Script generated by TTT

Title: Petter: Compiler Construction (02.07.2020) - 51: Arithmetic Expressions

Date: Wed Jul 01 15:08:40 CEST 2020

Duration: 23:04 min

Pages: 11

Translation of Simple Expressions

Using variables stored in registers; loading constants:

instruction	semantics	intuition
$\text{loadc } R_i\ c$	$R_i = c$	load constant
$\text{move } R_i\ R_j$	$R_i = R_j$	copy R_j to R_i

Translation of Simple Expressions

Using variables stored in registers; loading constants:

instruction	semantics	intuition
$\text{loadc } R_i\ c$	$R_i = c$	load constant
$\text{move } R_i\ R_j$	$R_i = R_j$	copy R_j to R_i

We define the following translation schema (with $\rho\, x = a$):

$$\begin{array}{lcl} \text{code}_R^i\ c\ \rho & = & \text{loadc } R_i\ c \\ \text{code}_R^i\ x\ \rho & = & \text{move } R_i\ R_a \\ \text{code}_R^i\ x = e\ \rho & = & \text{code}_R^i\ e\ \rho \\ & & \text{move } R_a\ R_i \end{array}$$

Translation of Expressions

Let op $= \{add, sub, div, mul, mod, le, gr, eq, leq, geq, and, or\}$. The R-CMa provides an instruction for each operator op.

$$\text{op}\ R_i\ R_j\ R_k$$

where R_i is the target register, R_j the first and R_k the second argument.

Correspondingly, we generate code as follows:

$$\begin{array}{lcl} \text{code}_R^i\ e_1\ \text{op}\ e_2\ \rho & = & \text{code}_R^i\ e_1\ \rho \\ & & \text{code}_R^{i+1}\ e_2\ \rho \\ & & \text{op}\ R_i\ R_i\ R_{i+1} \end{array}$$

Translation of Expressions

Let op $= \{add, sub, div, mul, mod, le, gr, eq, leq, geq, and, or\}$. The R-CMa provides an instruction for each operator op.

$$\text{op } R_i \; R_j \; R_k$$

where R_i is the target register, R_j the first and R_k the second argument.

Correspondingly, we generate code as follows:

$$\begin{array}{lll} \text{code}_R^i \; e_1 \text{ op } e_2 \; \rho & = & \text{code}_R^i \; e_1 \; \rho \\ & & \text{code}_R^{i+1} \; e_2 \; \rho \\ & & \text{op } R_i \; R_i \; R_{i+1} \end{array}$$

Example: Translate `3*4` with $i = 4$:

$$\begin{array}{lll} \text{code}_R^4 \; 3*4 \; \rho & = & \text{code}_R^4 \; 3 \; \rho \\ & & \text{code}_R^5 \; 4 \; \rho \\ & & \text{mul } R_4 \; R_4 \; R_5 \end{array}$$

Translation of Expressions

Let op $= \{add, sub, div, mul, mod, le, gr, eq, leq, geq, and, or\}$. The R-CMa provides an instruction for each operator op.

$$\text{op } R_i \; R_j \; R_k$$

where R_i is the target register, R_j the first and R_k the second argument.

Correspondingly, we generate code as follows:

$$\begin{array}{lll} \text{code}_R^i \; e_1 \text{ op } e_2 \; \rho & = & \text{code}_R^i \; e_1 \; \rho \\ & & \text{code}_R^{i+1} \; e_2 \; \rho \\ & & \text{op } R_i \; R_i \; R_{i+1} \end{array}$$

Example: Translate `3*4` with $i = 4$:

$$\begin{array}{lll} \text{code}_R^4 \; 3*4 \; \rho & = & \text{loadc } R_4 \; 3 \\ & & \text{loadc } R_5 \; 4 \\ & & \text{mul } R_4 \; R_4 \; R_5 \end{array}$$

Managing Temporary Registers

Observe that temporary registers are re-used: translate `3*4+3*4` with $t = 4$:

$$\begin{array}{lll} \text{code}_R^4 \; 3*4+3*4 \; \rho & = & \text{code}_R^4 \; 3*4 \; \rho \\ & & \text{code}_R^5 \; 3*4 \; \rho \\ & & \text{add } R_4 \; R_4 \; R_5 \end{array}$$

where

$$\begin{array}{lll} \text{code}_R^i \; 3*4 \; \rho & = & \text{loadc } R_i \; 3 \\ & & \text{loadc } R_{i+1} \; 4 \\ & & \text{mul } R_i \; R_i \; R_{i+1} \end{array}$$

we obtain

$$\text{code}_R^4 \; 3*4+3*4 \; \rho \quad =$$

Managing Temporary Registers

Observe that temporary registers are re-used: translate `3*4+3*4` with $t = 4$:

$$\begin{array}{lll} \text{code}_R^4 \; 3*4+3*4 \; \rho & = & \text{code}_R^4 \; 3*4 \; \rho \\ & & \text{code}_R^5 \; 3*4 \; \rho \\ & & \text{add } R_4 \; R_4 \; R_5 \end{array}$$

where

$$\begin{array}{lll} \text{code}_R^i \; 3*4 \; \rho & = & \text{loadc } R_i \; 3 \\ & & \text{loadc } R_{i+1} \; 4 \\ & & \text{mul } R_i \; R_i \; R_{i+1} \end{array}$$

we obtain

$$\begin{array}{lll} \text{code}_R^4 \; 3*4+3*4 \; \rho & = & \text{loadc } R_4 \; 3 \\ & & \text{loadc } R_5 \; 4 \\ & & \text{mul } R_4 \; R_4 \; R_5 \\ & & \text{loadc } R_5 \; 3 \\ & & \text{loadc } R_6 \; 4 \\ & & \text{mul } R_5 \; R_5 \; R_6 \\ & & \text{add } R_4 \; R_4 \; R_5 \end{array}$$

Semantics of Operators

The operators have the following semantics:

add $R_i\ R_j\ R_k$	$R_i = R_j + R_k$
sub $R_i\ R_j\ R_k$	$R_i = R_j - R_k$
div $R_i\ R_j\ R_k$	$R_i = R_j / R_k$
mul $R_i\ R_j\ R_k$	$R_i = R_j * R_k$
mod $R_i\ R_j\ R_k$	$R_i = signum(R_k) \cdot k$ with $\lvert R_j \rvert = n \cdot \lvert R_k \rvert + k \wedge n \geq 0, 0 \leq k < \lvert R_k \rvert$
le $R_i\ R_j\ R_k$	$R_i =$ if $R_j < R_k$ then 1 else 0
gr $R_i\ R_j\ R_k$	$R_i =$ if $R_j > R_k$ then 1 else 0
eq $R_i\ R_j\ R_k$	$R_i =$ if $R_j = R_k$ then 1 else 0
leq $R_i\ R_j\ R_k$	$R_i =$ if $R_j \leq R_k$ then 1 else 0
geq $R_i\ R_j\ R_k$	$R_i =$ if $R_j \geq R_k$ then 1 else 0
and $R_i\ R_j\ R_k$	$R_i = R_j\ \&\ R_k$ // bit-wise and
or $R_i\ R_j\ R_k$	$R_i = R_j \mid R_k$ // bit-wise or

Semantics of Operators

The operators have the following semantics:

add $R_i\ R_j\ R_k$	$R_i = R_j + R_k$
sub $R_i\ R_j\ R_k$	$R_i = R_j - R_k$
div $R_i\ R_j\ R_k$	$R_i = R_j / R_k$
mul $R_i\ R_j\ R_k$	$R_i = R_j * R_k$
mod $R_i\ R_j\ R_k$	$R_i = signum(R_k) \cdot k$ with $\lvert R_j \rvert = n \cdot \lvert R_k \rvert + k \wedge n \geq 0, 0 \leq k < \lvert R_k \rvert$
le $R_i\ R_j\ R_k$	$R_i =$ if $R_j < R_k$ then 1 else 0
gr $R_i\ R_j\ R_k$	$R_i =$ if $R_j > R_k$ then 1 else 0
eq $R_i\ R_j\ R_k$	$R_i =$ if $R_j = R_k$ then 1 else 0
leq $R_i\ R_j\ R_k$	$R_i =$ if $R_j \leq R_k$ then 1 else 0
geq $R_i\ R_j\ R_k$	$R_i =$ if $R_j \geq R_k$ then 1 else 0
and $R_i\ R_j\ R_k$	$R_i = R_j\ \&\ R_k$ // bit-wise and
or $R_i\ R_j\ R_k$	$R_i = R_j \mid R_k$ // bit-wise or

Note: all registers and memory cells contain operands in $\mathbb{Z}$

Translation of Unary Operators

Unary operators op $= \{neg,\ not\}$ take only two registers:

$$\text{code}^i_R\ \text{op}\ e\ \rho \quad = \quad \begin{array}{l}\text{code}^i_R\ e\ \rho \\ \text{op}\ R_i\ R_i\end{array}$$

Note: We use the same register.

Translation of Unary Operators

Unary operators op $= \{neg,\ not\}$ take only two registers:

$$\text{code}^i_R\ \text{op}\ e\ \rho \quad = \quad \begin{array}{l}\text{code}^i_R\ e\ \rho \\ \text{op}\ R_i\ R_i\end{array}$$

Note: We use the same register.

Example: Translate -4 into R_5:

$$\text{code}^5_R\ -4\ \rho \quad = \quad \begin{array}{l}\text{loadc}\ R_5\ 4 \\ \text{neg}\ R_5\ R_5\end{array}$$