# *Architettura degli elaboratori – A. A. 2008-2009*

*Prova scritta - Terzo appello – 19 giugno 2009*

*Indicare su tutti i fogli consegnati Nome, Cognome, numero di matricola e corso (A/B) I risultati e il calendario degli orali saranno resi noti sulle pagine web dei docenti appena pronti.*

**Domanda 1**

Un'unità $U_c$ è dotata di una memorietta interna M da 256 parole, riceve da $U_a$ due valori X e VAL, entrambi da 32 bit, e ed esegue le seguenti operazioni:

- se X è maggiore di 0, controlla se VAL è uguale alla i-esima posizione di M, dove i è la posizione del bit a 1 più significativo di X. Nel caso i due valori siano uguali, invia tale valore ad una terza unità $U_b$, altrimenti invia ad $U_b$ il valore 0.
- se X è minore o uguale a 0, conta gli elementi di M minori di 0 e ne invia il numero a $U_b$

Si fornisca il microprogramma dell'unità realizzato in modo da eseguire la prima operazione in un unico ciclo di clock e se ne determini la tempo medio di elaborazione assumendo che le due operazioni abbiamo probabilità $p_{OP=0}$=0.75 e $p_{OP=1}$=0.25, che il tempo di accesso ad M sia pari a $10t_p$, e che il ritardo di una ALU sia pari a $5t_p$ ($t_p$ ritardo di una porta logica di al più 8 ingressi).

**Domanda 2**

Si consideri il seguente frammento di programma scritto in LC:

```
channel out chan1,chan2; ...
...
for(int i=0; i<N; i++)
  acc = acc + f(x[i]);
if(acc<0)
  send(chan,acc);
else
  send(chan2,acc);
...
```

Se ne fornisca la compilazione in linguaggio assembler, utilizzando le convenzioni adottate nel corso, tenendo conto che

a. f è una funziona precompilata che riceve il parametro in ingresso e restituisce il risultato tramite locazioni di memoria, il cui indirizzo è passato per valore dal chiamante,
b. chan1 e chan2 sono canali asincroni con grado di asincronia pari ad 1,
c. la send è la procedura del supporto a tempo di esecuzione di LC vista nel corso.

Si discutano infine le eventuali differenze nel codice compilato derivanti dall'uso di canali sincroni.

**Domanda 3**

Si dica se le seguenti affermazioni sono vere o false (e si specifichi sotto quali condizioni) e si motivi la risposta (risposte senza adeguata motivazione verranno considerate comunque errate):

1) un processo ha nella propria memoria virtuale i descrittori dei soli canali utilizzati dal processo stesso (cioè di quelli nominati in una *send* o in una *receive* nel codice del processo),
2) in un automa che modella una rete sequenziale di Mealy, ogni arco ha associate due distinte etichette: una che indica lo stato di ingresso che determina la transizione su quell'arco e una che determina lo stato di uscita in corrispondenza dello stato di arrivo dell'arco stesso,
3) una cache associativa su insiemi a *k* vie contiene *k* m* blocchi, dove *m* è funzione di σ (ampiezza dei blocchi della cache).

**Bozza di soluzione**

**Domanda 1**

Il microprogramma dell'unità può essere espresso come segue:

0. ($RDY_{in}$, $X_0$, zero(VAL-M[f(X)]), $ACK_{out}$ = 0---) nop, 0
   (=10-0) nop, 0
   (=1011) reset $RDY_{in}$, set $ACK_{in}$, reset $ACK_{out}$, set $RDY_{out}$, VAL → OUT, 0
   (=1001) reset $RDY_{in}$, set $ACK_{in}$, reset $ACK_{out}$, set $RDY_{out}$, 0 → OUT, 0
   (=11--) 255→IND, 0→COUNT, reset $RDY_{in}$, set $ACK_{in}$, 1
1. ($IND_0$, $ACK_{out}$, $M[IND]_0$=0-0) IND-1→IND, 1
   (=0-1) IND-1→IND, COUNT+1→COUNT, 1
   (=11-) COUNT→OUT, reset $ACK_{out}$, set $RDY_{out}$, 0
   (=10-) nop, 1

In questo caso si assume che:

- $M[IND]_0$ sia il bit del segno del valore letto dalla memoria all'indirizzo IND
- IND sia un registro da 9 bit, di cui testiamo il bit del segno per capire quando il ciclo debba terminare
- M sia una memoria a doppia porta. Questo è necessario per rispettare la condizione di correttezza, dal momento che utilizziamo f(X) per indirizzarla per la generazione della variabile di condizionamento e IND per indirizzarla durante la generazione del risultato della seconda operazione esterna.

La f(X) è calcolata da una rete combinatoria. La tabella della verità segue lo schema:

| x1 | x2 | x3 | ... | x31 | y0 | y1 | y2 | y3 | y4 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | - | - | | - | 1 | 1 | 1 | 1 | 1 |
| 0 | 1 | - | | - | 1 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | | - | 1 | 1 | 1 | 0 | 1 |
| | | | | | | | | | |
| 0 | 0 | 0 | | 1 | 0 | 0 | 0 | 0 | 0 |

Ognuno dei bit in uscita yi è dunque il risultato della somma (OR) di al più 31 valori risultato del prodotto (AND) di al più 31 variabili di ingresso (x0 non si considera, dal momento che la f(X) ha senso solo per valori positivi, secondo la specifica delle operazioni esterne). Con porte da 8 ingressi occorrono due livelli per l'AND e 2 per l'OR, per un complessivo ritardo di $4t_p$. Si noti che la tabella della verità qui sopra calcola solo 5 degli 8 bit necessari a indirizzare la memorietta e pertanto, detta g la funzione implementata da questa tabella della verità avremo che

$$f(X) = 000.g(X)$$

ovvero gli 8 bit sono ottenuti concatenando 3 zeri in testa ai 5 calcolati come indicato sopra.

Il ciclo di clock unità può essere valutato come segue:

- $T_{\omega PO}=19t_p$ (tempo di accesso alla memoria, più ritardo della f(X), più ritardo della ALU)

- $T_{\sigma PO}= 5t_p$ (ritardo di una ALU, ne usiamo due in parallelo per la terza frase della seconda microoperazione, e ne usiamo una dedicata per la variabile di condizionamento)
- $T_{\omega PC}=2t_p$ (2 soli stati interni e 6 variabili di condizionamento)
- $T_{\sigma PC}=2t_p$ (come sopra)

per un totale di $\tau = 27t_p$.

Ai fini del calcolo del tempo medio di elaborazione consideriamo

$$T = p_{op0} * 1\tau + p_{op1} * (1+256)\tau = 3/4\ \tau + 1/4\ (257\tau) \cong 65\tau = 1755\ t_p$$

Punti da mettere in evidenza:

a. utilizzo di una ALU dedicata per la variabile di condizionamento zero(VAL-M[f(X)]) e di una memoria a coppia porta (1o indirizzo f(X), 2o indirizzo IND) per rispettare la condizione di correttezza
b. uso diretto del bit più significativo (bit del segno) dell'uscita della memoria M come variabile di condizionamento per verificare se il valore letto è minore di 0
c. uso di una rete combinatoria dedicata per il calcolo di f(X) (vedi sopra)

**Domanda 2**

In questo caso, la compilazione del segmento di programma LC dovrebbe essere:

```
        ...
        add R0, R0, Rind                    indice del vettore
        add R0, R0, Racc                    azzeramento accumulatore
        add R0, RbaseTemp, RparamUscita     preparazione indirizzo parametro uscita
loop:   add RbaseX, Rind, RparamIngresso    preparazione indirizzo parametro ingresso
        call Rf, Rret                       chiamata procedura f
        LD RparamUscita, R0, Rtemp          recupero parametro uscita
        add Rtemp, Racc, Racc               accumulazione risultato finale
        inc Rind                            incremento variabile di iterazione
        IF< Rind, Rn, loop                  controllo fine ciclo
        IF<0 Racc, send1                    fine ciclo, controllo if then else
send2:  mov Rchan2, Rchan                   ramo else (prima parte)
        goto end
send1:  mov Rchan1, Rchan                   ramo then (prima parte)
end:    st RbaseMsg, R0, Racc               parte comune a ramo then e ramo else
        mov RbaseMsg, Rmsg
        mov #1, Rlen
        call Rsend, Rret                    spedizione risultato
        ...
```

Non vi è alcuna differenza nel codice compilato se consideriamo l'uso di canali sincroni anzichè di quelli asincroni. Le differenze saranno confinate nel supporto a tempo di esecuzione di LC (implementazione della send, in questo caso, che tuttavia viene chiamata con gli stessi parametri).

Si noti che:

- add $R_0$, $R_0$, $R_{ind}$ corrisponde ad una clear $R_{ind}$
- $R_{paramIngresso}$ ed $R_{paramUscita}$ rappresentano i registri utilizzati per passare per valore gli indirizzi in memoria dei parametri in ingresso ed in uscita alla procedura che calcola f
- $R_{msg}$, $R_{chan}$ e $R_{len}$ sono i tre parametri di ingresso alla send (indirizzo del messaggio, identificatore del canale e lunghezza del messaggio)

- l'indirizzo del parametro di ingresso, essendo deciso dal chiamante, può essere trattato in modo da non rendere necessaria una load del valore x[i] dalla sua posizione corrente seguita da una store in una locazione il cui indirizzo sia poi passato alla f, semplicemente utilizzando l'indirizzo della x[i] (dove si trova in memoria) come parametro in ingresso a f
- le mov $R_{chanX}$, $R_{chan}$ sono necessarie perchè il parametro canale varia nei due rami dell'i*f-then-else*, mentre la mov $R_{baseMsg}$,$R_{msg}$ e la mov #1, $R_{len}$ poteva essere evitata avendo cura di inizializzare opportunamente $R_{msg}$ e $R_{len}$ con la solita tecnica vista a lezione (costante opportuna nella posizione del PCB da dove viene inizializzato il registro in fase di commutazione di contesto)
- la compilazione dell'ultimo *if-then-else* è stata ottimizzata riconoscendo le parti a comune fra i due rami (inizializzazione dei parametri lunghezza del messaggio e messaggio stesso

**Domanda 3**

1. Falso a condizione che le unità di I/O non siano capaci di eseguire il supporto per le comunicazioni con processi interni (in questo caso il processo corrente deve essere in grado di eseguire la virtualizzazione dell'interruzione eseguendo una send su un canale che non ha dichiarato nel proprio codice), vero altrimenti.
2. Falso. L'etichetta relativa all'uscita è quella che determina l'uscita nello stato corrente, non nello stato successivo, a seguito del verificarsi dell'ingresso con cui è etichettato l'arco
3. Vero. *m* è di fatto dato da

$$m(\#C,\sigma,b,k) = (\#C / (\sigma + b)) / k$$

dove *#C* è la capacità totale (in parole) della cache, σ è il numero di parole per blocco di cache, *k* è il numero di blocchi per insieme, e *b* è la lunghezza (in parole) delle informazioni di controllo associate ad ogni singolo blocco: verosimilmente *b=1* dato che le informazioni di controllo contengono il TAG, il bit di presenza ed eventuali bit per la gestione dell'algoritmo di rimpiazzamento.
Se la domanda fosse stata intesa come "funzione *unicamente* di σ", questa affermazione sarebbe stata ovviamente falsa.