RAM машини

Иво Стратев

14 ноември 2020 г.

Увод

Няма да дефинираме формално понятието RAM машина, нито ще въвеждаме формализъм за работа с тях. Защото единствено ще пишем програми за такива машини. Реално всяка RAM машина донякъде е теоретичен аналог на съвременните компютри.

Инструкции

В програмите, които ще пишем ще позволим само четири вида инструкции/команди, които по идея са аналог на базови инструкци част от, който да е асемблер. Инструкциите са следните:

- $\mathsf{Z}(\mathfrak{n})$ - нулирай стойността на клетка $\mathfrak{n}$ от паметта и изпълни следващата инструкция.
- $\mathsf{S}(\mathfrak{n})$ - увеличи стойността на клетка $\mathfrak{n}$ от паметта с единица и изпълни следващата инструкция.
- $\mathsf{T}(\mathfrak{n}, \mathsf{k})$ - копирай стойността на клетка $\mathfrak{n}$ в клетка k от паметта и изпълни следващата инструкция.
- $\mathsf{J}(\mathfrak{n}, \mathsf{k}, \mathfrak{l})$ - ако стойностите на клетки $\mathfrak{n}$ и k съвпадат премини към изпълнението на $\mathfrak{l}$-тата инструкция на програмата в противен случай изпълни следващата инструкция.

RAM е съкращение за Random access memory, което се превежда на български като памет с произволен достъп. Тоест директно можем да четем и променяме стойността на всяка клетка от паметта. Докато при машините на Тюринг достъпа до паметта е последователен - имаме глава, която се движи наляво и надясно. Въпреки произволният достъп до паметта не получаваме по-мощен изчислителен модел. Действието на всяка RAM машина може да бъде симулирано от машина на Тюринг и обратното.

Програми

С всяка RAM машина чрез нейната програма бива изчислявана някоя частична функция над естествени числа. Приемаме, че в началото на изпълнение на програмата на една RAM машина в клетки от 1 до k са записани входните стойности, а в клетка 0 е резултатната стойност ако машината успее да завърши достигайки до празен ред от програмата.

Безопасно вадене на единица

Искаме да напишем програма за машина, която да изчисли фунцкията $n \mapsto n \dot{-} 1$, където

$$n \dot{-} 1 = \begin{cases} 0 & , n = 0 \\ n - 1 & , n > 0 \end{cases}$$

Стойността на входа ще бъде в клетка 1, изхода трябва да е в клетка 0, за това ще ползваме клетка 2 като помощтен брояч. Идеята е следната пръво ще запишем 0 в клетка 0 след това ще направим проверка дали стойността в клетка 1 е нула. Ако не е, първо ще увеличаваме стойността в клетка 2 и ще я сравняваме с тази в 1, ако не са равни, то тя е по-малка и ще увеличаваме стойността в клетка 0 и след това ще се връщаме на проверката. Така реално ще имаме един цикъл, при който стойността в клетка 2 винаги е с единица по-голяма от тази в клетка 0.

$$\begin{aligned} 1 &: Z(0) \\ 2 &: J(0, 1, \_) \\ 3 &: Z(2) \\ 4 &: S(2) \\ 5 &: J(1, 2, \_) \\ 6 &: S(0) \\ 7 &: J(0, 0, \_) \end{aligned}$$

Подчертавките са защото докато пишем програмата е по-лесно да съобразяваме алгоритъма, а след това редовете, на които трябва да се направят скоковете, така че алгоритъма да е коректен. След като имаме всички инструкции лесно

се съобразява как да попълним празните места.

$$\begin{aligned}
1 &: Z(0)\\
2 &: J(0,1,8)\\
3 &: Z(2)\\
4 &: S(2)\\
5 &: J(1,2,8)\\
6 &: S(0)\\
7 &: J(0,0,4)
\end{aligned}$$

Сега понеже командите не са особено лесни за четене ще въведем означения, които да са по-удобни.

- Вместо $Z(n)$ ще пишем $X_n := 0$.
- Вместо $S(n)$ ще пишем $X_n := X_n + 1$.
- Вместо $T(n,k)$ ще пишем $X_k := X_n$.
- Вместо $J(n,k,l)$ ще пишем $\mathsf{if}\ X_n = X_k\ \mathsf{then\ goto}\ l$.

Така горната програма става:

$$\begin{aligned}
1 &: X_0 := 0\\
2 &: \mathsf{if}\ X_0 = X_1\ \mathsf{then\ goto}\ 8\\
3 &: X_2 := 0\\
4 &: X_2 := X_2 + 1\\
5 &: \mathsf{if}\ X_1 = X_2\ \mathsf{then\ goto}\ 8\\
6 &: X_0 := X_0 + 1\\
7 &: \mathsf{if}\ X_0 = X_0\ \mathsf{then\ goto}\ 4
\end{aligned}$$

Отсечена разлика

Искаме да напишем програма за RAM машина, която да изчислява функцията $n, k \mapsto n \dot{-} k$, където

$$n \dot{-} k = \begin{cases} 0 & , n < k \\ n - k & , n \geqslant k \end{cases}$$

В кода на програмата ще използваме идеята на кода, от предната програма, защото забележете, че например $2 \dot{-} 4 = 0$, което е същото като

$$(((2 \dot{-} 1) \dot{-} 1) \dot{-} 1) \dot{-} 1$$

Тоест можем един вид да итерираме безопасното вадене на единица. Съобразяваме, че в X_0 трябва да е резултата, а входните променливи са X_1 и X_2. Следователно можем да ползваме X_3 като брояч за итерациите, а X_4, X_5 и X_6 заваденето на единица. Като на всяка итерация X_5 ще има стойност $X_1 \dot{-} X_3$. Така когато $X_3 = X_2$ е истина, то $X_5 = X_1 \dot{-} X_2$ ще е истина и значи накрая ще трябва да направим присвояването $X_0 := X_5$.

$$
\begin{aligned}
&1 \ : \ X_3 := 0\\
&2 \ : \ X_5 := X_1\\
&3 \ : \ \text{if } X_3 = X_2 \text{ then goto } 14\\
&1+3 \ : \ X_{0+4} := 0\\
&2+3 \ : \ \text{if } X_{0+4} = X_{1+4} \text{ then goto } 8+3\\
&3+3 \ : \ X_{2+4} := 0\\
&4+3 \ : \ X_{2+4} := X_{2+4} + 1\\
&5+3 \ : \ \text{if } X_{1+4} = X_{2+4} \text{ then goto } 8+3\\
&6+3 \ : \ X_{0+4} := X_{0+4} + 1\\
&7+3 \ : \ \text{if } X_{0+4} = X_{0+4} \text{ then goto } 2+3\\
&11 \ : \ X_5 := X_4\\
&12 \ : \ X_3 := X_3 + 1\\
&13 \ : \ \text{if } X_3 = X_3 \text{ then goto } 3\\
&14 \ : \ X_0 := X_5
\end{aligned}
$$

Буквално "наляхме" кода на програмата за безопасно вадене на единица, като сме направили необходимите отместванията. Бихме могли този код да го съкратим като $X_4 := X_5 \dot{-} 1$ и тогава да получим по-лесна за четене и разбиране програма.

$$
\begin{aligned}
&1 \ : \ X_3 := 0\\
&2 \ : \ X_5 := X_1\\
&3 \ : \ \text{if } X_3 = X_2 \text{ then goto } 8\\
&4 \ : \ X_4 := X_5 \dot{-} 1\\
&5 \ : \ X_5 := X_4\\
&6 \ : \ X_3 := X_3 + 1\\
&7 \ : \ \text{if } X_3 = X_3 \text{ then goto } 3\\
&8 \ : \ X_0 := X_5
\end{aligned}
$$

Тоест всеки път когато имаме програма за машина, която изчислява дадена функция можем директно да я използваме в следваща програма стига да съобразим, кои променливи е безопасно да ползваме.

За домашно

Напишете програми, които да изчисляват следните две функции

$$g(x,y) = \begin{cases} 1 & , \ x < y \\ 0 & , \ x \geqslant y \end{cases}$$

и

$$f(x,y) \simeq \begin{cases} \neg! & , \ y = 0 \\ x \ \mathbf{mod}\ y & , \ y > 0 \end{cases}$$

За целта използвайте следната рекурсивна програма

$$\mathsf{F}(\mathsf{X},\mathsf{Y}) \text{ where}$$
$$\mathsf{F}(\mathsf{X},\mathsf{Y}) = \text{if } \mathsf{X} < \mathsf{Y} \text{ then } \mathsf{X} \text{ else } \mathsf{F}(\mathsf{X} \dot{-} \mathsf{Y}, \mathsf{Y})$$