Bus sincroni ed asincroni

Descrivere le differenze tra i bus sincroni e quelli asincroni e descrivere un protocollo asincrono

Esistono due schemi principali di comunicazione (temporizzazione) su di un bus:

- Bus sincroni:

  -Il protocollo di comunicazione e' scandito dal clock.

  -Ogni ciclo di bus per r/w richiede piu' cicli di clock

  I Vantaggi del sincrono sono:

  1. Molto veloce
  2. non richiede molta logica perche' tutti gli eventi sono sincronizzati con il clock

  Gli svantaggi invece:

  1. Ogni dispositivo deve lavorare alla stessa velocita' del clock, quindi sincronizzato
  2. Non puo' essere troppo lungo per non avere problemi di clock skew

     (clock skew e' la differenza tra l'istante atteso di clock ed il clock effettivo)

  Normalmente i bus processore-memoria sono sincroni: Lunghezza ridotta e pochi elementi.

- Bus asincroni

  -Non ha clock

  -Comunicazione avviene tramite handshaking

  I suoi vantaggi sono :

  1. Puo' essere molto lungo
  2. Il tempo impiegato dalle singole operazioni e' legato esclusivamente dalla velocita' delle parti coinvolte

  Svantaggi:

  1. Piu' lento
  2. Piu' complesso da progettare e gestire

     Normalmente i bus di I/O sono asincroni

Arbitraggio del bus

Spiegare le tecniche di arbitraggio dei bus, farne vedere una possibile implementazione(tramite disegno) e come si possa aumentare la banda passante di un bus.

Serve a decidere quale dispositivo sara' il prossimo autorizzato ad utilizzare il bus, serve ad evitare contese per l'accesso al bus:

- Assegna il bus ai dispositivi con priorita' piu' alta.
- Garantisce che non si verifichino situazioni di attesa indefinita o di blocco del sistema.

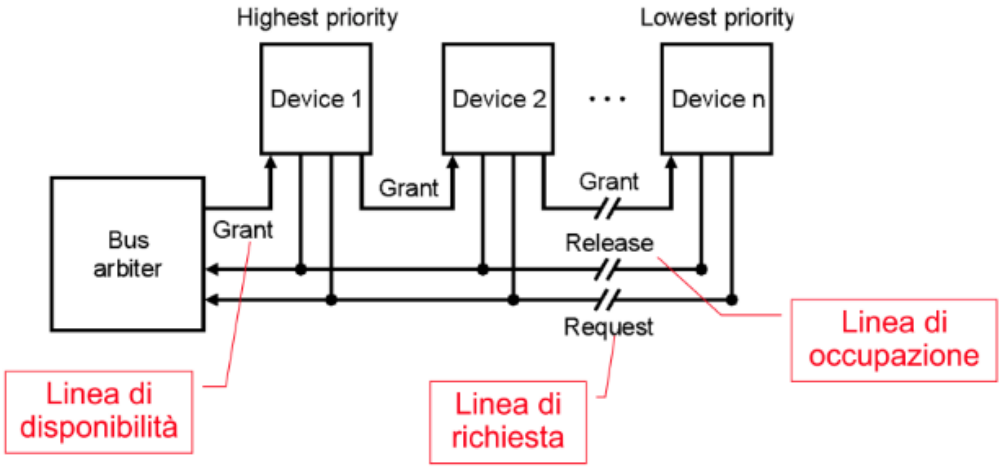
Gli schemi di arbitraggio si dividono in due:

- Arbitraggio centralizzato:

  Include un'unita controllore chiamata arbitro che arbitra il bus.

  Esistono 2 tipi di abitraggio centralizzato :

  1. Daisy Chaining: ad ogni dispositivo viene assegnata una priorita' (piu' e' vicino piu' priorita'). Il suo problema e' che non garantisce la fairness(ovvero la garanzia a tutti i dispositivi di accedere al bus)



  2. Indipendent requesting: ogni dispositivo ha una linea di richiesta e una linea di grant. Se il dispositivo ha bisogno di trasferire piu' dati e nessun'altro lo richiede puo' continuare ad utilizzarlo.

- Arbitraggio distribuito/decentralizzato:

  Non ha nessun controllore centralizzato, quindi i dispositivi seguono un algoritmo per l'accesso al bus.

Dire cosa e' la banda passante e far vedere come e' possibile aumentarne la capacita'

La banda passante di un bus e' determinata dalla quantita' di dati che possono essere trasportati sul bus in un unita' di tempo.

-Le tecniche per aumentare la banda passante sono:

1. Split transaction

   La transazione sul bus viene divisa in due parti: Transazione di richiesta e transazione di riposta. In questo modo si evitano i tempi di non utilizzo del bus, sfruttando meglio la banda del Bus.

   Svantaggio tempi di transazione lunghi

2. Parallelismo delle linee dati

   Aumento il numero di linee dati per far passare piu' dati

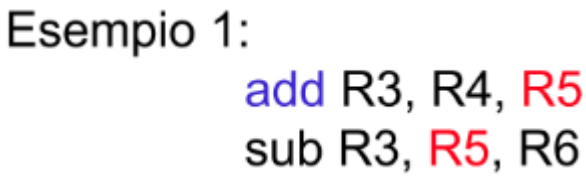
3. Linee per dati e linee per indirizzi separate
4. Tasferimento dei dati a blocchi

   Riduzione del tempo di risposta tramite il trasferimento di piu' parole contemporaneamente senza dover trasferire piu' indirizzi.

Conflitti di tipo define use

Dire cosa sono i conflitti di tipo define use e in che modo si risolvono via software, disegna pipeline RISC e scrivi frammenti di programmi per evidenziare i fenomeni e come risolverli

Abbiamo criticita' sui dati quando un'istruzione dipende dal risultato di un'istruzione precedente che e' ancora sulla pipeline.



In questo caso uno degli operandi sorgente di sub(R5) e' prodotto da add che e' ancora nella pipeline.

Questo tipo di criticita si possono risolvere a livello software inserendo delle istruzioni nop, ma peggiora il throughput, oppure semplicemente riordinando le istruzioni che non sono legate ad altre istruzioni.

Altro metodo e' risolvere le criticita' a livello hardware inserendo bolle/stalli ma si inseriscono tempi morti e peggioriamo il throughput. Si puo' utilizzare anche il metodo della propagazione, propagando dati in avanti (non appena disponibili) verso l'unita' che li richiedono.

Criticita' strutturali sui dati e sul controllo di un processore RISC

Descrivere cosa sono le criticita' strutturali, sui dati e sul controllo di un processore RISC, e come si possono risolvere, a tal fine si utilizzi l'architettura di riferimento introdotta a lezione.

Le criticita' delle architetture pipeline nascono quando non e' possibile eseguire un'istruzione nel ciclo immediatamente successivo.

Esistono 3 tipi di criticita'

1. Criticita' strutturali:

   Quando si tenta di utilizzare la stessa risorsa hardware da parte di diverse istruzioni in modo diverso ma nello stesso ciclo di clock. Per esempio non si puo' leggere e scrivere contemporaneamente ad una fase di writeback.
2. Criticita' sui dati:

   Si possono dividere in 2 tipi : define-use e load-use

Quando si tenta di utilizzare un risultato prima che sia disponibile. Per esempio non posso eseguire una istruzione che dipende dal risultato dell'istruzione precedente se questa e' ancora nella pipeline

3. Criticita' sul controllo:

   Quando nel caso dell'uso di salti condizionati decido quale sia la prossima istruzione prima che la condizione sia valutata

Modalita' interazione tra DMAC e CPU

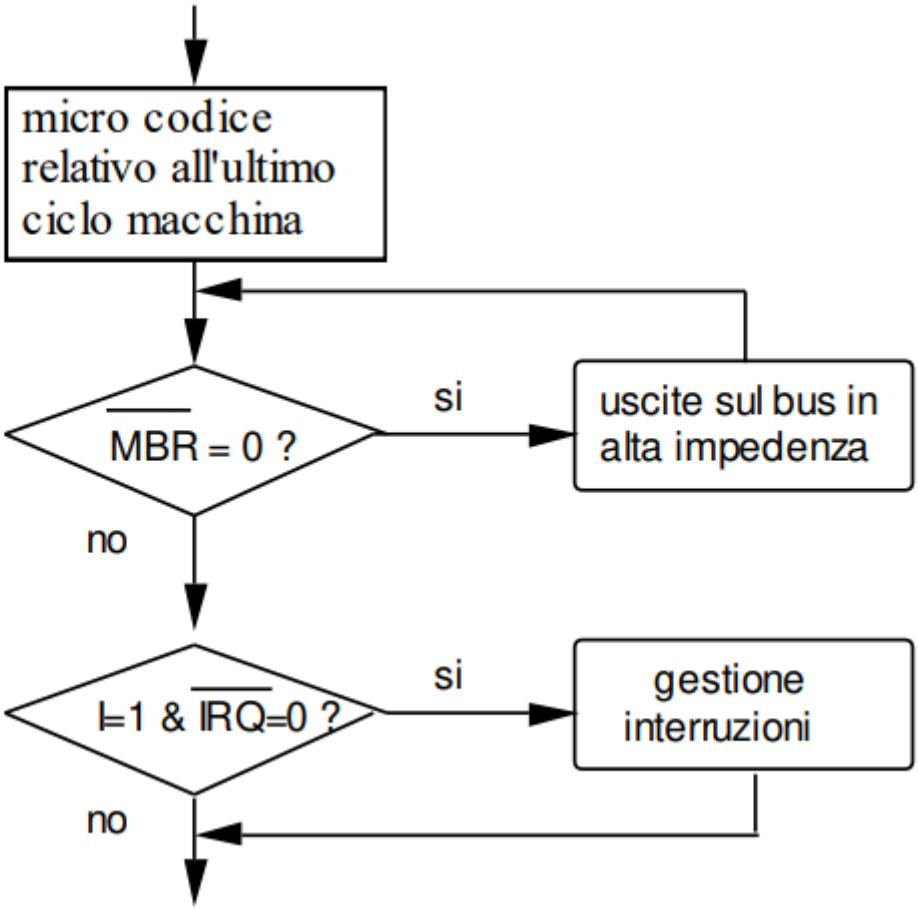
Descrivere le possibili modalita' di acquisizione di un file dati da una periferica e sua memorizzazione in memoria da parte di uno Z64 e quando e' conveniente utilizzare un DMAC

Il DMAC e' un'architettura costruita appositamente per rendere piu' veloce il trasferimento di dati tra CPU e Memoria. Questo emula gli stessi segnali che il processore invia alla memoria e puo' essere posizionato tra processore e periferiche, oppure puo' utilizzare lo stesso bus che usa il processore verso la memoria. Per evitare la concorrenza sull'utilizzazione del bus tra CPU e DMAC si utilizzano dei segnali chiamati MBR (Memory Bus Request) e MBG (Memory Bus Grant). Nel momento in cui il DMAC invia un MBR il processore mette in alta impedenza le uscite del processore verso i bus e rilascia il bus. Il DMAC utilizza il bus e al termine avverte il processore del termine dell'utilizzo.

Esiste la modalita' BURST e la modalita' BUS STEALING.

Nella prima durante l'utilizzo del bus da parte del DMAC il processore sara' inutilizzato durante tutto il trasferimento dei dati.

Nella seconda modalita' invece il DMAC richiede e rilascia il bus per ogni singolo dato, di modo che il processore possa continuare a lavorare mentre il DMAC trasferisce i dati



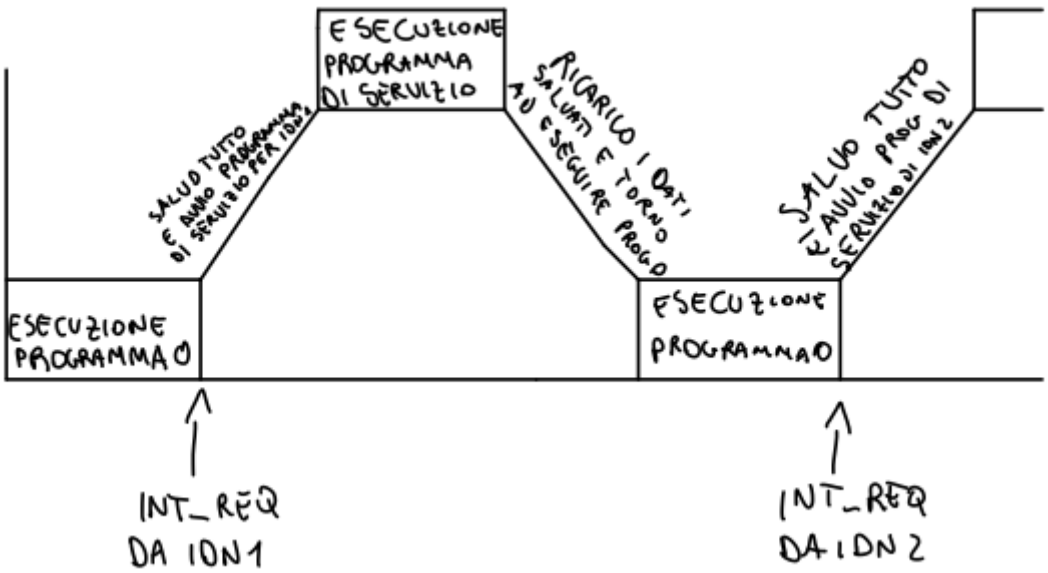
Come il processore gestisce le interrupt

Descrivere come il processore gestisce le interruzioni facendo vedere come viene modificato il ciclo istruzione

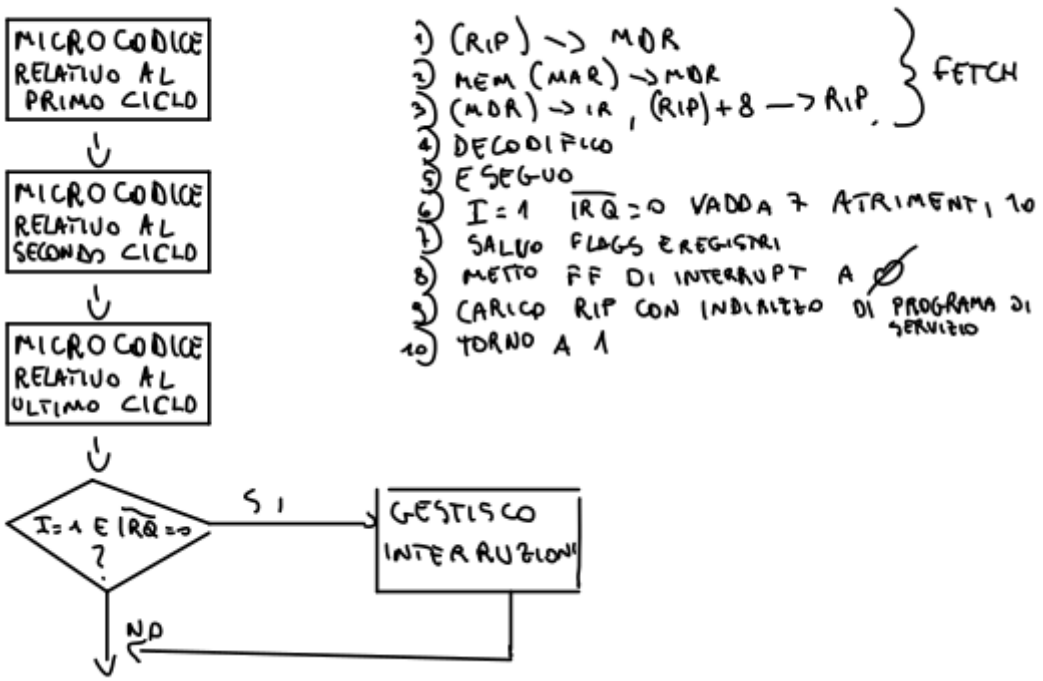
Nel momento in cui il processore riceve un interruzione vengono eseguite 4 fasi principali:

1. Salva lo stato del processo in esecuzione, ovvero contenuto dei registri, locazioni di memoria utilizzate , stack , flags...
2. Identifica il programma di servizio relativo all'identificativo (IDN) che ha mandato l'interrupt
3. Esegue il programma di servizio
4. Ricarica tutte le informazioni salvate nel punto 1 e continua l'esecuzione delle attivita' lasciate in sospeso.

Il processore puo' decidere quando essere interrotto o meno utilizzando un flip-flop. Questo flip-flop puo' essere manipolato tramite le istruzioni STI e CLI.



Come viene modificato il SCO di Z64:



Cache completamente associativa vs cache set-associativa a 4 vie

Cache completamente associativa

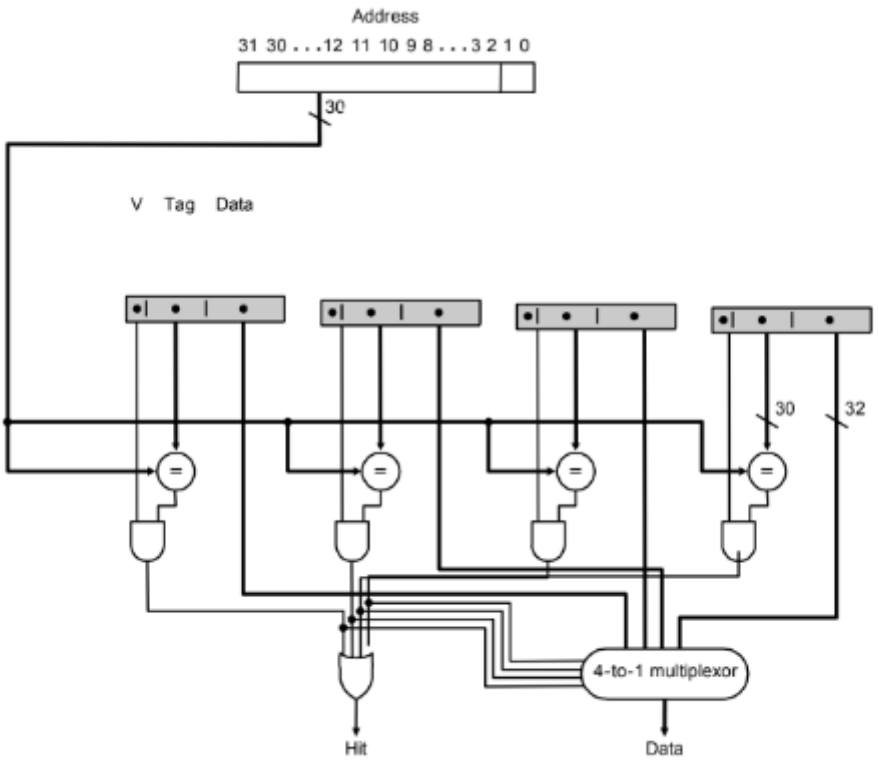
Il contenuto di un blocco e' identificato tramite un indirizzo completo di memoria e il suo tag e' costituito dall'indirizzo completo della parola. Inoltre ogni blocco di memoria puo' essere mappato in qualsiasi linea di cache. L'accesso in questo caso e' indipendente dall'indice di cache

I vantaggi sono :

- I tag hanno tutti i bit dell'indirizzo
- Non e' ordinata, ogni blocco puo' essere mappato ovunque
- Non ci sono conflict miss, ma solo capacity miss per capacita' insufficiente della cache

Gli svantaggi sono:

- Hardware molto complesso
- Realizzabile solo con piccole quantita' di blocchi



Cache set-associativa a 4 vie

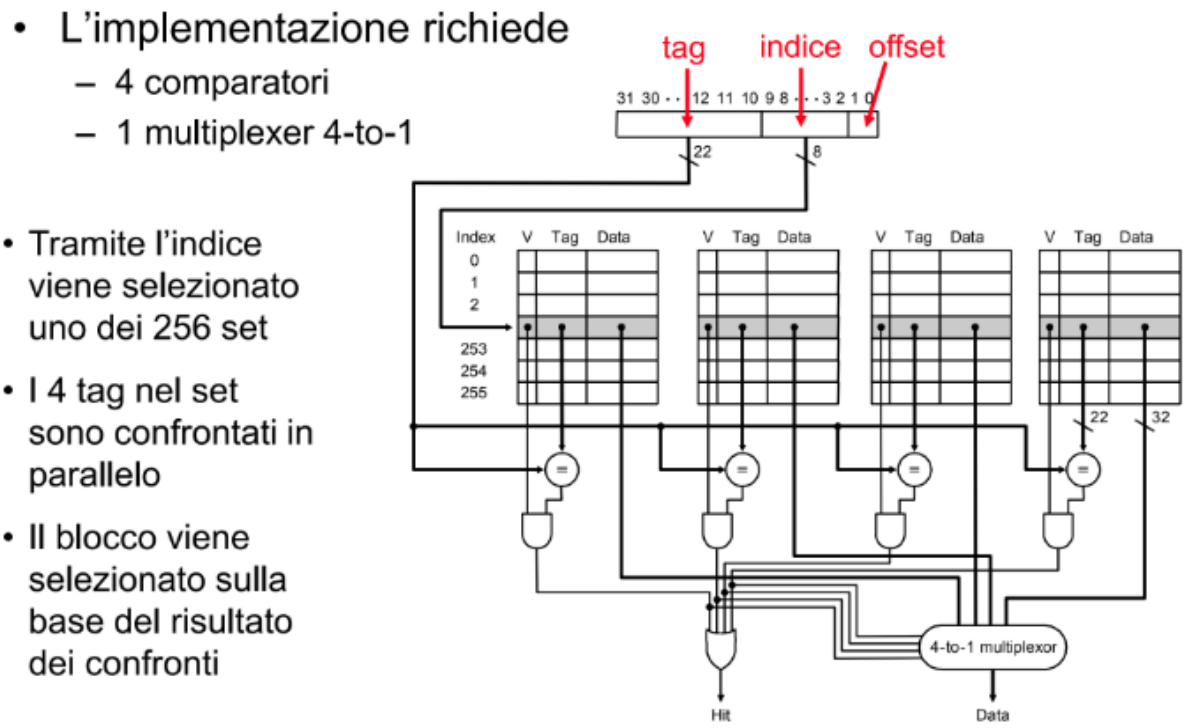
E' un compromesso tra soluzione ad indirizzamento diretto ed e' completamente associativa. La cache e' organizzata come insieme di set ognuno dei quali contiene 4 blocchi.

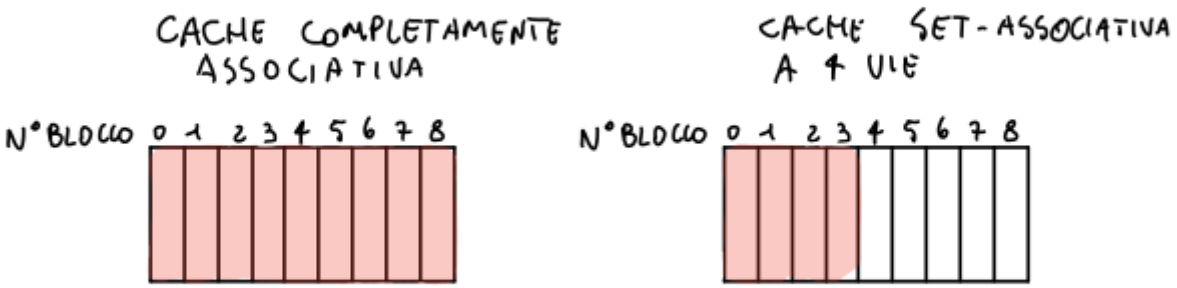
Vantaggi:

- Implementazione facile
- Richiede poco spazio
- E' molto piu' veloce

Gli svantaggi sono:

- Non e' efficiente per quanto riguarda la politica di sostituzione.

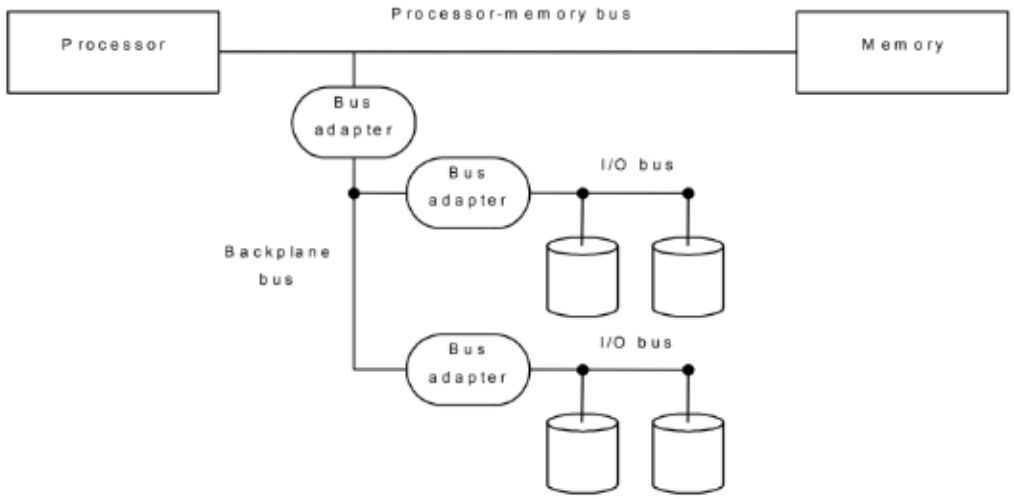




Organizzazione dei bus di connessione tra processore, memoria e dispositivi I//O

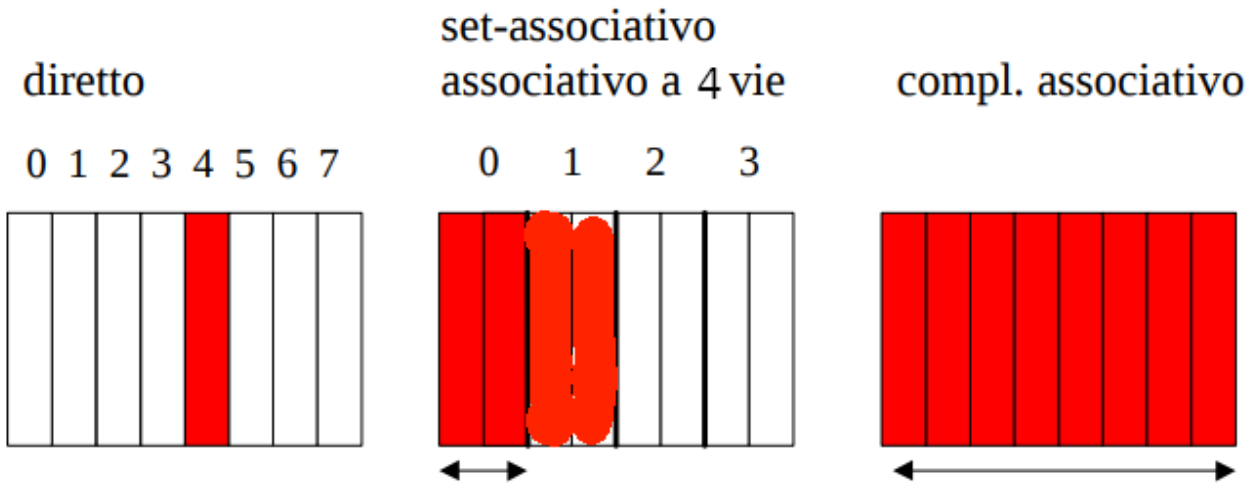
Solitamente l'organizzazione dei bus e' composta nel seguente modo:

- Bus che collega processore e memoria, con lunghezza ridotta per favorire la velocita' ed e' progettato per avere la maggior banda di trasferimento possibile
- Bus che collega input e output, che ha lunghezza solitamente maggiore e puo' avere tanti dispositivi I/O connessi. Esempi sono USB, Firewire etc..
- Bus backplane che fa da tramite tra il bus di I/O e il bus processore - memoria



Cache associative e a corrispondenza diretta

Illustrare con un disegno le cache associative e a corrispondenza diretta evidenziando i vantaggi di ognuna delle due.



Cache a indirizzamento diretto

Blocchi di grande dimensione comportano il miss penalty maggiore, quindi ed e' necessario un tempo di trasferimento del blocco maggiore

La sostituzione della cache in una linea di cache gia' occupata comporta l'eliminazione del contenuto precedente e si rimpiazza con il nuovo blocco. Questa sostituzione non tiene conto della localita' temporale quindi lo stesso blocco potrebbe essere stato usato di recente (fenomeno di thrashing)

Vantaggi:

1. Facile da implementare
2. Dimensione ridotta
3. veloce

Svantaggi:

1. Non efficiente per la sostituzione

Cache associativa

Ogni blocco puo' essere mappato in qualsiasi linea di cache, evitando conflict miss e generando esclusivamente capacity miss. Il contenuto di un blocco in cache e' identificato dall'indirizzo completo in memoria.

Vantaggi:

1. Diminuzione del miss rate

Svantaggi:

1. Costo maggiore
2. Incremento dell'hit time

Possibili modalita' di acquisizione di un file dati da periferica e memorizzazione in memoria.

Descrivere le possibili modalita' di acquisizione di un file dati da una periferica e la sua memorizzazione da parte di uno Z64. Spiegare quando e' conveniente utilizzare un DMAC

I possibili tipi di interazione tra processore e periferica si classificano in:

- Interazioni previste dai programmi che vengono eseguiti nella cpu, chiamate I/O programmato

  Queste possono essere in busy waiting (firmware / software) e polling
- Su richiesta esterna

  Possono essere gestite tramite interruzioni
- Gestite da processori dedicati

  Gestite dal DMAC

  #### Spiegare qundo e' meglio utilizzare un DMAC

  Nel caso in cui i dati richiesti hanno dimensione elevata e' preferibile utilizzare il DMAC. Il DMAC ha il compito di gestire i dati presenti nel bus permettendo a periferiche piu' veloci di continuare a lavorare alla loro velocita' massima. Evita inoltre una grande quantita' di interrupt alla CPU, permettendo di svolgere con piu' facilita' le operazioni.

Busy waiting software per input

1. Il processore avverte il dispositivo che vuole un dato resettando il FF di status

2. Il processore verifica che il dispositivo abbia prodotto il dato verificando il FF di status
3. Se il FF e' zero torno al punto 2
4. Se FF e' uno esegue un'istruzione di input

Busy waiting software per output

1. Il processore esegue una output e trasferisce il contenuto nel registro di interfaccia del dispositivo
2. Il processore avverte il dispositivo che vuole un nuovo dato resettando il FF di status
3. Il processore verifica che abbia prodotto il dato verificando il FF di status
4. Se e' zero torno a al punto 3
5. Se e' uno esce dall'attesa e continua le sue attivita'

Polling

Il processore interroga tutte le periferiche una alla volta eseguendo la routine della prima che risponde

Nel caso in cui una periferica risponde esegue:

1. Salva lo stato del processo in esecuzione
2. identifica la routine da eseguire in base all'IDN
3. Esegue la routine di servizio
4. Riprende le attivita'

In questo caso il device deve avere al suo interno un FF per l'interrupt request e un registro dove e' contenuto il suo identificativo (IDN).

Interazione tra processore con memoria e periferiche

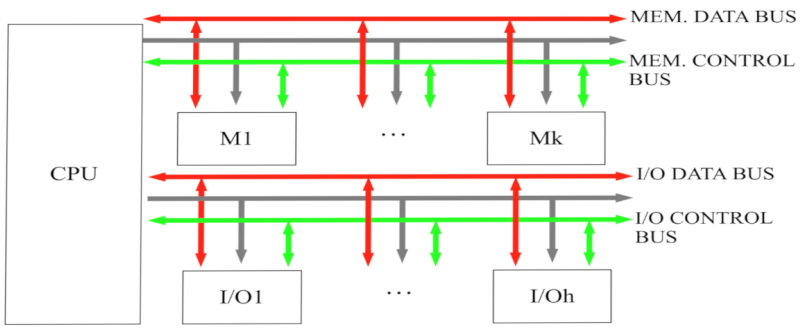
Descrivere le differenze di come il processore gestisce le interazioni con le periferiche nel caso di architettura con un solo bus e con due bus

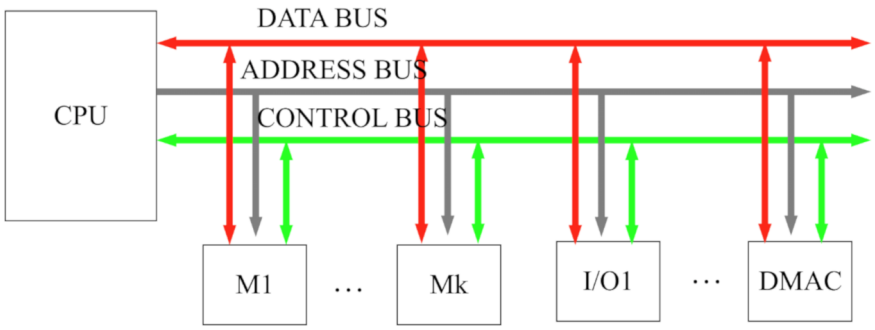
L'organizzazione dei bus normalmente puo' avvenire in 2 modi principali:

1. Singolo bus

   Ovvero un unico bus dove memoria e I/O condividono lo spazio di indirizzamento

2. Bus dedicati Ovvero bus dedicato alla memoria e bus dedicato all'I/O





Bus sincrono:

Il bus sincrono e' un bus che ha una particolare temporizzazione, ovvero e' scandito dal clock.

Per ogni operazione di R/W sono necessari piu' ciclo di clock

i Vantaggi sono:

1. Molto veloce
2. Non richiede molta logica perche' ogni device e' sincronizzato

Gli svantaggi invece:

1. Ogni device deve essere sincronizzato agli altri
2. Non puo' essere troppo grande per evitare il problema del clock skew, clock skew = differenza tra l'istante atteso di clock ed il clock effettivo

   Normalmente utilizzato per l'interazione tra CPU - Memoria, perche' puo' avere lunghezza ridotta e pochi elementi

Bus asincrono

Non e' necessario il clock e funziona tramite un protocollo di handshaking.

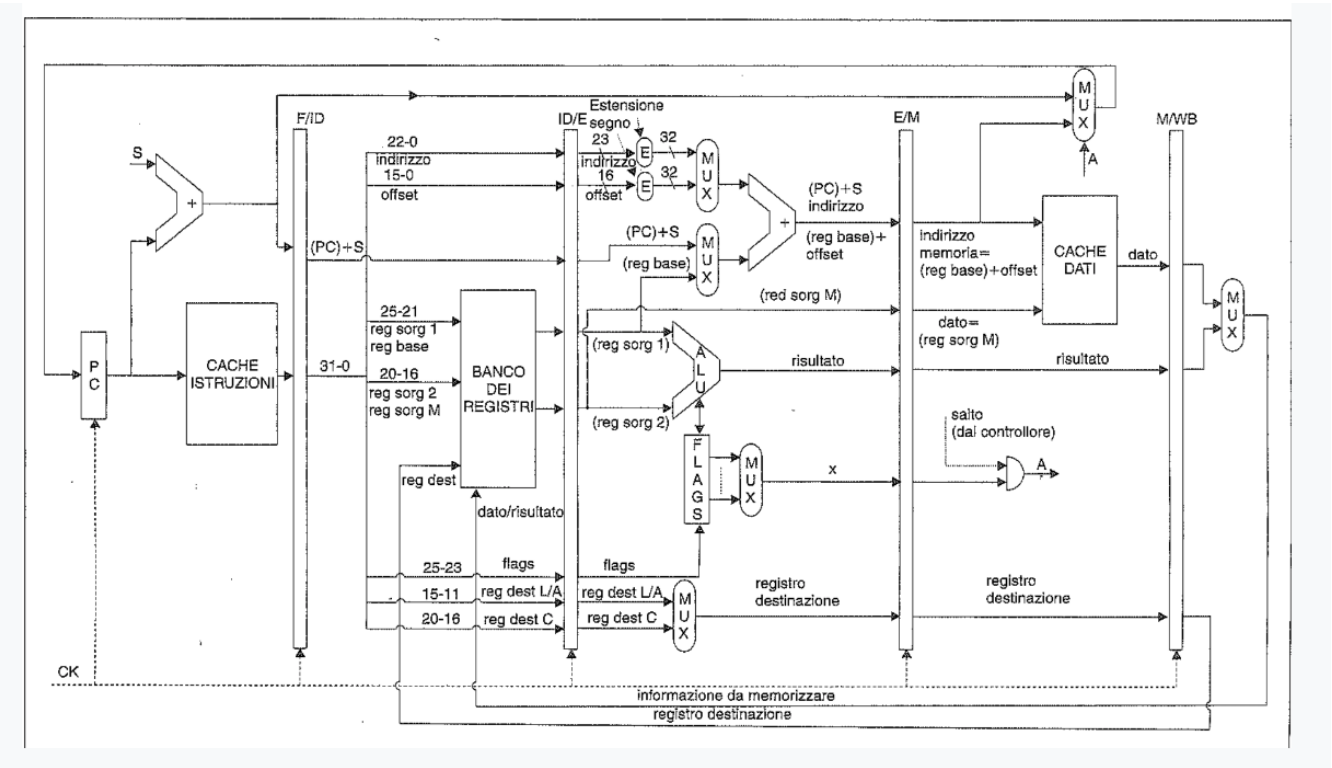
I Vantaggi sono:

1. Puo' essere piu' lungo
2. Il tempo di ogni operazione e' legato solamente alla velocita' del devices

Gli svantaggi:

1. Piu' complessa la gestione
2. Piu' lento
   Normalmente vengono utilizzati per l' I/O.

Disegna l'architettura PIPELINE



PROGETTO

Programmare DMAC per output

Si usa **outsX** inizializzando i seguenti registri: **X puo'essere B W L Q

- %RCX con quantita' di blocchi da dare al DMAC
- %RSI con indirizzo sorgente
- %DX con l'indirizzo del device a cui inviare i blocchi
- dopo scrivo l'istruzione cld
  infine outsX
  Esempio: Scrivi 10 Byte su un device tramite DMAC

```
1 movq $10, %rcx
2 movq $dest, %rsi
3 movq $dev_mem, %dx
4 cld
5 outsb
```

Programmare DMAC per Input

Si usa **insX** inizializzando i seguenti registri: **X puo'essere B W L Q

- %RCX con quantita' di blocchi da prendere dal DMAC
- %RDI con indirizzo di destinazione
- %DX con l'indirizzo del device da cui prendere i blocchi
- dopo scrivo l'istruzione cld

  infine insX

  Esempio: Prendi 10 Byte da un device tramite DMAC

```
1 movq $10, %rcx
2 movq $dest, %rdi
3 movq $dev_mem, %dx
4 cld
5 insb
```

put