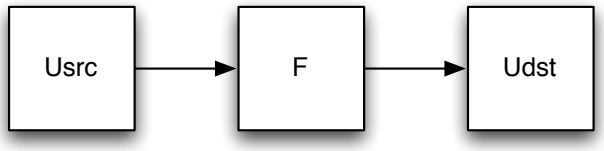
Appello straordinario per studenti lavoratori A.A. 2008-2009

Architetture degli elaboratori

7 aprile 2009

Domanda 1

Un'unità firmware F ha il compito di filtrare indirizzi IP. Dall'unità U_{src} riceve indirizzi IPv4 (parole da 32 bit), controlla che se siano presenti in una memorietta interna M da 4K parole e successivamente, solo nel caso in cui l'indirizzo non sia presente, lo invia all'unità U_{dst}. Qualora l'indirizzo fosse fra quelli in M, F invia a U_{dst} l'indirizzo *0.0.0.0* (32 bit tutti a zero). Si realizzi l'unità discutendo in particolare le scelte effettuate fra varie alternative possibili, e si forniscano le prestazioni dell'unità F in funzione di t_a, tempo di accesso per M, e t_p, ritardo di una porta logica di al più 8 ingressi.



Domanda 2

Si consideri il microcodice:

0. A+1→A,1
1. B+1→B,2
2. (zero(A+B)=0) C+1→C,3; (=1) C-1→C,4
3. TEMP + IN → A,5
4. TEMP - IN → B,5
5. A + B → OUT

Si fornisca un microprogramma che calcola gli stessi risultati con un minor numero di cicli di clock.

Domanda 3

1. Descrivere mediante pseudo-codice l'implementazione di una send asincrona con grado di asincronia pari a *k*.
2. Descrivere mediante pseudo-codice l'implementazione della primitiva *commuta_contesto*.

Traccia di soluzione

Domanda 1

Il microcodice potrebbe essere strutturato come segue:

0. (RDY$_{in}$=0) nop, 0; (=1) IP$_{in}$ → TEMP, 1 → I, M[0] → MM, reset RDY$_{in}$, set ACK$_{in}$, 1

1. (zero(TEMP-MM),I0,RDY$_{out}$=001) 0→IP$_{out}$, set ACK$_{out}$, reset RDY$_{out}$, 0
   (000) nop, 1
   (10-) I+1 → I, M[i] → MM, 1
   (111) TEMP → IP$_{out}$, reset RDY$_{out}$, set ACK$_{out}$, 0
   (110) nop, 1

Alla prima microistruzione si attende di ricevere un IP, nel caso lo si riceva, lo si salva in un temporaneo, si legge la prima posizione del vettore e si salta alla 1. Alla 1. si confronta l'ip con quanto appena letto dalla memoria. Se uguale, il ciclo termina immediatamente, passando all'unità U$_{dst}$ uno 0.0.0.0. Se diverso si prosegue fino ad aver scandito tutta M.

Scelte effettuate

- non introdurre una *zero(TEMP-M[I])* come variabile di condizionamento, ma spezzare il test fra un'istruzione di lettura e una di test fra registri
- utilizzo di un contatore per scorrere la memoria dal basso verso l'alto, Si esce dal ciclo quando si ha un trabocco sul bit 0

Valutazione delle prestazioni:

$T\omega_{PO} = T_{ALU} = 5T_p$

$T\sigma_{PO} = T_a$

$T\omega_{PC} = T\sigma_{PC} = 2t_p$ (due stati, 4 variabili di condizionamento)

$tau = 5t_p + T_a + 2t_p + t_p = 8t_p + t_a$

Numero delle istruzioni eseguite: $1 + 4K/2 * 1 + 1$ (inizio, ciclo, fine) $\approx 2K$

$T = 2K * (8t_p + t_a)$

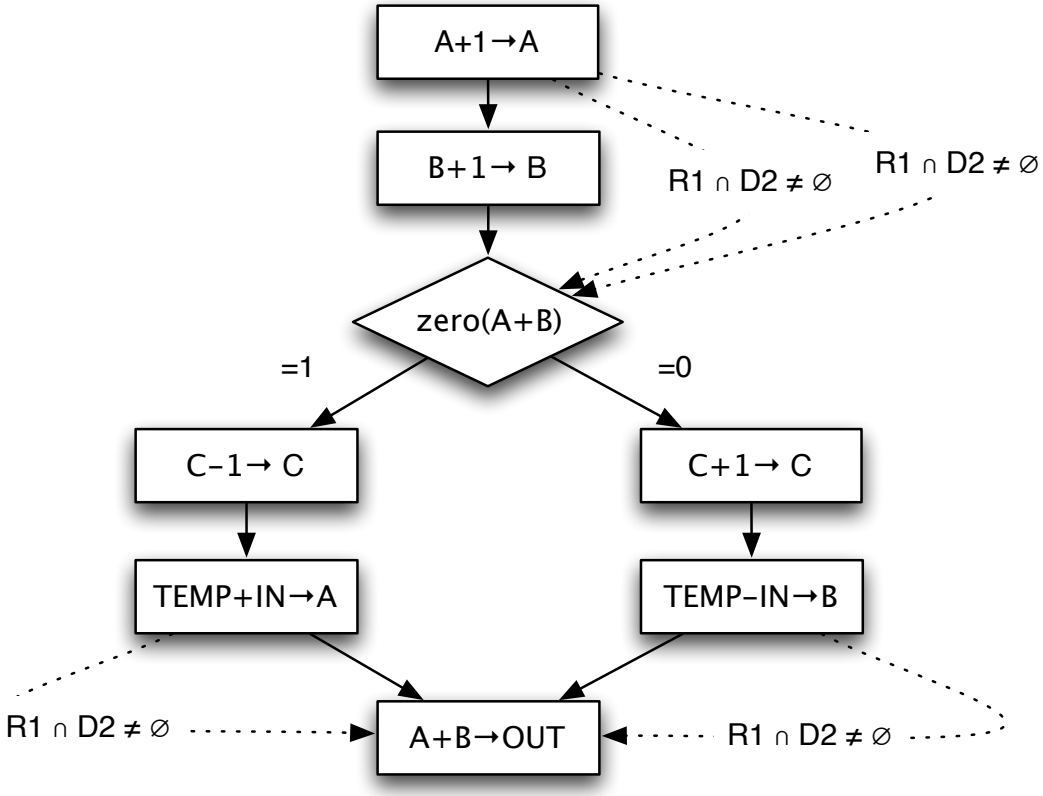
$P = 1/T$

Domanda 2

Occorre considerare le condizioni di Bernstein per cercare di raggruppare le microoperazioni che possono essere effettivamente parallelizzate. Va ovviamente anche tenuto conto del flusso del controllo nel microprogramma. Le due microoperazioni alla 0. e 1. non hanno dipendenze, quindi possono essere raggruppate in un'unica microistruzione. La 2. va considerata come composta da 2 microoperazioni: zero(A+B) e, a seconda dell'esito del test o la C+1 → C o la C-1 → C. Pertanto non può essere parallelizzata insieme alle microperazioni della 0. e 1. originale, visto che sia A che B sono nel rango di tali operazioni e nel dominio della 2. Per continuare la parallelizzazione, conviene analizzare il flusso del controllo dopo l'if della 2.

I due flussi di controllo derivanti dall verifica o meno della condizione zero(A+B) possono essere analizzati separatamente. Si nota che sia la microoperazione che riscrive C che quella che riscrive A non sono in dipendenza fra di loro nè con la zero(A+B) e quindi possono essere raggruppate nella stessa microistruzione, ovviamente mantenendo i due rami della condizione come originariamente prescritto dal codice sequenziale. A+B→OUT

ha una dipendenza con entrambe le TEMP+IN→A e TEMP-IN→B e dovrebbe quindi



rimanere in una microistruzione separata e successiva a quella che ospita le TEMP +IN→A e TEMP-IN→B. Il microcodice quindi potrebbe essere:

```
0.  A+1→A,B+1→B,1
1.  (zero(A+B)=0) C+1→C, TEMP + IN → A,2; (=1) C-1→C, TEMP - IN → B,2
2.  A + B → OUT
```

Osservando che nella 2. A e B sono rispettivamente il risultato appena calcolato di TEMP +IN e TEMP-IN, si potrebbe anche scrivere:

```
0.  A+1→A,B+1→B,1
1.  (zero(A+B)=0) C+1→C, TEMP + IN → A,TEMP+IN+B→OUT,2;
    (=1) C-1→C, TEMP - IN → B,TEMP-IN+A→OUT,2
2.  ...
```

Mentre il passaggio dalla forma sequenziale alla prima forma parallela richiede solo un aumento di risorse (2 ALU, oltre a quella per la condizione, ne bastava una sola per le microistruzioni nella forma sequenziale) e quindi non influisce sul ciclo di clock, questa seconda forma parallela aumenta il ciclo di clock in quanto richiede un utilizzo in cascata di due ALU (per la TEMP+IN+B e TEMP-IN+A) mentre precedentemente si aveva al massimo l'utlizzo di una sola ALU, oltre agli eventuali commutatori.

Domanda 3

Vedi dispensa.