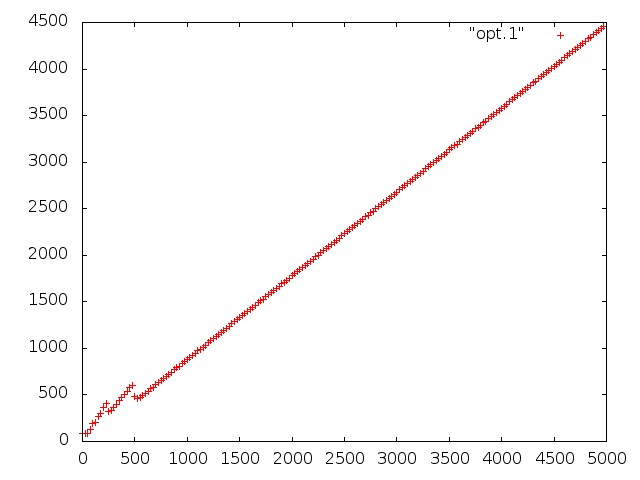
Hayden Miedema and Douglas Money

Professor Kurmas

CIS 451

Lab 9: IA32 Differences

1. Save the plot to a file and include it in your lab writeup.
2. What is the (approximate) slope of the line in your plot? (Find a line that matches reasonably well. You need not do a formal regression analysis.) Remember, these steps will save the plot to a file.

Slope ~ .9

The slope varies a bit, but averages right around a slope of 1. This is especially apparent when the data points are higher than 500.

* + set term jpeg
  + set output "myPlot.jpeg"

1. plot "opt.1"
2. Write timingos.iso to a USB drive and run the experiment on a Mac. Note any differences between the last data points printed and your results from running ./timingos.debug. Is the slope the same? (It would be nice to be able to generate another plot, but I don't have that working yet.)
   * The slope hovers around 1. The ending data points hover around 4900, which is a bit larger than the 4500.
3. Run the experiment on the AMD and note any differences between the AMD and the Mac.
   * The AMD is again around .9, like that of the first linux hardware, and the overall data points again hover around 4500, whereas the data points in this region for the Mac hover around 5000.
4. You will notice that the slope of the line for the Intel processors is less than 1. Explain how this is possible given the data dependency between each instruction. Hint: Read page 2-9 of [Intel Software Develeloper's Manual (volume 1)](http://www.intel.com/content/dam/www/public/us/en/documents/manuals/64-ia-32-architectures-software-developer-vol-1-manual.pdf).
   * The Intel machines have ALU units that run at twice the processor frequency, which greatly increases the speed that arithmetic instructions can be run, but more importantly it also allows instructions that rely on each other to be run in parallel.
5. Run ./timingos.debug | grep -v cur > opt.2 and plot the results.
6. What is the slope of the line in the plot?
   * The slope of this line is around 1-1.1. It is definitely greater than it was before.
7. Write timingos.iso to the USB drive, run the experiment on the Mac and AMD, and note any differences (including in the slope).
   * This time, about double the instructions are being processed at once (a slope of around .5). This means that instructions are more capable of being run in parallel.
8. What does the change in slope indicate? Why did
   * The change in slope indicates that about 2 instructions are now being processed in every cycle instead of 1.
9. The slopes of the line should be significantly less than 1. What does this tell you?
   * Again, a slope of less than 1 means that more than 1 instruction is being processed through each cycle. This tells us that there must be something that the CPU is doing to allow these instructions to be run at the same time.
10. The first set of experiments produces code that looks like this:

addl $1, %eax  
 addl $1, %eax  
 addl $1, %eax  
 addl $1, %eax  
 ...

whereas the second set produces code that looks like this:

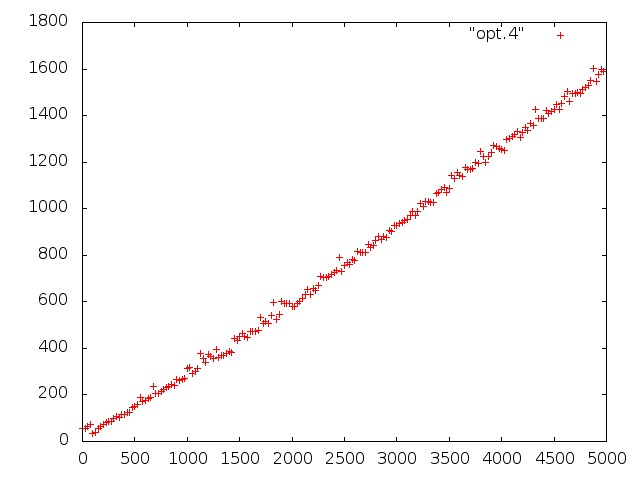
addl $1, %eax  
 addl $1, %ecx  
 addl $1, %eax  
 addl $1, %ecx  
 ...

Why can the CPU run the second example faster?

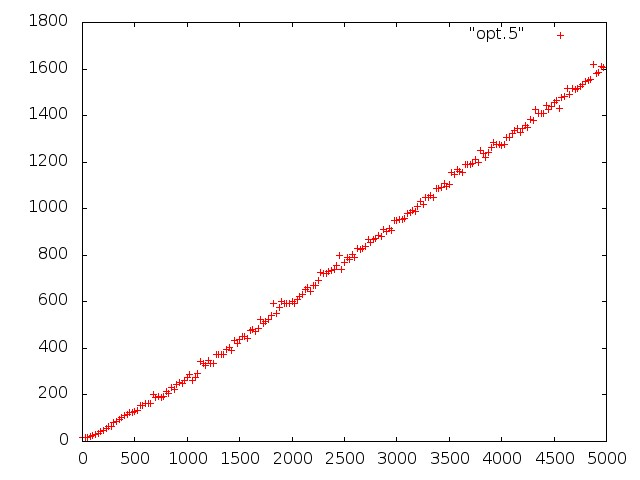
The CPU is running the second example faster because each following instruction is using a different register (%eax or %ecx).This allows double the instructions to run, or the running of instructions in parallel, because the needed data is being accessed in different registers

1. Figure out how many add instructions each different CPU can do in parallel. Attach graphs demonstrating this. (In other words, show me how you figured it out.) In general, each plot should contain more than one line. **WARNING:** Make sure your timing code doesn't clobber %ebx, %esp or any other register that is still in use when your code reaches the first rdtsc.

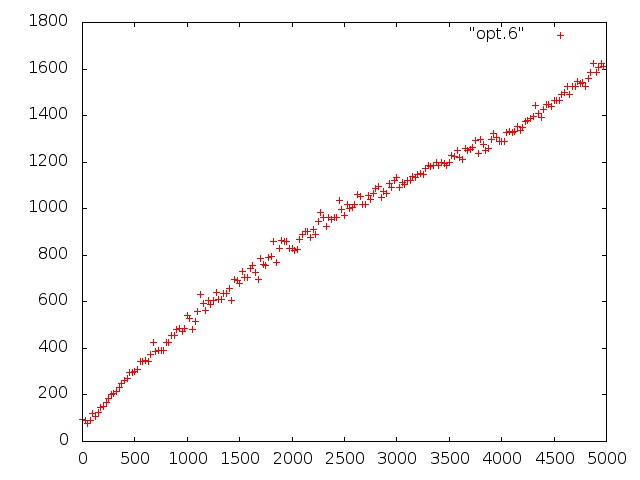
6 different registers:



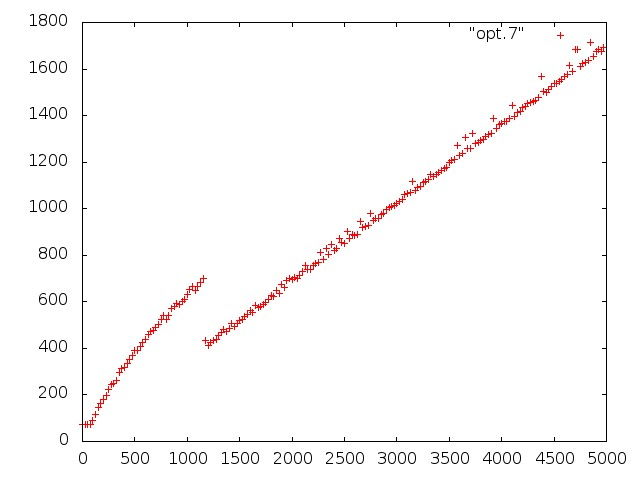
5 different register:



4 different registers:



3 different registers:



From these graphs, we can see a level off at around 1600. This is about 5000/3, meaning that the max number of instructions that are able to be run in parallel is 3. After trying 4, 5, and 6 registers we see no increase in the number of instructions being run.