COMPITO ARCH ELAB 19-09-2011

Es1

La riduzione dei problemi di congestione di traffico del bus di sistema può essere ottenuta grazie all'introduzione di uno o più bus di espansione.Tale approccio tuttavia produce problemi in caso di:
a) utilizzo di uno o più dispositivi DMA
b) presenza di uno o più moduli di I/O
c) gestione da programma dell'I/O con tecnica memory-mapped
d) presenza di cache
e) presenza di dispositivi di memorizzazione secondaria a tecnologia ottica
f) nessuna delle precedenti

Es2
Si consideri una cache set associativa a 2 vie da 1 MB. La cache è inserita in una gerarchia di memoria insieme a una memoria centrale suddivisa in 2^19 blocchi e 1 GB. Il formato degli indirizzi della memoria centrale è
a) Etichetta= 10 bit Set= 10 bit Parola= 10 bit
b) Etichetta= 8 bit Set= 11bit Parola= 11 bit
c) Etichetta= 11 bit Set= 11 bit Parola= 8 bit
d) Etichetta= 11 bit Set= 8 bit Parola= 11 bit
e) nessuna delle precedenti

ES3
Si consideri una cache di 64 KB set-associativa a 4 vie in congiunzione con una memoria centrale di 256 MB, indirizzabile al byte. Le locazioni di memoria con indirizzi (in esadecimale) D507690 e F303635 hanno possibilità di essere caricate all'interno dello set di linee se la dimensione del blocco è di:
a) 128 B
b) 32 B
c) 256 B
d) 64 B
e) nessuna delle precedenti

ES4
Spiegare la differenza fra interruzioni multiple e interruzioni annidate, discutendo criticamente le differenti modalità di trattamento da loro richieste

ES5
Si descrivano le caratteristiche e le operazioni principali delle memorie a semiconduttore, considerando sia memorie RAM che ROM

ES6
Nel contesto di una gerarchia di memoria, spiegare i possibili modi di realizzazione del mapping dei blocchi, discutendo criticamente i vantaggi e gli svantaggi in ogni modo

ES7

Descrivere la gestione dell'I/O tramite DMA

Es8

Sia dato un disco rigido con le seguenti caratteristiche

Capacità	Numero piatti/facce	Tracce per faccia/settori per traccia	Velocità rotazione
32GB	4/8	32768/512	7200 rpm

Inoltre il tempo totale medio per accedere a 32KB memorizzati in settori contigui su uno stesso cilindro è di cica 9,927083 ms.

Si calcoli il tempo medio di posizionamento della testina, descrivendo dettagliatamente tutti i passi per la soluzione.

PARTE DUE

ES1

Quante volte la CPU deve accedere alla memoria quando preleva ed esegue

un'istruzione che ha 2 operandi, uno con modo di indirizzamento registro e uno

con modo di indirizzamento immediato?

a) 2

b) 3

c) 1

d) 4

e) nessuna delle precedenti

Ho 2 operandi:
 primo operando: indirizzamento da registro ==> Non accede alla memoria per avere il dato, ma accede ad un registro ==> nessun accesso alla memoria
 secondo operando: indirizzamento immediato ==> Non accede alla memoria per avere il dato in quanto tale dato è parte dell'istruzione ==> nessun accesso alla memoria
Globalmente non ho quindi nessun accesso, MA solo il fetch
Risposta C

ES2
Si consideri la rappresentazione di numeri a virgola mobile che utilizza 4 bit per il campo esponente e 11 bit per la mantissa. Il numero 2,03515625 viene rappresentato dalla sequenza di bit
a) 0100000000100100
b) 0000001001000001
c) 0000101001001000
d) 0000001001001000
e) nessuna delle precedenti

ES3
Si consideri la pipeline a 4 stadi: fetch (IF), decodifica (ID), elaborazione (EI), scrittura (WO), per cui:
· i salti incondizionati sono risolti (identificazione salto e calcolo indirizzo target) alla fine del secondo stadio (ID)
· i salti condizionati sono risolti (identificazione salto, calcolo indirizzo target e calcolo condizione) alla fine del terzo stadio (EI)
· il primo stadio (IF) è indipendente dagli altri
Inoltre si assuma che non ci siano altre istruzioni che possono mandare in stallo la pipeline e che non sia implementato alcun meccanismo di trattamento dei salti.
Sapendo che: il 13% delle istruzioni sono di salto condizionato
il 30% delle istruzioni sono di salto incondizionato
il 60% delle istruzioni di salto condizionato hanno la condizione soddisfatta (prese)
il fattore di velocizzazione della pipeline è di:
a) 3,230988
b) 3,508772
c) 2,356502
d) 2,747253
e) nessuna delle precedenti

- Istruzioni di salto incondizionato 30% → 0,30 x 1= 0,30
- Istruzioni di salto condizionato prese 60% di 13% = 0,13x0,6= 0,078
- Istruzioni di salto condizionato non prese= non cosidero il caso, non in quanto ho caricato istruzione corretta!

$$S_k = \frac{1}{1 + 0{,}30 + 0{,}078x2} 4 = 2{,}747253$$

Risposta D

Es4

Si spieghi in dettaglio la rappresentazione dei numeri reali secondo lo standard IEEE 754

Lo standard IEEE 754 è lo standard di rappresentazione dei numeri in virgola mobile. È possibile rappresentare i numeri in due formati differenti: a precisione singola con 32 bit e precisione doppia con 64 bit. La rappresentazione singola si compone di un bit per il segno, 8 bit per l'esponente e 23 bit per la mantissa. Il formato doppio a 64 bit si compone di 1 bit per il segno, 11 bit per l'esponente e 52 bit per mantissa.

Inoltre possiamo avere altri due formati (singolo e doppio) che ci permettono di avere bit aggiuntivi per l'esponente e per la matissa, usati per rappresentare risultati intermedi di operazioni.

Per il formato singolo, a 32 bit posso avere:

- esponente polarizzato, con numeri da 1 a 254, quindi esponente da -126 a +127.
- Per esprimere i numeri non nulli utilizzo è
  $\pm 1,ff...f \times 2^{e-127}$

Es5

Si descrivano nel dettaglio le modalità di indirizzamento con spiazzamento e a pila. In particolare, si confrontino criticamente i 2 metodi di indirizzamento e se ne discutano pregi e difetti

Lo spiazzamento è una combinazione di indirizzamento diretto e indirizzamento registro indiretto. Il campo indirizzo ha due sottocampi:

A è il valore base, l'indirizzamento diretto

R il registro che contiene l'indirizzo di un valore da sommare ad A per ottenere l'indirizzo effettivo (EA).

Abbiamo 3 tipi di indirizzamento con spiazzamento:

- Relativo: dove R è il program counter, quindi il mio indirizzo effettivo è

  EA= A+PC

- Registro-base: A contiene uno spiazzamento positivo, mentre R contiene il puntatore all'indirizzo base
- Indicizzazione: A è la base mentre R lo spiazzamento positivo da tale base. Questo tipo è utile se devo considerare un elenco di dati memorizzati uno dietro l'altro.

l'indirizzamento a pila è una particolare organizzazione della memoria dove si lavora sull'elemento che si trova in cima alla pila. Un particolare registro del processore, chiamato Stack Pointer, mantiene il puntatore sulla cima della pila.

Il vantaggio dello spiazzamento è senza dubbio la grande mole di dati che si possono indirizzare anche se gli accesi alla memoria possono rallentare il metodo. La pila non presenta accessi alla memoria, in quanto il riferimento è implicito ma non è possibile applicare tale metodo a tutto il sistema.

Es6

Nel contesto di una pipeline, descrivere la problematica della dipendenza dei dati e si discutano in dettaglio le tecniche viste a lezione per trattare il problema

La dipendenza dal controllo è un problema che si genera nelle pipeline, la quale può essere invalidata poiché si carica l'istruzione sbagliata. Ci sono due macro soluzioni a tale problema. La prima è l'introduzione di fasi non operative, in cui si aspetta che venga calcolato l'indirizzo dell'istruzione successiva. Questo metodo ha una pessima efficienza ma è facile da implementare.

La seconda consiste nell'individuazione delle istruzioni critiche in modo da anticiparne il fetch ed eseguirla in maniera ottimale. Tale metodo è efficiente ma molto complicato da implementare.

Altre soluzioni sono l'ottimizzazione del compilatore e il data forwarding un sistema che se il tipo di dipendenza lo permette, grazie ad appositi circuiti di ByPass e Mutex regolati da Unità di Controllo e Unità di Forward, è possibile trasferire i dati dall'output della ALU in ingresso alla ALU. Questa soluzione è estremamente funzionale nel caso di operazioni aritmetiche sequenziali ma è implementabile solo in certi tipi di macchine, come il MIPS.

Es7

Si metta a confronto criticamente il modo in cui una architettura RISC utilizza l'ampio banco di registri a sua disposizione rispetto alla gestione di una cache.

L'architettura RISC utilizza l'ampio banco di registri per conservare le variabili (prevalentemente locali) che hanno un'alta probabilità di essere utilizzate con maggior frequenza. Sotto questo punto di vista il banco dei registri assomiglia molto alla memoria cache, sebbene sia molto più veloce. Ci sono però alcune differenze.

1) Il banco registri contiene tutti gli scalari locali delle n-1 procedure più recentemente attivate. La cache contiene solo gli scalari usati di recente.
2) 2) per le variabili globali sono tutte contenute nel banco registri secondo le scelte del cpmpilatore, mentre la cache solo quelle usate di recente.
3) La cache importa blocchi di variabili, alcune delle quali potrebbero non essere utilizzate, mentre il banco dei registri solo le variabili realmente usate.
4) Il trasferimento di dati tra registri e memoria è determinato in base alla profondità dell'annidamento delle procedure mentre per la cache è determinato dall'algoritmo di sostituzione.
5) Per acceder a una variabile locale, in un banco di registri, viene utilizzato esclusivamente l'indirizzamento registro, molto veloce e semplice.
   Nella cache l'indirizzamento è molto più lento, in quanto la maggiorparte delle cache è set-associativo.

ES8

Sia data la seguente sequenza di istruzioni assembler, dove i dati immediati sono espressi in esadecimale:

```
LW $4, 10($2)
ADDI $8, $4, 3
SW $8, 150($0)
SUB $2, $0, $8
ADDI $2, $4, 4
SW $4, 208($2)
ADD $4, $4, $2
```

Si consideri la pipeline MIPS a 5 stadi vista a lezione con possibilità di data forwarding e con possibilità di scrittura e successiva lettura dei registri nello stesso ciclo di clock.

a) si individuino e discutano le dipendenze dovute ai dati

b) mostrare come evolve la pipeline durante l'esecuzione del codice

soluzione

LW	R4← MEM [10 + R2]
ADDI	R8 ← R4 + 3
SW	MEM [150 + R0] ← R8
SUB	R2 ← [R0 + R8]
ADDI	R2 ← [R4 + 4]
SW	MEM [208 + R2] ← R4
ADD	R4 ← [R4 + R2]

dipendenze e by-pass dovute ai dati

- Per EX_{addi} della seconda riga, mi serve il valore aggiornato di r4 da output di MEM_{lw}
- SW è di sola lettura, quindi non mi crea dipendenze
- R8 di SUB, ha bisogno del valore aggiornato ADDI
- ADDI (quinta riga), ho bisogno del valore aggiornato di r4 da output di MEM_{lw} in prima riga.
- Ultima riga, ho bisogno del valore R2 aggiornato da ADDI della quinta riga e di R4 aggiornato dalla prima.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
LW	IF	ID	EX	MEM	WB									
ADDI		IF	ID	ID	EX	MEM	WB							
SW			IF	IF	ID	EX	MEM	WB						
SUB					IF	ID	ID	EX	MEM	WB				
ADDI						IF	IF	ID	EX	MEM	WB			
SW								IF	ID	EX	MEM	WB		
ADD									IF	ID	EX	MEM	WB	

Frecce verticali sono by-pass, le altre dipendenze

Questa è la mia risoluzione dell'appello, alcune domande potrebbero mancare o essere incomplete o errate!
Lo carico affinchè possa essere d'aiuto a qualcuno e in caso vogliate correggere contattatemi via facebook con la modifica o fate voi!

Nicolò scapin