LiP

Interpreter of an Assembly language

Instruction set

Op	Instruction	Example	Description
halt	Termination	Halt	Termine program execution
nop	No Operation	Nop	Skip to next instruction
add	Addition	Add $a $b $c	Write $b + $c to $a
addi	Addition (immediate operand)	Addi $a N $c	Write N + $c to $a
sub	Subtraction	Sub $a $b $c	Write $b – $c to $a
mul	Multiplication	Mul $a $d $e	Write $d * $e to $a
load	Transfer mem → reg	Load $a $b[$i]	Write M[$b+$i] to $a
store	Transfer reg → mem	Store $b[$i] $a	Write $a ton M[$b+$i]
jmp	Jump	Jmp L	Jump to label L
bne	Branch if Not Equal	Bne $a $b L	Jump to label L if $a ≠ $b
beq	Branch if Equal	Beq $a $b L	Jump to label L if $a = $b
slt	Set if Lower Than	Slt $a $b $c	Write $a=1 if $b<$c, $a=0 otherwise

Example

```
        addi $f 1 $0        // f:=1
        addi $i 1 $0        // i:=1
Loop:   slt $t $n $i        // t:=1 if n<i, t:=0 if i≤n
        bne $t $0 End       // jump to End if t≠0
        mul $f $f $i        // f:=f*i
        addi $i 1 $i        // i:=i+1
        jmp Loop
End:  ...
```

ASM programs

An ASM program is modelled as a list of pairs:

$$(L_1, istr_1) \quad (L_2, istr_2) \quad \ldots \quad (L_n, istr_n)$$

each occurrence of a non-empty L_i must be unique (no ambiguity).

$$\text{Lookup}(P,L) = n \qquad \text{iff fst } P(n) = L$$

Registers

$R : \text{String} \rightarrow \text{Int}$ (partial function)

$R(\$0) = 0$ (always)

Substitution R{v/i} defined as follows:

$$(\forall j) \quad R\{v/i\}(j) = \begin{cases} v & \text{if } j=i \\ R(j) & \text{otherwise} \end{cases}$$

Memory

$M : [0, 2^{16}-1] \rightarrow Int$ (partial function)

$M(i)$ with $i<0$ or $i>2^{16}-1$ must raise an exception

You can decide what to do with $M(i)$ uninitialized:

- $M(i) = 0$ if not initialized
- $M(i)$ raises an exception if not initialized

Halt

$$\frac{P(n) = \text{Halt}}{(P, R, M, n) \rightarrow_{\text{exec}} (R, M)}$$

Nop

$$\frac{P(n) = \mathrm{Nop}}{(P, R, M, n) \rightarrow_{\mathrm{exec}} (P, R, M, n+1)}$$

Add

$$\frac{P(n) = \text{Add } \$a\ \$b\ \$c \qquad a \neq 0 \qquad R(\$b) + R(\$c) = v}{(P, R, M, n) \rightarrow_{\text{exec}} (P, R\{v/a\}, M, n+1)}$$

Addi

$$\frac{P(n) = \text{Addi } \$a \; N \; \$b \qquad a \neq 0 \qquad R(\$b) + N = v}{(P, R, M, n) \rightarrow_{exec} (P, R\{v/a\}, M, n+1)}$$

Sub

$$\frac{P(n) = \text{Sub } \$a\ \$b\ \$c \qquad a \neq 0 \qquad R(\$b) - R(\$c) = v}{(P, R, M, n) \rightarrow_{\text{exec}} (P, R\{v/a\}, M, n+1)}$$

Mul

$$\frac{P(n) = \text{Mul } \$a\ \$b\ \$c \qquad a \neq 0 \qquad R(\$b) * R(\$c) = v}{(P, R, M, n) \rightarrow_{exec} (P, R\{v/a\}, M, n+1)}$$

Load

$$P(n) = \text{Load } \$a\ \$b[\$i]$$

$$a \neq 0$$

$$M(R(b) + R(i)) = v$$

$$(P, R, M, n) \rightarrow_{\text{exec}} (P, R\{v/a\}, M, n+1)$$

Store

$$\frac{P(n) = \text{Store } \$b[\$i]\ \$a \qquad \ell = R(b) + R(i) \qquad R(a) = v}{(P, R, M, n) \rightarrow_{exec} (P, R, M\{v/\ell\}, n+1)}$$

Jmp

$$\frac{P(n) = \text{Jmp } L \qquad \text{lookup}(P,L) = n'}{(P, R, M, n) \rightarrow_{\text{exec}} (P, R, M, n')}$$

Beq

$$\frac{P(n) = \text{Beq } \$a\ \$b\ L \qquad R(a) = R(b) \qquad \text{lookup}(P,L) = n'}{(P, R, M, n) \rightarrow_{\text{exec}} (P, R, M, n')}$$

$$\frac{P(n) = \text{Beq } \$a\ \$b\ L \qquad R(a) \neq R(b)}{(P, R, M, n) \rightarrow_{\text{exec}} (P, R, M, n+1)}$$

Bne

$$\frac{P(n) = \text{Bne } \$a\ \$b\ L \qquad R(a) \neq R(b) \qquad \text{lookup}(P,L) = n'}{(P, R, M, n) \rightarrow_{\text{exec}} (P, R, M, n')}$$

$$\frac{P(n) = \text{Bne } \$a\ \$b\ L \qquad R(a) = R(b)}{(P, R, M, n) \rightarrow_{\text{exec}} (P, R, M, n+1)}$$

Slt

$$\frac{P(n) = \text{Slt } \$a\ \$b\ \$c \qquad R(b) < R(c) \qquad a \neq 0}{(P, R, M, n) \rightarrow_{\text{exec}} (P, R\{1/a\}, M, n+1)}$$

$$\frac{P(n) = \text{Slt } \$a\ \$b\ \$c \qquad R(b) \geq R(c) \qquad a \neq 0}{(P, R, M, n) \rightarrow_{\text{exec}} (P, R\{0/a\}, M, n+1)}$$