Si progetti una unità firmware U contenente una memoria M di 1K parole da 32 bit, in cui sono presenti N dati significativi (N>1); i dati significativi sono numeri interi positivi, e devono occupare sempre le prime N posizioni di M. U interagisce con due unità, U1 ed U2 : U1 richiede di eliminare da M il dato memorizzato nella parola di memoria di indirizzo IND (IND è fornito da U1 ad U, ed è compreso tra 0 e 1023) se tale dato è un multiplo di 256 (per semplicità si supponga che l'elemento indicato da U1 non sia l'ultimo dato significativo), mentre U2 richiede ad U di fornirgli il numero degli elementi presenti in memoria il cui valore è dispari. In caso di richieste contemporanee, viene data la precedenza alla richiesta di U1. Nel caso di eliminazione di un elemento, la porzione dei dati significativi va ricompattata in modo da soddisfare il vincolo che vuole i dati significativi sempre nelle prime posizioni della memoria interna.
Si progetti l'unità U

- fornendo il microprogramma utilizzato,
- determinando la lunghezza del ciclo di clock, (si supponga di avere a disposizione porte logiche ad 8 ingressi che si stabilizzano in 1 tp, e solamente due ALU che fanno addizioni e sottrazioni)
- calcolare il tempo necessario (in termini di τ) ad eseguire le due richieste, quella di U1 e quella di U2
- è richiesto che l'unità minimizzi il numero di microistruzioni necessarie all'esecuzione delle operazioni esterne.

Sono necessarie almeno 3 microistruzioni, perché ce ne vuole una per i controlli iniziali e le inizializzazioni, una per la richiesta di U1 ed una per la richiesta di U2. Si testano variabili di condizionamento complesse che potevano essere eliminate solo al prezzo di introdurre un'altra microistruzione per l'esecuzione delle operazioni esterne.
La memoria M deve essere una memoria a "doppia porta".
Il confronto tra due dati A e B si farà mediante il segno di A-B, e il test per sapere se un dato C è multiplo di 256 o meno si farà mediante l'OR degli 8 bit meno significativi di C (questo è 0 se e solo se C è multiplo di 256), chiamato OR8() nel seguito. Per sapere se un elemento di M è dispari o meno, basta guardare il suo bit meno significativo (che vale 1 se e solo se è dispari): nel seguito sarà indicato con UB(M[I]).

Il microcodice è il seguente:

0. (RDY1,RDY2,segno(UE-IND),OR8(M[IND])=0 0 - -) NOP, 0 //non ci sono richieste
(=1 - 1 -) reset RDY1, set ACK1, 0 // richiesta da U 1 ma dato non esiste
(=1 - 0 1) reset RDY1, set ACK1, 0 // richiesta da U 1 ma dato non multiplo
(=1 - 0 0) M[IND+1] →M[IND] , IND+1→I , 1 //richiesta da U 1 , inizio ciclo compattamento
(=0 1 - -) 0 →I , 0→A , 2 //richiesta da U 2 , inizio ciclo conteggio elementi dispari
1. (segno(UE-I)=0) M[I+1] →M[I], I+1→I , 1 // esistono elementi significativi, si compattamento
(=1) reset RDY1, set ACK1, UE-1→UE, 0 //fine elementi significativi

2. (segno(UE-I), UB(M[I])=0 0) I+1→I, 2 //elemento pari
(=0 1) I+1→I, A+1→A, 2 // elemento dispari
(=1 -) reset RDY2, set ACK2, A →RIS, 0 // fine operazioni, restituzione del risultato

La prima frase della microistruzione 1. viola le regole di Bernstein (scrivo in I mentre lo leggo). La 1. sopra è possibile se mettiamo un ritardo tra ALU e ingresso in I (ma è comunque rischioso). Altrimenti bisogna usare una microistruzione in più. Cioè, la 1. diventa:

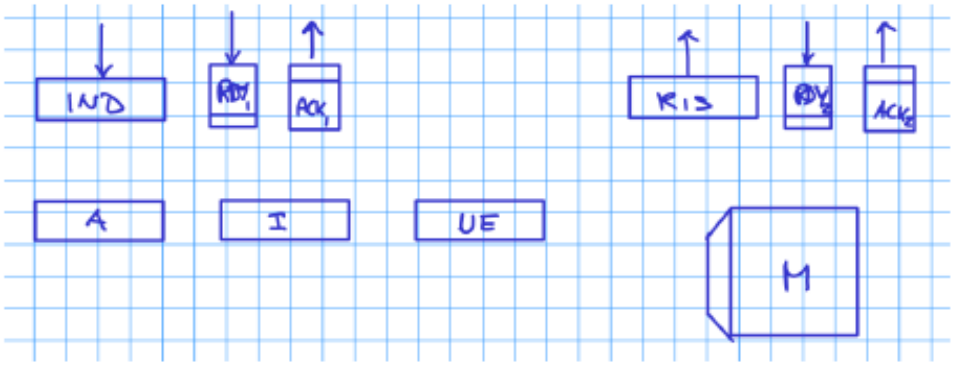
1. (segno(UE-I)=0) M[I+1] →M[I] , 3

3. I+1→I, 1

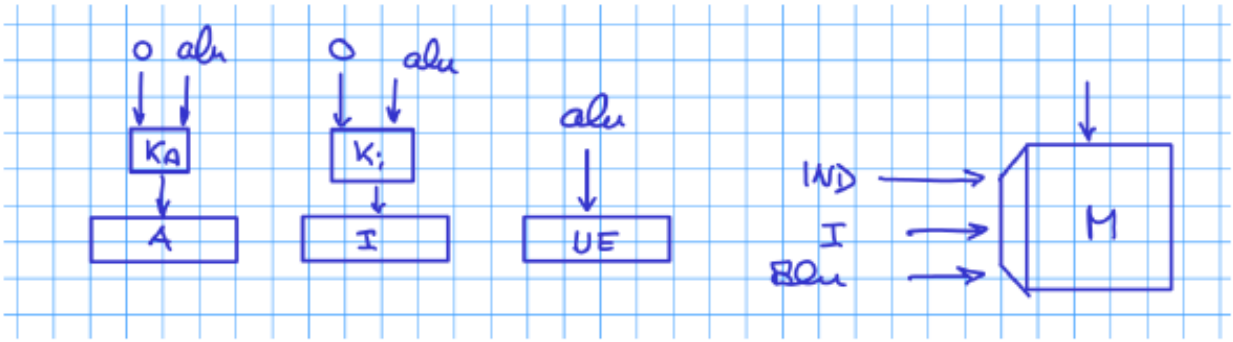
e le microistruzioni necessarie diventano 4. Sarebbe possibile avere 3 microistruzioni solamente essendo sicuri che il nuovo valore di I viene scritto nel registro solamente dopo che sono state effettuate le operazioni sulla memoria, ad esempio introducendo un ritardo sulla scrittura in I, ma è molto rischioso.

L’interfaccia dell’unità U con U1 è costituita da RDY1 e da un registro a 10 bit IND (in ingresso) e da ACK1 (in uscita), mentre quella con U2 è costituita da RDY2 (in ingresso), e da ACK2 e da un registro a 11 bit RIS (in uscita).
Si utilizzeranno i registri a 10 bit UE, che contiene l’indirizzo in M dell’ultimo elemento significativo, ed I usato per scorrere la memoria, ed un registro A di 11 bit, per contare il numero di dati di valore dispari.
La minimalità del numero di istruzioni eseguite è determinata anche dal fatto che i controlli necessari e le inizializzazioni dei cicli sono fatte tutte nelle frasi della 0. Si testano variabili di condizionamento complesse che potevano essere eliminate solo al prezzo di introdurre un’altra microistruzione per l’esecuzione delle operazioni esterne.

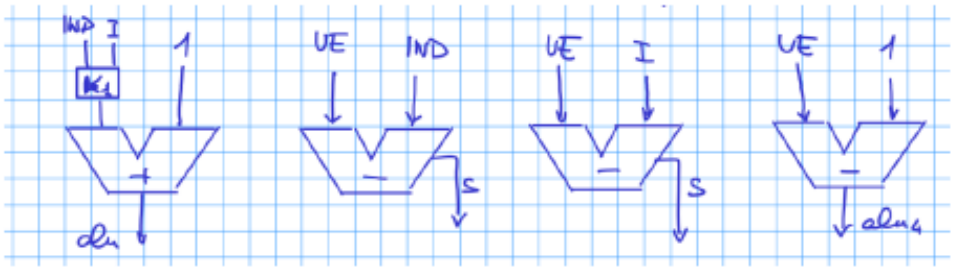
Risorse di stato



Ingressi delle risorse



Risorse di calcolo



La memoria M deve essere una memoria a “doppia porta”.

Parte operativa:
• TsigmaPO: tempo della microistruzione più lenta (la quarta frase della 0, ad esempio): max{ta +tALU, tALU +tK} (cioè tempo di accesso alla memoria+tempo della alu o tempo della alu + tempo del commutatore di I).
• TomegaPO: segno richiede tALU mentre OR8 si fa con una porta OR e quindi 1 tp ; le altre variabili di condizionamento richiedono tempo 0.

Parte controllo:
• abbiamo 4 microistruzioni e quindi 2 bit di stato. Inoltre, ci sono 11 frasi e 5 variabili di condizionamento, di cui al massimo 4 testate contemporaneamente. Quindi abbiamo un solo livello di porte AND (2+4 ingressi massimo per ogni porta AND) ed un solo livello di porte OR (1 ingresso per ogni frase che produce un "1" relativamente ad una uscita, cioè ad una colonna per un α o un β quindi 9 valori al massimo. Ma se abbiamo più di 5 valori ad 1, codifichiamo gli "0" e complementiamo).
Quindi TsigmaPC = TomegaPC = 2tp ;

Pertanto:
Tau = tALU + max {2tp ; ta +tALU +2tp } + delta = ta +2tALU +3tp (supponendo che ta +tALU> tALU +tK).

Il tempo T1 per eseguire la richiesta di U1 è (2K+2)τ (τ per eseguire la microistruzione 0 + K volte la prima frase della microistruzione 1 e la microistruzione 3 + la seconda frase della microistruzione 1); Il tempo T2 per eseguire la richiesta di U2 è (N+2)τ (microistruzione 0 + microistruzione 2).
Supponendo i dati in M distribuiti uniformemente, la probabilità che il dato richiesto da U1 vada eliminato è 1/256. Supponendo che il dato eventualmente da eliminare sia distribuito uniformemente in M, in media si deve compattare mezza memoria, cioè K=N/2. In media, quindi si ha per T1: 255/256 τ (dato non multiplo) + 1/256((N+2)τ).