MoHPC *The Museum of HP Calculators*

HP Forum Archive 21

[Return to Index | Top of Index]

WP 34S accuracy is excellent

*Message #1 Posted by **Jeff Johnson** on 1 June 2012, 8:56 a.m.*

In the somewhat pedantic area of calculator torture testing, it's refreshing to see that the WP 34S shines in the area of extreme calculations (for example http://voidware.com/calcs/torturetest.htm). It has miffed me that the HP calculators have historically not performed well in this area, but "lesser" calculators have. The HP-15C and LE rank humorously low in this area, running neck and neck with my HP-45 and -25. The HP 30b transmogrified into WP 34S leap-frogs the entire pack of commercial calculators, including the TI-89/92. The only exception I found was in the calculation of mod(73^55,31)=26, which I found on a web site a few years back and with which I have tested a number of calculators. Few are capable of calculating 73^55, and of those that can, few come up with the correct modulo remainder of 26. The TI-89/92 will calculate it correctly, as will the HP-50g, if it's in CAS mode. OK, so none of this really matters in daily use, but in the company of this hard-nosed, hard-line group, it's interesting. Someone may know how to coax the 34S into correctly performing this calculation.

Re: WP 34S accuracy is excellent

*Message #2 Posted by **Walter B** on 1 June 2012, 10:33 a.m.,*
in response to message #1 by Jeff Johnson

DBLON 73 ENTER^ 55 y^x 31 RMDR results in 2 :-(Not too bad, however, since 73^55 has 103 digits ...

Edit: May be solved this way: DBLON 73 ENTER^ 11 y^x 31 RMDR 5 y^x 31 RMDR resulting in 26 :-)
73^11 features 21 digits, well within double precision d:-)

Edited: 1 June 2012, 10:43 a.m.

Re: WP 34S accuracy is excellent

*Message #3 Posted by **C.Ret** on 1 June 2012, 10:49 a.m.,*
in response to message #1 by Jeff Johnson

I don't know how to make it on a WP34S.

But, you have to add in your calculators capable list the HP28C and HP28S.

In a few minute, the HP 28C/S Advanced Scientific Calculator may print you how exactly 73^55 is and get in the second what is the MOD(73^55,31) :

Edited: 1 June 2012, 10:51 a.m.

Re: WP 34S accuracy is excellent

Message #4 Posted by ***Eduardo Duenez*** *on 1 June 2012, 11:11 a.m.,*
in response to message #1 by Jeff Johnson

> Quote:
>
> The only exception I found was in the calculation of mod(73^55,31)=26, which I found on a web site a few years back and with which I have tested a number of calculators. Few are capable of calculating 73^55, and of those that can, few come up with the correct modulo remainder of 26. The TI-89/92 will calculate it correctly, as will the HP-50g, if it's in CAS mode. OK, so none of this really matters in daily use, but in the company of this hard-nosed, hard-line group, it's interesting. Someone may know how to coax the 34S into correctly performing this calculation.

Some comments:

Torture arguments notwithstanding, it is incredibly, egregiously wrong, from an efficient computational perspective, to compute mod(73^55,31) in the form written, by first computing the huge integer 73^55 and then finding its remainder modulo 31. The HP-50g, in particular, has a POWMOD command exactly to carry out this calculation efficiently: After setting 31 as the modulus under CAS settings the command POWMOD(73,55) quickly returns the answer (of course the 50g is fast enough that MOD(73^55,31) is also computed instantaneously, but whereas POWMOD(73,1234567890) is still instantaneous, I *somehow doubt* the 50g can handle 73^1234567890.

Eduardo

[Edited to remove erroneous reference to an HP-50g bug.]

Edited: 2 June 2012, 12:16 a.m. after one or more responses were posted

Re: WP 34S accuracy is excellent

Message #5 Posted by ***Jeff Johnson*** *on 1 June 2012, 11:26 a.m.,*
in response to message #4 by Eduardo Duenez

Eduardo,

You're absolutely correct - it's a terrible calculation to subject the calculator to, but that actually is the point of the original author of the problem (don't remember who he was) - to test the edges/limits of the hardware and algorithms.

FYI, the modulo function on the 34S (vers 3.1) is RMDR, which is the green function on the division key.

Re: WP 34S accuracy is excellent

Message #6 Posted by ***Eduardo Duenez*** *on 1 June 2012, 6:01 p.m.,*
in response to message #5 by Jeff Johnson

Quote:

> You're absolutely correct - it's a terrible calculation to subject the calculator to, but that actually is the point of the original author of the problem (don't remember who he was) - to test the edges/limits of the hardware and algorithms.

However, the problem as posed is to find mod(73^55,31)=26---not to find 73^55. The latter calculation can hardly be asked to be performed by a fixed-precision calculator. I do not know of any non-CAS calculators that can perform modular arithmetic, esp. modular exponentiation, although the 34s has internal algorithms in that regard (per Pauli's post below)---which is awesome and useful.

Eduardo

[Edit: Of course I mean fixed-precision calculators with *built-in* modular arithmetic operations. Ad-hoc programs to perform modular arithmetic can be written for basically any programmable, including the 34s as well as Valentin Albillo's HP-71B programs below. Now, if only I could reasonably afford a 71B...]

Edited: 2 June 2012, 12:12 a.m.

Re: WP 34S accuracy is excellent

*Message #7 Posted by **Walter B** on 2 June 2012, 12:28 a.m.,*
in response to message #6 by Eduardo Duenez

Quote:

> However, the problem as posed is to find mod(73^55,31)=26---not to find 73^55. The latter calculation can hardly be asked to be performed by a fixed-precision calculator.

IMHO the method explained above is a perfectly legal solution to this problem on a fixed-precision calculator - it only requires 'brains on'. So, where's the problem now still? Do you want a function solving it with 'brains off'?

Re: WP 34S accuracy is excellent

*Message #8 Posted by **Eduardo Duenez** on 2 June 2012, 12:22 p.m.,*
in response to message #7 by Walter B

Quote:

> IMHO the method explained above is a perfectly legal solution to this problem on a fixed-precision calculator - it only requires 'brains on'. So, where's the problem now still? Do you want a function solving it with 'brains off'?

Walter,

I'm not sure what you mean. My claim is that solving the original problem Mod(73^55,31) in a fixed-precision calculator can hardly be expected to be done by computing 73^55 first, then reducing modulo 31. I was the person who originally pointed out the "sane" way to do this is computing the power using modular arithmetic.

An algorithm (e.g., "brains on" approach) to solve the original problem can be written for any minimally programmable calculator, since basically all that is needed is the ability to repeatedly multiply numbers <= 30 and take reminders modulo 31 of numbers up to 900. However, the spirit of "torture tests" is to test a calculator's built-in algorithms, and it is in this sense that I interpret the question. A "brains off" solution on the 34s would be made possible by exposing its built-in modular exponentiation functions, and then the 34s would be the *only* fixed-precision calculator capable of passing this torture test.

Eduardo

Re: WP 34S accuracy is excellent

Message #9 Posted by ***Paul Dale*** *on 1 June 2012, 5:07 p.m.,*
in response to message #4 by Eduardo Duenez

There are overflow resistant routines for A times B mod C and A^B mod C in the primality tester. These are not, however, exposed to the user.

These could be wrappered up and exposed in integer mode for a couple of hundred bytes of flash. Exposing them in real mode would be more space and there would be overflow risks to deal with.

- Pauli

Re: WP 34S accuracy is excellent

Message #10 Posted by ***Eduardo Duenez*** *on 1 June 2012, 6:06 p.m.,*
in response to message #9 by Paul Dale

> Quote:
>
> There are overflow resistant routines for A times B mod C and A^B mod C in the primality tester. These are not, however, exposed to the user.
>
> These could be wrappered up and exposed in integer mode for a couple of hundred bytes of flash. Exposing them in real mode would be more space and there would be overflow risks to deal with.

Even just exposing them in integer mode would be incredibly useful. The (pseudo)primality tester is a cool application, but there are plenty more uses to be found in elementary number theory, not to mention in cryptography (mostly for didactic purposes, of course, due to word length limitations).

Eduardo

Re: WP 34S accuracy is excellent

*Message #11 Posted by **Jeff Johnson** on 1 June 2012, 11:13 a.m.,*
in response to message #1 by Jeff Johnson

Good info from Walter and C.Ret. In my tests on previous calculators, my results were based on direct keyboard calculation, which is pretty stringent. On that basis, my 28S performed the 73^55, but failed to come up with the correct remainder. ON the 34S, I tried single and double precision as Walter did, with similar results. Breaking the problem up is definitely a good work around, though.

Re: WP 34S accuracy is excellent

*Message #12 Posted by **C.Ret** on 1 June 2012, 11:26 a.m.,*
in response to message #11 by Jeff Johnson

Hi,

Yes, I also get the result by direct keyboard calculation.

The exact keyboard sequence on my HP28S was (after turning on HP82240A IR-printer) :

```
73 [space] 55 [space] 31 [space] LPOWMOD [ENTER]
```

Following this direct keyboard operation to print results

```
SWAP PRMAT SWAP PR1
```

Of course, LPOWMOD and PRMAT have been previously direct keyboard recorded in ! That what makes the big difference with actual capable calculators who have powerfull built-in facilities.

But every thing on an HP-28S is direct keyboard input, there is no USB link or SD Card exchange process !

Edited: 1 June 2012, 11:31 a.m.

Re: WP 34S accuracy is excellent

*Message #13 Posted by **Jeff Johnson** on 1 June 2012, 11:32 a.m.,*
in response to message #12 by C.Ret

C.Ret - Thanks, I will check that out. I did my check a couple of years ago on my 28S, but was unaware of the LPOWMOD function. Will revisit it now that I have this knowledge. Thanks for the info.

Re: WP 34S accuracy is excellent

*Message #14 Posted by **Paul Dale** on 1 June 2012, 5:25 p.m.,*
in response to message #1 by Jeff Johnson

I went to some effort to make the modulo reduction accurate across the entire range of single precision. There should be no single precision real value where the trig functions fail bizarrely. The modulo reduction of trig function arguments is done before before converting to radians both to avoid a source of error and to speed up the computation. The normal RMDR function should work well across the whole domain too.

Double precision can cause problems due to the extra range provided and getting failures here isn't difficult if you know what to do.

I'd argue that 73^55 modulo 31 is being calculated correctly as it stands.

```
73 ^ 55 = 3039186186495313642719567997295985951911833259711396052783359084390154265029162986660468288337710185657
```

Rounding this to sixteen significant digits gives:

```
3.039186186495314e102
```

The 34S produces this value in single precision mode.

```
3039186186495314 = 31 * 98038264080494
```

Therefore, 3.039186186495314e102 modulo 31 is zero.

- Pauli

Re: WP 34S accuracy is excellent

Message #15 Posted by ***C.Ret*** *on 3 June 2012, 5:31 a.m.,*
in response to message #14 by Paul Dale

Good news, I have here two new capable calculators, one SHARP PC-1211 Pocket Computer and one Hewlett Packard HP-41C Handled Calculator that confirm your full digit computation of 73^55.

So we have to add both on Jeff's 'capable calculator list' where already sat HP-50g, TI-89/92 and HP-28C/S.

HP-41C Program Listing:

```
    @@~~ Initiate computation :


001 LBL "POWMOD"
    STO 21 RCL Z x>y? x<>y                @ keep MAX(F,M)
    RDN ABS LOG INT 8 x<>y x>y? STOP      @ stop on error condition MAX(F,M) too large
    - 10^X STO 22                         @ store EE in R(22)
    RDN INT STO 23                        @ store  P in R(23)
    @@~~ Print  F ^ P = :                \ "~  is the alpha 'append' caracter \

020 CLA FIX 0 ARCL Y "~^" ARCL X "~ = " PRA

    @@~~ Initiale storage area R(0 to 19), set R(0) to 1, set pointer R(20)

027 Clx STO 20 1 STO 00 RDN RDN INT

    @@~~ Outer loop : storage area multiplication

034 LBL 01                              @ Reset storage area pointer
    RCl 20 FRC STO 20
038 Clx                                 @ Clear carry

    @@~~ Inner loop : process each storage register

039 LBL 02
    RCL IND 20 RCL Z * + RCL X RCL 22
    MOD STO IND 20                     @ store R(R(20)) digits without carry
    - RCL 22 / INT                     @ store carry in stack
    ISG 20 GTO 02                      @ loop for each storage register
    x=0? GTO 03

056 STO IND 20 .001 ST+ 20             @ if carry after last storage register
                                       @ expand storage register area

    @@~~ Repeat outer loop up to M multiplications

059 LBL 03
    R^ DSE 23 GTO 01                   @ loop P = R(23) times

063 RCL 21                             @ Recall M

    @@~~ Initiate MOD loop pointer R(23) from final R(20) value

064 LBL 04
    RCL 20 1 - INT CHS STO 23          @ R(23) is now MOD loop counter

071 Clx                                @ Clear carry
```

```
    @@~~ MOD computation loop


072 LBL 05
    RCL 22 * RCL IND 23 PRX              @ Print all digits of F^M
    + RCL Y MOD                          @ add carry , compute MOD
080 ISG 23 GTO 05                        @ loop for each storage register (in reverse order)


    @@~~ end of MOD loop


082 "MOD " ARCL Y "~ = " PRA             @ Print/display   "MOD M ="
086 PRX                                  @ Print/display result
087 FIX 3
088 STOP                                 @ End Of Program, wait for a new module entry
089 GTO 04
```

Registers :

```
R(0) - R(19)   Full Digits Storage Area of F^P
R(20)          Storage Pointer  I.nnn
R(21)          Store M value
R(22)          Exponant Extand (EE) , base of digits storage area, all digits storage register less than EE or carry is reported to next storage postion.
R(23)          Power loop counter 0 <- M   / Mod loop Counter  R(20)->0
```

Usage:

```
73 [ENTER^] 55 [ENTER^] 31 [XEQ][ALPHA]POWMOD[ALPHA]
```

Since full digits store in registers, any further modulo computation of the same F^P can be immediately achieve by entering M value and pressing [R/S]. Computation on a new F^P value need reseting computation by keystrokes:

```
 F [ENTER] M [ENTER^] M [ENTER^] [shift][RTN][R/S]
```

Listing obtained on Infrared Printer HP 82240A through IR module.

BASIC program lsting :

```
10:CLEAR :USING :INPUT "MOD(F^P,M): ENTER F,P,M ?";A,B,C
20:LET E=10^(8-INT LOG ABS C),I=1
30:FOR F=1 TO B:FOR G=9 TO 9+D
40:   LET A(G)=H+A*A(G),H=0:IF A(G)>E LET H=INT (A(G)/E),A(G)=A(G)-E*H
50:NEXT G:IF H>0 LET D=D+1,A(9+D)=H,H=0
60:PAUSE F,D:NEXT F
```

```
70:PRINT A;"^";B;"=":FOR G=9+D TO 9 STEP -1:PRINT A(G)
80:   LET H=E*H+A(G),F=INT (H/C),H=H-F*C
90:NEXT G:BEEP 1:PRINT "MOD ";C;" = ";H:END
```

Usage (in DEF or RUN mode):

```
>                           RUN[ENTER]
MOD(F^P,M) F,P,M ?          73[ENTER]
?                           55[ENTER]
?                           31[ENTER]
>
```

Listing obtained on SHARP Printer & Casette Interface CE-122.

Both algorthims use the same tactis as HP28C/S's LPOWMOD code:

Edited: 3 June 2012, 6:19 a.m.

Re: WP 34S accuracy is excellent

Message #16 Posted by ***Valentin Albillo*** *on 1 June 2012, 10:41 p.m.,*
in response to message #1 by Jeff Johnson

Hi, Jeff:

As has already been pointed out in this thread, computing such modular exponentiations is straightforward for any decent programmable calculator or handheld computer using a minimum of memory and time and achieving exact results.

For instance, for the **HP-71B** we have (straight from Wikipedia):

- first, let's program this simple 2-line user-defined function:

```
    10 DEF FNM(B,E,M) @ R=1 @ WHILE E @ IF MOD(E,2) THEN R=MOD(R*B,M)
    20 E=E DIV 2 @ B=MOD(B*B,M) @ END WHILE @ FNM=R @ END DEF
```

- now, let's use it with your particular example and assorted others:

```
    >FNM(73,55,31)

              26

    >FNM(23,391,55)
```

```
          12

>FNM(31,397,55)

          26

>FNM(2,994786,994787)

          563329

>FNM(2,974152,974153)

          46457

>FNM(2,994792,994793)

          1
```

All of them run in negligible time. The last three demonstrate that 994787 and 974153 are both composite numbers and suggest (but not prove) that 994793 *might* be a prime number (which it is).

Regards from V.

[Return to Index | Top of Index]

GTO Go back to the main exhibit hall