

HOLD E HLDA

Costituiscono l'interfaccia verso il controllore di DMA. Quando un dispositivo desidera acquisire il controllo del bus, porta a 1 il segnale HOLD.

A questo punto il processore, terminato il corrente ciclo di bus, pone in alta impedenza i segnali di ABUS, e i segnali di controllo. Il bus è quindi liberamente utilizzabile dal DMA. Quando il dispositivo rilascia il bus, riporta a 0 il segnale HOLD.

SEGNALI INTERRUPT

Esistono:

- + interruzioni mascherabili mediante istruzioni assembler.
- + interruzioni non mascherabili

In ogni sistema esiste un controller dell'interrupt che raccoglie le richieste di interruzioni e le veicola in modo opportuno al microprocessore (mediante il piedino di interruzione) e in ultima analisi all'unità di controllo.

Per quale ragione è possibile disabilitare gli interrupt:?

- sono in una sezione critica di codice che non deve essere interrotto (ad esempio il codice del kernel sta schedulando dei processi e non si vuole che una ipotetica stampante mandi il suo interrupt e lo interrompa)
- sto servendo un'interruzione di un periferico ad alta priorità (es. disco)

Esistono due istruzioni (EI ed DI) che lavorano in modo da attivare o disattivare il mascheramento degli interrupt (ad esempio agendo opportunamente su un flip flop).

Esiste comunque una opportuna categoria di interrupt chiamati interrupt non mascherabili che non sono governati dalle istruzioni EI ed DI. Questi interrupt sono particolarmente importanti per il sistema e per questo motivo devono essere recepiti in ogni caso dalla CPU, anche se questa sta eseguendo sezioni critiche di codice.

SEGNALE READY

Ready rappresenta un segnale di sincronizzazione con l'esterno.

All'esecuzione dell'istruzione WAIT, il processore testa il segnale READY e, se vale 1, inizia ad eseguire dei cicli di idle; quando READY torna a 0, il processore esegue l'istruzione successiva alla WAIT.

BEi

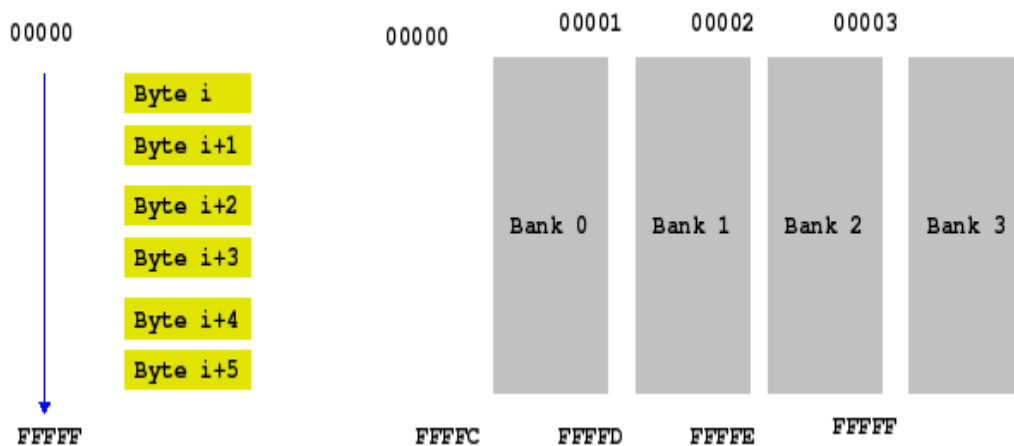
I segnali Bank Enable selezionano i/il byte interessati al trasferimento determinato da una data istruzione.

Quando la CPU mette sul bus un particolare indirizzo di memoria non specifica la tipologia di dato che vuole leggere (byte, word,...). Questa informazione è specificata dal codice operativo dell'istruzione. Il codice operativo dell'istruzione non è però noto al bus (e quindi alla memoria). E' necessario veicolare sul bus un'informazione che permetta di specificare il tipo di trasferimento: i

segnali BEi servono proprio a questo scopo.

La memoria è logicamente organizzata come una sequenza senza soluzione di continuità di byte (ognuno raggiungibile da un indirizzo). La visione fisica della memoria deve essere coerente con l'ampiezza del data bus. Questo fa sì che l'organizzazione fisica della memoria non sia composta da un byte dietro l'altro ma secondo uno schema a banchi.

Ogni banco contiene una parte della parola di larghezza corrispondente al data bus. Per esempio, in presenza di quattro banchi, il primo conterrà la parte alta della parola all'indirizzo, il secondo la seconda parte più alta della parola e così via. La seconda parola sarà di nuovo memorizzata divisa sui quattro banchi. Ovviamente una parola corrisponde in dimensione al parallelismo del data bus. I segnali BEi non fanno altro che selezionare il banco o i banchi fisici della memoria cui corrisponde un dato indirizzo (andando dunque a selezionare i gruppi di bit che ci interessano trasferire e la loro dimensione).



N.B. se vogliamo definire un vettore il cui indirizzo di testa appartiene all'ultimo banco e ogni elemento è da 16 bit, ci ritroviamo di fronte a un caso critico. Infatti il primo elemento ha 8 bit che appartengono a una parola, mentre gli altri 8 bit appartengono alla parola successiva.

Con un unico ciclo di bus non si riesce a prelevare l'intero elemento in quanto è possibile prelevare con un ciclo di bus solamente elementi appartenenti alla stessa parola.

Un'unica istruzione viene dunque divisa in due istruzioni equivalenti che prelevano metà elemento per volta.

Anche in questo caso i segnali BEi selezionano i banchi che ci interessano.

LOCK

Serve per eseguire sul bus una sequenza di cicli non interrompibili.

Un bus può avere più bus master e in certi casi specifici il master deve poter compiere due cicli di bus in sequenza senza essere interrotto. Andrà quindi ad attivare il segnale di Lock che finché non verrà rilasciato gli garantirà il possesso del bus. Un tipico esempio è dato dalla realizzazione di semafori per sezioni critiche (magari in architetture con multiprocessore).

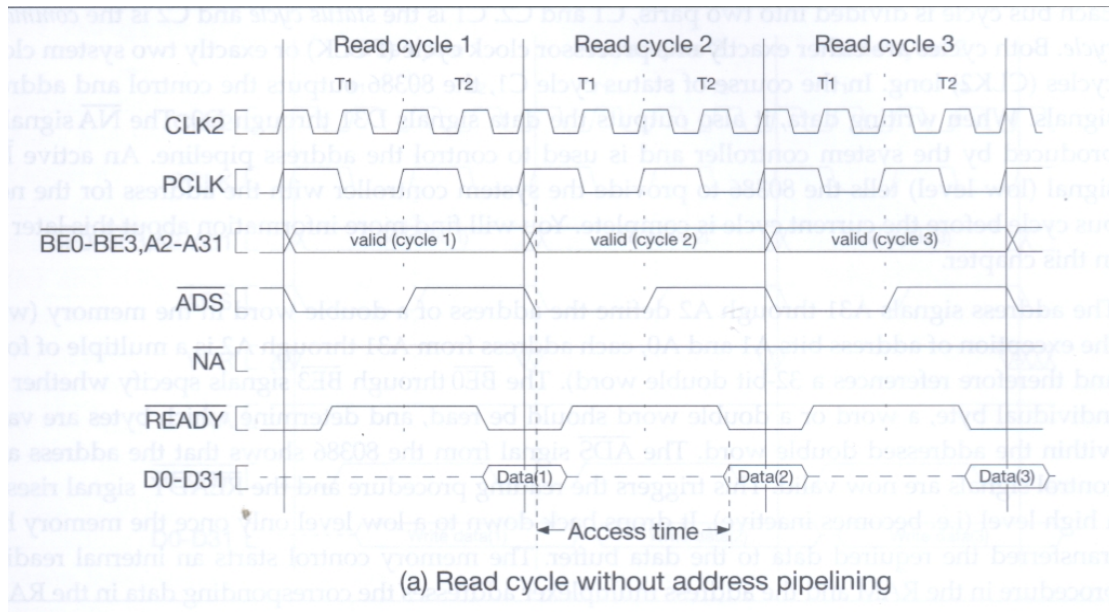
PENTIUM BUS CYCLE

Tutti i processori moderni hanno il supporto per due tipologie di cicli di bus:

- single transfer: transazione sul bus relativo al trasferimento di un singolo dato
- burst cycle: permette di trasferire una molteplicità di dati corrispondente a una linea di cache L1.

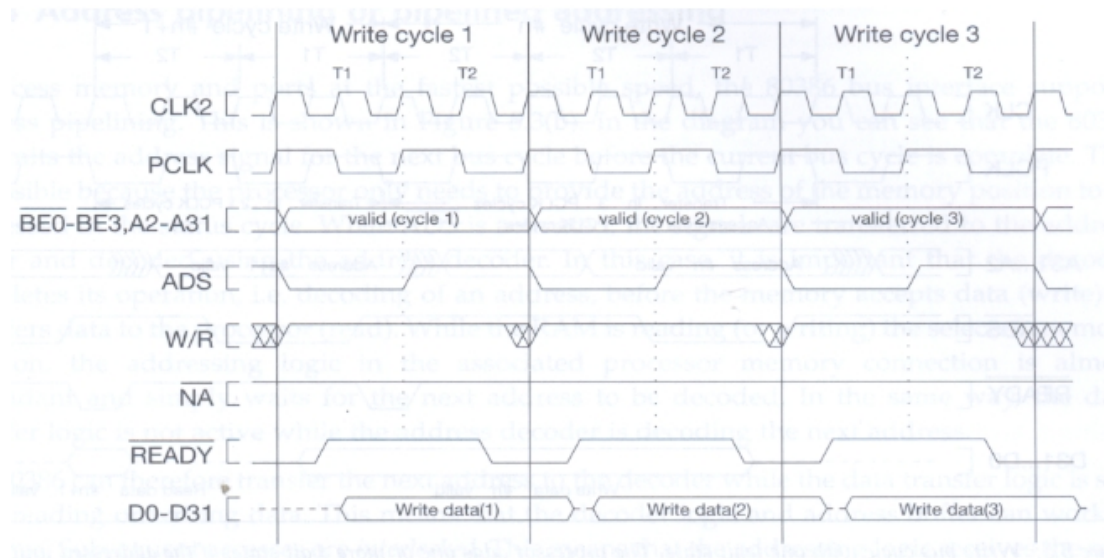
Permette cioè l'aggiornamento della cache nei casi di miss.

CICLO DI LETTURA

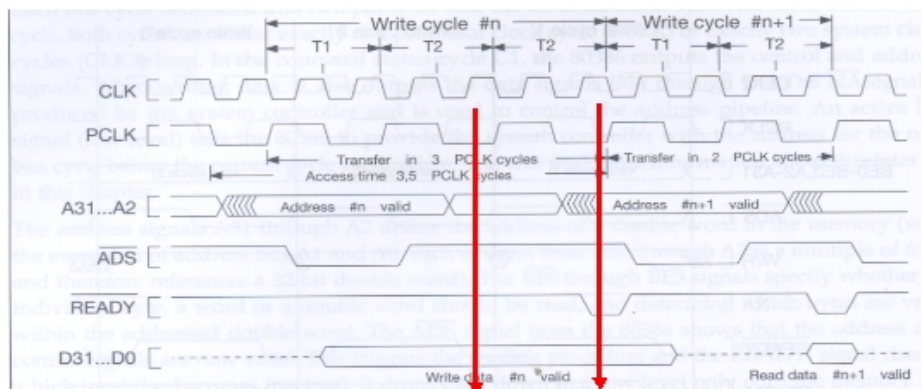


N.B. il bus di indirizzi dovrebbe essere da 32 bit, ma per risparmiare pin se ne usano solo 30. La parte di indirizzo mancante viene veicolata dai segnali BEi. Infatti se si utilizzassero 32 bit di indirizzo più i 2 segnali di BEi si avrebbe sovrabbondanza di informazione.

CICLO WRITE



CICLO DI WRITE CON WAIT

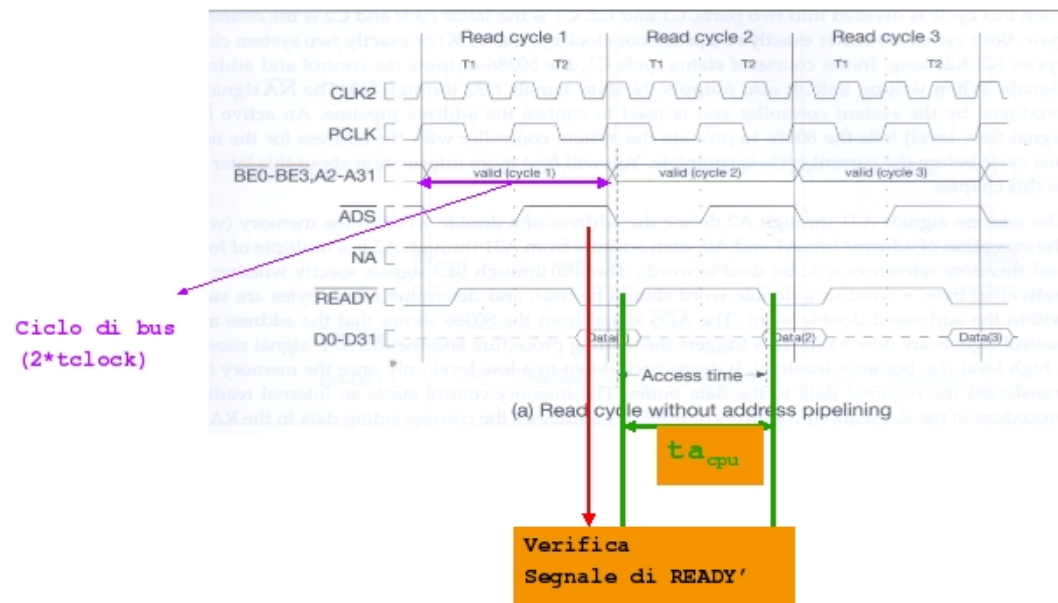


Verifica

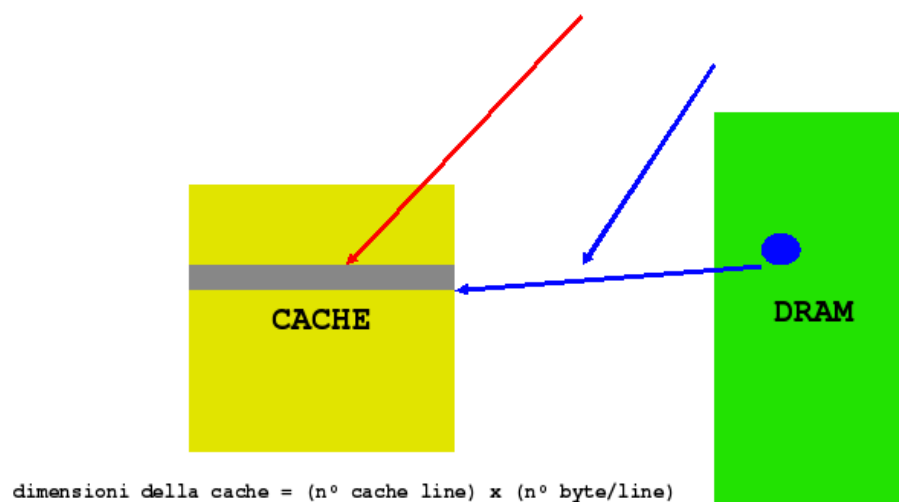
Segnale di READY': nel primo caso

La memoria NON è pronta, nel secondo SI

PENTIUM BUS CYCLE (ESEMPIO CON DRAM)



CACHE



La cache (full-associative, direct mapping, ...) possiamo vederla come una memoria organizzata in una serie di linea. Ogni linea è composta da n byte ed essendo gli n byte prelevati da zone contigua di memoria.

La capacita di una cache è $\text{Capacita} = N \text{ Linee} * N \text{ byte per linea}$. Ovviamente piu linee ci sono piu sono le aree di memoria che riusciamo a mappare. Piu le linee sono capienti e piu queste aree di memoria sono grandi.

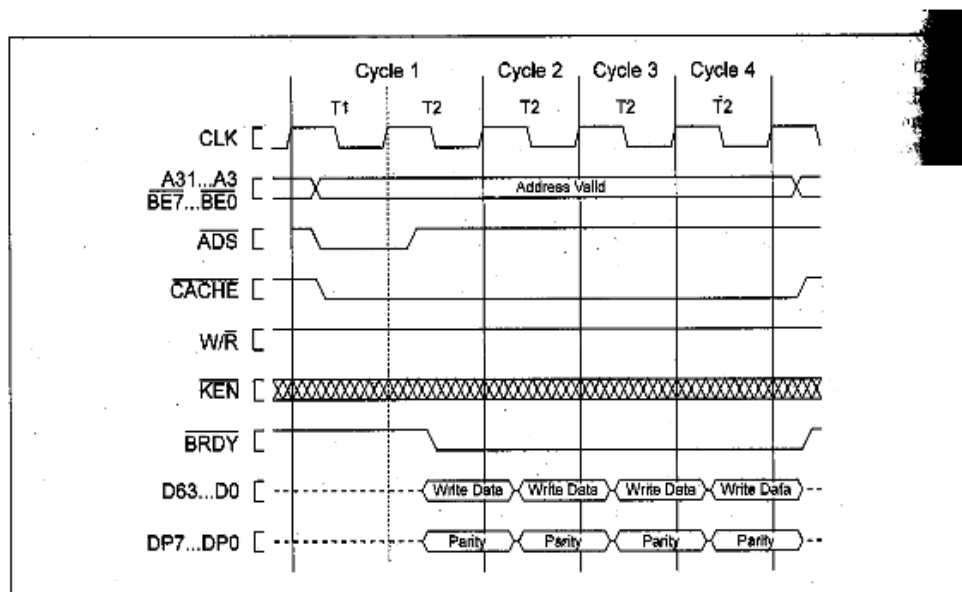
E' comunque da considerare il fatto che il trasferimento da memoria a cache è molto lento (infatti la velocita è imposta dal dispositivo piu lento, in questo caso dalla DRAM). L'operazione di trasferimento è molto dispendiosa per la cpu. Il tempo di aggiornamento della cache deve essere contenuto in un intervallo di tempo che non sia penalizzante per le prestazioni del processore.

Da notare è il fatto che all'aumentare della dimensione della linea aumenta il tempo richiesto per il trasferimento (le linee non devono quindi essere esageramente grandi).

Nei processori senza cache il parallelismo del data bus coincide con l'ampiezza massima dei registri (com'e' ovvio che sia). Invece, nei processori aventi cache-on-chip il parallelismo del data bus è tale da ottimizzare il numero di transazioni da eseguire per aggiornare le linee di cache ed è svincolato dalla dimensione dei registri della cpu.

Esiste una convenzione rispettata da quasi tutti i processori che prevede che il tempo massimo di aggiornamento della cache è di quattro cicli di bus. E' quindi implicitamente definita la dimensione della linea di cache: sara infatti $4 * \text{parallelismo DBus}$.

BURST CYCLE



Un ciclo di bus è composto da 4 transazioni da due periodi di clock ciascuno. E' possibile pero

sapere a priori l'indirizzo che dobbiamo prelevare nel prossimo ciclo (saranno infatti tutte parole di indirizzi contigui). Non c'e' dunque bisogno di generare un nuovo address strobe in quanto è implicitamente noto l'indirizzo.

Nel primo ciclo si eseguo un ciclo generico. A partire dal secondo ciclo si puo realizzare un ciclo ristretto in cui l'indirizzo è implicito (la durata è solo di un ciclo di clock).

Un burst cycle è dunque un insieme di cicli di bus in cui solo il primo è completo.

Esempio di definizio di ciclo di burst

2-1-1-1

che indica il numero di periodi di clock per ogni transazione.