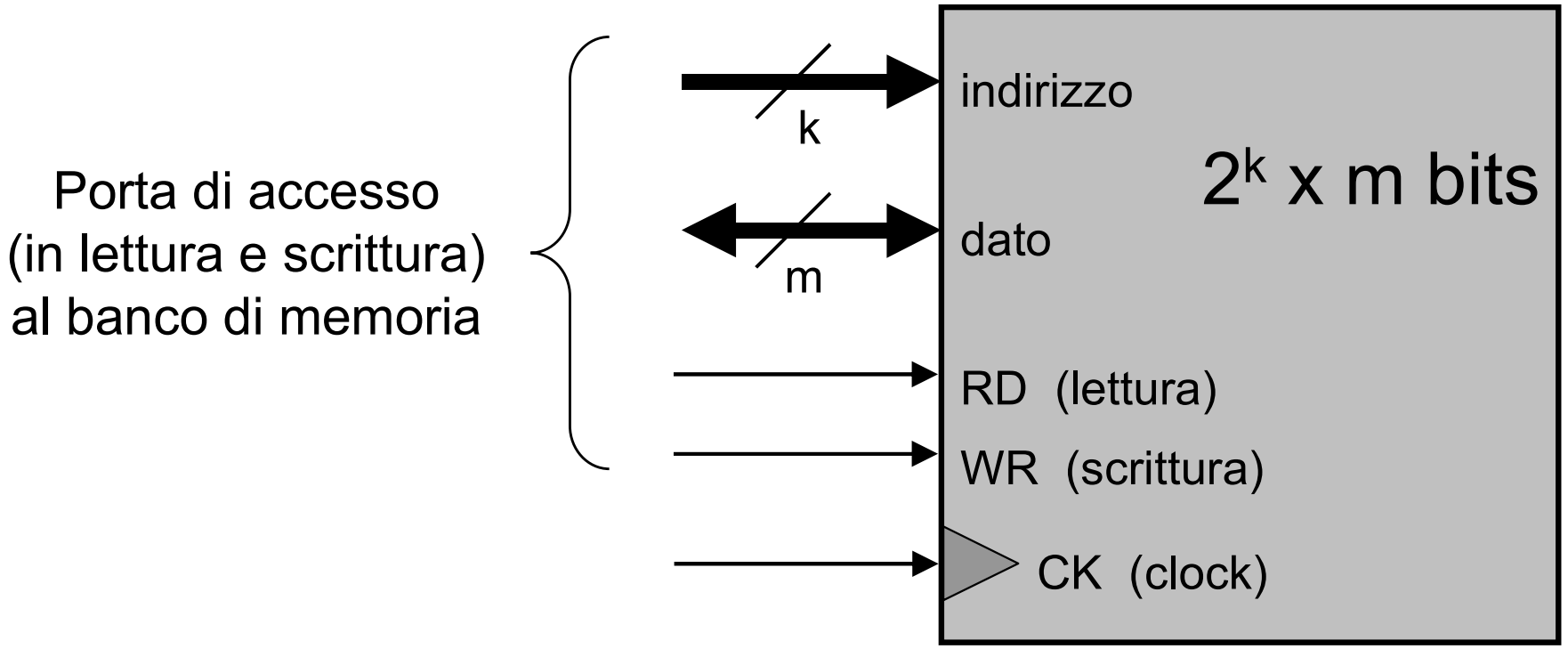
- Se CS = 0, allora anche OE = 0

Accesso alla componente

- Ciclo di lettura
- indirizzo della parola da leggere
- comando di lettura (RD a livello 1)
- non isolare le uscite dati (OE = 1)
- abilitare il componente (CS = 1)
- contenuto della parola disponibile sulle uscite

- Ciclo di scrittura
- indirizzo della parola da scrivere
- dato da scrivere in ingresso
- comando di scrittura (RD a livello 0)
- isolare le uscite dati (OE = 0)
- abilitare il componente (CS = 1).

Interfaccia alternativa



RD e WR = comandi di lettura e scrittura dedicati

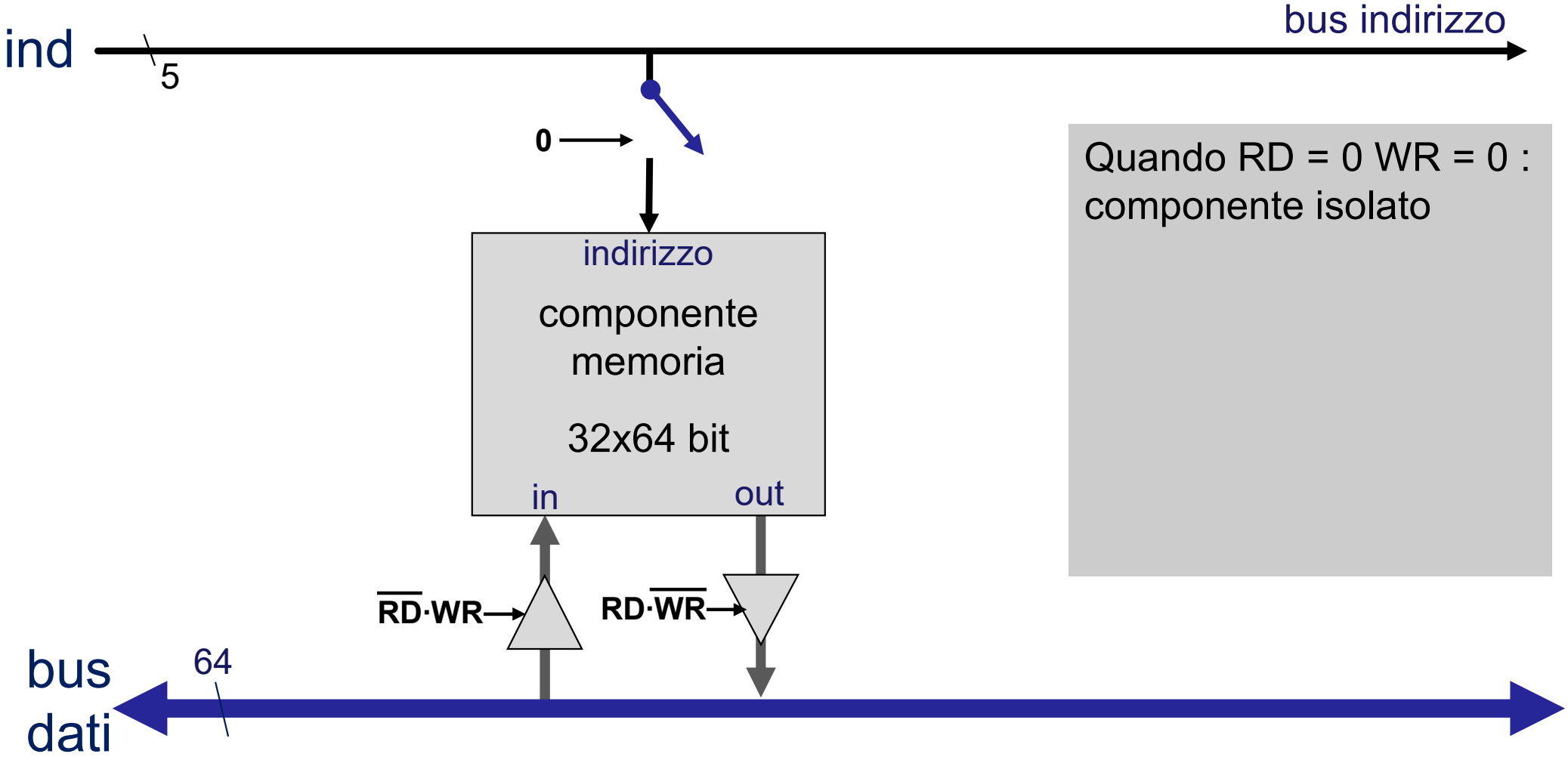
RD = 1 e WR = 0: eseguiamo una lettura (dal banco)

RD = 0 e WR = 1: eseguiamo una scrittura (sul banco)

RD = 0 e WR = 0: il banco di memoria è isolato (inattivo)

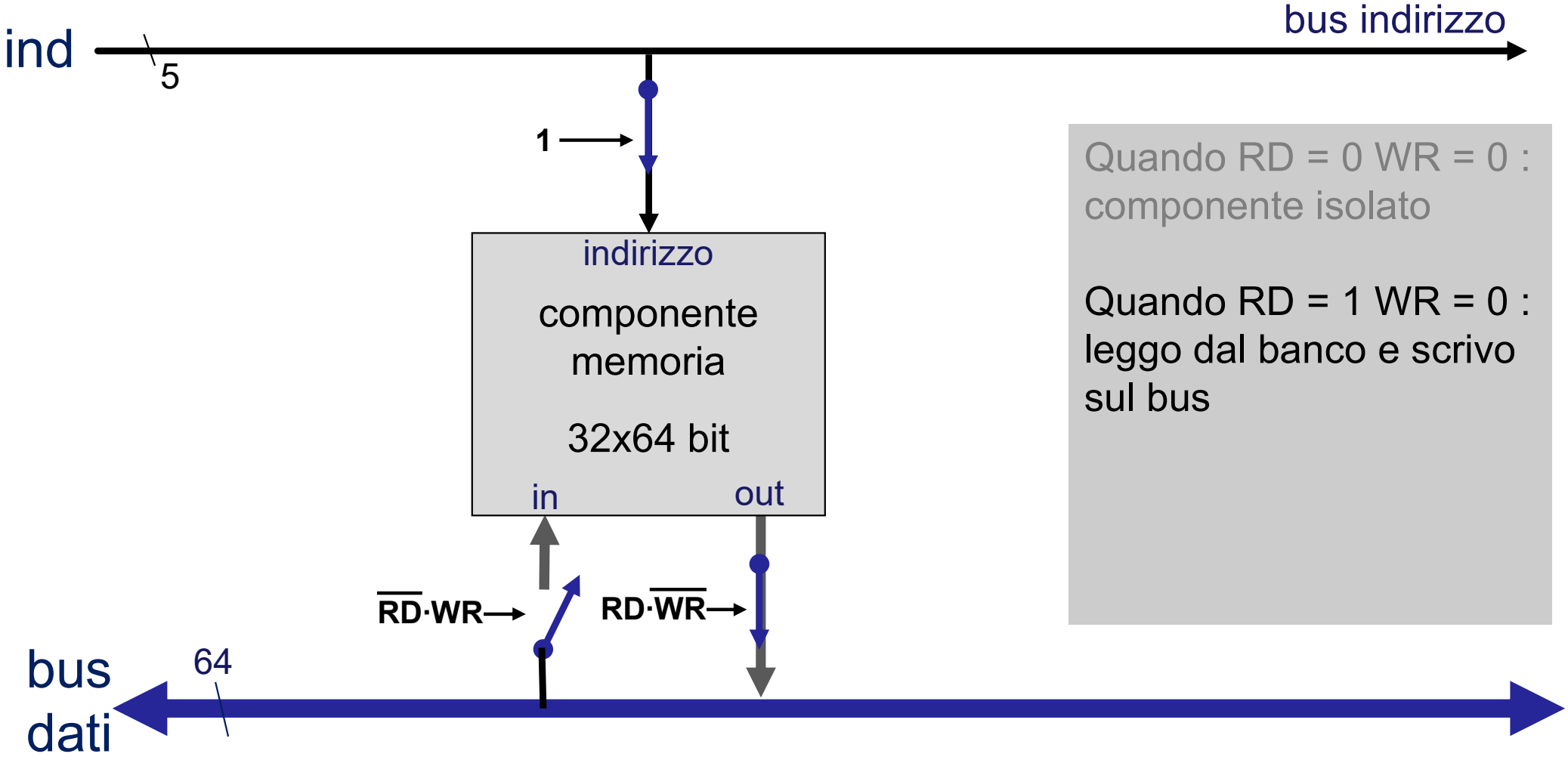
RD = 1 e WR = 1: non consentito

Un Memory Bank (da 32x64 bit)
con un bus di uscite/ingressi unificato



Quando RD = 0 WR = 0 :
componente isolato

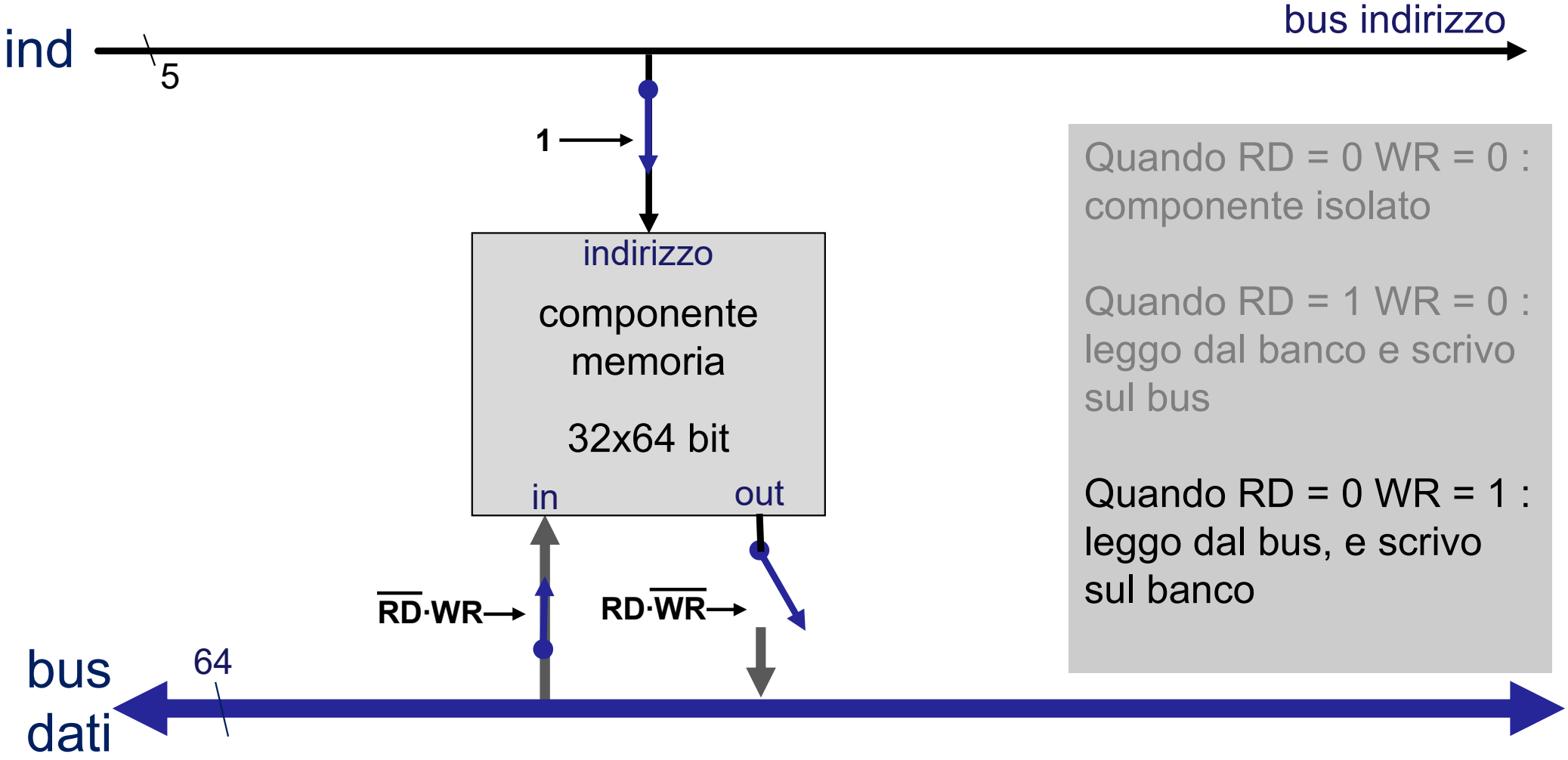
Un Memory Bank (da 32x64 bit)
con un bus di uscite/ingressi unificato



Quando RD = 0 WR = 0 : componente isolato

Quando RD = 1 WR = 0 : leggo dal banco e scrivo sul bus

Un Memory Bank (da 32x64 bit)
con un bus di uscite/ingressi unificato

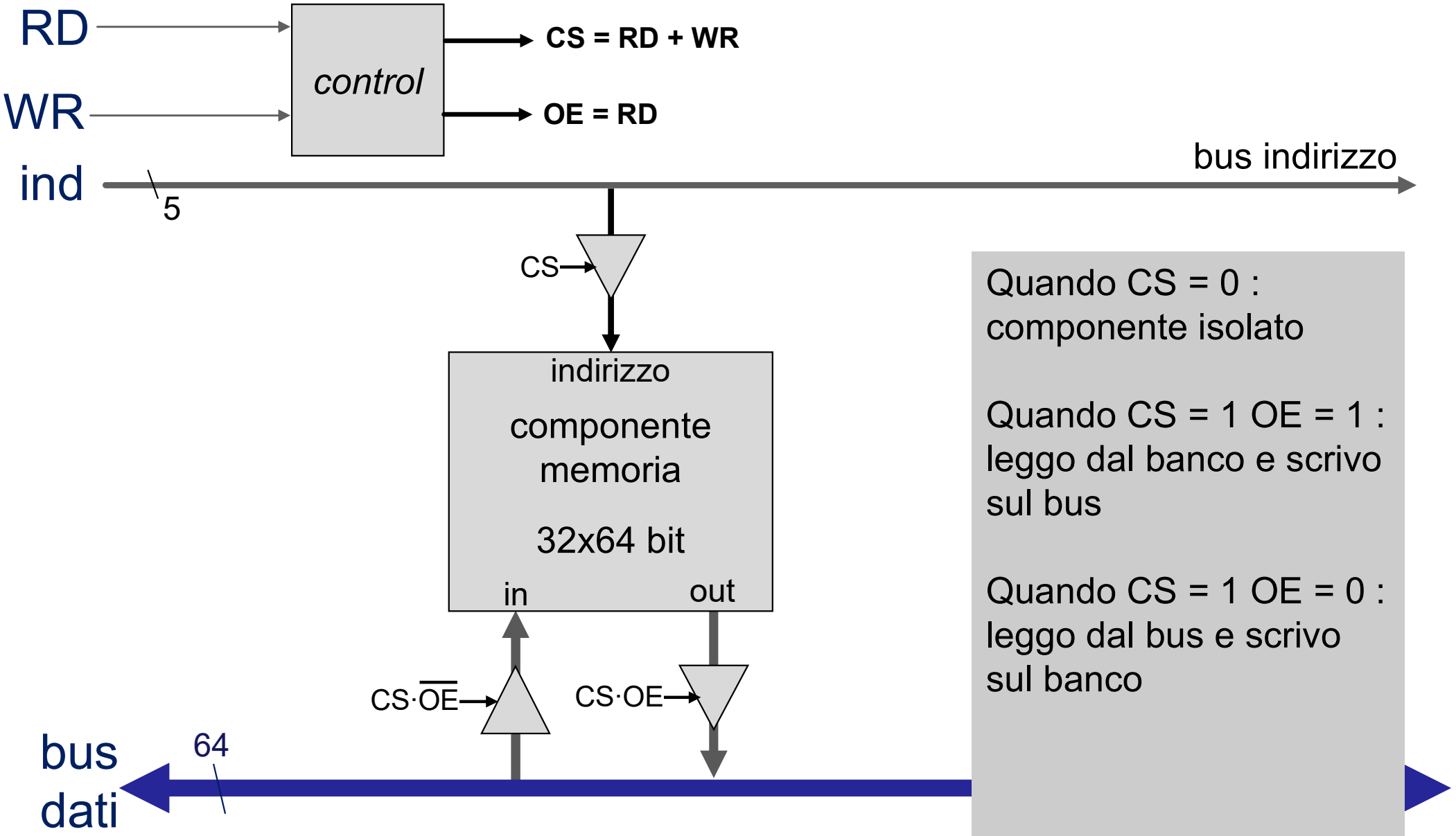


Quando RD = 0 WR = 0 : componente isolato

Quando RD = 1 WR = 0 : leggo dal banco e scrivo sul bus

Quando RD = 0 WR = 1 : leggo dal bus, e scrivo sul banco

Un Memory Bank (da 32x64 bit)
con un bus di uscite/ingressi unificato



Quando CS = 0 :
componente isolato

Quando CS = 1 OE = 1 :
leggo dal banco e scrivo sul bus

Quando CS = 1 OE = 0 :
leggo dal bus e scrivo sul banco

Costruire memorie grandi

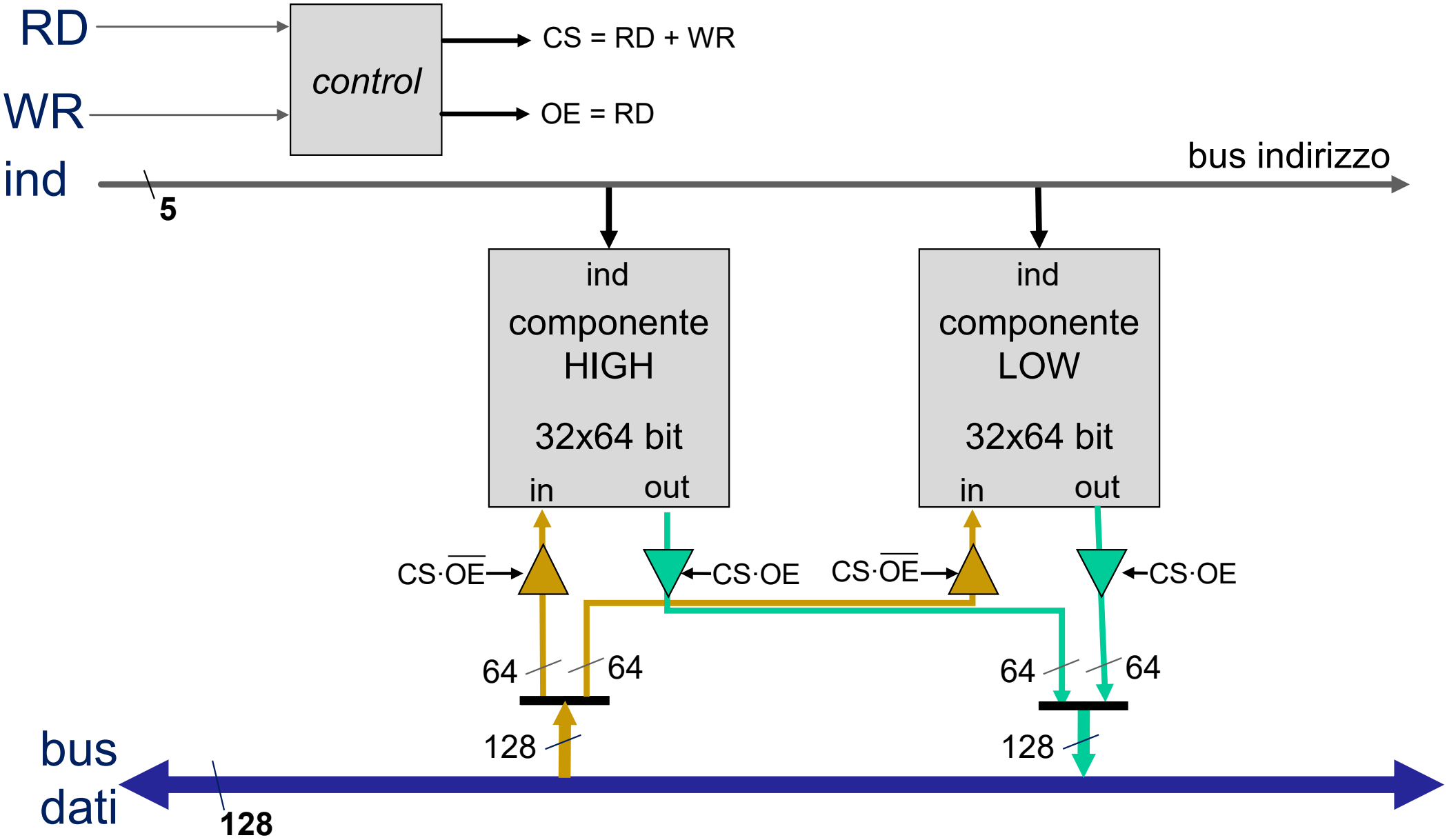
N componenti di memoria possono essere assemblati per ottenere banchi di memoria più grandi

- O per ottenere memorie con parole più lunghe. Per es:
  - 2 Componenti da 32x**64** in un banco da 32x**128**
  - una componente fornisce i 64 bit HIGH, e l'altra i 64 LOW della parola finale (HIGH = most significant, LOW = least significant)
- O per ottenere memorie con più parole. Per es:
  - 4 Componenti da **32**x64 in un banco da **128**x64
  - una componente memorizza tutte le parole di indirizzo 00…..
    una componente memorizza tutte le parole di indirizzo 01…..
    una componente memorizza tutte le parole di indirizzo 10…..
    una componente memorizza tutte le parole di indirizzo 11…..
  - CS a 1 solo per la componente attiva.
    OE a 1 solo su quella componente, e se lettura.
- O entrambe le cose contemporaneamente. Per es: …

Un Memory Bank da 32x128 bit
implementato con **2** componenti da 32x64 bit

Come lo implemento?

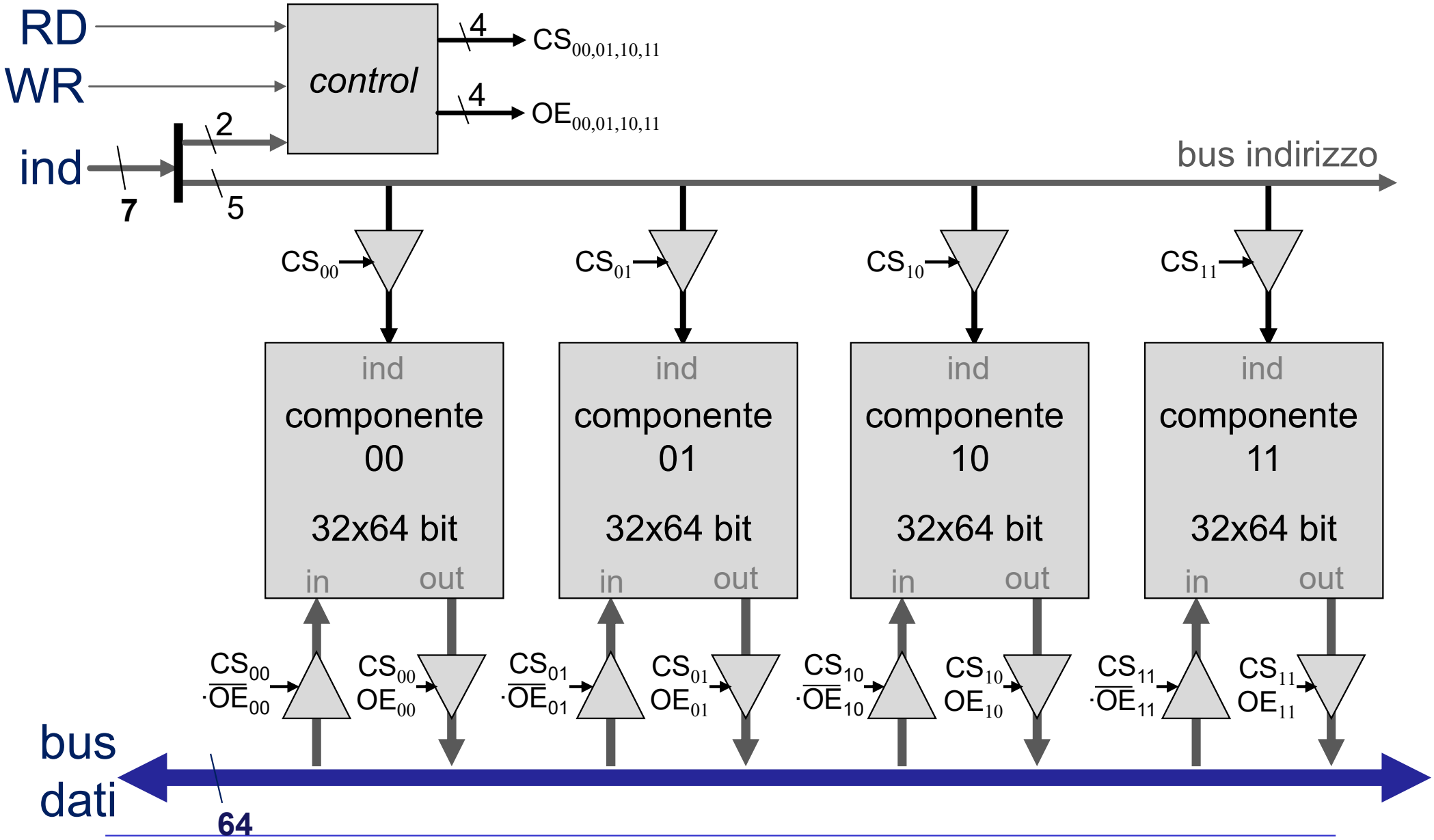
Un Memory Bank da 32x128 bit
implementato con **2** componenti da 32x64 bit



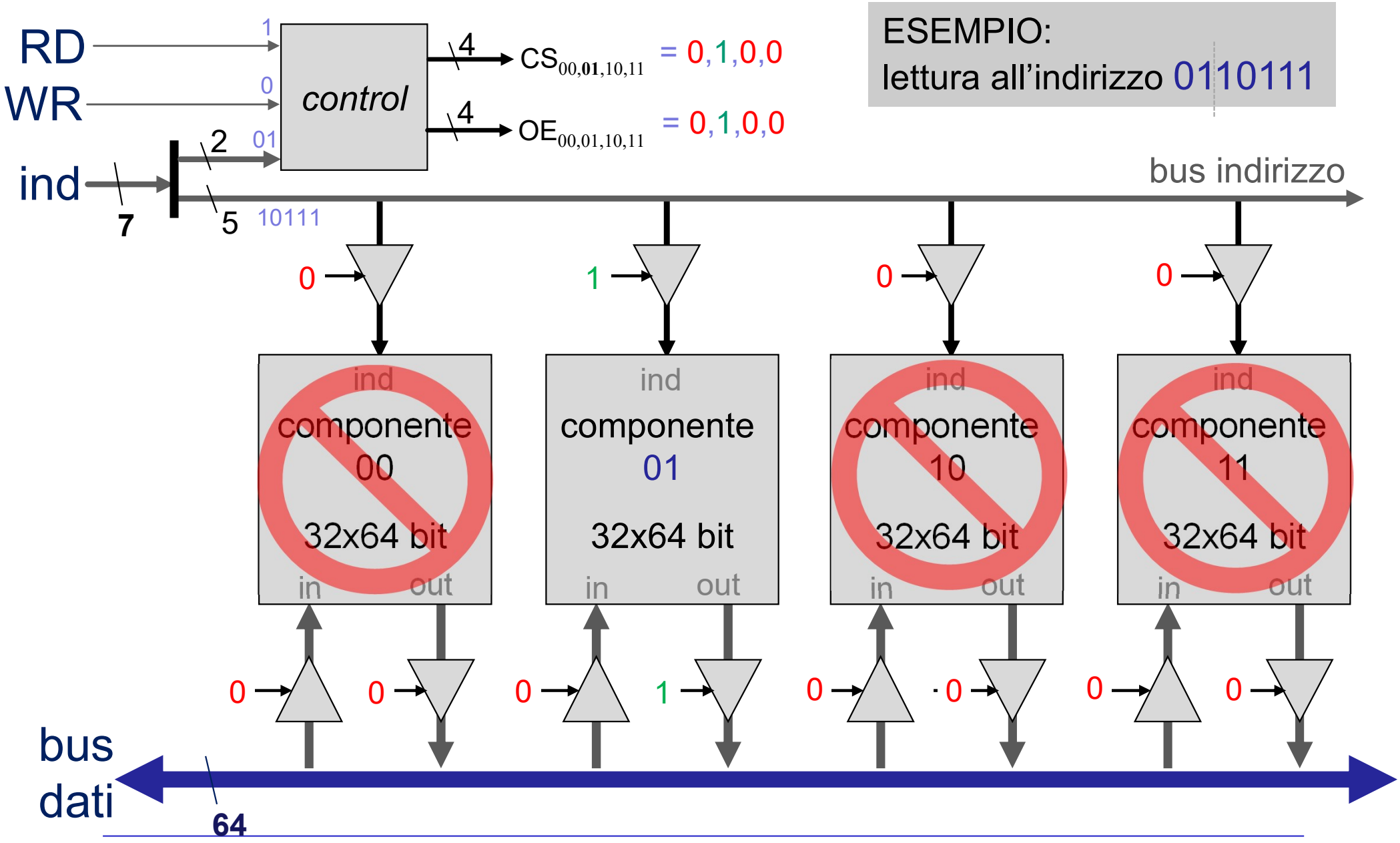
Un Memory Bank da 128x64 bit
implementato con **4** componenti da 32x64 bit

Come lo implemento?

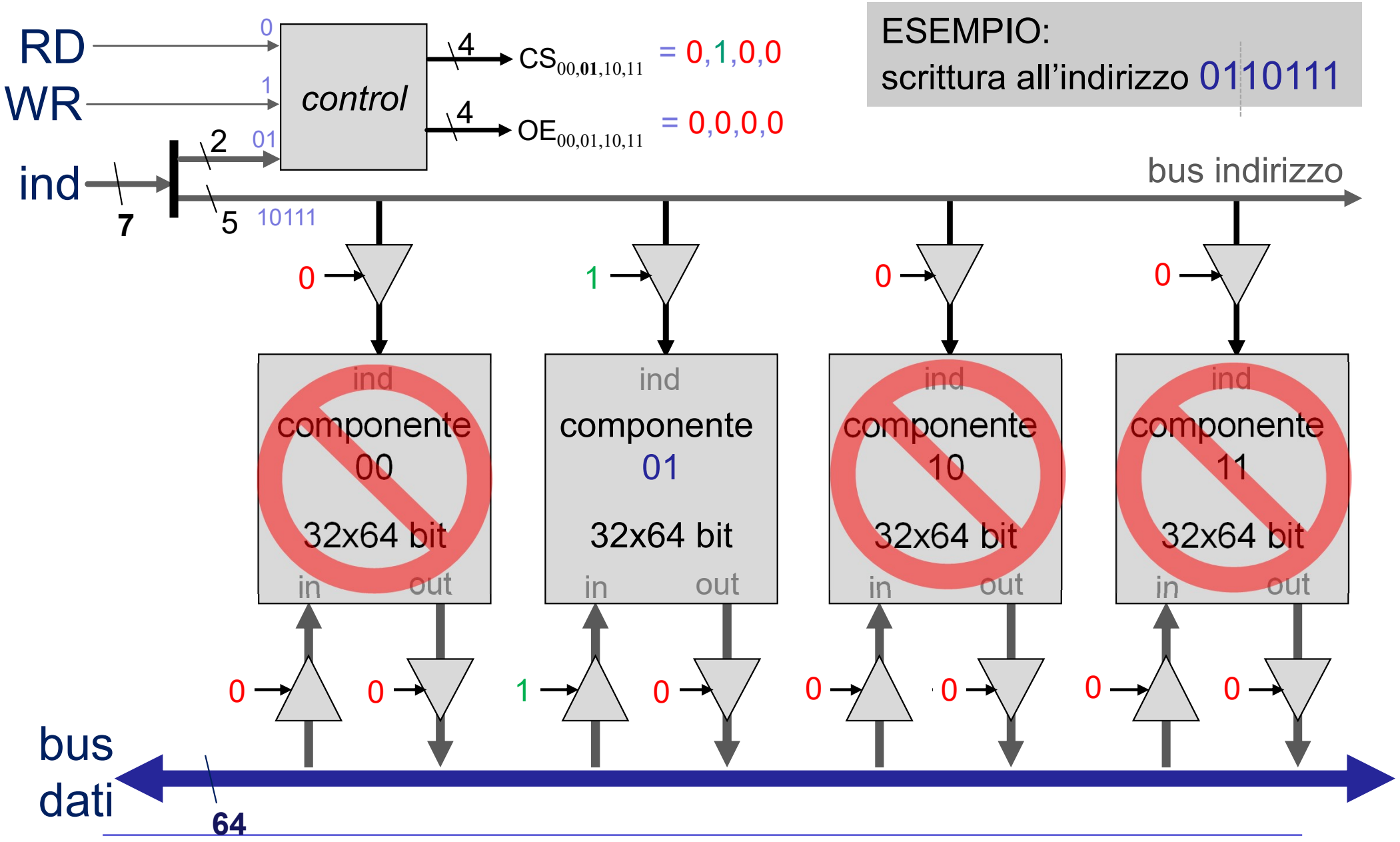
Un Memory Bank da 128x64 bit
implementato con 4 componenti da 32x64 bit



Un Memory Bank da 128x64 bit
implementato con 4 componenti da 32x64 bit



Un Memory Bank da 128x64 bit
implementato con **4** componenti da 32x64 bit



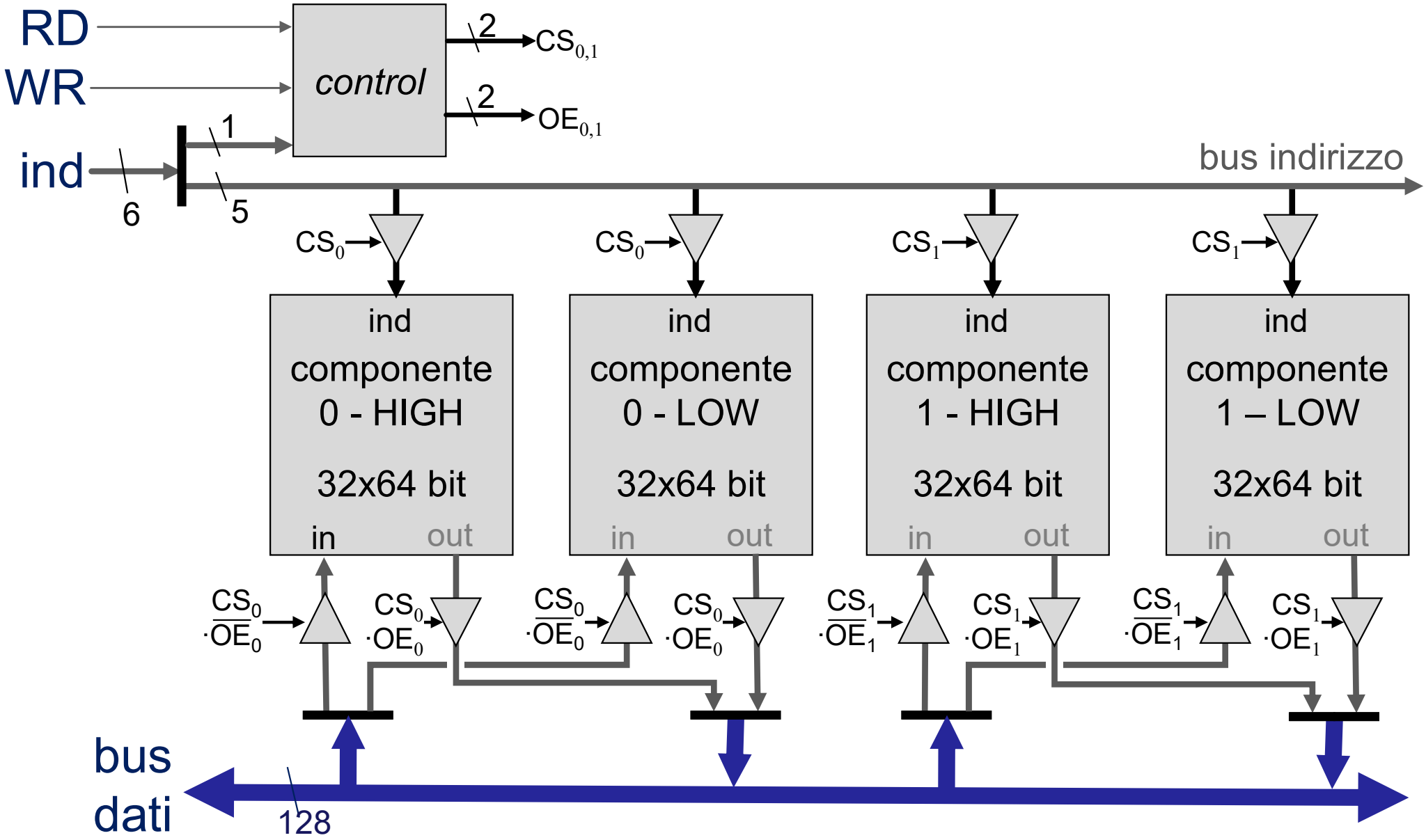
Un Memory Bank da 128x64 bit
implementato con **4** componenti da 32x64 bit

- Note e spiegazioni:
  - per indicizzare una delle 128 parole, mi servono 7 bit
  - i 2 bit più significativi = l'indice del blocco in cui cercare la parola
  - i 5 bit meno significativi = l'indice della parola in quel blocco
- Analogia intuitiva (in base 10):
  - Ho un hotel composto da 10 piani (da 0 a 9),
    e 100 stanze in ogni piano (da 00 a 99)
  - L'hotel ha quindi 10x100 = 1000 stanze, (da 000 a 999)
  - La stanza 749 sarà la stanza 49 del piano 7

Un Memory Bank da 64x128 bit
implementato con ? componenti da 32x64 bit

Un Memory Bank da 64x128 bit
implementato con **4** componenti da 32x64 bit

Un Memory Bank da 64x128 bit
implementato con 4 componenti da 32x64 bit

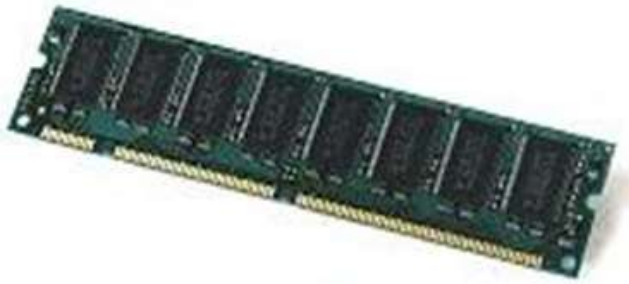


Banco di memoria realizzati in un chip: (chip di memoria)

- Negli esempi abbiamo visto memory bank ancora molto piccoli.
- La dimensione di memory bank realizzato in un chip è limitata da quante porte (e quindi transistor) riesco ad alloggiare nel chip
- Un memory bank reale può essere ad esempio:
  1 M × 32
  cioè 2^(20) × 32
  cioè ~un milione di parole da 32 bit
  - bit per l'indirizzo: 20
  - bit per ogni parola: 32
- Oppure per es
  64 K × 8,
  1 M × 8,
  1 M × 1,
  256 M × 1

$2^{20} \cdot 2^5 = 2^{25}$ bit $< 2^{22}$ byte

e aggregando più chip di memoria nei modi visti si ottengono memory bank *su più chip* ancora più capienti



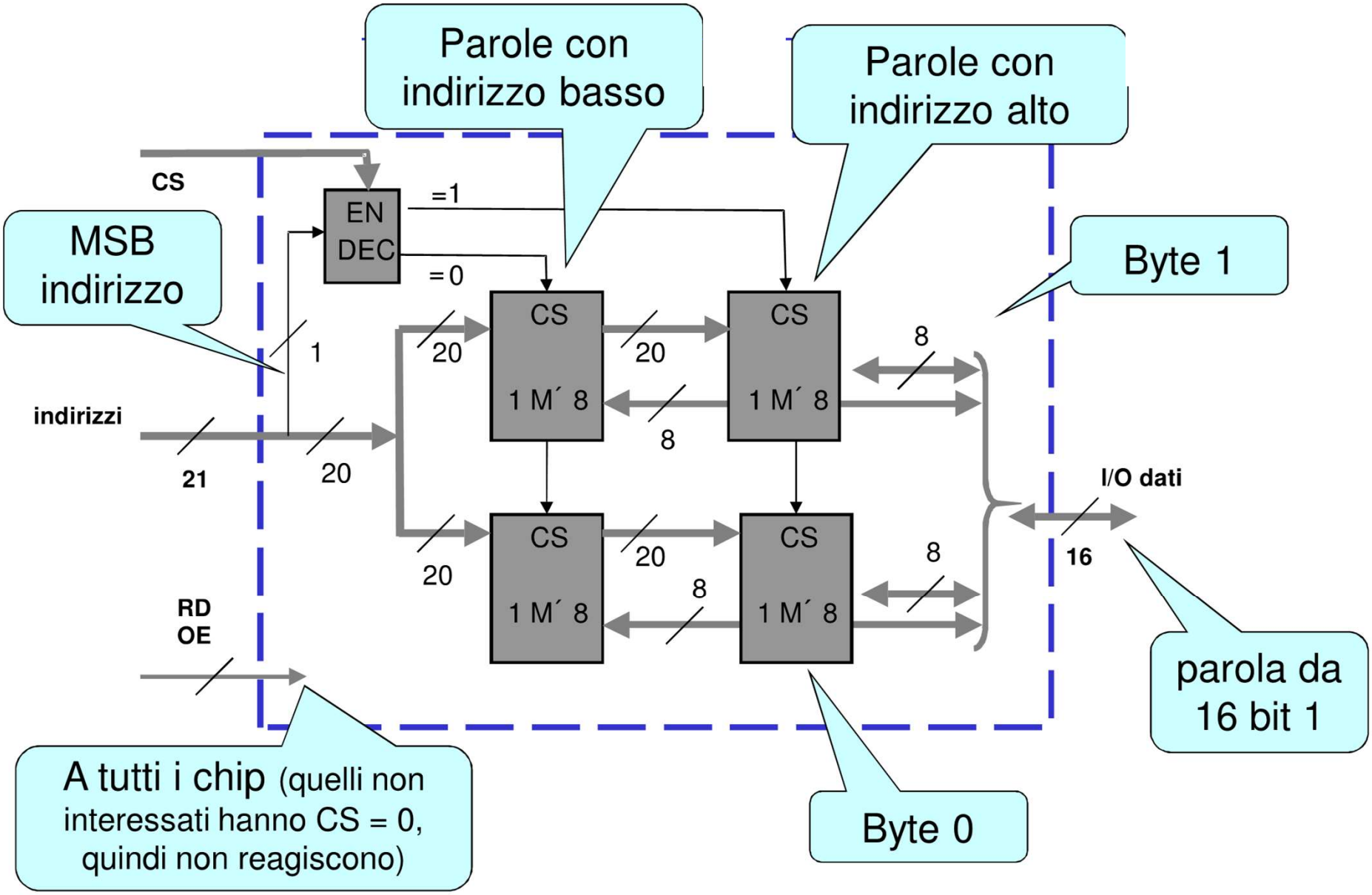
Organizzazione a matrice

- Le memorie le immaginiamo come vettori (1-dimensionali) di parole
- Ma le memorie di elevata capacità spesso hanno una struttura fisica interna a matrice (2-dimensionale)
  - scopo: ridurre la lunghezza dei collegamenti interni

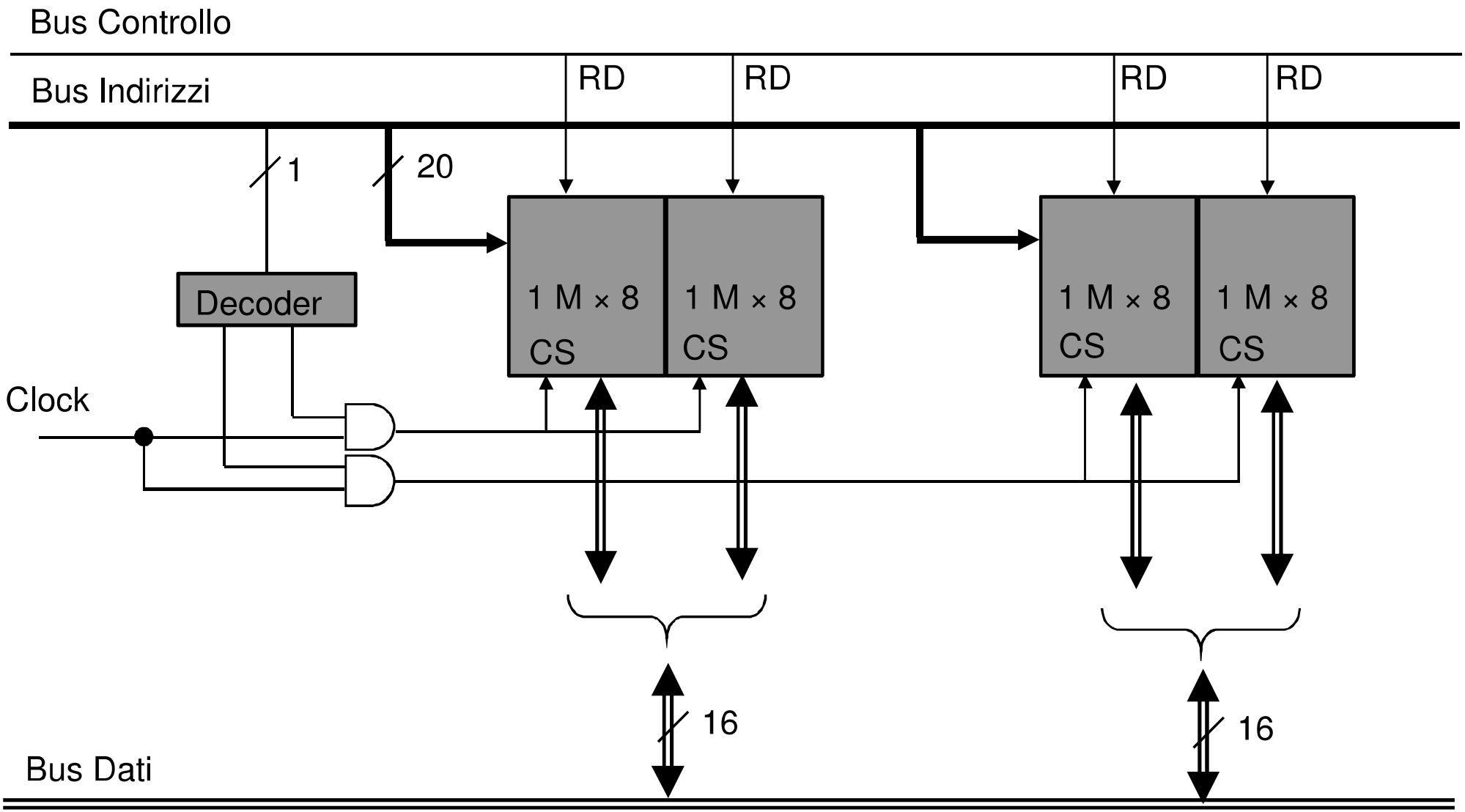
Esempio

- Si desidera ottenere un banco di memoria da 2 M × 16 bit
  - Indirizzo da 21 bit
  - Dati (parole) da 16 bit
- Si dispone di 4 chip di memoria
  da 1 M × 8 ciascuno
  - Indirizzo da 20 bit
  - Dati da 8 bit
- Il numero di chip è sufficiente:
  4 (1 M × 8) = 32 Mbit e
  2 M × 16 = 32 Mbit
- Implementazione:
  matrice 2 × 2 (2 righe di 2 chip di memoria ciascuna)

Struttura del banco



Struttura del banco



Tecnologie per realizzare una memoria

- Esistono svariate tecnologie per implementare una memoria
- Differiscono per:
  - capacità realizzabile
  - tempo di accesso a una parola (in lettura, o in scrittura)
  - politica di accesso:
    lettura e scrittura,
    sola lettura,
    «programmabilità sul campo» (scrittura lentissima)
  - stabilità: volatile o persistente
  - costo / scalabilità
- I dettagli appartengono al livello dei dispositivi, ma vediamo alcune caratteristiche generali

RAM Statica (SRAM)

- Memoria RAM (Random Access Memory) realizzata attraverso bistabili
  - flip-flop
  - come abbiamo visto, si implementano attraverso porte logiche (piccoli circuiti sequenziali bistabili)
- *Costo:* elevato (molte porte logiche, ~6 transistor per bit)
- *Capacità:* medio-piccola (di conseguenza)
- *Politica di accesso:* lettura e/o scrittura
- *Tempo di accesso:* molto breve (sia in lettura che in scrittura)
- *Stabilità:* volatile
  (senza alimentazione il contenuto della memoria svanisce)

- *Usi:* svariati, in particolare registri (che abbiamo visto) e memorie *cache* (che vedremo)

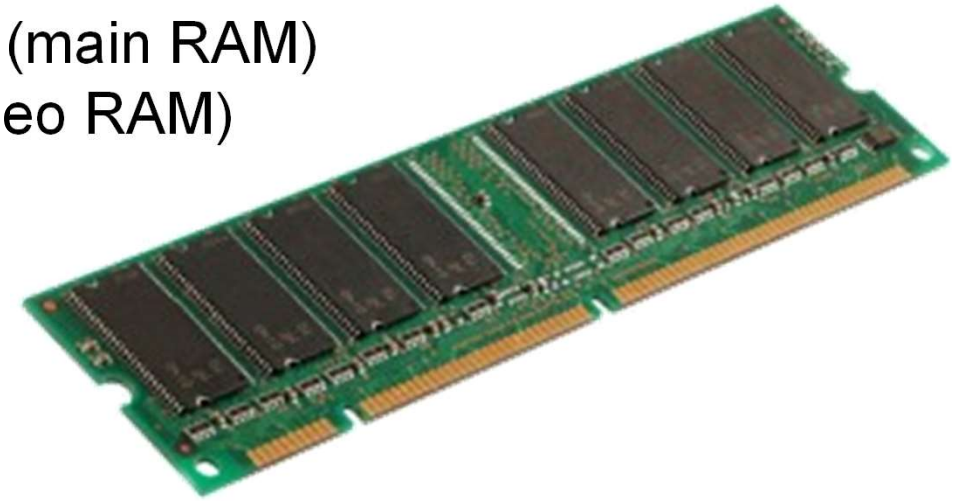
RAM Dinamica (DRAM)

- La SRAM è costosa (molti transistor per bit memorizzato)
- La tecnologia DRAM usa circa 1 transistor per bit memorizzato
- Sfrutta il fenomeno dell'accumulo temporaneo di carica sul transistor (può essere immaginato come un piccolo accumulatore)
- Internamente, contiene un circuito di rinfresco (*refresh*) che rigenera ciclicamente le cariche sui transistor prima che queste svaniscano
- Realizzata come matrice di transistor, ad altissima densità
- Risultato: altissima densità di bit, ma costo energetico elevato, e tempi più lenti

RAM Dinamica (DRAM)

- *Capacità:* grande-grandissima
- *Tempo di accesso:* medio
- *Politica di accesso:* lettura e scrittura
- *Stabilità:* volatile:
  senza alimentazione il contenuto della memoria svanisce
  (anche se non proprio subito, come la SRAM)
- *Costo:* basso
- *Consumo:* alto (richiede continui *refresh*)
- *Usi*: numerosissimi.
  memoria centrale dei calcolatori (main RAM)
  memoria delle schede video (video RAM)

Approssimativamente:
quando senti parlare
di RAM, è DRAM.



ROM

- Memoria ROM (Read Only Memory), realizzata come matrice di transistor
- *Capacità:* molte dimensioni possibili (anche molto grande)
- *Tempo di accesso:* medio
- *Politica di accesso:* sola lettura
- *Stabilità*: Persistente
  (il contenuto permane anche in assenza di alimentazione)
- *Usi*: per memorizzare programmi permanenti, non modificabili; microprogrammazione; grandi volumi di produzione

PROM, EPROM, EEPROM

- Capacità e tempo simili alla ROM
- Sola lettura e persistenti
- E' «programmabile sul campo» («field programmable»)
  tramite un apposito dispositivo programmatore:
  - PROM: programmabile una volta sola
  - EPROM: cancellabile (con raggi UV, tipicamente tutto o niente)
  - EEPROM: cancellabile elettricamente
    (si può anche scrivere un solo byte per volta)
- Usi: piccoli volumi di produzione, prototipi

Memoria FLASH

- *Capacità e tempo:* varie, spesso di poco inferiori alla DRAM
- *Costo:* basso
- *Politica di accesso:* lettura e scrittura, ma la scrittura
  - è effettuabile a blocchi di byte
  - si può effettuare solo un numero limitato di volte (es. alcune migliaia)
- *Stabilità:* persistente - contenuto permane anche senza alimentazione
- *Usi*:
  - memory sticks (USB drives),
  - SSD (Solid State Drives),
  - memory cards per fotocamere o videocamere…

Considerazioni tecniche sulla Memoria di un computer

- La **memoria centrale** di un computer è molto grande ma
  - ha ***un*** bus dati **unificato**, bidirezionale, usato in lettura e in scrittura (quindi in ogni dato momento: o ci si legge, o ci si scrive)
  - è esterna alla CPU, in un chip separato (lentezza)
  - è implementata come tipo DRAM (consumo, lentezza)
  - *un ciclo di clock tipicamente non basta per una scrittura o lettura.*
- Il **register file** è molto piccolo ma
  - ha bus multipli di lettura e/o scrittura (capace di eseguire, per es: due letture e una scrittura per colpo)
  - è stampata nello stesso Chip della CPU, insieme alla ALU (fa parte del circuito integrato della CPU) (velocità)
  - è di tipo SRAM (velocità)

Possiamo avere il meglio dei due mondi? Prossima lezione