Azzolini Riccardo 2019-02-27

Modelli di calcolo

1 Modello RASP

Il **modello RASP** (**Random Access Stored Program**) mantiene il programma in memoria e permette di *modificare le istruzioni durante l'esecuzione* (al contrario del modello RAM).

Il set di istruzioni è identico a quello del modello RAM, ma *senza l'indirizzamento indiretto*.

Il programma viene caricato in memoria associando a ogni istruzione due registri:

1. il primo contiene l'opcode, che codifica l'istruzione e il tipo di operando;
2. il secondo contiene l'indirizzo (operando o etichetta).

Opcode	Istruzione
1	`LOAD ()`
2	`LOAD =()`
3	`STORE ()`
4	`ADD ()`
5	`ADD =()`
6	`SUB ()`
7	`SUB =()`
8	`MULT ()`
9	`MULT =()`
10	`DIV ()`
11	`DIV =()`
12	`READ ()`
13	`WRITE ()`
14	`WRITE =()`
15	`JUMP ()`
16	`JGTZ ()`
17	`JZERO ()`
18	`JBLANK ()`
19	`HALT`

L'esecuzione presenta alcune differenze rispetto alla macchina RAM:

- la memoria *contiene inizialmente il programma* in una sequenza prefissata di registri;
- il *location counter* viene incrementato di 2 invece che di 1 (perché ogni istruzione occupa 2 registri).

I concetti di

- *stato*
- *computazione*
- *funzione* calcolata da un programma
- *tempo e spazio*, con i criteri *uniforme* e *logaritmico*

e le relative notazioni sono invece definiti analogamente alla macchina RAM.

1.1 Esempio di programma

Calcolo del massimo di n interi:

$i+s$	R_{i+s}	R_{i+s+1}		
$0+s$	12	1	`READ`	1
$2+s$	18	$18+s$	`JBLANK`	$18+s$
$4+s$	1	1	`LOAD`	1
$6+s$	12	2	`READ`	2
$8+s$	6	2	`SUB`	2
$10+s$	16	$2+s$	`JGTZ`	$2+s$
$12+s$	1	2	`LOAD`	2
$14+s$	3	1	`STORE`	1
$16+s$	15	$2+s$	`JUMP`	$2+s$
$18+s$	13	1	`WRITE`	1
$20+s$	19	0	`HALT`	

Osservazione: Gli indirizzi delle istruzioni sono relativi all'*indirizzo di impianto in memoria* s.

2 Equivalenza tra RAM e RASP

È possibile dimostrare che i due insiemi

$$\mathcal{F}_{RAM} = \{F_P \mid P \text{ programma RAM}\}$$
$$\mathcal{F}_{RASP} = \{F_P \mid P \text{ prograspma RASP}\}$$

delle funzioni calcolate, rispettivamente, da programmi RAM e RASP, sono equivalenti, $\mathcal{F}_{RAM} \equiv \mathcal{F}_{RASP}$, mostrando che una macchina RASP può simulare qualunque programma RAM, e viceversa.

2.1 Da RAM a RASP

Teorema: Per ogni programma RAM Φ, esiste un programma RASP Ψ tale che, $\forall n \in \mathbb{N}$, $\underline{x} \in \mathbb{Z}^n$:

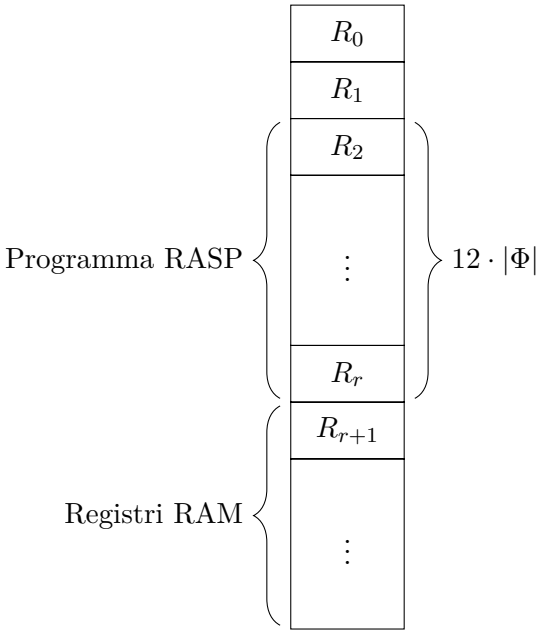
- $F_\Phi(\underline{x}) = F_\Psi(\underline{x})$, cioè ai due programmi corrisponde la stessa funzione calcolata;
- $T_\Psi(\underline{x}) \leq 6 \cdot T_\Phi(\underline{x})$.

Dimostrazione: Ogni istruzione RAM può essere tradotta in una sequenza di al più 6 istruzioni RASP:

- se $op \neq *i$, si può effettuare la traduzione diretta;
- se invece $op = *i$, è necessario simulare l'indirizzamento indiretto, sfruttando la possibilità di modificare il codice durante l'esecuzione.

Per la simulazione, i registri vengono utilizzati come segue:

- R_0 è l'accumulatore;
- R_1 è utilizzato come backup dell'accumulatore;
- i registri da R_2 a R_r, con $r = 12 \cdot |\Phi| + 1$, contengono il programma RASP (che richiede al massimo $6 \cdot |\Phi|$ istruzioni, e quindi $2 \cdot 6 \cdot |\Phi| = 12 \cdot |\Phi|$ registri);
- i registri da R_{r+1} in poi corrispondono ai registri RAM (ogni registro RAM R_k, con $k \geq 1$, viene "shiftato" di r e diventa R_{r+k}).



Il codice RASP che simula, ad esempio, l'istruzione RAM `MULT *k`, è:

$M+0$	3	`STORE`	`1`	Salva il valore dell'accumulatore R_0
$M+1$	1			
$M+2$	1	`LOAD`	$r+k$	Carica l'indirizzo a cui accedere
$M+3$	$r+k$			dal registro RAM k (RASP $r+k$)
$M+4$	5	`ADD`	`=`r	Aggiunge l'offset r all'indirizzo
$M+5$	r			
$M+6$	3	`STORE`	$M+11$	Scrive l'indirizzo calcolato come
$M+7$	$M+11$			operando dell'istruzione `MULT`
$M+8$	1	`LOAD`	`1`	Ripristina il valore di R_0
$M+9$	1			
$M+10$	8	`MULT`	$S(r+k)+r$	Esegue la moltiplicazione tra R_0
$M+11$	-			e il registro all'indirizzo calcolato

2.2 Da RAM a RASP con costo logaritmico

Teorema: Per ogni programma RAM Φ, esistono un programma RASP Ψ e una costante $C > 0$ tali che, $\forall n \in \mathbb{N}$, $\underline{x} \in \mathbb{Z}^n$:

- $F_\Phi(\underline{x}) = F_\Psi(\underline{x})$;
- $T^l_\Psi(\underline{x}) \leq C \cdot T^l_\Phi(\underline{x})$.

2.3 Da RASP a RAM

Teorema: Per ogni programma RASP Ψ, esistono un programma RAM Φ e due costanti $C_1, C_2 > 0$ tali che, $\forall n \in \mathbb{N}$, $\underline{x} \in \mathbb{Z}^n$:

- $F_\Phi(\underline{x}) = F_\Psi(\underline{x})$;
- $T_\Phi(\underline{x}) \leq C_1 \cdot T_\Psi(\underline{x})$;
- $T^l_\Phi(\underline{x}) \leq C_2 \cdot T^l_\Psi(\underline{x})$;

Una macchina RAM può infatti simulare programmi RASP, grazie all'indirizzamento indiretto.

3 Calcolabilità

Gli insiemi $\mathcal{F}_{RAM}$ e $\mathcal{F}_{RASP}$ sono equivalenti agli insiemi delle funzioni calcolabili da molti altri linguaggi.

La classe di funzioni $\mathcal{F} = \mathcal{F}_{RAM}$ è chiamata classe delle **funzioni ricorsive parziali**.

Essa è una classe molto *robusta* rispetto alle tecnologie, che formalizza il concetto intuitivo di **calcolabilità** (**Tesi di Church-Turing**).

4 Calcolabilità effettiva

Siano

$$\mathcal{F}_{RAM}(f) = \{F_P \in \mathcal{F}_{RAM} \mid T^l_P(n) = O(f(n))\}$$
$$\mathcal{F}_{RASP}(f) = \{F_P \in \mathcal{F}_{RASP} \mid T^l_P(n) = O(f(n))\}$$

gli insiemi delle funzioni calcolabili, rispettivamente, da programmi RAM e RASP con complessità $O(f(n))$ nel *caso peggiore*. Si ha che $\mathcal{F}_{RAM}(f) \equiv \mathcal{F}_{RASP}(f)$, ma, se si sceglie una funzione f precisa, questa proprietà non vale in generale con altri linguaggi (perché il costo della simulazione non è sempre un fattore costante, e quindi non è garantito che lo si possa trascurare).

Se si considera invece l'insieme

$$\mathbf{P}_{RAM} = \left\{F_P \in \mathcal{F}_{RAM} \mid \exists k,\, T^l_P(n) = O\left(n^k\right)\right\}$$

è possibile definire la classe $\mathbf{P} = \mathbf{P}_{RAM}$ dei **problemi risolubili in tempo polinomiale**, la quale rimane invariata anche per altri linguaggi.

Per la sua proprietà di invarianza, questa classe consente di formalizzare il concetto intuitivo di **calcolabilità effettiva** (**Tesi di Church estesa**).