Architettura degli Elaboratori, 2013-14

Prima prova di verifica intermedia

20 dicembre 2013

Riportare nome, cognome e numero di matricola

Domanda 1

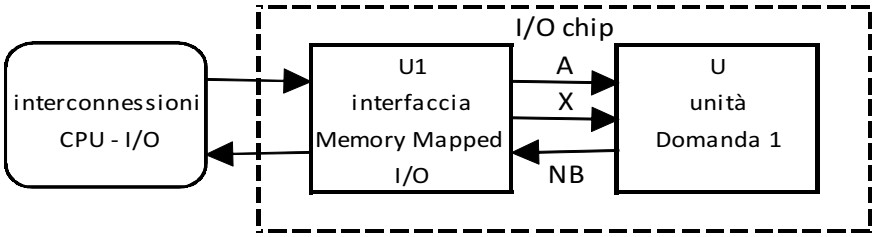
Un'unità di elaborazione U interagisce a domanda-risposta con U1: U riceve una parola A di 32 bit e un byte X, e restituisce il numero NB di byte di A aventi valore uguale a X.

Progettare U con l'obiettivo di minimizzarne il tempo di servizio T. Spiegare le scelte fatte per soddisfare questa specifica. Valutare T in funzione di t_p e t_{alu}.

Domanda 2

Compilare una procedura D-RISC, con parametri d'ingresso e di uscita in registri generali, che calcola la stessa funzione della Domanda 1 (numero di byte di una parola A uguali a X) nei due casi in cui il sistema contenga o non contenga un co-processore di I/O specializzato a firmware.

La struttura del co-processore è mostrata in figura:



U1, che costituisce l'unità di I/O vera e propria, operante in Memory Mapped I/O, fa in modo che i valori A, X siano inoltrati a U, e il valore NB sia inoltrato correttamente alla CPU. Se ritenuto necessario o conveniente, fare ipotesi sulla realizzazione firmware di U1.

Nel caso di *non* co-processore, valutare il tempo di completamento e il tempo di servizio per istruzione in funzione del ciclo di clock della CPU, del ciclo di clock della memoria principale, e della latenza di trasmissione dei collegamenti inter-chip. Agli effetti della valutazione, si assume trascurabile la probabilità che X sia uguale a un byte di A.

Facoltativo: dare la valutazione anche nel caso di esistenza del co-processore, assumendo che questo abbia lo stesso ciclo di clock della CPU, che la latenza di trasmissione inter-chip sia la stessa di cui sopra, e che l'interconnessione con la CPU sia costituita da collegamenti dedicati.

Domanda 3

Dire quali informazioni sono contenute nel file di configurazione di un processo, e spiegare come tali informazioni sono utilizzate in fase di creazione e caricamento del processo.

Soluzione

Domanda 1

La computazione richiesta

$$NB = f(A, X)$$

è una funzione pura, per la quale è agevole definire e realizzare una rete combinatoria con componenti logici standard. Di conseguenza, il microprogramma è descrivibile da una sola microistruzione:

0. (RDYIN = 0) nop, 0; (= 1) reset RDYIN, set RDYOUT, $f(A, X) \rightarrow$ NB, 0

e l'unità è realizzabile come singola rete sequenziale. Questa soluzione ha tempo di servizio uguale a un ciclo di clock:

$$T = \tau = T_\sigma + \delta = T_f + t_p$$

La rete combinatoria *f* comprende il confronto in parallelo dei quattro byte di A con X, seguito dalla *reduce* ad albero dei risultati dei confronti.

Questo tempo di servizio risulta certamente minore di quello di qualunque altra soluzione che richieda più cicli di clock per quanto con ciclo di clock minore. Tali soluzioni, infatti, sarebbero realizzate secondo il modello generale con due reti sequenziali, per cui il ritardo complessivo sarebbe *almeno* uguale a T_f più i ritardi della Parte Controllo e/o della funzione delle uscite della Parte Operativa (ad esempio, utilizzando quattro cicli di clock in ognuno dei quali si confronta un byte distinto, oppure due cicli di clock in ognuno dei quali se ne confrontano due in parallelo).

La funzione di transizione dello stato interno è data da:

$$in_{NB} = when\ (\beta_{NB} = 1)\ do\ f(A, X)$$

$$in_{RDYOUT} = when\ (\beta_{RDYOUT} = 1)\ do\ not\ RDYOUT$$

$$in_{Y\_RDYIN} = when\ (\beta_{RDYIN} = 1)\ do\ not\ Y_{RDYIN}$$

$$\beta_{NB} = \beta_{RDYOUT} = \beta_{RDYIN} = RDYIN$$

La rete combinatoria *f* è costituita dalla cascata di

- quattro confrontatori a 8 bit in parallelo, ognuno con ingresso X e un byte di A e seguito da una porta OR a 8 bit con uscita negata,
- un albero di ALU con profondità due: nelle foglie due ALU con ingressi da 1 bit e uscita da 2 bit, nella radice una ALU con uscita a 3 bit.

Realizzando i confrontatori a due livelli di logica:

$$\tau = 2\ t_{alu} + 4t_p$$

Realizzando il confronto come funzione zero della differenza mediante ALU:

$$\tau = 3\ t_{alu} + t_p$$

Domanda 2

Nella *versione senza co-processore*, in D-RISC la procedura deve fare uso di operazioni di estrazioni di campo realizzate con istruzioni DIV e MOD (o MUL a seconda della posizione del campo) aventi il secondo operando potenza di due. Il risultato dell'estrazione di campo fornisce il campo stesso nei bit meno significativi ponendo a zero tutti i rimanenti bit.

Lo stesso parametro X è rappresentato nel byte meno significativo di una parola (parametri passati in registri generali).

Lo pseudo-codice della procedura è:

```
{ NB = 0;
  for (i = 0; i < 3; i++) {
        lastbyte = A % 256;
        if (X == lastbyte)
                    NB++;
        A = A/256 }
}
```

Oltre ai registri generali per i parametri d'ingresso (RA, RX), di uscita (RNB) e ritorno da procedura (Rret), il compilatore alloca i registri temporanei Ri, Rlastbyte, e Rcost_256 inizializzato alla costante $2^8 = 256$ (unico registro inizializzato per la compilazione della procedura). I registri modificati verranno, se necessario, salvati dal main nell'immagine del PCB prima della chiamata e ripristinati al ritorno da procedura.

La compilazione D-RISC:

```
        CLEAR   Ri
        CLEAR   RNB
LOOP:   MOD     RA, Rcost_256, Rlastbyte
        IF ≠    RX, Rlastbyte, CONT
        INCR    RNB
CONT:   DIV     RA, Rcost_256, RA
        INCR    Ri
        IF <    Ri, 4, LOOP
        GOTO    Rret
```

Agli effetti della valutazione, il numero di istruzioni eseguite è:

$$N_{istr} = 23$$

Il tempo di completamento per il calcolatore elementare:

$$T_c = 23\,T_{ch} + 8\,T_{ex-aritm-lunga} + 8\,T_{ex-IF} + 6\,T_{ex-aritm-corta} + T_{goto} = \\ = 23(2\tau + t_a) + 40\tau + 16\tau + 6\tau + \tau = 109\,\tau + 23\,t_a$$

Esprimendo il tempo di accesso equivalente in memoria $t_a = 2(\tau + T_{tr}) + \tau_M$, si ha infine:

$$T_c = 155\tau + 23(2T_{tr} + \tau_M)$$

Il tempo di servizio per istruzione vale:

$$T = \frac{T_c}{N_{istr}} \sim 6.74\,\tau + 2T_{tr} + \tau_M$$

La *versione con co-processore* utilizza la tecnica Memory Mapped I/O per passare i valori A, X all'unità U1 e per ricevere da essa il risultato NB.

Il compilatore alloca nella memoria virtuale del processo una pagina il cui indirizzo logico base costituisce l'inizializzazione di un registro generale RI/O. Questa pagina, a tempo di caricamento (in base alle informazioni contenute nel file di configurazione), *viene mappata nella memoria fisica locale di U1*.

Come da specifiche, è compito di U1 interagire con la CPU e con U:

- attende i due valori A, X, inviati *in sequenza* dalla CPU mediante due istruzioni di STORE mappate nello spazio fisico di U1; invia A, X *in parallelo* a U;
- attende il risultato NB da U; non appena riceve la richiesta di lettura del risultato da parte della CPU (LOAD), gli ritorna il valore NB.

In generale (indipendentemente dai ritardi temporali) U1 attende la richiesta di lettura dalla CPU e il risultato NB da U.

La memoria locale di U1 in realtà viene implementata con tre registri indipendenti (A_{U1}; X_{U1}, NB_{U1}).

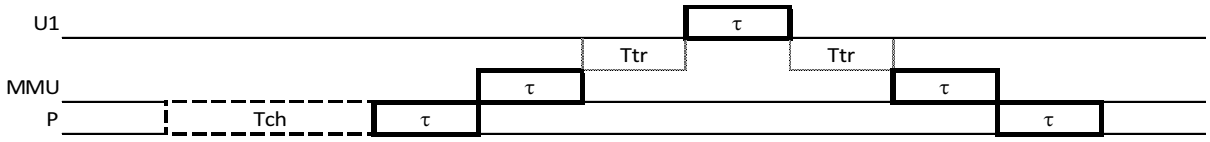
La procedura viene compilata come:

```
STORE   RI/O, 0, RA
STORE   RI/O, 1, RX
LOAD    RI/O, 2, RNB
GOTO    Rret
```

In alternativa, la sincronizzazione per "risultato pronto" può essere effettuata utilizzando l'istruzione WAITINT in attesa attiva, e di conseguenza realizzando U1 in modo da segnalare tale evento con una interruzione alla quale, in questo semplice caso, è associato anche il valore del risultato comunicato nel messaggio di interruzione stesso (evento = risultato_pronto, dato = NB):

```
STORE   RI/O, 0, RA
STORE   RI/O, 1, RX
WAITINT   Rmask, Rrisultato_pronto, RNB
GOTO    Rret
```

Agli effetti della valutazione, con le ipotesi del testo circa l'interconnessione CPU-I/O, le istruzioni di LOAD e STORE nello spazio fisico di U1 sono eseguite come segue:



Quindi:

$$T_{ex-LD/ST} = 5\,\tau + 2\,T_{tr}$$

Di conseguenza (il calcolo in U è interamente "coperto" dalla seconda STORE o dalla LOAD):

$$T_c = 4\,T_{ch} + 3\,T_{ex-LD/ST} + T_{goto} = 4\,(2\tau + t_a) + 3\,(5\tau + 2T_{tr}) + \tau = 32\tau + 14T_{tr} + 4\tau_M$$

che risulta notevolmente inferiore (circa 5 – 10 volte) rispetto al tempo di completamento della versione senza co-processore, grazie al bassissimo tempo di servizio della funzione *f(A, X)* realizzata a firmware e grazie al minor numero di istruzioni eseguite. Per quest'ultimo motivo il tempo di servizio per istruzione risulta superiore, anche se paragonabile:

$$T = \frac{T_c}{4} = 8\,\tau + 3.5\,T_{tr} + \tau_M$$

Domanda 3

Il file di configurazione è costruito a tempo di compilazione usando

- la struttura della memoria virtuale prodotta dal compilatore stesso,
- le informazioni, in possesso del compilatore, sull'architettura assembler-firmware e sul supporto a tempo di esecuzione dei processi,
- eventuali informazioni addizionali o annotazioni (pragma) fornite dall'utente.

Il file di configurazione descrive (meta-file) la struttura del file eseguibile specificando la posizione e dimensione allineate alla pagina di *ogni oggetto* nella memoria virtuale del processo:

- codice visibile a programma
- dati visibili a programma
- PCB del processo
- Tabella di Rilocazione del processo
- codice (librerie) del supporto a tempo di esecuzione del processo
- dati del supporto a tempo di esecuzione del processo.

Inoltre, per ogni oggetto viene specificato:

- se è condiviso e con quali processi è condiviso,
- se deve fisicamente essere allocato nella memoria principale o nella memoria locale di una determinata unità di I/O in Memory Mapped I/O.

Infine, viene specificato:

- l'insieme di lavoro, o comunque un sottoinsieme minimo di oggetti che deve essere caricato in memoria principale: almeno PCB, Tabella di Rilocazione, (parte iniziale del) codice, (parte riferita inizialmente dei) dati.

Queste informazioni sono utilizzate per la funzione di allocazione (iniziale) della memoria principale. Il gestore della memoria

- tranne che per le pagine condivise già allocate, sceglie le pagine (di memoria principale e di memoria di I/O) in cui caricare l'insieme di lavoro,

e, utilizzando anche gli identificatori di pagina fisica delle pagine condivise già caricate,

- inizializza la Tabella di Rilocazione inserendo l'identificatore di pagina fisica e settando il bit di presenza delle entrate relative alle pagine logiche allocate (gli altri campi possono essere inizializzati a compilazione).

Quindi, utilizzando il file system per riferire le pagine del file eseguibile e dando mandato all'unità di I/O (o driver) della memoria secondaria, viene provocato il trasferimento delle pagine nella memoria fisica (principale o di I/O).

L'ultimo atto è il concatenamento del PCB nella Lista Pronti.