Arithmetische Befehle

add i	$c_0' = c_0 + c_i$	$b' = b+1$	$i \in \mathbb{N}_+$
cadd i	$c_0' = c_0 + i$	$b' = b+1$	$i \in \mathbb{N}_+$
SUB i	$c_0' = \max\{0, c_0 - c_i\}$	$b' = b+1$	$i \in \mathbb{N}_+$
CSUB i	$c_0' = \max\{0, c_0 - c_i\}$	$b' = b+1$	$i \in \mathbb{N}_+$
MULT i	$c_0' = c_0 \cdot c_i$	$b' = b+1$	$i \in \mathbb{N}_+$
CMULT i	$c_0' = c_0 \cdot i$	$b' = b+1$	$i \in \mathbb{N}_+$
DIV i	$c_0' = \lfloor c_0 / c_i \rfloor$	$b' = b+1$	$i \in \mathbb{N}_+$ (Division ad lib)
CDIV i	$c_0' = \lfloor c_0 / c_i \rfloor$	$b' = b+1$	

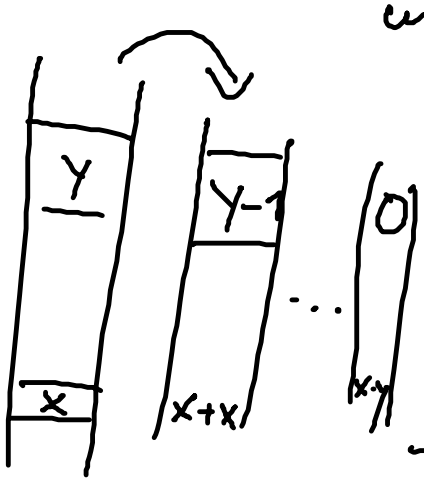
+ Sprungbefehle, Ein- & Ausgabebefehle

Fragen der Registermaschine:

1. Warum diese Befehle?

System ist nicht minimal!
Wie ließe sich MULT / CMULT sparen?

um $x \cdot y$: y-mal x addieren



Speichere y und reduziere in jedem Schritt um 1 → bis das Speicherregister gleich 0 ist.

- aufwendig so zu programmieren

- Vorübersichtlich

Wir wollen:

logische Einheiten „sehen"
↳ Methoden
↳ Kontrollstrukturen

Weitere Fragen: • Gibt es andere Maschinen?
↳ RAM
↳ Turingmaschine
↳ ...

Wie beweise ich, dass zwei Rechnermodelle die gleichen Probleme lösen können?

Eine gute Idee: Zeige $RM_1 \subseteq RM_2$
& $RM_2 \subseteq RM_1$

z.B. indem man die Grundfunktionen von RM_2 durch RM_1 ausdrückt.

Registermaschine $=: RM_1$

Registermaschine ohne MULT; CMULT; $=: RM_2$

$RM_1 \subseteq RM_2$: drücke MULT; & QUOT; durch die übrigen Befehle aus (s.o.).

$RM_2 \subseteq RM_1$: klar.

3. Frage : Berechbarkeit $\leftrightarrow$ Registermaschine

- endlicher Speicher
- wir können in der Berechbarkeit nicht jede natürliche Zahl darstellen.

Binäre oder k-adische Kodierung von Zahlen

$$\forall a \in \mathbb{N}_0 : a = \sum_{i=0}^{\infty} a_i 2^i \;,\; a_i \in \{0,1\}$$

$$a = \sum_{i=0}^{\infty} a_i k^i \;,\; a_i \in \{0,\dots,k-1\} =: [k-1] \cup \{0\} \;|\; k \geq 2$$

Satz: Jede natürliche Zahl läßt sich eindeutig als $\sum_{i=0}^{\infty} a_i k^i$ $a_i \in \{0,\dots,k-1\}$ darstellen, für $k \geq 2$. $\sum_{}^{\log(N)}$

Beweisidee: 1. Jede Zahl in $\mathbb{N}$ lässt sich zur Basis 10 darstellen. → konstruktiv mit Division mit Rest.

2. Die Darstellung ist eindeutig.

Annahme: es gibt zwei solche Folgen $(a_1, \dots, a_j, \dots)$ und $(a'_1, \dots, a'_j, \dots)$. In beiden Folgen sind alle $a_s = 0$ für $s > a$. Bilde Differenzfolge $(a_1 - a'_1, \dots a_s - a'_s, 0 \dots 0 \dots)$ *

Bemerkung: nur endlich viele a_i sind von Null verschieden.

* Irgendwann gibt es: $a_r - a'_r \neq 0$

dann gibt es maximales r^*: $a_{r^*} - a'_{r^*} \neq 0$

- Wenn $a_{r^*} - a'_{r^*} > 0 \Rightarrow$ Folge $(a_1 \dots a_{z_1} \dots)$ besteht eine größere Zahl als die Folge $(a'_1 \dots)$
- Wenn $a_{r^*} - a'_{r^*} < 0 \Rightarrow$ " "$(a'_1 \dots a'_{z_1} \dots)$" "

$\Rightarrow$ Differenzfolge hat nur 0 Einträge.

$\hookrightarrow (a'_1 \dots)$ und $(a_1 \dots)$ sind identisch.

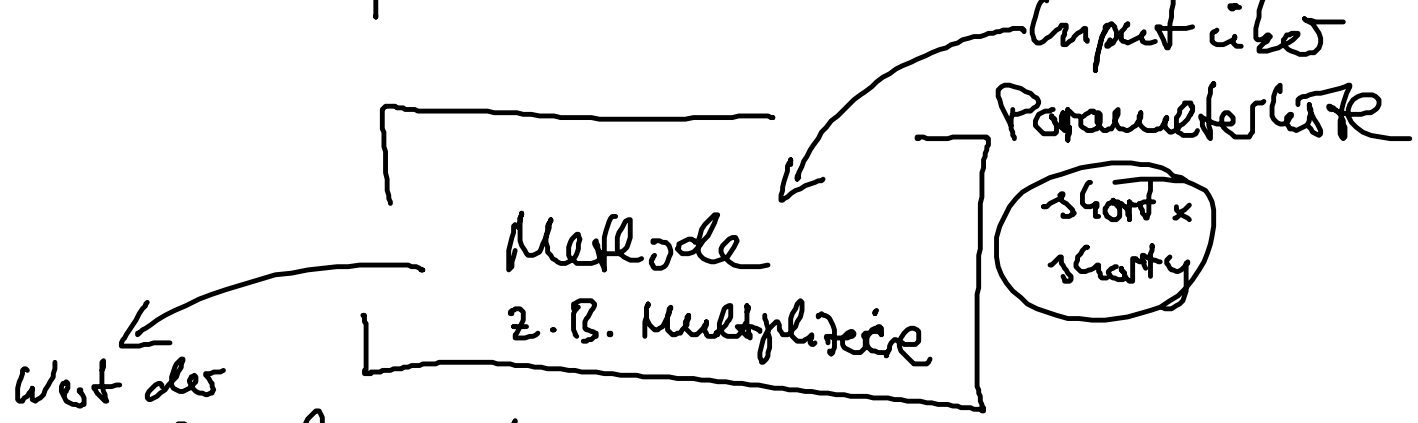
Wieviel Platz braucht man für eine Zahl kleiner N? $\log(N) + 1$

Standardtypen in Java:

	Bit	
boolean	1	true / false

(für $\pm$)

char	16	Unicode
byte	8	-128 ... +127
short	16	-32768 ... 32767
int	32	-2147483648 +2147483647
long	64	-9223372036854775808 +1" "807
float	32	$\pm 1{,}4 \cdot 10^{-45}$ ... $\pm 3{,}4\,10^{38}$
double	64	$\pm 4{,}9 \cdot 10^{-324}$ ... $\pm 1{,}7 \cdot 10^{308}$



Was wenn short Ausgabetyp aber große shorts eingegeben werden?

→ Exception handling

Rückgabetyp name (formale Parameter) {

...

return ...;

}

Bsp: double max (double a, double b) {

```
if (a > b)
    return a;
else
    return b;
}
```

- Methode hat genau einen Rückgabetyp!
- Parameterliste darf leer sein „( ) {"
- Rückgabetyp void

  in void methode gibt es auch ein (oder mehrere) return Anweisung(en).

```
return;
```