Prova Finale di Reti Logiche

AA. 2019/2020

Stefano Dalla Longa, pID 10535602
Nicolò Brandolese, pID 10531144

Politecnico di Milano, scaglione prof. Fornaciari

Contents

1 Introduzione

This is the first section.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortisfacilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdietmi nec ante. Donec ullamcorper, felis non sodales...

2 Descrizione della macchina

L'implementazione si basa sulla scomposizione in tre sottoproblemi:

1. comunicazione con la RAM, intesa come l'insieme di operazioni di lettura e scrittura
2. computazione per determinare l'appartenenza dell'indirizzo base ad una working zone
3. gestione del flusso in base al risultato della computazione

La scelta di risoluzione è stata la creazione di una macchina a stati finiti, in particolare una macchina di Mealy, basata su tre macroprocessi:

- *state_register*:

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisissem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi necante...

3 Test Eseguiti

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisissem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi necante...

4 Limitazioni e ottimizzazioni possibili

4.1 Hardcoding

Gli unici valori hardcodati sono dovuti al formato di codifica del risultato (work in progress)

4.2 Segnale di reset

Incidentalmente, la macchina viene già inizializzata allo stato di *START_IDLE*, stato nel quale avviene il reset di tutti i segnali interni e dal quale esce solo tramite il segnale di *i_start*. Ne consegue che **non è necessario avviare la computazione con il segnale di** *i_reset*, ma l'utilizzo di *i_reset* è comunque gestito coerentemente con la specifica. Inoltre, *i_reset* resetta la macchina riportandone lo stato a *START_IDLE* indipendentemente dal valore di *i_start*.

4.3 Ottimizzazione dei dati nella RAM

La macchina è progettata per produrre il risultato corretto anche se gli indirizzi relativi alle working zone contenuti nella RAM non sono in ordine (vedasi la sezione 3 relativa ai test eseguiti). La macchina infatti esegue una scansione lineare di tutte le working zone, decidendo di volta in volta se richiedere la working zone successiva o fermarsi. Perciò, tra tutte le possibili computazioni il caso pessimo è quello in cui l'indirizzo base non appartiene a nessuna working zone, oppure quello in cui il base address appartiene alla working zone codificata in *RAM(7)*. Tuttavia, se venisse garantito che gli indirizzi delle working zone nella RAM siano in ordine crescente (decrescente), sarebbero possibili due modifiche:

- *Riduzione dei casi pessimi*: se l'indirizzo della working zone appena esaminata ha un valore più alto dell'indirizzo base, si può concludere che l'indirizzo base non appartiene a nessuna working zone, riducendo così il numero dei casi pessimi. Tale modifica al progetto è piuttosto modesta, dal momento che basterebbe modificare lo stato di *WZ_DECISION*.
- *Ricerca binaria*: anziché eseguire una ricerca lineare della working zone risulterebbe più efficiente eseguire una ricerca binaria. Tale strategia ridurrebbe l'intera complessità temporale asintotica della macchina: se n fosse il numero di working zone, si passerebbe da $T(n) \in \mathcal{O}(n)$ a $T(n) \in \mathcal{O}(\log n)$. Tuttavia, questa modifica è più onerosa di quella del punto precedente, dal momento che richiederebbe una modifica all'attuale metodo di scansione (affidato al semplice contatore di working zone *wz_counter*) e almeno un nuovo stato per il calcolo del prossimo indirizzo di memoria da richiedere.