Inline assembly, cycle counters, Gnuplot, LaTeX
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;
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.
Inline Assembly

• Needed for accessing cycle counters
• Key point: there is no magic
• \texttt{asm()} directive tells GCC “emit this assembly code here”
• You give it a template
• Same idea as \texttt{printf()} format strings
Accessing the Cycle Counter

```c
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.
- Careful with newer processors and out-of-order execution: see `cpuid`, `rdtscp` instructions.
For more information

• More information about inline assembly available online:

  
  • [http://www.ibiblio.org/gferg/ldp/GCC-Inline-Assembly-HOWTO.html#ss5.3](http://www.ibiblio.org/gferg/ldp/GCC-Inline-Assembly-HOWTO.html#ss5.3)

• 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 MHz;
int sleep_time = 10;
start_counter();
sleep(sleep_time);
MHz = get_counter() / (sleep_time * 1e6);
```

• This is a bit too simple
• Assumes sleep() actually sleeps for 10 seconds
• 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/
Using Gnuplot

Program → Performance Logs → Perl/Python/Ruby Script → Script to Drive Gnuplot → GRAPH
Gnuplot example

#!/bin/sh

gnuplot << ---EOF---
set title "Activity periods, load = 2"
set xlabel "Time (ms)"
set nokey
set noytics
set term 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---
Title, Key, etc.

set title “Activity periods, load = 2”
set xlabel “Time (ms)”
set nokey
set noytics

• title - the main title
• xlabel - the label of the x-axis
• nokey - no legend
• noytics - don’t print numbers on the y-axis, since we aren’t using it
Output

set term 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:
  fc rgb “black” fs solid

• Empty rectangle, <fill> is:
  fs empty
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
Gnuplot example

• On course website

• A contrived example of automating running an experiment, parsing the output, and generating a plot.

• See README.txt for details.
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?”
- R, Matlab, ..., but please no Excel
LaTeX

• A markup language, like HTML
• Easiest to learn by example
• There’s lots of online documentation:
  • [http://www.ctan.org/](http://www.ctan.org/)
LaTeX – General Structure

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

\begin{figure}
\centering

\begin{tabular}{c c c}
\textbf{command} & \textbf{optional} & \textbf{required} \\
\end{tabular}

\includegraphics[scale=1.25]{random.eps}
\caption{Random periods (red).}
\label{fig:random}
\end{figure}
LaTeX – Compiling

• Commonly:

$ latex report.tex$
$ latex report.tex$
$ dvips -o report.ps report.dvi$
$ ps2pdf report.ps$

• Compile twice!
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 (\cite), figure entries, sections, etc. => aux file

2. BibTeX: creates an initial set of text for the bib entries => 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/](http://www.lyx.org/)

• TexnicCenter (Windows) [http://www.texniccenter.org/](http://www.texniccenter.org/)
  • needs MikTex or TexLive distributions of Latex:
    • [http://miktex.org/](http://miktex.org/)
    • [http://www.tug.org/texlive/](http://www.tug.org/texlive/)

• TexMaker (Mac/Linux)

• TexShop

• ShareLatex (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 – chche size vs cacheline size. Which levels are per-core and which are shared

• Choice of threshold – either determined experimentally (explain and state typical values) or based on some references (perhaps lower and upper bounds)
  • “Makes the graph look good” is not valid

• 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 CDF server" as opposed to "and then I ran the benchmark on CDF")

• 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