

Toward Maximally Useless Computation

Thermal Computing group*

Den Frie Vilje Research, Denmark · github.com/den-frie-vilje/hotfix

ABSTRACT. Every watt a computer draws ends up as heat, and what it computes on the way is, thermodynamically, optional. We ask how little a heating computation can mean. We set out three criteria for useless computation, which are occupancy, speed and meaninglessness, and assess five candidate calculations on a desktop PC. We present a calculation, $x \leftarrow x/2 + 2^{-126}$, which settles on 2^{-125} within about 150 steps and from then on returns its input unchanged, and we prove that no multiply-add with a positive multiplier can settle lower while every value remains a normal 32-bit float. On an NVIDIA GeForce RTX 5070 Ti it runs 21.5 trillion multiply-adds a second, every one of them returning 2^{-125} , at about the same board power as the calculation it replaces, which computed infinity. On the processor it draws slightly more. Once settled, it erases no information, so Landauer's bound on the heat it must dissipate is zero. It dissipates the heat anyway. We consider this a success.

Keywords: heating, floating point, subnormal numbers, thermodynamics of computation, offices

1 Introduction

For as long as computers have run warm, the field has asked what they should compute. We ask the opposite question. Hotfix [5] is a Windows application that warms a room by keeping a PC's processor and graphics card busy while nothing else needs them. By the first law of thermodynamics, the electrical energy those parts draw leaves them as heat, whatever they compute. If the result is going to be thrown away anyway, how little can it mean?

This report makes the following contributions:

- three criteria for useless computation (Section 3);
- an evaluation of five candidate calculations against them (Section 4);
- a calculation whose answer, 2^{-125} , is as close to nothing as a processor will compute at full speed, with a proof that no calculation of its kind does better (Section 5);
- measurements showing that meaning less costs no heat (Section 6);
- an open-source implementation, and a test that holds it to all of the above [5].

2 Background

2.1 Heat

A PC converts practically all the electrical energy it draws into heat in the room it stands in, so as a heater it is exactly as efficient as any other resistive heater. Landauer [1] showed that a computation must dissipate at least $k_B T$

ln 2 of heat for each bit of information it erases, about 2.8×10^{-21} J at room temperature. The bound applies only to logically irreversible steps. A step that erases nothing has no lower bound at all.

2.2 Floating point

A normal 32-bit float carries a 24-bit significand and an exponent from -126 to 127 [2]. The smallest positive normal float is $2^{-126} \approx 1.18 \times 10^{-38}$. Below it lie the subnormal numbers, down to 2^{-149} , which trade precision for range. Processors often handle subnormal operands and results on a slow path, in microcode, unless told to flush them to zero [3]. Direct3D's floating-point rules let graphics cards flush subnormal 32-bit values to zero [4]. Either way, a calculation that strays into subnormal numbers does less work per second, and makes less heat, than one that stays normal.

3 Criteria

We hold a candidate calculation to three criteria.

C1, occupancy. It must keep every arithmetic unit busy. On the processor that means enough independent chains of fused multiply-adds to fill both vector pipelines of every core; on the graphics card, enough threads to fill every streaming multiprocessor. It must not touch memory beyond registers, so that the caches and memory bandwidth remain available to the user's own work.

C2, speed. Every value must stay a normal float, so that no cycle is lost to the slow path of Section 2.2.

C3, meaninglessness. Its answer must be as close to nothing as possible, and nobody may read it.

4 Candidates

Table 1 lists the candidates. Two of them shipped. Hotfix 0.1.0 ran $x \leftarrow 0.9999999x + 0.0000001$ on both parts. On the processor its eight chains per thread settled on 0.58886087 after 9,263,816 steps, about a hundredth of a second, at which point the change at each step was smaller than half a unit in the last place and every step returned its input. That satisfied C1 and C2 but not C3: 0.58886087 is not nothing.

On the graphics card, the same calculation summed its eight chains and wrote the sum back for the next pass to start from, so each pass multiplied the stored value by

about eight. In simulation the value overflowed to infinity after 43 passes, about a seventh of a second of heating, and stayed there. Infinity satisfies C2 in the sense that the hardware handles it at full speed. It is, however, the opposite of nothing.

We also considered, without building them, $x \leftarrow 0 \cdot x + 0$, whose answer is exactly nothing but which an optimising shader compiler is entitled to delete, and any calculation settling below 2^{-126} , which fails C2 by Section 2.2. Figure 1 places the answers on a number line.

Candidate	Settles on	Processor ^a	Graphics card ^b	Verdict
Doing nothing	Nothing	Idle, 23–47 W	Idle, 50–58 W	Fails C1. Cold.
$x \times 0.9999999 + 0.0000001$ Hotfix 0.1.0, processor	0.58886087, after 9.3 million steps	121–122 W		Fails C3.
The same, eight chains summed Hotfix 0.1.0, graphics card	$+\infty$, after 43 passes		170–171 W	Fails C3 emphatically.
$x \times 0 + 0$	0	Not built		May be deleted by the compiler.
Anything settling below 2^{-126}	A subnormal number	Not built		Fails C2.
$x \times \frac{1}{2} + 2^{-126}$ Hotfix 0.1.1, both parts	2^{-125} , within about 150 steps	126–130 W	163–170 W	Selected.

Table 1. Candidate calculations. ^aPackage power, the calculation running flat out on all 20 cores, two runs. ^bBoard power with Hotfix running, two runs. Idle ranges were measured on the same PC on the same afternoon. Section 6 describes the setup.

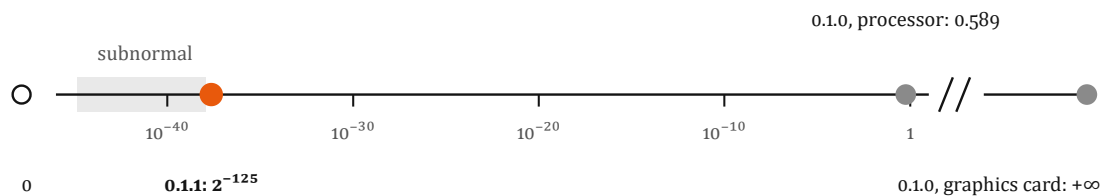


Figure 1. Distance from nothing, on a logarithmic scale. Left is better.

5 The selected calculation

Hotfix 0.1.1 runs, on both parts,

$$x_{n+1} = \frac{x_n}{2} + 2^{-126}. \quad (1)$$

Writing $e_n = x_n - 2^{-125}$ gives $e_{n+1} = e_n/2$, so each step halves the distance to the answer, and from any starting value up to 8 the chain lands on 2^{-125} exactly within about 150 steps, once the remaining distance rounds away. There it stays, because

$$2^{-125} \times \frac{1}{2} + 2^{-126} = 2^{-126} + 2^{-126} = 2^{-125} \quad (2)$$

exactly, with or without fused rounding, and every value involved, including the product, is a normal float. Written out, the answer is 0.000 000 000 000 000 000 000 000 000 000 000 000 000 023 509 887.

Theorem 1 (*no smaller nothing*). A calculation $x \leftarrow m \cdot x + c$ with $0 < m < 1$, in which every value is a normal 32-bit float, cannot settle on an answer smaller in magnitude than 2^{-125} .

Proof. At its answer, $x = m \cdot x + c$, so $(1 - m)x = c$, and x , $m \cdot x$ and c share a sign. The product and the constant are normal floats, so each is at least 2^{-126} in magnitude, and $|x| = |m \cdot x| + |c| \geq 2^{-125}$. Equation (2) shows the bound is attained. ■

Remark 1. With $-1 < m < -\frac{1}{2}$ a calculation can settle strictly between 2^{-126} and 2^{-125} , approaching its answer from alternate sides. We consider bouncing undignified.

Listing 1 shows the processor's inner loop. Each of a PC's logical processors runs one such thread, at idle priority, with eight independent chains of eight floats each, so that both fused multiply-add pipelines stay full. The chains start on the answer, so on the processor every step returns its input from the first.

```
// Hotfix 0.1.1, src/Hotfix/Heater.cs (abridged)
var m = Vector256.Create(Burn.Half); // 1/2
var c = Vector256.Create(Burn.Tiny); // 2^-126
var a = Vector256.Create(Burn.Answer); // 2^-125
Vector256<float> b = a, d = a, e = a, f = a,
                g = a, h = a, k = a;
for (int i = 0; i < 1_000_000; i++)
{
    a = Vector256.FusedMultiplyAdd(a, m, c);
    b = Vector256.FusedMultiplyAdd(b, m, c);
    // ... and d, e, f, g, h, k likewise
}
sink = (a + b + d + e + f + g + h + k)
        .GetElement(0); // never read
```

Listing 1. The processor's calculation, one million steps between checks on whether the user needs the core back.

Listing 2 shows the graphics card's compute shader, which runs over 2^{21} threads per pass. Its chains must start apart, or the shader compiler would merge them into one and fail C1; they start at a , $a + 1$, ..., $a + 7$ and converge within the first 150 of each pass's 4,096 steps. The thread writes back the mean of its chains, which is the answer again, so the next pass starts where this one finished.

```
// Hotfix 0.1.1, the Burn compute shader (abridged)
float a = buffer[ThreadIds.X], b = a + 1, ...,
      h = a + 7;
for (int i = 0; i < 4096; i++)
{
    a = a * Half + Tiny;
    // ... and b to h likewise
}
buffer[ThreadIds.X] =
    (a + b + ... + h) * 0.125f; // the mean
```

Listing 2. The graphics card's calculation. The buffer starts on the answer and never leaves the card.

6 Evaluation

6.1 Setup

We measured on a Dell Pro Max Tower T2 with an Intel Core Ultra 7 265K (20 cores, 20 threads) and an NVIDIA GeForce RTX 5070 Ti, running Windows 11 version 25H2, in an office that was cold. Package power came from the processor's RAPL counter through Windows' Energy Meter, and board power from `nvidia-smi`. Other applications ran throughout, as they would in use, using about 15 to 20 per cent of the processor.

6.2 Heat

To compare the two processor calculations without Hotfix's own yielding getting in the way, a harness ran each flat out on all 20 threads for nine seconds, measuring the last six, alternating old and new twice. The old calculation drew 121 and 122 W, the new one 126 and 130 W. On the graphics card we ran Hotfix itself, alternating 0.1.0 and 0.1.1: the old shader drew 171 and 170 W, the new one 163 and 170 W. We conclude that computing 2^{-125} makes at least as much heat as computing 0.58886087 or infinity, and slightly more on the processor. Two runs each do not support a finer conclusion, and we do not draw one.

6.3 Throughput and correctness

The answer check [5] runs the shipped shader on the shipped buffer. On the RTX 5070 Ti all 2,097,152 threads returned exactly 2^{-125} after three passes, and fifty further passes ran at 21.5 and 21.9 trillion multiply-adds a second in two runs. In continuous integration, on a GitHub-hosted runner with no graphics card, the same check passed on Windows' software renderer for 4,096 threads.

7 Discussion

Landauer accounting. Once settled, the calculation maps 2^{-125} to 2^{-125} : it is the identity, which erases no information, so Landauer's bound on the heat it must dissipate is zero. On the processor that holds from the

first step. On the graphics card, the first 150 or so of each chain's 4,096 steps per pass converge from where the chain started, and those steps do erase information; across the eight chains, about 3 per cent of all steps. The remaining 97 per cent are, thermodynamically, free. The graphics card nevertheless dissipates about 5 pJ per multiply-add above idle (115 W over 21.5×10^{12} a second), roughly two billion times the Landauer cost of erasing one bit, for operations that erase none. All of Hotfix's heat is therefore surplus to thermodynamic requirements, which is what a heater is for.

Usefulness. Hotfix writes the processor's answer to a field that nothing reads, and the graphics card's to a buffer that is never copied back to the processor. As far as we can determine, the answer has never been observed in use.

8 Limitations

We measured one PC, with other applications running, two runs per condition. Heat is not comfort: a room's temperature depends on its size, its insulation and its occupants, none of which we controlled. Hotfix also turns its heat down while the user's own applications are busy, which this report does not evaluate; the yield check in the repository does [5]. The research took one afternoon.

9 Future work

Two directions threaten the results reported here. The first lends the same watts to volunteer climate research [5, issue 1]; the second heats only while electricity is cheap and the grid is clean, using Energinet's open price and emissions data [6] [5, issue 2]. Both would make the computation more useful, and we pursue them with reluctance.

Acknowledgements

We thank the office, for being cold.

Funding

This work received no funding. It consumed some.

Code and data

All code, including the answer check that reproduces Section 6.3 and the yield check, is MIT-licensed at github.com/den-frie-vilje/hotfix. To reproduce Section 6.3, run:

```
dotnet run --project tests/AnswerCheck -c Release
```

References

- [1] R. Landauer. Irreversibility and heat generation in the computing process. *IBM Journal of Research and Development*, 5(3):183–191, 1961. doi:10.1147/rd.53.0183
- [2] IEEE. *IEEE Standard for Floating-Point Arithmetic*, IEEE Std 754-2019. doi:10.1109/IEEESTD.2019.8766229
- [3] Intel Corporation. *Intel 64 and IA-32 Architectures Optimization Reference Manual*, on denormal numbers and the flush-to-zero and denormals-are-zero modes.
- [4] Microsoft. *Direct3D 11.3 Functional Specification*, floating-point rules. microsoft.github.io/DirectX-Specs
- [5] Den Frie Vilje. *Hotfix*, version 0.1.1, 2026. github.com/den-frie-vilje/hotfix
- [6] Energinet. *Energi Data Service*: datasets DayAheadPrices, CO2Emis and CO2EmisProg. energidataservice.dk

*Correspondence: github.com/den-frie-vilje/hotfix/issues. This report parodies a register, not a company; Den Frie Vilje Research is not affiliated with Microsoft Research. Windows, Direct3D, Intel Core and NVIDIA GeForce are trademarks of their owners.