Kevin Slachta

CS415

Project 1

During this project I used 3 different algorithms to implement my top down, simple top down and bottom up allocators. For the first part I'll talk about my simple top down allocator. Simply put, I iterated through the entire instruction set generating an array of occurrences for each register. I then assigned k-f registers to be assigned to the registers with the highest occurrence rate, in the event of a tie I chose the register that came first. Pretty simple not much else to this algorithm.

For the top down with live ranges I took the same approach in the beginning, iterating through all the instructions to generate an occurrence array. For the live range approach to work though, I needed to generate extra data about the instructions. Iterating through the instruction list I generated 2 more arrays. The first being the maxlive of each instruction and the second before the live range for each register. To save space and lessen the workload later on I combined the livemap, maxlive and live ranges into a single array called livemap. Column 0 was reserved for the maxlive of each instruction and the final row was used to save the live range of each register. Every column number referred to a register number ex livemap[1][2] referred to instruction 1 register 2 and so on and so forth. After this I iterated through my live map, where ever maxlive > k – f I assigned one register to be spilled and continued, looping here until all lines had a maxlive <= k -f. I first spilled registers with the least occurrences on lines with maxlive > k -f and in the event of a tie I referred to the live range to pick the winner.

Bottom Up allocator is simply put, not yet finished. I still have some kinks to iron out but it works nearly 50% of the time. Simply put my current iteration of bottom up works very crudely. Step 1 is to generate the live ranges for all registers and store that for later spilling. Step2 is to check if any k registers that are currently loaded are no longer live. If any of these k registers are no longer live it can safely take that register and reside within it until it becomes dead. Otherwise I have to choose a register to spill. Using the live range generated before hand it chooses the register with the longest live range and spills that register, swapping in the data to the register that was just spilled.

Based on my results, as you can see below, a lot of cycles have to be wasted and many more instructions executed to allocate the tiny amount of physical registers to perform operations. Simply put more registers allow for less wasted cycles moving data in and out of virtual registers. However I would like to compare the results more closely of the live range top down vs the simple top down allocators.

During the live range top down allocator, there are a lot fewer wasted cycles and instructions versus the simple top down. Using the extra information generated from the live range and maxlive you can save a

lot more cycles rather than using just occurrences as the dictator. Of course, there is no free lunch, the cycles and time you save generating the live range and maxlive have to outweigh the cost of generating all this information and applying it. For example if using the simple top down it was 40 cycles to generate and 100 to execute that's 140 cycles total or using live range top down it would take 100 cycles to generate and 60 to execute, that's 160 cycles, you paid extra for less cycles later but the exchange might not always be better.