Introduction

- In many programming languages, the programmer has the illusion of allocating arbitrarily many variables.
- However, during compilation, the compiler must decide how to allocate these variables to a small, finite set of registers.
- Not all variables are in use (or “live”) at the same time, so some registers may be assigned to more than one variable. However, two variables in use at the same time cannot be assigned to the same register without corrupting its value.
- Variables which cannot be assigned to some register must be kept in RAM and loaded in/out for every read/write, a process called spilling.
- Accessing RAM is significantly slower than accessing registers and slows down the execution speed of the compiled program, so an optimizing compiler aims to assign as many variables to registers as possible.
- Register pressure is the term used when there are fewer hardware registers available than would have been optimal; higher pressure usually means that more spills and reloads are needed.

Better approach = register allocation: keep variable values in registers as long as possible.

Best case: keep a variable’s value in a register throughout the lifetime of that variable.

- In that case, we don’t need to ever store it in memory
- We say that the variable has been allocated in a register
- Otherwise allocate variable in activation record
- We say that variable is spilled to memory

Which variables can we allocate in registers?

Depends on the number of registers in the machine. Depends on how variables are being used.

- Main Idea: cannot allocate two variables to the same register if they are both live at some program point.

Spilling

- In most register allocators, each variable is assigned to either a CPU register or to main memory.
- The advantage of using a register is speed. Computers have a limited number of registers, so not all variables can be assigned to registers.
- A “spilled variable” is a variable in main memory rather than in a CPU register. The operation of moving a variable from a register to memory is called spilling, while the reverse operation of moving a variable from memory to a register is called filling. For example, a 32-bit variable spilled to memory gets 32 bits of stack space allocated and all references to the variable are then to that memory.
- Such a variable has a much slower processing speed than a variable in a register. When deciding which variables to spill, multiple factors are considered: execution time, code space, data space.

Iterated Register Coalescing

- Register allocators have several types, with Iterated Register Coalescing (IRC) being a more common one.
- IRC was invented by LAL George and Andrew Appel in 1996, building on earlier work by

Gregory Chaitin.

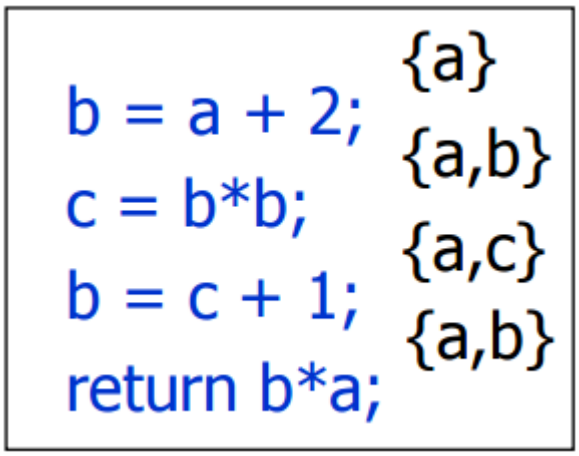
- IRC works based on a few principles. First, if there are any non-move related vertices in the graph with degree less than K the graph can be simplified by removing those vertices, since once those vertices are added back in it is guaranteed that a color can be found for them (simplification).

Register Allocation Algorithm

Basic algorithm for register allocation is:

1. Perform live variable analysis (over abstract assembly code!)
2. Inspect live variables at each program point
3. If two variables are ever in same live set, they can't be allocated to the same register – they interfere with each other
4. Conversely, if two variables do not interfere with each other, they can be assigned the same register. We say they have disjoint live ranges.

Interference Graph



- Nodes=program variables
- Edges = connect variables that interfere with each other

Register allocation = graph coloring