1. Task 1
    a. I just used the default parameters for the baseline time

```
sethan@sethan-700Z3A-700Z4A-700Z5A-700Z5B:~/4200/assignment4$ make time
/usr/bin/time mandel/mandel output/mandel.bmp -r 1024 -x 0.5 -y 0.5 -s 1.0 -c 1
-i 512
Center:      [0.500000,0.500000]
Zoom:        100%
Iterations:  512
Window:      Re[-1.500000,0.500000], Im[-1.000000,1.000000]
Output:      output/mandel.bmp
12.06user 0.01system 0:12.11elapsed 99%CPU (0avgtext+0avgdata 12184maxresident)k
88inputs+6152outputs (1major+2862minor)pagefaults 0swaps
sethan@sethan-700Z3A-700Z4A-700Z5A-700Z5B:~/4200/assignment4$
```

    And it took 12,11 seconds.

    b. Resolution and iterations changed:

```
sethan@sethan-700Z3A-700Z4A-700Z5A-700Z5B:~/4200/assignment4$ make RES=2560  I=1
024 time
/usr/bin/time mandel/mandel output/mandel.bmp -r 2560 -x 0.5 -y 0.5 -s 1.0 -c 1
-i 1024
Center:      [0.500000,0.500000]
Zoom:        100%
Iterations:  1024
Window:      Re[-1.500000,0.500000], Im[-1.000000,1.000000]
Output:      output/mandel.bmp
166.49user 0.03system 2:46.55elapsed 99%CPU (0avgtext+0avgdata 71408maxresident)
k
0inputs+38408outputs (0major+17665minor)pagefaults 0swaps
sethan@sethan-700Z3A-700Z4A-700Z5A-700Z5B:~/4200/assignment4$
```

    By just changing the number of iterations to 1024 you would have expected something like 24 seconds, therefore the resolution parameter have a significant computation cost.

    Changed x and y:

```
sethan@sethan-700Z3A-700Z4A-700Z5A-700Z5B:~/4200/assignment4$ make X=50 Y=50 tim
e
/usr/bin/time mandel/mandel output/mandel.bmp -r 1024 -x 50 -y 50 -s 1.0 -c 1 -i
 512
Center:      [1.000000,1.000000]
Zoom:        100%
Iterations:  512
Window:      Re[0.500000,2.500000], Im[1.000000,3.000000]
Output:      output/mandel.bmp
0.29user 0.01system 0:00.30elapsed 99%CPU (0avgtext+0avgdata 12204maxresident)k
0inputs+6152outputs (0major+2864minor)pagefaults 0swaps
sethan@sethan-700Z3A-700Z4A-700Z5A-700Z5B:~/4200/assignment4$
```

    This took almost no time, and after examining the image it is obvious that in this area there are very few computations to be made.

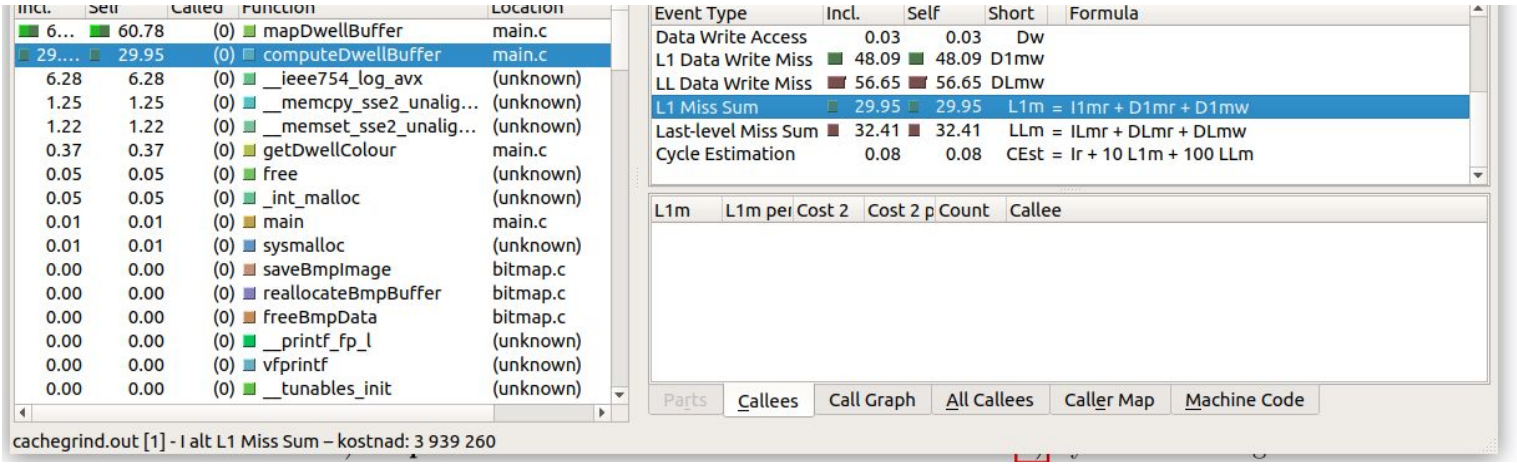
    Changed zoom level:

```
sethan@sethan-700Z3A-700Z4A-700Z5A-700Z5B:~/4200/assignment4$ make S=0.5 time
/usr/bin/time mandel/mandel output/mandel.bmp -r 1024 -x 0.5 -y 0.5 -s 0.5 -c 1
-i 512
Center:      [0.500000,0.500000]
Zoom:        200%
Iterations:  512
Window:      Re[-1.000000,0.000000], Im[-0.500000,0.500000]
Output:      output/mandel.bmp
24.68user 0.01system 0:24.70elapsed 99%CPU (0avgtext+0avgdata 12204maxresident)k
0inputs+6152outputs (0major+2864minor)pagefaults 0swaps
sethan@sethan-700Z3A-700Z4A-700Z5A-700Z5B:~/4200/assignment4$
```

    This took about twice as long, my guess is that some parts of the image went

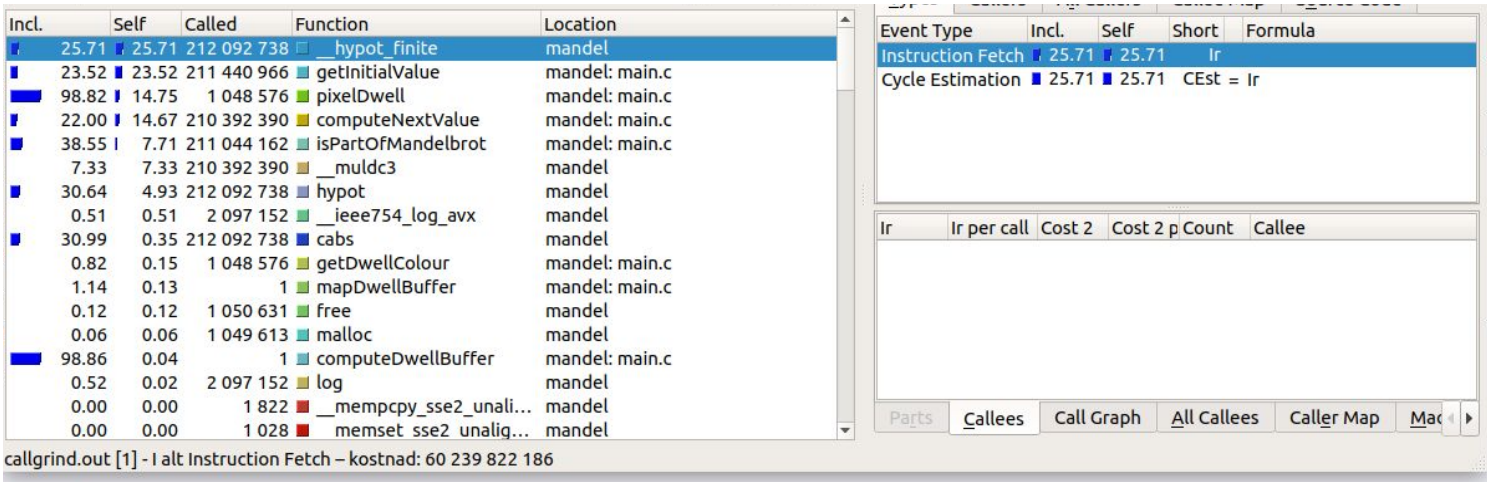
above what was previously not being drawn or computed which resulted in more computations.

2. Task 2.
    a. .
    b.



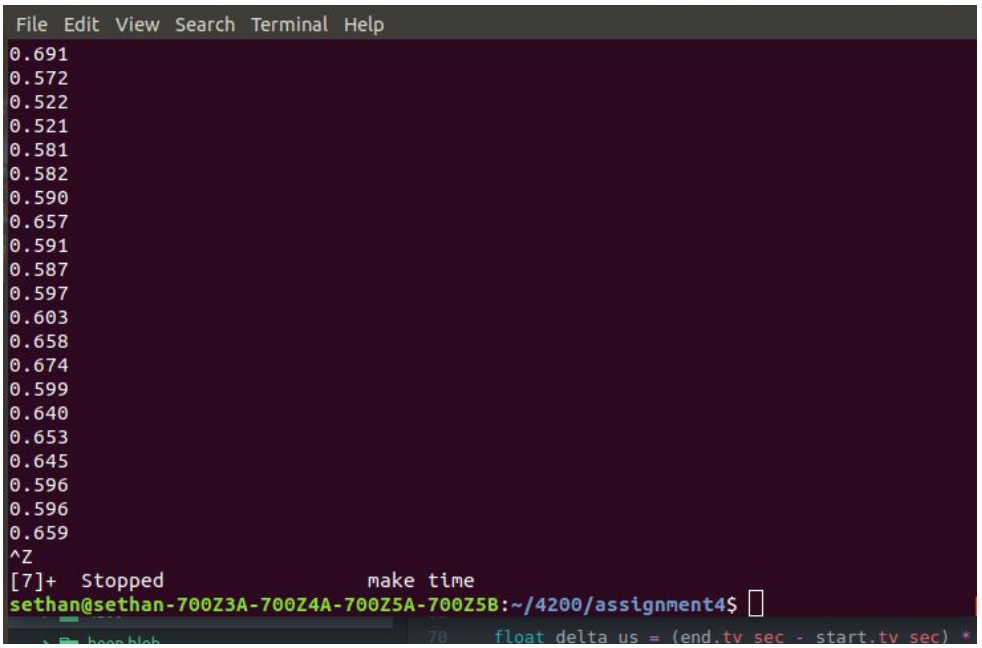
The two methods mapDwellBuffer and computeDwellBuffer, make up of around 90 percent of all cache misses.

c. .
d.



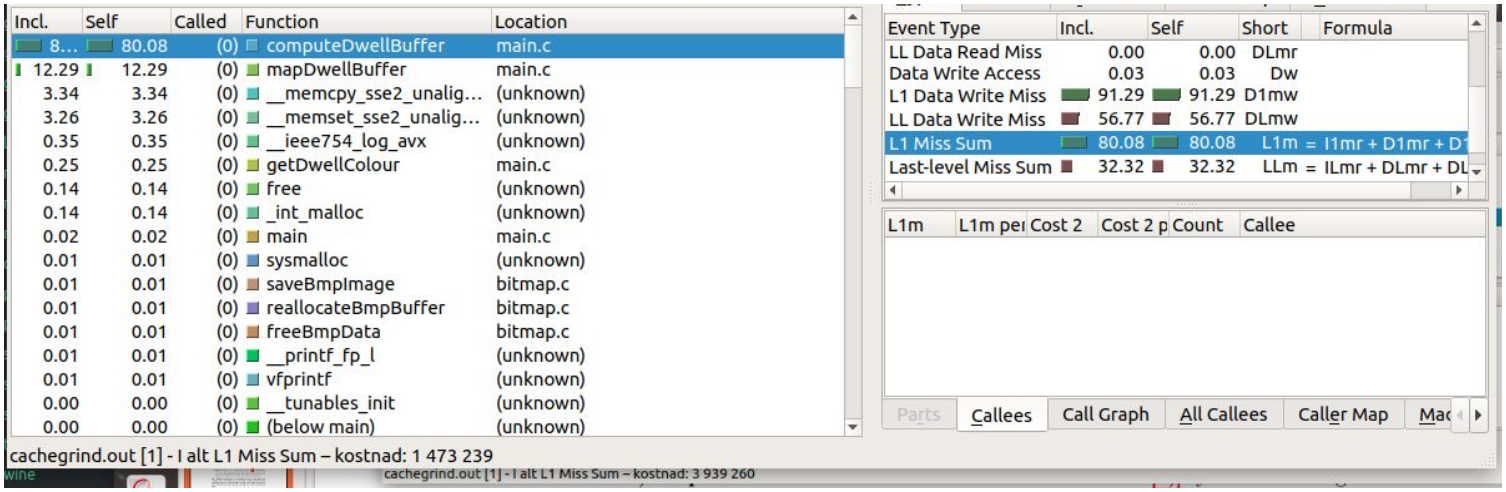
The functions __hypot_finite, getInitialValue, pixelDwell, and computeNextValue takes a lot the runtime almost 80% of it.

e.



Times measured at getInitialValue. The time measuring code was removed to increase performance.

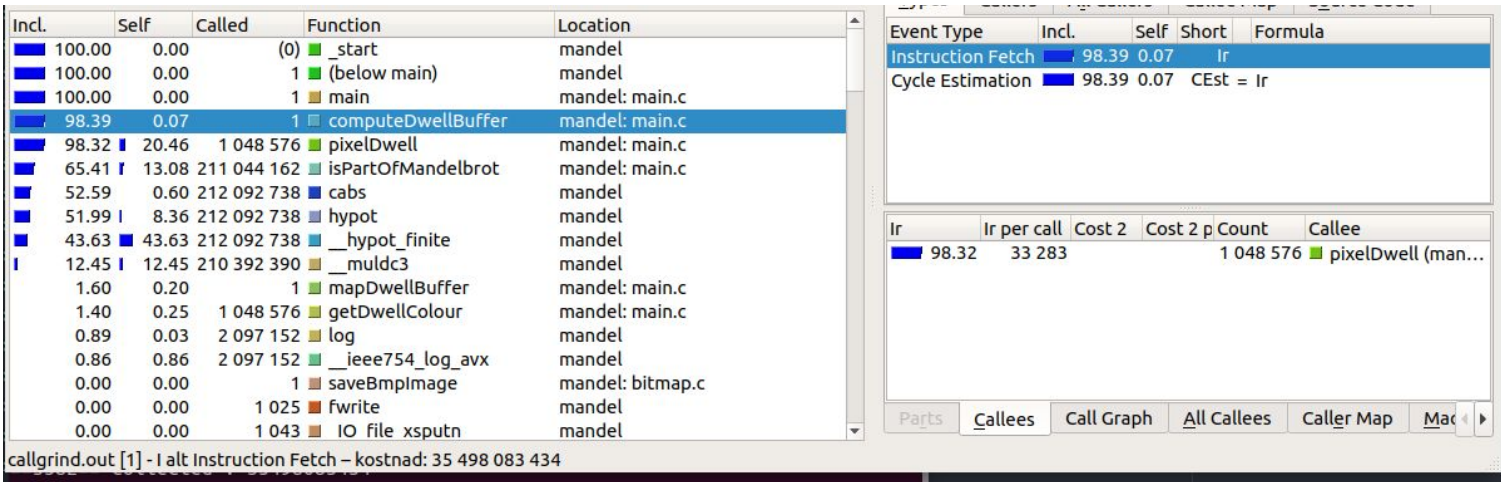
3. Task 3
   a. .
   b.



Just looking at it, computeDwellBuffers share of cachemisses went up, but looking at L1 miss sum it is now 1,5 million, and it was 4 million before. So its an improvement.

   c. .

d.



Incl.	Self	Called	Function	Location
100.00	0.00	(0)	_start	mandel
100.00	0.00	1	(below main)	mandel
100.00	0.00	1	main	mandel: main.c
98.39	0.07	1	computeDwellBuffer	mandel: main.c
98.32	20.46	1 048 576	pixelDwell	mandel: main.c
65.41	13.08	211 044 162	isPartOfMandelbrot	mandel: main.c
52.59	0.60	212 092 738	cabs	mandel
51.99	8.36	212 092 738	hypot	mandel
43.63	43.63	212 092 738	__hypot_finite	mandel
12.45	12.45	210 392 390	__muldc3	mandel
1.60	0.20	1	mapDwellBuffer	mandel: main.c
1.40	0.25	1 048 576	getDwellColour	mandel: main.c
0.89	0.03	2 097 152	log	mandel
0.86	0.86	2 097 152	__ieee754_log_avx	mandel
0.00	0.00	1	saveBmpImage	mandel: bitmap.c
0.00	0.00	1 025	fwrite	mandel
0.00	0.00	1 043	IO_file_xsputn	mandel

Event Type	Incl.	Self	Short	Formula
Instruction Fetch	98.39	0.07	Ir	
Cycle Estimation	98.39	0.07	CEst = Ir	

Ir	Ir per call	Cost 2	Cost 2 p	Count	Callee
98.32	33 283			1 048 576	pixelDwell (man…

Parts | Callees | Call Graph | All Callees | Caller Map | Mac

callgrind.out [1] - I alt Instruction Fetch – kostnad: 35 498 083 434

The improvements i made also included removing two functions and doing them instead inline so they won't show up here on this list. The total instruction fetch cost went from 60 billion to 35.5 billions. And the time i got now is

```
sethan@sethan-700Z3A-700Z4A-700Z5A-700Z5B:~/4200/assignment4$ make time
rm -f mandel/.flags_*
rm -f mandel/mandel
rm -f src/main.o src/libs/bitmap.o
gcc -O0 -static  -Isrc/libs -Isrc -c src/main.c -o src/main.o
gcc -O0 -static  -Isrc/libs -Isrc -c src/libs/bitmap.c -o src/libs/bitmap.o
gcc -O0 -static src/main.o src/libs/bitmap.o -lm -lrt -o mandel/mandel
/usr/bin/time mandel/mandel output/mandel.bmp -r 1024 -x 0.5 -y 0.5 -s 1.0 -c 1
-i 512
Center:      [0.500000,0.500000]
Zoom:        100%
Iterations:  512
Window:      Re[-1.500000,0.500000], Im[-1.000000,1.000000]
Output:      output/mandel.bmp
6.67user 0.01system 0:06.68elapsed 99%CPU (0avgtext+0avgdata 12204maxresident)k
0inputs+6152outputs (0major+2863minor)pagefaults 0swaps
sethan@sethan-700Z3A-700Z4A-700Z5A-700Z5B:~/4200/assignment4$
```

6.68 seconds.