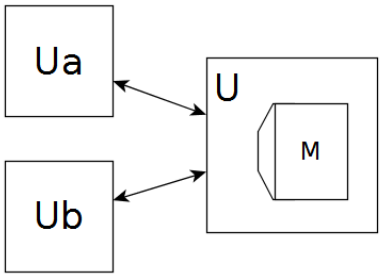
Architettura degli elaboratori – Prima prova intermedia A.A. 2015/16 – 2/11/2015

Riportare su tutti i fogli consegnati Nome, Cognome, Matricola, Corso A o B.
I risultati saranno resi disponibili via WEB appena disponibili (home page docenti e/o didawiki.di.unipi.it)

Si richiede di progettare una unità firmware U che interagisce con altre due unità firmware U_a e U_b. U mantiene al suo interno una memoria M di 2K posizioni (parole da 32 bit) e implementa tre distinte operazioni esterne:

- scrive un valore X nella memoria all'indirizzo IND
- cerca un valore X nella memoria e ne sostituisce tutte le eventuali occorrenze con X/8192 (si assume che tutte le posizioni di memoria siano significative ai fini della ricerca)
- calcola il minimo fra tutti gli elementi contenuti nella memoria e lo restituisce come risultato dell'operazione esterna



Le prime due operazioni vengono richieste solo dall'unità U_a, la terza solo dall'unità U_b. U_a ed U_b interagiscono con U secondo un protocollo a domanda risposta.

Si richiede di progettare l'unità U e di fornirne il tempo medio di elaborazione, sotto le seguenti assunzioni:

- siano disponibili porte logiche AND e OR da al più 4 ingressi
- il ritardo introdotto da ALU che implementano operazioni standard sia $t_{alu}=6t_p$
- siano disponibili unicamente ALU che eseguono sottrazione e addizioni fra parole da 32 bit, quindi con un α di controllo da 1 bit
- il tempo di accesso alla memoria M sia $t_a=10t_p$
- il numero di microistruzioni da eseguire per ciascuna della operazioni esterne sia minimo.

Traccia di soluzione

Interfacce

Con U_a:

- in ingresso X (32 bit), IND (11 bit), RDY_1 (indicatore a transizione di livello), OP (1 bit);
- in uscita ACK_1 (indicatore a transizione di livello)

Con U_b:

- in ingresso RDY_2 (indicatore a transizione di livello)
- in uscita ACK_2 (indicatore a transizione di livello), OUTMIN (32 bit)

Microprogramma

Prima versione: diamo precedenza alle richieste di U_a (arbitrariamente). Utilizzando variabili di condizionamento complesse possiamo scrivere il codice come segue:

0. (RDY_1, RDY_2, OP = 00-) nop, 0;
   (=1-0) X → MEM[IND], reset RDY_1, set ACK_1, 0;
   (=1-1) 0 → I, 1;
   (=01-) MEM[0] → MIN, 1 → I, 2
1. (I0,zero(MEM[I]-X)=00) I+1 → I, 1;
   (=01) SHIFTR(MEM[I],13)→ MEM[I], I+1 → I, 1;
   (=1-) reset RDY_1, set ACK_1, 0;
2. (I0,segno(MIN-MEM[I]=00) I+1 → I, 2;
   (=01) MEM[I] → MIN, I+1 → I, 2;
   (=1-) MIN → OUTMIN, reset RDY_2, set ACK_2, 0

Il numero di microistruzioni eseguite è minimo. Per ciclare su N posizioni servono almeno N istruzioni più una istruzione per l'inizializzazione (o la finalizzazione) del calcolo. Questo accade per le operazioni esterne 2 e 3, mentre per l'operazione esterna 1 si esegue un'unica microistruzione.

Classi di registri

- OUTMIN: vi si scrive solo MIN, direttamente, serve β_{outmin}
- I: si inizializza e si incrementa, serve un commutatore in ingresso, con relativo α_{kI}, e un β_i
- MEM: indirizzata con IND, 0 e I. Serve una memoria a doppia porta visto che MEM[I] è utilizzato per le variabili di condizionamento. Sul primo indirizzo serve un commutatore per scegliere fra IND e 0, dunque serve un α_{ind}. Vi si scrive X o il risultato dello shift, quindi serve un commutatore sugli ingressi con relativo α_{kM}. Serve un β_m per regolare le scritture.
- MIN: si inizializza e si aggiorna o con le due uscite della memoria, dunque serve un commutatore con relativo α_{kmin}. Serve un β_{min}.
- Per RDY_1, ACK_1, RDY_2, ACK_2 servono i relativi β per set e reset.

Nel microprogramma ci sono 10 frasi e vengono testate al più 3 variabili di condizionamento. Servono 2 bit di stato perché abbiamo 3 microistruzioni. Dunque gli ingressi per il livello AND della σPC e ωPC sono 5. Gli ingressi per il livello OR sono sembre 5 (10 frasi, al più 5 “1” (altrimenti si codificano gli zeri negando l'uscita). Con porte da 4 ingressi a disposizione servono due livelli di porte AND e due livelli di porte OR. Il ritardo delle reti σPC e ωPC sarà dunque di $4t_p$.

Costo delle singole frasi

ωPO

Per la prima microistruzione il costo è 0, per la seconda e la terza è $t_a + t_{alu}$

σPO

0.0 0

0.1 $t_k + t_a$ (i commutatori sull'ingresso di M e sugli indirizzi stabilizzano in parallelo)

0.2 t_k

0.3 $t_k + t_a + t_k$

1.0 $t_{alu} + t_k$

1.1 $t_k + t_a$

1.2 0

2.0 t_k

2.1 $t_a + t_{alu} + t_k$

2.2 t_k

Calcolo del ciclo di clock

Il ciclo di clock è calcolato come quello necessario per completare la frase più lunga, cioè la 2.1

$\tau = t_a + t_{alu} + \max\{ 4t_p, 4t_p+t_a+t_{alu}+t_k\} + t_p = 10t_p + 6t_p + 4t_p+10t_p+6t_p+2t_p + t_p = 37t_p$

Calcolo del tempo medio di elaborazione

Per le operazioni esterne:

- k = 1 per la prima operazione esterna
- k = 1 + 2K per la seconda e per la terza

Non essendo specificato nulla riguardo la frequenza con cui vengono richieste le operazioni esterne, assumiamo che siano equiprobabili ed otterremo

$T = \tau/3 + (1 + 2K)\tau/3 + (1 + 2K)\tau/3 = \tau/3 + (1 + 2K)2\tau/3 \approx 4K\tau/3$

Seconda versione: arbitraggio fair.

Si mantiene una variabile turno per dare precedenza, in caso di richiesta di operazione da parte di entrambe le unità Ua e Ub, all'unità che non ha richiesto l'ultima operazione eseguita.

Assumento che tutti i registri siano inizializzati a zero, utilizziamo TURNO per decidere a chi tocca in caso di richieste multiple: TURNO = 0 → precendeza a Ua, TURNO = 1 → precedenza a Ub.

Il microcodice è quasi identico a quello precendete, a parte il fatto che si testa anche TURNO e si assegna TURNO nelle varie fasi:

0. (RDY_1, RDY_2, Op, TURNO = 00-0) nop, 0;
   (=1-00) X → MEM[IND], reset RDY_1, set ACK_1, 1→TURNO, 0;
   (=1-10) 0 → I, 1→ TURNO, 1;
   (=-1-1) MEM[0] → MIN, 1 → I, 0→ TURNO, 2
1. (I0,zero(MEM[I]-X)=00) I+1 → I, 1;
   (=01) SHIFTR(MEM[I],13)→ MEM[I], I+1 → I, 1;
   (=1-) reset RDY_1, set ACK_1, 0;
2. (I0,segno(MIN-MEM[I]=00) I+1 → I, 2;

(=01) MEM[I] → MIN, I+1 → I, 2;

(=1-) MIN → OUTMIN, reset RDY_2, set ACK_2, 0

con valutazioni che portano al tempo medio di elaborazione che sono le stesse del caso precedente.