Task 1a

Using the default values as a baseline, the execution time measured to 35.89 seconds.

Task 1b

Tests:

- resolution=500, iterations=50 -> exec time = 0.36 seconds
- resolution=5000, iterations=50 -> exec time = 33.37 seconds
- resolution=500, iterations=500 -> exec time = 2.48 seconds
- resolution=5000, iterations=500 -> exec time = 207.63seconds

When the resolution increase by x the number of pixels increase by x^2. This makes so the amount of data to be calculated and processed is bigger. The iterations has not such a big impact as probably when the resolution is increased the amount of cache misses and memory access is increased drastically.

Task 2b

The methods mapDwellBuffer and computeDwellBuffer utilises the cache badly with many misses in both read and write.

Write misses

Incl.	Self	Called	Function	Location
49…	49.59	(0)	mapDwellBuffer	main.c
48…	48.09	(0)	computeDwellBuffer	main.c

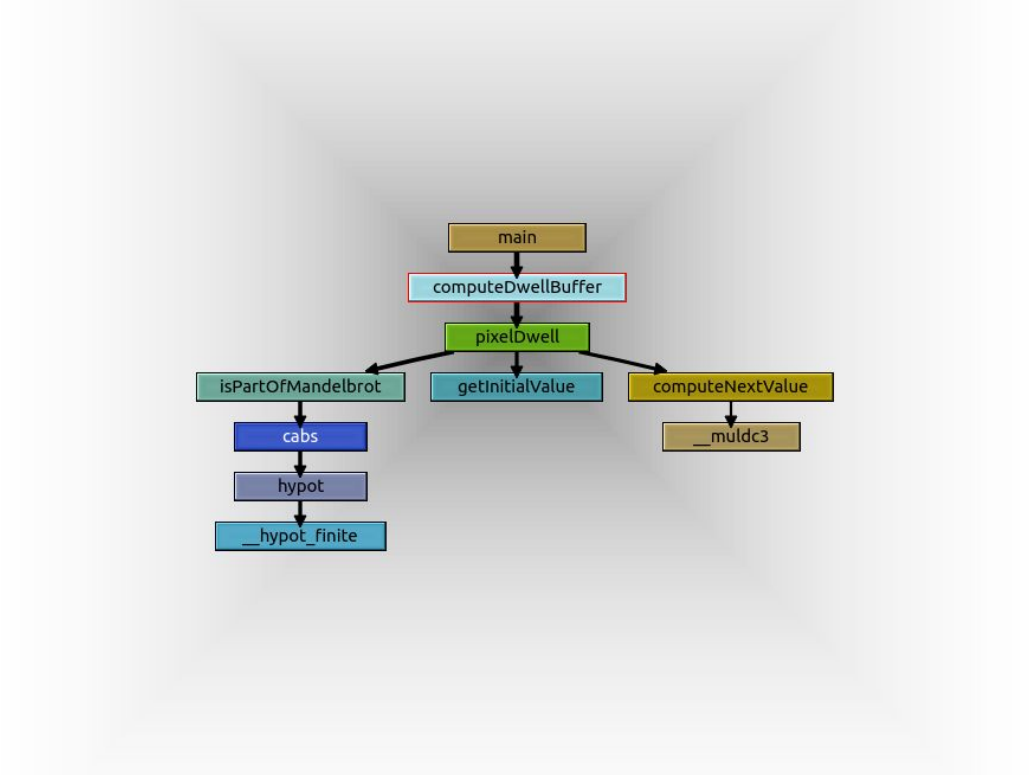
Read misses

Incl.	Self	Called	Function	Location
99.35	99.35	(0)	mapDwellBuffer	main.c

Task 2d

Here we see the pixelDwell method is called 1 million times. As the call graph shows the pixelDwell, part of the computeDwellBuffer, takes about 99.1% of the execution. Further the isPartOfMandelbrot uses 40% of the execution, getInitialValue 24% and computeNextValue 22%.

Incl.	Self	Called	Function
100.00	0.00	(0)	_start
100.00	0.00	1	(below main)
100.00	0.00	1	main
99.14	0.04	1	computeDwellBuffer
99.10	12.38	1 048 576	pixelDwell
39.94	7.26	211 044 162	isPartOfMandelbrot
32.84	0.36	212 092 738	cabs
32.48	5.11	212 092 738	hypot
27.37	27.37	212 092 738	__hypot_finite
24.36	24.36	211 440 966	getInitialValue
22.43	13.39	210 392 390	computeNextValue
9.04	9.04	210 392 390	__muldc3



Task 2e

I chose the computeDwellValue function, this uses 32.021 seconds to run. L1 misses 294 000 accesses.

Event Type	Incl.	Self	Short
Instruction Fetch	5 771 826	5 771 826	Ir
L1 Instr. Fetch Miss	6	6	I1mr
LL Instr. Fetch Miss	6	6	ILmr
Data Read Access	3 410 446	3 410 446	Dr
L1 Data Read Miss	31 897	31 897	D1mr
LL Data Read Miss	0	0	DLmr
Data Write Access	524 817	524 817	Dw
L1 Data Write Miss	262 142	262 142	D1mw
LL Data Write Miss	32 384	32 384	DLmw
L1 Miss Sum	294 045	294 045	L1m
Last-level Miss Sum	32 390	32 390	LLm

Task 3b

After optimization the function used 11.999 seconds, and only 32000 L1 cache misses, a reduction from 294 000.

Event Type	Incl.	Self	Short
Instruction Fetch	5 771 826	5 771 826	Ir
L1 Instr. Fetch Miss	6	6	I1mr
LL Instr. Fetch Miss	6	6	ILmr
Data Read Access	3 410 446	3 410 446	Dr
L1 Data Read Miss	66	66	D1mr
LL Data Read Miss	0	0	DLmr
Data Write Access	524 817	524 817	Dw
L1 Data Write Miss	32 896	32 896	D1mw
LL Data Write Miss	32 384	32 384	DLmw
L1 Miss Sum	32 968	32 968	L1m
Last-level Miss Sum	32 390	32 390	LLm
Cycle Estimation	9 340 506	9 340 506	CEst

Task 3d

I did not achieve any optimization this task, used 12.126 seconds (about the same as 3b), this may be cause when I optimized the computeDwellValue for cache misses I optimized for calls aswell. From the timing

Incl.	Self	Called	Function
100.00	0.00	(0)	_start
100.00	0.00	1	(below main)
100.00	0.00	1	main
98.51	19.49	1	computeDwellBuffer
62.20	0.69	53 025 864	cabs
61.50	9.68	53 025 864	hypot
51.83	51.83	53 025 864	__hypot_finite
17.14	17.14	52 600 784	__muldc3

The computeDwellBuffer have increased self-contained instructions since so little as possible branching.