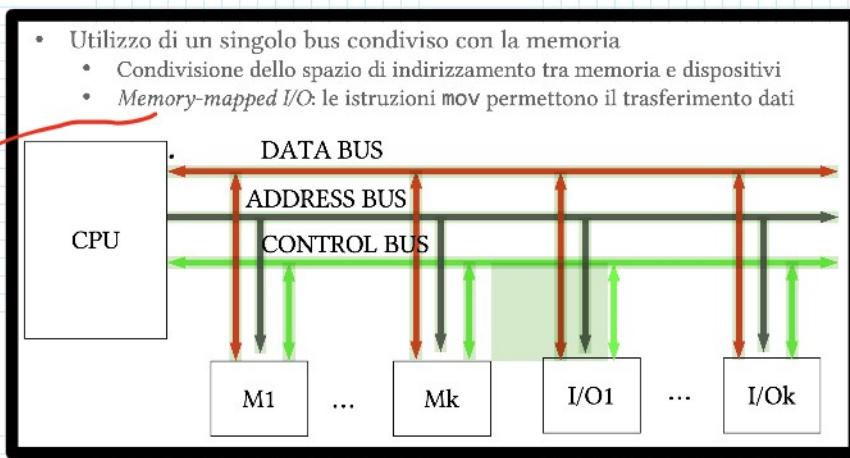




Possibili connessioni CPU-dispositivi

Meniamo un Bus così colleghiamo più dispositivi. Ma se ho più dispositivi sullo stesso bus, con quale sto parlando? Nel protocollo di handshaking nasce un problema di identificazione per capire con chi sto andando a parlare!

MA IO UN BUS GIA LO HO! Bus di sistema condiviso con la memoria, dove sopra ho i miei moduli di memoria ma anche i dispositivi.



SE LO DEVO SCRIVERE UN DATO IN MEMORIA USO LA MOV. SCRIVO dati sul data bus, SCRIVO l'indirizzo sull'address bus, SCRIVO dei segnali di controllo sul control bus.
SE I MIEI DISPOSITIVI DI I/O SOLO SULLO STESSO BUS POSSO USARE QUESTA STESSA ISTRUZIONE PER SCRIVERE O LEGGERE I DATI DA QUEI DISPOSITIVI!

COME DISTINGUO UN BLOCCO DI MEMORIA DA UN DISCO?
L'INDIRIZZO.

Anche i dispositivi all'interno della nostra architettura sono identificati da degli indirizzi!

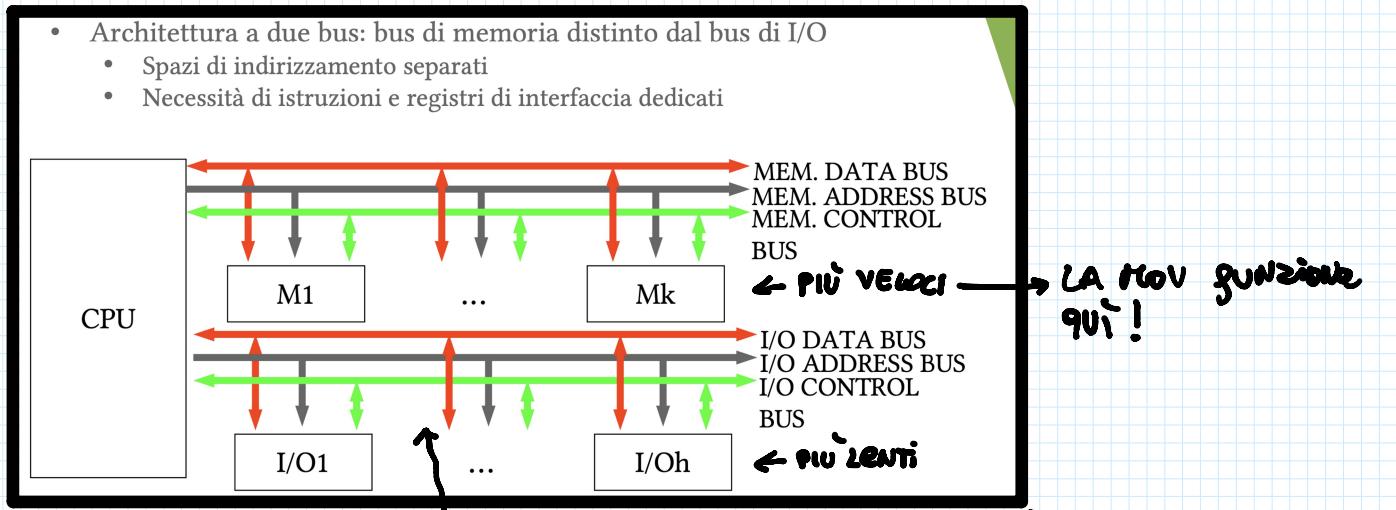
Nel caso di *Memory mapped I/O* non posso usare un indirizzo per identificare un dispositivo e lo stesso indirizzo per identificare una voceabile in memoria!

PERÒ LA RAM NON PUÒ ANDARE ALLA VELOCITÀ DI MIA NONNA.

UTILIZZO UN BUS PER INTERFACCIARMI CON LA MEMORIA.

UTILIZZO UN BUS PER INTERFACCIARMI CON I DISPOSITIVI DI I/O!

- Architettura a due bus: bus di memoria distinto dal bus di I/O
 - Spazi di indirizzamento separati
 - Necessità di istruzioni e registri di interfaccia dedicati



LA STORIA DELLA MOV FUNZIONA SOLO SU UN BUS, COME RISOLVO?

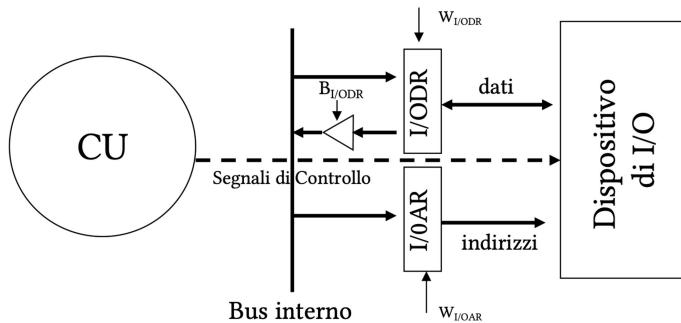
COME FARCI AD USARE QUEST'ALTRO BUS? DEVO AVERE ALTRE ISTRUZIONI PER QUEL BUS!
DEVE NECESSARIAMENTE UTILIZZARE DUE REGISTRI TRAMPOLE DIFFERENTI!

I/O ADDRESS REGISTER
I/O DATA REGISTER

USUALI A MAR E MDR CHE PERÒ SONO PIÙ COLLEGATI AL BUS VERSO LA MEMORIA, SONO COLLEGATI ALL'IOPUS.

Interfaccia dello z64

- Registro Dati (I/ODR)
- Registro Indirizzo (I/OAR)
- Segnali di Controllo (I/ORD, I/OWR, ...)



Innamoriamoci di essere in grado di distinguere un dispositivo dall'altro.

Utilizzeremo un indirizzo per identificare uno.

Ogni dispositivo ha un decodice, che serve a ricevere dall'indirizzo bus un indirizzo e se l'indirizzo è associato a quel dispositivo abilita un segnale, ossia il segnale di SELECT.

Dopo di che andiamo a differenziare le operazioni di lettura e scrittura:

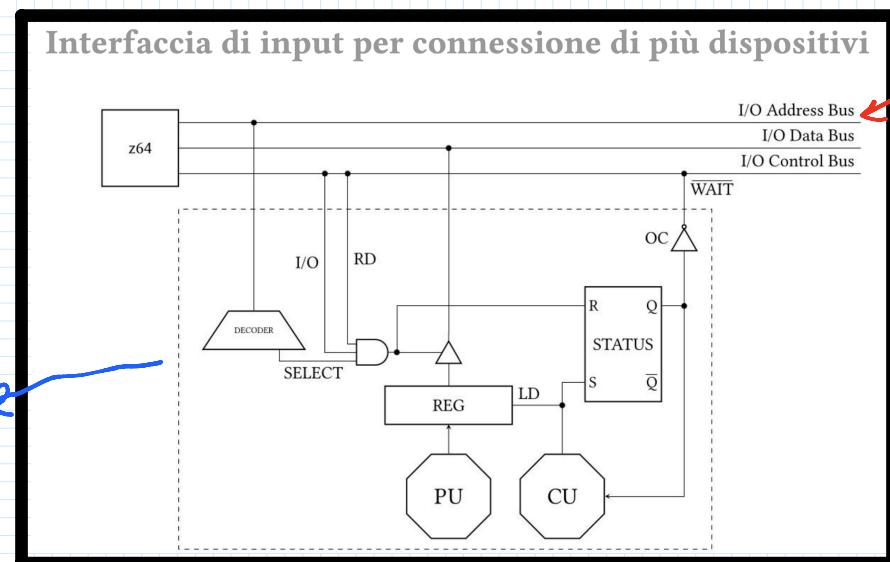
"Sto interrogando con te per leggere un dato, sto interagendo con te per scrivere un dato!"

PREVEDIAMO UN SEGNALE DI READ E UNO DI WRITE!

I/O È UN SEGNALE CHE DIFFERENZIA L'INTERAZIONE CON LA MEMORIA RISpetto ALL'INTERAZIONE CON I DISPOSITIVI;

ALL'INTERAZIONE CON I DISPOSITIVI:

COME FACCIO A GARANTIRE CHE UN SOLO REGISTRO METTA I DATI SUL DATA BUS?
Ci SERVE UN BUFFER THREE STATE!



è solo
dispositivo!
2

scrivo solo indirizzo
per volta.

→ 1 solo device
no scrivere
sul data bus perché
c'è il buffer 3-state
che dipende dal
segnale dell'address bus.

Il processore metterà sull'address bus l'indirizzo del dispositivo, sul control bus scritta a 1 I/O e READ, l'AND vale 1, si ABILITA il Buffer three state, e il contenuto del registro viene inviato dentro al processore.

Come facciamo a fare sì che il mio processore stia in stand-by mentre questo dato viene letto? se la CPU ha completato il suo lavoro, WAIT sarà 0.

Il Flip Flop viene pilotato dal segnale di status;

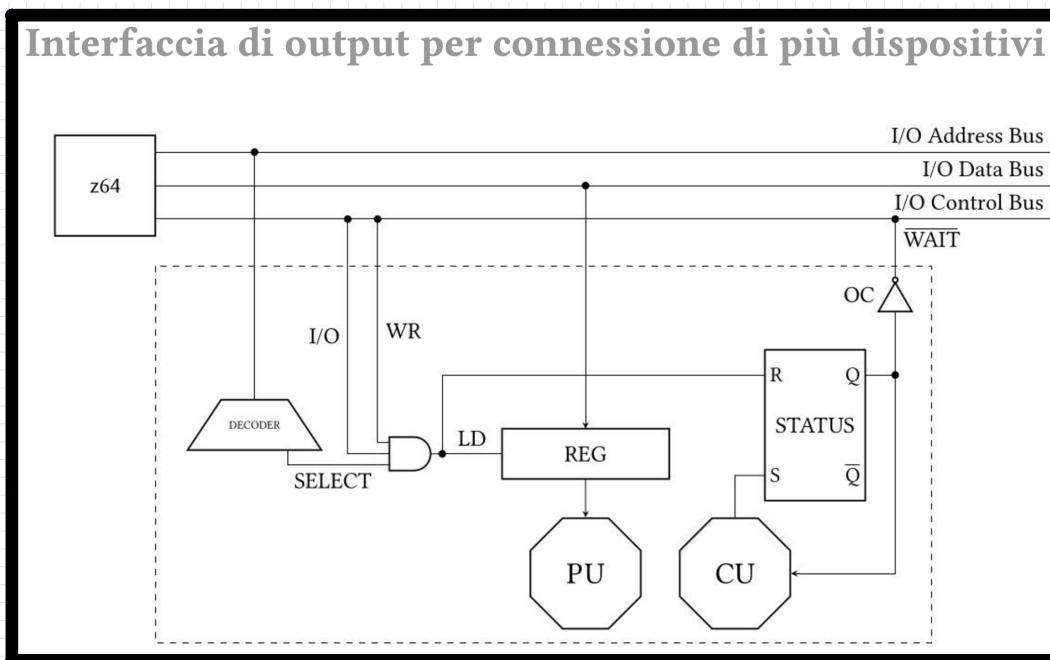
↳ la CPU escita dalla sua fase di stand-by;

Però più dispositivi vogliono dare al mio processore di stare in stand-by, e quindi tutti i dispositivi mandano corrente su quell'filo e la corrente si somma

↳ meglio che WAIT sia negato;

Perciò se lo I/O device dice 1 = devi staccare, in realtà logicamente deve zero e la tensione va a zero!

Interfaccia di output per connessione di più dispositivi



È il processore che trasferisce dati verso il dispositivo;

Il processore scriverà i dati sul DATA BUS, tutti i dispositivi vedranno il dato che il processore sta scrivendo, perché è un canale condiviso;

Soltanto uno dei esce il dispositivo che li acquisisce!

Il processore scriverà sull' ADDRESS BUS l'indirizzo del dispositivo verso cui vuole effettuare questa trasformazione!

Il DATA BUS ABILITERÀ il segnale di select tramite il decoder e la CPU comunica anche che sta interagendo con il dispositivo per scrivere dei dati!

Lo fa tramite i segnali di I/O e di WR!

È solo il dispositivo che riceve il segnale di SELECT, ed è questo quello che ci permette di identificare il dispositivo su cui scrivere.

L'AND VALE 1 E ABILITIAMO LA SCRITTURA SUL REGISTRO

PERÒ INFORMIAMO che c'è un dato in output che può essere consumato dal dispositivo stesso!

↳ INFORMIAMO LA CU E RESETTIAMO IL FLIP FLOP DI STATO;

IL VALORE viene letto dalla CU.

QUANDO L'UNITÀ DI PROCESSAMENTO HA FINITO DI SCRIVERE I DATI SUL REGISTRO DELL' UNITÀ DI CONTROLLO AL PROCESSORE CHE IL TRASFERIMENTO È COMPLETATO. IL SEGNALE DI WAIT E' VA IN STALLO FINCHÉ IL DISPOSITIVO ha finito di consumare quel dato.

PROBLEMA: CI VUOLE TEMPO;

ci sono tanti cicli di clock!

per tutto il tempo che la CU e la PU lavorano su quelli dati, la CPU STA FERMATA.
NON VA BENE.

IL PROBLEMA STA SULLO STALLO DEL PROCESSORE, IL PROCESSORE AFFIENA A WIA IL DISPOSITIVO SI GRAMMA.

SARÀ IL SOFTWARE CHE RISOLVERÀ QUESTO PROBLEMA;

Dobbiamo programmare e vedere se un dispositivo ha effettivamente finito di processare i dati oppure no; POSSIAMO INVENTARCI UN MECCANISMO SOFTWARE CHE CONSENTE ALLA CPU DI FARE ALTRO MENTRE I DISPOSITIVI STANNO CONSUMANDO I DATI O GENERANDO DATI NEL CASO DI DISPOSITIVI DI INPUT.

SPOSTIAMOCI SUL IO BASATO SU SOFTWARE, E POSSIAMO VEDERE 3 INTERAZIONI: