Linux – Performance Analysis

3/27/2025

Overview

- How much time does the code take?
  - Basic measurement provides coarse-grain information

- Which part of the code uses the most time?
  - That's the best place to start optimization for speed, as it has the largest impact

Measuring Total Execution Time

Timing Measurement with clock_gettime

- Does not include time taken by other processes
- Time reported in nanoseconds

- #include <time.h>
- Data Type: struct timespec
  - time_t tv_sec: number of whole seconds of elapsed time.
  - long int tv_nsec: Rest of the elapsed time in nanoseconds.

- Function: clock_gettime(clockid_t clk_id, struct timespec *tp)
  - clk_id selects which time to measure
    - CLOCK_REALTIME: System-wide realtime clock.
    - CLOCK_MONOTONIC: Represents monotonic time since some unspecified starting point.
    - CLOCK_PROCESS_CPUTIME_ID: High-resolution per-process timer from the CPU.
    - CLOCK_THREAD_CPUTIME_ID: Thread-specific CPU-time clock.
  - Returns 0 for success, -1 for failure

Example: Speed/Scalar/SG1/main.c

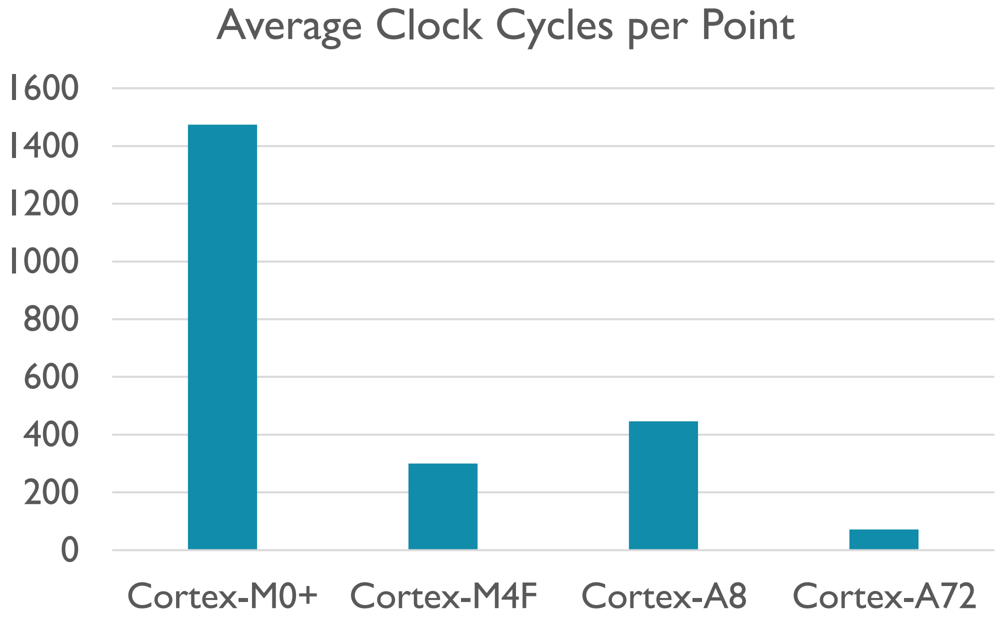
```
struct timespec start, end;
unsigned long diff, total=0; // times in ns

clock_gettime(CLOCK_THREAD_CPUTIME_ID, &start);
Find_Nearest_Waypoint(cur_pos_lat, cur_pos_lon,  &dist, &bearing, &name);
clock_gettime(CLOCK_THREAD_CPUTIME_ID, &end);
diff = 1e9 * (end.tv_sec - start.tv_sec) + end.tv_nsec - start.tv_nsec;
```

Spherical Geometry (SG) Performance Across Processors

- Unsurprisingly, Cortex-A processors much faster than Cortex-M

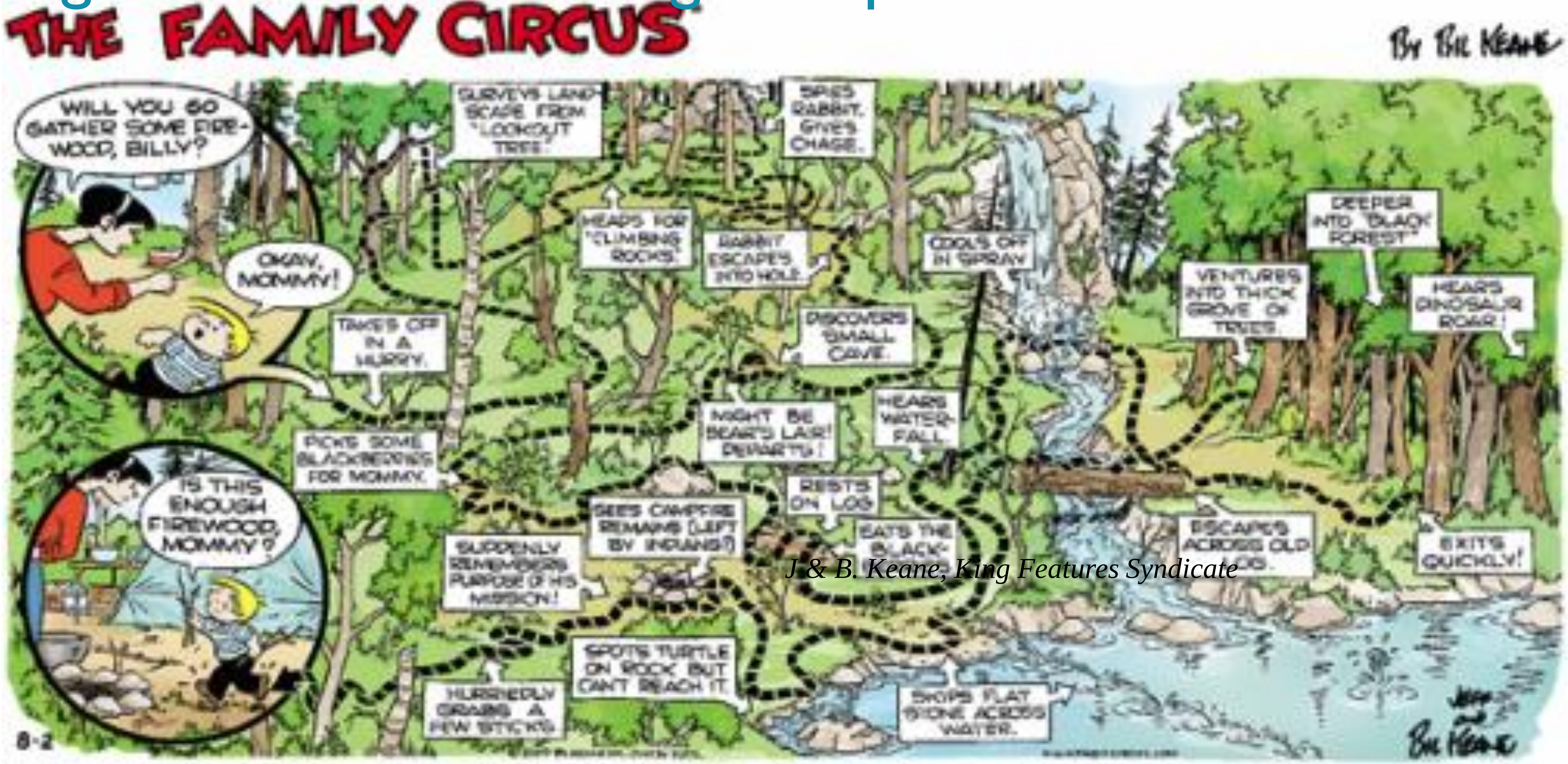
	CM0+ @ 48 MHz		CM4F @ 120 MHz		CA8 @ 1 GHz		CA72 @ 1.5 GHz		
Pipe Stages	2		3		18+		16+		
Version	Time/Pt	Clocks/Pt	Time/Pt	Clocks/Pt	Time/Pt	Clocks/Pt	Total Time	Time/Pt	Clocks/Pt
1	1.60E-03	76800	3.18E-04	38160	1.01E-05	10100	9.60E-05	5.89E-07	883.4
13	3.07E-05	1473.6	2.50E-06	300	4.46E-07	446	7.80E-06	4.78E-08	71.7
						Why so bad? Worse than CM4F!			

- Factor out clock speed to get clock cycles per point
  - Can see efficiency of architecture and microarchitecture
  - Big improvement: 1474 to 72 cycles
- Why does Cortex-A8 perform worse than Cortex-M4F?



Profiling the Distribution of Execution Time in Code

Profiling: How Does The Program Spend Its Time?


THE FAMILY CIRCUS
By Bil Keane
WILL YOU GO GATHER SOME FIRE-WOOD, BILLY?
OKAY, MOMMY!
TAKES OFF IN A HURRY.
SURVEYS LANDSCAPE FROM "LOOKOUT TREE."
HEADS FOR "CLIMBING ROCKS."
RABBIT ESCAPES INTO HOLE.
SPIES RABBIT, GIVES CHASE.
DISCOVERS SMALL CAVE.
MIGHT BE BEAR'S LAIR! DEPARTS!
HEARS WATER-FALL
COOLS OFF IN SPRAY
VENTURES INTO THICK GROVE OF TREES
DEEPER INTO "BLACK FOREST"
HEARS DINOSAUR ROAR!
EXITS QUICKLY!
ESCAPES ACROSS OLD LOG
RESTS ON LOG
EATS THE BLACK-
SKIPS FLAT STONE ACROSS WATER.
SPOTS TURTLE ON ROCK BUT CAN'T REACH IT.
SEES CAMPFIRE REMAINS (LEFT BY INDIANS?)
PICKS SOME BLACKBERRIES FOR MOMMY.
SUDDENLY REMEMBERS PURPOSE OF HIS MISSION!
HURRIEDLY GRABS A FEW STICKS
IS THIS ENOUGH FIREWOOD, MOMMY?
8-2
Jeff and Bil Keane

J & B. Keane, King Features Syndicate

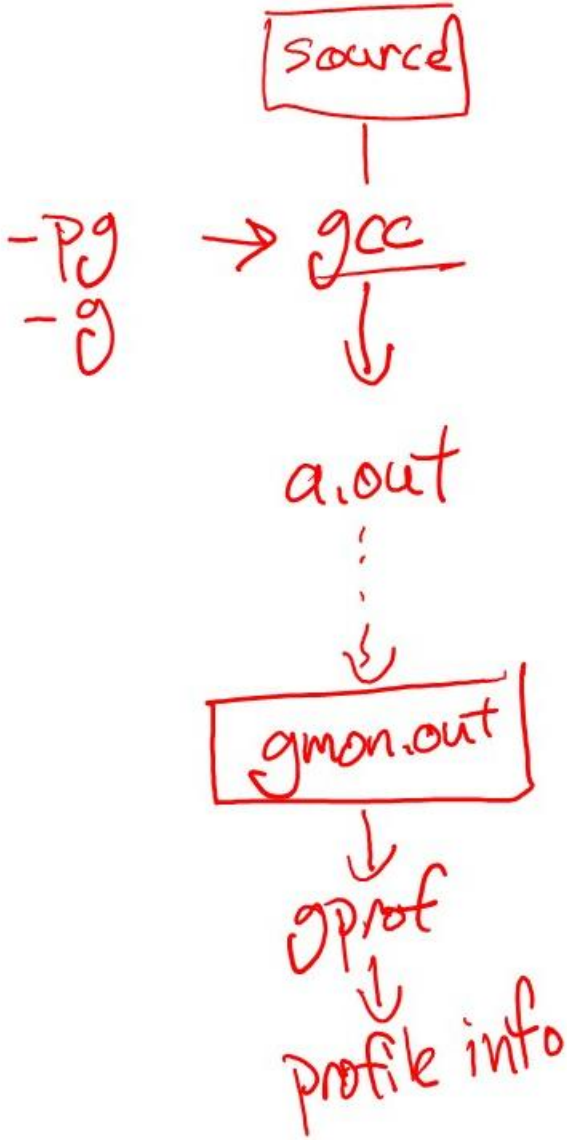
- **What** is the program really doing? Anything unexpected or extra?
- **How** is the program doing it? Is it reasonably efficient?

- The 80/20 rule, Pareto Principle, Juran's Principle
- Tools: gprof and perf

PROFILING WITH GPROF

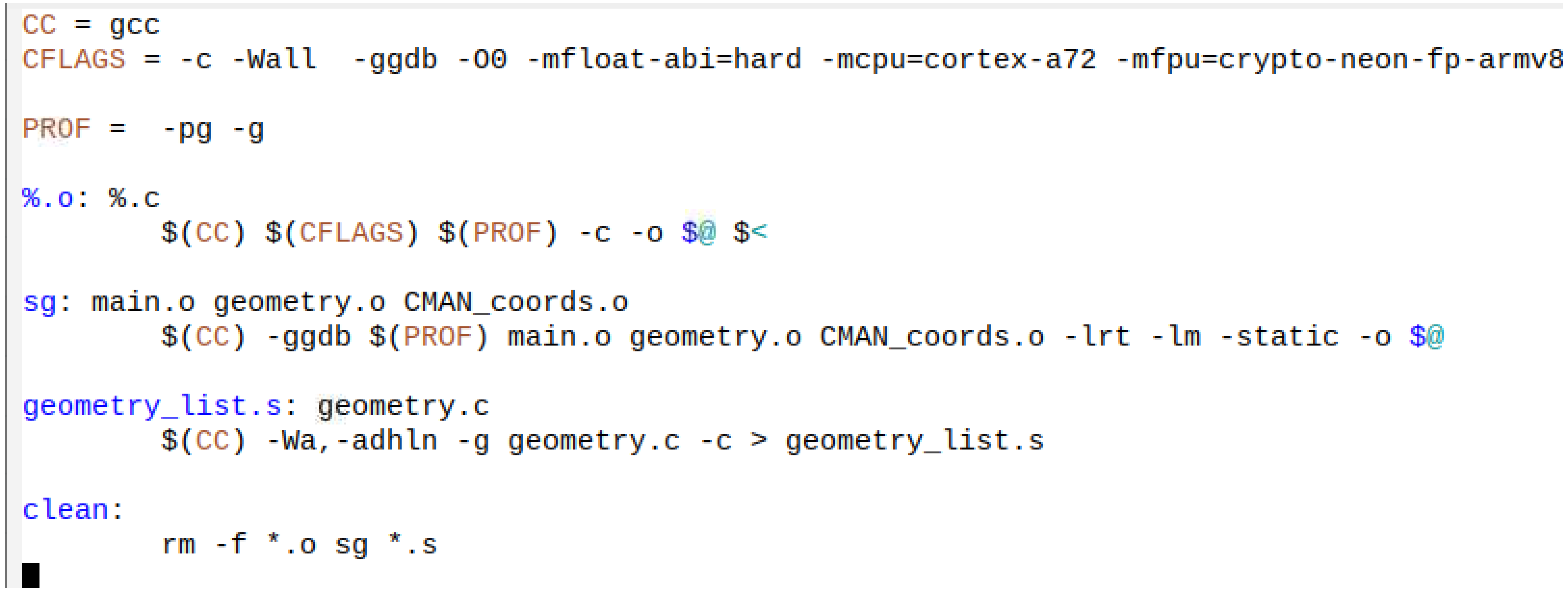
Profiling with GProf

- GNU tool for profiling a program to determine which functions dominate the execution time
  - https://sourceware.org/binutils/docs/gprof/
- Basic process
  - Build your program with gprof profiling support
    - Modifies program (adding *instrumentation code*) to generate execution profile raw data file (gmon.out) when it runs
  - Run your program
    - This also generates gmon.out
  - Use gprof to process the profile raw data (gmon.out) and generate the profile



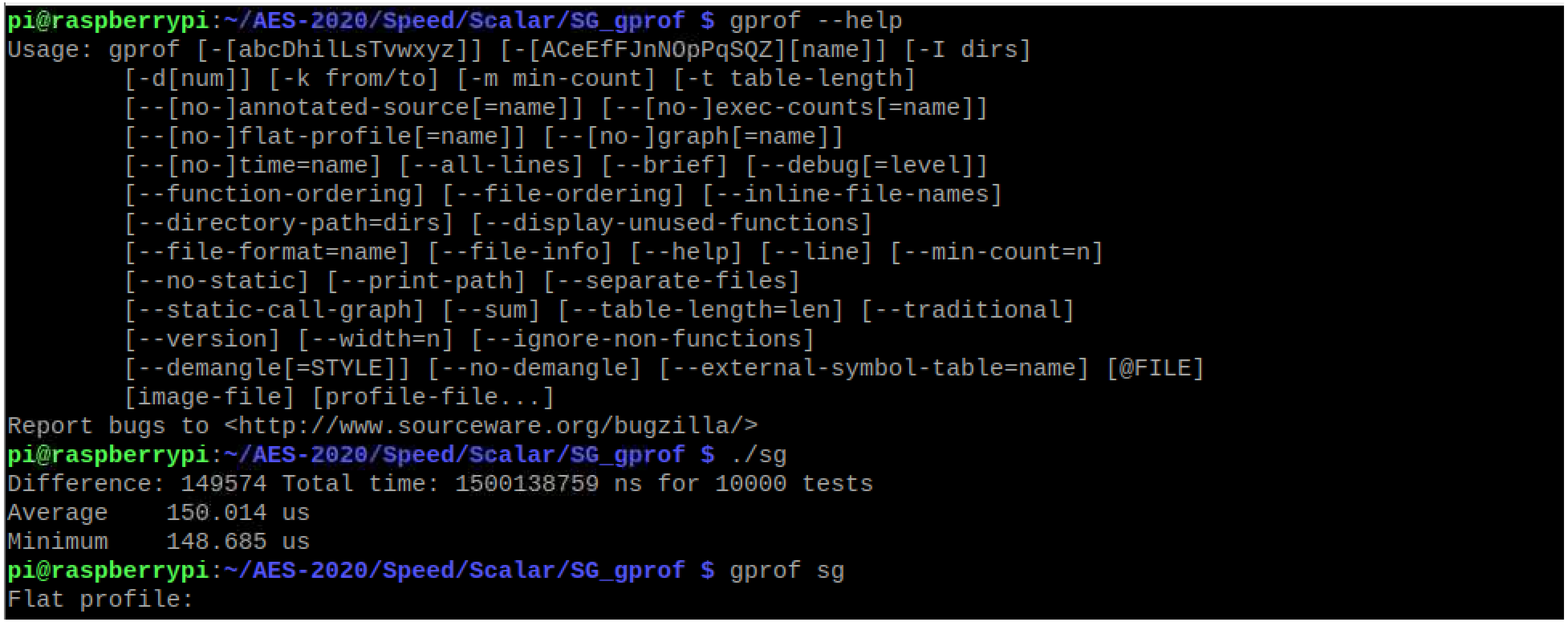
Build Settings Needed for GProf

- See Speed/Scalar/SG_gprof
- Compile ***and*** link program with gcc/g++ profiling options
  - **-pg** to include profiling code (instrumentation)
  - **-g** to support line-by-line profiling
- Makefile must have these options twice
  - when compiling the source files (.c->.o)
  - when linking them (.o->test_program)



```
CC = gcc
CFLAGS = -c -Wall  -ggdb -O0 -mfloat-abi=hard -mcpu=cortex-a72 -mfpu=crypto-neon-fp-armv8

PROF =  -pg -g

%.o: %.c
	$(CC) $(CFLAGS) $(PROF) -c -o $@ $<

sg: main.o geometry.o CMAN_coords.o
	$(CC) -ggdb $(PROF) main.o geometry.o CMAN_coords.o -lrt -lm -static -o $@

geometry_list.s: geometry.c
	$(CC) -Wa,-adhln -g geometry.c -c > geometry_list.s

clean:
	rm -f *.o sg *.s
```

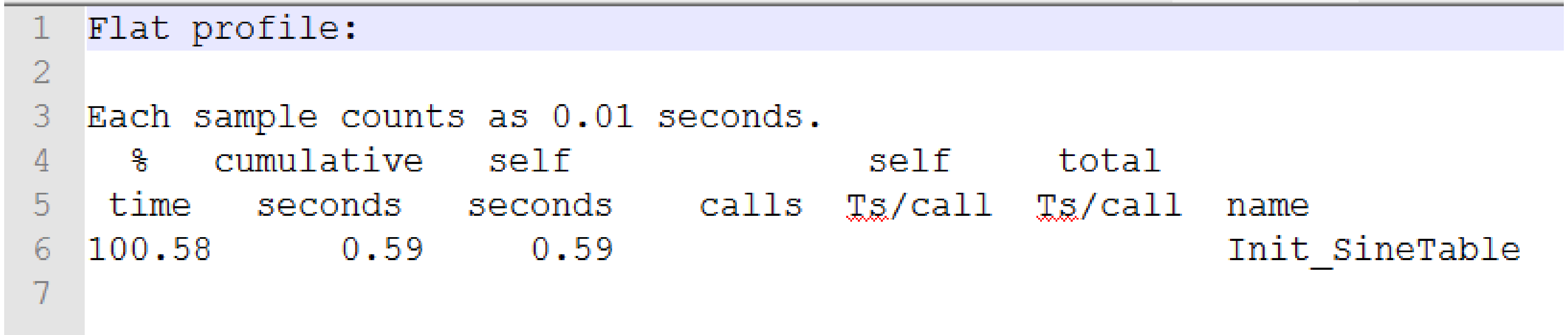
Using GProf



```
pi@raspberrypi:~/AES-2020/Speed/Scalar/SG_gprof $ gprof --help
Usage: gprof [-[abcDhilLsTvwxyz]] [-[ACeEfFJnNOpPqSQZ][name]] [-I dirs]
        [-d[num]] [-k from/to] [-m min-count] [-t table-length]
        [--[no-]annotated-source[=name]] [--[no-]exec-counts[=name]]
        [--[no-]flat-profile[=name]] [--[no-]graph[=name]]
        [--[no-]time=name] [--all-lines] [--brief] [--debug[=level]]
        [--function-ordering] [--file-ordering] [--inline-file-names]
        [--directory-path=dirs] [--display-unused-functions]
        [--file-format=name] [--file-info] [--help] [--line] [--min-count=n]
        [--no-static] [--print-path] [--separate-files]
        [--static-call-graph] [--sum] [--table-length=len] [--traditional]
        [--version] [--width=n] [--ignore-non-functions]
        [--demangle[=STYLE]] [--no-demangle] [--external-symbol-table=name] [@FILE]
        [image-file] [profile-file...]
Report bugs to <http://www.sourceware.org/bugzilla/>
pi@raspberrypi:~/AES-2020/Speed/Scalar/SG_gprof $ ./sg
Difference: 149574 Total time: 1500138759 ns for 10000 tests
Average    150.014 us
Minimum    148.685 us
pi@raspberrypi:~/AES-2020/Speed/Scalar/SG_gprof $ gprof sg
Flat profile:
```

- Run instrumented executable from the shell
  - $ ./sg
  - Generates gmon.out file in directory where the program runs
- Run gprof to analyze gmon.out against executable sg
  - $ gprof sg
  - Generates profile in flat and call graph formats
  - -b option for brief (not verbose) output

Gprof and Libraries



```
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls  Ts/call  Ts/call  name
100.58      0.59     0.59                             Init_SineTable
```

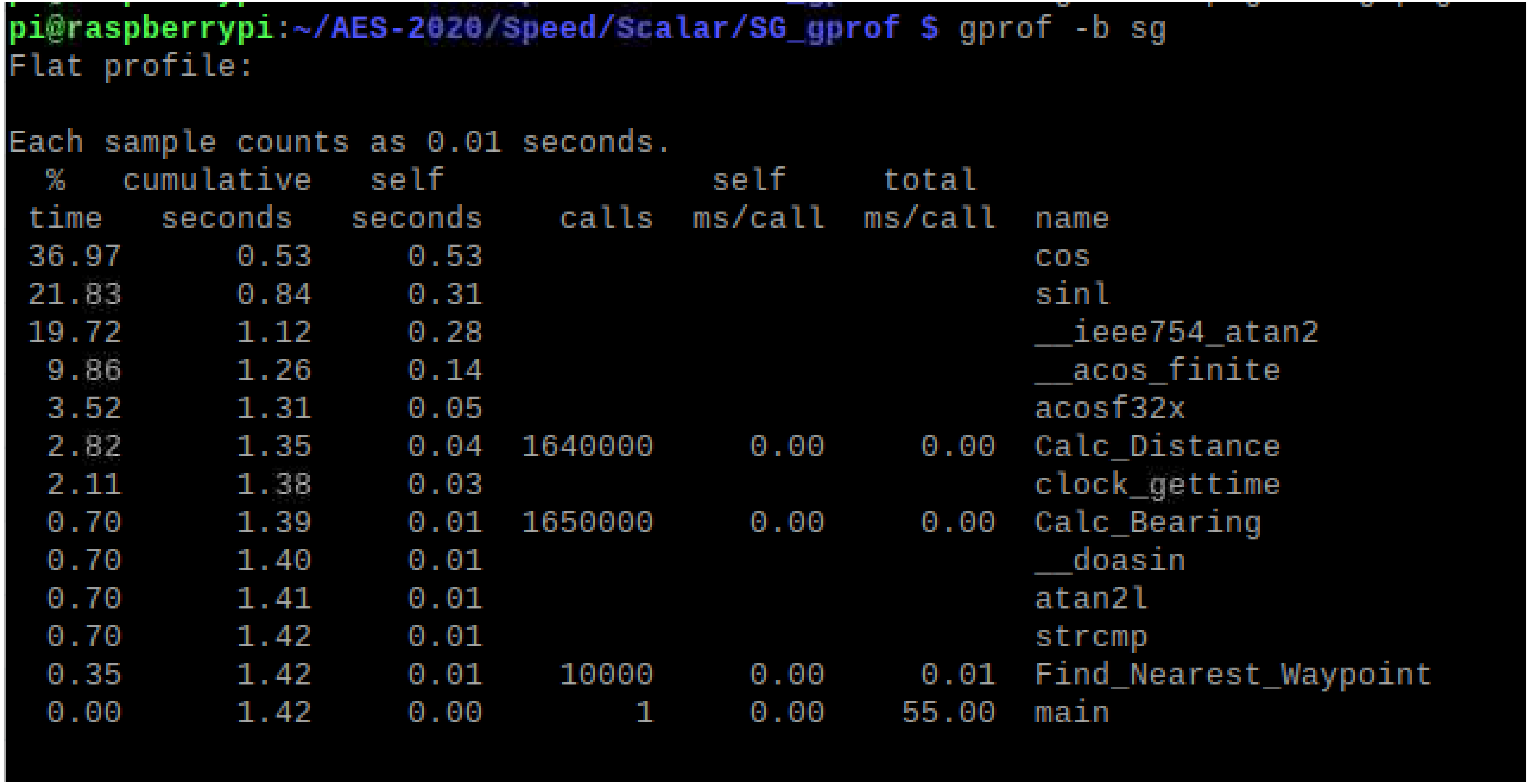
- Not very useful! What's happening?

- This does not handle dynamically linked libraries (default for linker)
  - .so = shared object = dynamically linked library)
  - .a = static library

- Solution: Ensure linker uses **static** libraries
  - Makefile: **-static**

- Verify they are on your system
  - $ sudo find / -name "libm.a"

Flat Profile



```
pi@raspberrypi:~/AES-2020/Speed/Scalar/SG_gprof $ gprof -b sg
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls  ms/call  ms/call  name
 36.97      0.53     0.53                             cos
 21.83      0.84     0.31                             sinl
 19.72      1.12     0.28                             __ieee754_atan2
  9.86      1.26     0.14                             __acos_finite
  3.52      1.31     0.05                             acosf32x
  2.82      1.35     0.04  1640000     0.00     0.00  Calc_Distance
  2.11      1.38     0.03                             clock_gettime
  0.70      1.39     0.01  1650000     0.00     0.00  Calc_Bearing
  0.70      1.40     0.01                             __doasin
  0.70      1.41     0.01                             atan2l
  0.70      1.42     0.01                             strcmp
  0.35      1.42     0.01    10000     0.00     0.01  Find_Nearest_Waypoint
  0.00      1.42     0.00        1     0.00    55.00  main
```

Call Graph

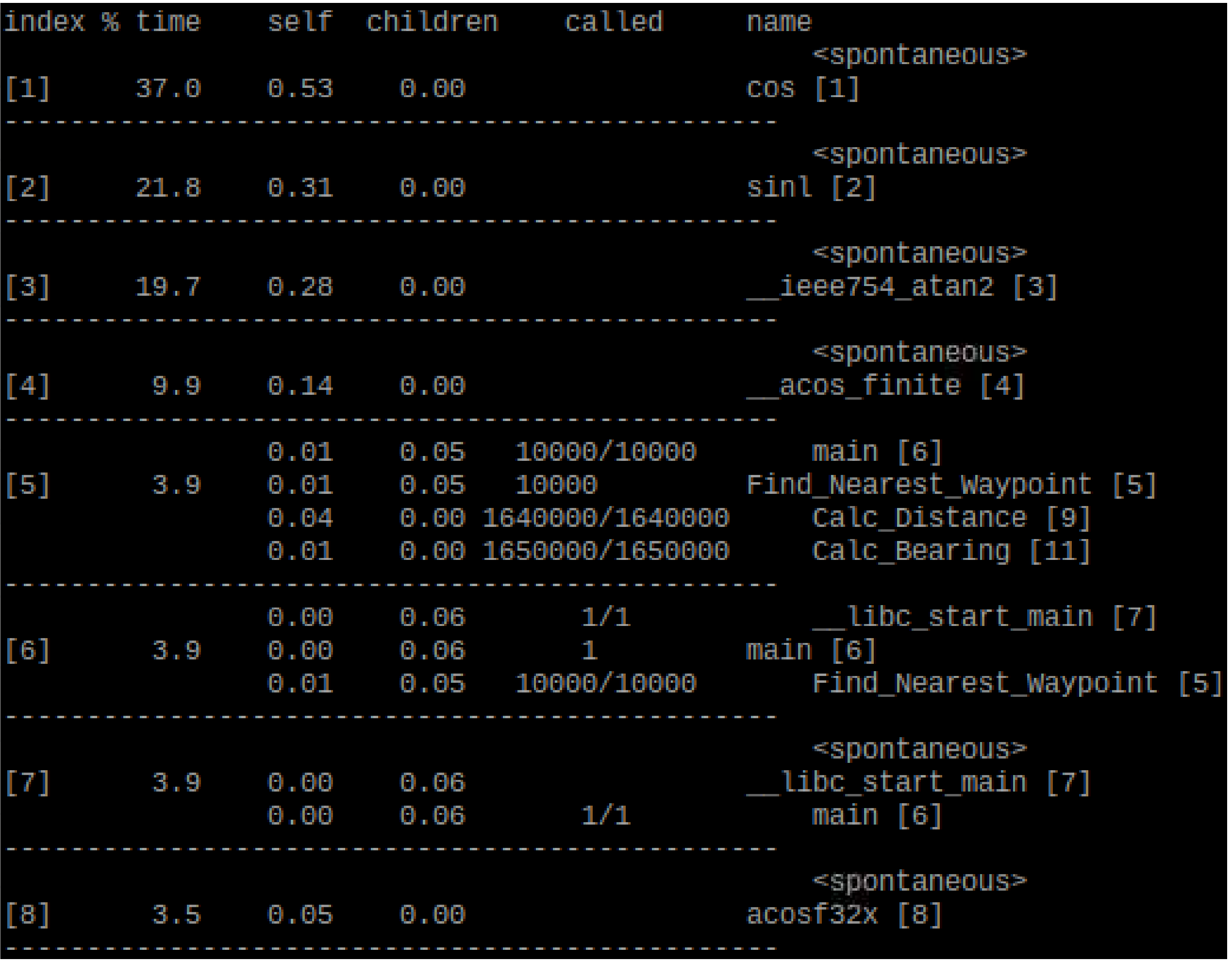
```
index % time    self  children    called     name
                                                 <spontaneous>
[1]     48.4    0.60    0.00                 sincos [1]
-----------------------------------------------
                                                 <spontaneous>
[2]     20.2    0.25    0.00                 __ieee754_atan2 [2]
-----------------------------------------------
-----------------------------------------------
                                                 <spontaneous>
[8]      0.8    0.01    0.00                 Find_Nearest_Waypoint [8]
                0.00    0.00    1000/1000        Calc_Distance [12]
                0.00    0.00    1000/1000        Calc_Bearing [11]
-----------------------------------------------
-----------------------------------------------
                0.00    0.00    1000/1000        Find_Nearest_Waypoint [8]
[11]     0.0    0.00    0.00    1000         Calc_Bearing [11]
-----------------------------------------------
                0.00    0.00    1000/1000        Find_Nearest_Waypoint [8]
[12]     0.0    0.00    0.00    1000         Calc_Distance [12]
-----------------------------------------------
                0.00    0.00       1/1           __libc_start_main [672]
[13]     0.0    0.00    0.00       1         main [13]
-----------------------------------------------
```

Who called current function (spontaneous == don’t know)

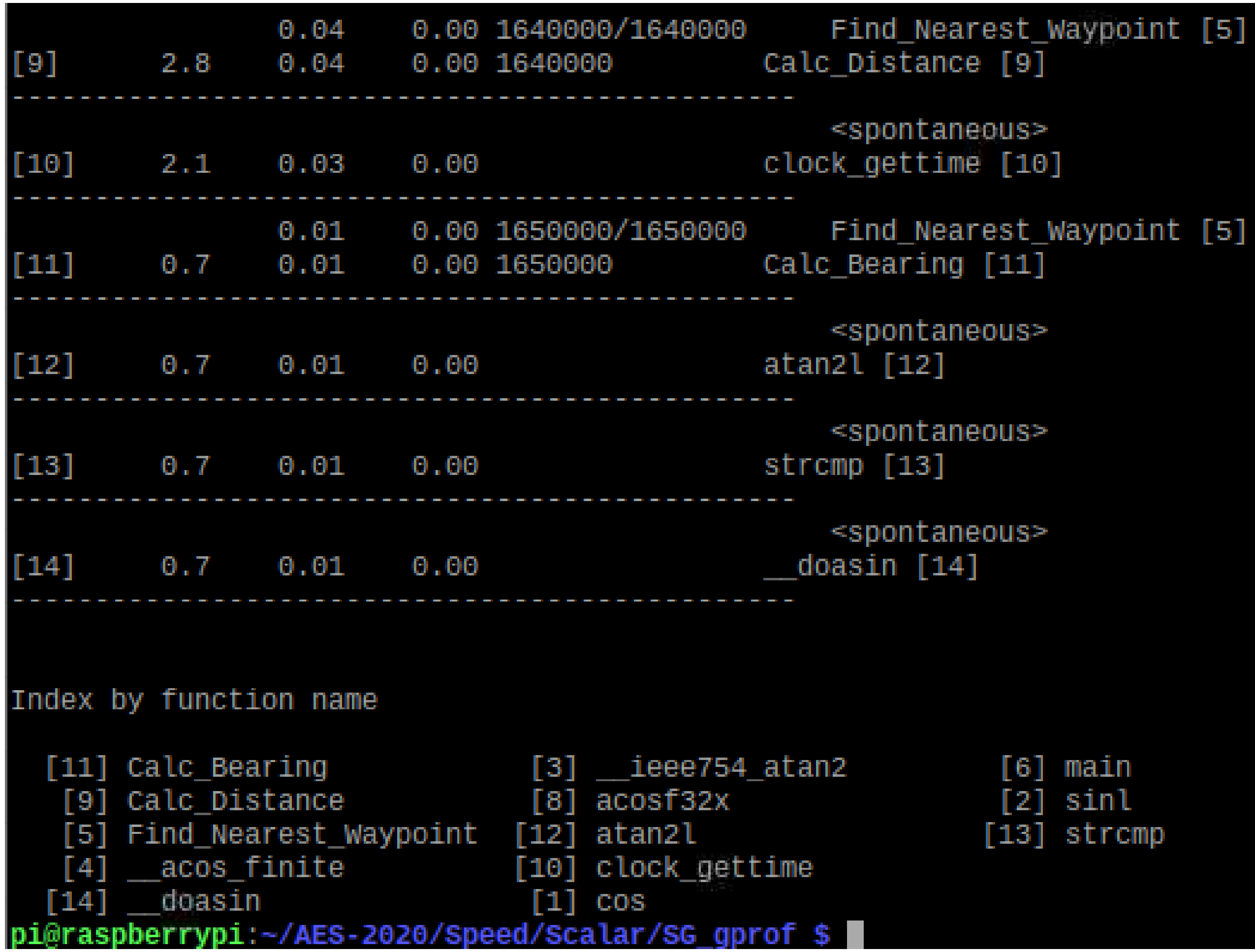
Current function

Children called by current function

Call Graph (in Text)



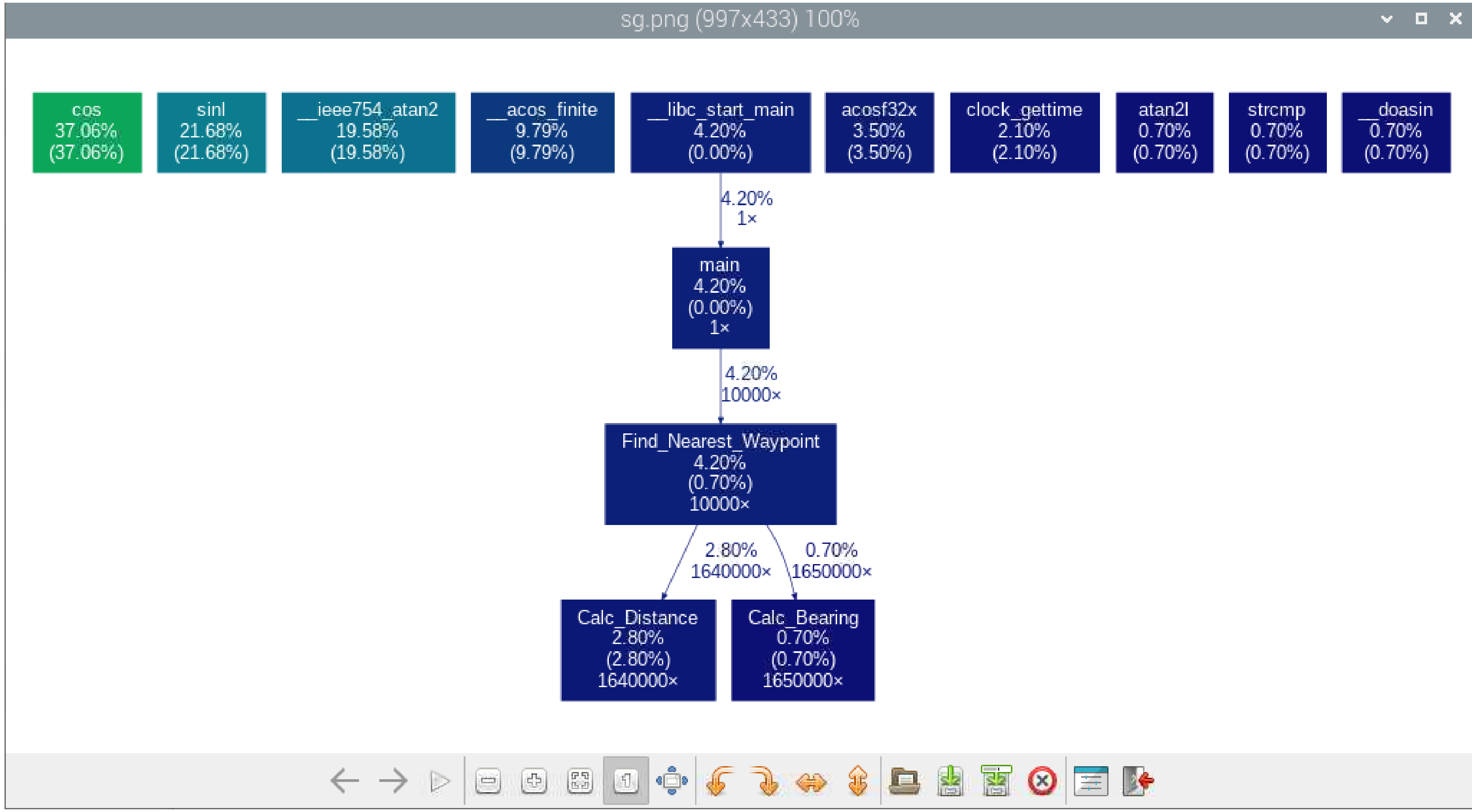
```
index % time    self  children    called     name
                                                 <spontaneous>
[1]     37.0    0.53    0.00                 cos [1]
-----------------------------------------------
                                                 <spontaneous>
[2]     21.8    0.31    0.00                 sinl [2]
-----------------------------------------------
                                                 <spontaneous>
[3]     19.7    0.28    0.00                 __ieee754_atan2 [3]
-----------------------------------------------
                                                 <spontaneous>
[4]      9.9    0.14    0.00                 __acos_finite [4]
-----------------------------------------------
                0.01    0.05   10000/10000       main [6]
[5]      3.9    0.01    0.05   10000         Find_Nearest_Waypoint [5]
                0.04    0.00 1640000/1640000     Calc_Distance [9]
                0.01    0.00 1650000/1650000     Calc_Bearing [11]
-----------------------------------------------
                0.00    0.06       1/1           __libc_start_main [7]
[6]      3.9    0.00    0.06       1         main [6]
                0.01    0.05   10000/10000       Find_Nearest_Waypoint [5]
-----------------------------------------------
                                                 <spontaneous>
[7]      3.9    0.00    0.06                 __libc_start_main [7]
                0.00    0.06       1/1           main [6]
-----------------------------------------------
                                                 <spontaneous>
[8]      3.5    0.05    0.00                 acosf32x [8]
-----------------------------------------------
```



```
                0.04    0.00 1640000/1640000     Find_Nearest_Waypoint [5]
[9]      2.8    0.04    0.00 1640000         Calc_Distance [9]
-----------------------------------------------
                                                 <spontaneous>
[10]     2.1    0.03    0.00                 clock_gettime [10]
-----------------------------------------------
                0.01    0.00 1650000/1650000     Find_Nearest_Waypoint [5]
[11]     0.7    0.01    0.00 1650000         Calc_Bearing [11]
-----------------------------------------------
                                                 <spontaneous>
[12]     0.7    0.01    0.00                 atan2l [12]
-----------------------------------------------
                                                 <spontaneous>
[13]     0.7    0.01    0.00                 strcmp [13]
-----------------------------------------------
                                                 <spontaneous>
[14]     0.7    0.01    0.00                 __doasin [14]
-----------------------------------------------


Index by function name

  [11] Calc_Bearing            [3] __ieee754_atan2         [6] main
   [9] Calc_Distance           [8] acosf32x                [2] sinl
   [5] Find_Nearest_Waypoint  [12] atan2l                 [13] strcmp
   [4] __acos_finite          [10] clock_gettime
  [14] __doasin                [1] cos
pi@raspberrypi:~/AES-2020/Speed/Scalar/SG_gprof $
```

Call Graph Visualization

- Install visualization tools
  - $ sudo pip install gprof2dot
  - $ sudo apt-get install graphviz
- Run tools
  - $ gprof ./sg | gprof2dot > sg.dot
  - $ dot sg.dot –Tpng –o sg.png

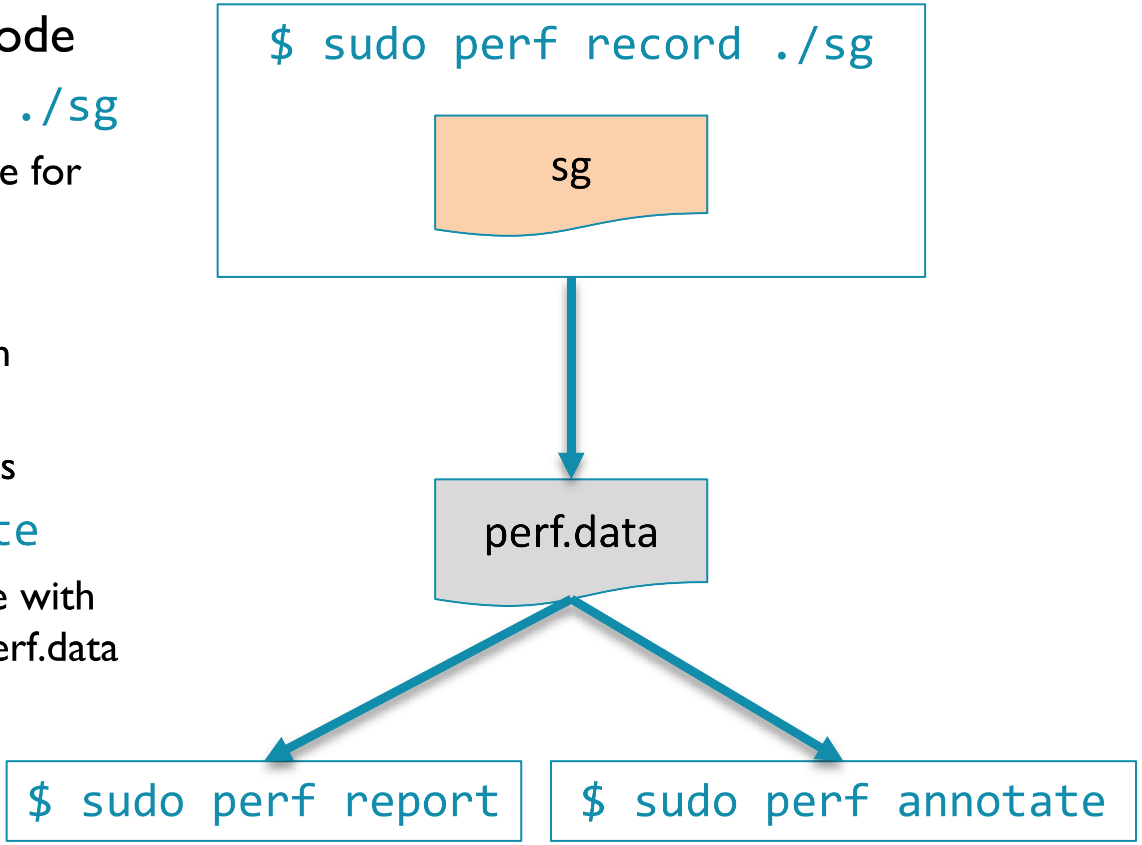

sg.png (997x433) 100%
cos 37.06% (37.06%)
sinl 21.68% (21.68%)
__ieee754_atan2 19.58% (19.58%)
__acos_finite 9.79% (9.79%)
__libc_start_main 4.20% (0.00%)
acosf32x 3.50% (3.50%)
clock_gettime 2.10% (2.10%)
atan2l 0.70% (0.70%)
strcmp 0.70% (0.70%)
__doasin 0.70% (0.70%)
4.20% 1×
main 4.20% (0.00%) 1×
4.20% 10000×
Find_Nearest_Waypoint 4.20% (0.70%) 10000×
2.80% 1640000×
0.70% 1650000×
Calc_Distance 2.80% (2.80%) 1640000×
Calc_Bearing 0.70% (0.70%) 1650000×

PROFILING WITH PERF

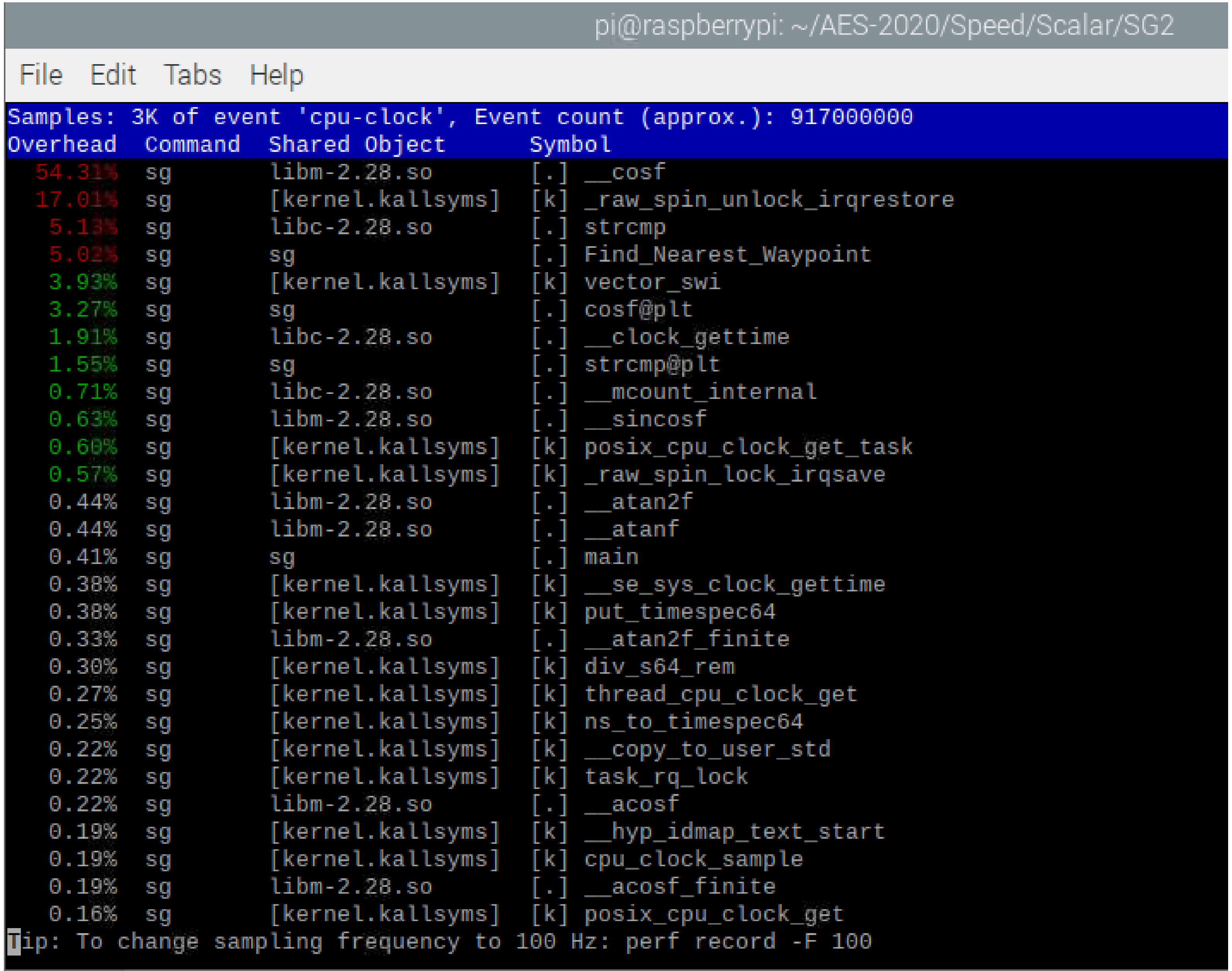
Perf

- Measurement modes
  - Sampling mode (default 1000 Hz)
    - `record, report, annotate`
  - Event-counting mode using software counters in kernel and hardware counters in PMU
    - stat
- Installation
  - sudo apt-get install linux-perf
  - or use Raspberry Pi Preferences->Add/Remove Software

- Basic use in sampling mode
  - `sudo perf record ./sg`
    - Generates perf.data file for analysis
  - `sudo perf report`
    - Generates profile from perf.data file
    - Can annotate functions
  - `sudo perf annotate`
    - Annotates object code with sample counts from perf.data

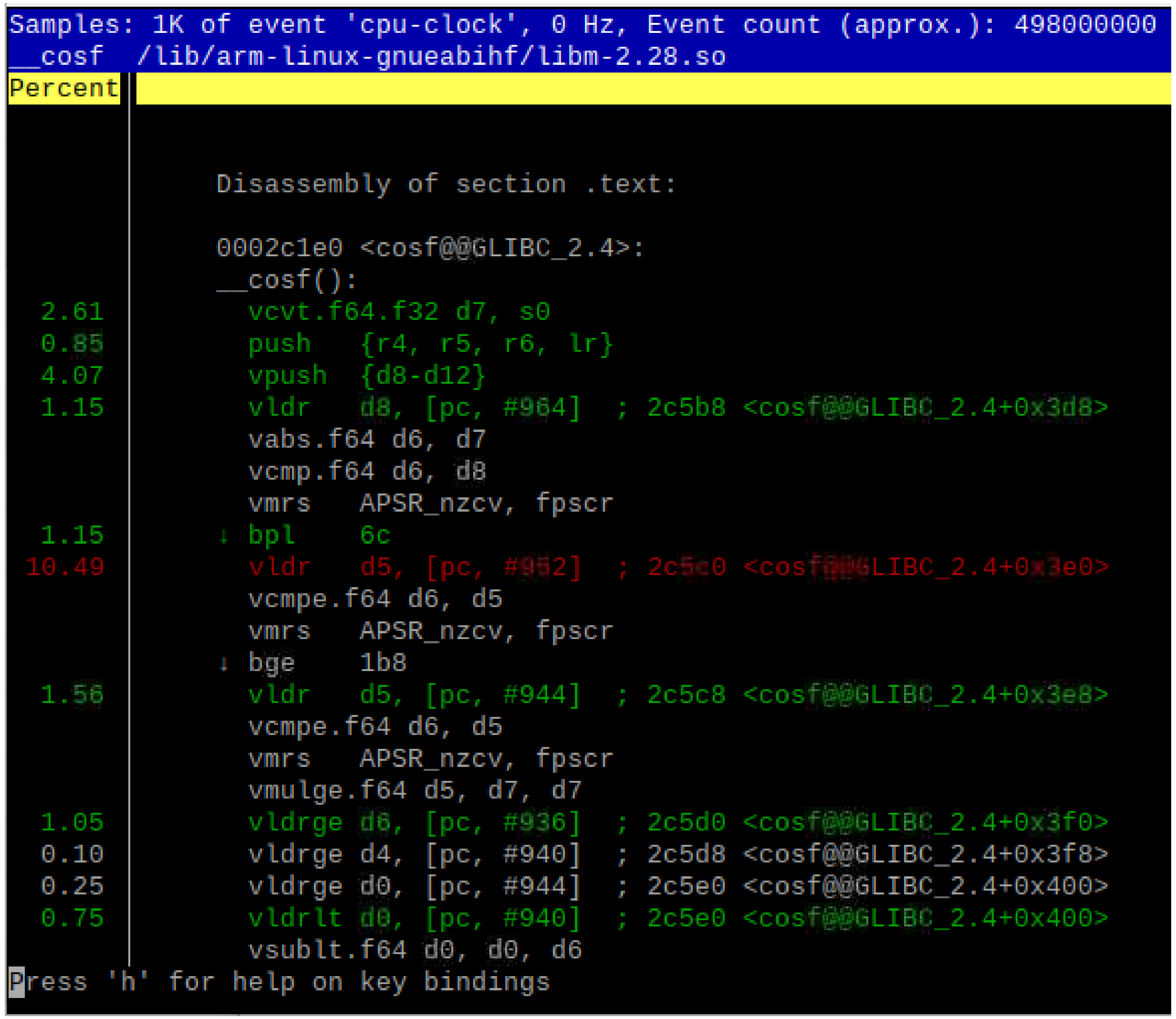


Function Profile: perf report

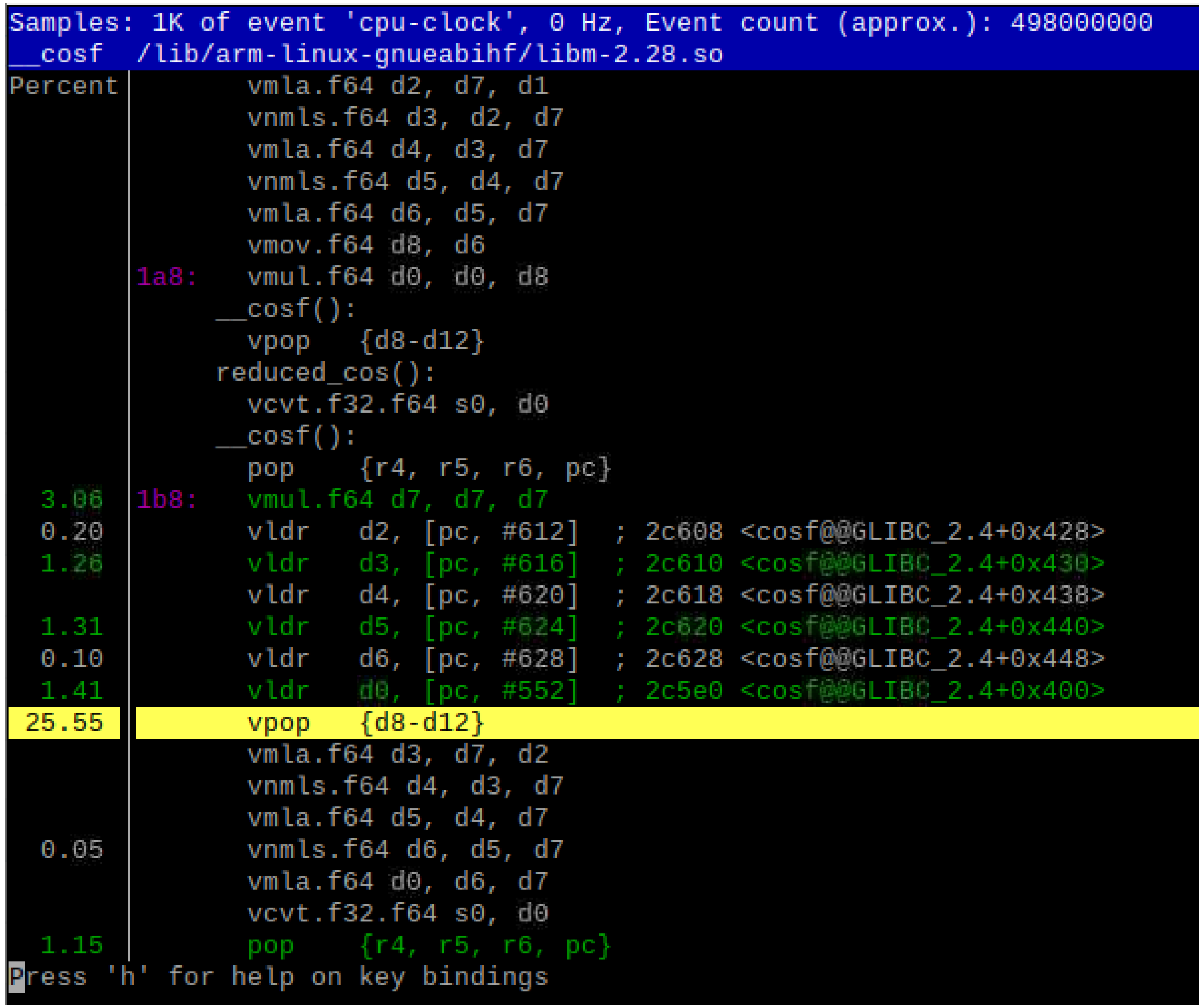


```
Samples: 3K of event 'cpu-clock', Event count (approx.): 917000000
Overhead  Command  Shared Object      Symbol
  54.31%  sg       libm-2.28.so       [.] __cosf
  17.01%  sg       [kernel.kallsyms]  [k] _raw_spin_unlock_irqrestore
   5.13%  sg       libc-2.28.so       [.] strcmp
   5.02%  sg       sg                 [.] Find_Nearest_Waypoint
   3.93%  sg       [kernel.kallsyms]  [k] vector_swi
   3.27%  sg       sg                 [.] cosf@plt
   1.91%  sg       libc-2.28.so       [.] __clock_gettime
   1.55%  sg       sg                 [.] strcmp@plt
   0.71%  sg       libc-2.28.so       [.] __mcount_internal
   0.63%  sg       libm-2.28.so       [.] __sincosf
   0.60%  sg       [kernel.kallsyms]  [k] posix_cpu_clock_get_task
   0.57%  sg       [kernel.kallsyms]  [k] _raw_spin_lock_irqsave
   0.44%  sg       libm-2.28.so       [.] __atan2f
   0.44%  sg       libm-2.28.so       [.] __atanf
   0.41%  sg       sg                 [.] main
   0.38%  sg       [kernel.kallsyms]  [k] __se_sys_clock_gettime
   0.38%  sg       [kernel.kallsyms]  [k] put_timespec64
   0.33%  sg       libm-2.28.so       [.] __atan2f_finite
   0.30%  sg       [kernel.kallsyms]  [k] div_s64_rem
   0.27%  sg       [kernel.kallsyms]  [k] thread_cpu_clock_get
   0.25%  sg       [kernel.kallsyms]  [k] ns_to_timespec64
   0.22%  sg       [kernel.kallsyms]  [k] __copy_to_user_std
   0.22%  sg       [kernel.kallsyms]  [k] task_rq_lock
   0.22%  sg       libm-2.28.so       [.] __acosf
   0.19%  sg       [kernel.kallsyms]  [k] __hyp_idmap_text_start
   0.19%  sg       [kernel.kallsyms]  [k] cpu_clock_sample
   0.19%  sg       libm-2.28.so       [.] __acosf_finite
   0.16%  sg       [kernel.kallsyms]  [k] posix_cpu_clock_get
Tip: To change sampling frequency to 100 Hz: perf record -F 100
```

Instruction Profile: perf annotate



```
Samples: 1K of event 'cpu-clock', 0 Hz, Event count (approx.): 498000000
__cosf  /lib/arm-linux-gnueabihf/libm-2.28.so
Percent


             Disassembly of section .text:

             0002c1e0 <cosf@@GLIBC_2.4>:
             __cosf():
  2.61         vcvt.f64.f32 d7, s0
  0.85         push   {r4, r5, r6, lr}
  4.07         vpush  {d8-d12}
  1.15         vldr   d8, [pc, #964]  ; 2c5b8 <cosf@@GLIBC_2.4+0x3d8>
               vabs.f64 d6, d7
               vcmp.f64 d6, d8
               vmrs   APSR_nzcv, fpscr
  1.15       ↓ bpl    6c
 10.49         vldr   d5, [pc, #952]  ; 2c5c0 <cosf@@GLIBC_2.4+0x3e0>
               vcmpe.f64 d6, d5
               vmrs   APSR_nzcv, fpscr
             ↓ bge    1b8
  1.56         vldr   d5, [pc, #944]  ; 2c5c8 <cosf@@GLIBC_2.4+0x3e8>
               vcmpe.f64 d6, d5
               vmrs   APSR_nzcv, fpscr
               vmulge.f64 d5, d7, d7
  1.05         vldrge d6, [pc, #936]  ; 2c5d0 <cosf@@GLIBC_2.4+0x3f0>
  0.10         vldrge d4, [pc, #940]  ; 2c5d8 <cosf@@GLIBC_2.4+0x3f8>
  0.25         vldrge d0, [pc, #944]  ; 2c5e0 <cosf@@GLIBC_2.4+0x400>
  0.75         vldrlt d0, [pc, #940]  ; 2c5e0 <cosf@@GLIBC_2.4+0x400>
               vsublt.f64 d0, d0, d6
Press 'h' for help on key bindings
```



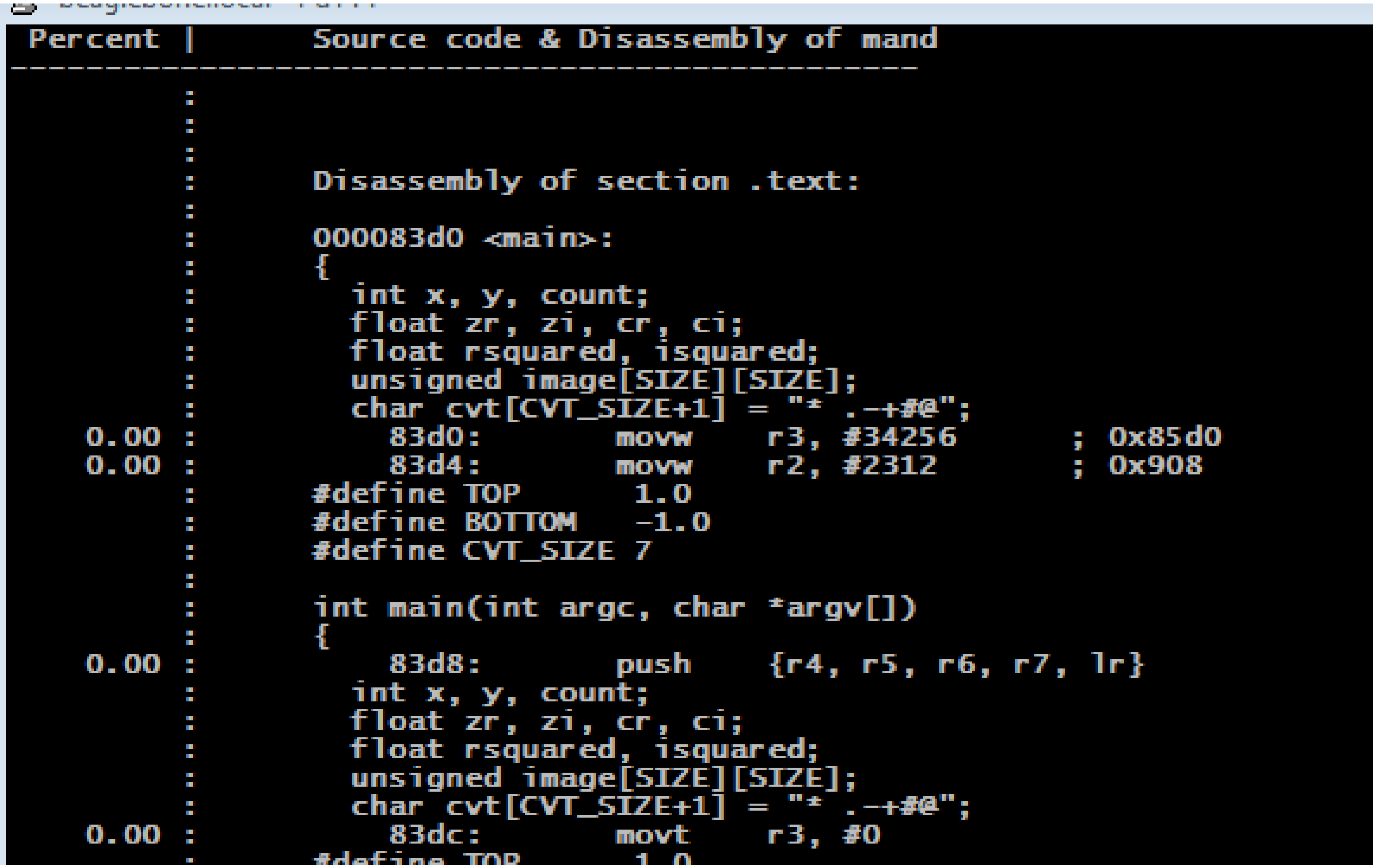
```
Samples: 1K of event 'cpu-clock', 0 Hz, Event count (approx.): 498000000
__cosf  /lib/arm-linux-gnueabihf/libm-2.28.so
Percent          vmla.f64 d2, d7, d1
                 vnmls.f64 d3, d2, d7
                 vmla.f64 d4, d3, d7
                 vnmls.f64 d5, d4, d7
                 vmla.f64 d6, d5, d7
                 vmov.f64 d8, d6
        1a8:     vmul.f64 d0, d0, d8
               __cosf():
                 vpop   {d8-d12}
               reduced_cos():
                 vcvt.f32.f64 s0, d0
               __cosf():
                 pop    {r4, r5, r6, pc}
  3.06  1b8:     vmul.f64 d7, d7, d7
  0.20           vldr   d2, [pc, #612]  ; 2c608 <cosf@@GLIBC_2.4+0x428>
  1.26           vldr   d3, [pc, #616]  ; 2c610 <cosf@@GLIBC_2.4+0x430>
                 vldr   d4, [pc, #620]  ; 2c618 <cosf@@GLIBC_2.4+0x438>
  1.31           vldr   d5, [pc, #624]  ; 2c620 <cosf@@GLIBC_2.4+0x440>
  0.10           vldr   d6, [pc, #628]  ; 2c628 <cosf@@GLIBC_2.4+0x448>
  1.41           vldr   d0, [pc, #552]  ; 2c5e0 <cosf@@GLIBC_2.4+0x400>
 25.55           vpop   {d8-d12}
                 vmla.f64 d3, d7, d2
                 vnmls.f64 d4, d3, d7
                 vmla.f64 d5, d4, d7
  0.05           vnmls.f64 d6, d5, d7
                 vmla.f64 d0, d6, d7
                 vcvt.f32.f64 s0, d0
  1.15           pop    {r4, r5, r6, pc}
Press 'h' for help on key bindings
```

References on How to Use Perf

- Drongowski:
  - Tutorial:
    - *Part 1 demonstrates how to use PERF to identify and analyze the hottest execution spots in a program. It covers the basic PERF commands, options and software performance events.*
    - *Part 2 introduces hardware performance events and demonstrates how to measure hardware events across an entire application. It defines and discusses several useful rates and ratios for performance assessment and analysis.*
    - *Part 3 uses hardware performance event sampling to identify and analyze program hot-spots.*
  - Performance events on Raspberry Pi 4:Tips
- Performance Analysis in Linux: https://www.linux.com/training-tutorials/performance-analysis-linux/
- Good summary with one-liners (example command invocations): http://www.brendangregg.com/perf.html
- Intro: http://www.baptiste-wicht.com/2011/07/profile-applications-linux-perf-tools/
- https://dvinfo.ifh.de/perf
- Exhaustive: https://perf.wiki.kernel.org/index.php/Tutorial
- Use the Source!
  - /usr/src/kernel/tools/perf

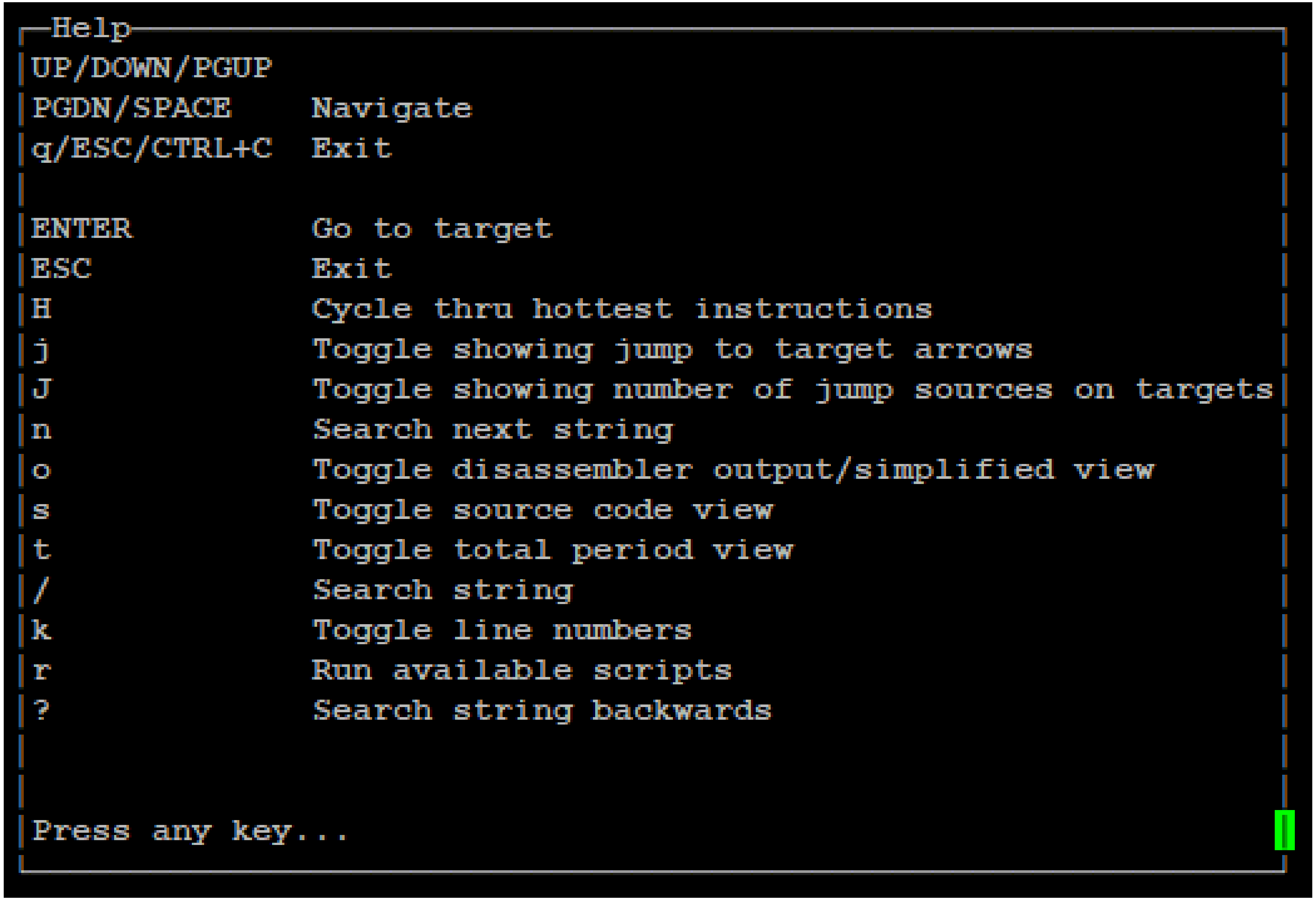
Annotated Mixed Asm & Source Code

- Makefile: include -ggdb option when compiling and linking to get source code listing
- Record a run, then annotate source code with that perf data
- $ sudo perf record ./sg
- $ sudo perf annotate
- Note: may need to change terminal remote character set to ISO-8859:1 1998



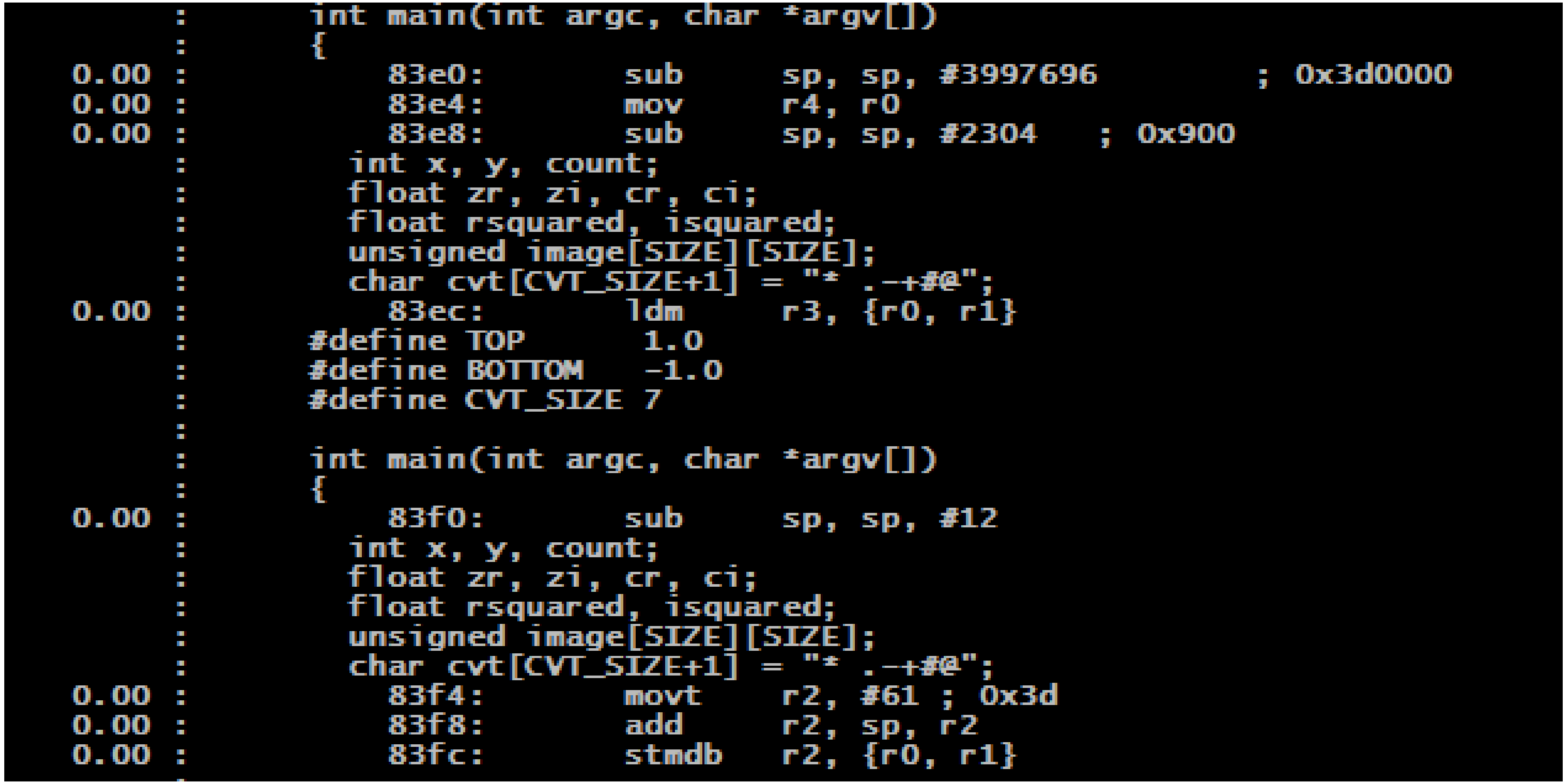
```
 Percent |      Source code & Disassembly of mand
------------------------------------------------
         :
         :
         :
         :
         :      Disassembly of section .text:
         :
         :      000083d0 <main>:
         :      {
         :        int x, y, count;
         :        float zr, zi, cr, ci;
         :        float rsquared, isquared;
         :        unsigned image[SIZE][SIZE];
         :        char cvt[CVT_SIZE+1] = "* .-+#@";
    0.00 :          83d0:       movw    r3, #34256      ; 0x85d0
    0.00 :          83d4:       movw    r2, #2312       ; 0x908
         :      #define TOP     1.0
         :      #define BOTTOM  -1.0
         :      #define CVT_SIZE 7
         :
         :      int main(int argc, char *argv[])
         :      {
    0.00 :          83d8:       push    {r4, r5, r6, r7, lr}
         :        int x, y, count;
         :        float zr, zi, cr, ci;
         :        float rsquared, isquared;
         :        unsigned image[SIZE][SIZE];
         :        char cvt[CVT_SIZE+1] = "* .-+#@";
    0.00 :          83dc:       movt    r3, #0
         :      #define TOP     1.0
```

Perf Annotate User Interface



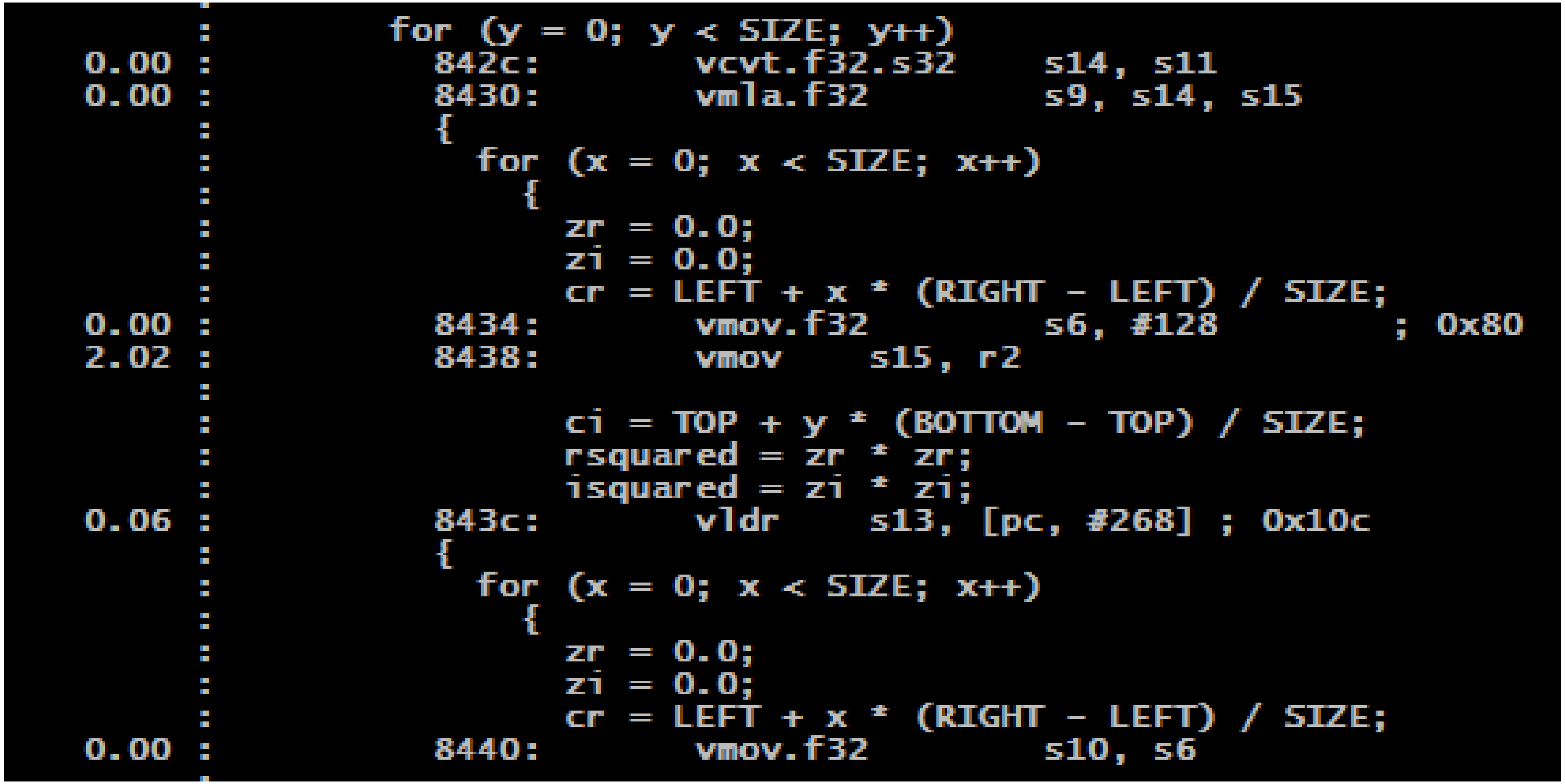
```
Help
UP/DOWN/PGUP
PGDN/SPACE     Navigate
q/ESC/CTRL+C   Exit

ENTER          Go to target
ESC            Exit
H              Cycle thru hottest instructions
j              Toggle showing jump to target arrows
J              Toggle showing number of jump sources on targets
n              Search next string
o              Toggle disassembler output/simplified view
s              Toggle source code view
t              Toggle total period view
/              Search string
k              Toggle line numbers
r              Run available scripts
?              Search string backwards


Press any key...
```

Annotated Main Listing



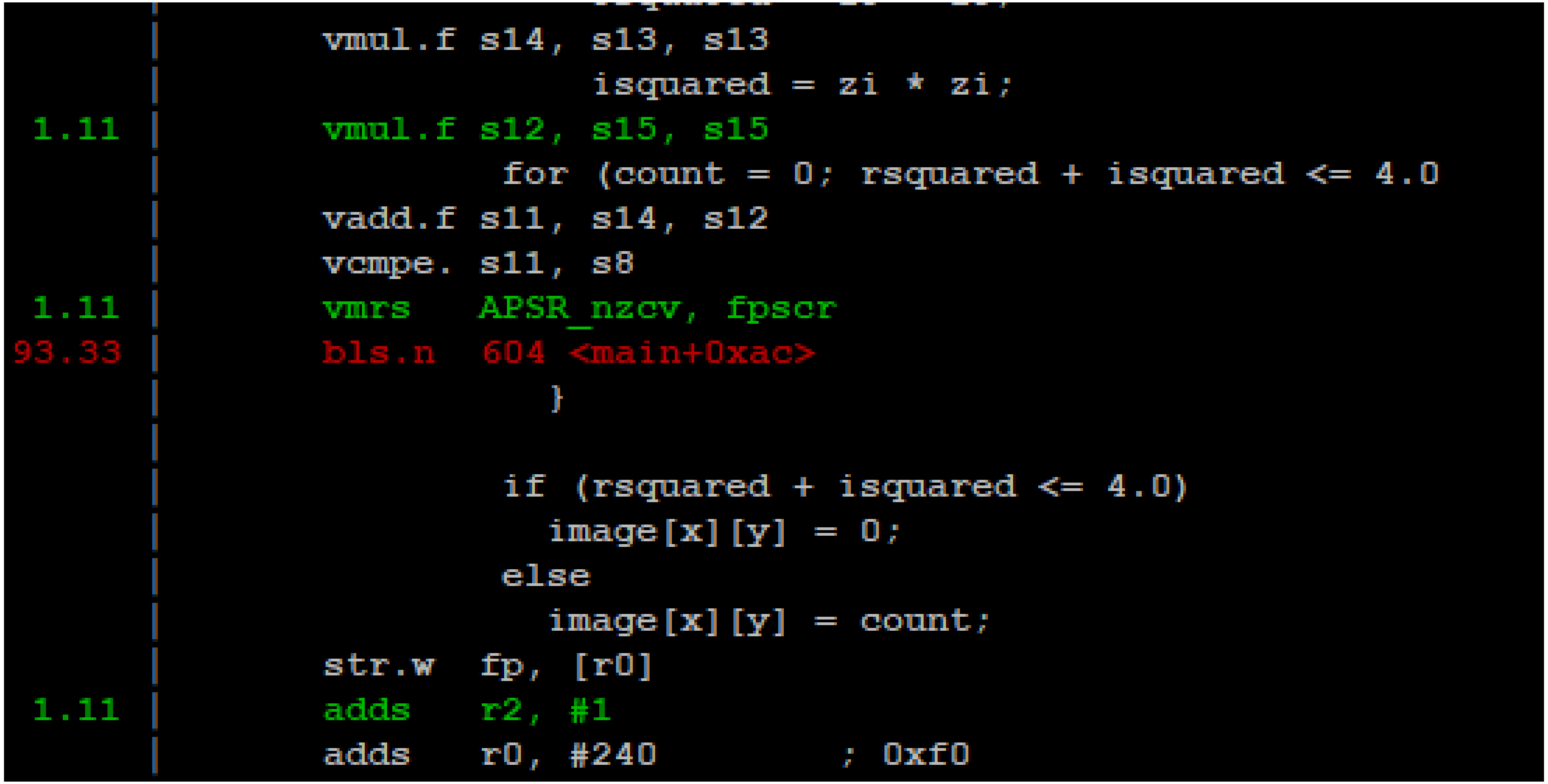
```
     :      int main(int argc, char *argv[])
     :      {
0.00 :          83e0:       sub     sp, sp, #3997696        ; 0x3d0000
0.00 :          83e4:       mov     r4, r0
0.00 :          83e8:       sub     sp, sp, #2304   ; 0x900
     :        int x, y, count;
     :        float zr, zi, cr, ci;
     :        float rsquared, isquared;
     :        unsigned image[SIZE][SIZE];
     :        char cvt[CVT_SIZE+1] = "* .-+#@";
0.00 :          83ec:       ldm     r3, {r0, r1}
     :      #define TOP      1.0
     :      #define BOTTOM   -1.0
     :      #define CVT_SIZE 7
     :
     :      int main(int argc, char *argv[])
     :      {
0.00 :          83f0:       sub     sp, sp, #12
     :        int x, y, count;
     :        float zr, zi, cr, ci;
     :        float rsquared, isquared;
     :        unsigned image[SIZE][SIZE];
     :        char cvt[CVT_SIZE+1] = "* .-+#@";
0.00 :          83f4:       movt    r2, #61 ; 0x3d
0.00 :          83f8:       add     r2, sp, r2
0.00 :          83fc:       stmdb   r2, {r0, r1}
```

More Annotated Listing



```
     :        for (y = 0; y < SIZE; y++)
0.00 :          842c:       vcvt.f32.s32    s14, s11
0.00 :          8430:       vmla.f32        s9, s14, s15
     :          {
     :            for (x = 0; x < SIZE; x++)
     :              {
     :                zr = 0.0;
     :                zi = 0.0;
     :                cr = LEFT + x * (RIGHT - LEFT) / SIZE;
0.00 :          8434:       vmov.f32        s6, #128        ; 0x80
2.02 :          8438:       vmov    s15, r2
     :
     :                ci = TOP + y * (BOTTOM - TOP) / SIZE;
     :                rsquared = zr * zr;
     :                isquared = zi * zi;
0.06 :          843c:       vldr    s13, [pc, #268] ; 0x10c
     :          {
     :            for (x = 0; x < SIZE; x++)
     :              {
     :                zr = 0.0;
     :                zi = 0.0;
     :                cr = LEFT + x * (RIGHT - LEFT) / SIZE;
0.00 :          8440:       vmov.f32        s10, s6
```

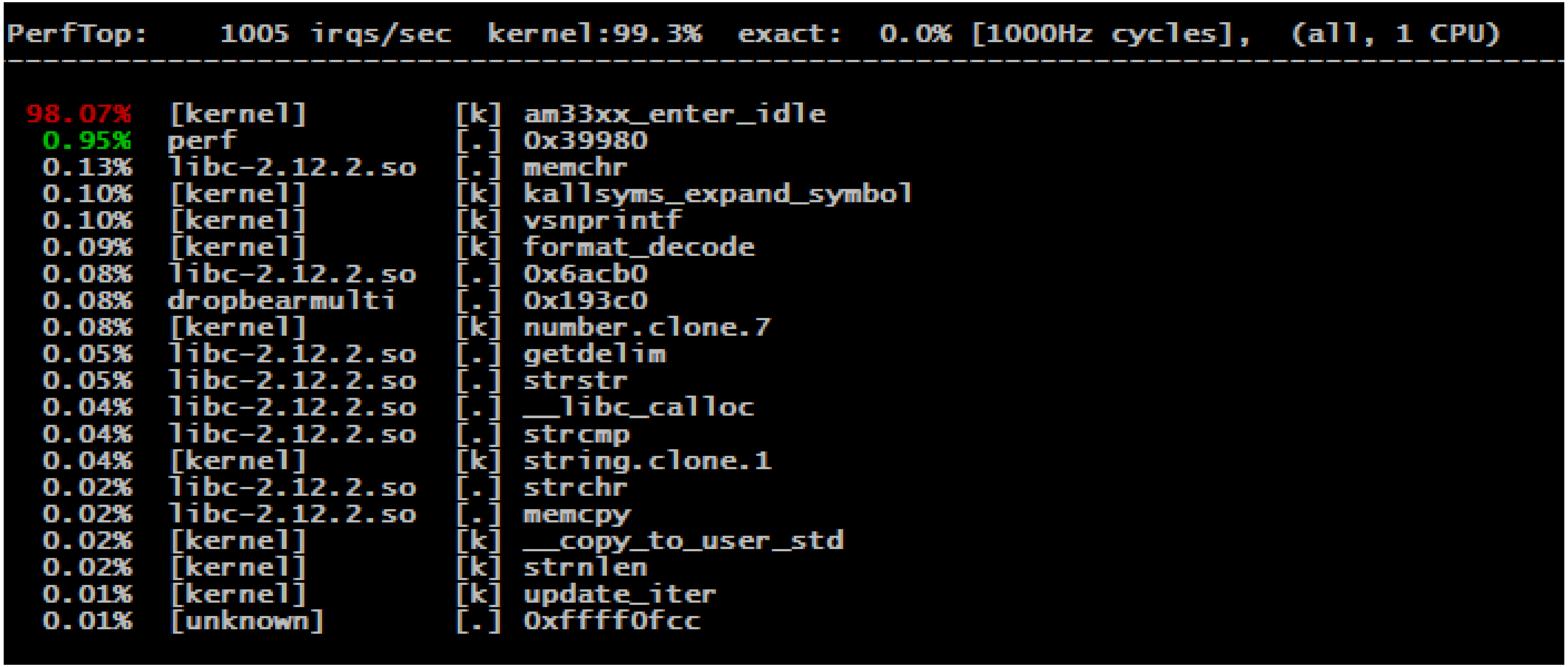
Aha! (on Cortex-A8)



```
       │        vmul.f s14, s13, s13
       │                       isquared = zi * zi;
  1.11 │        vmul.f s12, s15, s15
       │                   for (count = 0; rsquared + isquared <= 4.0
       │        vadd.f s11, s14, s12
       │        vcmpe. s11, s8
  1.11 │        vmrs   APSR_nzcv, fpscr
 93.33 │        bls.n  604 <main+0xac>
       │                    }
       │
       │                   if (rsquared + isquared <= 4.0)
       │                     image[x][y] = 0;
       │                   else
       │                     image[x][y] = count;
       │        str.w  fp, [r0]
  1.11 │        adds   r2, #1
       │        adds   r0, #240          ; 0xf0
```

- bls takes most of the time
- Pipeline stalls after vmrs instruction

perf top (Cortex-A8)



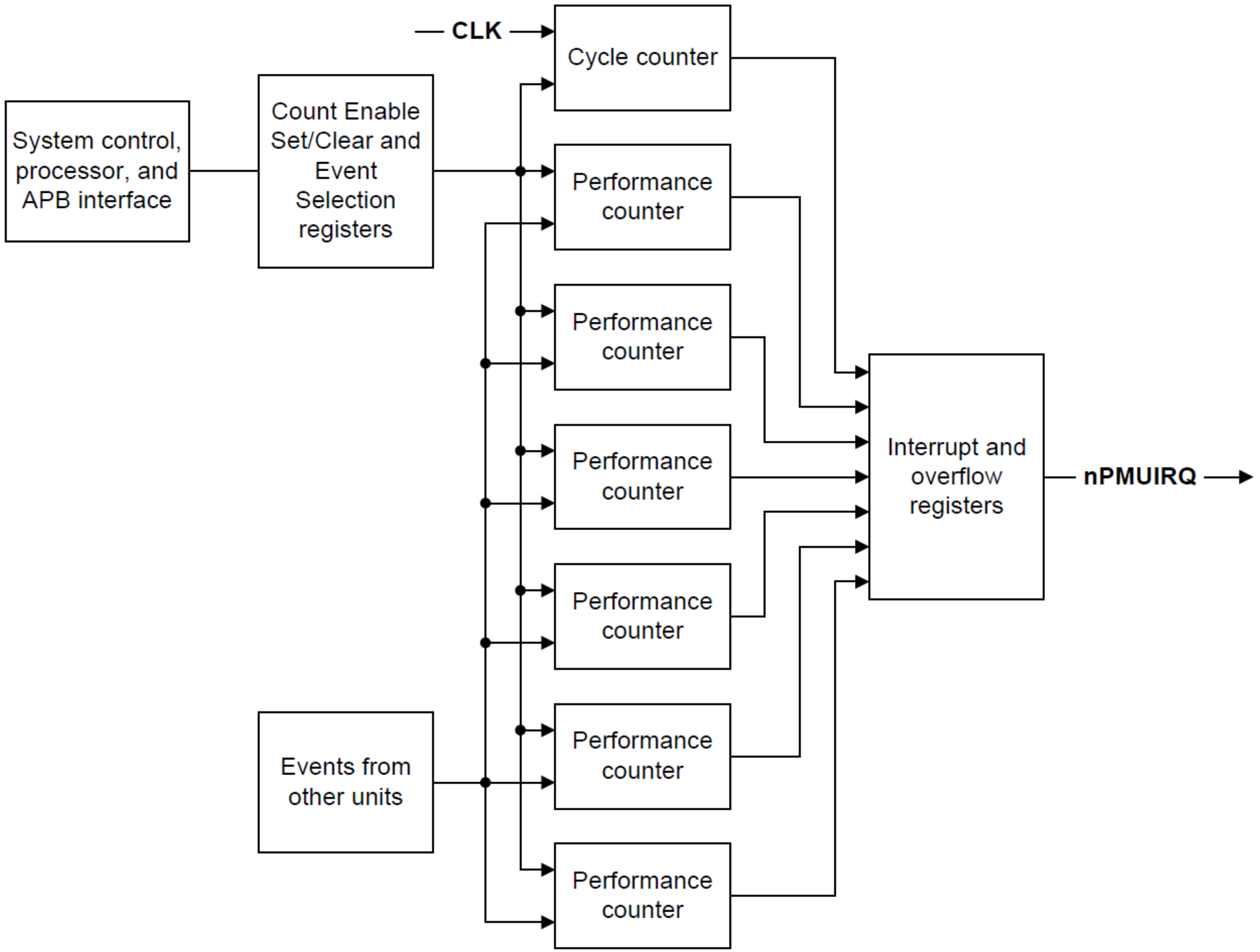
```
PerfTop:    1005 irqs/sec  kernel:99.3%  exact:  0.0% [1000Hz cycles],  (all, 1 CPU)
-------------------------------------------------------------------------------------

  98.07%  [kernel]          [k] am33xx_enter_idle
   0.95%  perf              [.] 0x39980
   0.13%  libc-2.12.2.so    [.] memchr
   0.10%  [kernel]          [k] kallsyms_expand_symbol
   0.10%  [kernel]          [k] vsnprintf
   0.09%  [kernel]          [k] format_decode
   0.08%  libc-2.12.2.so    [.] 0x6acb0
   0.08%  dropbearmulti     [.] 0x193c0
   0.08%  [kernel]          [k] number.clone.7
   0.05%  libc-2.12.2.so    [.] getdelim
   0.05%  libc-2.12.2.so    [.] strstr
   0.04%  libc-2.12.2.so    [.] __libc_calloc
   0.04%  libc-2.12.2.so    [.] strcmp
   0.04%  [kernel]          [k] string.clone.1
   0.02%  libc-2.12.2.so    [.] strchr
   0.02%  libc-2.12.2.so    [.] memcpy
   0.02%  [kernel]          [k] __copy_to_user_std
   0.02%  [kernel]          [k] strnlen
   0.01%  [kernel]          [k] update_iter
   0.01%  [unknown]         [.] 0xffff0fcc
```

- sudo perf top

Hardware Event Counters

Performance Monitor Unit (PMU)

- Details in Tech. Ref. Manuals (TRMs)
  - Cortex-A53 MPCore TRM: Chapter 12
  - Cortex-A72 MPCore TRM: Chapter 11
- Cycle counter
  - Can count processor cycles
  - Or processor cycles / 64
- Many types of events can be monitored
  - Table 12-28 or 11-24 is 3½ pages long
- Six performance event counters
  - 32-bits wide
  - Each can be configured to count a given type of event



Event Types

- Instructions
  - Speculatively executed
    - Load, store, integer data processing, ASIMD, VFP, crypto, PC change, branch immediate, branch return, branch indirect, barrier
  - Retired
  - Exceptions taken, returned
- Exceptions
  - Types of exceptions
- Branches
  - Predicted, mispredicted
- L1 (I/D), L2 (U) Caches
  - Access, refill, write-back, read, write, read refill, write refill, write-back victim, write-back cleaning and coherence, invalidate

- Memory
  - Access, read, write, unaligned, unaligned write, unaligned read
- Bus
  - Access, cycle, read, write, shared access, not shared access
- L1 TLBs
  - Refill
- Exceptions

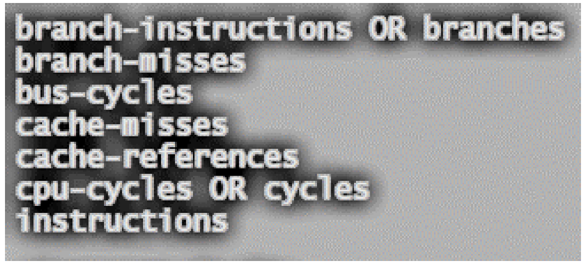
- *Sources*
  - *CA53 TRM, Section 12.9*
  - *CA72 TRM, Section 11.8*

Listing Perf Events (perf list)

```
Pre-defined events (to be used in -e or -M):
  branch-misses                                    [Hardware event]
  bus-cycles                                       [Hardware event]
  cache-misses                                     [Hardware event]
  cache-references                                 [Hardware event]
  cpu-cycles OR cycles                             [Hardware event]
  instructions                                     [Hardware event]
  alignment-faults                                 [Software event]
  bpf-output                                       [Software event]
  cgroup-switches                                  [Software event]
  context-switches OR cs                           [Software event]
  cpu-clock                                        [Software event]
  cpu-migrations OR migrations                     [Software event]
  dummy                                            [Software event]
  emulation-faults                                 [Software event]
  major-faults                                     [Software event]
  minor-faults                                     [Software event]
  page-faults OR faults                            [Software event]
  task-clock                                       [Software event]
  duration_time                                    [Tool event]
  user_time                                        [Tool event]
  system_time                                      [Tool event]

armv8_cortex_a72:
  L1-dcache-loads OR armv8_cortex_a72/L1-dcache-loads/
  L1-dcache-load-misses OR armv8_cortex_a72/L1-dcache-load-misses/
  L1-dcache-stores OR armv8_cortex_a72/L1-dcache-stores/
  L1-dcache-store-misses OR armv8_cortex_a72/L1-dcache-store-misses/
  L1-icache-loads OR armv8_cortex_a72/L1-icache-loads/
  L1-icache-load-misses OR armv8_cortex_a72/L1-icache-load-misses/
  dTLB-load-misses OR armv8_cortex_a72/dTLB-load-misses/
  dTLB-store-misses OR armv8_cortex_a72/dTLB-store-misses/
  iTLB-load-misses OR armv8_cortex_a72/iTLB-load-misses/
  branch-loads OR armv8_cortex_a72/branch-loads/
  branch-load-misses OR armv8_cortex_a72/branch-load-misses/
  node-loads OR armv8_cortex_a72/node-loads/
  node-stores OR armv8_cortex_a72/node-stores/

branch:
  br_immed_spec
       [Branch speculatively executed, immediate branch]
  br_indirect_spec
       [Branch speculatively executed, indirect branch]
  br_mis_pred
       [Mispredicted or not predicted branch speculatively executed.]
  br_pred
       [Predictable branch speculatively executed]
  br_return_spec
       [Branch speculatively executed, procedure return]

bus:
  bus_access
       [Attributable Bus access]
  bus_access_normal
       [Bus access, Normal]
  bus_access_not_shared
       [Bus access, not Normal, Cacheable, Shareable]
  bus_access_periph
       [Bus access, peripheral]
  bus_access_rd
       [Bus access read]
  bus_access_shared
       [Bus access, Normal, Cacheable, Shareable]
  bus_access_wr
       [Bus access write]
  bus_cycles
       [Bus cycle]
  cpu_cycles
       [Cycle]

cache:
  l1d_cache
       [Level 1 data cache access]
  l1d_cache_inval
       [L1D cache invalidate]
  l1d_cache_rd
       [L1D cache access, read]
  l1d_cache_refill
       [Level 1 data cache refill]
  l1d_cache_refill_rd
       [L1D cache refill, read]
  l1d_cache_refill_wr
       [L1D cache refill, write]
  l1d_cache_wb
       [Attributable Level 1 data cache write-back]
  l1d_cache_wb_clean
       [L1D cache Write-Back, cleaning and coherency]
  l1d_cache_wb_victim
       [L1D cache Write-Back, victim]
  l1d_cache_wr
       [L1D cache access, write]
  l1d_tlb_refill
       [Attributable Level 1 data TLB refill]
  l1d_tlb_refill_rd
       [L1D tlb refill, read]
  l1d_tlb_refill_wr
       [L1D tlb refill, write]
  l1i_cache
       [Attributable Level 1 instruction cache access]
  l1i_cache_refill
       [Level 1 instruction cache refill]
  l1i_tlb_refill
       [Attributable Level 1 instruction TLB refill]
  l2d_cache
       [Level 2 data cache access]
  l2d_cache_inval
       [L2D cache invalidate]
  l2d_cache_rd
       [L2D cache access, read]
  l2d_cache_refill
       [Level 2 data refill]
  l2d_cache_refill_rd
       [L2D cache refill, read]
  l2d_cache_refill_wr
       [L2D cache refill, write]
  l2d_cache_wb
       [Attributable Level 2 data cache write-back]
  l2d_cache_wb_clean
       [L2D cache Write-Back, cleaning and coherency]
  l2d_cache_wb_victim
       [L2D cache Write-Back, victim]
  l2d_cache_wr
       [L2D cache access, write]

exception:
  exc_dabort
       [Exception taken, Data Abort and SError]
  exc_fiq
       [Exception taken, FIQ]
  exc_hvc
       [Exception taken, Hypervisor Call]
  exc_irq
       [Exception taken, IRQ]
  exc_pabort
       [Exception taken, Instruction Abort]
  exc_smc
       [Exception taken, Secure Monitor Call]
  exc_svc
       [Exception taken, Supervisor Call]
  exc_taken
       [Exception taken]
  exc_trap_dabort
       [Exception taken, Data Abort or SError not taken locally]
  exc_trap_fiq
       [Exception taken, FIQ not taken locally]
  exc_trap_irq
       [Exception taken, IRQ not taken locally]
  exc_trap_other
       [Exception taken, Other traps not taken locally]
  exc_trap_pabort
       [Exception taken, Instruction Abort not taken locally]
  exc_undef
       [Exception taken, Other synchronous]
  memory_error
       [Local memory error]

instruction:
  ase_spec
       [Operation speculatively executed, Advanced SIMD instruction]
  cid_write_retired
       [Instruction architecturally executed, condition code check
        pass, write to CONTEXTIDR]
  crypto_spec
       [Operation speculatively executed, Cryptographic instruction]
  dmb_spec
       [Barrier speculatively executed, DMB]
  dp_spec
       [Operation speculatively executed, integer data processing]
  dsb_spec
       [Barrier speculatively executed, DSB]
  exc_return
       [Instruction architecturally executed, condition check pass,
        exception return]
  inst_retired
       [Instruction architecturally executed]
  inst_spec
       [Operation speculatively executed]
  isb_spec
       [Barrier speculatively executed, ISB]
  ld_spec
       [Operation speculatively executed, load]
  ldrex_spec
       [Exclusive operation speculatively executed, LDREX or LDX]
  ldst_spec
       [Operation speculatively executed, load or store]
  pc_write_spec
       [Operation speculatively executed, software change of the PC]
  rc_ld_spec
       [Release consistency operation speculatively executed,
        Load-Acquire]
  rc_st_spec
       [Release consistency operation speculatively executed,
        Store-Release]
  st_spec
       [Operation speculatively executed, store]
  strex_fail_spec
       [Exclusive operation speculatively executed, STREX or STX fail]
  strex_pass_spec
       [Exclusive operation speculatively executed, STREX or STX pass]
  sw_incr
       [Instruction architecturally executed, Condition code
        check pass, software increment]
  ttbr_write_retired
       [Instruction architecturally executed, Condition code check
        pass, write to TTBR]
  vfp_spec
       [Operation speculatively executed, floating-point instruction]

memory:
  mem_access
       [Data memory access]
  mem_access_rd
       [Data memory access, read]
  mem_access_wr
       [Data memory access, write]
  unaligned_ld_spec
       [Unaligned access, read]
  unaligned_ldst_spec
       [Unaligned access]
  unaligned_st_spec
       [Unaligned access, write]
  rNNN                                     [Raw hardware event descriptor]
  cpu/t1=v1[,t2=v2,t3 ...]/modifier        [Raw hardware event descriptor]
       [(see 'man perf-list' on how to encode it)]
  mem:<addr>[/len][:access]                [Hardware breakpoint]
```

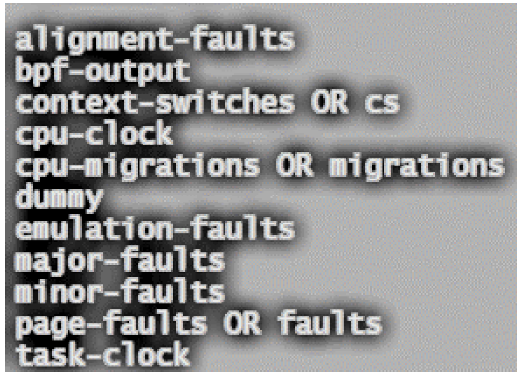
Listing Perf Events

- perf list
- Hardware events



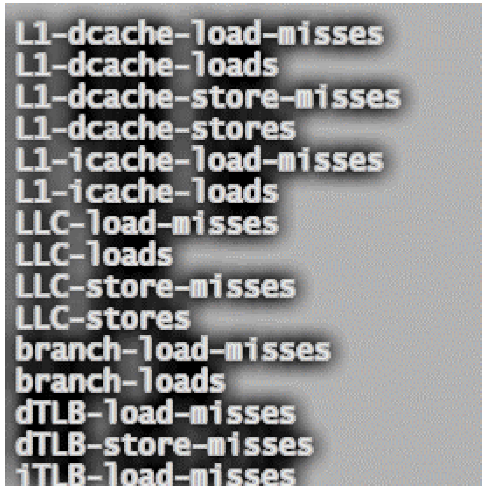
```
branch-instructions OR branches
branch-misses
bus-cycles
cache-misses
cache-references
cpu-cycles OR cycles
instructions
```

- Software events



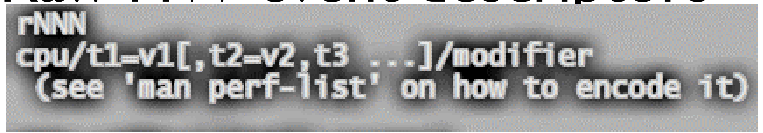
```
alignment-faults
bpf-output
context-switches OR cs
cpu-clock
cpu-migrations OR migrations
dummy
emulation-faults
major-faults
minor-faults
page-faults OR faults
task-clock
```

- Hardware cache events



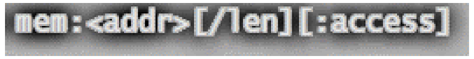
```
L1-dcache-load-misses
L1-dcache-loads
L1-dcache-store-misses
L1-dcache-stores
L1-icache-load-misses
L1-icache-loads
LLC-load-misses
LLC-loads
LLC-store-misses
LLC-stores
branch-load-misses
branch-loads
dTLB-load-misses
dTLB-store-misses
iTLB-load-misses
```

- Raw HW event descriptors



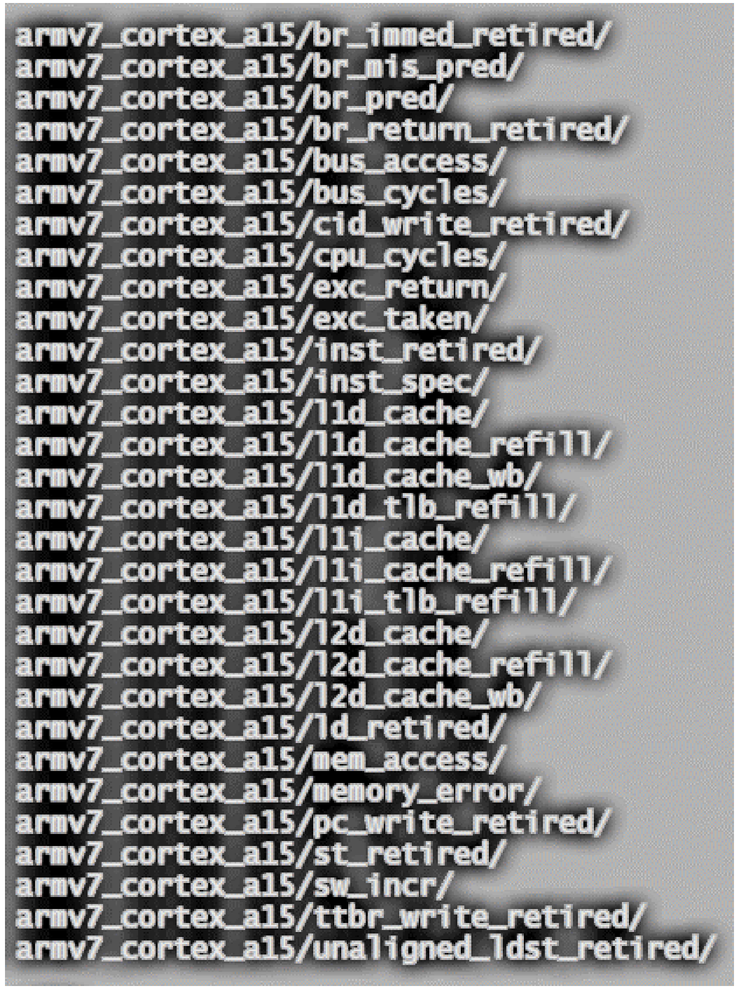
```
rNNN
cpu/t1=v1[,t2=v2,t3 ...]/modifier
  (see 'man perf-list' on how to encode it)
```

- Hardware breakpoint



```
mem:<addr>[/len][:access]
```

- Kernel PMU events



```
armv7_cortex_a15/br_immed_retired/
armv7_cortex_a15/br_mis_pred/
armv7_cortex_a15/br_pred/
armv7_cortex_a15/br_return_retired/
armv7_cortex_a15/bus_access/
armv7_cortex_a15/bus_cycles/
armv7_cortex_a15/cid_write_retired/
armv7_cortex_a15/cpu_cycles/
armv7_cortex_a15/exc_return/
armv7_cortex_a15/exc_taken/
armv7_cortex_a15/inst_retired/
armv7_cortex_a15/inst_spec/
armv7_cortex_a15/l1d_cache/
armv7_cortex_a15/l1d_cache_refill/
armv7_cortex_a15/l1d_cache_wb/
armv7_cortex_a15/l1d_tlb_refill/
armv7_cortex_a15/l1i_cache/
armv7_cortex_a15/l1i_cache_refill/
armv7_cortex_a15/l1i_tlb_refill/
armv7_cortex_a15/l2d_cache/
armv7_cortex_a15/l2d_cache_refill/
armv7_cortex_a15/l2d_cache_wb/
armv7_cortex_a15/ld_retired/
armv7_cortex_a15/mem_access/
armv7_cortex_a15/memory_error/
armv7_cortex_a15/pc_write_retired/
armv7_cortex_a15/st_retired/
armv7_cortex_a15/sw_incr/
armv7_cortex_a15/ttbr_write_retired/
armv7_cortex_a15/unaligned_ldst_retired/
```

Useful Perf Commands

- Get information on perf's capabilities
  - perf stat --help
  - perf list sw
- Measure a program
  - sudo perf record ./istool1 *(samples program*)
  - sudo perf stat –e instructions,cycles,branches,branch-misses ./istool1 *(uses PMU event counters)*
- Measure system
  - sudo perf top
- Evaluate data
  - sudo perf report
  - sudo perf annotate

Summary

- Review of "Optimization" Process: Analyze, then "Optimize"
- Analysis
  - Measuring total code execution time
  - Measuring time distribution within code (*profiling*)
  - Measuring key performance event counts
- Analysis is key to optimization
  - Examine compiler output, do easy optimizations
  - Then do harder optimizations
  - Apply SIMD if worthwhile
  - Apply multithreading if worthwhile