Fun with CBM arithmetics

Moderator: Moderators

Fun with CBM arithmetics

by **Mike** » Sun Dec 08, 2013 6:58 pm

Just thought I'd like to share this one:

CODE: SELECT ALL

```
1 FORT=-100TO100
2 PRINT3/4+T/1E9
3 NEXT
```

Look what happens after '.75' is output.

Mike
Herr VC

Posts: 5111
Joined: Wed Dec 01, 2004 1:57 pm
Location: Munich, Germany
Occupation: electrical engineer

Re: Fun with CBM arithmetics

by **wimoos** » Mon Dec 09, 2013 8:05 am

I reckon this behaviour has to do with rounding the accumulator.

The remarkable thing is that when you reverse the calculation, and do a little rounding, the oddity is compensated:

CODE: SELECT ALL

```
X=3/4+T/1E9
X=X-3/4
X=X*1E9
X=SGN(X)*INT(ABS(X)+.5)
```

Returns X equal to T.

Regards,

Wim

VICE; selfwritten 65asmgen; tasm; maintainer of WimBasic

wimoos
Vic 20 Afficionado

Posts: 360
Joined: Tue Apr 14, 2009 8:15 am
Website: http://wimbasic.webs.com
Location: Netherlands
Occupation: farmer

Re: Fun with CBM arithmetics

by **Jeff-20** » Fri Dec 13, 2013 5:26 pm

I guess I shouldn't calculate my taxes on the VIC.

High Scores, Links, and Jeff's Basic Games page.

Jeff-20
Denial Founder

Posts: 5763
Joined: Wed Dec 31, 1969 6:00 pm

Re: Fun with CBM arithmetics

by **Mike** » Sat Dec 14, 2013 2:14 am

> **Jeff-20 wrote:**
> I guess I shouldn't calculate my taxes on the VIC.

For the moment being, yes.

However, I've found strong hints, that only the conversion of CBM floats to decimal output is affected. C64, C16/C116/+4 and even the V4 machines show the same error, but the C128 gets it right, so Commodore got aware of the issue at some time.

I'm investigating further, and want to provide a solution in the next time.

Mike
Herr VC

Posts: 5111
Joined: Wed Dec 01, 2004 1:57 pm
Location: Munich, Germany
Occupation: electrical engineer

Re: Fun with CBM arithmetics

by **FD22** » Sat Dec 14, 2013 2:11 pm

I'd be inclined to suspect Monte Davidoff, since he wrote the FP maths package for Microsofts' Altair BASIC, and it's that code that Commodore licensed for the majority of their 8-bit machines - with the exception being the C128 which had BASIC 7 largely re-written from scratch (and that, presumably, included the math logic).

FD22
Vic 20 Hobbyist

Posts: 148
Joined: Mon Feb 15, 2010 12:31 pm

Re: Fun with CBM arithmetics

by **groepaz** » Sat Dec 14, 2013 7:38 pm

it's probably just a different way of implementing floats - with different kind of errors (every implementation has those by principle)

> I guess I shouldn't calculate my taxes on the VIC.

you should never do that on anything that uses floating point arithmetics. financial packages usually use so called "bignum" libraries for that (which work eg in BCD with a fixed resolution)

you should be able to find demonstrations on how these errors show in excel too, for example

I'm just a Software Guy who has no Idea how the Hardware works. Don't listen to me.

groepaz
Vic 20 Scientist

Posts: 1255
Joined: Wed Aug 25, 2010 5:30 pm

Re: Fun with CBM arithmetics

by **Mike** » Sun Dec 15, 2013 2:45 am

groepaz,

this is not just the occasional rounding error with 1, maybe 2 units wrong in the lowest place we're talking about here. It's a *defect* in the float to ASCII conversion which happens under reproducable circumstances and which, when happens, outputs a grossly wrong result.

If you enter **PRINT.750000059** (five zeroes) and get back **.75000003** as a result, that is just not acceptable.

Mike
Herr VC

Posts: 5111
Joined: Wed Dec 01, 2004 1:57 pm
Location: Munich, Germany
Occupation: electrical engineer

Re: Fun with CBM arithmetics

by **groepaz** » Sun Dec 15, 2013 1:21 pm

well, i dont know according to what standard (if any) the floats in cbm basic have been implemented, so its hard to tell whether this is acceptable or not.

and dont forget that the typical error is 1 or 2 _binary_ units (powers of two!) - and the decimal/float conversion involves multiplications and divisions - errors in the range you are showing there are well expected. (again: you can find similar "strange errors" in excel. or the windows calc.exe. or pretty much everything =P)

I'm just a Software Guy who has no Idea how the Hardware works. Don't listen to me.

groepaz
Vic 20 Scientist

Posts: 1255
Joined: Wed Aug 25, 2010 5:30 pm

Re: Fun with CBM arithmetics

by **Mike** » Sun Dec 15, 2013 4:04 pm

> **groepaz wrote:**
> and dont forget that the typical error is 1 or 2 _binary_ units (powers of two!)

Yes, that's what I meant in the first place.

> well, i dont know according to what standard (if any) the floats in cbm basic have been implemented, so its hard to tell whether this is acceptable or not.

The implementation surely predates IEEE 754 by a few years, but generally, the routines for basic arithmetic in CBM BASIC are just fine. They round correctly to the last digit for all basic arithmetic functions (i.e. addition, subtraction, multiplication and division).

> - and the decimal/float conversion involves multiplications and divisions - errors in the range you are showing there are well expected.

No, *they* *are* *not*.

All conversion routines worth their salt don't use the standard float routines for digit extraction, only for scaling the number with the necessary powers

Mike
Herr VC

Posts: 5111
Joined: Wed Dec 01, 2004 1:57 pm
Location: Munich, Germany
Occupation: electrical engineer

of 10 to get an integer number. The digit extraction then is done with integer arithmetic. And then there's no other reason to introduce an error of 29 decimal units in the last place - as shown in the example -, other than a plain defect.

Re: Fun with CBM arithmetics

by **groepaz** » Sun Dec 15, 2013 5:36 pm

groepaz
Vic 20 Scientist

Posts: 1255
Joined: Wed Aug 25, 2010 5:30 pm

> but generally, the routines for basic arithmetic in CBM BASIC are just fine.

depends on your expectations when i made a wrapper for them and used them in C, they behaved surprisingly different to what you'd expect in some corner cases. really impossible to say if the float routines are to blame here - since we dont know what to expect =P

> All conversion routines worth their salt don't use the standard float routines for digit extraction, only for scaling the number with the necessary powers of 10 to get an integer number. The digit extraction then is done with integer arithmetic. And then there's no other reason to introduce an error of 29 decimal units in the last place - as shown in the example -, other than a plain defect.

time for disassembling

I'm just a Software Guy who has no Idea how the Hardware works. Don't listen to me.

Re: Fun with CBM arithmetics

by **Commodore Explorer** » Mon Dec 30, 2013 1:07 am

Commodore Explorer
Vic 20 Newbie

Posts: 11
Joined: Thu Jan 31, 2008 12:37 pm

It's not just immediately after ".75" is output. Keep reading:

CODE: SELECT ALL

```
.75000027
.75000027
.75000028
.75000028
.75000029
.75000029
.7500006
.75000061
.75000062
.75000063
```

At some point, a threshold is crossed, and it starts rounding differently. Very strange.

I must confess, I have never investigated floating-point arithmetic very closely, so this should be interesting. (I did try writing my own **printf()** function in C and the digit extraction for the **%f** specifier was a pain to implement. Maybe I was doing it incorrectly?)

This reminds me of a joke about a bug in the "Calculator" program that shipped with Windows 3.1.

Q: What is the difference between Windows 3.11 and Windows 3.1?
A: None! Enter **3.11 - 3.1** on Windows calculator. Press = and instead of showing the correct answer, **0.01**, it shows **0.00**.

At some point (starting with the "Calculator" that shipped with Windows 95, I think?), Microsoft fixed this bug.

Re: Fun with CBM arithmetics

by **Kweepa** » Tue Dec 31, 2013 2:49 am

I did a little investigation.
It seems to be the function that converts FAC1 to a string.
PRINT .750000059 has FAC1 E = 80 M = C0 00 00 FD on entering $DDDD which is correct.
If 0.5 <= FAC1 < 1.0, FAC1 is multiplied by 1000000000 using the routine at $DA28.
At this point, the value in FAC1 is 750000030 - so the multiply seems to lose a little accuracy.

You can see the same thing by typing PRINT 1000000000*.750000059-750000000, which returns 29.5.

Could be something to do with the FAC rounding byte...

Kweepa
Vic 20 Scientist

Posts: 1303
Joined: Fri Jan 04, 2008 5:11 pm
Location: Austin, Texas
Occupation: Game maker

Re: Fun with CBM arithmetics

by **Mike** » Tue Dec 31, 2013 9:13 am

Seems like we're zeroing in on the bug. I had hopes it would 'only' be the conversion routine which was broken, but it's (sadly) most probably the multiplication at fault: 😕

CODE: SELECT ALL

```
1 A=.75+59E-9
2 PRINTA*1E9-7.5E8,1E9*A-7.5E8
3 PRINT(A+0)*(1E9+0)-7.5E8,(1E9+0)*(A+0)-7.5E8
4 PRINT(A*1)*(1E9*1)-7.5E8,(1E9*1)*(A*1)-7.5E8
```

The left column gives the correct result of **59**, while the right column errorneously prints **29.5**. In lines 3 and 4 the '+0' and '*1' are two possible ways to force an operand out of the FAC into memory and back again, thus the rounding byte is not directly involved. Nonetheless, the multiplication is not commutative (which I already found out about in an earlier thread), I hadn't expected the current issue though.

Other numbers with both the middle bytes 0 in the mantissa seem to be affected the same way.

Mike
Herr VC

Posts: 5111
Joined: Wed Dec 01, 2004 1:57 pm
Location: Munich, Germany
Occupation: electrical engineer

Re: Fun with CBM arithmetics

by **Mike** » Tue Dec 31, 2013 3:30 pm

I've found the bug.

The circumstances that trigger the bug are a bit complicated, but I'll try to explain.

Mike
Herr VC

Posts: 5111
Joined: Wed Dec 01, 2004 1:57 pm

Location: Munich, Germany
Occupation: electrical engineer

During multiplication, this routine is called 4 times for each byte of the mantissa of one of the factors. The current mantissa byte is contained in A, and the Z flag has been set accordingly:

CODE: SELECT ALL

```
.DA59  D0 03     BNE $DA5E
.DA5B  4C 83 D9  JMP $D983
.DA5E  4A        LSR A
.DA5F  09 80     ORA #$80
.DA61  A8        TAY
.DA62  90 19     BCC $DA7D
.DA64  18        CLC
.DA65  A5 29     LDA $29
.DA67  65 6D     ADC $6D
.DA69  85 29     STA $29
.DA6B  A5 28     LDA $28
.DA6D  65 6C     ADC $6C
.DA6F  85 28     STA $28
.DA71  A5 27     LDA $27
```

Now, the first two instructions, **BNE $DA5E** and **JMP $D983** are supposed to execute a shortcut in case the mantissa byte to multiply with is 0. It is sensible to optimize for this, especially because small integer constants do contain zeroes in their lower significant parts of the mantissa.

The routine would still work without that optimization, but the branch at $DA62 would always be executed (for 8 times in total), skipping the additions to $26 .. $29 - and all the routine then does is painstakingly move the bytes $26 .. $29 over to $27 .. $29, $70, one bit at a time. Of course this can be done much faster with just four load and store instrúctions. For this another routine at $D983 is 'reused', which normally 'normalizes' the mantissa:

CODE: SELECT ALL

```
.D983  A2 25     LDX #$25
.D985  B4 04     LDY $04,X
.D987  84 70     STY $70
.D989  B4 03     LDY $03,X
.D98B  94 04     STY $04,X
.D98D  B4 02     LDY $02,X
.D98F  94 03     STY $03,X
.D991  B4 01     LDY $01,X
.D993  94 02     STY $02,X
.D995  A4 68     LDY $68
.D997  94 01     STY $01,X
.D999  69 08     ADC #$08
.D99B  30 E8     BMI $D985
.D99D  F0 F6     BEQ $D985
```

The instructions from $D983 to $D993 do as supposed, however already beginning at $D995, **LDY $68** with **STY $01,X** is a little bit dubious: Without the optimization, a 0 would end up at $26 as the result of being rotated to the right 8 times. Whether there is a 0 at $68 at all times the routine is called with this purpose cannot be guaranteed. But anyway, that is not what triggers the bug in the first place.

The routine at $DA59 is first called with a non-0 byte - for the examples in the earlier posts this is ultimately the result of adding something around 1..59 divided by 10^9 to the constant 0.75. The routine exits with C=1, which will become important now!

The next mantissa byte is zero, so now the shortcut at $D983 is called. With

A=0 and C=1 on entry, the two instructions ADC #$08 and SBC #$08 result in A=0 and C=1 again. At $D9A4, the instruction **BCS $D9BA** skips the second half of the routine, which is a good thing, however it also executes a **CLC** at $D9BA, and from there everything goes downhill.

The third mantissa byte is *also* zero, so now the shortcut gets called a second time. This time, however C is 0 (with A again being 0), which results in C=0 and A=255 after SBC #$08. The instructions from $D9A6 .. $D9B8 are now executed, *which shift the whole mantissa at least one bit to the right*!

For the remaining mantissa byte, the check for a shortcut is skipped. Whatever is finally added to the resulting mantissa, the earlier parts have been inadvertently divided by 2 before, which is exactly what can be seen as false result.

Re: Fun with CBM arithmetics

by **Kweepa** » Tue Dec 31, 2013 8:29 pm

Nice find!
The same bug is of course in the c64...

Kweepa
Vic 20 Scientist

Posts: 1303
Joined: Fri Jan 04, 2008 5:11 pm
Location: Austin, Texas
Occupation: Game maker

Post Reply

34 posts 1 2 3

Return to "Programming"

Jump to

Board index

Delete cookies All times are UTC-06:00

Fun with CBM arithmetics

Moderator: Moderators

Proposal for a patch

by **Mike** » Wed Jan 01, 2014 6:13 am

Here's my proposal for a patch:

1. It is intended to keep the optimization,
2. It must be ensured, that the routine at $D983 is always called with C=1,
3. It must be ensured, that the contents of $68 are 0 upon calling $D983.

The routine in question is buried (too) deep into the BASIC ROM, and it is not vectored.
A patch will require the replacement of the BASIC ROM.

4. DA5B JMP $D983 needs to be rerouted to a suitably 'empty' place within $C000 .. $DFFF,
5. one possible candidate might be the range $DF53 .. $DF70:

The addresses $DF53 .. $DF70 contain only $AA's and are located after two tables, which contain powers of 10 and TI conversion constants, but the conversion loop beginning at $DE66 (spanning to $DEC2) stops at Y=$24 or Y=$3C and never reaches into $DF53 .. $DF70. There are also no other references in the BASIC ROM to that address range.

[...]

Edit: I replaced the original version of the patch with a revised one (download). For the source, see a few posts further down this topic.

Last edited by Mike on Thu Feb 20, 2014 5:10 pm, edited 2 times in total.

Mike
Herr VC

Posts: 5111
Joined: Wed Dec 01, 2004 1:57 pm
Location: Munich, Germany
Occupation: electrical engineer

Re: Fun with CBM arithmetics

by **orion70** » Wed Jan 01, 2014 10:36 am

Congrats on being the first to release software in 2014

orion70
VICtalian

Posts: 4272
Joined: Thu Feb 02, 2006 4:45 am
Location: Piacenza, Italy
Occupation: Biologist

Re: Fun with CBM arithmetics

by **Mike** » Wed Jan 01, 2014 2:13 pm

> **orion70 wrote:**
> Congrats on being the first to release software in 2014

PM sent.

Mike
Herr VC

Posts: 5111
Joined: Wed Dec 01, 2004 1:57 pm
Location: Munich, Germany
Occupation: electrical engineer

Re: Fun with CBM arithmetics

by **Kweepa** » Thu Jan 02, 2014 5:03 pm

Checked out the other emulators in VICE.
BASIC 2, 3.5 and 4 suffer from this bug, but it's absent in BASIC 7.

Kweepa
Vic 20 Scientist

Posts: 1303
Joined: Fri Jan 04, 2008 5:11 pm
Location: Austin, Texas
Occupation: Game maker

Re: Fun with CBM arithmetics

by **Mike** » Fri Jan 03, 2014 8:56 am

In the last two days I checked the patched BASIC ROM in two installations of VICE on my notebook.

As a picture is more worth than a thousand words, I got the idea to use my Mandelbrot zoomer for illustrating purposes:

Here's a co-ordinate set, where both x- and y-coordinates just straddle across a critical interval:

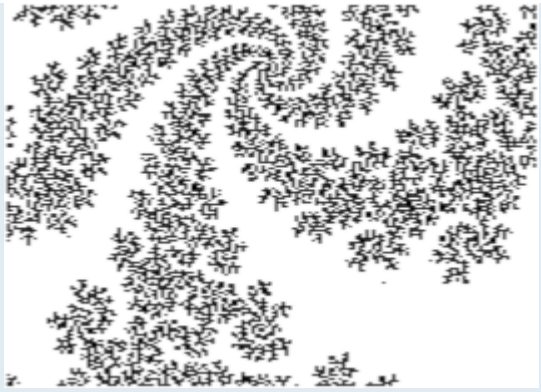
CODE: SELECT ALL

```
x_min = -0.531250128
x_max = -0.531249872
y_min =  0.531249904
y_max =  0.531250096
1024 Iterations
```

Now for the results, with the **original** ROM to the left (download) and with the **patched** ROM to the right (download):

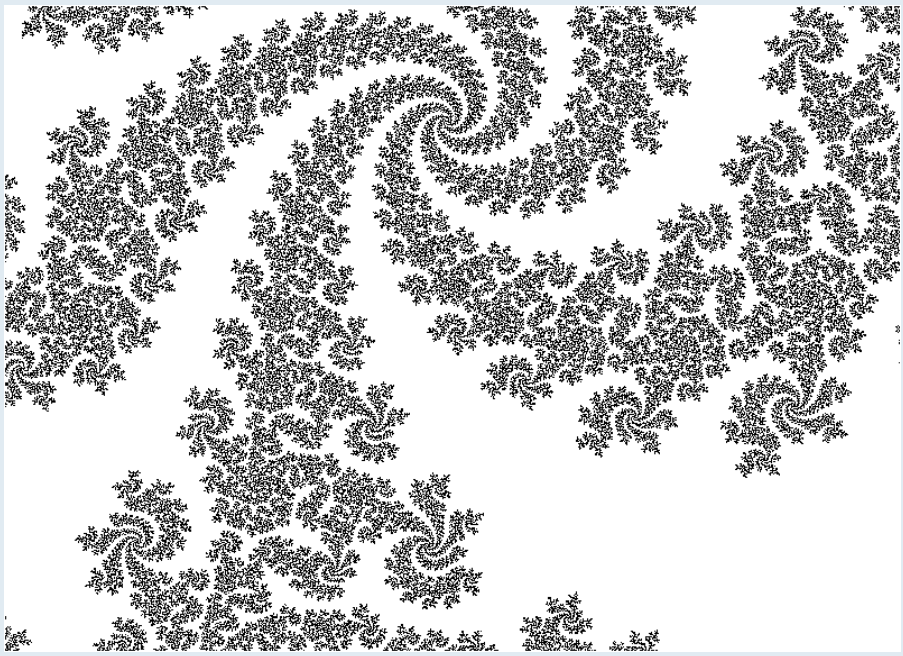
Mike
Herr VC

Posts: 5111
Joined: Wed Dec 01, 2004 1:57 pm
Location: Munich, Germany
Occupation: electrical engineer



Parts of the left picture are distorted and torn, compared to the right picture.

Here's the same co-ordinate set computed on the PC, same display algorithm, just higher resolution:



Edit: refined example

Last edited by Mike on Thu Feb 20, 2014 5:11 pm, edited 1 time in total.

Re: Fun with CBM arithmetics

by **Jeff-20** » Fri Jan 03, 2014 7:16 pm

I love it.

High Scores, Links, and Jeff's Basic Games page.

Jeff-20
Denial Founder

Posts: 5763
Joined: Wed Dec 31, 1969 6:00 pm

Re: Fun with CBM arithmetics

by **buzbard** » Sun Jan 19, 2014 1:24 pm

Nice work Mike! Thanks.

> **Kweepa wrote:**

buzbard

> Nice find!
> The same bug is of course in the c64...

Here's the same patch modified for the C64.

CODE: SELECT ALL

```
10 open2,8,2,"basic,s,r"
11 open3,8,3,"basic-p,s,w"
12 fort=40960to49151
13 get#2,a$:a=asc(a$+chr$(0))
14 ift=47708thena= 83:goto22
15 ift=47709thena=191:goto22
16 ift=48979thena=133:goto22
17 ift=48980thena=104:goto22
18 ift=48981thena= 56:goto22
19 ift=48982thena= 76:goto22
20 ift=48983thena=131:goto22
21 ift=48984thena=185
22 print#3,chr$(a);
23 next
```

The VIC and the C64 have almost exactly the same BASIC ROM just located in different memory ranges.
VIC - C000-DFFF
C64 - A000-BFFF

Ray..

Vic 20 Devotee

Posts: 213
Joined: Sun Jul 03, 2005 9:10 am

Re: Fun with CBM arithmetics

by **Mike** » Sun Jan 19, 2014 4:19 pm

On the C64, it is also possible to copy the BASIC-ROM into the RAM, and patch it there. Here's a program which does exactly this:

CODE: SELECT ALL

```
1 POKE1,PEEK(1)AND248OR7
2 FORT=40960TO53247:POKET,PEEK(T):NEXT
3 POKE47708,83:POKE47709,191
4 FORT=0TO5:READA:POKE48979+T,A:NEXT
5 POKE1,PEEK(1)AND248OR6
6 DATA133,104,56,76,131,185
```

Here's a revised version of the patch, which installs an own routine to handle the mantissa in the free range of $DF53 .. $DF70, and does not anymore 're-use' the normalize routine. Even if it now uses up some more bytes of that free space, it is now implemented as it should have been done in the first place. The jump at $DA5B now is altered to:

CODE: SELECT ALL

```
.DA5B  4C 5A DF   JMP $DF5A
```

... and the routine at $DF5A now is:

CODE: SELECT ALL

```
.DF5A  A5 29      LDA $29
.DF5C  85 70      STA $70
.DF5E  A5 28      LDA $28
.DF60  85 29      STA $29
.DF62  A5 27      LDA $27
.DF64  85 28      STA $28
.DF66  A5 26      LDA $26
```

Mike
Herr VC

Posts: 5111
Joined: Wed Dec 01, 2004 1:57 pm
Location: Munich, Germany
Occupation: electrical engineer

```
.DF68  85 27     STA $27
.DF6A  A0 01     LDY #$01
.DF6C  98        TYA
.DF6D  4A        LSR A
.DF6E  85 26     STA $26
.DF70  60        RTS
```

This is, what the original shortcut at $D983 was supposed to do, only that one fails under those circumstances I already described in the earlier postings. The following program applies the new patch to a *.bin file of the BASIC ROM:

CODE: SELECT ALL

```
10 OPEN2,8,2,"BASIC,S,R"
11 OPEN3,8,3,"BASIC-P,S,W"
12 READB,C,D
13 FORT=49152TO57343
14 GET#2,A$:A=ASC(A$+CHR$(0))
15 IFB<>TTHEN18
16 IFA<>CTHENPRINT"BAD SOURCE FILE!":GOTO20
17 A=D:READB,C,D
18 PRINT#3,CHR$(A);
19 NEXT
20 CLOSE3
21 CLOSE2
22 :
23 DATA 55900,131,90
```

I also found some interesting reference about those early implementations of Microsoft BASIC on the 65xx. The owner of the blog reconstructed one single source code repository for CBM BASIC V1, OSI BASIC, AppleSoft I, KIM-1 BASIC, CBM BASIC V2 (PET), Intellivision Keyboard Component BASIC and MicroTAN BASIC ... and with the exception of OSI BASIC (it only uses a 3-byte mantissa, thus the error can't happen there) all share that bug in the multiplication routine ...!

> **Kweepa wrote:**
> Checked out the other emulators in VICE.
> BASIC 2, 3.5 and 4 suffer from this bug, but it's absent in BASIC 7.

Yes, I saw that, too. The bug is absent in V7, because Commodore effectively disabled the optimization that caused it. Of course, that means that multiplication now is generally slower on the C128.

Even *more* interesting is, that they somehow came to the same conclusions about the carry flag as I did, but they were not quite sure about how to handle it. In the BASIC ROM, the routine is located at $8A55:

CODE: SELECT ALL

```
.8A55  D0 04     BNE $8A5B
.8A57  EA        NOP
.8A58  4C 62 89  JMP $8962
.8A5B [...]
```

... with $8962 again being the short-cut. At $8A57, they had possibly put a SEC before - and then decided to go the save route by disabling the optimization instead (only in one of five cases, the routine is still called over $8A55, all other 4 calls enter at $8A5B). Anyway they had more than enough space in the ROM to do it right, but alas. 😟

Re: Fun with CBM arithmetics

by **Mike** » Thu Dec 14, 2017 1:29 pm

> **Mike wrote:**
> A patch will require the replacement of the BASIC ROM.

In case you've been wondering what became of this, look here.

Mike
Herr VC

Posts: 5111
Joined: Wed Dec 01, 2004 1:57 pm
Location: Munich, Germany
Occupation: electrical engineer

Re: Fun with CBM arithmetics

by **Kakemoms** » Sat Feb 17, 2018 3:35 pm

I am hoping to get this patch (and others) into the SuperVixen. It will be in a copy of Basic in the external cartridge, so not a replacement ROM, but patched during copy to external RAM.

I am wondering if anyone has a list of such patches? I am thinking about a file format to get them loaded and patched during boot, so that anyone finding such problems in the future, will also be able to add their own patch.

What do you think about it?

Last edited by Kakemoms on Sat Feb 17, 2018 6:01 pm, edited 1 time in total.

Kakemoms
Vic 20 Nerd

Posts: 722
Joined: Sun Feb 15, 2015 8:45 am

Re: Fun with CBM arithmetics

by **Mike** » Sat Feb 17, 2018 5:02 pm

Regards your question of a feasible file format: you'd most probably be content with a *.prg file that executes the patch by copying the original ROM contents, doing a check whether it works on the original byte values it aims to replace, and putting the replacement bytes there. Much similar to how I proceeded with the file based patch here.

> **Kakemoms wrote:**
> I am wondering if anyone ha[s] a list of such patches?

From what I know, this is probably the first - and thus far only one - *direct* patch of the VIC-20 BASIC ROM.

There are other known bugs in the CBM BASIC interpreter, but they are mostly irrelevant or easy to avoid: broken type checks lead to overflow of the string stack in the **IF ... THEN** clause and **POS()** function with bogus string arguments, or to crashes as with the statement **PRINT 5+"A"+-5** ... then there is also a bug in the line number parse routine, which results in a warm start or crash upon entering a certain range of (invalid!) line numbers. In other cases, it is rather easy to improve the BASIC interpreter by means of a BASIC extension/wedge or USR() function - if you count the **slowness of the SQR()** function as "bug".

As I wrote earlier in the thread, the multiplication routine is buried too deep in the interpreter, and it is not vectored. A soft replacement would require a duplication of a good deal of ROM code of the expression parser down to the multiplication routine itself, and that fixed shortcut routine. We are talking about way more than 1K code here. And it also only helps

Mike
Herr VC

Posts: 5111
Joined: Wed Dec 01, 2004 1:57 pm
Location: Munich, Germany
Occupation: electrical engineer

BASIC programs: machine code programs that happen to use the arithmetic routines of the interpreter have no other choice than to call the multiply routine in the BASIC ROM directly - exactly because there is no vector pointing there - and thus will be in the dark about a soft loaded fix!

Those are the reasons I made a real job of replacing the BASIC ROM. The other alternative that came to my mind, supplying RAM under ROM, was more like taking a sledgehammer to crack a nut - at least for that single fix.

Re: Fun with CBM arithmetics

by **Kakemoms** » Sat Feb 17, 2018 6:02 pm

Kakemoms
Vic 20 Nerd

Posts: 722
Joined: Sun Feb 15, 2015 8:45 am

> **Mike wrote:**
> Those are the reasons I made a real job of replacing the BASIC ROM. The other alternative that came to my mind, supplying RAM under ROM, was more like taking a sledgehammer to crack a nut - at least for that single fix.

Well, there is no way to get the 65C02 to see the internal ROM, so it will have to be in RAM anyway. Thus a patch is straightforward..

Re: Adding 0.1 + 0.2 in CBM float

by **wimoos** » Mon Sep 24, 2018 11:29 am

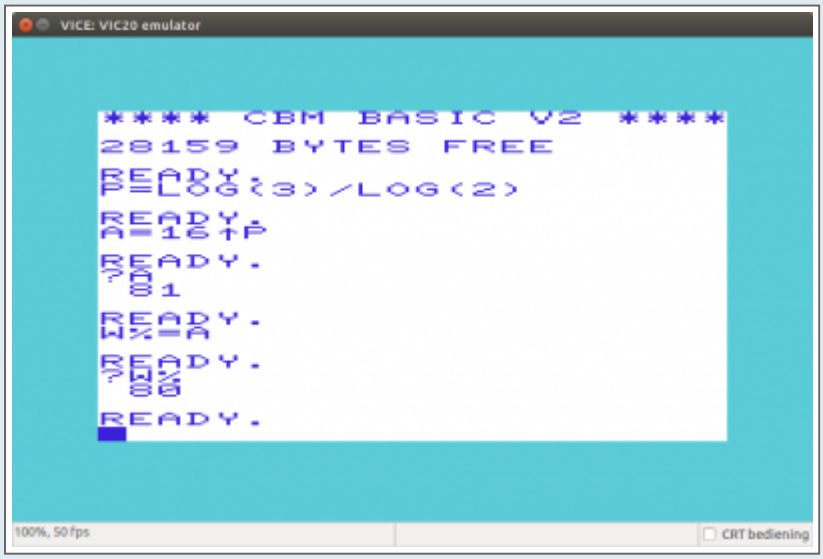
wimoos
Vic 20 Afficionado

Posts: 360
Joined: Tue Apr 14, 2009 8:15 am
Website: http://wimbasic.webs.com
Location: Netherlands
Occupation: farmer

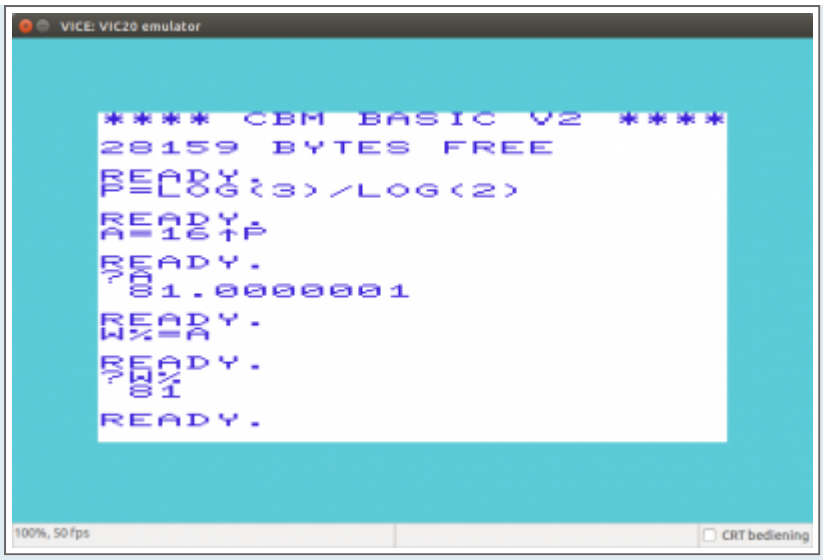
Mike,

This is just to let you know about this test I did.

I have now used your most recent patch on the Basic image. Starting VICE gives: "VIC20MEM: Error - Warning: Unknown Basic image. Sum: 31364 ($7A84)."



Reverting to the original Basic image:



Regards,

Wim.

VICE; selfwritten 65asmgen; tasm; maintainer of WimBasic

Re: Adding 0.1 + 0.2 in CBM float

by **Mike** » Mon Sep 24, 2018 1:27 pm

Hi, wimoos,

> **wimoos wrote:**
> This is just to let you know about this test I did.
>
> I have now used your most recent patch on the Basic image. Starting VICE gives: "VIC20MEM: Error - Warning: Unknown Basic image. Sum: 31364 ($7A84)."
>
> [...]

I suppose this counts as expected behaviour. It *is* an unknown BASIC ROM image as far as VICE is concerned.

> [...]
>
> CODE: SELECT ALL
>
> ```
> P=LOG(3)/LOG(2)
> A=16^P
> PRINT A    (... prints 81)
> W%=A
> PRINT W%   (... prints 80)
> ```
>
> Reverting to the original Basic image:
>
> CODE: SELECT ALL
>
> ```
> P=LOG(3)/LOG(2)
> A=16^P
> PRINT A    (... prints 81.0000001)
> W%=A
> PRINT W%   (... prints 81)
> ```

Mike
Herr VC

Posts: 5111
Joined: Wed Dec 01, 2004 1:57 pm
Location: Munich, Germany
Occupation: electrical engineer

Now, if you do slightly more complex calculations with non-integer numbers (log(3)/log(2) = 1.5849625...) in the presence of rounding errors and implementation details of a given floating point system to ultimately try to construct *integer* numbers (+/- 1 or 2 ULP) - and **then** apply the INT() function (which is implicit here in the assignment to an integer variable) ... you just get those kind of results. Again, this is expected behaviour.

The patched BASIC calculates 16^(log(3)/log(2)) as just a little bit smaller than 81. Upon output with PRINT, that value is rounded up and prints as 81, but INT() (or the assignment to an integer variable) rounds down to 80. When you calculate **A=-16^P** (note the sign-inversion!), **W%=A** for the patched BASIC yields **-81** and the original BASIC will result in **-82**, so there ...

Greetings,

Michael

Re: Fun with CBM arithmetics

by **MrSterlingBS** » Wed Aug 23, 2023 4:30 am

I am playing with the NEW BASIC ROM and MG extension and found that the NEW BASIC ROM is about 0.8% faster than the OLD one!

ATTACHMENTS

milkiway.zip
(195 Bytes) Downloaded 187 times

Last edited by Mike on Wed Aug 23, 2023 5:54 am, edited 1 time in total.

MrSterlingBS
Vic 20 Devotee

Posts: 293
Joined: Tue Jan 31, 2023 2:56 am

Post Reply

34 posts 1 2 3

Return to "Programming"

Jump to

Board index

Delete cookies All times are UTC-06:00

Fun with CBM arithmetics

Moderator: Moderators

Re: Fun with CBM arithmetics

by **MrSterlingBS** » Wed Aug 23, 2023 4:34 am

without BASIC-patch ---------- with BASIC-patch

117521 ---------- 116585
117519 ---------- 116573
117567 ---------- 116509
117508 ---------- 116551
117553 ---------- 116550

117534 ---------- 116554

100% ---------- 99,2%

MrSterlingBS
Vic 20 Devotee

Posts: 293
Joined: Tue Jan 31, 2023 2:56 am

Re: Fun with CBM arithmetics

by **Mike** » Wed Aug 23, 2023 6:05 am

> **MrSterlingBS wrote:**
> I am playing with the NEW BASIC ROM and MG extension and found that the NEW BASIC ROM is about 0.8% faster than the OLD one!
>
> without BASIC-patch ---------- with BASIC-patch
>
> [...]
>
> 100% ---------- 99,2%

That's an unintended but surely not unwelcome side effect. 😉

The routine in the patch gets called in 1 of 256 cases (when the given mantissa byte is zero) and does its job a little bit faster than the original re-used mantissa normalization routine. **Edit**: actually, the statistics are more in our favour. Quite some values processed in the program are small integer values (the screen co-ordinates in particular) which do have zeroes in their low mantissa bytes.

...

Something went wrong with "milkiway.zip". The *.prg file inside is corrupted and only 37 bytes long.

Mike
Herr VC

Posts: 5111
Joined: Wed Dec 01, 2004 1:57 pm
Location: Munich, Germany
Occupation: electrical engineer

Re: Fun with CBM arithmetics

by **MrSterlingBS** » Wed Aug 23, 2023 7:18 am

Okay, sorry for posting some source code again as text.
This version includes the faster sqr routine from WIMOS.
The graphic demo is from the FORUM64.

MrSterlingBS
Vic 20 Devotee

Posts: 293
Joined: Tue Jan 31, 2023 2:56 am

CODE: SELECT ALL

```
0 sa=828
10 forn=0to57
20 read a%:pokesa+n,a%:next
30 data 32,43,220,240,52,16,3,76
40 data 72,210,32,199,219,165,97,56
50 data 233,129,8,74,24,105,1,40
60 data 144,2,105,127,133,97,169,4
70 data 133,103,32,202,219,169,92,160
80 data 0,32,15,219,169,87,160,0
90 data 32,103,216,198,97,198,103,208
100 data 233,96
105 @on:@clr
110 poke1,60:poke2,03
115 x1=80:x2=79:y1=96:y2=95
```

Re: Fun with CBM arithmetics

by **Mike** » Wed Aug 23, 2023 8:05 am

For the *.prg file and further discussion about this demo, check out this thread: MINIGRAFIK lineart.

Mike
Herr VC

Posts: 5111
Joined: Wed Dec 01, 2004 1:57 pm
Location: Munich, Germany
Occupation: electrical engineer