Architettura degli Elaboratori

a.a. 2013/14 - terzo appello, 10 settembre 2014

Riportare nome, cognome, numero di matricola e corso A/B

Domanda 1

Sia U un'unità di I/O operante in Memory Mapped I/O e in DMA. Ad U è associata, sullo stesso chip, una gerarchia di memoria cache a due livelli identica a quella della CPU (vedi sotto). Il ciclo di clock τ è uguale a quello della CPU. Si assuma che tutte le interfacce con la CPU e con la memoria principale siano a transizione di livello.

U riceve da un qualunque processo CPU l'indirizzo *fisico* di un array A[N] di interi, con N = 24K, presente in memoria principale. U calcola l'operazione *reduce(A, +)* (somma di tutti gli elementi di A) e restituisce il risultato mediante un'interruzione. Lo handler di tale interruzione memorizza il risultato della *reduce* in una locazione X di indirizzo logico noto.

La memoria virtuale ha pagine di ampiezza 1K parole.

La memoria principale, di capacità massima 2Tera, è interallacciata con 8 moduli e ciclo di clock 30τ.

La gerarchia di cache a due livelli ha cache primaria associativa di capacità 32 K e blocchi di 8 parole, e cache secondaria associativa di capacità 512K e blocchi di 128 parole.

La latenza di trasmissione di qualunque collegamento inter-chip è uguale a 4τ. Le interconnessioni della CPU e dell'unità di I/O con la memoria principale sono entrambe ad albero binario.

La CPU ha architettura D-RISC pipeline scalare con comunicazioni a singola bufferizzazione e unità esecutiva con operazioni lunghe in pipeline a 4 stadi.

Esprimere la computazione mediante: *1)* il codice assembler, eseguito dalla CPU, per innescare la reduce e per il trattamento interruzione, *2)* il microprogramma di U. La computazione complessiva ha come parametro d'ingresso l'indirizzo *logico* di A presente in un registro generale, e si conclude quando il risultato della *reduce* viene scritto nella locazione X. Inserire adeguate spiegazioni.

Descrivere la fase firmware del trattamento interruzioni per l'architettura considerata.

Valutare, in funzione di N e τ, il tempo di completamento della computazione complessiva.

Domanda 2

Dire se la seguente affermazione è vera, falsa o vera sotto determinate condizioni, e spiegare la risposta: "per unità di elaborazione descritte da un microprogramma contenente solo condizioni logiche semplici e non facente uso di controllo residuo, la minimizzazione del ciclo di clock implica sempre la minimizzazione del tempo di servizio".

Traccia di soluzione

Domanda 1

La computazione complessiva consta delle seguenti fasi:

1. traduzione a programma dell'indirizzo logico di A nell'indirizzo fisico, e invio dell'indirizzo fisico di 41 bit all'unità di I/O mediante due STORE (identificatore di pagina fisica di 31 bit e displacement di 10 bit);
2. esecuzione della *reduce* nell'unità di I/O, e invio del risultato come secondo parametro del messaggio di interruzione;
3. fase firmware del trattamento interruzione;
4. fase software del trattamento interruzione: routine di interfaccia e handler (una STORE e ritorno da procedura).

In 1 si può supporre che il PCB e la tabella di rilocazione siano presenti in cache secondaria. La traduzione fa uso di operazioni di divisione e modulo.

In 2 si sfrutta la gerarchia a due livelli parallelizzando la lettura del blocco-C2 con l'elaborazione su blocchi-C1.

Per le fasi 1 e 4 il tempo di completamento si valuta sul codice assembler della CPU, per le fasi 2 e 3 mediante il microprogramma di U inclusa la gestione dei fault di cache per la *reduce*, e precisando le interazioni tra processore (IU, DM, EU), UNINT e unità di I/O.

1) La traduzione a programma dell'indirizzo logico di A fa uso esplicito della tabella di rilocazione, puntata dal PCB del processo in esecuzione, ed utilizza opportune estrazioni di campo mediante operazioni di divisione intera e modulo. Il codice, ottimizzato per la CPU pipeline, è il seguente:

1. LOAD Rpcb, Roffset, Rtabril // indirizzo della tabella di rilocazione //
2. DIV RA, Rpagesize, Ripl // identificatore di pagina logica dell'indirizzo base di A ; Rpagesize contiene la costante1024 //
3. MOD RA, Rpagesize, Rdispl // displacement //
4. LOAD Rtabril, Ripl, Rentry // entrata della tabella di rilocazione relativa alla prima pagina di A; I 31 bit meno significativi di questa parola contengono IPF //
5. MOD Rentry, Rnumpage, Ripf // estratto l'identicatore di pagina fisica; Rnumpage contiene la costante 2^{31} //
6. STORE RU, 0, Rdispl // RU contiene un indirizzo logico cui corrisponde l'indirizzo fisico di una struttura dati di due parole mappata nello spazio fisico dell'unità U //
7. STORE RU, 1, Ripf

É lecito assumere che PCB e tabella di rilocazione siano presenti in C2, per cui avremo due fault di blocco generati dalla 1 e 4, con penalità:

$$T_{fault-CPU} = 2 * 2\sigma_1\tau = 32\,\tau$$

Il codice contiene le dipendenze logiche intrecciate IU-EU della 2 su 4, 3 su 6, 4 su 5, 5 su 7, delle quali hanno effetto solo la 2 su 4 e 5 su 7, entrambe con k = 2, NQ = 2, Lpipe-k = 4, probabilità 1/7, senza ripercussioni delle dipendenze logiche EU-EU in quanto sempre mascherate. Quindi:

$$T_{servizio-0} = t + \frac{2}{7}\,t\,(N_Q - 1 + L_{pipe-k}) = \frac{17}{7}\,t$$

$$T_{c-0} = 7\,T_{servizio-0} = 34\,\tau$$

$$T_{c-CPU} = T_{c-0} + T_{fault-CPU} = 66\,\tau$$

2) Microprogramma dell'unità U. Le richieste di lettura memoria sono effettuate all'unità C1 associata:

0. (RDYCPU = 0) nop, 0; (= 1) reset RDYCPU, set ACKCPU, IN_CPU → DIPL, 1
1. (RDYCPU = 0) nop, 1; (= 1) reset RDYCPU, set ACKCPU, IN_CPU → IPF, IN_CPU.IPF ° DISPL → ADDR, 0 → REDUCE, 0 → I, 2
2. (I_0 = 0) ADDR → IND, set RDYOUTC1, 3;

   (= 1) set INT, 4
3. (RDYINC1 = 0) nop, 3; (= 1) reset RDYINC1, REDUCE + DATAIN → REDUCE, ADDR + 1 → ADDR, I + 1 → I, 2
4. (ACKINT = 0) nop, 4; (= 1) reset ACKINT, codice_evento → EVENTO_CPU, REDUCE → DATO_CPU, 0

Il tempo di servizio è, in pratica, quello del loop 2-3 per gli accessi in memoria:

$$T_U \sim 2N\,(\tau + t_c) + T_{fault-U} = 6\,N\,\tau + T_{fault-U}$$

Per quanto riguarda la gestione dei fault

$$T_{fault-U} = N_{fault-C2}\,T_{trasfM-C2} + N_{fault-C1}\,T_{trasfC2-C1}$$

A gode solo di località per cui:

$$N_{fault-C2} = \frac{N}{\sigma_2} \quad , \quad N_{fault-C1} = \frac{N}{\sigma_1}$$

Grazie al parallelismo tra il trasferimento di blocchi da M a C2 e daC2 a C1, si ha:

$$T_{trasfM-C2} = 2T_{tr} + \tau_M + lg_2 m\,(\tau + T_{tr}) = 53\,\tau$$

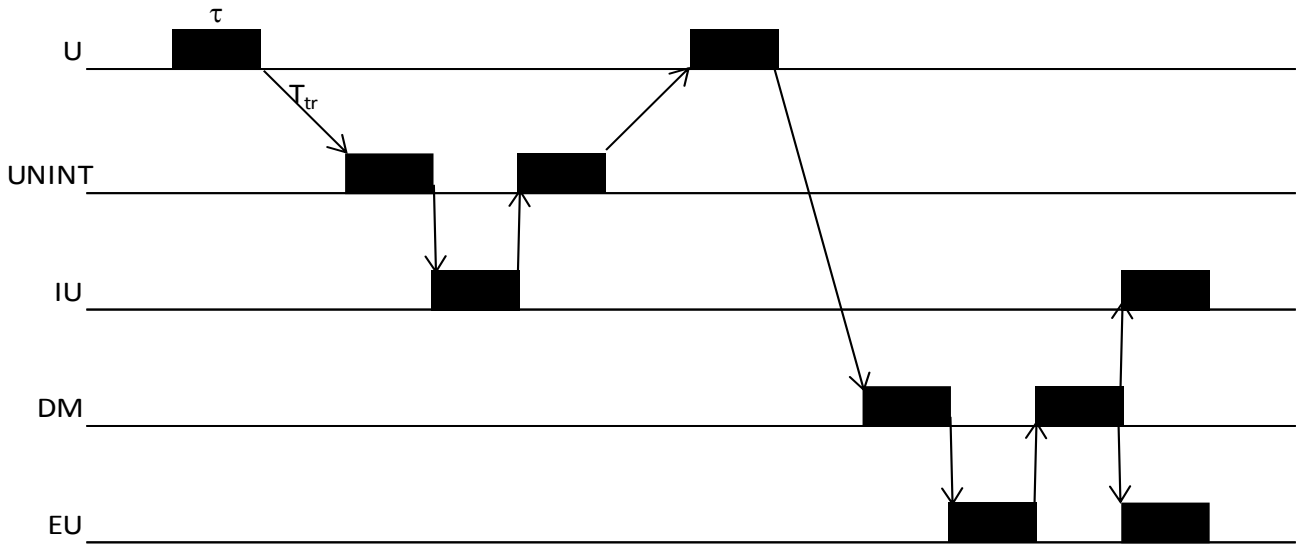
$$T_{trasfC2-C1} = 2\,\sigma_1\,\tau$$

Quindi si ricava:

$$T_{fault-U} = 2.42\,N\,\tau$$

$$T_U = 6N\tau + T_{fault-U} = 8.42\,N\,\tau$$

3) La fase firmware del trattamento interruzione ha il funzionamento mostrato nel seguente diagramma temporale:



L'interruzione arriva a IU, il messaggio di interruzione (due parole) a DM (MMU_D); le due parole sono scritte nei registri della EU. La IU è informata della ricezione delle due parole, in modo da emulare la chiamata di procedura alla routine di interfacciamento del trattamento interruzioni (ultimo ciclo di clock mostrato)

Il ritardo complessivo è quindi:

$$T_{int-fw} = 20\tau$$

4) Siano Revento e Rdato gli indirizzi dei registri generali in cui sono copiate le due parole del messaggio di interruzione, e Rretint quello per l'indirizzo di ritorno da interruzione. Il codice assembler per il trattamento interruzione è:

Routine di interfaccia:

```
LOAD   Rpcb, Roffset1, Rtabint
LOAD   Rtabint, Revento, Rhandler
CALL   Rhandler, Rrethandler
GOTO   Rretint
```

Handler:

```
STORE   RX, 0, Rdato
GOTO   Rrethandler
```

che comporta due fault di blocco con trasferimenti da C2 a C1 (anche la tabella dei puntatori agli handler può essere supposta in C2). Quindi:

$$T_{fault-int} = 32\ \tau$$

Il codice, di 6 istruzioni, contiene tre dipendenze logiche di distanza 1 (NQ = 2) e tre salti, quindi:

$$T_{int-servizio-0} = \frac{9}{6}\,t + \frac{3}{6}\,t\,(NQ + 1 - k) = \frac{15}{6}\,t$$

$$T_{int-c-0} = 6\,T_{int-servizio-0} = 30\,\tau$$

$$T_{int-c} = T_{int-c-0} + T_{fault-int} = 62\,\tau$$

In conclusione, il tempo di completamento della computazione complessiva è dato da:

$$T_c = T_{c-CPU} + T_U + T_{int-fw} + T_{int-c}$$

ovviamente dominato da T_U.

Domanda 2

Falsa. Nelle ipotesi date, il ciclo di clock è influenzato prevalentemente dal ritardo della funzione σ_{PO}. In presenza di composizione di funzioni, un solo ciclo di clock si rivela più conveniente di più cicli di clock in ognuno dei quali eseguire solo alcune funzioni.

Il semplice esempio del calcolo dell'espressione:

A + B + C

è sufficiente. Se eseguita in un solo ciclo di clock, usando due ALU in cascata, il tempo di servizio è:

$$T_1 = \tau_1 = T_{\omega-PC} + 2\,T_{ALU} + \delta$$

Se eseguita in due cicli di clock:

$$T_2 = 2\,\tau_2 = 2\,(T_{\omega-PC} + T_{ALU} + T_k + \delta)$$

$T_{\omega\text{-}PC}$ è uguale nelle due realizzazioni.