Architettura degli Elaboratori

Esempio Primo Compitino 2011

ESERCIZIO 1:

Fra le informazioni riportate di seguito dire quale non è comunicata dalla CPU al di itivo DMA

a) se l'operazione è di lettura

b) se l'operazione è di scrittura

c) quantità di dati da trasferire

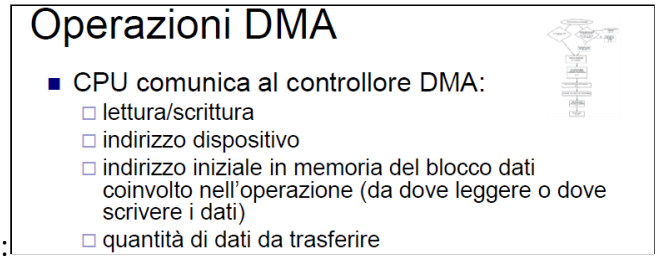
d) codice di interruzione

e) indirizzo iniziale in memoria del blocco dati coinvolto nell'operazione

f) tutte le informazioni di sopra sono comunicate;

Soluzione

slide:



quindi risposta D

ESERCIZIO 2:

Si consideri una cache set-associativa a 4 vie (4-way) da 8MB, con linee da 2KB. La cache è inserita in una gerarchia di memoria insieme ad una memoria centrale suddivisa in 2^{20} blocchi. Assumendo un indirizzamento al singolo byte, il formato degli indirizzi della memoria centrale è:

a) Etichetta = 10 bit; Set = 10 bit; Parola = 11;

b) Etichetta = 10 bit; Set = 11 bit; Parola = 10;

c) Etichetta = 9 bit; Set = 11 bit; Parola = 10;

d) Etichetta = 11 bit; Set = 9 bit; Parola = 11;

e) nessuna delle risposte precedenti è corretta;

Soluzione

Dati

- n° vie = 4 =2^2
- Dimensione cache = 8MB = $2^3*2^{20}=2^{23}$

- Dimensione linea = 2KB =2^1*2^{10}=2^{11}
- (ricordo che dimensione linea di cache = dimensione blocco in memoria = 2^{11})
- n° blocchi in memoria = 2^{20}
- Indirizzamento al singolo byte, cioè ogni parola in un blocco occupa 1 byte

Soluzione
Dimensione memoria = n° blocchi in memoria * Dimensione blocco = 2^{20}*2^{11} = 2^{31} ==> Indirizzo = 31 bit
Essendo ogni linea di 2^{11} byte, ed essendo l'indirizzamento al singolo byte ==> parola = 11 bit
n° linee = Dimensione cache/Dimensione linea = $2^{23}/2^{11}$ = 2^{13}
n° insiemi composti da 4 linee = n° linee/ n° vie = $2^{13}/2^2$ = 2^{10} ==> set = 10 bit
Indirizzo = tag + set + parola ==> tag = Indirizzo -(set + parola) = 31 - (11+10) = 10 bit

quindi risposta A

ESERCIZIO 3:

Si consideri una cache di 256MB con associazione a gruppi a 256 vie (256-way set associative) e dimensione di linea di 8KB. Supponendo che il campo tag sia di 12 bit, la dimensione massima (in byte) di memoria principale che la cache è in grado di gestire è:

a) 32768MB;

b) 4096MB;

c) 524288KB;

d) 256MB;

e) nessuna delle risposte precedenti è corretta;

Soluzione
Dati
- n° vie = 256 =2^8
- Dimensione cache = 256MB = 2^8*2^{20}=2^{28}
- Dimensione linea = 8KB =2^3*2^{10}=2^{13}
- tag = 12 bit
- Indirizzamento al singolo byte, cioè ogni parola in un blocco occupa 1 byte

Soluzione
Essendo ogni linea di 2^{13} byte, ed essendo l'indirizzamento al singolo byte ==> parola = 13 bit
n° linee = Dimensione cache/Dimensione linea = $2^{28}/2^{13}$ = 2^{15}
n° insiemi composti da 4 linee = n° linee/ n° vie = $2^{15}/2^8$ = 2^7 ==> set = 7 bit
Indirizzo = tag + set + parola = 12 + 7 + 13 = 32 bit
Dimensione memoria = $2^{Indirizzo}$ = 2^{32} byte = 2^{12}MB = 4096MB

quindi risposta B

ESERCIZIO 4:

Descrivere in dettaglio il ciclo di esecuzione con trattamento delle interruzioni

Soluzione **-** in collaborazione con Tony

Il ciclo di esecuzione è l'elaborazione di una singola istruzione di un programma. Il ciclo si suddivide in due fasi principali: Fetch,cioè il reperimento dell'istruzione, ed Execute, ed esecuzione dell'istruzione.

Il reperimento dell'istruzione avviene leggendo dal registro PC (program counter) l'indirizzo dell'istruzione da eseguire, ed il PC verrà poi incrementato (salvo diverse indicazioni) così da puntare all'indirizzo dell'istruzione successiva. Tale fase viene definita come **calcolo indirizzo istruzione** (CII).

L'indirizzo verrà copiato nel registro MAR, il quale comunicherà al bus l'indirizzo della cella di memoria da cui prelevare l'istruzione che verrà memorizzata nel registro MBR. Dal registro MBR l'istruzione verrà copiata nell'IR. Si conclude la fase di **fetch**.

Inizia poi l'**Execute** che si suddivide in molteplici sottofasi.

A questo punto avviene la **decodifica dell'istruzione** (DI), che consiste nel determinare l'operazione da eseguire, mediante il riconoscimento del codice operativo(opcode, solitamente di 4 bit), e l'operando da usare. I codici operativi possono identificare 4 macro tipologie di operazioni: trasferimento dati tra memoria e CPU, trasferimento dati tra periferiche e CPU, operazioni aritmetico logiche sui dati, controllo (come ad esempio gestire un salto nella sequenza di istruzioni da eseguire).

Segue la fase di **calcolo indirizzo operando** (CIO), che consiste nel determinare l'indirizzo dell'operando e individua se esso è riferito alla memoria o ad una periferica.

Segue poi la **lettura dell'operando** (LO). Nel caso di più operandi le fasi CIO e LO vengono ripetute tante volte quanti sono gli operandi.

Avendo a disposizione gli operandi, viene eseguita l'**operazione sui dati** (OD) indicata precedentemente dall'opcode. Si ripete poi la fase CIO per individuare l'indirizzo dove scrivere il risultato dell'operazione. Segua a ciò la **memorizzazione dell'operando** (MO).

Prima di proseguire con la successiva istruzione la CPU effettua il **controllo d'interrupt** in cui verifica se ci sono interruzioni pendenti, indicati da segnali di interrupt. Nel caso in cui non vi siano, procede rieseguendo il ciclo regolarmente. Nel caso in cui vi sia almeno un'**interruzione** pendente, viene sospesa l'esecuzione del programma corrente, viene salvato il contesto, il PC viene impostato all'indirizzo di partenza del programma di gestione dell'interruzione e viene quindi eseguito il programma di gestione, il quale determina la natura dell'interrupt ed esegue le azioni necessarie. Quando il programma di gestione termina, viene ricaricato il contesto e ripesa l'esecuzione del programma dal punto di interruzione.

- In aggiunta -

Le **interruzioni** sono un meccanismo che consente ad altri moduli di interrompere la normale sequenza di esecuzione del processore al fine di migliorare l'efficienza dell'elaborazione. Le interruzioni si dividono in 4 classi: programma, timer, I/O e guasto.

Nel caso di **interruzioni multiple** vi due approcci di gestione: disabilitare le interruzioni, cioè durante l'esecuzione di un interrupt vengono ignorate altre possibili interruzioni, che verranno al termine gestite in modo sequenziale; definire delle priorità, cioè ad ogni interruzione viene assegnata una priorità ed interruzioni a bassa priorità potranno essere interrotte da interruzioni con priorità più alta. Questo tipo di gestione può portare all'esecuzione annidata di interruzioni.

ESERCIZIO 5:

Spiegare in dettaglio le differenze fra un modulo di memoria DRAM ed un modulo di memoria SRAM. Discuterne vantaggi e svantaggi.

Soluzione

- Premessa-

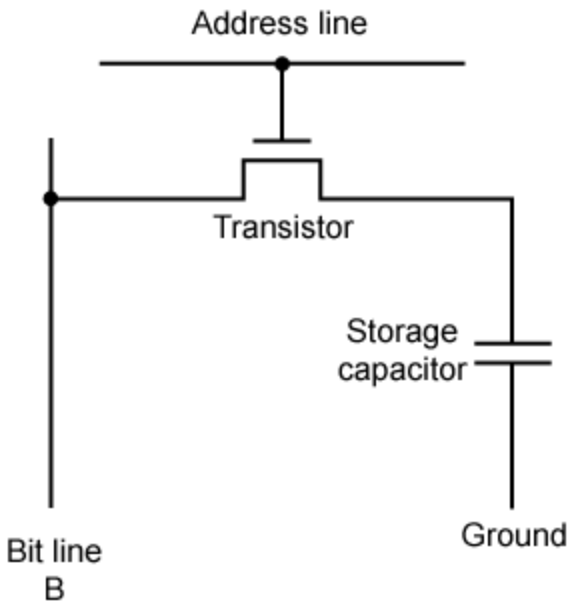
La RAM è un tipo di memoria a semiconduttore. E' una memoria volatile, ad accesso casuale, in cui è permessa sia la scrittura che la lettura e viene utilizzata per la memorizzazione dei dati.
Vi sono tue tipologie di RAM: la RAM Dinamica D-RAM e la RAM Statica.

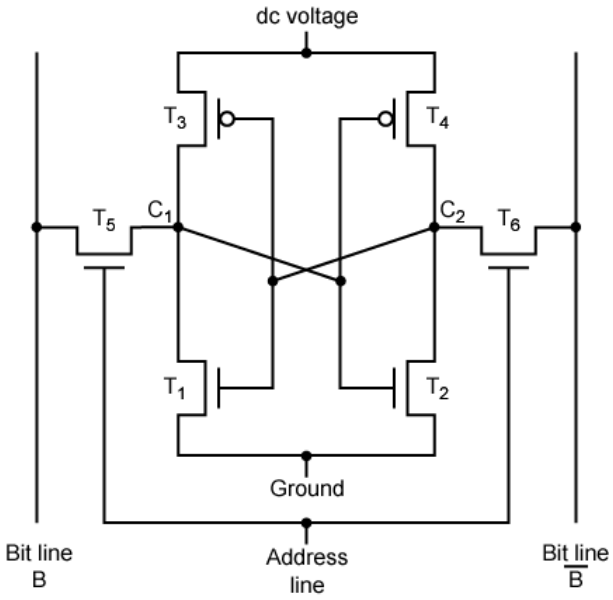
- -

Nelle **D-RAM** i bit vengono memorizzati sottoforma di carice in condensatori. Con il tempo vi è però un decadimento di queste cariche e sono quindi necessari degli appositi circuiti di refresh. Il refresh è inoltre necessario dopo la lettura di un dato, in quanto il condensatore per essere letto deve'essere scaricato e quindi ricaricato. Il bit viene letto a 1 se nel condensatore vi è una carica sufficientemente alta, confrontata con un segnale di riferimento, mentre viene messo a 0 se la carica è sotto la soglia di riferimento. Questa tipologia di RAM opera quindi in modo analogico.
Questa tipologia di RAM e meno complessa(1 transistor ed 1 codensatore), meno costosa, più capiente e con una densità d'informazione maggiore rispetto alla S-RAM, ma rispetto a questa risulta essere molto più lenta.
Viene quindi utilizzata per la memoria centrale.
Nelle **S-RAM** i dati vengono memorizzati attraverso porte logiche(filp-folp), composte di soli transistor, che quindi, differentemente dalle D-RAM, non comportano un decadimento della carica. La lettura inoltre non comporta la cancellazione dello stato. Non quindi necessari i refresh ed i relativi circuiti. In questo tipo di RAM ogni cella può trovarsi in due diversi stati, in cui la carica è localizzata in zone diverse. In base a dove è localizzata la carica verrà letto 1 o 0. Questa tipologia di RAM opera quindi in modo digitale.
Questa tipologia di RAM e però più complessa(6 transistor, nessun condensatore), più costosa e con minor densità d'informazione, in quanto rappresentare un bit occupa fisicamente più spazio, rispetto alla D-RAM. Bisogna precisare che la mancanza dei circuiti di refresh diminuisce leggermente la complessità dell'hardware. Il grosso punto a favore delle S-RAM è la velocità, in fatti sono molto più veloci delle D-RAM e per questo vengono utilizzate per la cache.

La maggiore complessità hardware delle S-RAM può essere facilmente visualizzata confrontando i due circuiti che descrivono le due tipologie:

D-RAM	S-RAM





ESERCIZIO 6:

Nel contesto di una gerarchia di memoria, spiegare come i "miss" possono essere categorizzati in diversi tipi e dire quali strategie, per ogni tipo, si possono adottare per tentare di diminuirne il numero. Discutere criticamente tali strategie.

Soluzione

Nel contesto di una gerarchia di memoria, i miss possono essere categorizzati in tre tipi:

- miss di primo accesso: sono invitabili e non riducibili, in quanto ogni blocco dev'essere copiato almeno una volta da memoria centrale a cache
- miss per capacità insufficiente: avvengono in quanto la cache non può contenere tutti i blocchi necessari all'esecuzione del programma
- miss per conflitto: avvengono con associazione diretta o a gruppi in quanto più blocchi possono essere mappati in uno stesso gruppo. Tale raggruppamento può portare all'eliminazione di un blocco nel gruppo che viene poi nuovamente richiesto, causando un miss di questo tipo.

Le strategie che si possono adottare per diminuire il numero di miss sono:

- aumentare la quantità di cache, così da aumentare la quantità di dati immagazzinabili in essa ==> Miss per capacità
- aumentare l'associatività, cioè il numero di linee per gruppo in cache. Tale tecnica però causa un incremento del tempo richiesto per localizzare il corretto gruppo in cui è presente l'istruzione o il dato cercato, e vi è inoltre la regola del 2:1, cioè una cache con N blocchi con associazione diretta ha quasi la stessa probabilità di miss di una cache con dimensione N/2 con associazione a 2 vie. ==> Miss per conflitto
- aumentare la dimensione di blocco, così da utilizzare la località spaziale
- separare la cache in cache istruzioni e cache dati, così da rendere indipendente la scrittura/lettura dei dati dalla singola lettura delle istruzioni
- utilizzare cache multilivello, cioè utilizzare una gerarchia di cache. Vi sarà un cache piccola, molto veloce e sullo stesso cip della CPU, detta cache on-chip, che permetterà ala CPU di accedere ai dati a "tempo 0" (cache L1). Vi è poi un'altra cache(L2), più grande e leggermente più lenta, associare alla cache L1. Ve ne può anche essere una terza(L3), ancor

più grande e più lenta, associata alla L2. Le cache sono connesse tra di loro in modo indipendente dal bus di sistema, così da non pesare su di esso.
Nonostante l'aumento dei dispositivi tra CPU e memoria centrale, vi è un aumento prestazionale rispetto alla connessione diretta della L1 alle memoria centrale, in quanto, grazie alla localizzazione spaziale, vi sarà un alto numero di hit dentro i dati nella L3, la quale, in caso di miss potrà inviare i dati sino alla L1 in un tempo estremamente ridotto rispetto a quello che farebbe la memoria centrale.

- ottimizzare gli accessi alla cache mediante un compilatore che sia in grado di farlo. Tale compilatore deve essere in grado di: utilizzare in modo ottimale l'architettura della cache, ad esempio se multilivello; posizionamento accurato delle procedure ripetitive; incrementare la località spaziale mediante la fusione di vettori in strutture ottimali; incrementare la località spaziale mediante l'ottimizzazione di iterazioni annidate.

ESERCIZIO 7

Discutere le ragioni per cui è stato sviluppato il sistema RAID. Inoltre si descriva in dettaglio il livello 0, 2 e 4 del RAID.

Soluzione

RAID è l'acronimo di Redudant Array Indipendent (o Inexpensive) Disks. Il sistema RAID è stato sviluppato principalmente per tre ragioni, che non necessariamente devono coesistere:

- le altre componenti erano molto più veloci del disco, quindi si è pensato di aumentare le prestazioni premettendo un accesso parallelo a più dischi
- avere dei dati ridondanti, quindi maggiormente preservati da possibili malfunzionamenti hardware
- avere un maggior spazio di memorizzazione in modo economico, cioè senza dover spendere tempo e denaro per lo sviluppo di dispositivi più capienti

- Premessa-

Il sistema RAID permette al sistema operativo di vedere un insieme di dischi fisici come un singolo dispositivo logico ed i dati vengono distribuiti sui vari dispositivi fisici. Il RAID è suddiviso in 7 livelli (dallo 0 al 6) non gerarchici che definisco diverse scelte architetturali.

--

RAID0

in questo livello di RAID l'obbiettivo è il solo aumento prestazionale . I dati infatti non sono ridondanti e la rottura di un disco porta alla perdita integrale dei dati. I dati sono distribuiti su tutti i dischi sottoforma di stip, e vengono allocati seguendo una strategia Round Robin.
I dischi risultano essere indipendenti, e questo permette una velocità totale del sistema accresciuta, infatti: la ricerca dei settori avviene in parallelo tra i vari dischi, un linseme di dati contigui sarà probabilmente distribuito su più dischi e quindi letto e scritto in parallelo, richieste multiple di dati difficilmente richiederanno dati sullo stesso disco, quindi i dischi dove sono i dati possono rispondere alla richiesta e non vi è conflitto di risorse.

RAID2

questo RAID permette una forte ridondanza dei dati. Viene adottata le tecnica dell'accesso parallelo, cioè i dischi sono sincronizzati in modo che la testina di ciascun disco si trovi nello stessa posizione. Tale imposizione fa perdere l'indipendenza dei dischi, ed oltretutto è un obbiettivo difficilmente raggiungibile in pratica.

L'unità d'informazione utilizzata è molto piccola. Vengono poi calcolati dei codici di correzione(es: Hamming) a livello molto basso, cioè tra i bit corrispondenti sui vari dischi, e posizionati in un altro disco in posizione corrispondente. Vi sono quindi dei dischi adibiti alla memorizzazione di questi codici.
Questo RAID permette quindi un'elevata ridondanza dei dati, ma è una soluzione costosa e difficilmente realizzabile e per questo non commercializzata.

RAID4
questo RAID permette la ridondanza dei dati mediante il calcolo della parità. Ogni disco è indipendente, quindi richieste separate possono essere soddisfatte in parallelo. Viene scelta una unità d'informazione ampia, e tra le unità d'informazione dei vari dischi, viene calcolata la parità bit a bit. Vi è un disco ah hoc, detto disco di parità, dove vengono memorizzate tutte le informazioni di parità.
Con tale configurazione ogni scrittura su un qualsiasi disco, richiede la scrittura anche sul disco di parità. Questo causa un rallentamento globale del sistema in cui il disco di parità è il collo di bottiglia. Questa configurazione non è stata commercializzata.

ESERCIZIO 8

Sia dato un disco rigido da 128GB con 2 piatti (4 facce), 524288 tracce per faccia e 1024 settori per traccia. La velocità di rotazione del disco è di 10000 rpm, mentre il tempo medio di posizionamento della testina è di 1,4 ms. Si calcoli il tempo totale medio (in millisecondi, e senza contare l'attesa che il dispositivo ed uno dei suoi canali sia libero) che occorrono per trasferire 32KB, assumendo che i byte da trasferire siano memorizzati in settori contigui di una singola traccia.

Si descrivano dettagliatamente tutti i passi per ottenere la soluzione.

Soluzione
-Disco Fisso-
Dati:

- Capacità totale disco = 128GB = $2^7*2^{30}=2^{37}$
- n° piatti = 2 ==> n° facce = 4 =2^2
- n° tracce/faccia = 524288 = $2^9*2^{20} = 2^{29}$
- r = velocità di rotazione = 10000 RPM
- T_S = Tempo di posizionamento = seek time = 1.4 ms
- b = byte da trasferire = 32KB = $2^5*2^{10} = 2^{15}$

Soluzione:
T_L= Tempo latenza in millisecondi = (1000/(velocità di rotazione in RPS))*1/2 = 1000/(10000 RPM /60) * 1/2 = 3 ms
Capacità di una faccia = Capacità totale disco/n° facce = $2^{37}/2^2 = 2^{35}$
N = byte/taccia = Capacità di una faccia/(n° tracce/faccia) = $2^{35}/2^{29} = 2^{16}$
T_T = Tempo di trasferimento in millisecondi = b/(rN) * 1000 =(r deve essere in RPS) = 2^{15}/((10000 RPM/60) *2^{16}) *1000 = 3 ms
T = tempo toltale medio = $T_S + T_L + T_T$ = 1.4 +3 +3 = 7.4 ms

NON HO LA PRESUZIONE DI DIRE CHE QUESTO E' IL CORRETTO MODO DI SVOLGERE IL COMPITINO.
LO CARICO PERCHE' POTREBBE ESSERE UTILE AVERE UN CONFRONTO O MAGARI UNO SPUNTO PER RISOLVERE ALCUNI ESERCIZI.
SE VEDETE ERRORI AGGIORNATE PURE IL FILE CON UNA NUOVA VERSIONE CORRETTA E MIGLIORATA.
ABBIATE SPIRITO CRITICO =)

SALUTI

Caesar