

# 1. Introduction

Low-level programs are sometimes hand-written to facilitate efficient computing. Another situation where low-level programs are used is extensible, performance-conscious systems. Such systems exploit low-level portable programs. However, the safety of most low-level programs is not guaranteed since most low-level languages provide only inferior safety mechanisms and don't have their own type systems.

Typed assembly languages are introduced in a paper “From System F to Typed Assembly Language” (Morrisett et al., 1998).

In this article, we define a general-purpose typed assembly language which targets abstract machines. Its syntax is given in Figure 1.

$r ::=$	registers:
$r1 \mid r2 \mid \dots \mid rk$	general-purpose registers
$\nu ::=$	operands:
$r$	register
$i$	integer
$\iota ::=$	instructions:
$\text{mov } r \ \nu$	move
$\text{add } r \ \nu \ \nu$	add
$\text{sub } r \ \nu \ \nu$	subtract
$\text{and } r \ \nu \ \nu$	logical and
$\text{or } r \ \nu \ \nu$	logical or
$\text{not } r \ \nu$	logical not
$\text{shl } r \ \nu \ \nu$	logical shift left
$\text{shr } r \ \nu \ \nu$	logical shift right

Figure 1: Instructions and operands