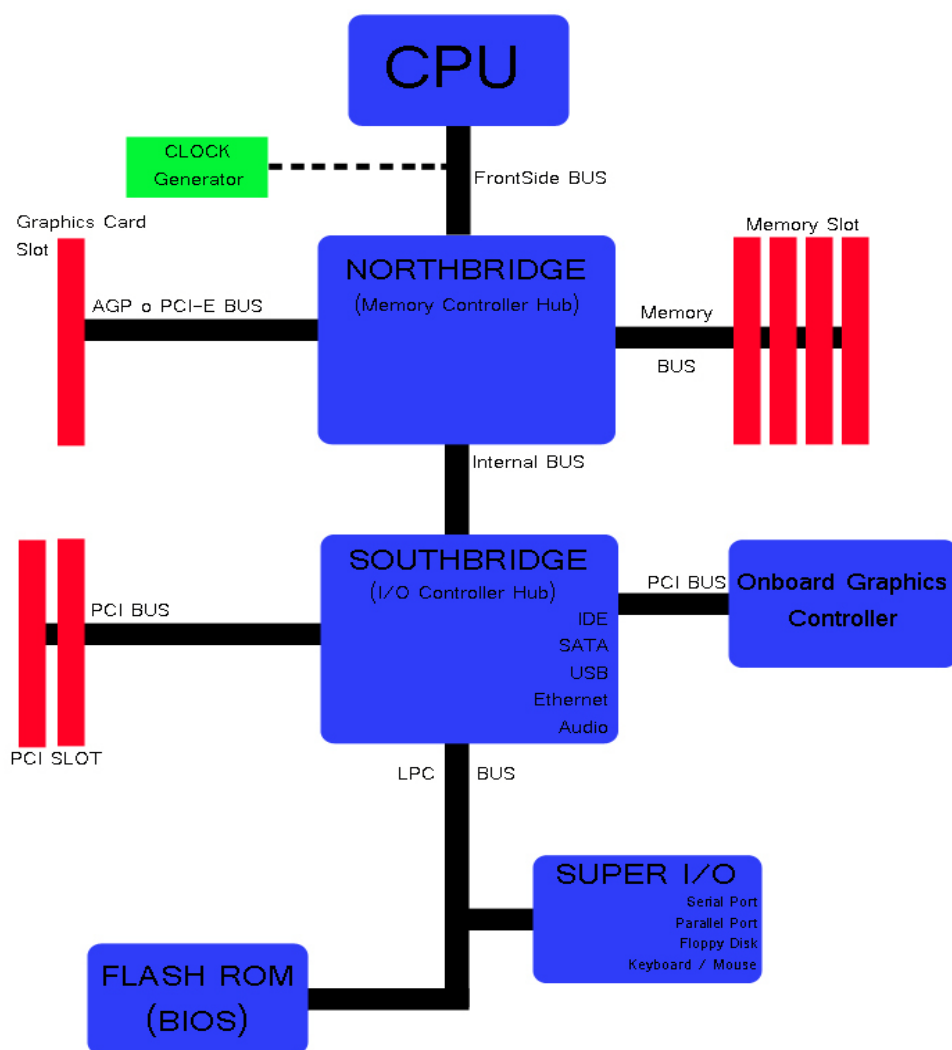


## Lezione 02-10-08

### Bus di sistema e di CPU nell'architettura 80x86 (vedi slide da 102 a 107)

L'architettura di un desktop può essere rappresentata con uno schema a blocchi. In questo caso l'architettura non è più a singolo bus ma è multi level bus. L'immagine di seguito mostra una sua rappresentazione:



Il Northbridge si occupa della gestione efficace della memoria, contiene il cache control (attraverso opportune transazioni sul bus si occupa dei miss e degli hit, quindi del caricamento e dello scarico in memoria cache delle porzioni di memoria principale) e non gestisce gli I/O.

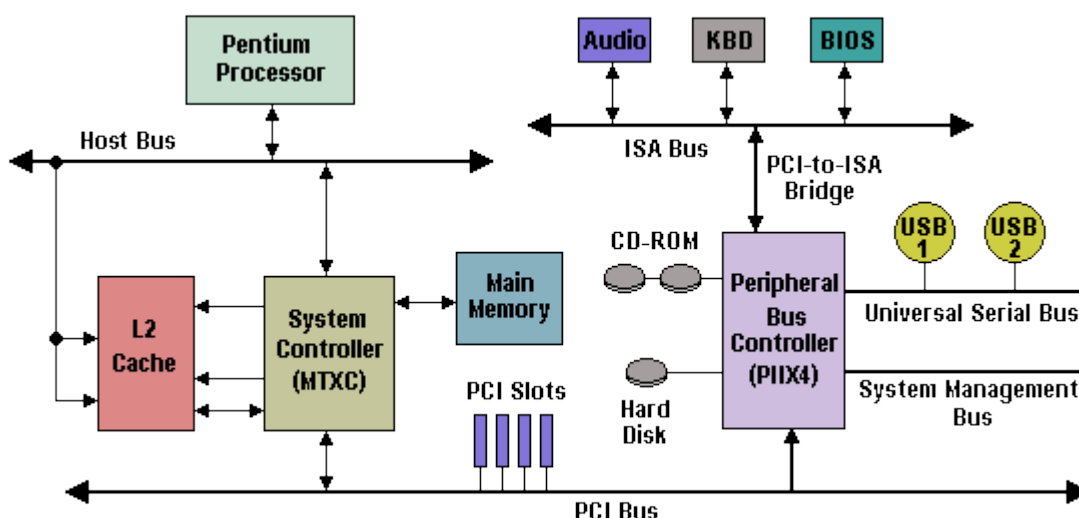
Ha come bus di riferimento il Front Side Bus (FSB), il bus di memoria e il bus veloce e orientato alla grafica (PCIExpress, AGP).

Il Southbridge e il Super I/O, quando presente, gestiscono le periferiche di I/O. Oltre ad avere una connettività di tipo PCI e Bus, ha una connettività quasi punto-punto con le periferiche.

Il BIOS è collegato alle periferiche attraverso il bus LPC: in questo modo, quando un processo invoca un'istruzione del BIOS, per raggiungere la CPU tale istruzione deve attraversare tutti livelli della pila architetturale, rallentando considerevolmente la velocità di esecuzione del processo stesso. Per ovviare a questo problema si adotta la procedura nota come Shadow RAM, che consiste nella copia del BIOS dalla EEPROM alla più veloce RAM di sistema per migliorare le prestazioni nell'esecuzione delle routine del BIOS. Questa procedura permette di ottenere un guadagno nelle prestazioni del sistema. Naturalmente, funziona esclusivamente con quei sistemi operativi che si interfacciano al BIOS per alcune operazioni (come MS-DOS che si appoggia al BIOS per la scrittura a schermo).

Per poter richiamare correttamente tutte le funzioni del BIOS, la porzione di memoria RAM che conterrà la copia di quest'ultimo deve mantenere gli stessi indirizzi del BIOS. Per far ciò si usa la tecnica del BANK SWITCHING<sup>d</sup>.

Il bus ISA è stato sostituito dal bus LPC (Low Pin Count) che minimizza il numero di piedini di connessione tra le periferiche, sostituendo ad un bus molto articolato come l'ISA (che conteneva tra le altre cose il DMA controller) delle connessione numericamente più ridotte.



<sup>d</sup> [http://en.wikipedia.org/wiki/Bank\\_switching](http://en.wikipedia.org/wiki/Bank_switching)

Nel Triton 430TX chipset, si può notare come sia la memoria che la L2 Cache sfruttano lo stesso bus (FSB) annullando la possibilità di svolgere delle azioni che risiedono sia in memoria che nella L2 Cache. Ad esempio, in caso di miss nella L1 Cache si deve aspettare che il trasferimento dalla L2 alla L1 termini prima di passare ad istruzioni successive.

Per ovviare a questo problema si adopera una tecnica chiamata Dual Independent Bus, figlia della cosiddetta Harward Architecture<sup>e</sup>, ovvero un'architettura di molti anni fa in cui il processore si interfacciava con due coppie di bus. Un bus che permetta di fare il fetch delle istruzioni e un altro di andare a scrivere in memoria nella fase di esecuzione. Si ha quindi un BACKSIDE BUS privilegiato tra CPU e cache e un FRONT SIDE BUS separato per la main memory, in modo tale che le transazioni sia sulla memoria che sulla L2 cache possano essere eseguite in parallelo.

## **CICLO DI BUS (vedi slide da 108 a 117)**

Un ciclo di bus è composto da un insieme di periodi di clock.

Al giorno d'oggi, il clock del FSB del Pentium è 200 MHZ, mentre quello della CPU è dell'ordine dei GHZ. Nell'8086 il ciclo di clock interno ed esterno durava 4 periodi, mentre nei processori o bus di sistema attuali è stato standardizzato ed è tendenzialmente pari a 2 periodi di clock, salvo casi particolari.

La velocità di clock è legata al throughput del bus, che dipende dalla frequenza di clock e dal parallelismo del bus dati. A parità di parallelismo, il throughput è tanto più alto quanto più la frequenza è alta e viceversa.

Esiste comunque un limite all'aumento della frequenza del Front Side Bus perché, superata una certa soglia, le piste che collegano i chip hanno delle lunghezze paragonabili a quelle della lunghezza d'onda generando possibili comportamenti indesiderati.

Quindi, per aumentare il throughput non viene modificato il clock, ma si utilizza una tecnologia a bus double data rate<sup>f</sup>, in cui i trasferimenti vengono effettuati sia sul fronte di salita che su quello di discesa del clock.

Se prima un periodo di clock effettuava un solo trasferimento, ora si cerca di effettuarne di più nell'unità di clock e la soluzione più banale è di trasferirli sia sul fronte di salita che su quelli di discesa.

I global data rate sono dei meccanismi che permettono di eseguire o adottare questa tecnica.

<sup>e</sup> [http://en.wikipedia.org/wiki/Harward\\_architecture](http://en.wikipedia.org/wiki/Harward_architecture)

<sup>f</sup> [http://en.wikipedia.org/wiki/Double\\_data\\_rate](http://en.wikipedia.org/wiki/Double_data_rate)

Non si parla più di bit/s ma di trasferimenti al secondo (MT/s o GT/s) per rendere l'indice di prestazioni indipendente dal timing e indipendente dal parallelismo.

Se si ha, ad esempio, una frequenza di clock a 100 MHz double data rate, è come avere una frequenza di salita e discesa e quindi un clock doppio; in tal caso si ottengono 200 MT/s.

Se poi si ha un bus da 64 bit (quindi 8 byte) si ottengono  $200 \times 8$  MT/s; se si ha un bus da 1 byte si otterrà  $200 \times 1$  MT/s e così via.

Il Transfer Rate attuale del Pentium 4<sup>g</sup> ha un fattore di moltiplicazione 4 (four data rate, quad data rate<sup>h</sup>): in un periodo di clock di sistema il bus trasferisce 4 volte, con tecniche<sup>i</sup> di un certo tipo.

Il front side bus di un Pentium varia da 100 MHz in avanti (100, 133, 200 MHz), fino ad una velocità massima equivalente di 1066 (quindi 266 MHz), come mostrato nella tabella sottostante.

Real Clock	Performance	Transfer Rate
100 MHZ	400 MHZ	3.2 GB/s
133 MHZ	533 MHZ	4.2 GB/s
200 MHZ	800 MHZ	6.4 GB/s
266 MHZ	1066 MHZ	8.5 GB/s

Dal punto di vista hardware, il ciclo di bus è un'entità atomica non ulteriormente divisibile, cui è affidata una transazione tra il processore e il mondo esterno. Quindi le transazioni di bus sono fondamentalmente 4:

- leggere memoria
- scrivere memoria
- scrivere I/O
- leggere I/O

La transazione di bus è un concetto logico che si esplica in un certo numero di periodi di clock, ma che è logicamente e funzionalmente un corpus unico legato a quelle che sono le entità atomiche più piccole ed elementari che si realizzano in un computer. Un elaboratore lavora per transazioni sul bus.

<sup>g</sup> [http://it.wikipedia.org/wiki/Pentium\\_4](http://it.wikipedia.org/wiki/Pentium_4)

<sup>h</sup> [http://en.wikipedia.org/wiki/Quad\\_Data\\_Rate](http://en.wikipedia.org/wiki/Quad_Data_Rate)

<sup>i</sup> [http://it.wikipedia.org/wiki/Quad\\_pumped](http://it.wikipedia.org/wiki/Quad_pumped), [http://en.wikipedia.org/wiki/Pumping\\_\(computer\\_systems\)](http://en.wikipedia.org/wiki/Pumping_(computer_systems)),  
<http://www.wipo.int/pctdb/en/wo.jsp?wo=2001048621>

Esempi:

### **ADD AX, BX**

È un'istruzione di 2 byte, che riesco a leggere in parallelo o no a seconda di quanto è largo il databus (il DBUS è almeno 16 bit). Richiede una transazione sul bus per caricare quell'istruzione nella memoria (fetch istruzione) e una seconda transazione sul bus per leggere il dato (fetch operando). Non ci sono altre transazioni perché il risultato è nel registro.

### **ADD [BX], AX**

Richiede 3 transazioni, 1 fetch istruzione, 1 fetch operando, 1 scrittura risultato.

Sarà anche vero che il risultato lo scrivo sulla L1 cache, ma dalla L1 il risultato deve essere poi scritto in memoria.

Quindi, dal punto di vista architetturale del sistema la funzione fondamentale è la transazione sul bus che non è suddividibile.

Nell'architettura 8086, i cicli di bus erano determinati da 4 periodi di clock che poi sono stati ridotti a 2 pur mantenendo internamente un'articolazione ancora a 4.

Se si analizza la BIU e il clock della BIU, si può notare che è il doppio della velocità del clock del front side bus. Questo perché è una macchina a stati ed è più semplice, data la complessità, fare una macchina a 4 stati piuttosto che a 2 stati.

Vediamo come avviene questo tipo di sequenza di operazioni<sup>j</sup>.

Nel ciclo di lettura devo inizialmente scrivere l'indirizzo, poi attivare la memoria affinché si svegli dal “sopore” e mi riporti il dato. Dopodiché devo leggere questo dato: mando l'indirizzo sull'address bus e avviso la memoria che deve rispondermi; lascio passare un po' di tempo per 2 tipologie di operazioni:

- **DECODIFICA:** scrivo l'indirizzo e attivo i segnali di controllo. Poi però si ha la decodifica della memoria e la logica combinatoria veloce che deve rendere compatibile lo standard di richiesta sul bus con gli standard della memoria (la CPU non vede una memoria in banchi).
- **LETTURA DEL BUS** => trasferendo ciò che leggo in un registro.

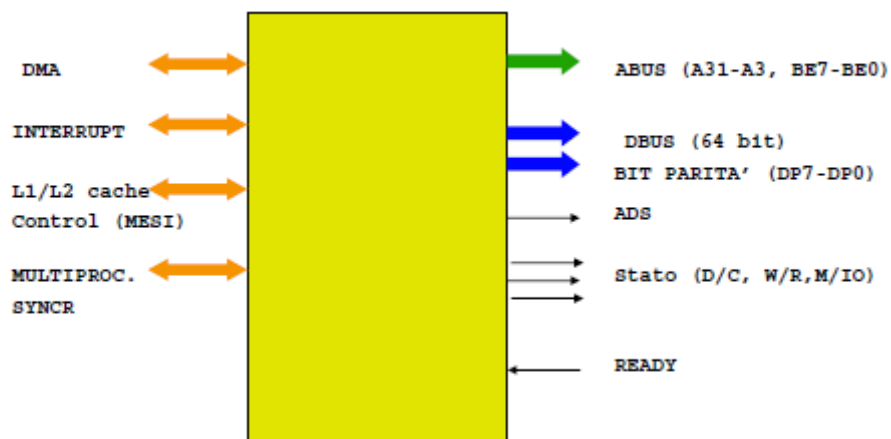
Il ciclo di scrittura è pressoché analogo con l'unica differenza che è “l'utente” a mettere il dato sul bus, quindi nel primo periodo si inserisce l'indirizzo e poi immediatamente il dato sul DBUS; si ha poi un lasso di tempo perché termini l'operazione.

Poiché il bus è di tipo sincrono non vi è nessun feedback da parte della memoria o delle interfacce sul fatto che l'operazione richiesta sia stata correttamente eseguita dalla CPU.

<sup>j</sup> [http://it.wikipedia.org/wiki/Architettura\\_di\\_un\\_processore\\_basato\\_su\\_registri\\_generali](http://it.wikipedia.org/wiki/Architettura_di_un_processore_basato_su_registri_generali),  
[http://it.wikipedia.org/wiki/Ciclo\\_del\\_processore](http://it.wikipedia.org/wiki/Ciclo_del_processore)

Avendo delle cache e delle code di prefetch esiste la possibilità che sul bus non ci sia niente da fare. Quindi tutti i processori, specialmente quelli un po' evoluti, prevedono i cicli o stati di IDLE che sono quei periodi di clock durante il quale non avviene alcuna transazione sul bus. I bus di tipo sincrono presuppongono che la periferica risponda nella temporizzazione che abbiamo detto, quindi bisogna trovare un meccanismo che permetta a memorie lente di sincronizzarsi. Questo meccanismo è il cosiddetto meccanismo di inserimento degli stati di WAIT.

### **Architettura di riferimento del Pentium (vedi slide da 118 a 123)**



L'architettura del Pentium ai piedini è determinata dai segnali realizzati e soprattutto dalle funzioni che sono assegnate ai gruppi di segnali.

L'address bus è di 32 bit (per avere una migliore gestione dei banchi fisici di memoria di fatto è articolato su un numero di bit inferiore). Indirizza 4 GB, cioè  $2^{32}$  bit, ma il numero di segnali che escono dalla CPU non sono 32.

In particolare, vi sono 3 segnali che permettono la selezione del banco e sono A0, A1 e A2 (quindi 3 bit che permettono di contare gli 8 byte). Per facilitare la gestione della memoria, invece di andare ad emettere da 0 a 31, gli ultimi 3 bit che indicano il byte all'interno del databus lo specificano in modo diretto.

Il databus fondamentalmente è composto da 2 flussi: i 64 bit tradizionali e gli 8 bit che sono associati ciascuno ad un byte e riportano in ingresso (e anche in uscita) il bit di parità associato al byte.

Il problema è che, su base statistica, il tasso degli errori o permanenti o non permanenti indotti sulla memoria è elevato. Questo fa sì che dal punto di vista architetturale si debbano introdurre dei meccanismi di rilevazione e/o correzione dell'errore.

Uno dei possibili meccanismi è quello di introdurre il bit di parità che deve essere attribuito ad una sequenza di bit corta per essere significativo. Per ovvie ragioni viene associato un bit di parità ad ogni byte di memoria.

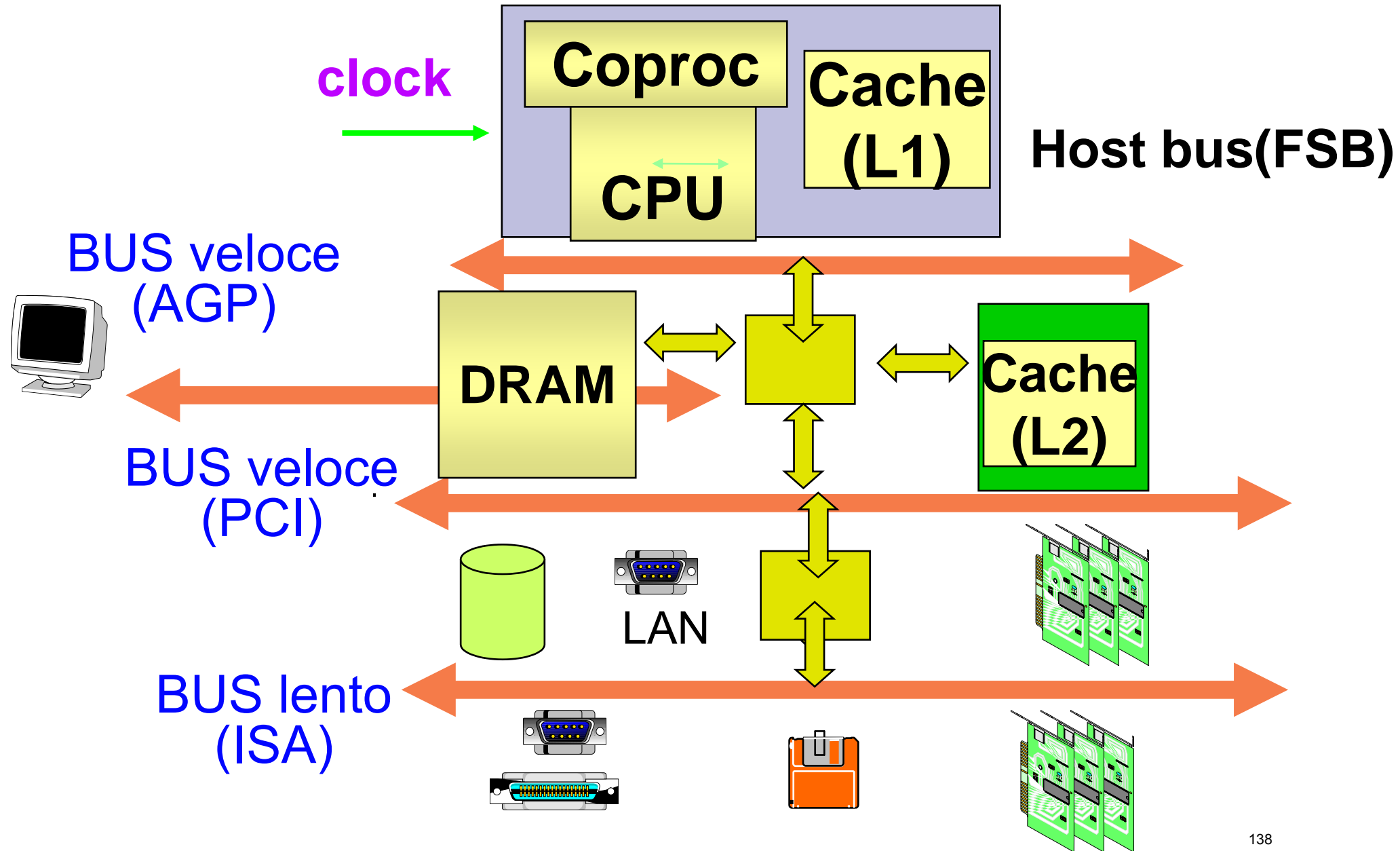
Altri segnali di controllo sono poi ad esempio:

- Address STROBE: sta ad indicare che gli indirizzi ad inizio del ciclo sono strobat (consolidati/disponibili) sul bus; è il segnale che determina l'inizio del ciclo.
- Segnali di stato: utili per definire il tipo di ciclo (lettura, scrittura, I/O o memoria).
- Segnale READY: permette la gestione del bus semisincrono; permette inoltre l'inserimento del ciclo di WAIT e la dilatazione di un ciclo di bus in funzione di un vincolo.

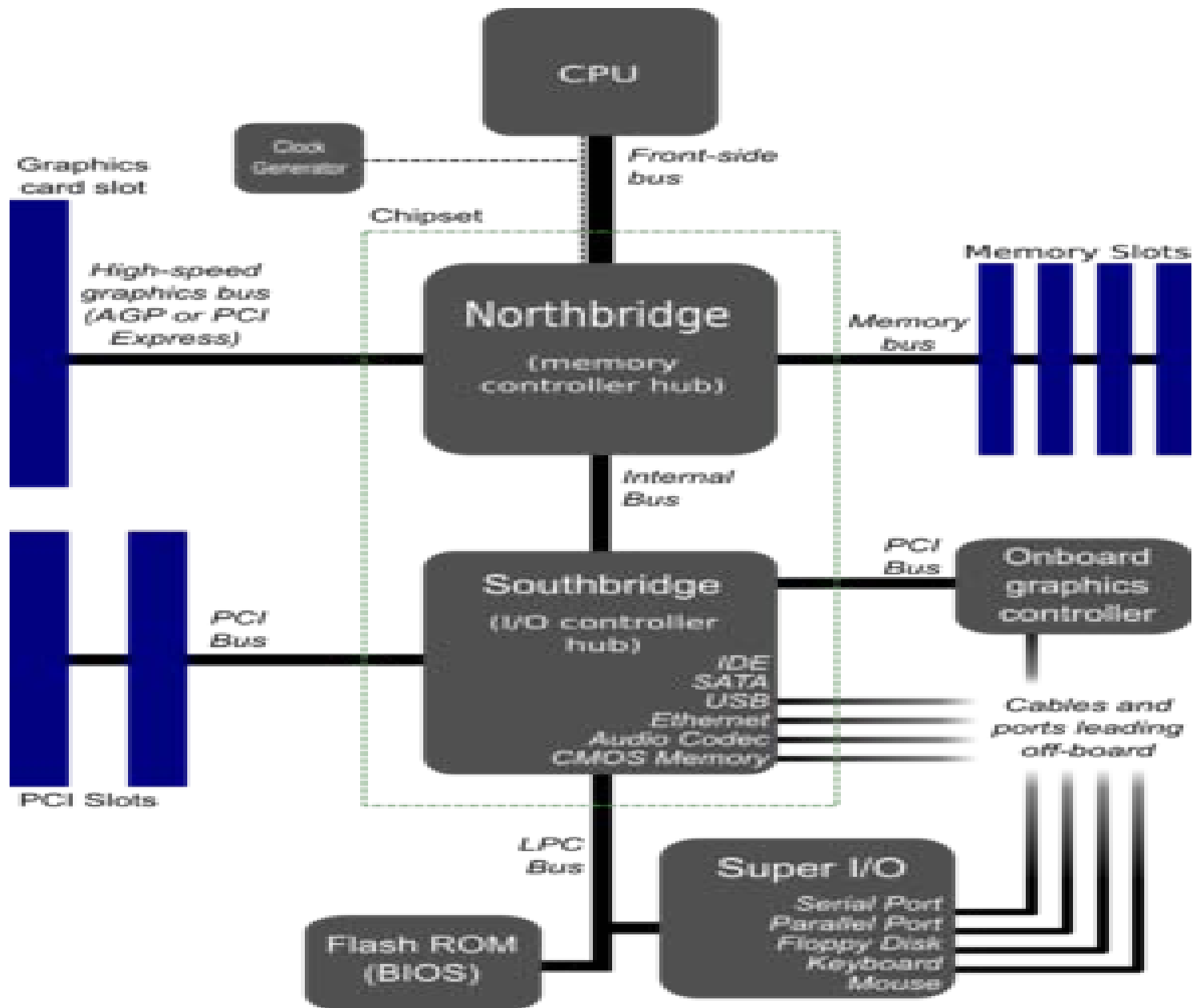
# LEZIONE 2 Ottobre 2008



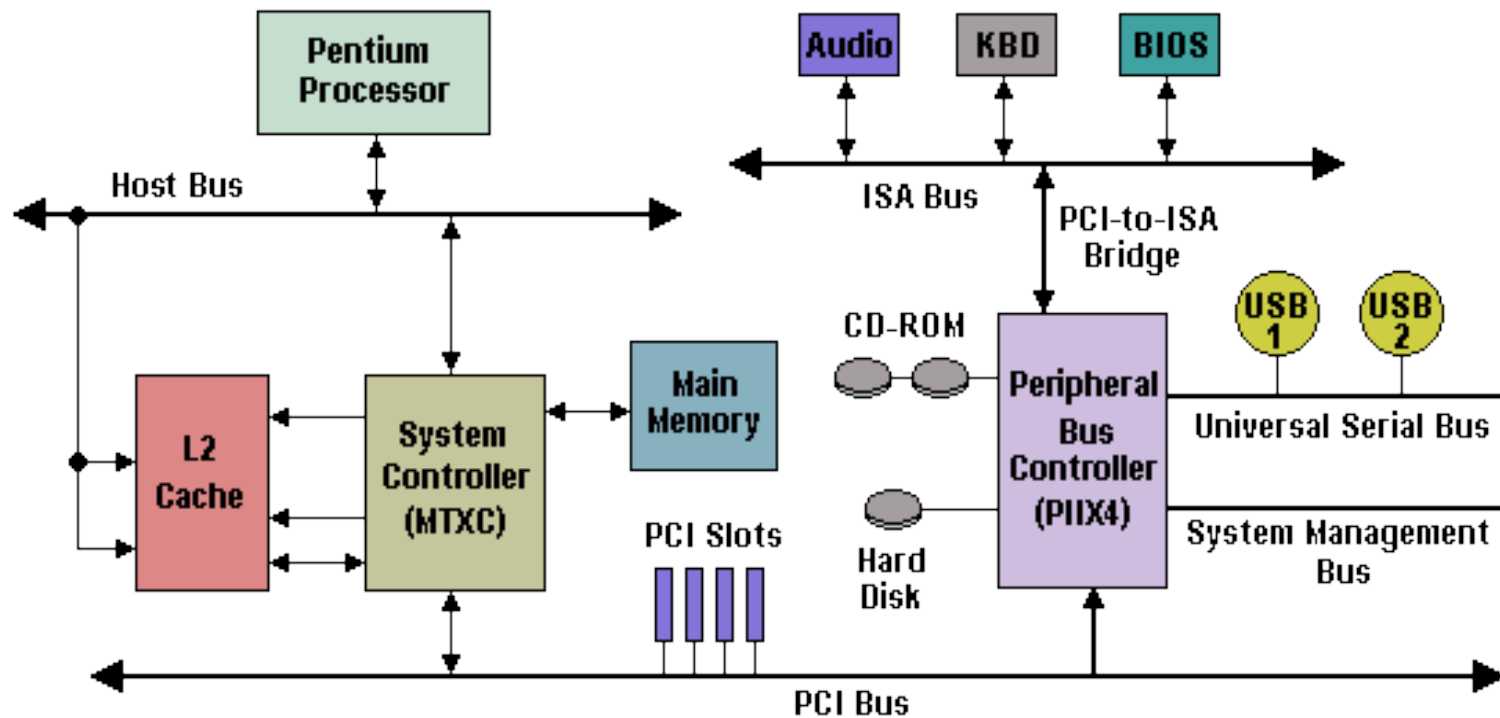
# DESKTOP SYSTEM



# DESKTOP SYSTEM



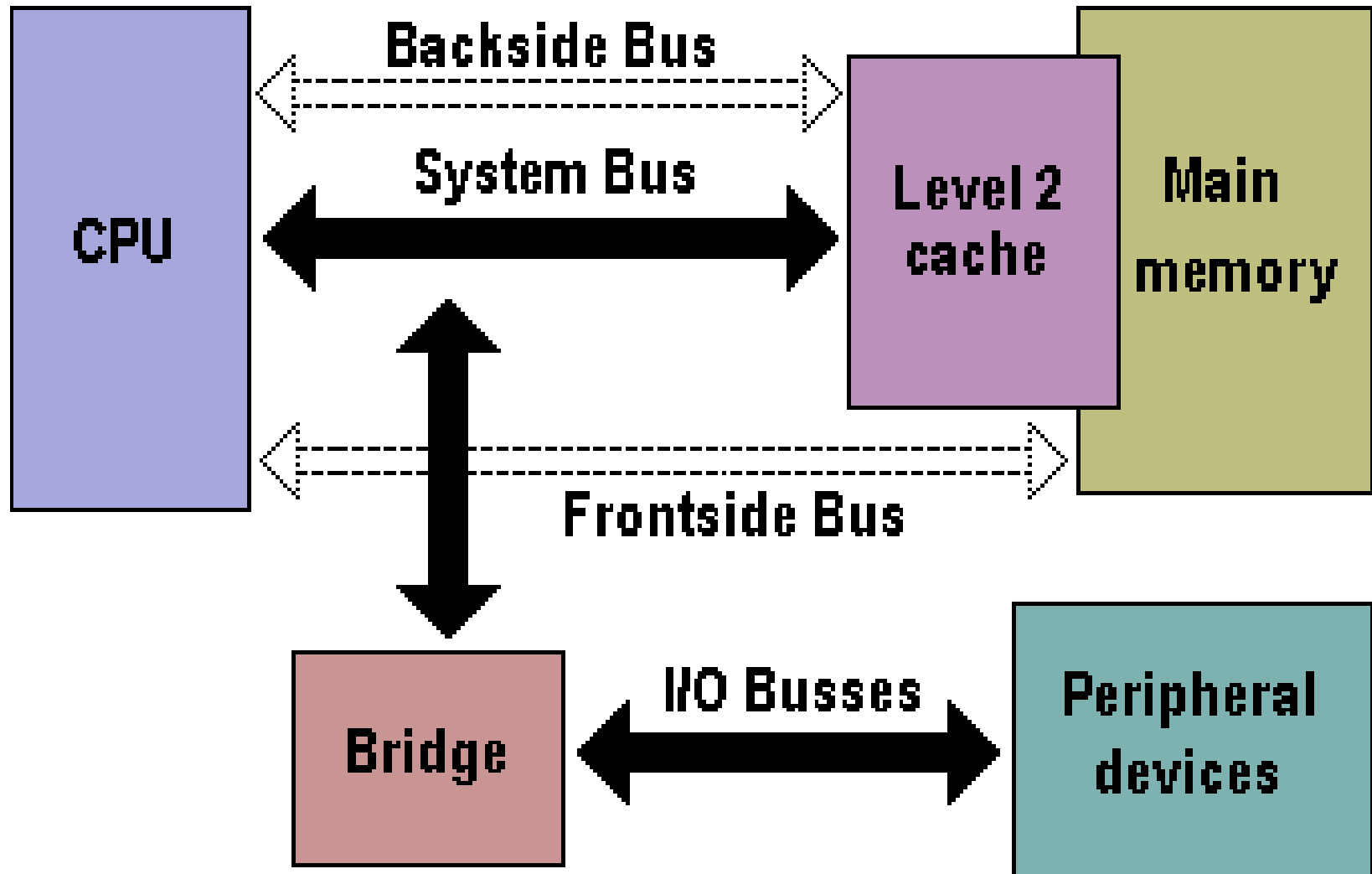
# DESKTOP SYSTEM (Triton430TX chipset)



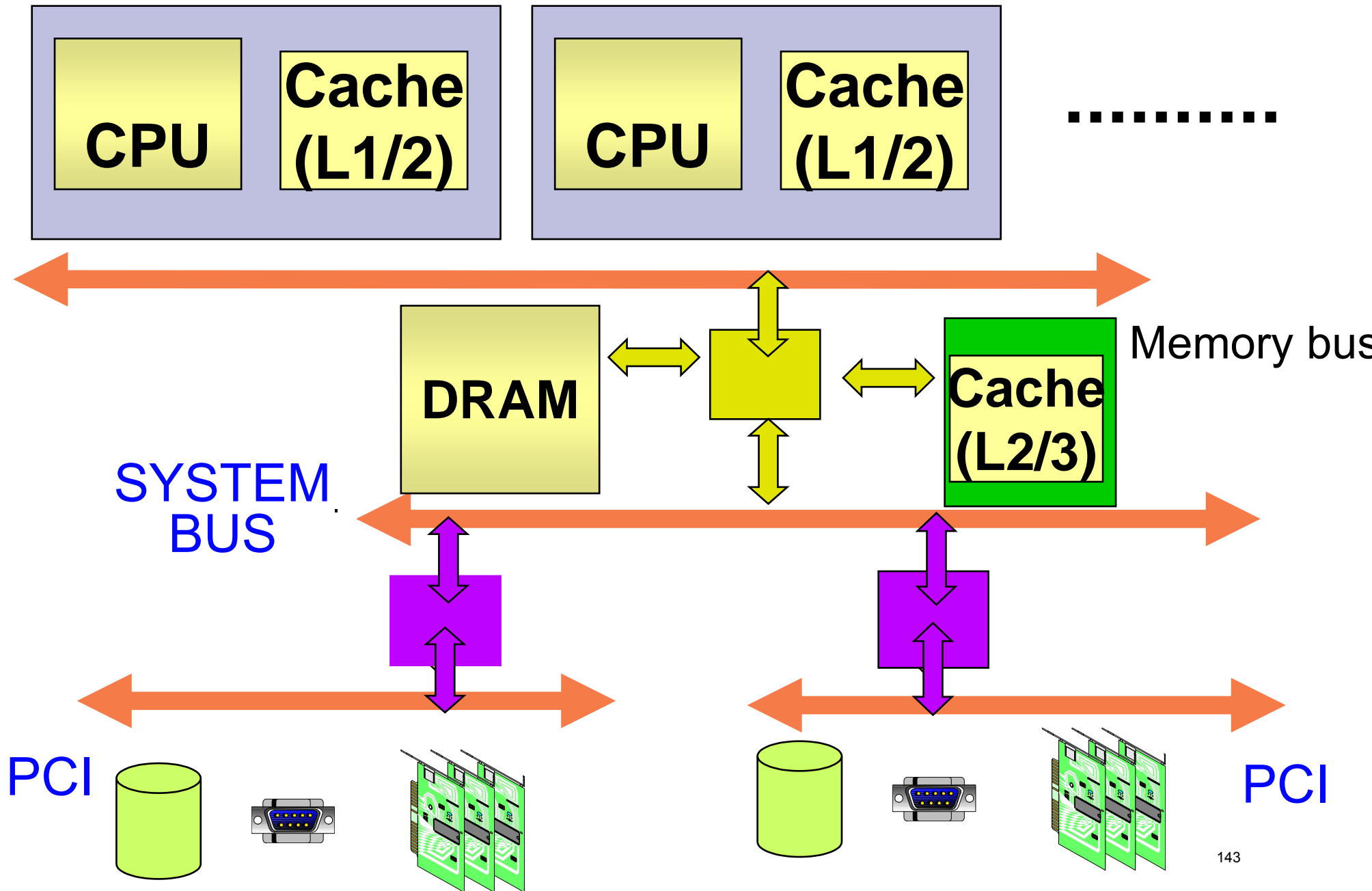
# Dual Independent Bus Arch.

In Dual Independent Bus (DIB) architecture systems the single system bus is replaced by a "frontside bus" for shuttling data between the CPU and main memory, and between the CPU and peripheral buses and a "backside bus" for accessing Level 2 cache. The use of dual independent buses boosts performance, enabling the CPU to access data from either of its buses simultaneously and in parallel.

# Dual Independent Bus Arch.



# SERVER SYSTEM



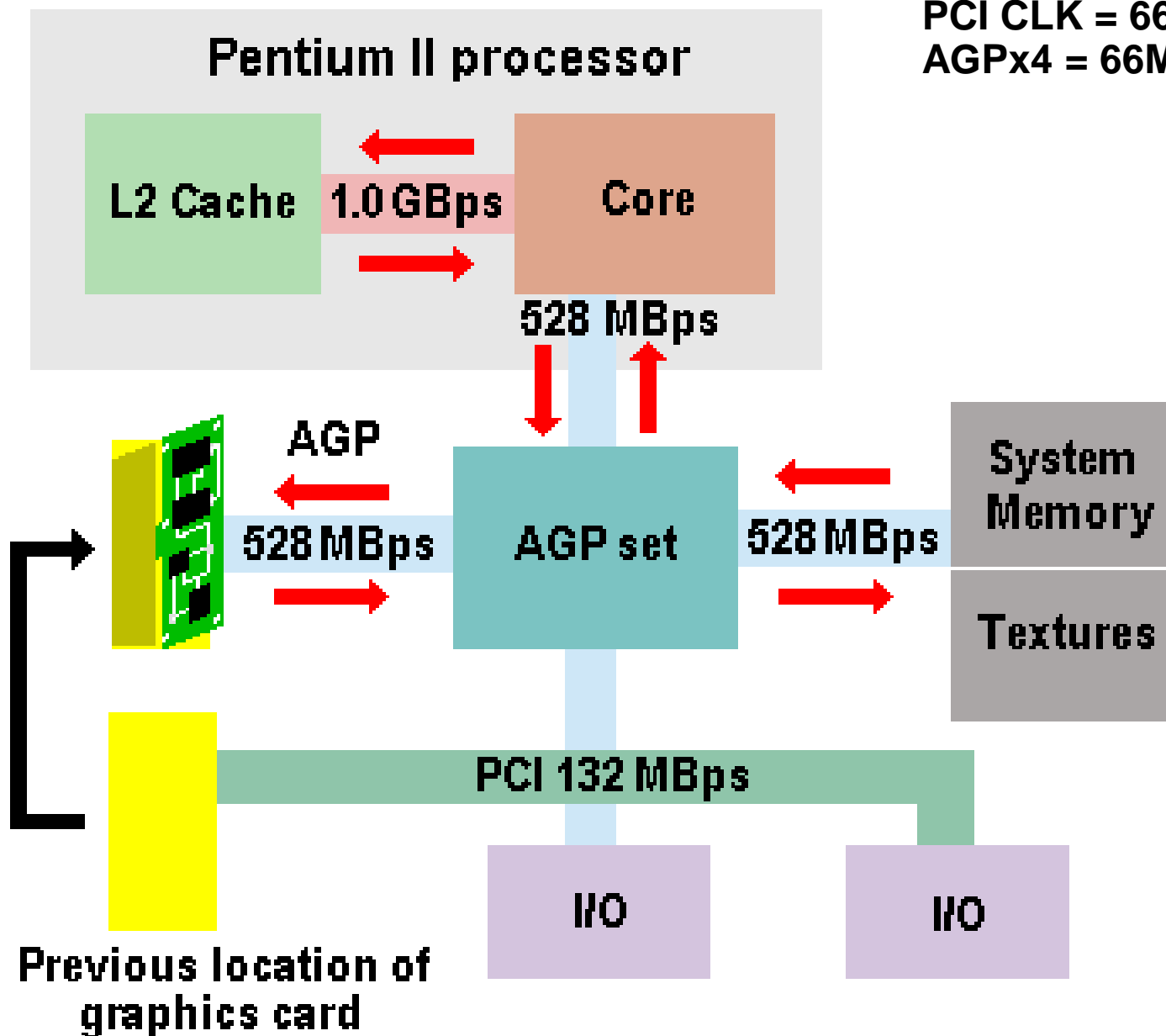
# Ciclo di Bus

- È la sequenza di eventi attraverso la quale la CPU comunica con la memoria, con un dispositivo di I/O, con l'Interrupt Controller.
- In funzione dei processori può essere costituito da 2 o 4 cicli di clock di sistema.
- Internamente esistono sempre 4 periodi
- Si compone di almeno 4 fasi, denominate  $T_1$ ,  $T_2$ ,  $T_3$ ,  $T_4$ .
  - $T_1$ : sull'address bus viene scritto l'indirizzo
  - $T_2$ ,  $T_3$ ,  $T_4$ : sul data bus viene messo il dato.
- Se la CPU non deve accedere all'esterno, i segnali di controllo del bus sono inattivi ed i relativi piedini sono in alta impedenza.
- Se nessun altro dispositivo utilizza il bus, questo si trova allora nello stato di *idle*.

# Ciclo di Bus

- Nel 8086 il ciclo interno (4 CLK) è identico al ciclo esterno di bus
- Nei processori tipo pentium o nei bus di sistema (es. PCI) il ciclo di bus è di **2 CLK**
- Ad esempio nel bus del pentium (host bus o memory bus) a 100MHz il trasferimento avviene in 2 CLK, cioè in 20ns.
- In generale si indica la capacità di trasferimento del bus espressa in MBps (Mega Byte per secondo)
- Esempio: pentium con DBUS a 64 bit e 100MHz si ha:
- $(64/8)*100M/2 = 400MBps$
- Essendo 2 i periodi di clock per ciclo di bus





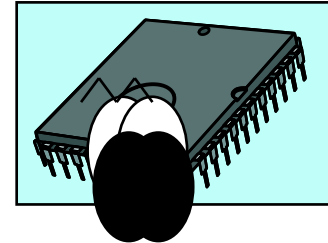
PCI CLK = 66MHz, 32 bit DBU  
AGPx4 = 66MHz x 4

# Gigatransfer - Megatransfer

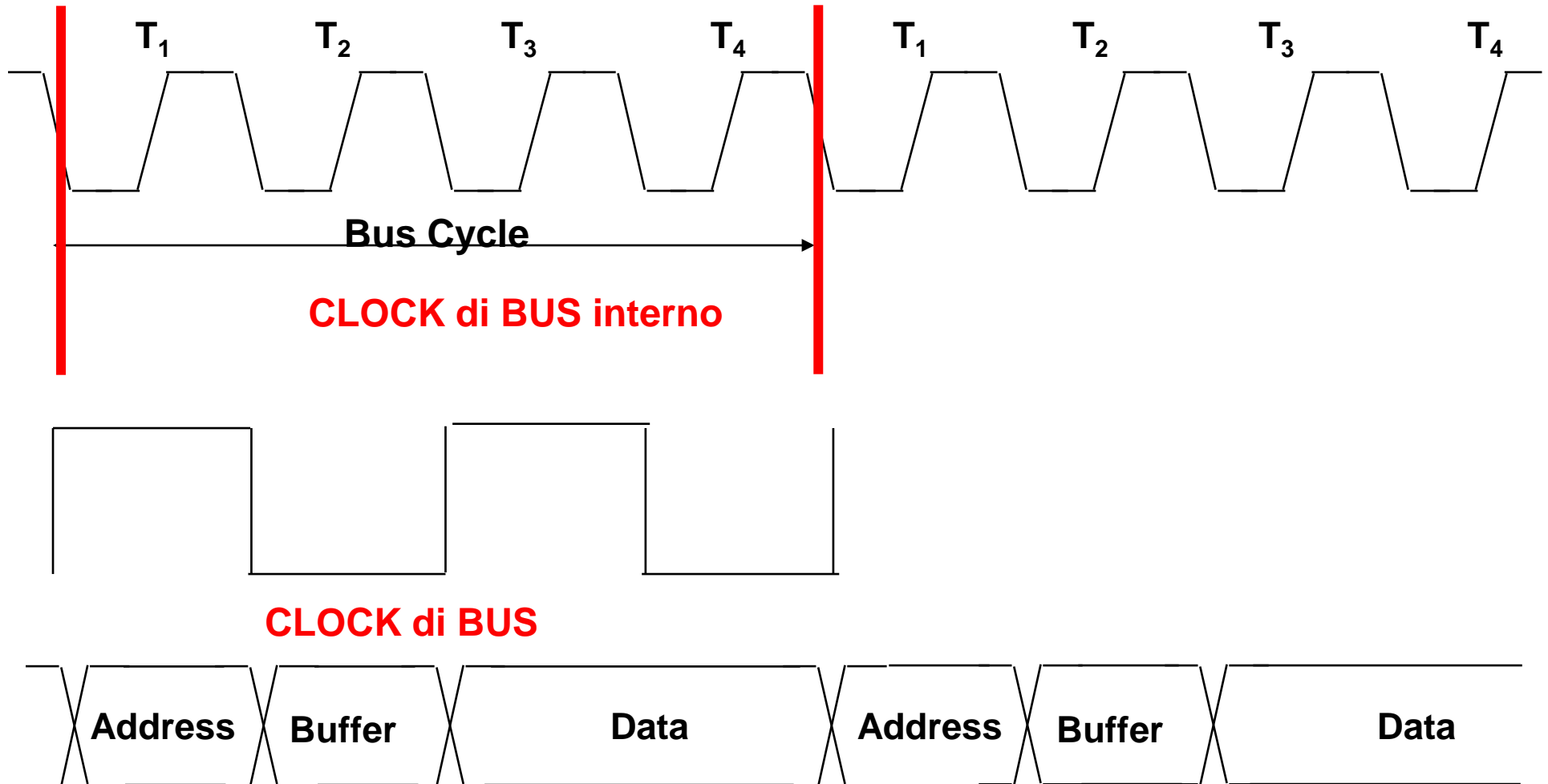
- **Gt/s** o **MT/s** indicano il n° di trasferimenti al secondo in alternativa a Bps. In tal modo si indicano le prestazioni del bus indipendentemente dal parallelismo o dal timing. Per esempio nei bus double data rate, in cui i dati sono trasferiti sul fronte di salita e di discesa.
- Se il clock è 100 MHz in tal caso(double data rate) il trasferimento effettivo è 200 MT/s,
- SCSI (Small Computer Systems Interface), PCI Express vengono usualmente definiti in termini di GT/s o MT/s
- Pentium IV transfer rate: 400 – 1066 MT/s

# Frequenza FSB

Perché non far  
crescere  
la frequenza del  
host bus?



# Ciclo di Bus



# Ciclo di Lettura

- $T_1$ : sull'address bus viene scritto l'indirizzo
- $T_2$ : la CPU forza sul data bus il valore Z
- $T_3, T_4$ : la memoria scrive il dato sul data bus.

# Ciclo di Scrittura

- $T_1$ : sull'address bus viene scritto l'indirizzo
- $T_2$ : la CPU scrive il dato sul data bus
- $T_3, T_4$ : la memoria legge il dato dal data bus.

# Cicli di Idle

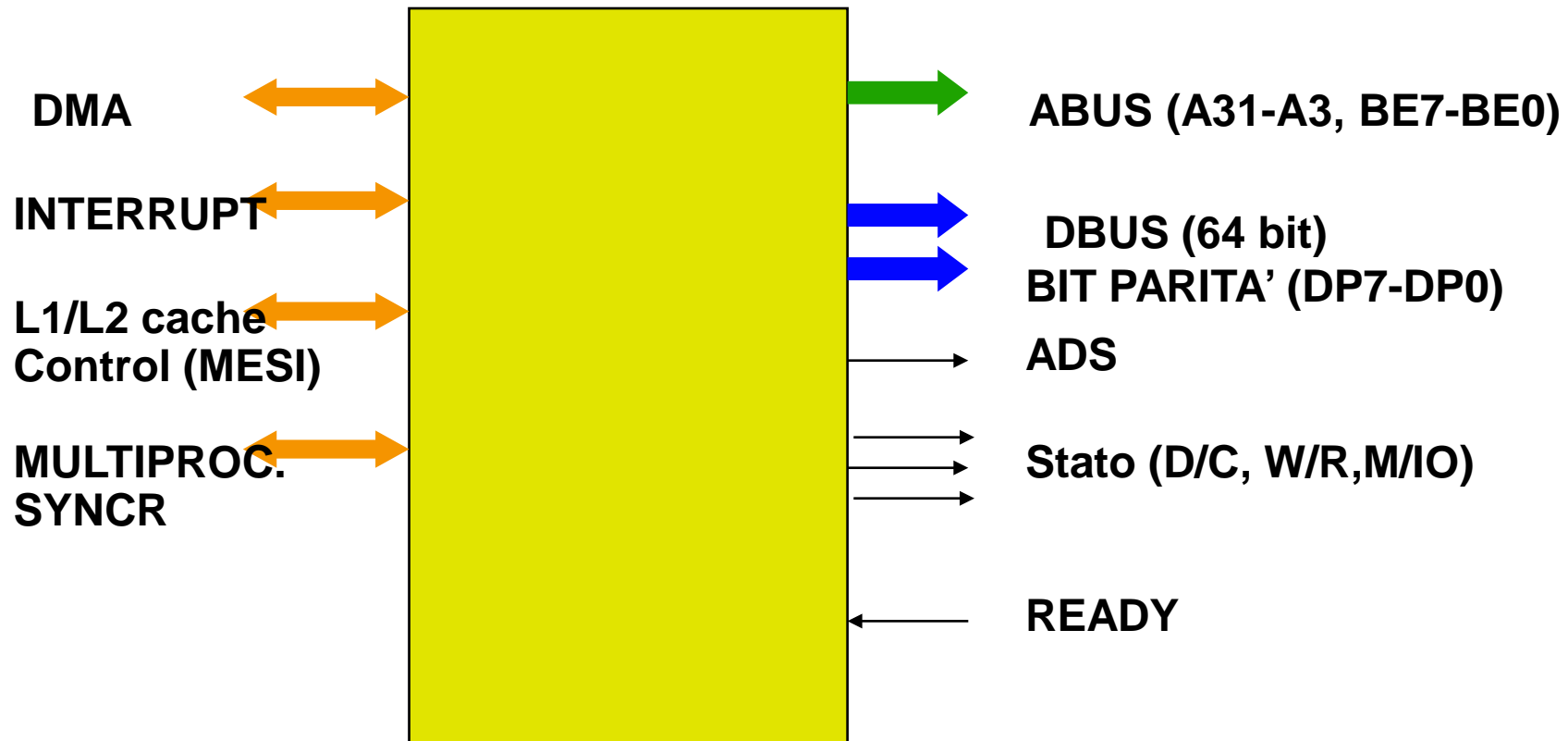
- Vengono inseriti dalla CPU quando necessario, ossia quando:
  - la CPU non necessita di nuovi dati e
  - la coda interna delle istruzioni è piena, e non può essere eseguita alcuna fase di prefetch.

# Cicli di Wait

- Se la memoria non è sufficientemente veloce, lo segnala alla CPU, e questa inserisce tra  $T_3$  e  $T_4$  una serie di stati di attesa (*wait states*) fino a che la memoria risponde.
- Per comunicare all'8086 la necessità di uno o più cicli di wait, la memoria esterna invia un segnale sul pin READY.



# PENTIUM



A partire dal Pentium II il bus indirizzi è da 36 bit (64 GB) e successivamente  
Il bus dati ha parallelismo 128/256

# Segnali di Controllo

- **ADS (o ALE):** il fronte di salita segnala durante  $T_1$  che sull'address bus è pronto un indirizzo.
- **IO/M\*:** indica se il ciclo di bus fa riferimento alla memoria o a un dispositivo di I/O;
- **R/W\*:** indica se si tratta di un ciclo di lettura o scrittura
- **C/D:** indica se sul bus sono presenti dei dati o no

# 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**
- **Quando il dispositivo rilascia il bus, riporta a 0 il segnale HOLD.**

# Segnali di Interrupt

- Sono:
  - INTR (input): richiesta di interrupt da parte di un dispositivo esterno
  - INTA\* (output): accettazione della richiesta da parte della CPU, e temporizzazione del trasferimento del codice di interrupt
  - NMI (input): richiesta di interrupt non mascherabile.

# 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

- Nell'80x86 il segnale BEi\* (*Bank Enable*), con i che varia tra 0 e il numero di byte di parallelismo del DBUS, definisce la dimensione e l'allineamento del tipo trasferito:
- BE0\* = 0 trasferimento del byte meno significativo
- BE1\* = 0 trasferimento del secondo byte
- ES.
- nell'istruzione MOV AL, (BX) viene attivato BE0
- nell'istruzione MOV AX, (BX) viene attivato BE0 e BE1
- nell'istruzione MOV EAX, (BX) viene attivato BE0, BE1, BE2 e BE3