Architettura degli Elaboratori, 2007-08

Appello del 6 febbraio 2008

Domanda 1

Si consideri il programma D-RISC di figura.

I registri generali di indirizzo RA, RB contengono gli indirizzi base di due array A, B di *N* interi ciascuno, con *N* dell'ordine delle decine di K. Il registro di indirizzo Ri è inizializzato a zero.

Si consideri la sequenza delle prime 10 istruzioni eseguite a partire dall'istruzione immediatamente precedente a quella di etichetta LOOPj quando RG[Ri] = 0.

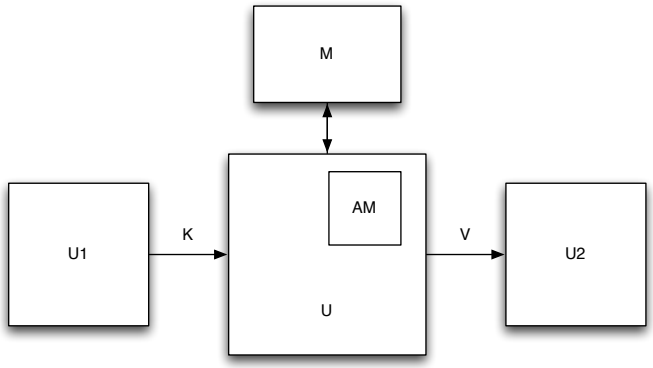
Dettagliare la sequenza degli accessi in effettuati memoria, evidenziando gli indirizzi logici e fisici interessati (con tutte le eventuali assunzioni del caso), nonché gli eventuali fault di cache e di pagina. Si descrivano (a parole e in modo sintetico) le azioni svolte dalle unità di elaborazione della CPU in corrispondenza di tali fault.

```
LOOPi:  LOAD   RA, Ri, Rai
        …
        CLEAR  Rj
LOOPj:  LOAD   RB, Rj, Rx
        ADD    Rx, Rai, Rai
        INCR   Rj
        IF<    Rj, RN, LOOPj
        STORE  RA, Ri, Rai
        INCR   Ri
        IF<    Ri, RN, LOOPi
        …
```

Domanda 2

Si consideri il sistema a livello firmware riportato in figura.

L'unità di elaborazione U contiene un componente logico memoria associativa AM di 16 posizioni; ogni posizione consta di un campo *chiave* di 24 bit e di un campo *valore* di 32 bit; il valore è accedibile per contenuto o per indirizzo, la distinzione essendo effettuata in funzione di un ingresso secondario (1 bit) chiamato *tipo_accesso*. *Gli altri ingressi e uscite di AM devono essere definiti*. Il tempo di stabilizzazione di AM, per qualunque tipo di accesso, è uguale a $5t_p$, dove t_p è il ritardo di stabilizzazione di una porta logica con al più 8 ingressi.



L'unità M contiene un componente logico memoria RAM di capacità 16M parole, con parole di 32 bit. Il ciclo di clock di M è uguale a $200t_p$. La latenza di trasmissione dei collegamenti tra unità è uguale a $50t_p$.

U riceve da un'unità U_1 una chiave *K* e invia ad un'unità U_2 il valore *V* corrispondente a *K* in AM, se esiste. Se non esiste in AM, U ottiene il valore *V* dall'unità M all'indirizzo *K*; oltre che essere inviato a U_2, V è memorizzato in AM nella posizione di indirizzo *f(K)*, con *f* implementata da una rete combinatoria data avente ritardo di stabilizzazione di $3t_p$.

Scrivere il microprogramma di U e determinarne il tempo medio di elaborazione in funzione di t_p e della probabilità *p* che in AM non esista un valore corrispondente a *K*.

Domanda 3

a) Una computazione a livello di processi contiene i processi A, B, C. Il processo A trasmette il valore di un array M di 1K interi a B e, solo quando è sicuro che B abbia ricevuto tale valore, trasmette a C lo stesso valore. Supponendo che tutta la computazione sia descritta in LC, scrivere il processo A e le parti di B e C corrispondenti. *Usare due forme di comunicazione alternative per la cooperazione tra A e B*.

b) Supponendo che le azioni di A siano svolte da un'unità di I/O, ma che non siano esprimibili in LC, descrivere come deve essere implementata la stessa computazione del punto *a)*, spiegando le azioni svolte da A, B e C, con B e C processi interni.

Domanda 1

La sequenza degli accessi in memoria è determinata dalle istruzioni nel frammento del programma assembler. In particolare, dal programma assembler si può evincere la traccia degli indirizzi logici generati dal programma, che comporteranno un accesso in memoria. La traccia degli indirizzi logici è la seguente:

LOOPj-1, LOOPj, RG[RB], LOOPj+1, LOOPj+2, LOOPj+3, LOOPj, RG[RB+1], LOOPj+1, LOOPj+2, LOOPj+3, …

I simboli nella forma *LOOPi* indicano indirizzi logici di istruzioni assembler, quelli nella forma *RG[RB+k]* indicando invece indirizzi logici degli elementi dell'array B. Gli indirizzi fisici sono noti solo a tempo di esecuzione, e si sono ricavati utilizzando la tabella di rilocazione TABRIL; ciò è implementato nella MMU utilizzando un sottoinsieme di TABRIL allocato dinamicamente in una memoria associativa. Indicando con *IPL(ind)* l'identificatore di pagina logica dell'indirizzo *ind* e con *OFF(ind)* l'offset all'interno della pagina logica, la sequenza di indirizzi fisici effettivamente generati è la seguente:

TABRIL[IPL(LOOPj-1)]..OFF(LOOPj-1),
TABRIL[IPL(LOOPj)]..OFF(LOOPj),
TABRIL[IPL(RG[RB])]..OFF(RG[RB]),
TABRIL[IPL(LOOPj+1)]..OFF(LOOPj+1),
…

(dove .. rappresenta la concatenazione). In generale, occorre tener conto del caso in cui l'entry di TABRIL relativa a IPL(xx) nella tabella di rilocazione contenga il flag di presenza a FALSE, ad indicare che la corrispondente pagina fisica non è ancora stata caricata in memoria. Se questo fosse il caso, si avrebbe un *fault di pagina*.

Per il frammento di codice considerato, potremmo assumere senza perdita di generalità che la pagina corrispondente a IPL(LOOPj) sia già stata caricata in memoria, visto che sono state eseguite già almeno una volta le istruzioni che precedono la LOOPj (nel solo caso in cui il codice da LOOPj in poi occupi le prime posizioni di una nuova pagina logica si potrebbe verificare anche un fault di pagina).

Viceversa, non dovremmo fare assunzioni sulla pagina corrispondente a IPL(RG[RB]), dal momento che l'array B non è ancora stato acceduto (RG[Ri] vale 0) e quindi dovremmo assumere che il primo accesso allo stesso array (lettura di TABRIL[IPL(RG[RB])]..OFF(RG[RB])) generi un fault di pagina.

Per quanto riguarda invece i fault di cache, assumiamo che la cache abbia blocchi (linee) di ampiezza σ parole:

- l'istruzione all'indirizzo LOOPj-1 è stata sicuramente appena eseguita, quindi si troverà in cache. Se le istruzioni da LOOPj a LOOPj+3 si trovano nello stesso blocco (ampio σ) della LOOPj-1, non avremmo altri fault di cache, diversamente ne avremmo (per il segmento di codice considerato, lungo 4 istruzioni) $4/\sigma$.
- la prima parola dell'array B, all'indirizzo RG[RB]+RG[Rj] (=RG[RB] visto che RG[Rj] vale 0), non si troverà sicuramente in cache e quindi all'accesso verrà generato un fault di cache, risolto peraltro interamente da quest'ultima unità, senza che il processore se ne accorga.
- i successivi accessi per il frammento di codice considerato non genereranno altri fault (sia il codice che l'array sono in cache) a meno che la cache non sia ad indirizzamento diretto e BC(TABRIL[IPL(LOOPj+k)]..OFF(LOOPj+k)) sia uguale a BC(RG[RB]+RG[Rj]).

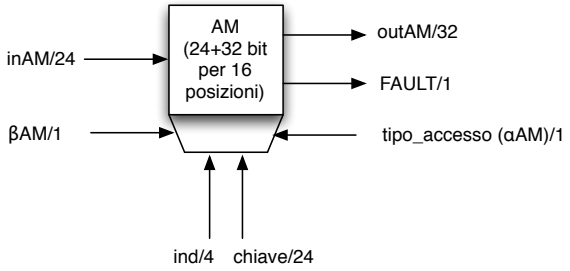
In caso di fault di pagina, la MMU lo rileva, restituisce un ESITO diverso da 0 al processore che, nel esegue una prima fase di trattamento dell'eccezione a livello firmware, la quale provoca (eventualmente tramite una routine di interfaccia) la chiamata all'handler corrispondente al tipo di eccezione. Tale handler chiama, o comunica con, il gestore della memoria. Il processo passa in attesa o esplicitamente ("chiama") o implicitamente ("comunica") per effetto dell'esecuzione di primitive di comunicazione bloccanti con il gestore della memoria come processo partner. Il gestore della memoria provvede a trovare una pagina di memoria "vittima", ordinarne lo scaricamento in memoria secondaria (se modificata), ordinare il caricamento della pagina che ha generato il fault nella pagina fisica appena

allocata, e modificare la corrispondente entrata della tabella di rilocazione. A questo punto il gestore della memoria, o l'unità disco, provvede alla sveglia del processo che aveva avuto fault (mediante l'esecuzione della primitiva sbloccante).

In caso di fault di cache, l'unità cache provvede a trovare un blocco "vittima", a scaricarlo in memoria principale (se modificato e se viene adottata la tecnica Write Back) e successivamente a caricare tutto il blocco che contiene la parola richiesta al suo posto, passando quindi la parola richiesta al processore.

Domanda 2

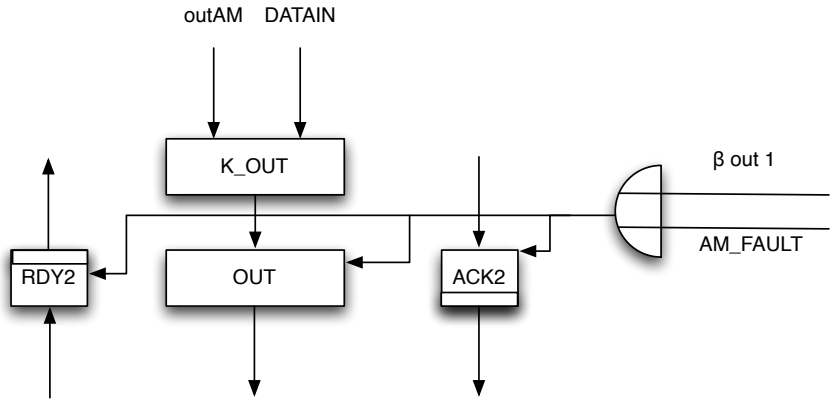
Si assume (come da testo) che la memoria associativa sia realizzata come in figura:



Facendo uso di controllo residuo per modificare i valori dei "beta" dell'interfaccia verso U_2, microprogramma relativo alla unità U può essere scritto come segue:

0. (RDY1,ACK2=0-) nop, 0;
   (=11) reset RDY1, setACK1, K->K1, AM[K,"per_contenuto"].FAULT -> FAULT,
   (AM[K,"per_contenuto"].VAL -> OUT,set RDY2, reset ACK2) | AM[K,"per_contenuto"].FAULT==0,1
1. (FAULT=0) nop, 1;
   (=1) K1->IND, "read"->OP, set RDYoutM, f(K1)->IND_AM, 2
2. (RDYinM=0) nop, 2;
   (=1) reset RDYinM, DATAIN->OUT, reset RDY2, set ACK2, DATAIN->AM[IND_AM,"per_indirizzo"],0

La *nop* nella 1 è eliminabile. Potremmo anticipare le operazioni della 0 nel ramo FAULT=0 della 1, testando nella 1 anche RDY1 e ACK2. Si noti anche che potremmo eliminare K1 facendo sì che nella 0 f(K)->IND_AM e K->IND.
Infine, si noti l'uso del controllo residuo per far sì che scrittura nel registro OUT verso U_2 e le relative operazioni di sincronizzazione non avvengano nella 0 in caso di fault:



Analizzando il microprogramma, secondo il procedimento di derivazione formale PO/PC, possiamo disegnare l'intera PO e derivare la PC. Entrambe risultano molto semplici e non vengono qui riportate.

Il calcolo del ciclo di clock avviene come segue:

- $T_{\omega PO}$ vale 0, tutte le variabili di condizionamento sono uscite di registri
- $T_{\sigma PO}$ vale un tempo di accesso alla AM più il tempo di attraversamento del commutatore prima del registro OUT, necessario perché in OUT andrà o l'uscita della memoria associativa (in caso di hit) o DATAIN (in caso di fault), quindi $T_{\sigma PO} = 7\ t_p$
- $T_{\omega PC}$ e $T_{\sigma PC}$ valgono entrambi $2t_p$ (si vede dal microprogramma: 3 stati (2 bit, quindi) e 4 variabili di condizionamento).

Quindi complessivamente

$$\tau = 0 + max\ \{2t_p,\ 2t_p + 7t_p\} + t_p = 10\ t_p$$

Il tempo medio di elaborazione è dato da:

$$T = \tau + p\ t_a = \tau + p\ (2\tau + 2T_{tr} + \tau_m) = (1+2p)\ \tau + p\ (2T_{tr} + \tau_m) =$$
$$= 10t_p + 20p\ t_p + p\ (2*50+200)t_p = \ = (10 + 320\ p)\ t_p$$

Domanda 3

a) Per essere certo dell'avvenuta ricezione del messaggio trasmesso, A può utilizzare una canale sincrono verso B: sia *chB*. In questo caso, il codice di A sarà del tipo:

```
A:: channel out chB, chC, …

send(chB,M);
send(chC,M);
…
```

In questo caso la send su *chC* sarà eseguita senz'altro dopo che il messaggio M è stato ricevuto dal processo che effettuerà la receive corrispondente sul canale *chB*.
Diversamente, il canale da A a B avrebbe potuto anche essere asincrono, ma in questo caso occorre una nuova comunicazione per la conferma della ricezione (un messaggio senza valore ( )), con una primitiva bloccante per A. Potevamo quindi utilizzare un canale asincrono con una successiva receive, su un canale (preferibilmente asincrono, in modo da non rallentare le operazioni di B):

```
A:: channel out chB, channel in risp(1), … ::
…
send(ch,M);
receive(risp,( ));
send(chC,M);
…
```

Nel primo caso, il codice di B sarà una cosa del tipo:

```
B:: channel out chB (0), …
…
receive(chB,arrayM);
…
```

mentre nel secondo caso sarà :

```
B:: channel out ch (1), …
…
receive(chB,arrayM);
send(risp, ( ) );

…
```

Il codice di C, uguale in tutti e due i casi, sarà

```
C:: channel out chC (1), …

receive(chC,arrayM);
…
```

b) In questo caso dobbiamo assumere che l'interazione con l'unità di I/O A debba avvenire senza la mediazione di LC. Le azioni relative alla comunicazione con A sono dunque in tutto e per tutto simili a quelle ipotizzate nella soluzione del primo esercizio del primo appello (vedi anche seocnda prova di verifica intermedia):

a. il processo B richiede la lettura di M, utilizzando MMI/O per passare il comando di lettura ed i relativi parametri all'unità A: *riferimento alla variabile M* e *riferimento al* PCB_B, dove "riferimento" assume un significato diverso a secondo del metodo usato per risolvere il problema delle strutture condivise riferite indirettamente (*da specificare*), ad esempio capability;

b. A trasferisce i dati in M (allocato in memoria principale, se si dispone di DMA, o comunque nella memoria di I/O) e genera un'interruzione il cui messaggio associato contiene il riferimento a PCB_B;

c. il trattamento di tale interruzione provoca la sveglia del processo B;

d. il processo B effettua una comunicazione al processo C per comunicare l'avvenuta lettura di M;

e. il processo C, con meccanismi analoghi a quelli appena visti per B, ordina la lettura di M all'unità A, …

f. …