# Program computing sum of a list

| | | | | |
|---|---|---|---|---|
| 0 | sum: | INV | $\mathtt{r}_0 : \mathtt{int\ list}$ | % invariant |
| 1 | | MOV | $\mathtt{r}_1, 0$ | % initialize accumulator |
| 2 | $L_2$ | INV | $\mathtt{r}_0 : \mathtt{int\ list} \wedge \mathtt{r}_1 : \mathtt{int}$ | % loop invariant |
| 3 | | BEQ | $\mathtt{r}_0, L_8$ | % is list empty? |
| 4 | | LD | $\mathtt{r}_2, 0(\mathtt{r}_0)$ | % load head |
| 5 | | LD | $\mathtt{r}_0, 4(\mathtt{r}_0)$ | % load tail |
| 6 | | ADD | $\mathtt{r}_1, \mathtt{r}_2, \mathtt{r}_1$ | % do the addition |
| 7 | | BR | $L_2$ | % loop |
| 8 | $L_8$ | RET | | % result is in $\mathtt{r}_1$ |

# Typing rules

$$\text{List} \quad \frac{m \vdash e : \texttt{int list} \qquad e \neq 0}{\begin{array}{c} m \vdash e : \texttt{addr} \ \wedge \ m \vdash e + 4 : \texttt{addr} \ \wedge \\ m \vdash \texttt{sel}(m, e) : \texttt{int} \ \wedge \ m \vdash \texttt{sel}(m, e+4) : \texttt{int list} \end{array}}$$

$$\text{Add} \quad \frac{m \vdash e_1 : \texttt{int} \qquad m \vdash e_2 : \texttt{int}}{m \vdash e_1 + e_2 : \texttt{int}}$$

$$\text{Zero} \quad \frac{}{m \vdash 0 : \texttt{int}}$$

# PCC versus TAL

- Typed Assembly Language (TAL) can be viewed as a special case of PCC, but providing compiler writers with a higher-level set of abstractions.
- Claim by Morrisett et al: *the ideal system contains both the compiler support and compact annotations of TAL and the flexibility of PCC*.