

# ARCHITETTURA ELABORATORI B:

SOLUZIONI ALLE  
DOMANDE DI TEORIA  
APPELLI 2007 – 2009



di  
Sebastiano Vascon

## Indice generale

19 Febbraio 2009.....	4
Il forwarding elimina stalli successivi a istruzioni aritmetiche .....	4
Write-back e Write-through.....	4
Parte Controllo della CPU Multiciclo [ DA FARE ].....	4
Cache Miss => Page fault [ DA FARE ].....	4
Miss Certi, per Capacità e per Conflitti.....	4
29 gennaio 2009.....	5
PC-Relative e Branch: Calcolo dell'indirizzo target.....	5
Benchmark.....	5
Cache Miss => Page Fault o TLB miss [ DA FARE ].....	6
Polling e Interrupt.....	6
Banda reale di un disco.....	6
9 settembre 2008.....	7
Parte Controllo della CPU Multiciclo.....	7
TBL e page table nella traduzione da add.virtuale a fisico [ DA FARE ].....	7
Write-back e Write-through.....	7
Prestazioni: Frequenza e MIPS.....	7
CPU Multiciclo e riuso delle risorse.....	7
10 luglio 2008.....	8
PC-Relative e Branch: Calcolo dell'indirizzo target.....	8
Benchmark.....	8
TLB Miss => Page Fault.....	8
Organizzazione di un disco ( CHS ).....	8
Dipendenza WAR (Write After Read) e RAW (Read After Write).....	8
24 giugno 2008.....	9
Caratteristiche CPU singolo ciclo, multi-ciclo e pipeline [ DA FARE ].....	9
I/O Interrupt Driver – Inefficienza e soluzioni.....	9
Cache Miss => Page Fault ? [ DA VERIFICARE ].....	9
CPU multi-ciclo e le operazioni di ADD e LW.....	9
Trattamento del Page Fault come un'eccezione del S.O.....	10
19 febbraio 2008.....	10
CPU singolo ciclo e addizionatori aggiuntivi per LW, ADD, BEQ [ DA FARE ].....	10
Località.....	11
TBL miss => Page Fault ? Cache miss => Page Fault ? [ DA FARE ].....	11
Pipeline, stalli e forwarding.....	11
Write-back e Write-through.....	11
29 gennaio 2008.....	11
PC-Relative [ DA FARE ].....	11
Parte controllo della CPU multiciclo.....	11
Memoria Virtuale [ DA FARE ].....	12
Località e dischi rigidi.....	12
I/O interrupt-driven e trasferimento continuo.....	12
12 settembre 2007.....	12
Polling VS Interrupt.....	12
CICLI per una LOAD con CPU Multiciclo.....	13
LW e memoria virtuale [ DA FARE ].....	13
Forwarding e collegamenti datapath [ DA FARE ].....	13
Località Spaziale e Fetch delle istruzioni.....	13

4 luglio 2007.....	14
Memoria Virtuale.....	14
Località e array.....	14
Forwarding ed eliminazione dipendenze RAW.....	14
Previsione dei salti nella pipeline [ DA FARE ].....	14
BEQ + EXT-SIGN + SHIT << 2 perchè ?.....	14
13 giugno 2007.....	15
PC-Relative, completezza e regolarità delle istruzioni [ DA FINIRE ].....	15
Cache miss => Page Fault ?.....	15
Polling e trasmissione continua.....	15
Architettura multiciclo, ALU e l'istruzione BEQ.....	15
Organizzazione di un H.D. ( CHS) e valori di overhead seek e rotation [ DA FINIRE ].....	15
14 febbraio 2007.....	16
Cache miss => Page Fault ? TLB miss => Page fault ?.....	16
Bus sincroni e asincroni.....	16
Caratteristiche CPU singolo ciclo, multiciclo e pipeline.....	16
Forwarding: Esempi di op. aritmetiche dove c'è RAW.....	16
Semantica istruzioni MIPS.....	17
24 gennaio 2007.....	18
Cache ad accesso diretto (parte 1):.....	18
Cache ad accesso diretto (parte 2):.....	18
Ruolo di TLB, Page Table e Cache [ DA CONTROLLARE ].....	18
DMA vs Interrupt.....	19
Predizione dei salti [ DA FARE ].....	19

**19 Febbraio 2009**

### ***Il forwarding elimina stalli successivi a istruzioni aritmetiche***

*Spiegare come, rispetto alla CPU pipeline a 5 stadi vista a lezione, il forwarding elimina possibili stalli successivi a istruzioni aritmetiche di tipo R, come ad esempio le add/sub. Spiegare con un esempio.*

**Risposta:**

Consideriamo due istruzioni consecutive [add \$s0 , \$t0 , \$t1] e [sub \$t2 , \$s0 , \$t3 ] nel diagramma di esecuzione notiamo che il nuovo valore del registro \$s0 lo troviamo dopo la fase di EXE della ADD e dovrebbe essere utilizzato immediatamente nell'operazione di SUB.

Eseguendo le operazioni senza il forwarding ci troveremo nella fase di EXE della SUB mentre saremo nella fase di MEM della ADD quindi il valore non sarebbe aggiornato con la somma di \$t0 e \$t1 di conseguenza il valore risultante dalla SUB sarebbe sbagliato.

Grazie al forwarding questo tipo di dipendenze viene individuato e al posto dell'inserimento delle NOP della Hazard Detect Unit possiamo recuperare il valore della ADD già nella fase di EXE (viene salvato in un Register File aggiuntivo) e passarlo alla SUB per una corretta esecuzione, le fasi della ADD che vanno dalla EXE alla fine dei 5 stadi non modificheranno il risultato dell'operazione.

### ***Write-back e Write-through***

*A cosa si riferiscono e in che cosa consistono le politiche write-back e write-through?*

**Risposta:**

Sono due politiche per la coerenza dei dati nei livelli utilizzate nei sistemi di caching.

La politica Write-back aggiorna solamente il valore della cache e non si preoccupa del valore contenuto in memoria a meno che questo non debba essere rimpiazzato

La politica Write-Through aggiorna sia il valore in cache sia il valore in memoria

### ***Parte Controllo della CPU Multiciclo [ DA FARE ]***

*In cosa si differenziano le parti controllo della CPU multiciclo e pipeline visti a lezione?*

**Risposta:**

### ***Cache Miss => Page fault [ DA FARE ]***

*Avendo una cache indirizzata con l'indirizzo fisico, si potrebbe verificare un page fault come conseguenza di un cache miss? (Motivare)*

**Risposta:**

### ***Miss Certi, per Capacità e per Conflitti***

*Rispetto alle gerarchie di memoria, abbiamo parlato di miss Certi, per Capacità e per Conflitti esemplificare rispetto ad un qualsiasi livello della gerarchia.*

**Risposta:**

Sono un modello di classificazione dei miss.

Miss Certi: Miss di partenza a freddo che si verifica quando il blocco deve essere portato nella cache la prima volta

Miss per Capacità: La cache non è in grado di contenere tutti I blocchi necessari all'esecuzione del programma

Miss per Conflitti: Più blocchi sono in conflitto (anche se la cache non è piena) per una posizione.

## 29 gennaio 2009

### **PC-Relative e Branch: Calcolo dell'indirizzo target**

*Considerando la modalità di indirizzamento PC-relative delle istruzioni di branch, se all'indirizzo di memoria 0x4000fb00 abbiamo l'istruzione beq \$5, \$6, 0xffff, qual sarebbe l'indirizzo target assoluto nel caso in cui il branch fosse taken? (Considerare che 0xffff è una costante in complemento a 2. Specificare procedimento e motivazioni)*

**Risposta:**

Dal testo ho che PC = 0x4000FB00

Il salto avviene a questo indirizzo:  $PC \leftarrow PC + 4 + \text{sign\_ext}(0xFFFFE) \ll 2$

Essendo il salto "PC relative" l'istruzione confronta \$5 e \$6 intanto il controllo ha settato il valore di PCSrc in modo da ricevere il valore della costante esteso di segno e shift  $\ll 2$  ora il dato per il salto viene sommato al valore del PC + 4 se il confronto andrà a buon fine (cioè sarà taken) la prossima istruzione che verrà eseguita sarà all'indirizzo  $PC + 4 + \text{sign\_ext}(0xFFFFE) \ll 2$

### **Benchmark**

*Illustrare la necessità di avere programmi di benchmark accuratamente selezionati per valutare e comparare le prestazioni dei sistemi. Esemplicare rispetto ai benchmark SPEC e al relativo indice di prestazioni, e comparare rispetto a misure come i MIPS.*

**Risposta:**

La necessità di utilizzare programmi di benchmark selezionati deriva dal bisogno di valutare le effettive prestazioni di una CPU e poter quindi decidere se un'acquisto è buono o meno relativamente alle necessità computazionali.

SPEC (System Performance Evaluation Cooperative) sono un insieme di programmi standard reali con input stabiliti sulla base di accordi tra aziende.

Cerca di limitare al minimo gli abusi e sono quindi considerati un indicatore significativo delle prestazioni.

Si dividono in due categorie SPECInt e SPECfp la prima dedicata alle prestazioni su calcoli in valori interi la seconda su valori in virgola mobile.

MIPS (Milioni di Istruzioni Per Secondo) sono una misura media di quanti milioni di operazioni una CPU riesce a computare in un secondo, non è un indicatore significativo in quanto è facilmente falsabile semplicemente eseguendo programmi con un'alto numero di istruzioni semplici.

## **Cache Miss => Page Fault o TLB miss [ DA FARE ]**

*Supporre che, come conseguenza di una lw o di un fetch di un'istruzione, si verichi un cache miss. Potrebbe verificarsi un page fault o un TLB miss? (Motivare)*

### **Risposta:**

Visto che si verifica un cache miss questo implica che l'istruzione non è in cache, bisogna quindi andare a cercarla nella memoria primaria (page table). Nella memoria primaria il dato c'è sicuramente perché o è stato

## **Polling e Interrupt**

*Discutere vantaggi e svantaggi del polling e dell'interrupt come metodo per programmare l'I/O.*

### **Risposta:**

#### **POLLING**

*Vantaggi:*

L'overhead pagato dal processore è relativamente piccolo

*Svantaggi:*

La cpu è costretta a sondare periodicamente se il dispositivo è pronto, risulta quindi difficoltoso calibrare correttamente il tempo di polling.

#### **INTERRUPT**

*Vantaggi:*

Con il metodo dell'interrupt la CPU è libera di eseguire altre operazioni mentre attende il completamento di un'operazione di I/O. Il dispositivo segnalerà il completamento con una interruzione asincrona al processore.

*Svantaggi:*

Il costo di una singola operazione di interrupt sono spesso maggiori del costo del polling, questo costo è dovuto al salvataggio/ripristino totale o parziale dello stato del programma in esecuzione e all'individuazione della corretta routine per gestire l'interruzione.

## **Banda reale di un disco**

*Supporre di conoscere il tempo medio di seek e la velocità di rotazione di un disco? Quali dati mancano per calcolare la banda reale? Come si calcola la banda reale del disco?*

### **Risposta:**

Sono necessari la Banda di Trasferimento del disco (solitamente espressa in MB/s) e i tempi di overhead necessari al controllore e la dimensione dei blocchi.

Con questi dati possiamo calcolare l'effettivo tempo impiegato per inviare un blocco:

$$\text{rotation\_latency} = 0.5 * 60 / \text{RPM}$$

$$\text{tempo\_trasferimento\_blocco} = \# \text{dim.blocco} / \text{banda disco}$$

$$\text{Tempo\_Trasferimento} = \text{seek\_time} + \text{rotation\_latency} + \text{overhead} + \text{tempo\_trasferimento\_blocco}$$

La banda si calcola come rapporto del numero effettivo di Byte inviati in un secondo.

$$\text{Banda Reale} = \# \text{dim.blocco (Byte)} / \text{Tempo\_Trasferimento (sec.)}$$

**9 settembre 2008**

### **Parte Controllo della CPU Multiciclo**

*In cosa si differenziano le parti controllo della CPU multiciclo e pipeline visti a lezione?*

**Risposta:** Vedi pag. 4

### **TBL e page table nella traduzione da add.virtuale a fisico [ DA FARE ]**

*Descrivere i ruoli di TLB e tabella delle pagine nella traduzione da indirizzo virtuale a fisico. Avendo una cache indirizzata con l'indirizzo fisico, si potrebbe verificare un page fault come conseguenza di un cache miss? (Motivare)*

**Risposta:**

### **Write-back e Write-through**

*A cosa si riferiscono e in che cosa consistono le politiche write-back e write-through?*

**Risposta:** Vedi pag 4

### **Prestazioni: Frequenza e MIPS**

*Perchè potrebbe essere scorretto comparare le prestazioni di due CPU usando la Frequenza del clock o la misura MIPS? Quali alternative?*

**Risposta:**

La frequenza ci dice quanti cicli di clock vengono eseguiti in un secondo (un processore molto veloce ma strutturalmente mal organizzato potrebbe necessitare di parecchi cicli per eseguire anche semplici operazioni), mentre la misura MIPS ci informa sul numero di milioni di istruzioni eseguite in un secondo (tale valore dipende dalla compilazione del codice in quanto è possibile che venga generato codice mal strutturato e di conseguenza operazioni semplici impieghino molte istruzioni ottenendo un MIPS elevato). Tali valori sono, quindi, spesso fuorvianti, le alternative sono il confronto tra lo speedup (rapporto dei tempi di esecuzione) su software noti e dei quali si dispone il sorgente da compilare sulla specifica macchina. Gli SPEC sono proprio questo.

### **CPU Multiciclo e riuso delle risorse**

*La CPU multiciclo vista a lezione riusa le stesse risorse in cicli di clock differenti. Esempificare rispetto all'istruzione di branch e all'uso della risorsa ALU.*

**Risposta:**

Nel caso delle istruzioni di BRANCH l'Alu viene utilizzata per calcolare l'indirizzo di salto, vediamo la sequenza di istruzioni:

1. (INSTRUCTION FETCH) Abbiamo il FETCH dell'istruzione e l'incremento del PC tramite la ALU per passare alla successiva operazione al termine della sequenza
2. (DECODE e LETTURA REGISTRI) In questo passo la ALU calcola il branch  $ALUOut = PC + (\text{sign-ext}(IR[15-0]) \ll 2)$ . Dobbiamo attendere che l'unità di controllo

decodifichi l'istruzione per confermare che si tratti di un branch e non, ad esempio, di un'operazione di somma con un imm16.

3. Se l' U.C. conferma che si tratta di una branch vengono confrontati i valori dei due registri RT ed RS e se sono uguali  $PC = ALUOut$

## 10 luglio 2008

### **PC-Relative e Branch: Calcolo dell'indirizzo target**

*Considerando la modalità di indirizzamento PC-relative delle istruzioni di branch, se all'indirizzo di memoria 0x4000ff00 abbiamo l'istruzione beq \$5,\$6,0x0080, qual'è l'indirizzo target nel caso in cui il branch fosse taken?*

**Risposta:** vedi pag 5

### **Benchmark**

*Illustrare la necessità di avere programmi di benchmark accuratamente selezionati per valutare e comparare le prestazioni dei sistemi. Esempificare rispetto ai benchmark SPEC e al relativo indice di prestazioni, e comparare rispetto a misure come i MIPS.*

**Risposta:** vedi pag 5

### **TLB Miss => Page Fault**

*Supporre che, come conseguenza di una lw o di un fetch di un'istruzione, si verichi un TLB miss. Potrebbe verificarsi un page fault? Come verrebbe risolto?*

**Risposta:**

Se si verifica un TLB miss bisogna andare a cercare la pagina nel Page Table, qui possiamo avere un HIT (la pagina c'è) o un FAULT (la pagina non è presente). Nel caso di fault bisogna ricaricare la pagina nella memoria e aggiornare la TLB.

### **Organizzazione di un disco ( CHS )**

*L'indirizzo di un blocco di dati in un disco viene indicato con la tripla Cilindro.Testina.Settore (CHS: Cylinder-Head-Sector). Spiegare rispetto alla tipica organizzazione di un disco.*

**Risposta:**

Ciascuna superficie (piatto) è divisa in tracce (che sono suddivise in settori) nello stesso modo. Ciò significa che quando la testina di una superficie è su una traccia, quella dell'altra superficie è anch'essa sulla traccia corrispondente. Tutte le tracce corrispondenti prese insieme si chiamano *cilindro*. Ne consegue che per individuare un blocco di dati la tripla Cilindro.Testina.Settore identifica univocamente il dato interessato nel disco rigido.

### **Dipendenza WAR (Write After Read) e RAW (Read After Write)**

*Perchè l'esistenza di una dipendenza WAR, anche se impedisce la modifica dell'ordine di esecuzione delle due istruzioni, non pone problemi alla nostra architettura pipeline a 5 stadi (non è necessario mettere in stallo la pipeline), mentre i problemi esistono (e sono risolti con il forwarding) per le dipendenze RAW?*

**Risposta:**

Le dipendenze WAR non sono critiche in quanto il dato viene scritto dopo che la precedente istruzione lo ha letto (quindi non modificato), in questo caso non ci sono problemi di congruenza quindi la dipendenza non è implica uno stallo. Nel caso delle RAW, se non ci fosse forwarding, avremo incongruenza nei dati in quanto il valore letto non corrispondente a quello modificato dalla precedente istruzione.

**24 giugno 2008****Caratteristiche CPU singolo ciclo, multi-ciclo e pipeline [ DA FARE ]**

*Discutere brevemente le caratteristiche della parte controllo nelle architetture delle CPU viste a lezione: a singolo ciclo, multi-ciclo e pipeline.*

**Risposta:****I/O Interrupt Driver – Inefficienza e soluzioni**

*Perchè la programmazione di I/O interrupt-driven è inefficiente per dispositivi veloci che trasferiscono in continuazione? Quale soluzione è adatta in questo caso?*

**Risposta:**

La gestione dell'I/O con interrupt è dispendiosa in quanto per ogni chiamata interrupt il programma in esecuzione viene bloccato (quindi salvato il suo stato) per passare alla routine di gestione dell'interrupt. Se gli interrupt sono frequenti allora ci troveremo in una situazione di “pseudo-stallo” del programma in quanto la CPU sarebbe sempre occupata a risolvere le chiamate di interrupt in arrivo. Una soluzione è l'utilizzo del DMA (Direct Memory Access) una tecnica che prevede una fase di inizializzazione del controllore nella quale si devono specificare le locazioni della memoria dedicate alle operazioni di I/O. Sarà quindi il controllore che si preoccuperà di inviare direttamente i dati in memoria senza passare per la CPU e di informarla solamente quando questi dati saranno disponibili e non per ogni transizione.

**Cache Miss => Page Fault ? [ DA VERIFICARE ]**

*Se accedendo la cache otteniamo un miss, è ancora possibile incorrere in un page fault? Discutere.*

**Risposta:**

Una cache miss non implica necessariamente un page fault in quanto ci troviamo a livelli diversi della memoria. Se si verifica una cache miss andiamo a cercare il dato in memoria quindi nella page table se anche lì non è presente allora avremo una page fault.

**CPU multi-ciclo e le operazioni di ADD e LW**

*La CPU multi-ciclo vista a lezione interpreta le istruzioni MIPS di add e lw rispettivamente in 4 o 5 cicli. Descrivere brevemente cosa accade nei vari cicli.*

**Risposta:**

I numeri indicati corrispondono ai cicli:

1. [ INSTRUCTION FETCH ] Viene presa l'istruzione da eseguire all'indirizzo indicato in PC dalla memoria e viene posta nell' INSTRUCTION REGISTER. Si incrementa  $PC = PC + 4$  e

lo si rimette in PC (siamo pronti per prendere la prossima istruzione alla fine dei cicli)

2. [ INSTRUCTION DECODE e LETTURA REGISTRI ] Vengono letti I registri RT ed RS dell'istruzione. Viene calcolato il salto di un'eventuale BEQ tramite il valore dei primi 16 bit dell'istruzione. Viene inviato il codice OP dell'istruzione all'Unità di Controllo per decodificare l'istruzione e determinare i segnali di controllo [DECODE].
3. [ DIPENDE DALLA DECODIFICA DELL'ISTRUZIONE ]
  - ADD : La ALU calcola  $ALUOut = A \text{ ADD } B$  (in A abbiamo RS e in B abbiamo RT, A e B sono due registri intermedi necessari nella CPU multiciclo per spezzare le operazioni)
  - LW : La ALU calcola l'indirizzo di memoria da leggere  
 $ALUOut = A + \text{sign-ext}(IR[15-0])$ ;
4. [ TERMINAZIONE DELL'ISTRUZIONE ]
  - ADD : Viene messo il risultato del passo 3 nel registro di destinazione  
 $REG [ IR [15-11] ] = ALUOut$
  - LW : Accediamo alla memoria in lettura per reperire il dato (stessa cosa in caso di scrittura)  
 $MDR (\text{Memory Data Register}) = MEMORY [ ALUOut ] \leftarrow$  in caso di lettura LW  
 $MEMORY [ ALUOut ] = B \leftarrow$  in caso di scrittura SW
5. [ WRITE – BACK ( LOAD) ]

In questo passo arriviamo esclusivamente per completare l'operazione di LW per poter salvare nel registro destinazione il valore presente in memoria e salvato nel MDR.  
 $REG [ IR [20-16] ] = MDR$

### **Trattamento del Page Fault come un'eccezione del S.O.**

*Il Page fault viene solitamente trattato come un'eccezione, ovvero come un salto ad una routine del Sistema Operativo. Motivare la scelta.*

#### **Risposta:**

Viene trattato come un'eccezione per permettere l'utilizzo di funzioni che, in user-mode, non sarebbero altrimenti disponibili. E' necessario passare in kernel-mode per poter avere accesso e modificare le entry della TLB, per modificare il Page Table Register. Quando succede una page-fault il processo deve essere bloccato fino al ripristino di uno stato sicuro (trasporto della pagina mancante in memoria), ma nel frattempo altri processi devono poter utilizzare la CPU questo cambio di contesto è un'operazione che fa il sistema operativo e non l'utente. Per queste motivazioni le page-fault si gestiscono come eccezioni del S.O.

### **19 febbraio 2008**

### **CPU singolo ciclo e addizionatori aggiuntivi per LW, ADD, BEQ [ DA FARE ]**

*Nel datapath della CPU a ciclo singolo vista a lezione sono stati inseriti addizionatori aggiuntivi (oltre alla ALU), per riuscire ad interpretare le istruzioni in un solo ciclo. Discutere come sono impiegati la ALU e gli addizionatori aggiuntivi rispetto alle istruzioni di lw, add e beq.*

#### **Risposta:**

## **Località**

*Discutere del concetto di località. C'è località nell'accesso alla memoria per il fetch delle istruzioni?*

### **Risposta:**

Se un elemento (ad es. Una Word) è riferita da un programma esso tenderà ad essere riferito ancora e presto (Località Temporale) e gli elementi vicini tenderanno ad essere riferiti anch'essi presto (Località Spaziale). Grazie al principio di località il funzionamento delle gerarchie di memoria è garantito. Nell'accesso alla memoria per il fetch delle istruzioni c'è località, essendo queste sequenziali (a meno di istruzioni di BRANCH o JUMP) viene aumentata la località spaziale quindi è assai probabile che l'istruzione successiva a quella eseguita sia immediatamente nella locazione successiva, sfruttando questo possiamo caricare un blocco di istruzioni abbastanza grande non pagando un miss per ogni caricamento di istruzione.

## **TBL miss => Page Fault ? Cache miss => Page Fault ? [ DA FARE ]**

*Come conseguenza di un TLB miss, si può verificare un page fault? E come conseguenza di un cache miss?*

### **Risposta:**

## **Pipeline, stalli e forwarding**

*Spiegare come, rispetto alla CPU pipeline a 5 stadi vista a lezione, il forwarding elimina possibili stalli successivi a istruzioni aritmetiche di tipo R, come ad esempio le add/sub. Spiegare con un esempio.*

**Risposta:** Vedi pag 4

## **Write-back e Write-through**

*A cosa si riferiscono e in che cosa consistono le politiche write-back e write-through?*

**Risposta:** Vedi pag 4

## **29 gennaio 2008**

## **PC-Relative [ DA FARE ]**

*Quando viene impiegata la modalità di indirizzamento PC-relative nelle istruzioni MIPS, e quali sono le motivazioni che hanno portato i progettisti ad adottare tale modalità?*

### **Risposta:**

## **Parte controllo della CPU multiciclo**

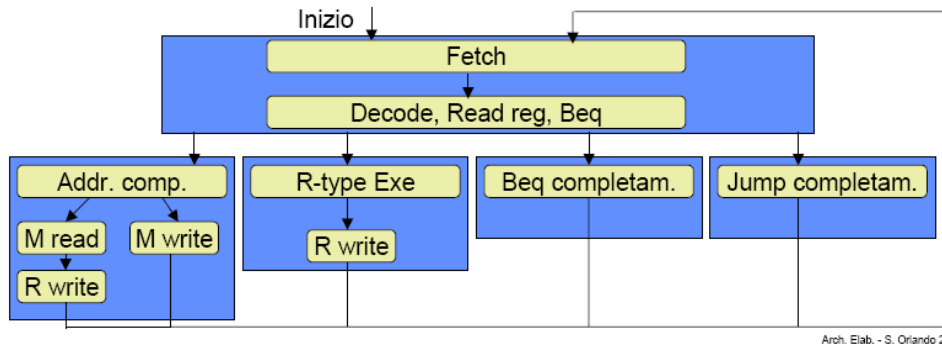
*Discutere la parte controllo del processore multiciclo visto a lezione relativamente al set limitato di istruzioni MIPS.*

### **Risposta:**

La parte controllo della CPU multiciclo si occupa di decodificare il codice OP dell'istruzione recuperata nella fase di FETCH di conseguenza invia i segnali di controllo ai singoli componenti per

la fase 3 che appunto è specifica di ogni singola operazione.

La parte di controllo viene implementata come un circuito sequenziale di Moore modellato come un automa a stati finiti. Gli output del controllo, quindi, dipendono dallo stato attuale.



## **Memoria Virtuale [ DA FARE ]**

Discutere in che ordine (facendo qualche ipotesi) sono acceduti TLB, Page Table, e cache nel caso di un sistema con memoria virtuale paginata, e con cache indirizzata con l'indirizzo fisico.

**Risposta:**

## **Località e dischi rigidi**

*Dato un file da memorizzare su disco, perchè è importante la disposizione dei blocchi di dati nelle varie tracce e settori?*

**Risposta:**

E' importante perchè ottimizzando la posizione dei blocchi del file si possono eliminare i ritardi dovuti agli spostamenti delle testine. Lo stesso guadagno si ottiene in fase di lettura, se i blocchi sono contigui la latenza di trasmissione si riduce di molto. Questo è il principio di località spaziale.

## **I/O interrupt-driven e trasferimento continuo**

Impiegare la programmazione di I/O interrupt-driven può essere vantaggioso rispetto all'utilizzo del processore nel caso di dispositivi veloci che trasferiscono in continuazione?

**Risposta:**

No, in quanto ogni interruzione provoca il blocco del processo attivo (con conseguente perdita di tempo per il salvataggio e ripristino dello suo stato), se il trasferimento è continuo sono continue le interruzioni quindi il processo risulta particolarmente lento perchè la CPU è spesso occupata a risolvere le interruzioni dovuto all'arrivo dei dati.

**12 settembre 2007**

## **Polling VS Interrupt**

*L'I/O si può programmare sia con il polling e sia tramite interrupt. Quando conviene l'uno o l'altro metodo? Distinguere tra dispositivi lenti o veloci, e per i veloci i casi in cui i dispositivi trasferiscono in continuazione o raramente.*

**Risposta:**

Il polling ha il vantaggio di occupare la CPU periodicamente per sondare lo stato dei dispositivi, nel caso di trasferimenti veloci il tempo di polling dovrà essere altrettanto veloce (altrimenti si rischia di perdere dei dati) quindi la CPU risulterebbe impegnata parecchio oppure ci potremo trovare nella situazione per cui la CPU sonda il dispositivo e questo non è mai pronto perdendo cicli utili per l'esecuzione di altri processi. Si può ovviare a questo problema utilizzando le interruzioni, in questo caso ribaltiamo la soluzione quindi il dispositivo segnalerà quando il dato è pronto e nel frattempo la CPU potrà eseguire altri processi.

Quindi nel caso di dispositivi lenti (come il mouse e il floppy disk) è sufficiente l'utilizzo del polling (tempo di controllo basso). Nel caso di dispositivi veloci (hard disk) dobbiamo distinguere tra il trasferimento continuo e quello frazionario. Nel trasferimento continuo il polling risulta vincente rispetto all'interrupt mentre per trasferimenti in brevi frazioni di tempo l'interrupt è la soluzione migliore.

**CICLI per una LOAD con CPU Multiciclo**

*Nel processore multiciclo visto a lezione, quanti cicli sono necessari per eseguire una load, come ad esempio lw \$2, 100(\$3)? Cosa succede al 3o ed al 4o ciclo di esecuzione della precedente istruzione?*

**Risposta:**

Per eseguire completamente una LW sono necessari 5 cicli.

Nel 3° ciclo viene calcolato dalla ALU l'indirizzo di memoria da leggere  
 $ALUOut = A + \text{sign-ext}(IR[15-0]);$

Nel 4° ciclo invece il valore viene portato nel Memory Data Register:  $MDR = MEMORY[ALUOut]$

**LW e memoria virtuale [ DA FARE ]**

*Discutere, rispetto ad un accesso alla memoria tramite lw, i ruoli di TLB, page table e cache. Si assuma che l'indirizzo fornito dalla lw sia virtuale, e che la cache sia acceduta con indirizzo fisico. Fare un esempio con un miss a livello di TLB: cosa possiamo aspettarci come conseguenza?*

**Forwarding e collegamenti datapath [ DA FARE ]**

*Discutere a grandi linee i collegamenti necessari nel datapath pipeline visto a lezione per realizzare il forwarding. In particolare, se un'istruzione add segue una lw, e tra le due esiste una dipendenza RAW, discutere come questo collegamento riduca (ad uno solo) gli stalli della pipeline.*

**Località Spaziale e Fetch delle istruzioni**

*Perchè nell'accesso in lettura alle istruzioni (fetch), la località spaziale contribuisce in maniera sostanziale all'incremento dell'hit-ratio? Abbiamo anche località temporale?*

**Risposta:**

La località spaziale contribuisce in maniera sostanziale in quanto le operazioni sono sequenziali quindi si troveranno in blocchi contigui ne consegue che le hit ratio saranno maggiori caricando blocchi vicini tra loro (nei quali con alta probabilità ci saranno le istruzioni successive da eseguire). Nel caso di loop abbiamo anche località temporale in quanto le stesse istruzioni saranno riferite con buona probabilità in archi di tempo molto brevi.

**4 luglio 2007**

## **Memoria Virtuale**

*Discutere in che ordine (facendo qualche ipotesi) sono acceduti TLB, Page Table, e cache nel caso di un sistema con memoria virtuale paginata, e con cache indirizzata con l'indirizzo fisico.*

**Risposta:** vedi pag 12

## **Località e array**

*Discutere del concetto e i tipi di località. C'è località nell'accesso alla memoria per il fetch delle istruzioni? E se accediamo un array e accumuliamo la somma dei vari elementi in un variabile in memoria?*

**Risposta:**

Per le prime due domanda vedi pag 11 .

In questo caso abbiamo località spaziale in quanto gli elementi di un array sono contigui di memoria di conseguenza caricando un blocco caricheremo n elementi (dipende dalle dimensioni del blocco e dalla dimensione di ogni elemento dell'array) in memoria che con buona probabilità saranno acceduti per effettuare il calcolo di somma.

## **Forwarding ed eliminazione dipendenze RAW**

*Nel microprocessore pipeline MIPS visto a lezione la presenza del forwarding e di un register file speciale (un register file che scrive nella prima parte del ciclo e legge nella seconda parte) garantisce nella maggior parte dei casi l'eliminazione degli stalli causati dalle dipendenze RAW. Spiegare con uno o più esempi.*

**Risposta:** vedi pag. 4

## **Previsione dei salti nella pipeline [ DA FARE ]**

*Perchè serve la previsione dei salti nei processori pipeline? In cosa consiste?*

## **BEQ + EXT-SIGN + SHIT << 2 perchè ?**

*Per implementare il beq abbiamo visto che sono necessari un circuito per l'estensione di segno, e uno per lo shift a sinistra di 2 bit. Perchè?*

**Risposta:**

E' necessario aggiungere tali circuiti in quanto l'ALU in quel momento risulta occupata da altre operazioni in questo modo non abbiamo attese e risulta possibile ottenere più risultati nello stesso ciclo di clock.

L'estensione di segno è necessaria in quanto l'indirizzo per il salto è composto da 32 bit mentre quello passato dall'istruzione è solo di 16 bit, in questo modo aggiungiamo i bit mancanti. Lo shift a sinistra di 2 bit è necessario in quanto tutte le istruzioni procedono a multipli di 4 (  $PC = PC + 4$  ) infatti shiftando di 2 bit a sinistra si moltiplica per 4.

**13 giugno 2007**

### **PC-Relative, completezza e regolarità delle istruzioni [ DA FINIRE ]**

*In cosa consiste la modalità di indirizzamento PC-relative impiegata dalle istruzioni beq? Quali sono i vantaggi rispetto alla completezza e regolarità delle istruzioni?*

**Risposta:**

La modalità di indirizzamento PC-Relative segue la regola:  $PC = PC + Costante$ , quindi il salto che andremo a fare è sempre relativo all'attuale istruzione.

Completezza e regolarità NON si trovano nelle slide

### **Cache miss => Page Fault ?**

*Supporre che come conseguenza di una lw o di un fetch di un'istruzione, si verifichi un cache miss. Potrebbe ancora verificarsi un page fault? Discutere.*

**Risposta:** vedi pag 9

### **Polling e trasmissione continua**

*Se abbiamo un dispositivo che trasferisce in continuazione, la modalità di programmazione dell'I/O denominata polling può essere efficiente? Discutere.*

**Risposta:**

Non è efficiente in quanto esistono tecniche (DMA) che permettono di gestire meglio queste situazioni. Nel polling il processore è impegnato per sondare periodicamente il dispositivo per valutare se i dati sono pronti o meno, nel caso di trasferimento continuo il polling dovrebbe avvenire con frequenza per non perdere dati, questo impegnerebbe parecchio la CPU quindi conviene adottare tecniche diverse.

### **Architettura multiciclo, ALU e l'istruzione BEQ**

*Rispetto all'architettura multiciclo, come viene riusata l'ALU durante i tre cicli di esecuzione dell'istruzione beq?*

**Risposta:**

Nel primo ciclo la ALU calcola  $PC = PC + 4$ , nel secondo ciclo la ALU è impiegata a calcolare il valore di  $ALUOut = PC + 4 + (\text{sign-ext}(\text{IR}[15-0]) \ll 2)$ , al terzo ciclo (se l'unità di controllo conferma che l'operazione è di branch e i valori dei registri RS ed RT è lo stesso) viene scritto il valore di ALUOut in PC e al prossimo ciclo verrà fatto il fetch dell'istruzione corrispondente al salto.

### **Organizzazione di un H.D. ( CHS ) e valori di overhead seek e rotation [ DA FINIRE ]**

L'indirizzo di un blocco di dati in un disco viene indicato con la tripla Cilindro.Testina.Settore (CHS: Cylinder-Head-Sector). Spiegare. Gli overhead di seek e rotation sono legati a quali valori della tripla?

**Risposta:**

Per la prima parte vedere pagina 8 la seconda è da finire

**14 febbraio 2007**

### **Cache miss => Page Fault ? TLB miss => Page fault ?**

*Se abbiamo una cache indirizzata con l'indirizzo fisico, può verificarsi un page fault come conseguenza di un cache miss? E come conseguenza di un TLB miss? Spiegare.*

**Risposta:** vedi pag 11

### **Bus sincroni e asincroni**

*Descrivere, a grandi linee, peculiarità, uso tipico e differenze dei bus sincroni e asincroni.*

**Risposta:**

BUS SINCRONO:

La linea di clock è condivisa tra tutti gli utenti del bus ne consegue che il protocollo che implementa la transazione sul bus sfrutta l'esistenza del clock comune (es. al ciclo  $x$  il dispositivo pone sul bus una richiesta di lettura alla memoria, al ciclo  $x + \Delta$  il dispositivo può leggere i dati posti sulle linee del bus dalla memoria,  $\Delta$  dipende dalla velocità della memoria). Viene limitato l'utilizzo a bus corti (come quello tra processore e memoria), se venisse utilizzato con bus più lunghi si possono manifestare problemi di sincronia dovuti al disallineamento del clock.

BUS ASINCRONO:

Permette l'utilizzo di periferiche con velocità diverse quindi per poter completare una procedura di I/O è necessario sincronizzarsi tramite un processo di handshaking (ad esempio DMA)

### **Caratteristiche CPU singolo ciclo, multiciclo e pipeline**

*Discutere brevemente le caratteristiche della parte controllo nelle architetture delle CPU viste a lezione: a singolo ciclo, multiciclo e pipeline.*

**Risposta:** vedi pag. 9

### **Forwarding: Esempi di op. aritmetiche dove c'è RAW**

*Il forwarding è una tecnica per eliminare gli stalli nelle architetture pipeline. Fare due esempi di forwarding con una coppia di istruzioni aritmetiche dipendenti (RAW), che appaiono consecutivamente o a distanza uno nel codice, rispetto all'architettura pipeline a 5 stadi.*

**Risposta:**

Es 1: add \$s0 , \$t0 , \$t1 sub \$t2 , \$s0 , \$t3  In questo caso se non ci fosse forwarding	Es 2: add \$s0 , \$t0 , \$t1 add \$s0 , \$s0 , \$t4 sub \$t2 , \$s0 , \$t3
--	---

l'operazione di SUB produrrebbe un risultato errato in quanto mentre viene eseguita la ADD non ha ancora memorizzato il valore in \$s0 (infatti si trova nella fase di MEM e non dopo WB)

In questo caso sia la 2a ADD che la SUB non darebbero un risultato corretto perchè il valore di \$s0 preso nella 2a ADD non è quello veritiero in quanto deve ancora essere scritto, stessa cosa quindi dicasi (in cascata) per la SUB

## **Semantica istruzioni MIPS**

*Qual'è la semantica della seguente istruzione macchina MIPS ? beq r1, r2, imm (Potete usare il Register Transfer Language per descrivere gli effetti dell'istruzione sul registro PC e commentare).*

### **Risposta:**

op | rs | rt | imm16 = M [ PC ] //eseguo la fetch dell'istruzione "beq r1 , r2 , imm"

la beq si traduce in:

```
if ( R[rs] == R[rt] ) then PC ← PC + 4 + (sign_ext(imm16)<<2);
                        else PC ← PC + 4;
```

Commento:

Nel caso il valore dei registri rs ed rt (r1, r2) fosse lo stesso il PC viene incrementato di 4 e del valore di imm aumentato di segno (portato a 32 bit) e shiftato a sinistra di 2.

**24 gennaio 2007**

**Cache ad accesso diretto (parte 1):**

Considerare una cache ad accesso diretto, con dimensione del blocco di 16 B (ovvero, il block displacement è uguale a 4 b), un INDEX di 12 bit. Quali dei seguenti riferimenti alla memoria fisica provocano conflitti, considerando che la cache all'inizio è vuota?

1. 0x1ABC0100
2. 0x1ABC0104
2. 0x1ABC0108
4. 0x1ABC010C
5. 0x00000100

**Risposta:**

L'indirizzo fisico è dunque a 32 bit di questi 32 bit abbiamo OFFSET = 4 bit e INDEX = 12 di conseguenza il TAG =  $32 - 4 - 12 = 16$  bit,

- |                | ----- TAG ----- ----- INDEX -----  OFF. |
|----------------|---|
| • 0x1ABC0100 = | 0001 1010 1011 1100 0000 0001 0000 0000 |
| • 0x1ABC0104 = | 0001 1010 1011 1100 0000 0001 0000 0100 |
| • 0x1ABC0108 = | 0001 1010 1011 1100 0000 0001 0000 1000 |
| • 0x1ABC010C = | 0001 1010 1011 1100 0000 0001 0000 1100 |
| • 0x00000100 = | 0000 0000 0000 0000 0000 0001 0000 0000 |

Alla fine della sequenza di letture sarà presente in memoria il blocco 0x010 = 16 quindi il 17° blocco di memoria. L'ultimo accesso crea un conflitto in quanto l'index è lo stesso delle precedenti operazioni mentre il TAG risulta diverso ne consegue che quando sarà richiesto quell'indirizzo fisico sarà rimossa dalla cache il blocco con TAG 0x1ABC e rimpiazzato da 0x0000.

**Cache ad accesso diretto (parte 2):**

Esiste località nei primi 4 accessi alla memoria? Di che tipo, spaziale o temporale, e perchè?

**Risposta:**

Abbiamo località spaziale in quanto sono tutti indirizzi che si riferiscono allo stesso blocco, cambia solamente l'OFFSET.

**Ruolo di TLB, Page Table e Cache [ DA CONTROLLARE ]**

Brevemente, qual'è il ruolo di TLB, Page Table e Cache nella realizzazione di un sistema di memoria (gerarchia di memoria)?

**Risposta:**

Cache: E' il livello di memoria (SRAM) più vicino alla CPU (oltre ai registri), grazie al principio di località la cache ha un senso in quanto i dati riferiti più di recente e quelli vicini saranno utilizzati

con buona probabilità nell'immediato, avere questi dati in cache (quindi in una unità di memorizzazione e trasmissione dati molto veloce) aumenta le prestazioni diminuendo il tempo computazionale dovuto alle attese. La cache è una memoria dal costo elevato in quanto veloce dunque la sua capacità risulta limitata da questo nasce la necessità di trovare dei sistemi per aumentare la capacità senza perdere in performance (memorie di secondo livello e memoria virtuale).

Page Table: La page table nasce con la memoria virtuale, all'interno della quale i dati sono organizzati secondo indirizzi virtuali, il suo scopo è mantenere la corrispondenza tra la pagina virtuale e la pagina fisica.

TLB: (Translation Lookaside Buffer) Risolve i problemi di latenza dovuti alla conversione tra indirizzo virtuale ed indirizzo fisico. E' una cache della page table e contiene al suo interno solamente le ultime pagine riferite (sfrutta il principio della località) permettendo un notevole guadagno di tempo in termini computazionali.

## **DMA vs Interrupt**

*Qual'è il ruolo e i vantaggi delle unità DMA per la realizzazione delle operazioni di I/O. Qual'è il vantaggio rispetto ad un sistema senza DMA, dove l'I/O è semplicemente interrupt-driven?*

### **Risposta:**

Le unità DMA permettono di demandare la gestione dell'I/O a dei controllori che opportunamente configurati accedono direttamente alla memoria per inserirvi il risultato di un'operazione di I/O. Una volta terminata l'operazione di inserimento viene avvisata la CPU che provvede a recuperare l'informazione. In questo modo la CPU non deve aspettare che l'operazione sia terminata ma può continuare ad eseguire i altri processi.

In un sistema senza DMA la CPU è direttamente coinvolta con la trasmissione dei dati dal dispositivo occupando cicli di clock che sarebbero dedicati all'esecuzione dei processi. Se la gestione dell'I/O avviene tramite interrupt c'è il rischio che la CPU sia impegnata costantemente per risolvere le chiamate di interruzione relative ai dati in arrivo (questo in caso di ricezione continua di dati da un dispositivo).

## **Predizione dei salti [ DA FARE ]**

*Perchè è importante la predizione dei salti nei computer moderni? Brevemente, discutere come viene solitamente realizzata.*

### **Risposta:**