Inline assembly & cycle counters, Gnuplot, LaTeX

CSC469
Fall 2019
Outline

• Inline assembly for cycle counters (get data)
• Using Gnuplot (plot data)
• LaTeX (report)
Measuring cycles

- Get current value of cycle counter
- Compute something
- Get new value of cycle counter
- Get elapsed time (in cycles) by subtraction
static u_int64_t start = 0; /* unsigned, 64-bit integer */
void access_counter(unsigned* hi, unsigned* low);

void start_counter() {
    unsigned hi, lo;
    access_counter(&hi, &lo);
    start = ((u_int64_t)hi << 32) | lo;
}

u_int64_t get_counter() {
    unsigned ncyc_hi, ncyc_lo;
    access_counter(&ncyc_hi, &ncyc_lo);
    return (((u_int64_t)ncyc_hi << 32) | ncyc_lo) - start;
}

• We need inline assembly to implement access_counter().

Cycle counter stored in two 32-bit unsigned ints
Combining hi & lo

• C bit-wise operators are used to combine the high-order 32 bits with the low-order 32 bits into a single 64-bit unsigned integer.

• Step-by-step:

1. Cast hi to 64-bit unsigned int
2. Shift left 32 bits
3. Bitwise-OR with lo

Returned by access_counter
Inline Assembly

• Needed for accessing cycle counters
• Key point: there is no magic
• `asm()` directive tells GCC “emit this assembly code here”
• You give it a template
• Same idea as `printf()` format strings
void access_counter(unsigned *hi, unsigned *lo) {
    asm volatile
        ("rdtsc; movl %%edx, %0; movl %%eax, %1" /* Format
         string */
          : "=r" (*hi), "=r" (*lo) /* Output list */
          : /* No inputs */
          : "%edx", "%eax"); /* Clobber list */
}

• Code only works on an x86 machine compiling with GCC
• Emits rdtsc and two movl instructions
• GCC automatically adds instructions to move symbolic register
  value %0 into *hi and %1 into *lo.
• GCC also adds instructions to save and restore the registers
  that rdtsc clobbers.
• Be careful with out-of-order execution: see cpuid, rdtscp
  instructions.
For more information

• More information about inline assembly available online:

  • Intel Architecture Software Developer Manuals (not for the faint-hearted)
Timing with Cycle Counter

- Need to convert cycles into time
- Determine clock rate of processor
- Count number of cycles required for some fixed number of seconds
- Naïve version:
  ```c
  double GHz;
  struct timespec sleep_time;
  sleep_time.tv_sec = 1; sleep_time.tv_nsec = 0;
  start_counter();
  nanosleep(&sleep_time, NULL);
  GHz = (double)get_counter() / 1.0e9);
  ```
- Too simple. Assumes `nanosleep()` actually sleeps for 1 sec
  - May be less (if interrupted) or more (if heavy load)
Gnuplot

• A graphing tool

• Advantages
  • Scriptable – workflow integration & reproducibility
  • Customize every aspect of the figure
  • Makes pretty pictures
  • Easy to learn by example
  • [http://www.gnuplot.info/](http://www.gnuplot.info/)
Using Gnuplot

1. Program → Performance Logs
2. Performance Logs → Perl/Python/Ruby Script
3. Perl/Python/Ruby Script → Script to Drive Gnuplot → GRAPH
#!/bin/sh
gnuplot << ---EOF---
set title "Activity periods, load = 2"
set xlabel "Time (ms)"
set key off
unset ytics
set terminal postscript eps 10
set size 0.45,0.35
set output "bars.eps"
set object 1 rect from 0, 1 to 8.76, 2 fs empty
set object 2 rect from 8.76, 1 to 11, 2 fc rgb "black" fs solid
set object 3 rect from 11, 1 to 13, 2 fs empty
set object 4 rect from 13, 1 to 14, 2 fc rgb "black" fs solid
set object 5 rect from 14, 1 to 19, 2 fs empty
set object 6 rect from 19, 1 to 31, 2 fc rgb "black" fs solid
set object 7 rect from 31, 1 to 33, 2 fs empty
set object 8 rect from 33, 1 to 34, 2 fc rgb "black" fs solid
plot [0:40] [0:3] 0
---EOF---
Gnuplot example (separate data file)

```bash
set terminal png;
set key off;

set title "Plotting Tools Preference in Computer Science Labs";
set xlabel "Tool Name";
set ylabel "Number of People";

set yrange [0:6];
set output "graph.png";
set style fill solid;
set boxwidth 0.5;

plot "data.dat" using 1:3:xtic(2) with boxes;
```

**set terminal** specifies output format

**Input file “data.dat”**

- 0  Matlab 5
- 1  R    5
- 2  Python 2
- 3  GnuPlot 3
- 4  D3   1
- 5  JMP  1
- 6  Excel 3

**set key off**

**set title** specifies the title of the graph

**set xlabel** and **set ylabel** specify the labels for the x and y axes

**set yrange** specifies the range of the y-axis

**set output** specifies the output file for the graph

**set style fill solid** specifies the fill style for the boxes

**set boxwidth 0.5** specifies the width of the boxes

**plot** command generates the graph
Title, Key, Axes

set title "Descriptive Title of My Graph"
set xlabel "This is the x-axis label"
set ylabel "This is the y-axis label"
set key off
unset xtics
set ytics 2

• key off - no legend (can also use “unset key”)
• unset xtics - don’t print numbers on the x-axis
• set ytics 2 – increment numbers on y-axis by 2
• Many more options!
Output

set terminal postscript eps 10
set size 0.45, 0.35
set output "bars.eps"

• Produce a postscript file called “bars.eps”
• size - width and height of chart
• More options: landscape/portrait, color, font, etc.
set object N rect from x0, y0 to x1, y1 <fill>

• Solid rectangle, <fill> is:
  \text{fc rgb “black” fs solid}

• Empty rectangle, <fill> is:
  \text{fs empty}
Other Plotting Tools

• Informal survey of grad students in systems lab, HCI lab, graphics lab, people doing CS research and writing research papers

• Question: “Which tools do you use to generate performance graphs for your research papers?”

• Gnuplot, pyplot, R, Matlab, ... Ok
  • but please no Excel
Recap

• Write your program with cycle counters
• Print active and inactive times to a log file
• Write a script to read your logs
• Scale the time durations
• Output a sequence of rectangles
• Write the rectangles into boilerplate
• Gnuplot script
LaTeX

• A markup language, like HTML
• Easiest to learn by example
• There’s lots of online documentation:
  • http://www.ctan.org/
  • http://tug.ctan.org/info/lshort/english/lshort.pdf
LaTeX – General Structure

\documentclass[options]{class}
\usepackage[pkgs]{pkgs}
definitions
\begin{document}
text
definitions
\end{document}
LaTeX - Sectioning

\section{Introduction}
\label{intro}

\section{Background}

\subsection{Ancient History}

\subsubsection{Batch Processing}

• LaTeX automatically adds numbering
  • So you can move sections / subsections around and not have to update the numbers yourself!
LaTeX – Figure

\begin{figure}
\centering
\begin{tabular}{ccc}
\text{command} & \text{optional} & \text{required} \\
\includegraphics[scale=1.25]{random.eps} & & \\
\caption{Random periods (red).} & & \\
\label{fig:random} & & \\
\end{tabular}
\end{figure}

• LaTeX numbers the figures, adds numbers to the caption.
  • Actual caption from markup becomes something like “Figure 1: Random periods (red).”
LaTeX – Cross-referencing

• Use \label anywhere that you might want to refer to from elsewhere in the document
  • E.g. \section{Introduction} \label{sec:intro}
  • E.g. \label{fig:random} on previous slide

• Use \ref{somelabel} to refer to labeled item
  • LaTeX replaces \ref{somelabel} with the section, subsection, or figure number, but does not add other text
  • E.g., in document “Figure \ref{fig:random} in Section \ref{sec:intro} shows ...”
    • Latex generates something like “Figure 1 in Section 1 shows...” (actual numbers may vary)

• Use \pageref{somelabel} to generate page number that item appears on
LaTeX – Compiling

• Commonly:

$ \text{latex } report\.tex$

$ \text{latex } report\.tex$

$ \text{dvips -o report\.ps report\.dvi}$

$ \text{ps2pdf report\.ps}$

• Compile twice!
  • Create a Makefile, or a script to automate this
LaTeX – Compiling

• BibTeX for bibliography:

$ \texttt{latex \ report\_tex}$

$ \texttt{bibtex \ report}$

$ \texttt{latex \ report\_tex}$

$ \texttt{latex \ report\_tex}$

1. LaTeX: finds all bib references ($\texttt{\cite}$), figure entries, sections, etc. $\Rightarrow$ aux file

2. BibTeX: creates an initial set of text for the bib entries $\Rightarrow$ bbl

3. LaTeX: populates the document with bib refs + update aux

4. LaTeX: knows what the correct labels are now and includes them in the final document.
LaTeX – Compiling

• OR directly using pdflatex:

$ pdflatex report.tex
$ bibtex report
$ pdflatex report.tex
GUI Alternatives

• LyX (Cross-platform) http://www.lyx.org/
• TexnicCenter (Windows) http://www.texniccenter.org/
  • needs MikTex or TexLive distributions of Latex:
    • http://miktex.org/
    • http://www.tug.org/texlive/
• TexMaker (Mac/Linux)
• TexShop
• Overleaf (online editor) and others
Report

• We’re doing science!
• Read the assignment, then read it again
• Think carefully about the different sections
• Describe the experiment details and results – but not just a data dump
  • Aim for reproducibility
• Discussion of results
• More to come in the next weeks
Report – Common Issues

• Units. ms? μs? MHz? GHz?
  • Proper place for decimal point or scientific notation

• Cache hierarchy – cache size vs cacheline size. Which levels are per-core and which are shared

• Pay attention to basic things like header numbers/formatting, paragraph spacing, justification (evenly spaced). A lot of these issues go away if you use LaTeX.

• Have a title (which includes your name/group)

• Use a professional/technical tone ("The benchmark was executed on teach.cs server" as opposed to "and then I ran it on the lab server")

• Other details – figures numbered, placed correctly, and referenced in text (...Figure 2 shows...), no rasterized images, screenshots, direct copy-paste of all the terminal output...and lastly, spelling and grammar