

**AMSC 662: Performance Analysis of the 1GHz Motorola G4
RISC processor versus the 1.13 GHz P3 and 2.4 GHz P4 Intel
CISC processors**

P. Aaron Lott

University of Maryland

Department of Applied Math &

Scientific Computation

AMSC 662

Nov. 6, 2003

Motivation

- PowerPC Chips use a RISC based architecture, and traditionally design chips with larger/more cache than Intel's CISC based processors.
- The Motorola 745x chipset series supposedly can perform 2 single precision floating point operations per clock cycle, or one double precision operation per clock cycles whereas on an Intel it takes two clock cycles for such operations.
- We would like a tool to measure the performance of basic operations to gain insight about how to optimize their code for a specific chip.

Code Design

- Generalized the “combine” functions from the notes to allow for abstract data types, user defined or built in i.e. float, char, int, double.
- Computes summation or product of vector elements. User defined vector length.
- Documentation for adding and testing other functions.
 - Character or string search, Matrix, matrix-vector multiply
 - Any function that requires large amounts of clock cycles
- Uses #ifdef statements to define what code is run depending on architecture and user preferences.
- Outputs data to MATLAB vector format for plotting/analysis purposes.

Code Specifics

- X86 cycle counters are read via the inline assembly code discussed in class.
- PowerPC cycle counters are read via CHUD (Computer Hardware Understanding Development) Tools.

Implementing Own Scheme

- To implement your own routine you will need to first define the function to receive values `void my_func(vec_ptr v, DATATYPE *dest)`. (you may want to use one of the `combine*` functions as a template).
- Then call the function from the `call_combine.c` function. You'll be able to insert your code beneath the code for `combine6aaa` for example.
- You will also want to create a variable for your count. Depending on the architecture you wish to test.
- Finally include your file in the `include.h` file directly beneath the `combine6aa.c`. Setup the `parameters.h` and `Makefile`

Makefile, include file, parameters file

- Makefile - Choose Optimization Flag
- parameters.h - Define DATATYPE, VECLength, type of test
i.e. X86, CHUD, Timing
- include_files.h - Include any new function you wish to call.

System Requirements

- For Intel: Known to work on Linux systems with GCC compiler
- For PPC: Known to work on G3 and G4 systems running OS X and GCC compiler. Needs the CHUD Toolkit.
Should work on G5 OS X systems as well.

Output

- The output is in MATLAB vector format and plotting scripts are available to visualize the results.
- The “cycles” vector lists the number of clock cycles it took to perform the operations defined in `call_combine.c`.
- On the G4, there is an additional vector, “instructions”, that lists the number of instructions needed to perform the same operations.

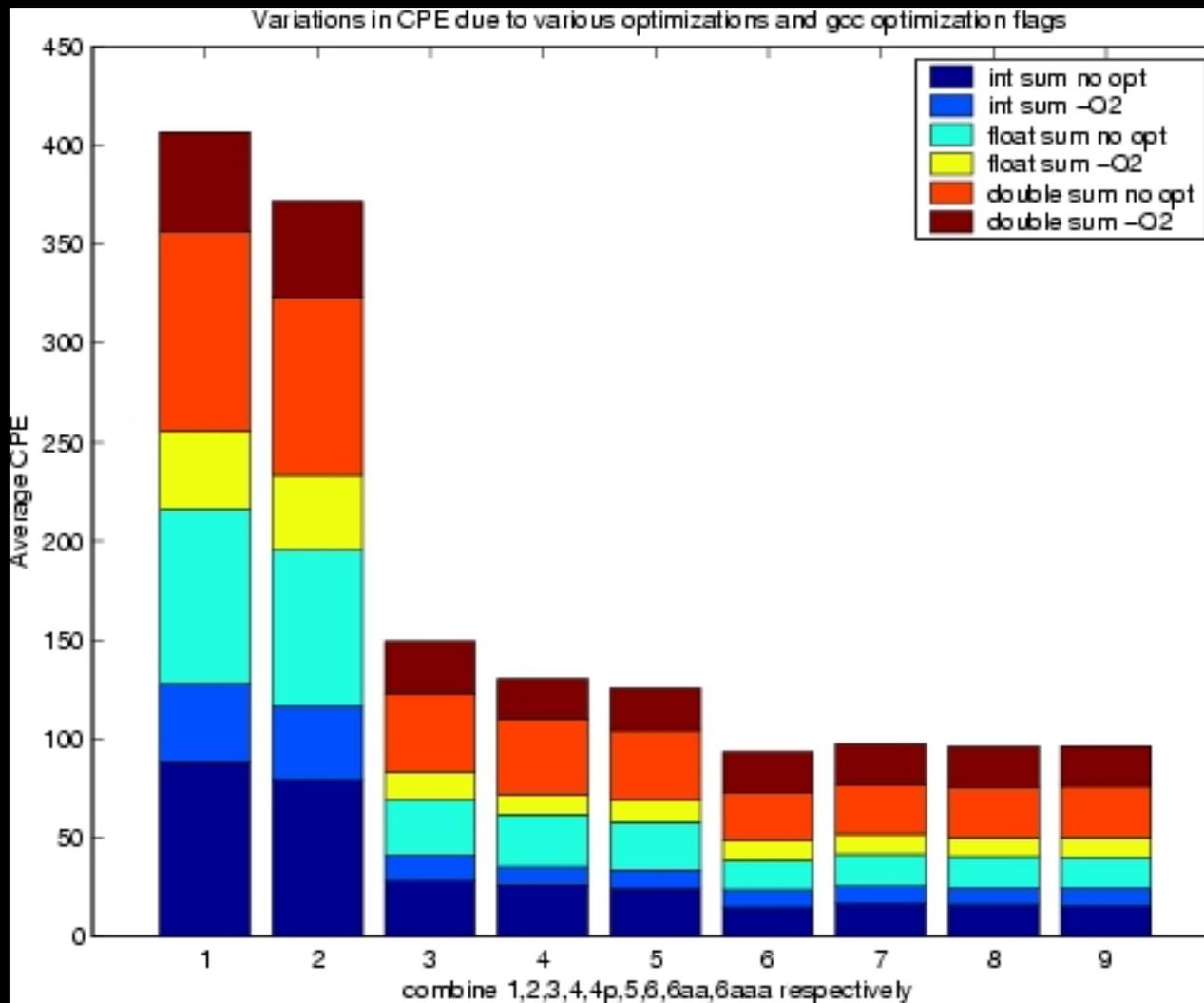


Figure 1: Performance Results from G4 Runs with 10000 elements. Cycles per element.

MONster

- Running with the CHUD_ON flag will allow the remote access of the MONster program to read the registers being updated during your code.
- Run your code and MONster will write the results from your run in the Results window with Labels corresponding to the the name of the function being tested. MONster will output the results to a text file if you wish.

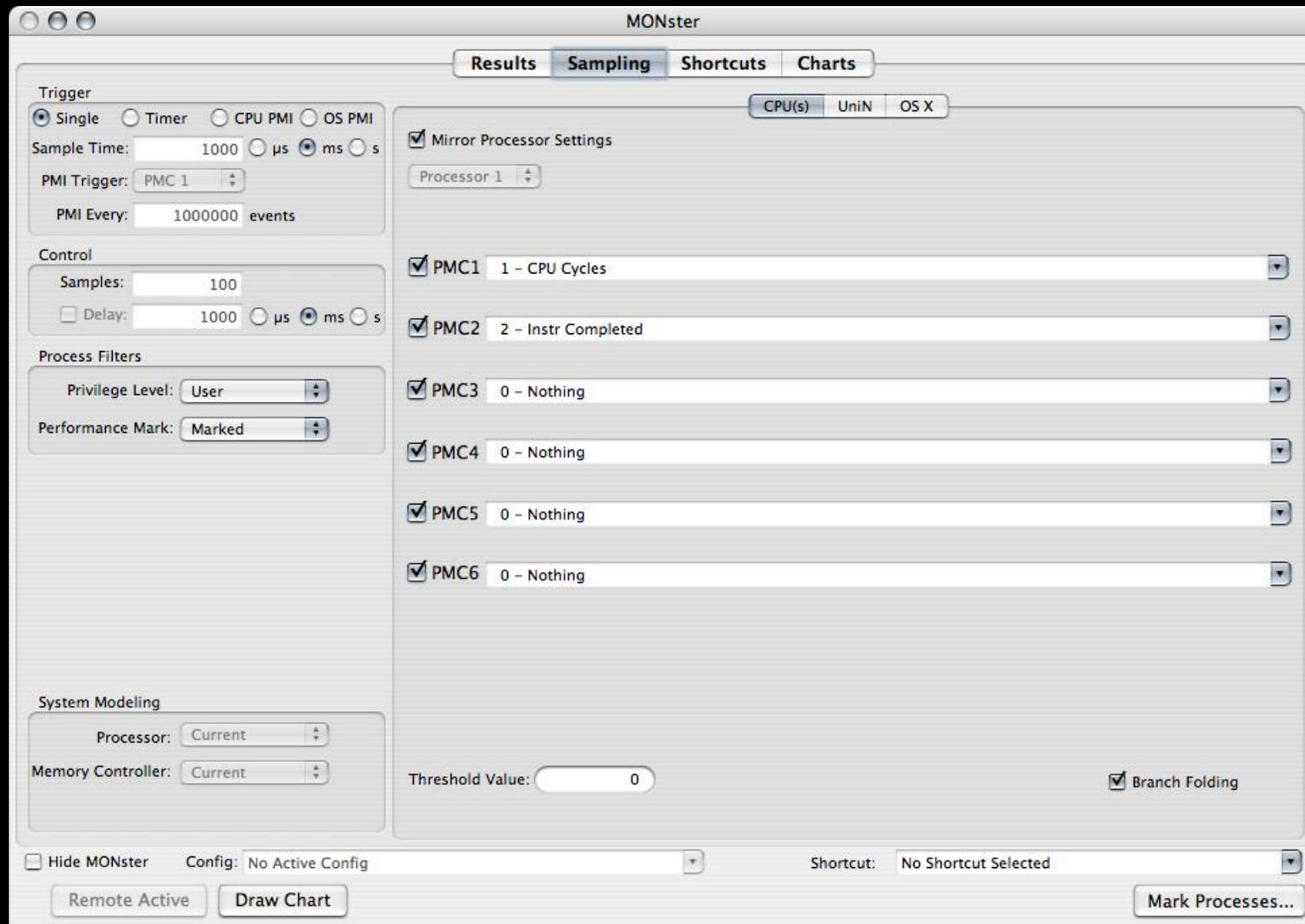


Figure 2: MONster Sampling Screen

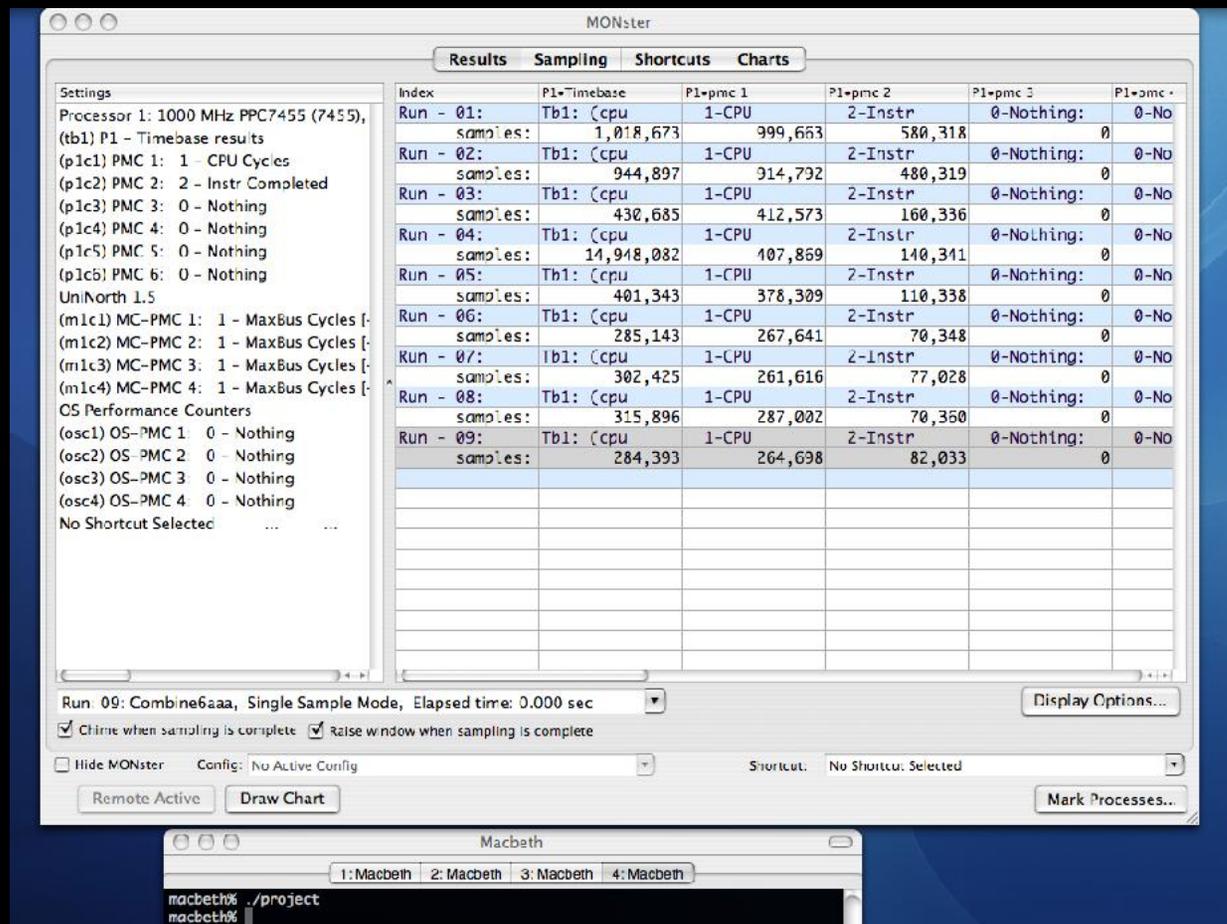


Figure 3: MONster Results Screen

Downloads

- All the code, including the C combine routines, performance routines as well as the MATLAB plotting routines, makefiles, and pre-compiled binaries for both x86-Linux and OS X-G4 platforms is available on my project web site at:

<http://www.lcv.umd.edu/~palott/research/graduate/662/>

- CHUD is available from the macupdate web site at:

<http://www.macupdate.com/info.php/id/8506>

Apple G4/ Motorola 7455

- Results are about 9 times slower than what we expected on the highly optimized combine5 code, with a theoretical CPE 1.0, but measured CPE 8.9928.
- G4 is supposed to be capable of performing 2 IPC for both integer and floating point arithmetic, and 1 IPC for doubles.
- Figure 6 shows us that even though we have optimized code, we don't necessarily get optimal results.

Although we have made great optimizations, we have made so few instructions that the clock is actually hungry for data. Thus we need a faster bus to feed the processor.

It would be ideal if we could indeed prove this by measuring the number of instructions performed by the x86 machines with faster buses, or a G5 with a faster bus.

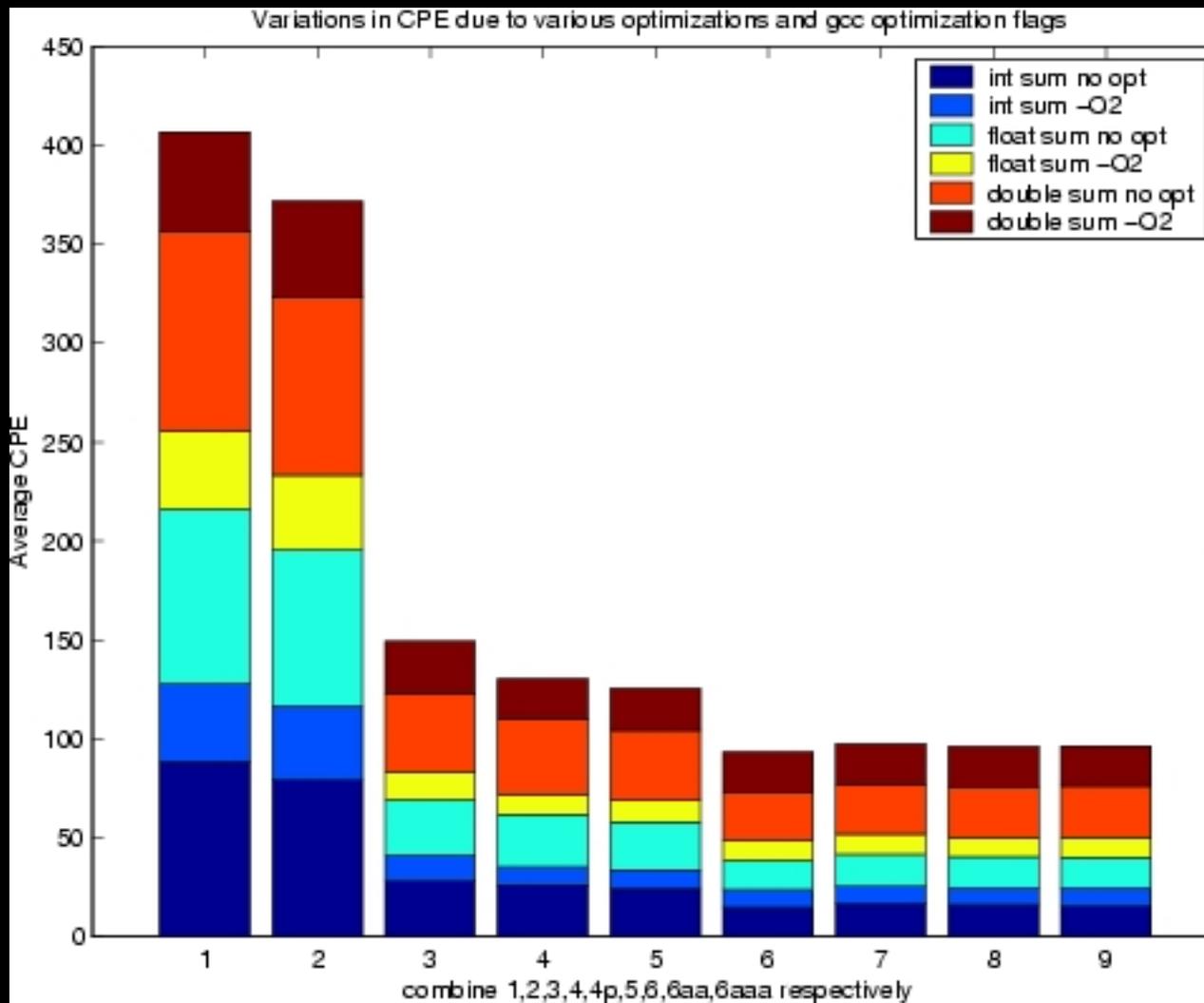


Figure 4: G4 cycles per element 10000 elements.

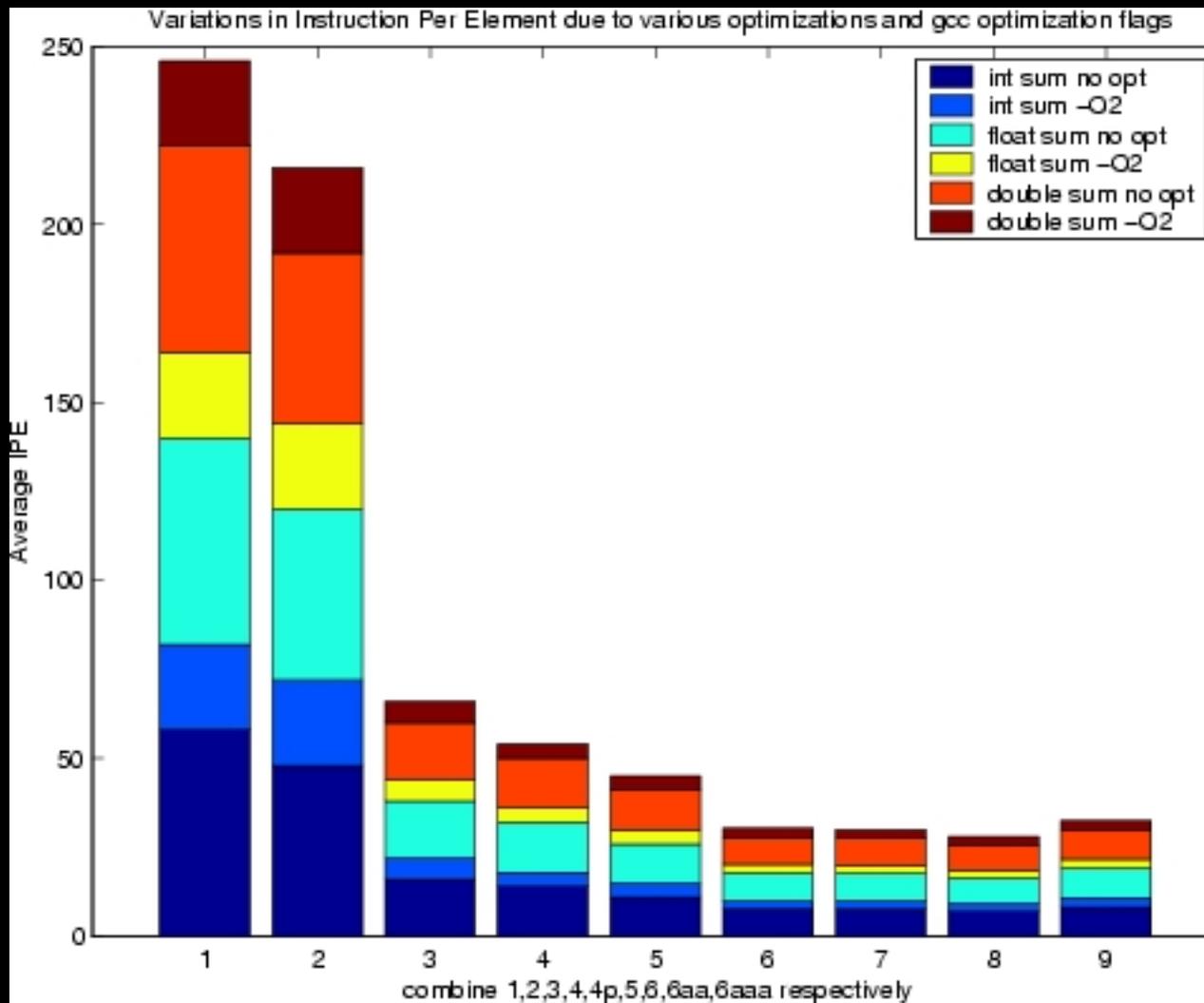


Figure 5: G4 instructions per element 10000 elements.

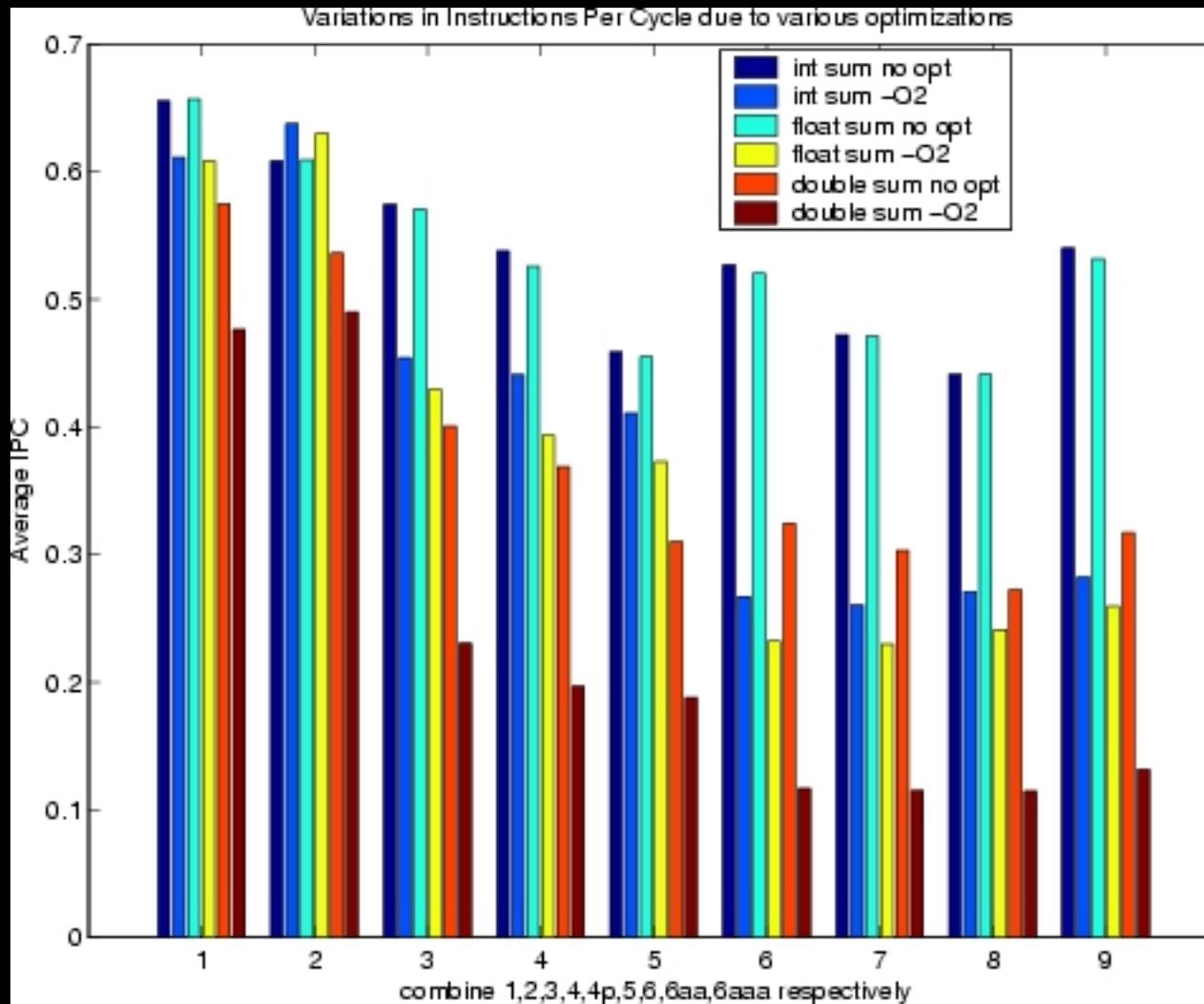


Figure 6: Performance Results from G4 Runs with 10000 elements. Instructions Per Cycle. Nowhere close to optimal performance

Pentium III and Pentium IV

- Measured results are near theoretical results in most cases.
- combine5 Theoretically gives CPE 1.00, 1.13 GHz P3 1.6915, 2.4 GHz P4 1.4432
- combine6 Performs terrible on the floating point computation. Latencies dominate the advantages of unrolling, and on the P4 we get stomped on even more

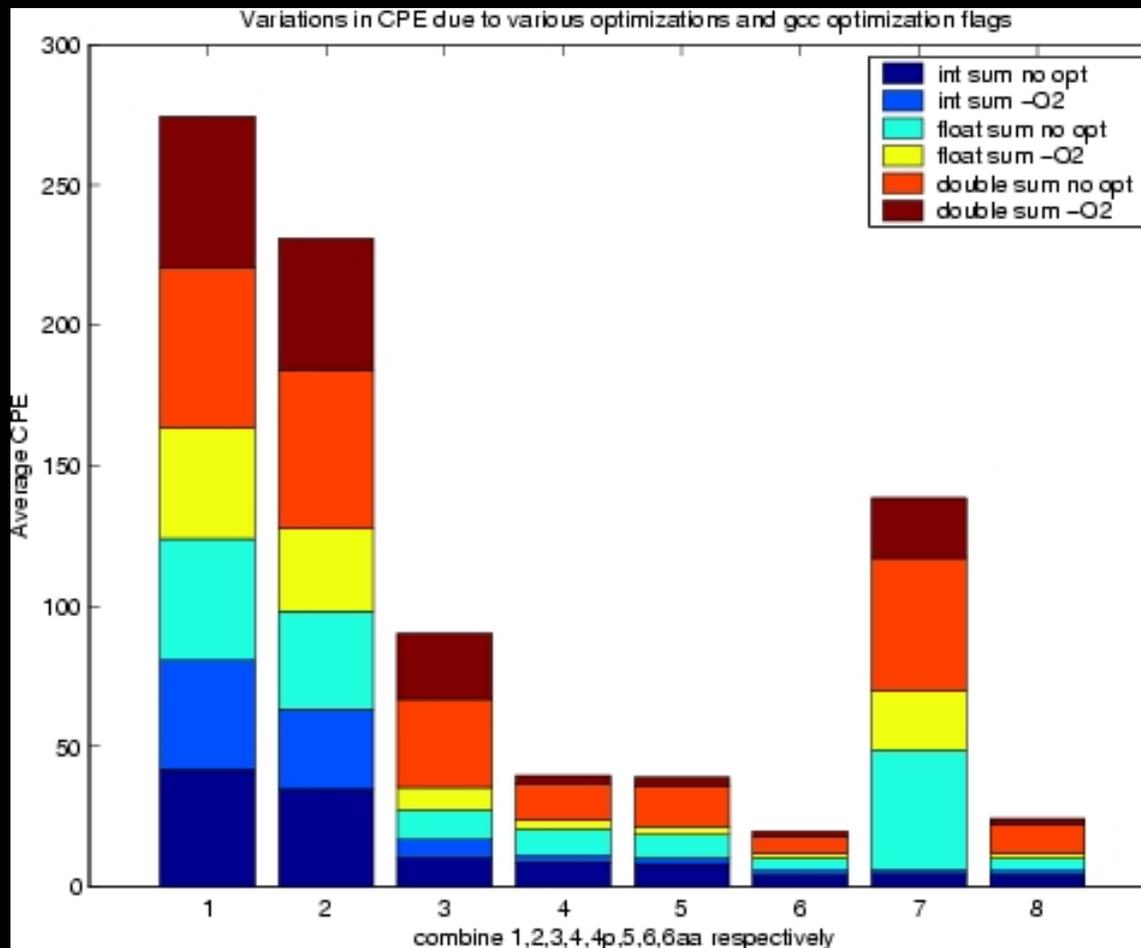


Figure 7: Performance Results from P3 Runs with 10000 elements.

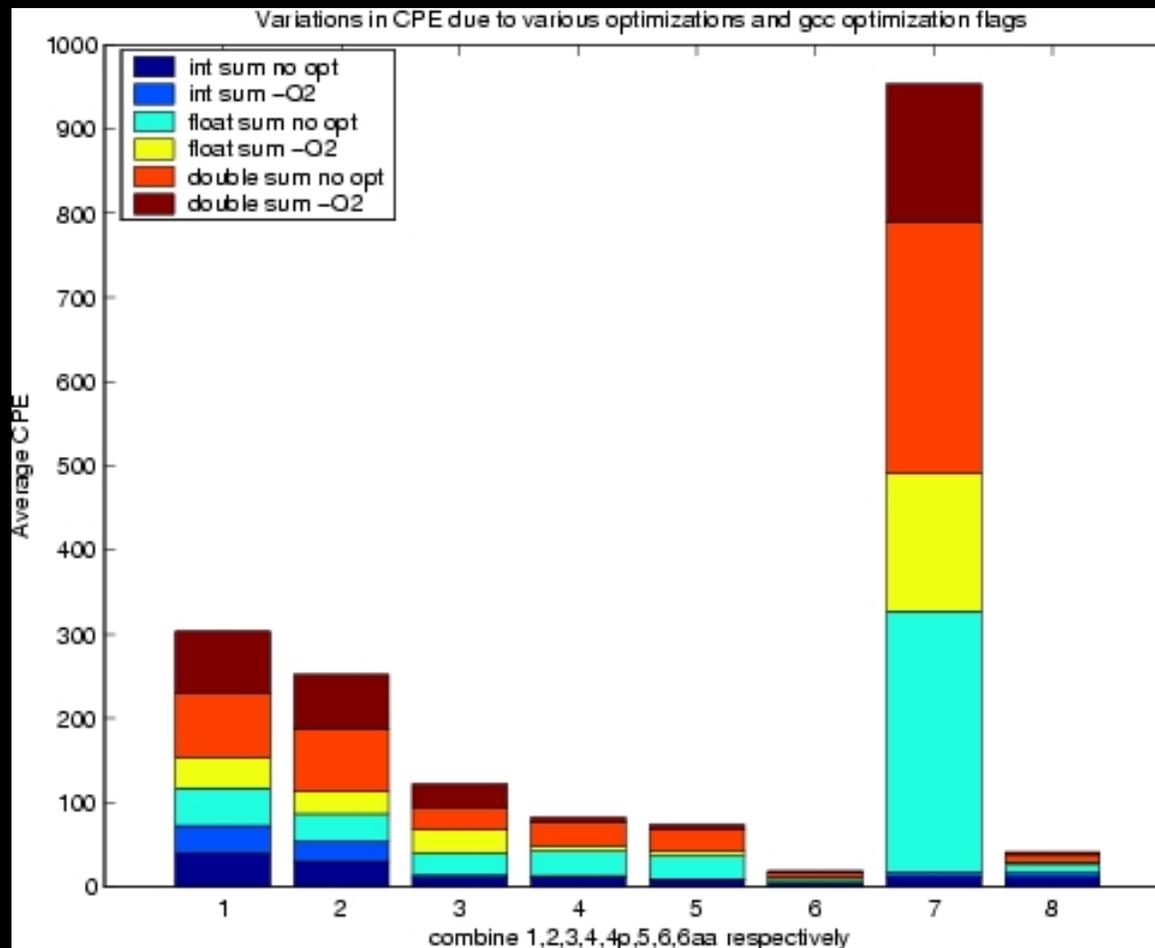


Figure 8: Performance Results from P4 Runs with 10000 elements.

```
void combine6(vec_ptr v, DATATYPE *dest)
{
    int i;
    int length=vec_length(v);
    int limit= length-1;
    DATATYPE *data=get_vec_start(v);
    DATATYPE x0=1; DATATYPE x1=1; DATATYPE sum=0;
    for (i=0;i<limit; i+=3) {
        x0*=data[i];
        x1*=data[i+1];
    }
    for(; i<length; i++)
        x0*=data[i];
    *dest=x0+x1;
}
```

Summary/Conclusion

- We have developed a suite of code that performs useful optimization analysis on several architectures, providing a simple way for programmers/scientists to analyze their code before investing in “better” hardware.
- From our tests, we have shown that on average a 1.13 GHz P3 processor performs less clock cycles for the same operations than a 2.4GHz P4 processor. While a 1GHz G4 processor performs almost 10 times more clock cycles than a 1.13 GHz P3 processor, thus questioning the claims of the G4 PPC processor.
- Testing one’s core code with such software can provide a useful measure of expectation before purchasing a new machine, or cluster to run ones code, and also serves as a development bed for testing new machine dependent optimization algorithms.

References

- Computer Systems A Programmer's Perspective. R. Bryant and D. O'Hallaron. Prentice Hall 2003.
- CHUD Documentation