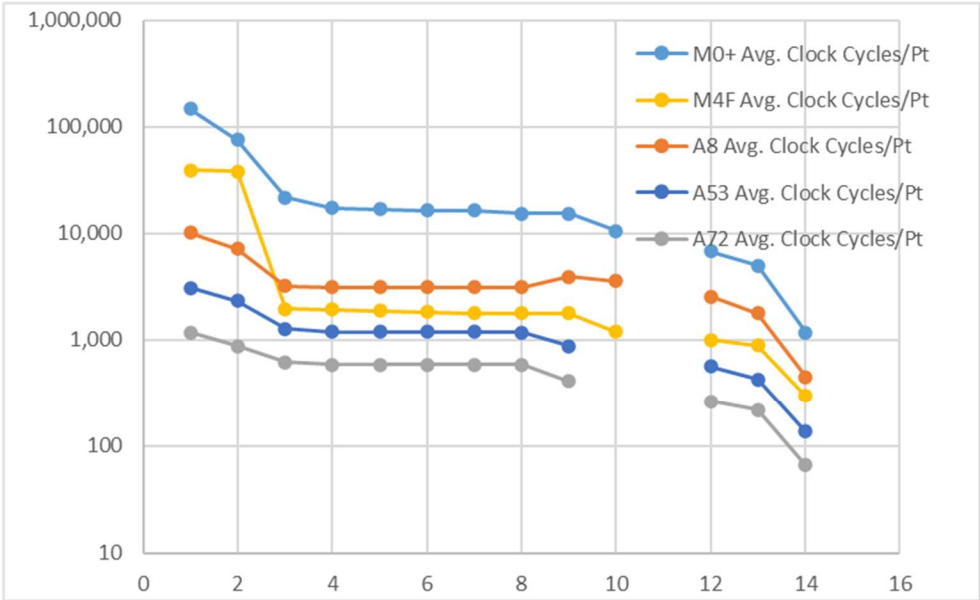
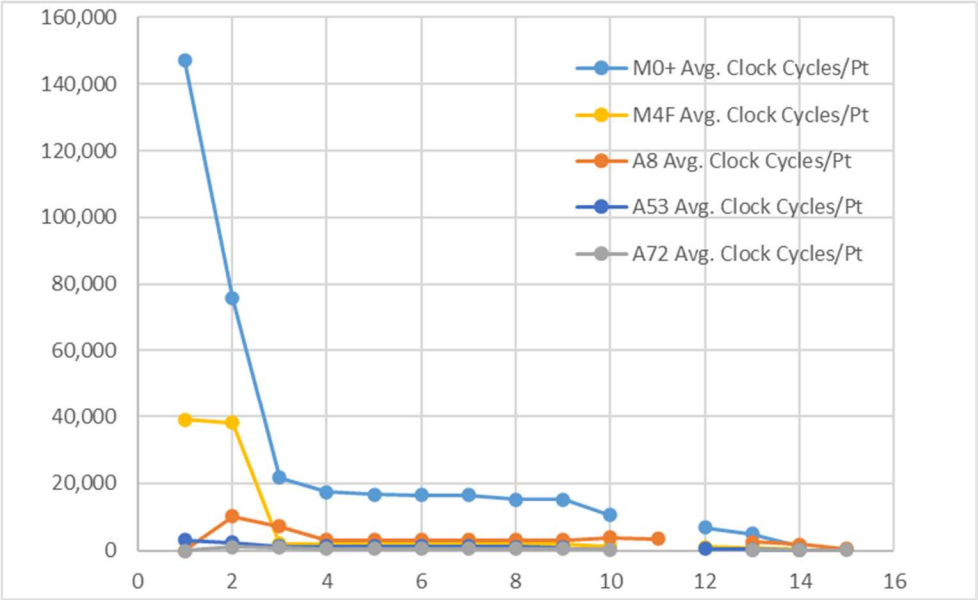
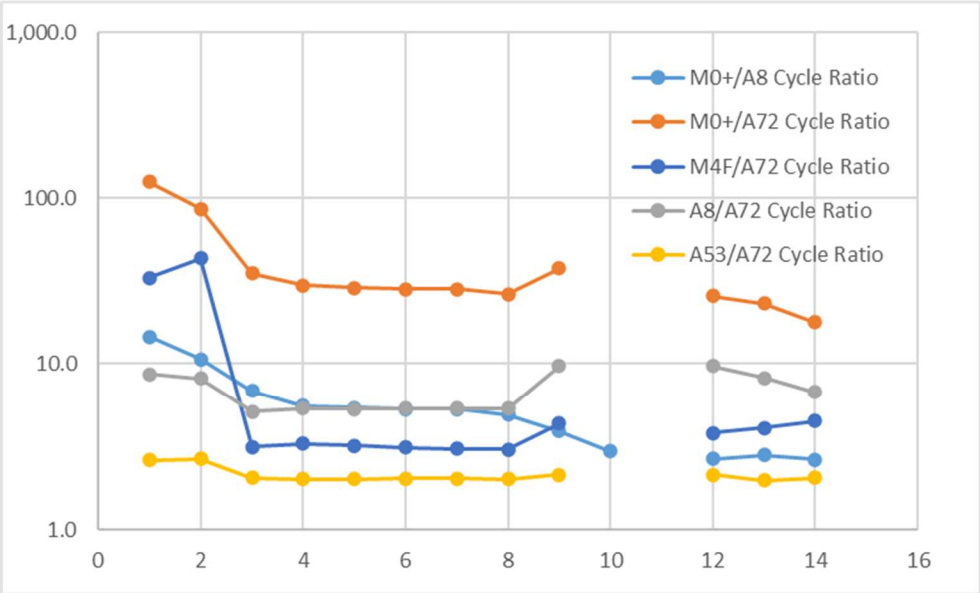
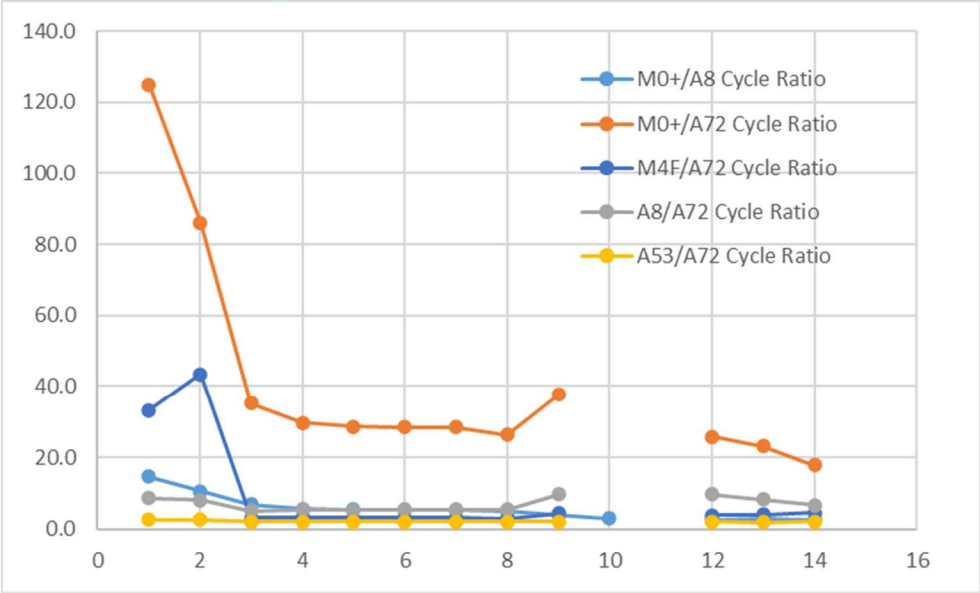


Optimizing the Spherical Geometry Program

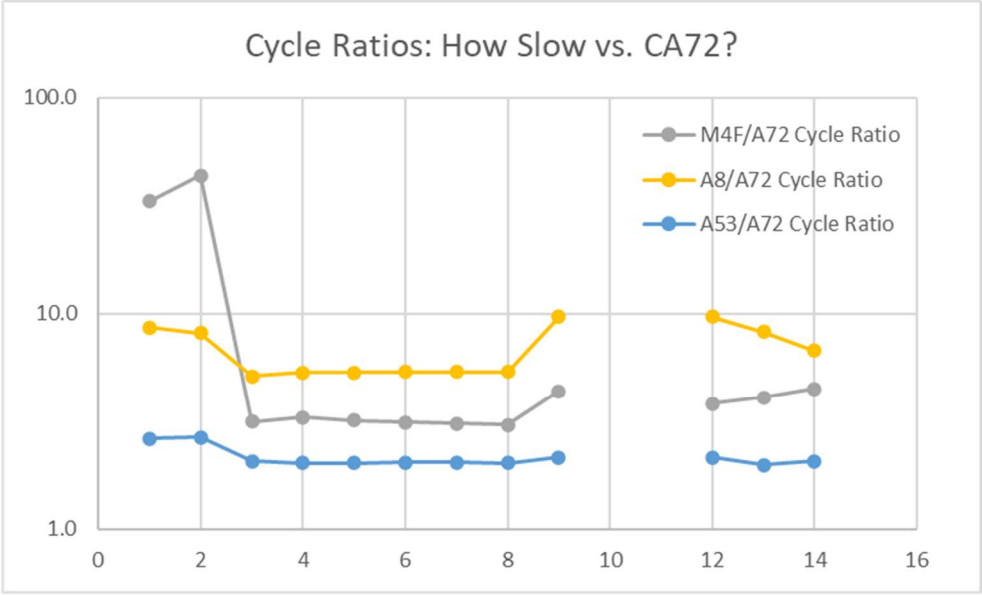
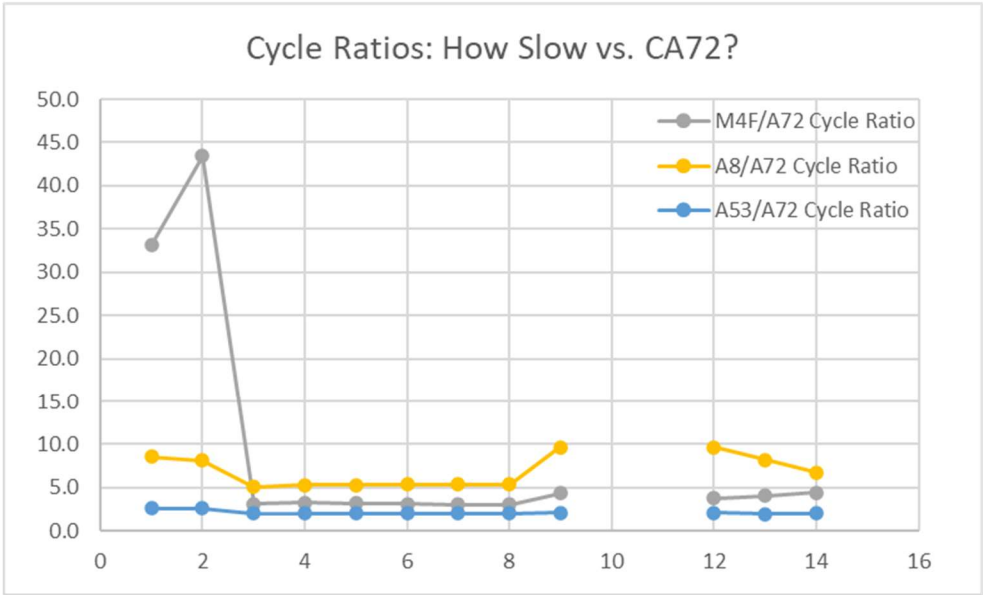
Clock Cycle Count per Point



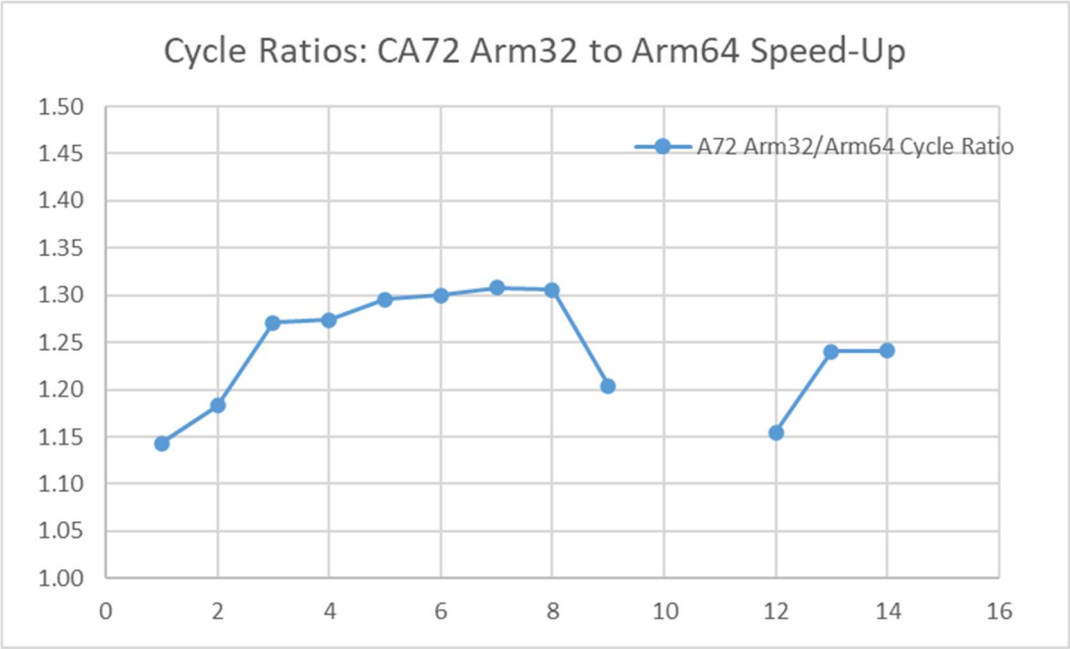
Clock Cycle Count Ratios



Clock Cycle Count Ratios vs. Cortex-A72 (Arm32)



Clock Cycle Count Ratio: Cortex-A72 Arm32 to Arm64



Analysis Process

- Build, run, update spreadsheet
 - Review spreadsheet
- Gather performance data
 - `sudo perf record ./sg`
 - `sudo perf annotate`
 - P saves to text file `<symbol_name>.annotation`
- Use Ghidra to analyze sg executable
 - Examine top functions

Arm64 Performance: Why so good? And why not even better?

alex@raspberrypi: ~/AES-24/Speed/Scalar/SG2

Samples: 2K of event 'cycles:P', Event count (approx.): 1002378189

Overhead	Command	Shared Object	Symbol
37.97%	sg	sg	[.] Find_Nearest_Waypoint
21.01%	sg	sg	[.] __cosf
9.91%	sg	sg	[.] strcmp
9.25%	sg	[kernel.kallsyms]	[k] el0_svc
8.61%	sg	[kernel.kallsyms]	[k] _raw_spin_unlock_irqrestore
5.50%	sg	[vdso]	[.] __kernel_clock_gettime
0.78%	sg	[kernel.kallsyms]	[k] _raw_spin_lock_irqsave
0.52%	sg	[kernel.kallsyms]	[k] __arm64_sys_clock_gettime
0.48%	sg	sg	[.] __atan2f
0.45%	sg	[kernel.kallsyms]	[k] el0_svc_common.constprop.0
0.41%	sg	sg	[.] __sincosf
0.41%	sg	sg	[.] __clock_gettime
0.41%	sg	[kernel.kallsyms]	[k] posix_cpu_clock_get

- Sanity check on results
 - Find_Nearest_Waypoint – OK, expected
 - __cosf – OK, expected
 - strcmp – OK, surprised it takes so much time
 - el0_svc – What is this? 9.25%
 - _raw_spin_unlock_irqrestore – What is this? 8.61%
 - __kernel_clock_gettime – Big! 5.50%
- P: Save this to perf.hist.N

alex@raspberrypi: ~/AES-24/Speed/Scalar/SG2

Samples: 13K of event 'cycles:P', Event count (approx.): 1024123918

Overhead	Command	Shared Object	Symbol
39.42%	sg	sg	[.] Find_Nearest_Waypoint
22.34%	sg	sg	[.] __cosf
10.49%	sg	sg	[.] strcmp
8.33%	sg	[kernel.kallsyms]	[k] _raw_spin_unlock_irqrestore
7.77%	sg	[kernel.kallsyms]	[k] el0_svc
4.59%	sg	[vdso]	[.] __kernel_clock_gettime
0.56%	sg	sg	[.] __atan2f
0.50%	sg	[kernel.kallsyms]	[k] __arm64_sys_clock_gettime
0.46%	sg	[kernel.kallsyms]	[k] _raw_spin_lock_irqsave
0.44%	sg	sg	[.] __sincosf
0.39%	sg	sg	[.] __clock_gettime
0.39%	sg	sg	[.] __ieee754_atan2f

- Rerun with higher sampling frequency to get more than 2K samples:
 - sudo perf record -F 20000 ./sg
 - get 13K samples
- Repeat the sanity check
 - Similar results, so probably ok

Nice Perf Annotate Features

```
alex@raspberrypi: ~/AES-24/Speed/Scalar/SG2
Samples: 13K of event 'cycles:P', 20000 Hz, Event count (approx.): 1024123918

0.04 |      fmul    s8, s8, s11
0.51 |      ldp     s11, s10, [sp, #152]
      fdiv     s8, s8, s12
      while (strcmp(waypoints[i].Name, "END")) {
      | b      d4
      Calc_Closeness():
3.05 | a8: UP/DOWN/PGUP
      PGDN/SPACE    Navigate
      </>           Move to prev/next symbol
      q/ESC/CTRL+C  Exit
3.12 | ENTER        Go to target
      H          Go to hottest instruction
      TAB/shift+TAB 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
      O          Bump offset level (jump targets -> +call -> all -> cycle thru)
      s          Toggle source code view
      t          Circulate percent, total period, samples view
      c          Show min/max cycle
      /          Search string
      k          Toggle line numbers
      l          Show full source file location
      P          Print to [symbol_name].annotation file.
      r          Run available scripts
      p          Toggle percent type [local/global]
      b          Toggle percent base [period/hits]
      ?          Search string backwards
84.36 | cc: f         Toggle showing offsets to full address
      d4: Press any key...
```

- h: Get this help screen
- p: Toggle percentage between local (this function) and global (entire program)
- t: Cycle percent -> period -> # samples -> ...
- o, s, k: Toggle code listing
- P: Save to [symbol_name].annotation file

```
C:\Users\Alex\Documents\Teaching\ECE785\AES-2024\Modules\Speed Optimization\Optimizing SG\Find_Nearest_Waypo...
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Find_Nearest_Waypoint.annotation
1 Find_Nearest_Waypoint() /home/alex/AES-24/Speed/Scalar/SG2/sg
2 Event: cycles:P
3
4 0.10      stp     x29, x30, [sp, #-160]!
5          mov     x29, sp
6 0.01      stp     x19, x20, [sp, #16]
7 0.01      stp     x21, x22, [sp, #32]
8 0.04      stp     x23, x24, [sp, #48]
9 0.01      stp     x25, x26, [sp, #64]
10 0.01      stp     x27, x28, [sp, #80]
11 0.02      stp     d8, d9, [sp, #96]
12 0.02      stp     d10, d11, [sp, #112]
13 0.02      str     d12, [sp, #128]
14          xpacrlri
15 0.02      fmov     s10, s0
16          fmov     s8, s1
17          mov     x26, x0
18          movi     v9.2s, #0x0
19          mov     x0, x30
20          ...
```

Try to Fill in Holes Between Samples

- Do these instructions *really* all complete in the same clock cycle?
- Maybe they don't but are missed by sampling
 - Program was sampled at 20 kHz, got 13,000 samples.
- Get more samples
 - Raise sampling frequency even more to 100 kHz
 - Increase program run time by raising NUM_TESTS to 1,000,000
- Now get 888,290 samples
- Eliminate possible truncation by display sample count, not percentage

Find_Nearest_Waypoint.annotation				_cosf.annotation			
43	0.20		ldp	s11, s10, [sp, #152]			
44			fdiv	s8, s8, s12			
45			↓ b	d4			
46	1.99	a8:	ldur	s0, [x19, #-4]			
47			fsub	s0, s0, s8			
48			→ bl	__cosf			
49	1.23		ldp	s2, s1, [x19, #-12]			
50			fmul	s1, s11, s1			
51			fmul	s0, s1, s0			
52			fmadd	s0, s10, s2, s0			
53	0.54		fcmp	s9, s0			
54			↓ b.mi	194			
55	33.26	cc:	add	w20, w20, #0x1			
56			add	x19, x19, #0x28			
57		d4:	mov	x1, x21			
58			mov	x0, x19			
59	0.08		→ bl	strcmp			
60	0.06		↑ cbnz	w0, a8			
61			fmov	s0, s9			
62			sbfiz	x19, x22, #2, #32			
63			add	x19, x19, w22, sxtw			
64			→ bl	__acosf			
65	0.03		mov	w0, #0x1800			

Comparison: 13k vs. 888k samples

Find_Nearest_Waypoint.annotation			
43	0.20	ldp	s11, s10, [sp, #152]
44		fdiv	s8, s8, s12
45		↓ b	d4
46	1.99	a8: ldur	s0, [x19, #-4]
47		fsub	s0, s0, s8
48		→ bl	__cosf
49	1.23	ldp	s2, s1, [x19, #-12]
50		fmul	s1, s11, s1
51		fmul	s0, s1, s0
52		fmadd	s0, s10, s2, s0
53	0.54	fcmpe	s9, s0
54		↓ b.mi	194
55	33.26	cc: add	w20, w20, #0x1
56		add	x19, x19, #0x28
57		d4: mov	x1, x21
58		mov	x0, x19
59	0.08	→ bl	strcmp
60	0.06	↑ cbnz	w0, a8
61		fmov	s0, s9
62		sbfiz	x19, x22, #2, #32
63		add	x19, x19, w22, sxtw
64		→ bl	__acosf
65	0.03	mov	w0, #0x1800

Find_Nearest_Waypoint.annotation			
43	765056	ldp	s11, s10, [sp, #152]
44		fdiv	s8, s8, s12
45		↓ b	d4
46	248473989	a8: ldur	s0, [x19, #-4]
47		fsub	s0, s0, s8
48		→ bl	__cosf
49	141652362	ldp	s2, s1, [x19, #-12]
50		fmul	s1, s11, s1
51		fmul	s0, s1, s0
52		fmadd	s0, s10, s2, s0
53	68382855	fcmpe	s9, s0
54		↓ b.mi	194
55	4021122350	cc: add	w20, w20, #0x1
56		add	x19, x19, #0x28
57	13359	d4: mov	x1, x21
58		mov	x0, x19
59	4093965	→ bl	strcmp
60	3981251	↑ cbnz	w0, a8
61	1299935	fmov	s0, s9
62		sbfiz	x19, x22, #2, #32
63		add	x19, x19, w22, sxtw
64		→ bl	__acosf
65	2891705	mov	w0, #0x1800

Latest Profile

alex@raspberrypi: ~/AES-24/Speed/Scalar/SG2

Samples: 888K of event 'cycles:P', Event count (approx.): 11898345696

Overhead	Command	Shared Object	Symbol
38.73%	sg	sg	[.] Find_Nearest_Waypoint
22.84%	sg	sg	[.] __cosf
10.56%	sg	[kernel.kallsyms]	[k] _raw_spin_unlock_irqrestore
10.43%	sg	sg	[.] strcmp
7.27%	sg	[kernel.kallsyms]	[k] el0_svc
3.43%	sg	[vdso]	[.] __kernel_clock_gettime
0.76%	sg	[kernel.kallsyms]	[k] _raw_spin_lock_irqsave
0.64%	sg	sg	[.] __atan2f
0.56%	sg	[kernel.kallsyms]	[k] __arm64_sys_clock_gettime
0.40%	sg	sg	[.] __ieee754_atan2f
0.37%	sg	[kernel.kallsyms]	[k] posix_cpu_clock_get
0.36%	sg	[kernel.kallsyms]	[k] invoke_syscall
0.34%	sg	[kernel.kallsyms]	[k] put_timespec64
0.32%	sg	[kernel.kallsyms]	[k] el0_svc_common.constprop.0
0.29%	sg	[kernel.kallsyms]	[k] do_el0_svc
0.29%	sg	sg	[.] __sincosf
0.27%	sg	sg	[.] __atanf
0.27%	sg	[kernel.kallsyms]	[k] __arch_copy_to_user

Find_Nearest_Waypoint: 38.7% of Program's Time

```
void Find_Nearest_Waypoint(float cur_pos_lat, float cur_pos_lon, float * distance, float * bearing,
```

```
    char * * name) {
```

```
    // cur_pos_lat and cur_pos_lon are in degrees
```

```
    // distance is in kilometers
```

```
    // bearing is in degrees
```

```
    int i=0, closest_i=0;
```

```
    PT_T ref;
```

```
    float d, b, c, max_c=0, closest_d=1E10;
```

```
    while (strcmp(waypoints[i].Name, "END")) {
```

```
        c = Calc_Closeness(&ref, &(waypoints[i]));
```

```
        if (c>max_c) {
```

```
            max_c = c;
```

```
            closest_i = i;
```

```
        }
```

```
        i++;
```

```
    }
```

```
    // Finish calculations for the closest point
```

```
    b = Calc_Bearing(&ref, &(waypoints[closest_i]));
```

```
    // return information to calling function about closest
```

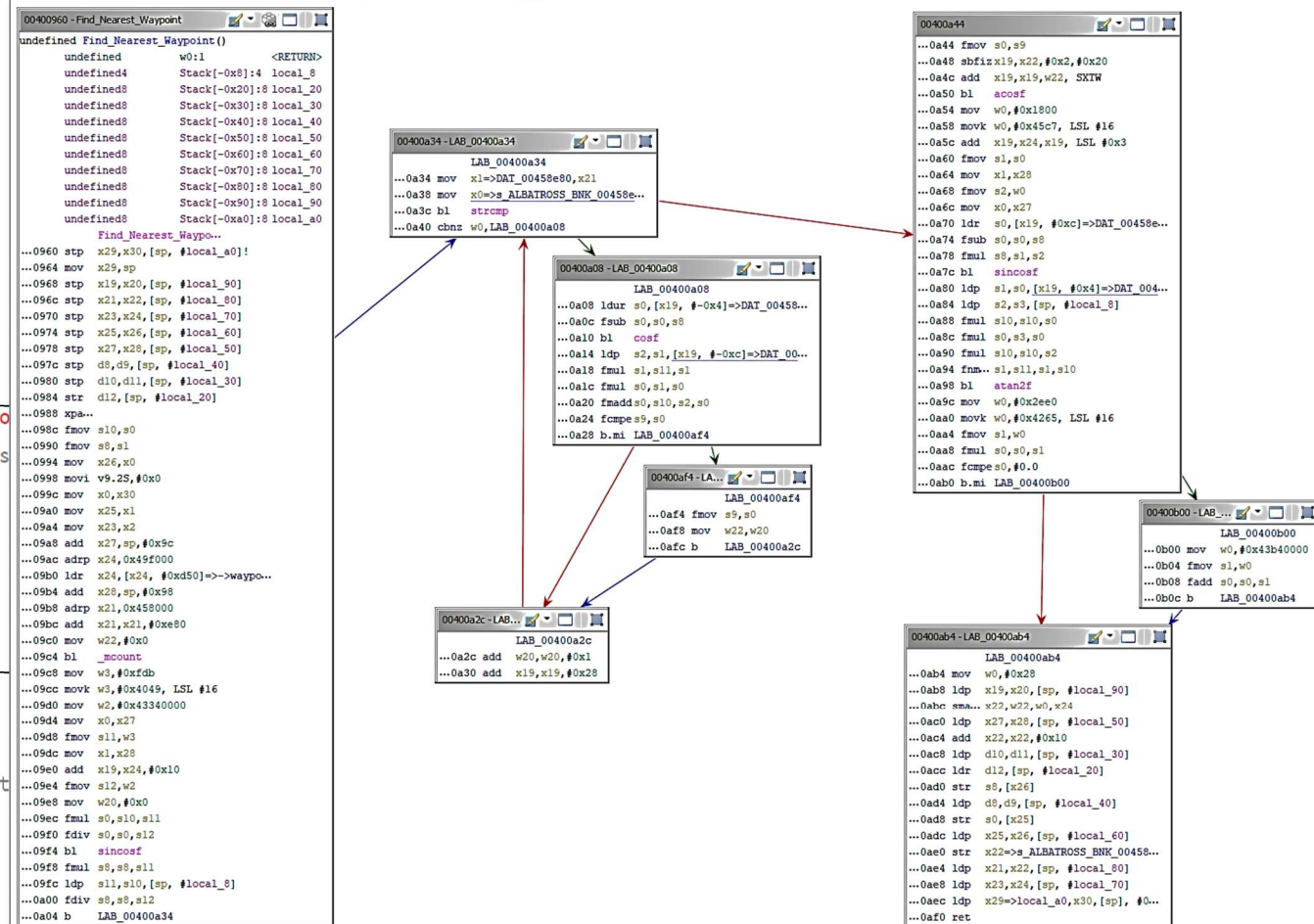
```
    *distance = d;
```

```
    *bearing = b;
```

```
    *name = (char * ) (waypoints[closest_i].Name);
```

```
}
```

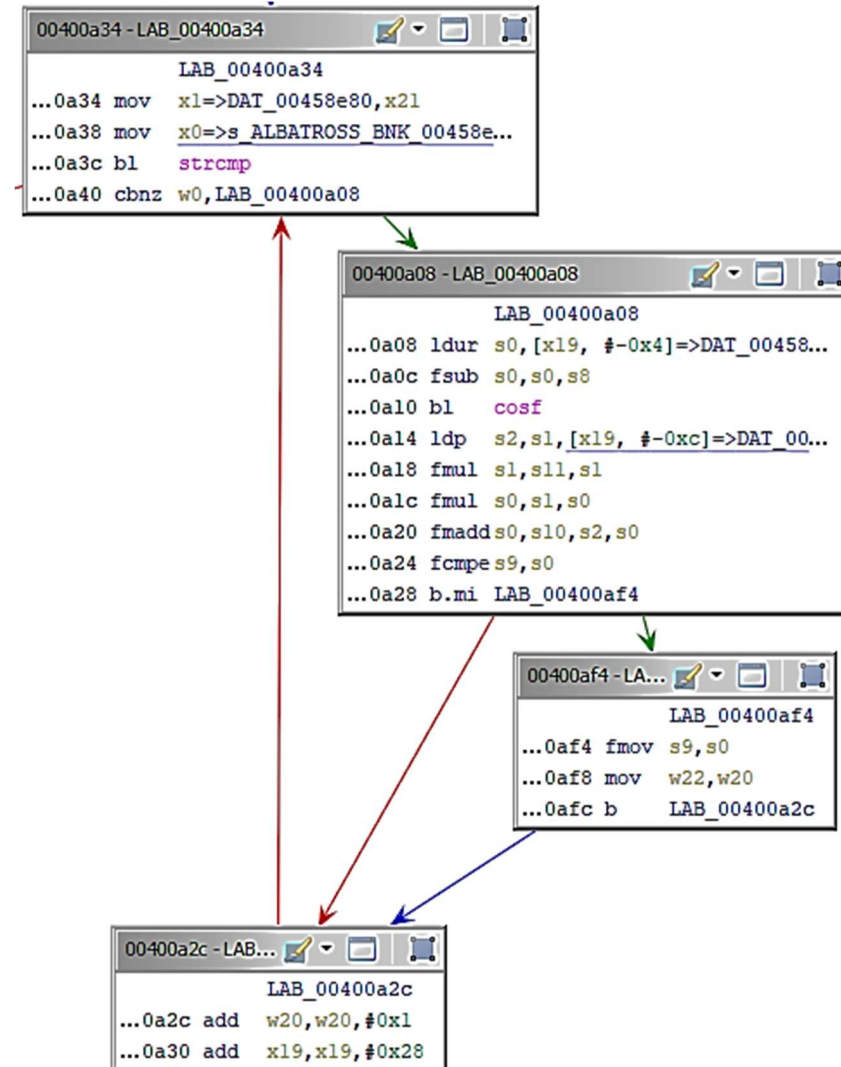
```
float Calc_Closeness( PT_T * p1, PT_T * p2 ) {
    // calculates closeness (decreases
    // as distance increases)
    return p1->SinLat * p2->SinLat +
        p1->CosLat * p2->CosLat *
        cosf(p2->Lon - p1->Lon);
}
```



Find_Nearest_Waypoint Loop

```
while (strcmp(waypoints[i].Name, "END")) {
    c = Calc_Closeness(&ref, &(waypoints[i]));
    return p1->SinLat * p2->SinLat +
        p1->CosLat * p2->CosLat *
        cosf(p2->Lon - p1->Lon);
    if (c > max_c) {
        max_c = c;
        closest_i = i;
    }
    i++;
}
```

- What is going on here? Mark up hot instructions from annotation information



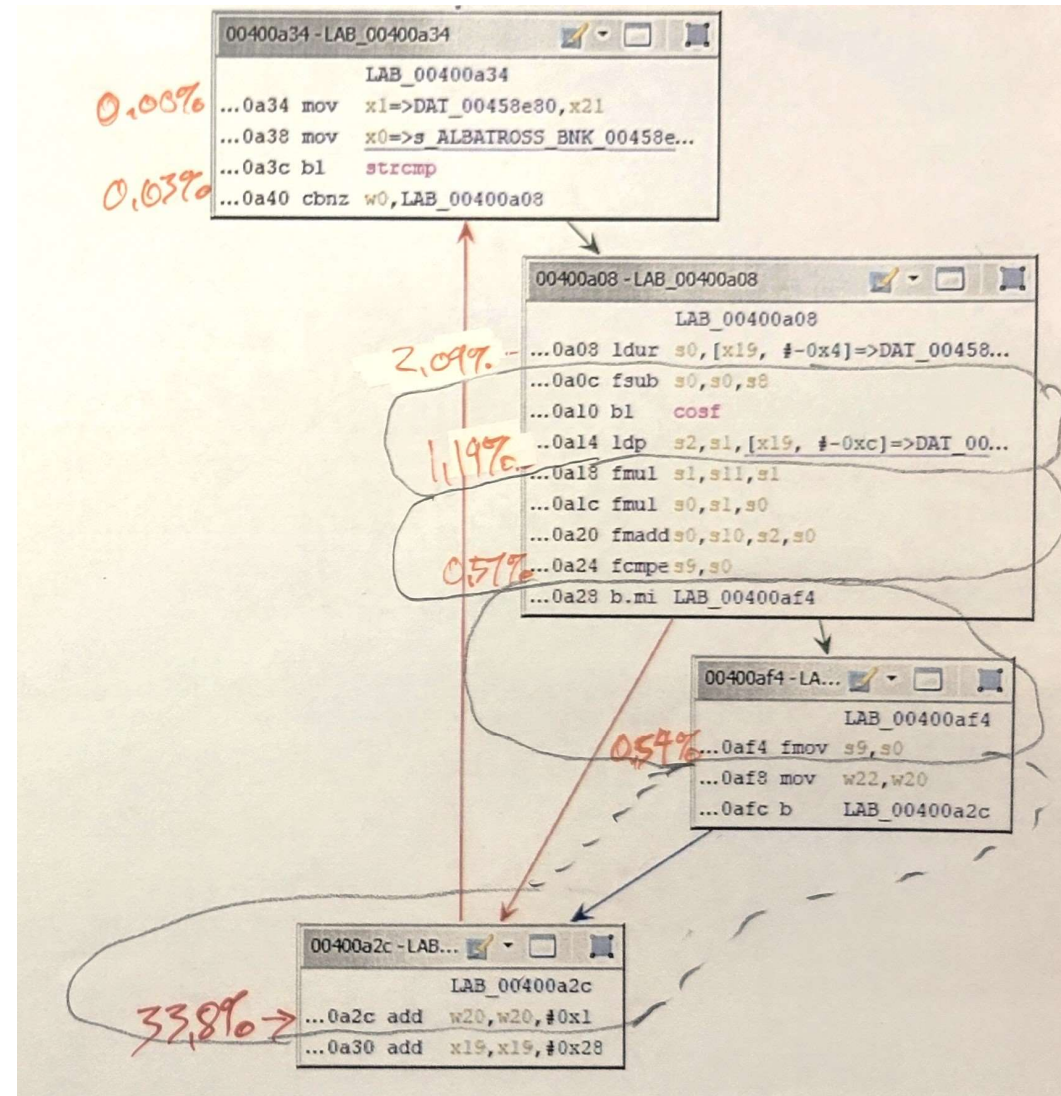
Find_Nearest_Waypoint Loop

```

while (strcmp(waypoints[i].Name, "END")) {
    c = Calc_Closeness(&ref, &(waypoints[i]));
    return p1->SinLat * p2->SinLat +
        p1->CosLat * p2->CosLat *
        cosf(p2->Lon - p1->Lon);
    if (c > max_c) {
        max_c = c;
        closest_i = i;
    }
    i++;
}

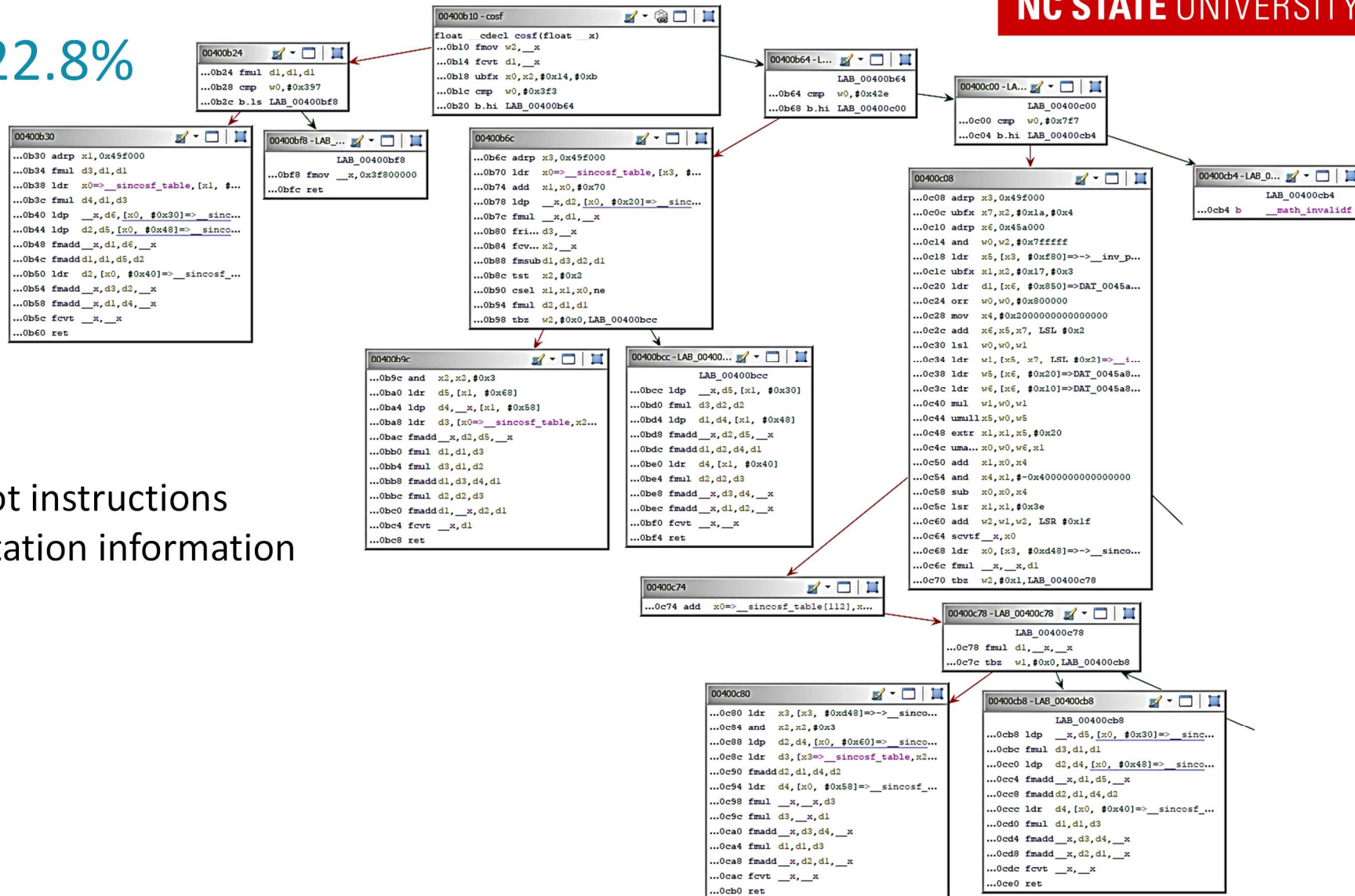
```

- What is going on here? Mark up hot instructions from annotation information

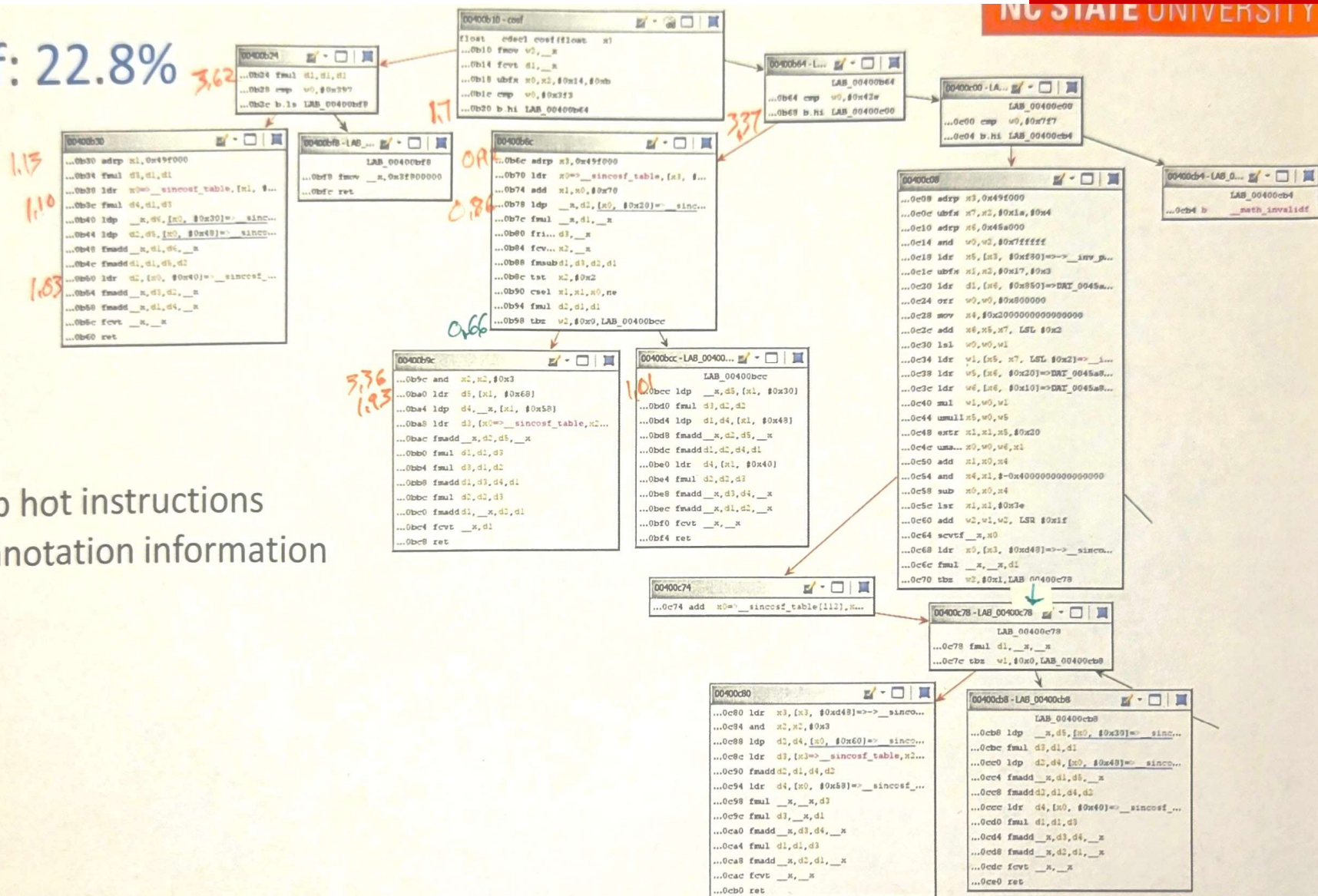


cosf: 22.8%

- Mark up hot instructions from annotation information

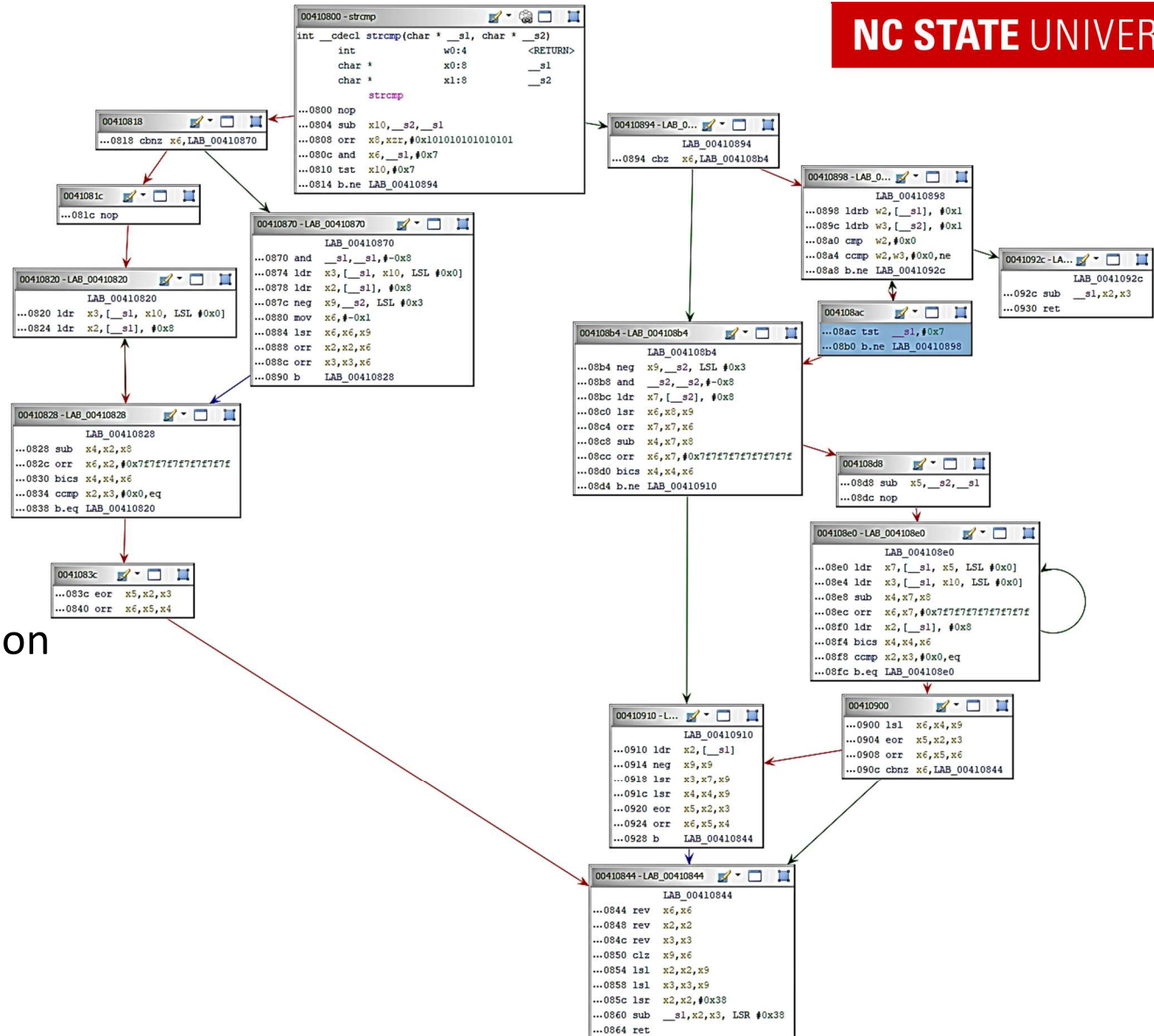


cosf: 22.8%



- Mark up hot instructions from annotation information

strcmp: 10.4%



- Mark up hot instructions from annotation information

strcmp: 10.5%

- Mark up hot instructions from annotation information

