Meeting 07: Register Allocation

HW2 Comments

What was hard?
1) Learning the “lingo”
2) Debugging the parse table (LALR(1) algorithm)
3) Po’s syntax (newlines / spaces important)
4) PLY tool

Fun assignment!
Lead board! Fun!

Register Allocation HW 3

Two week project
Get started right away!
HW 1

Parse PO AST → Flattened PO AST

( Get rid of bushy expr trees)

tmp0 = 1 + 2

tmp1 = tmp0 + 3

tmp2 = input()

Slow! Access Registers ~ 1 cycle
Access Memory
Cache ~ tens cycles
Main memory ~ hundreds

Unbounded number of variables

Finite registers

PO → Stack x86
2 variables \( \rightarrow \) \( \rightarrow \) registers

program

\( x86 \) x 8 registers

HW3

Flatten \( \rightarrow \) Flattened P0 AST \( \rightarrow \) Instruction Selection \( \rightarrow \) P0 IR

Register Allocation (HW3)
How do we want to allocate registers?

\[ x \rightarrow \text{reg eax} \]
\[ y \rightarrow \text{reg ebx} \]
\[ z \rightarrow \text{reg eax} \]
\[ w \rightarrow \text{stack} \]
\[ a \rightarrow \text{stack} \]

1. Maximize the number of variables we can put in registers.
2. Prioritize.

What was the purpose of computing liveness of variables?

I z’s live range
I y’s live range

non-overlapping

Which vars have overlapping live ranges?
"Conflict" = Interference

Interference Graph (undirected)

Nodes: Variables
Edge: Vars "interfer" i.e., they have overlapping use ranges

Colors = registers

1. Liveness
2. Interference graph
3. Coloring graph to maximize # variables in registers
4. Register allocation (stack)

Interference Graph Construction

\[ E = w, x, z \]

That program point witnesses the need for those variables to be separated (i.e., interference)
BFS Algo: Go through each live cell set

\[ \rightarrow \text{Add edge between each pair nodes} \]