Macchine a registri e linguaggi imperativi

Il linguaggio delle macchine a registri è basato su istruzioni fondamentali in qualsiasi linguaggio imperativo.
I risultati validi per le RAM si estendono a macchine astratte programmate con linguaggi piú complessi.

Ad esempio, le istruzioni del linguaggio Assembler INTEL (con istruzioni a due indirizzi) sono del tipo:
✓*trasferimento tra registri e memoria o tra registri di interfaccia e memoria* (MOV <locazione><registro>, IN <porta><registro>,...);
✓*aritmetiche* (ADD <locazione><registro>, SUB <locazione><registro>);
✓*logiche* (AND <locazione><registro>) o di *rotazione* (ADL <registro>);
✓di *controllo*;

Possiamo definire le funzioni calcolabili con questo linguaggio, e sembra chiaro che esse sono tutte calcolabili con macchine a registri.

Se consideriamo linguaggi *imperativi* ad alto livello (Pascal, Fortran, ...) possiamo definire le funzioni da essi calcolabili; i compilatori garantiscono che possiamo tradurre un programma ad alto livello in assembler.
Quindi tutte le funzioni calcolabili con linguaggi imperativi ad alto livello lo sono anche con le RAM (Church-Turing).

Macchina a registri elementare ($MREL$)

La *memoria* di una $MREL$ consiste in un numero finito di registri, ognuno dei quali contiene un intero grande a piacere, accessibili attraverso il loro numero d'ordine.

Il *contatore delle istruzioni* (CI) contiene il numero d'ordine della prossima istruzione da eseguire.

La macchina non prevede accumulatore e nastri di ingresso e uscita.

I valori degli argomenti della funzione da calcolare $x_1, \ldots, x_n$ sono forniti nei registri $R_1, \ldots, R_n$.
Il valore della funzione calcolato da programma $f(x_1, \ldots, x_n)$ è in R_1.
Tutti gli altri registri contengono il valore 0 all'inizio dell'esecuzione del programma.

Il *programma* per MREL consiste in un insieme di *istruzioni*, che agiscono su *operandi* contenuti nei registri; tre soli tipi: *incremento*, *decremento*, *salto condizionato*.

<programma> ::= <istruzione> {<istruzione>}
<istruzione> ::= R<intero positivo>:= R <intero positivo>+1
::= R<intero positivo>:= R <intero positivo>−1
::= IF R<intero positivo>= 0 THEN GOTO <etichetta
<etichetta> ::=< intero>

Per convenzione il programma termina se si effettua un salto all'istruzione 0. In ogni altra circostanza si dice che il calcolo non è definito.

Esempio 5 *Definire la $MREL$ per il calcolo della somma di due interi $f(x,y) = x + y$:*

$IF\ R2 = 0\ THEN\ GOTO\ 0$
$R2 := R2 - 1$
$R1 := R1 + 1$
$IF\ R3 = 0\ THEN\ GOTO\ 1$

Nota bene: Il fatto che il registro R3 sia inizializzato a zero ci consente di simulare un salto incondizionato.

Esercizio: calcolare la funzione uguaglianza $f(x,y) = 1$ se $x = y$, 0 altrimenti.

Possiamo simulare tutte le istruzioni di una RAM:

```
LOAD    1
ADD     2
STORE   1
```

si realizza nel seguente modo:
IF R2 = 0 THEN GO TO 6
R2 := R2 - 1
R1 := R1 + 1
R3 := R3 + 1
IF R4 = 0 THEN GOTO 1
IF R3 = 0 THEN GOTO 0
R3 := R3 - 1
R2 := R2 + 1
IF R4 = 0 THEN GOTO 6

Teorema. Ogni funzione $f : \mathbb{N}^n \longmapsto \mathbb{N}$ calcolabile con una RAM è calcolabile con una $MREL$ e viceversa.

Funzioni ricorsive

La caratterizzazione delle funzioni calcolabili è stata data attraverso delle *classi di macchine*. Una caratterizzazione matematica delle stesse funzioni è rappresentata dalle *funzioni ricorsive*.

Idea: assemblare una funzione mediante delle funzioni base estremamente elementari e calcolabili, più alcuni *operatori*.

Lambda-notation: permette di specificare l'insieme delle variabili su cui la funzione è definita. Una funzione di n variabili $x_1, \ldots, x_n$, associata ad un'espressione algebrica $f(x_1, \ldots, x_n)$ in forma chiusa, è descritta come

$$\lambda x_1 \lambda x_2 \ldots \lambda x_n . f(x_1, \ldots, x_n).$$

Ad esempio, la corrispondenza che ad ogni coppia di valori associa la somma si indica con $\lambda x_1 \lambda x_2 . x_1 + x_2$.

$\mathcal{F} = \{f | f : \mathbb{N}^n \longmapsto \mathbb{N}, n \geq 0\}$ è l'insieme di tutte le funzioni intere.
$f^{(n)}$ indica che la funzione f ha n argomenti.
$n = 0 \Longrightarrow f$ è una costante.

In $\mathcal{F}$ individuiamo la classe di funzioni che corrispondono al concetto di calcolabile, nel modo seguente:
definiamo *funzioni di base* ovviamente calcolabili e degli *operatori* (dei *costrutti*) con i quali definire nuove funzioni a partire da quelle giá disponibili.

Definizione 6 *Le* funzioni base *sono le seguenti:*

1. $0^{(n)} = \lambda x_1, \ldots \lambda x_n . 0$, *con* $n \geq 0$ *(funzioni zero);*
2. $S = \lambda x . x + 1$ *(funzione successore);*
3. $U_i^{(n)} = \lambda x_1, \ldots \lambda x_n . x_i$ *(funzioni selettrici o identitá).*

Definizione 7 *La classe delle* funzioni ricorsive primitive $\mathcal{P}$ *è la piú piccola classe che contiene le funzioni base e inoltre:*

1. *se le funzioni* $h^{(m)}, g_1^{(n)}, \ldots, g_m^{(n)}$ *sono ricorsive primitive allora anche la funzione* $f^{(n)}$ *definita per* composizione*:*

$$f(x_1, \ldots, x_n) = h(g_1(x_1, \ldots, x_n), \ldots, g_n(x_1, \ldots, x_n))$$

è ricorsiva primitiva;

2. *se le funzioni* $h^{(n+2)}, g^{(n)}$ *sono ricorsive primitive allora anche la funzione* $f^{(n+1)}$ *definita per* ricorsione primitiva*:*

$$\begin{cases} f(x_1, \ldots, x_n, 0) = g(x_1, \ldots, x_n), \\ f(x_1, \ldots, x_n, y+1) = h(x_1, \ldots, x_n, y, f(x_1, \ldots, x_n, y)) \end{cases}$$

è ricorsiva primitiva.

Esempio 8 *Le seguenti funzioni sono ricorsive primitive:*

$incr2 = \lambda x \lambda y.x + 2$. *Tale funzione si esprime utilizzando le funzioni base e la composizione come* $incr2 = S(S(U_1^2))$.

$somma = \lambda x \lambda y.x + y$. *Tale funzione si esprime utilizzando la ricorsione primitiva come*

$$\begin{cases} somma(x, 0) = x \\ somma(x, y+1) = somma(x, y) + 1. \end{cases}$$

La funzione g *coincide con la selettrice* U_1^1*; la funzione* h *ha tre argomenti ed è* $\lambda x_1 \lambda x_2 \lambda x_3 . S(U_3^3(x_1, x_2, x_3))$.

Esempio 9 $prodotto = \lambda x \lambda y.x * y$. *Tale funzione si esprime utilizzando la ricorsione primitiva come*

$$\begin{cases} prodotto(x, 0) = 0 \\ prodotto(x, y + 1) = x + prodotto(x, y). \end{cases}$$

La funzione g coincide con la funzione zero 0^1*; la funzione h ha tre argomenti ed è* $\lambda x_1 \lambda x_2 \lambda x_3 . U_1^3(x_1, x_2, x_3) + U_3^3(x_1, x_2, x_3)$.

Esempio 10 *Anche la funzione:*
$predecessore = \lambda x.x - 1$
è ricorsiva primitiva. Infatti:
$0 - 1$ *è 0*
$(x + 1) - 1 = x$ *per ricorsione*

Nota: Quando $n = 0$ la definizione di una funzione in termini ricorsivi prende la forma:

$$\begin{cases} f(0) = c \\ f(y + 1) = h(y, f(y)). \end{cases}$$

La definizione di una funzione ricorsiva ha spesso una struttura basata sul seguente schema:

$$\begin{cases} f(0) = c \\ f(y + 1) = h(g(y), f(y)). \end{cases}$$

dove le funzioni g e h sono ricorsive primitive, e c è una costante.

Esercizio:
Usando la def. di predecessore dimostrare che la cut-off subtraction così definita

$$x \dot{-} y = \begin{cases} 0 \text{ se } x < y \\ x - y \text{ altrimenti} \end{cases}$$

è ricorsiva primitiva.

Si dimostrare che è ricorsiva primitiva la funzione $sg = \lambda x.sg(x)$ così definita:

$$sg(x) = \begin{cases} 0 \text{ se } x = 0 \\ 1 \text{ se } x \neq 0 \end{cases}$$

Infatti la funzione è esprimibile ricorsivamente come segue:

$$\begin{cases} sg(y) = 0 \text{ se } y = 0 \\ sg(y+1) = 1 \text{ altrimenti} \end{cases}$$

In questo caso, $n = 0$, $g = 0^0$ e $h = \lambda y \lambda z.S(0^1(y))$.

Più in generale, se p è ricorsiva primitiva con valori 0 e 1, e g e h sono ricorsive primitive, la funzione

$$f(x) = \begin{cases} h(x) \text{ se } p(x) = 1 \\ g(x) \text{ se } p(x) = 0 \end{cases}$$

è ricorsiva primitiva. Infatti basta osservare che $f(x)$ è esprimibile come composizione di funzioni ricorsive primitive:

$f(x) = h(x) * p(x) + (1 - p(x)) * g(x)$

Esercizi:
Dimostrare che:

- $\lambda x \lambda y.|x - y|$ è ricorsiva primitiva (suggerimento: sfruttare la def. di cut-off subtraction)
- $\lambda x \lambda y.min(x, y)$ è ricorsiva primitiva
- $\lambda x \lambda y.max(x, y)$ è ricorsiva primitiva
- $\lambda x.x!$ è ricorsiva primitiva
- $\lambda x.2^x$ è ricorsiva primitiva.

Si dimostra che anche la funzione resto (o modulo) è ricorsiva primitiva. Infatti:

$$resto(x, y+1) = \begin{cases} resto(x,y) + 1 \text{ se } resto(x,y) + 1 \neq x \\ o \text{ se } resto(x,y) + 1 = x \end{cases}$$

Ciò porta alla seguente definizione per ricorsione primitiva:

$$\begin{cases} resto(x, 0) = 0 \\ resto(x, y+1) = (resto(x,y) + 1) * sg(|x - (resto(x,y) + 1)|) \end{cases}$$

Esercizio: Dimostrare che la funzione

$$div(x,y) = \begin{cases} 1 \text{ se } x \text{ divide } y \\ 0 \text{ altrimenti} \end{cases}$$

è ricorsiva primitiva (Suggerimento: utilizzare le funzioni sn e resto)

Esercizio: sia $f(x,z)$ una funzione ricorsiva primitiva. Dimostrare che la funzione "sommatoria limitata"

$S(x,y) = \sum_{z<y} f(x,z)$

è ricorsiva primitiva
(Osservazione: $\sum_{z<0} f(x,z) = 0$)

Esercizio: sia $f(x,z)$ una funzione ricorsiva primitiva. Dimostrare che la funzione "produttoria limitata"

$P(x,y) = \prod_{z<y} f(x,z)$

è ricorsiva primitiva
(Osservazione: $\prod_{z<0} f(x,z) = 1$)

Sfruttando la definizione di sommatoria limitata si pu dimostrare che anche la funzione $\lambda x.D(x)$ che restituisce il numero di divisori di x è ricorsiva primitiva (si assume $D(0) = 1$). Infatti essa è definibile come $\sum_{y \leq x} div(x,y)$.

Quindi si dimostra facilmente che $\begin{cases} Pr(x) = 1 \text{ se } x \text{ è primo} \\ Pr(x) = 0 \text{ se } x \text{ non è primo.} \end{cases}$
è una funzione ricorsiva primitiva. Infatti:
$Pr(x) = 1 - sg(|D(x) - 2|)$ (i soli divisori di x sono 1 e x stesso).

Si dimostra infine che la funzione $\lambda n.p_n$ restituisce lo n-esimo numero primo è ricorsiva primitiva. Infatti:

$$\begin{cases} p_0 = 0 \\ p_{n+1} = \sum_{v \leq (p_n!+1)} (\prod_{u \leq v} sg((u > p_n) * Pr(u))) \end{cases}$$

dove $\lambda x \lambda y.x > y$ è da intendersi come il predicato "maggiore stretto".

Esistono funzioni definite ricorsivamente, ma che non rientrano nella classe $\mathcal{P}$? Un esempio è la funzione di tre argomenti cosí definita:

$f(x,y,0) = x + y$
$f(x,y,1) = x \times y$
$f(x,y,2) = x^y$
e, per $n \geq 2$,

$$f(x,y,n+1) = \begin{cases} 1 & \text{se } y = 0, \\ \underbrace{f(x, f(x, \ldots, f}(x,0,n), \ldots, n), n) & \text{altrimenti.} \end{cases}$$

oppure, in termini finiti:
$f(x,0,0) = x$
$f(x,y+1,0) = f(x,y,0) + 1$
$f(x,0,1) = 0$
$f(x,0,n+2) = 1$
$f(x,y+1,n+1) = f(x, f(x,y,n+1), n).$

La funzione sopra definita (funzione di Ackermann),
non è ricorsiva primitiva, pur essendo definita in termini ricorsivi.

Introduciamo un nuovo operatore per estendere la classe delle funzioni ricorsive primitive (e il concetto di calcolabilitá).

Definizione 11 *Data una funzione* $g^{(n+1)}$*, la funzione* $f^{(n)}$ *si dice definita mediante l'operatore di* minimalizzazione μ *se* $f(x_1,\ldots,x_n)$ *assume come valore il minimo* t *tale che* $g(x_1,\ldots,x_n,t) = 0$*; se* $g(x_1,\ldots,x_n,t) \neq 0$ *per tutti i valori di* t*, allora la funzione* $f(x_1,\ldots,x_n)$ *è indefinita.*

Indichiamo f con la notazione $f = \lambda x_1,\ldots,\lambda x_n.\mu t(g(x_1,\ldots,x_n,t) = 0)$.

Definizione 12 *La classe delle* funzioni ricorsive *(dette anche* ricorsive generali *o* ricorsive parziali*)* $\mathcal{R}$ *è la piú piccola classe che contiene le funzioni base, ed è chiusa rispetto a composizione, ricorsione primitiva e minimalizzazione.*

Esempio 13 *La funzione* $\lambda n.\sqrt{n}$ *così definita:*

$$\sqrt{n} = \begin{cases} x & \text{se } n = x^2, \\ indefinito & \text{altrimenti} \end{cases}$$

è ricorsiva in quanto definibile come $\sqrt{n} = \mu t(|n - t^2| = 0)$

Esempio 14 *La funzione* $\lambda n.\sqrt{n}$ *può anche essere definita come funzione totale come segue:*
$\sqrt{n} = \mu t((n+1) - (t+1) \times (t+1) = 0)$

Funzioni ricorsive e linguaggi imperativi

Le funzioni ricorsive sono calcolabili con i modelli di calcolo giá visti. Ad esempio, appare banale dimostrare che il linguaggio C calcola le funzioni ricorsive e, quindi, le RAM calcolano le funzioni ricorsive.

Viceversa, ogni funzione calcolata da una RAM è ricorsiva?

Se questo è vero, le funzioni calcolabili con macchine a registri e i linguaggi di programmazione imperativi coincidono con le funzioni ricorsive.

Introduciamo le funzioni di *creazione di n-ple* che associano biunivocamente un intero ad una n-pla di interi:
$P_n = \lambda x_1, \ldots, \lambda x_n[2^{x_1} \cdot \ldots \cdot p_n^{x_n}]$, dove p_n è l'n-esimo numero primo ($p_1 = 2$).

Queste funzioni sono ricorsive primitive e saranno utilizzate per rappresentare lo stato di una $MREL$ descritto da n registri.

Notazione: $P_n(x_1, \ldots, x_n) \equiv \langle x_1, \ldots, x_n \rangle$.

Introduciamo inoltre le funzioni di *estrazioni di elementi da una n-pla*:
$K_i = \lambda z[$ esponente di p_i nella decomposizione in fattori primi di $z]$.

Queste funzioni prendono in input un numero intero, lo *fattorizzano* e restituiscono l'esponente di un fattore. Servono a estrarre il valore di un registro di uno stato rappresentato da $P_n(x_1, \ldots, x_n)$.

Notazione: $K_i(z) \equiv (z)_i$.

$M(t) =$ if $t = 0$ then 0 else $\max\{n | p_n$ divide $t\}$.
Valgono le proprietà:
$K_i(P_n(x_1, \ldots, x_n)) = x_i$ se $i \leq n$, 0 altrimenti
$P_n(K_1(z), \ldots, K_n(z)) = z$ se $n = M(z)$

Definizione 15 *Uno stato di una macchina a registri elementare con programma π, che utilizza m registri ed è costituito da l istruzioni, è rappresentato da un intero:*

$$s = \langle i, \langle r_1, \ldots, r_m \rangle \rangle$$

dove i ($0 \leq i \leq l$) è il numero d'ordine dell'istruzione da eseguire, e $r_1, \ldots, r_m$ sono gli interi contenuti nei registri.

Esempio 16

Codifica di uno stato

$$
\begin{array}{lllll}
s_1 & \langle 1, \langle 3,2,0 \rangle \rangle & = & 2 \cdot 3^{2^3 3^2} & = & 2 \cdot 3^{72} \\
s_2 & \langle 2, \langle 3,2,0 \rangle \rangle & = & 2^2 \cdot 3^{72} & & \\
s_3 & \langle 3, \langle 3,1,0 \rangle \rangle & = & 2^3 \cdot 3^{2^3 3} & = & 2^3 \cdot 3^{24} \\
s_4 & \langle 4, \langle 4,1,0 \rangle \rangle & = & 2^4 \cdot 3^{2^4 3} & = & 2^4 \cdot 3^{48} \\
s_5 & \langle 1, \langle 4,1,0 \rangle \rangle & = & 2 \cdot 3^{48} & & \\
s_6 & \langle 2, \langle 4,1,0 \rangle \rangle & = & 2^2 \cdot 3^{48} & & \\
s_7 & \langle 3, \langle 4,0,0 \rangle \rangle & = & 2^3 \cdot 3^{2^4} & = & 2^3 \cdot 3^{16} \\
s_8 & \langle 4, \langle 5,0,0 \rangle \rangle & = & 2^4 \cdot 3^{2^5} & = & 2^4 \cdot 3^{32} \\
s_9 & \langle 1, \langle 5,0,0 \rangle \rangle & = & 2 \cdot 3^{32} & & \\
s_{10} & \langle 0, \langle 5,0,0 \rangle \rangle & = & 2^{32} \quad . & &
\end{array}
$$