HP Forum Archive 18

[Return to Index | Top of Index]

BASIC to RPL translation.

*Message #1 Posted by **wildpig** on 26 Jan 2008, 1:21 a.m.*

I have a quick question just wondering if someone can help me.

I have this simple BASIC program for HP71B to do prime factor.

1 INPUT "N=";N

2 IF NOT MOD(N,2) THEN DISP 2; @ N=N DIV 2 @ GOTO 2 ELSE IF N=1 THEN DISP @ END

3 FOR D=3 TO SQR(N) STEP 2 @ IF NOT MOD(N,D) THEN DISP D; @ N=N DIV D @ GOTO 3

4 NEXT D @ DISP N

Just wondering how this would be translated to RPL. Should be pretty small RPL prog

Incidentally this program suggested by Valentin Albillo is extremely fast on the HP 71B at prime factoring. I have not tried the LEX programming but hard to believe anything can be much faster.

Re: BASIC to RPL translation.

*Message #2 Posted by **Raymond Del Tondo** on 26 Jan 2008, 4:15 a.m.,*
in response to message #1 by wildpig

It _will_ be a small program in RPL. Actually it's that easy you should find out yourself, given you read the manual of your RPL machine;-)

Hints: One of the features of the HP-71B BASIC for saving RAM is an obstacle regarding readability at the same time.
The HP-71B BASIC even allows labels in the middle of a program line. Nice from a memory saving point of view, but bad listing style IMHO.

First thing you should do is trying to recognize logical blocks of code, by separating the BASIC statements to individual lines where possible and meaningful.

Did you notice the 'recursion' in line 3 ?

HTH

Raymond

Edited: 26 Jan 2008, 4:16 a.m.

Re: BASIC to RPL translation.

*Message #3 Posted by **Thomas Klemm** on 29 Jan 2008, 3:31 a.m.,*
in response to message #2 by Raymond Del Tondo

> Quote:
>
> Did you notice the 'recursion' in line 3 ?

Just out of curiosity: is it really recursion or just something
similar to 'continue' in C/C++/Java or 'next' in Perl?
I mean: are the for-loops stacked?

Re: BASIC to RPL translation.

*Message #4 Posted by **Raymond Del Tondo** on 29 Jan 2008, 4:27 a.m.,*
in response to message #3 by Thomas Klemm

The NEXT for the FOR is in line four;-)

The HP-71B BASIC has some very special features, which make it very efficient,
but also encourage the user to produce spaghetti code, as can be seen by the initial example.

Actually the listing implies that the FOR loops are stacked, but it could even be different on the HP-71,
like with the implied END after certain inline IF's. However I won't take a look into the manuals,
as my interest for BASIC is quite limited these days. I like RPL much more;-)

Re: BASIC to RPL translation.

*Message #5 Posted by **Thor Lansen** on 26 Jan 2008, 11:49 a.m.,*
in response to message #1 by wildpig

Check out this website they might have something that works for you:

http://www.hpcalc.org/search.php?query=prime+factor

Re: BASIC to RPL translation.

*Message #6 Posted by **Gerson W. Barbosa** on 27 Jan 2008, 9:27 p.m.,*
in response to message #1 by wildpig

> Quote:
>
> Just wondering how this would be translated to RPL. Should be pretty small RPL prog

Not exactly small, I'd say. It could be slightly smaller if local variables were used instead of the stack. But this approach is faster, especially on real Saturn processors. The translated RPL program runs on the HP-50G. On the HP-28/48 just replace **IQUOT** with **/ IP**.

```
123456789 -> {3 3 3607 3803}  180.1 seconds on the HP-50G (approximate mode)
                               71.7 seconds on the HP-28S!(35.4 seconds in double-speed)
```

Apparently the HP-50G doesn't benefit from stack-only programs, because of the emulated Saturn processor. Anyway, on the HP-50G I prefer FACTOR in exact mode: less than one second for this example :-)

Gerson.

```
%%HP: T(3)A(R)F(.);
\<< { } SWAP DUP DUP 1 \=/
  IF
  THEN
    WHILE 2 MOD NOT
    REPEAT 2 SWAP 2 IQUOT DUP 4 ROLLD 4 ROLLD + ROT ROT
    END DUP 1 \=/
    IF
    THEN 3
      DO
        WHILE DUP2 MOD NOT
        REPEAT SWAP OVER IQUOT OVER 4 ROLL SWAP + ROT ROT SWAP
```

```
      END
    UNTIL 2 + OVER OVER SWAP \v/ >
    END DROP DUP 1 \=/
    \<< +
    \>>
    \<< DROP
    \>> IFTE
  ELSE DROP
  END
 ELSE DROP +
 END
\>>
```

Re: BASIC to RPL translation.

*Message #7 Posted by **Thomas Klemm** on 29 Jan 2008, 3:17 a.m.,*
in response to message #6 by Gerson W. Barbosa

While probably not faster this version is a little shorter
and maybe easier to understand:

```
%%HP: T(3)A(D)F(.);
\<< { } SWAP DUP 1 \=/
  IF
  THEN
    WHILE DUP 2 MOD NOT
    REPEAT 2 / SWAP 2 + SWAP
    END DUP 1 \=/
    IF
    THEN 3
      DO
        WHILE DUP2 MOD NOT
        REPEAT SWAP OVER / SWAP ROT OVER + ROT ROT
        END
      UNTIL 2 + DUP2 SWAP \v/ >
      END DROP DUP 1 \=/
      \<< +
      \>>
      \<< DROP
      \>> IFTE
    ELSE DROP
    END
  ELSE +
  END
\>>
```

However I think your program is not a pure translation of the BASIC
program but a slight improvement since the counter (D) isn't reset
to 3 in the case a factor was found. I'm not familiar with BASIC but
the following might do this:

```
1 INPUT "N=";N
2 IF NOT MOD(N,2) THEN DISP 2; @ N=N DIV 2 @ GOTO 2 ELSE IF N=1 THEN DISP @ END @ S=3
3 FOR D=S TO SQR(N) STEP 2 @ IF NOT MOD(N,D) THEN DISP D; @ N=N DIV D @ S=D + 2 @ GOTO 3
4 NEXT D @ DISP N
```

Calculating the root of N in each step is probably expensive and
could be avoided using a FOR loop. However you need a means of
breaking out in case a factor is found. Are there other ways than
using DOERR?
With an alternating step-size 2, 4, 2, 4, ... I expect the program to

run faster by a factor of ~1.5. But then the loop has to be started with 5 instead of 3.

> Quote:
>
> on the HP-50G I prefer FACTOR in exact mode

Does anybody know which algorithm is used in this program?

Re: BASIC to RPL translation.

*Message #8 Posted by **Gerson W. Barbosa** on 29 Jan 2008, 8:17 a.m.,*
in response to message #7 by Thomas Klemm

> Quote:
>
> While probably not faster this version is a little shorter and maybe easier to understand

Actually, your version is slightly faster: 36.0 seconds to factor 123456789 on the HP-48GX (mine took 38.0 seconds).

> Quote:
>
> However I think your program is not a pure translation of the BASIC program but a slight improvement since the counter (D) isn't reset to 3 in the case a factor was found.

You're right! Rather than translating the HP-71B program directly to RPL I translated it to Turbo Pascal 3. When I got a working Pascal program I translated it to RPL. Translations of translations of human languages generally yield degraded versions of the originals. It this has not happened in this example, so much the better :-)

I think the following is easier to read than any of both RPL versions:

```
----------------------------------------


Program PrimeFac;
var n, d: integer;
begin
  ClrScr;
  Read(n);
  ClrScr;
  if n<>1 then
    begin
      while n Mod 2 = 0 do
        begin
          n:=n div 2;
          Write('2':5)
        end;
      if n<>1 then
        begin
          d:=3;
          repeat
            while n Mod d = 0 do
              begin
                n:= n div d;
                Write(d:5)
              end;
            d:=d+2
          until d>Sqrt(n);
          if n<>1 then
            Write(n:5)
        end
    end
    else
```

```
      Write(n:5);
end.

---------------------------------------

32700

    2    2    3    5    5  109

---------------------------------------
>
```

Re: BASIC to RPL translation.

*Message #9 Posted by **Gerson W. Barbosa** on 30 Jan 2008, 9:10 a.m.,*
in response to message #8 by Gerson W. Barbosa

> Quote:
>
> Actually, your version is slightly faster: 36.0 seconds to factor 123456789 on the HP-48GX (mine took 38.0 seconds).

Your version is even faster on the HP-49G: 34.7 seconds (measured with the built-in timer). The second version of Valentin's BASIC program below runs this example in just 37.9 seconds! This is really impressive, considering the Saturn processor runs at 650 kHz on the HP-71B, while it runs at 4 MHz on the HP-49G.

Re: BASIC to RPL translation.

*Message #10 Posted by **Valentin Albillo** on 30 Jan 2008, 11:22 a.m.,*
in response to message #9 by Gerson W. Barbosa

Hi, Gerson:

Gerson posted:

> *"The second version of Valentin's BASIC program below runs this example in just 37.9 seconds! This is really impressive, considering the Saturn processor runs at 650 kHz on the HP-71B, while it runs at 4 MHz on the HP-49G."*

Thanks for your appreciative comment. Actually, I wrote the original program on the fly and sent it to the OP at his request in something like a couple of minutes.

A really optimized factoring program would be a much more complicated affair and would entail a number of different algorithms, for instance:

1. First of all, prime inputs must be detected beforehand so as to avoid a lengthy factoring process of something that can't be factored. To that effect, a prime-detector algorithm must be run, and very good and fast probabilistic ones are available. It's also useful to detect if the input is an exact power of a prime.

2. Secondly, now that we know the input to be composite, a fast trial division should be run for a few thousand primes, dividing only by prime factors taken from a pre-stored table. This will take care of small factors and with a little luck will reduce the size of the non-factored part should it be unable to factor it completely.

3. Finally, we are left with a composite number with no small factors. Now it's time to use heavy artillery, be it some Fermat- or congruence-based methods for inputs of reasonable size, or else elliptic curves or worse for really large inputs. This will split the composite part in some factors, not necessarily prime, which must be tested to see if they are prime or else be subject to further splitting, in a recursive fashion.

For instance, this factorization I've just performed demonstrates all those methods being used at once:

```
77777777777777777777777777777777777777777777777771  =  89 * 181 * 3413 * 88085341 * 12586899513131 * 1275934133688965411653
```

As you can see, the original input was determined to not be a prime or a power of a prime, then three small factors were duly detected and removed, then the remaining composite part was split into two large factors, one of them prime and the other still composite, which was finally split into two factors both of which were proved to be prime, thus finally ending the process.

Anyway, for the inputs discussed here (up to 12-digit numbers), all of this is mostly academic, as even my humble HP-71B program will be able to either completely factorize or report as a prime any 12-digit input in 2-3 hours at most. When running under Emu71, it takes a mere seconds.

Wish my 71 would be able to do this factorization :-) :

```
 71717171717171717171717171717171717171717171717171717171717171717171717171717171777777777777777777771

=

101 * 109 * 21525175387 * 218301576858349 * 13731482973783137 * 23326138687706820109 * 432824080117318843825281371694451836916 1
```

Best regards from V.

Edited: 30 Jan 2008, 11:41 a.m.

Re: BASIC to RPL translation.

*Message #11 Posted by **Gerson W. Barbosa** on 30 Jan 2008, 8:17 p.m.,*
in response to message #10 by Valentin Albillo

Hello Valentin,

> Quote:
>
> Actually, I wrote the original program on the fly and sent it to the OP at his request in something like a couple of minutes.

I don't doubt it! While I am always impressed at your programming and math skills, I am worried about the low performance of the RPL program on the 49G, when compared to yours on the HP-71B, whose clock is 6 times slower. I don't believe the HP-71B BASIC is six times more efficient than RPL. More likely the translated programs are not so faithful as I thought.

Thanks for your explanation on various factoring methods. Perhaps some of those, if not all, have been used on the built-in factorizer on the HP-49/50G.

> Quote:
>
> Wish my 71 would be able to do this factorization :-) :
>
> ```
>  71717171717171717171717171717171717171717171717171717171717171717171717171717171777777777777777777771
>
> =
>
> 101 * 109 * 21525175387 * 218301576858349 * 13731482973783137 * 23326138687706820109 * 432824080117318843825281371694451836916 1
> ```

My HP-49G does it under four minutes! :-)

Best regards,

Gerson.

Re: BASIC to RPL translation.

*Message #12 Posted by **Egan Ford** on 30 Jan 2008, 11:50 p.m.,*
in response to message #11 by Gerson W. Barbosa

> Quote:
>
> My HP-49G does it under four minutes! :-)

Physical or emulated? My 50g emulator takes 106 seconds to factor that. If a physical 49G can factor that in under four minutes, I'd like to see the algorithm.

Re: BASIC to RPL translation.

*Message #13 Posted by **Egan Ford** on 31 Jan 2008, 2:17 a.m.,*
in response to message #11 by Gerson W. Barbosa

> Quote:
>
> My HP-49G does it under four minutes! :-)

Update. The 49G/50G FACTOR only factors out the 101 and the 109 and then times out returning two primes and a composite.

Re: BASIC to RPL translation.

*Message #14 Posted by **Gerson W. Barbosa** on 31 Jan 2008, 6:52 a.m.,*
in response to message #13 by Egan Ford

> Quote:
>
> The 49G/50G FACTOR only factors out the 101 and the 109 and then times out returning two primes and a composite.

Oops! You are right! I can't believe I made this mistake. I looked the result at a glance and may have seen what I wanted to...

When factoring the larger factor alone, the calculator quits after some minutes. Conclusion: FACTOR on the 49/50G is not reliable.

Gerson.

Re: BASIC to RPL translation.

*Message #15 Posted by **Rodger Rosenbaum** on 2 Feb 2008, 8:01 a.m.,*
in response to message #14 by Gerson W. Barbosa

> Quote:
>
> Conclusion: FACTOR on the 49/50G is not reliable.
>
> Gerson.

It also fails to properly factor

```
77777777777777777777777777777777777777777777777771
```

It thinks 8 is a factor and also leaves a large composite unfactored.

Re: BASIC to RPL translation.

*Message #16 Posted by **Egan Ford** on 2 Feb 2008, 12:52 p.m.,*
in response to message #15 by Rodger Rosenbaum

I just factored it in 4 seconds on my 50g with the help of HPGCC:

```
89 181 3413 88085341 12586899513131 1275934133688965411653
```

Edited: 2 Feb 2008, 12:53 p.m.

Re: BASIC to RPL translation.

*Message #17 Posted by **Rodger Rosenbaum** on 2 Feb 2008, 1:45 p.m.,*
in response to message #16 by Egan Ford

Both my HP49G+ and my HP50G factor it as:

8 * 89 * 181 * 3413 * 14146553970286654836388606023489912303551 63

That last big number is actually the product:

88085341 * 12586899513131 * 127593413368896541 1653

but the calculator can't factor it. It goes for 85 seconds before it returns:

```
{ 14146553970286654836388606023489912303551 63 1 }
```

Is it actually unable to factor it, or did it just time out?

Is there some flag setting I'm unaware of that controls this?

Re: BASIC to RPL translation.

*Message #18 Posted by **Egan Ford** on 2 Feb 2008, 2:39 p.m.,*
in response to message #17 by Rodger Rosenbaum

> Quote:
>
> Both my HP49G+ and my HP50G factor it as:
>
> 8 * 89 * 181 * 3413 * 14146553970286654836388606023489912303551 63
>
> That last big number is actually the product:
>
> 88085341 * 12586899513131 * 127593413368896541 1653

Oddly your 50g is incorrect in returning an 8. 8 factors into 2*2*2, and nothing 2x can end in 1. My 50g returns: 89 * 181 * 3413 * 14146553970286654836388606023489912303551 63. What ROM version are you running?

> Quote:
>
> but the calculator can't factor it. It goes for 85 seconds before it returns:
>
> ```
> { 14146553970286654836388606023489912303551 63 1 }
> ```
>
> Is it actually unable to factor it, or did it just time out?
>
> Is there some flag setting I'm unaware of that controls this?

My guess is that after 20-25 seconds of trial division the internal FACTOR switches to Brent-Pollard and gives up after 60 seconds for each composite left over from the previous division.

The FACTOR that I am using is a C program cross compiled for the 50g and ran at 2.5x normal speed.

Re: BASIC to RPL translation.

*Message #19 Posted by **Valentin Albillo** on 31 Jan 2008, 10:22 a.m.,*
in response to message #13 by Egan Ford

Hi, Egan:

Egan posted:

> *"My 50g emulator takes 106 seconds to factor that. [...] Update. The 49G/50G FACTOR only factors out the 101 and the 109 and then times out returning two primes and a composite."*

It's still amazing that they can discover some small factors of such large integer input. I'm curious as to *what's the largest small factor they can detect* before giving up.

For instance, giving these factorizations (which I've concocted to feature the model numbers of my beloved 71 and your beloved 49/50 as well as the current year 2008), can it find all the small factors ? If yes, what's the timing ? If no, what small factors does it find and what's the timing ? Do both the 49G and 50G find the same number of small factors ?

$$71^{71} - 50^{49} = 3 * 7 * 13 * 587 * 787 * 1061 * 3238349 * 516126363001 * ...$$

$$71^{71} + 2008 = 3 * 3 * 7 * 11 * 73 * 97 * 1213931 * 68637133 * 131118479 * ...$$

Also I take it that this other very similar factorization would fail to result in any factors returned, as there are no small factors, right ? :

$$71^{71} + 50^{49} = 1038268631 * 1607804243009567 * ...$$

Best regards from V.

Edited: 31 Jan 2008, 10:34 a.m.

Re: BASIC to RPL translation.

*Message #20 Posted by **Egan Ford** on 31 Jan 2008, 11:20 a.m.,*
in response to message #19 by Valentin Albillo

> Quote:
>
> It's still amazing that they can discover some small factors of such large integer input. I'm curious as to *what's the largest small factor they can detect* before giving up.

I found this on comp.sys.hp48:

> Quote:
>
> Addr: 0CF006 Name: ^BFactor
>
> Factors long integer. Brent-Pollard, with the assumption that trial division has been done already. When a small factor is found SFactor is called to get full short factorization. Since the factorization can potentially take a very long time, an execution test is used to abort factoring very long integers (limit is 60s for each composite). The factors are sorted at exit.

It appears that time is the upper limit, not a number, given that Brent-Pollard is used, I'm sure somewhere in the code is an upper numeric limit.

I have a C program that can factor that 7171... number by using in order Trial, Brent, William(p+1), Pollard(p-1), Lenstra (elliptic curve), and finally quadratic sieve.

```
Trial:      101
            109
Pollard:    21525175387
Lenstra:    13731482973783137
            218301576858349
Quad Sieve: 432824080117318843825281371694451836916 1
            23326138687706820109
```

This program factors 7171... in 3.5 minutes, but only after quadratic sieve was used to get the larger 2 factors (3 minutes).

I thought about compiling this C code for the 50g, but the quadratic sieve is too memory intensive. However, I think I could produce the first 5 factors relatively quickly in C with the 50g clocked at 192MHz.

Edited: 31 Jan 2008, 11:56 a.m.

Re: BASIC to RPL translation.

*Message #21 Posted by **Valentin Albillo** on 31 Jan 2008, 11:55 a.m.,*
in response to message #20 by Egan Ford

Hi again, Egan:

> Very interesting info, thanks a lot for it. You mention there's a limit of 60 seconds per composite. Are these calculators fast enought to be able to reach up to the **3238349** factor or at least the **1213931** factor, or are they still not small enough ?
>
> In any case, I guess that the **20593** factor of $\mathbf{71^{71} + 2}$ or the **46549** factor of $\mathbf{71^{71} - 2}$ are easily within reach, right ?

Best regards from V.

Re: BASIC to RPL translation.

*Message #22 Posted by **Egan Ford** on 31 Jan 2008, 1:03 p.m.,*
in response to message #21 by Valentin Albillo

> Quote:
>
> Very interesting info, thanks a lot for it. You mention there's a limit of 60 seconds per composite. Are these calculators fast enought to be able to reach up to the **3238349** factor or at least the **1213931** factor, or are they still not small enough ?

See next post. The internal 50g FACTOR didn't make it. Give me a weekend or two and I think I can get a C program on the 50g to at least get 3238349 and perhaps 516126363001 from the first problem.

> Quote:
>
> In any case, I guess that the **20593** factor of $\mathbf{71^{71} + 2}$ or the **46549** factor of $\mathbf{71^{71} - 2}$ are easily within reach, right

I only get 13 for the first one, and 3 for the 2nd.

Re: BASIC to RPL translation.

*Message #23 Posted by **Valentin Albillo** on 1 Feb 2008, 8:48 a.m.,*
in response to message #22 by Egan Ford

Hi, Egan:

Egan posted:

> *"Give me a weekend or two and I think I can get a C program on the 50g to at least get 3238349 and perhaps 516126363001 from the first problem."*
>
> Excellent ! I'll take your word for it, and am eager to see what you can do with your extraordinary wits and two (or three) weekends to spend on it.
>
> The following example which I've just factored precisely for the occasion, might prove useful to test your program, as it has lots of small factors, intermediate factors, and large factors, all at once:
>
> $69^{96} - 1$

```
=

 338464177770582700867528197630612812643411481076411271352964279330592074344897475465841485100167185056251698842680702037916169590222050610365414966035295777475786702161805297280


=


2 * 2 * 2 * 2 * 2 * 2 * 2 * 2 * 5 * 7
  * 13 * 17 * 19 * 19 * 97
  * 113 * 193 * 373
  * 1697 * 2381 * 3329 * 4831
  * 60757 * 88129
  * 100297 * 339841 * 975553
  * 16274113 * 43001089 * 65655137 * 711105793
  * 10468797169 * 19009122241 * 1549499138353
  * 256899187214321 * 513798351761521
  * 780030795244524708969982O9
```

See just how many small (or intermediate, or large) factors your subsequent versions and improvements can find in the allotted time and/or limits.

Best regards from V.

[OT] A new thread to save :)

*Message #24 Posted by **Giancarlo (Italy)** on 1 Feb 2008, 9:51 a.m.,*
in response to message #23 by Valentin Albillo

Hi.
This is another of those threads that *do* deserve to be saved and stored for future study & reference!
According to my personal standards ;) this is the highest tribute to the quality of the discussion.
Thanks to all the knowledgeable contributors.
Best regards & enjoy your weekend.
Giancarlo

Edited: 4 Feb 2008, 5:16 a.m. after one or more responses were posted

Thanks, Giancarlo ! [NT]

*Message #25 Posted by **Valentin Albillo** on 4 Feb 2008, 4:56 a.m.,*
in response to message #24 by Giancarlo (Italy)

Best regards from V.

Re: BASIC to RPL translation.

*Message #26 Posted by **Marcus von Cube, Germany** on 1 Feb 2008, 3:12 p.m.,*
in response to message #23 by Valentin Albillo

Hi Valentin,

> Quote:
>
> $69^{96} - 1$

That's a real bummer! I've entered factor(69^{96}-1) into my TI nSpire and the poor thing has been working on the problem for more than two hours by now. Since there are no intermediate results to watch, I've no idea how long it will take.

I've recently replaced the batteries. Will they last long enough?

Marcus

Edit: The poor beast is still computing (almost 5 hours later)...

I'll go to bed now and will check tomorrow. Meanwhile the nspire-cas application is trying to solve the same problem, running under Parallels Workstation on my iMac. Lets see...

Edit again: the solution is here, the iMac has arrived at it in just a few minutes:

2^7*5*7*13*17*19^2*97*113*193*373*1697*2381*3329*4831*60757*88129 *100297*339841*975553*16274113*43001089*65655137*711105793 *10468797169*19009122241*1549499138353*256899187214321 *513798351761521*78003079524452470896998209

I haven't checked it for correctness, yet.

Edit again: The handheld has finally arrived. The results are the same as above and they seem to be correct. I don't know exactly how long it took, but it must have been more than 7 hours.

Edited: 2 Feb 2008, 5:38 a.m. after one or more responses were posted

Re: BASIC to RPL translation.

*Message #27 Posted by **Egan Ford** on 2 Feb 2008, 4:23 a.m.,*
in response to message #26 by Marcus von Cube, Germany

I get the same answer in 187 seconds (50g, 75 Mhz (normal speed), HPGCC2). I expect to cut this down to 90 seconds or less when running at 192 MHz (HPGCC3). The same program on a PC took 3 seconds. IOW, this was an easy problem.

The rapid factoring of $\mathbf{69^{96} - 1}$ used a combination of Trial Division, Brent-Pollard, William's (p+1), Pollard's (p-1), Lenstra's Elliptic Curve, and finally Multiple Polynomial Quadratic Sieve to factor the last composite (976685567585907036908856l). MPQS is a memory hog if the number exceeds 20 digits long. It would be nice if HPGCC2/3 could use the SD card as swap space.

All the above methods are part of MIRACL. I used HPGCC2/3 to port MIRACL to the 50g.

Factoring Valentin's constant (7171...) has been troublesome. On the 50g I can factor out primes 101 * 109 * 21525175387 * 13731482973783137 * 218301576858349, but run out of memory when trying to factor the composite factor: 10096114520195707345153354910922719092443596945224988025854 9. I need time to tune the other methods.

As for the other problems (BTW, they are very unpleasant), I can factor out the following:

$71^{71} - 50^{49} = 3 * 7 * 13 * 587 * 787 * 1061 * 3238349 * 516126363001 * (103 digits long)$

$71^{71} + 2008 = 3 * 3 * 7 * 11 * 73 * 97 * 1213931 * 68637133 * 131118479 * (103 digits long)$

$71^{71} + 50^{49} = 1038268631 * 1607804243009567 * (108 digits long)$

I factored the above using a 50g/HPGCC2 simulator (HPAPINE), and skipped MPQS because I knew it would fail on the 50g. The reported prime factors are found quickly, then there is a lot of computing with no results other than a large composite factor.

IANS, I may be able to improve on this as I find the time. The MIRACL FACTOR can factor very quickly any number that the 50g FACTOR can factor and many more numbers that the 50g cannot factor (e.g. 1152921515344265237).

Lastly, if any are interested in C programming on the 50g then read my tutorial at http://sense.net/~egan/hpgcc.

Re: BASIC to RPL translation.

*Message #28 Posted by **wildpig** on 2 Feb 2008, 4:24 a.m.,*
in response to message #26 by Marcus von Cube, Germany

That is the correct answer. Took derive 6 on my 1.5 Ghz laptop 2 sec to do ;)

Re: BASIC to RPL translation.

*Message #29 Posted by **Marcus von Cube, Germany** on 2 Feb 2008, 5:45 a.m.,*
in response to message #28 by wildpig

When installing the nSpire PC software it becomes clear that it is written -at least partly- in Java. If that is true for the handheld, too, I can understand the performance penalty. On the other hand, the package comes with a large native library. I assume, that many of the math functions are copied from the older TI CAS machines and put into the lib. I could try on my Voyage 200...

Re: BASIC to RPL translation.

*Message #30 Posted by **Xerxes** on 3 Feb 2008, 4:33 a.m.,*
in response to message #29 by Marcus von Cube, Germany

> Quote:
>
> I could try on my Voyage 200...

Hello Marcus,

I guess it will take some days on the Voyage 200. After 13.5 hours, there is still no solution on a double speed Voyage.

For TI-BASIC programs the Nspire seems to be about 13x faster than the 68k calculators. May be the same factor is valid for the CAS.

Re: BASIC to RPL translation.

*Message #31 Posted by **Egan Ford** on 31 Jan 2008, 11:52 a.m.,*
in response to message #19 by Valentin Albillo

I do not have a 49G, but I believe that it uses the same code as the 50g.

50g results:

> Quote:
>
> $71^{71} - 50^{49} = 3 * 7 * 13 * 587 * 787 * 1061 * 3238349 * 516126363001 * ...$

3 * 7 * 13 * 587 * 787 * 1061 * **2055189604...** in 269 sec

> Quote:
>
> $71^{71} + 2008 = 3 * 3 * 7 * 11 * 73 * 97 * 1213931 * 68637133 * 131118479 * ...$

3 * 3 * 7 * 11 * 73 * 97 * **5604216830...** in 291 sec

> Quote:
>
> $71^{71} + 50^{49} = 1038268631 * 1607804243009567 * ...$

Nada in 279 sec.

If I had to guess, trial division snaps up the smaller numbers quickly, then Brent-Pollard times out after 4 minutes.

Re: BASIC to RPL translation.

*Message #32 Posted by **Gerson W. Barbosa** on 31 Jan 2008, 6:42 p.m.,*
in response to message #13 by Egan Ford

The emulated 49G with ROM Revision 1.18 doesn't give up after a few minutes... but it keeps running for ever!

Re: BASIC to RPL translation.

*Message #33 Posted by **Egan Ford** on 31 Jan 2008, 7:51 p.m.,*
in response to message #32 by Gerson W. Barbosa

1.24 does give up.

Re: BASIC to RPL translation.

*Message #34 Posted by **wildpig** on 31 Jan 2008, 7:02 p.m.,*
in response to message #7 by Thomas Klemm

How about the following algorithm suggested by Joe Horn?

Name Description In Out +------+------------------------+--------+----+----------------+ | NP | Next Prime factor | n | -> | n factor | +------+------------------------+--------+----+----------------+

[#5AD9h, 122.5 w/ name] -- see pg. 627 for character translations.

\<< DUP \w/ \-> s \<< DUP IF 2 MOD THEN 3 WHILE DUP2 MOD OVER s < AND REPEAT 2 + END DUP s IF > THEN DROP DUP END ELSE 2 END \>> \>> @ end of NP

Name Description In Out +------+------------------------+--------+----+--------------------+ | PF | Prime Factors | n | -> | n: { factors of n } | +------+------------------------+--------+----+--------------------+

[#445Bh, 62 w/ name]

\<< DUP { } SWAP DO NP ROT OVER + ROT ROT / DUP UNTIL 1 == END DROP SWAP \->TAG \>> @ end of PF

Btw, very interesting that the built in FACTOR in 50G and 49G doesn't work completely. Guess we still need a program for that. Does HP know about this bug?

Btw, these RPL programs, I can just type them in as they are printed right?

Edited: 31 Jan 2008, 7:04 p.m.

Re: BASIC to RPL translation.

*Message #35 Posted by **Egan Ford** on 31 Jan 2008, 7:56 p.m.,*
in response to message #34 by wildpig

> Quote:
>
> Btw, very interesting that the built in FACTOR in 50G and 49G doesn't work completely. Guess we still need a program for that. Does HP know about this bug?

It's not a bug, the 50g/49g were designed to quit if factoring takes too long. If you are trying to factor a number on your calculator that takes longer than a few minutes then you are using the wrong tool.

BTW, the returned results are factors, but may not always be all prime factors.

Re: BASIC to RPL translation.

*Message #36 Posted by **wildpig** on 3 Feb 2008, 1:19 p.m.,*
in response to message #34 by wildpig

Using the above routine by J Horn,

123456789 = 3^2 * 3607 * 3803

in 26 sec by my stopwatch on HP 48G.

and 39 sec on HP 71B using the routine by V.

Hmm, if HP71B saturn is really only 600KHz and HP 48G is 3-4 MHz, something is really inefficient about the 48G? Is it because i am using a userRPL prog? bad routine?

Re: BASIC to RPL translation.

*Message #37 Posted by **wildpig** on 4 Feb 2008, 9:22 a.m.,*
in response to message #36 by wildpig

i take it 26 sec to factor 123456789 is about the fastest (at least by userRPL) that you can do on hp48g? anyone else better?

doesn;t scale well to the clock speed compared to hp 71b

Re: BASIC to RPL translation.

*Message #38 Posted by **Gerson W. Barbosa** on 3 Feb 2008, 7:26 p.m.,*
in response to message #7 by Thomas Klemm

Definitely not a tiny program anymore, but now it returns an algebraic instead of a list of factors. Of course, this can still be optimized.

```
 123456789 -> '3^2*3607*3803'    36.3 seconds (HP-48GX)
5961027250 -> '2*5^3*3011*7919'  30.7 seconds


%%HP: T(3)A(R)F(,);
\<< { } SWAP DUP 1 >
  IF
  THEN
    WHILE DUP 2 MOD NOT
    REPEAT 2 / SWAP 2 + SWAP
    END DUP 1 \=/
    IF
    THEN 3
      DO
        WHILE DUP2 MOD NOT
        REPEAT SWAP OVER / SWAP ROT OVER + ROT ROT
        END
      UNTIL 2 + DUP2 SWAP \v/ >
      END DROP DUP 1 \=/
      \<< +
      \>>
      \<< DROP
      \>> IFTE
    ELSE DROP
    END
  ELSE +
  END DUP SIZE 1 \=/
  IF
  THEN DUP HEAD 1 \->LIST 1 ROT DUP SIZE 1 - 1 SWAP
    FOR i DUP i GETI ROT ROT GET OVER ==
      IF
      THEN DROP SWAP 1 + SWAP
      ELSE DROP ROT ROT 1 \->LIST + SWAP DUP i 1 + GET 1 \->LIST ROT SWAP + 1 ROT
      END
    NEXT DROP 1 \->LIST + "'" SWAP DUP SIZE 1 - 1 SWAP
    FOR i DUP i GETI ROT ROT GET SWAP \->STR SWAP DUP 1 \=/
      IF
      THEN \->STR "^" SWAP + +
      ELSE DROP
```

```
    END "*" + ROT SWAP + SWAP 2
  STEP DROP DUP SIZE "'" REPL STR\->
ELSE LIST\-> DROP
END
\>>
```

Re: BASIC to RPL translation.

*Message #39 Posted by **Gerson W. Barbosa** on 4 Feb 2008, 6:27 a.m.,*
in response to message #38 by Gerson W. Barbosa

Changes so that it works on the HP-28S:

```
1 GET
```

instead of

```
HEAD
```

and

```
1 - 1 SWAP SUB "'" +
```

instead of

```
"'" REPL
```

I wonder who'd key all this into the 28S though :-)

Gerson.

Re: BASIC to RPL translation.

*Message #40 Posted by **wildpig** on 4 Feb 2008, 9:24 a.m.,*
in response to message #38 by Gerson W. Barbosa

G, what you think about the two short routines above from J Horn?

Edited: 4 Feb 2008, 9:25 a.m.

Re: BASIC to RPL translation.

*Message #41 Posted by **Gerson W. Barbosa** on 4 Feb 2008, 11:16 a.m.,*
in response to message #40 by wildpig

W, I cannot compete with JH. Anyway, this is not a programming contest :-)

The HP-48 directory below contains Joe Horn's program, in case someone wants to give it a try. RT has been added to check running time with the built-in clock. Perhaps his program would run a litte bit faster if no local variable were used. You can try to use only the stack but the program will be slightly longer.

```
 123456789 RT ->  123456789: { 3 3 3607 3803 }    25.82 seconds (HP-48GX)
5961027250 RT -> 5961027250: { 2 5 5 5 3011 7919} 21.95 seconds


%%HP: T(3)A(D)F(,);
DIR
  pf
    \<< DUP { } SWAP
      DO NP ROT
OVER + ROT ROT /
DUP
```

```
      UNTIL 1 ==
      END DROP SWAP
\->TAG
    \>>
  NP
    \<< DUP \v/ \-> s
      \<< DUP
        IF 2 MOD
        THEN 3
          WHILE
DUP2 MOD OVER s <
AND
          REPEAT 2
+
          END DUP s
          IF >
          THEN DROP
DUP
          END
        ELSE 2
        END
      \>>
    \>>
  RT
    \<< TIME SWAP pf
TIME ROT HMS- HMS\->
3600 * SWAP
    \>>
END
```

Re: BASIC to RPL translation.

*Message #42 Posted by **wildpig** on 10 Feb 2008, 5:20 p.m.,*
in response to message #41 by Gerson W. Barbosa

What i find is that usually the shorter the program, obviously the faster it usually runs. The Hp71b basic program by V was short and simple. Which is why it runs fast. The RPL routine by Joe Horn is also fairly short. Which is why it also runs fast.

I finally got the serial cable to transfer some programs for prime factor to the calculator. none so far was able to factor 123456789 in less than 26 sec on my hp 48g. Just a few more to try...

Re: BASIC to RPL translation.

*Message #43 Posted by **Egan Ford** on 10 Feb 2008, 5:34 p.m.,*
in response to message #42 by wildpig

I have a C version for the 48 that can factor 123456789 in about 3.5 sec.

Re: BASIC to RPL translation.

*Message #44 Posted by **wildpig** on 10 Feb 2008, 9:21 p.m.,*
in response to message #42 by wildpig

So i found this FACTOR.zip. It has several neat routines including using Pollard rho and Selfridge.

Extremely fast. it factors 123456789 in less than 1 sec

It even factor 3^37+1 in like less than 5 sec.

to enter 3^37+1 you would type 63 STWS #3 #37 #1 NEG POWMOD 1 + while you are inside the FACTOR dir. I guess this is a way to enter really large number in machine code (forth?). Does anyone know how this works so that i can enter something like 123456789123456789123456789?

Thanks

Re: BASIC to RPL translation.

*Message #45 Posted by **wildpig** on 12 Feb 2008, 3:16 a.m.,*
in response to message #44 by wildpig

> Quote:
>
> 63 STWS #3 #37 #1 NEG POWMOD 1 +

I found that 63 STWS is to specify binary word size. So max number you can enter into hp48g is 2^63 or 9223372036854775808. To enter this, you would type # 9223372036854775808d and put that on the stack.

I am sure this is nothing new to most people but I found this fairly incredible for a calculator .

Re: BASIC to RPL translation.

*Message #46 Posted by **Don Shepherd** on 12 Feb 2008, 9:07 a.m.,*
in response to message #45 by wildpig

Pig, CAS calculators that I am aware of (TI-89 Titanium, NSpire, HP-49g+ and probably HP-50g) can display any integer, up to available memory. For instance, the NSpire can display the actual value of 4^256 x 256!, which is some LARGE value!

Re: BASIC to RPL translation.

*Message #47 Posted by **wildpig** on 4 Feb 2008, 9:36 a.m.,*
in response to message #38 by Gerson W. Barbosa

pardon my ignorance, but i know \v/ stands for sqrt. what does \w/ stand for? thanks

Re: BASIC to RPL translation.

*Message #48 Posted by **Patrice** on 29 Jan 2008, 1:40 p.m.,*
in response to message #1 by wildpig

It seems that the program is not optimized.

Just splitting line 3 in 2 will greatly improve the computing time.

```
1 INPUT "N=";N
2 IF NOT MOD(N,2) THEN DISP 2; @ N=N DIV 2 @ GOTO 2 ELSE IF N=1 THEN DISP @ END
3 FOR D=3 TO SQR(N) STEP 2
4 IF NOT MOD(N,D) THEN DISP D; @ N=N DIV D @ GOTO 4
5 NEXT D @ DISP N
```

and logically speaking it is cleaner.

Moving the SQR(N) out of the loop will improve the program too.

I think that this little mod is closer to the RPL translations.

Re: BASIC to RPL translation.

*Message #49 Posted by **Thomas Klemm** on 29 Jan 2008, 2:02 p.m.,*
in response to message #48 by Patrice

Would you mind testing 948892238557 = 977 * 983 * 991 * 997 ?
I expect your program to take ~1000 times longer that the original.
It's because the loop is not left when a small factor is found.

Re: BASIC to RPL translation.

*Message #50 Posted by **Patrice** on 29 Jan 2008, 6:25 p.m.,*
in response to message #49 by Thomas Klemm

You should reread carefully the 2 programs.

My little mod is 3.5 times faster than the original one.

Just check it.

Re: BASIC to RPL translation.

*Message #51 Posted by **Valentin Albillo** on 29 Jan 2008, 7:34 p.m.,*
in response to message #50 by Patrice

You should reread carefully what Thomas says.

Your little mod breaks the original program.

Just check it.

Re: BASIC to RPL translation.

*Message #52 Posted by **Patrice** on 29 Jan 2008, 9:18 p.m.,*
in response to message #51 by Valentin Albillo

Flaming mode = on

Valentin said: You should reread carefully what Thomas says.

You should reread carefully what Thomas and myself said. He said my program should be slower and he gives a raison which precisely why it is faster.

Valentin said: Your little mod breaks the original program.

Yes, it is just the main principle of optimization: removing needless work.

Why opening a new loop on stack for each factor found when there is just no need to do it ?

Valentin said: Just check it.

You should just check it carefully yourself.

By the way the original program (and your own mod too) can not be translated in RPL, but with my little mod, it can be. And the RPL programs in this treat are reflecting the same structure.

flaming mode = off

Re: BASIC to RPL translation.

*Message #53 Posted by **Steve Perkins** on 30 Jan 2008, 1:30 p.m.,*
in response to message #52 by Patrice

I actually programmed both versions in EMU71 (in slow mode). The one by Patrice did run slower for the provided example. The funny thing is, it appears to finish, but is actually still running the program. I couldn't figure out why at first, since it looks correct, especially to a 'C' programmer like myself.

I finally realized that the HP-71B optimizes FOR loops by calculating the end test (SQR(N)) once and saving the value for all the later tests. When the value of N changes inside the loop, it doesn't recalculate the end test. That's why you have to jump outside the loop and re-initialize it.

Patrice's attempted optimization might work in another dialect of basic. It's also similar to the improvement posted later, which runs faster as advertised.

I hope my reasoning is correct, and I hope this helps clear up the different perspectives.

Re: BASIC to RPL translation.

*Message #54 Posted by **Patrice** on 30 Jan 2008, 2:32 p.m.,*
in response to message #53 by Steve Perkins

Nasty trick. :)

I checked the program on my HP85 and it run perfectly on it (with little mods because of differences in the basics).

I got catch because my HP71 did not let me store a value in N. When I type "N=1", it sayes "ERR:Data Type" and I have not been able to put my hand on the manuals until now.

Re: BASIC to RPL translation.

*Message #55 Posted by **Valentin Albillo** on 30 Jan 2008, 2:45 p.m.,*
in response to message #54 by Patrice

Hi, Patrice:

Patrice posted:

> *"Nasty trick. :)"*
>
> Thomas and I both told you that your "optimization" actually was breaking the original program. But you wouldn't listen.
>
> You should really understand what you're doing before "optimizing" anything, most specially if you didn't write it in the first place. And after you do your thing, you must check extensively to see that everything works as before and you haven't broken anything.
>
> *"I got catch because my HP71 did not let me store a value in N. When I type "N=1", it sayes "ERR:Data Type"*
>
> Execute
>
> ```
> DESTROY ALL  [ENDLINE]
> ```
>
> from the keyboard (or include "DESTROY ALL" as a statement at the very beginning of the program) and everything will be Ok with assignments to that pesky N variable.
>
> And next time, be a little less arrogant and pay attention to what well-meaning, knowledgeable people are telling you.

Best regards from V.

Re: BASIC to RPL translation.

*Message #56 Posted by **Patrice** on 30 Jan 2008, 9:00 p.m.,*
in response to message #55 by Valentin Albillo

Valentin said: DESTROY ALL
It did the trick. Thanks

Valentin said: And next time, be a little less arrogant and pay attention to what well-meaning, knowledgeable people are telling you.
A clear advice of why I was wrong, like the 2 following would have catch my attention diferently. The beavior been different from about every BASICs I know.
- In HP71B basic, in the FOR-NEXT loop, the end value is computed only once in the FOR statement.
- In HP71B basic, the end value is not recomputed when the NEXT is done.

Re: BASIC to RPL translation.

*Message #57 Posted by **Patrice** on 30 Jan 2008, 1:38 p.m.,*
in response to message #48 by Patrice

My optimization have a little bug which arise when the biggest factor is at least a square.

For example: 363 is giving 3 11 11 1 rather than 3 11 11

Here is the program corrected:

```
1 INPUT "N=";N
2 IF NOT MOD(N,2) THEN DISP 2; @ N=N DIV 2 @ GOTO 2 ELSE IF N=1 THEN DISP @ END
3 FOR D=3 TO SQR(N) STEP 2
4 IF NOT MOD(N,D) THEN DISP D; @ N=N DIV D @ GOTO 4
5 NEXT D @ IF N<>1 THEN DISP N
```

Re: BASIC to RPL translation.

*Message #58 Posted by **Gerson W. Barbosa** on 2 Feb 2008, 4:33 a.m.,*
in response to message #57 by Patrice

Patrice,

This fixes the final 1, but doesn't solve the slowness problem in the example Thomas has suggested, which would require about 200 minutes to run on a physical 71B.

The RPL program below (HP-28/48), which is *exactly* equivalent to your modified program, would take about three hours to factor 948892238557 on the 48GX. The small factors are quickly found but the for-loop keeps running for about 500 thousand times. Of course, if you run the program on a fast computer, this can hardly be noticed. I would suggest you run your BASIC program on emu71. Even at full speed, the difference can be perceived.

Regards,

Gerson.

```
%%HP: T(3)A(D)F(,);
\<< { } SWAP
  WHILE DUP 2 MOD
NOT
  REPEAT 2 / SWAP 2
+ SWAP
  END DUP
  IF 1 \=/
  THEN DUP DUP \v/ 3
SWAP
    FOR d
      WHILE d MOD
NOT
      REPEAT d SWAP
OVER / ROT ROT +
SWAP DUP
      END DUP 2
    STEP 1 \=/
    \<< +
    \>>
    \<< DROP
    \>> IFTE
```

```
  ELSE DROP
  END
\>>
```

Edited: 2 Feb 2008, 4:54 a.m.

Re: BASIC to RPL translation.

*Message #59 Posted by **Patrice** on 2 Feb 2008, 10:17 a.m.,*
in response to message #58 by Gerson W. Barbosa

I posted the program just before learning that the HP71 basic have a very unusual behavior with the FOR-NEXT loop.
I did not know the fact that contrary to every BASIC I know, the end value of a FOR-NEXT loop is computed only once when you enter the loop.
As I was unable to test the program on my HP71, I tested it on an HP85 and it is just fine on it. :)

Re: BASIC to RPL translation.

*Message #60 Posted by **Gerson W. Barbosa** on 2 Feb 2008, 11:30 a.m.,*
in response to message #59 by Patrice

Bonjour, Patrice!

It is not a problem in the BASIC dialect, but a problem in the algorithm. Consider the 71B program below:

```
1 INPUT "N=";N @ C=0
2 IF NOT MOD(N,2) THEN DISP 2; @ N=N DIV 2 @ GOTO 2 ELSE IF N=1 THEN DISP @ END
3 FOR D=3 TO SQR(N) STEP 2
4 IF NOT MOD(N,D) THEN DISP D; @ N=N DIV D @ GOTO 4
5 C=C+1 @ NEXT D @ IF N<>1 THEN DISP N
6 DISP @ DISP C


>RUN
N=948892238557
 977  983  991  997
 487054
```

As you can see, the for-loop is executed too many times. Now consider the equivalent QBASIC program:

```
1 CLS : DEFLNG C-D, N: INPUT "N="; N: C = 0
2 IF (N MOD 2) = 0 THEN PRINT 2; : N = N / 2: GOTO 2 ELSE IF N = 1 THEN PRINT
3 FOR D = 3 TO SQR(N) STEP 2
4 IF (N MOD D) = 0 THEN PRINT D; : N = N / D: GOTO 4
5 C = C + 1: NEXT D: IF N <> 1 THEN PRINT N
6 PRINT : PRINT C


N=? 951747481
 977  983  991
 15424
```

Again, the for-loop is executed more times than it should. This is the same result we get on the 71B:

```
>run
N=951747481
 977  983  991
 15424
```

QBASIC on the PC isn't slow enough we can see there is something wrong. I guess the same occurs on your HP-85.

Amicalement,

Gerson.

P.S.: I just want to share my findings. I made the same mistake yesterday when trying to get a more faithful translation of Valentin's program :-)

Looks like the first version, to which I gave more thought, is closer. However, I prefer Thomas Klemm's improved version, which is slightly faster.

Re: BASIC to RPL translation.

*Message #61 Posted by **Don Shepherd** on 2 Feb 2008, 2:11 p.m.,*
in response to message #59 by Patrice

Patrice, I don't think this behavior is unusual. Most BASIC's I am aware of evaluate the ending value of the loop only once, when you first enter the loop, and you can't change it later within the loop. I know VBA within Excel works this way, and I think that is the common behavior in all BASIC's I know.

Re: BASIC to RPL translation.

*Message #62 Posted by **Rodger Rosenbaum** on 2 Feb 2008, 8:34 p.m.,*
in response to message #61 by Don Shepherd

The HP-71 Owner's Manual, March 1987 edition, beginning on page 268 has a section titled, "Conformance of BASIC Interpreter to ANSI Standards". On the next page, they mention that "ANSI requires that the limit and step be evaluated once upon entering the loop".

Re: BASIC to RPL translation.

*Message #63 Posted by **Valentin Albillo** on 29 Jan 2008, 3:20 p.m.,*
in response to message #1 by wildpig

Hi, wildpig:

wildpig posted:

> *"Incidentally this program [...] is extremely fast on the HP 71B at prime factoring [...] hard to believe anything can be much faster."*

As Thomas Klemm kindly suggested below, an obvious optimization which I missed when concocting this on the fly will make it run noticeably faster for some difficult entries, namely:

1. Add " @ D=3" to the end of line 1
2. Line 3: Change "FOR D=3 TO ..." to "FOR D=D TO ..."

The resulting listing should look like this:

```
1 INPUT "N=";N @ D=3
2 IF NOT MOD(N,2) THEN DISP 2; @ N=N DIV 2 @ GOTO 2 ELSE IF N=1 THEN DISP @ END
3 FOR D=D TO SQR(N) STEP 2 @ IF NOT MOD(N,D) THEN DISP D; @ N=N DIV D @ GOTO 3
4 NEXT D @ DISP N
```

With these changes in place, some difficult cases will run up to 300% faster. Check for instance 948892238557 and 988053892081.

Best regards from V.

Re: BASIC to RPL translation.

*Message #64 Posted by **wildpig** on 30 Jan 2008, 3:45 a.m.,*
in response to message #63 by Valentin Albillo

Not bad. ;) It does speed up considerably. I cant wait to get my hand on an hp 75D and see how much faster this little BASIC program is. Would probably give an HP 48 a real run around with its RPL programs.

Thanks V and everyone

Re: BASIC to RPL translation.

*Message #65 Posted by **Raymond Del Tondo** on 30 Jan 2008, 7:15 a.m.,*
in response to message #64 by wildpig

<OT>
You could also get a picture if you compare the Sudoku solver for the HP-71B (implemented by Valentin),
and my SuDoKu solver for the HP-48 (SDK48). The 71B version can run
up to about a quarter of an hour (or maybe even more) for some complicated puzzles.

My SDK48 solves the hardest ones in about 6 to 10 _seconds_ , the others in less than 5 seconds,
typically less than 1 to 2 seconds.
With some effort I could speed up the beast even more, so that the most complicated puzzles
would be solved in about 3 (three) seconds or less, but the current speed seems to be sufficient IMHO.
At least there is no faster SDK solver for the HP-48;-)

The principal calculation mechanism of the 71B version and SDK48 is similar and freely available,
there were some (many!) discussions about this topic on Sudoku and software related forums and even here,
as well as some program listings (in C) for both brute force and hashed and indexed solvers.

</OT>

[Return to Index | Top of Index]

Go back to the main exhibit hall