HP Forum Archive 10

HP-41 / 42 style RPN challenge

Message #1 Posted by ***Gene*** *on 10 Feb 2003, 11:58 a.m.*

HP-41 / 42 style RPN challenge

Develop a short routine to reverse the order of a number in X.

1234 -> 4321 123456789 -> 987654321

Restrictions:

Begin the routine with LBL 01 Include an "END" in the byte count

Current "champion" routine is 24 bytes.

Timings: 123456789 -> 987654321 in just about 4 seconds.

Other programmable RPN calculators are welcome to do this too, in which case the total # of steps should be used. RPL calculator users are welcome to jump in if they like, but hey...let's not forget RPN!

Can anyone do it in less space and time? :-)

Gene

Not an answer, just a question...

Message #2 Posted by ***Vieira, Luiz C. (Brazil)*** *on 10 Feb 2003, 3:11 p.m.,*
in response to message #1 by Gene

Hi, Gene;

will the number be always a positive integer? Will it have an expoent of ten greather than 9?

If the answer for these two questions are YES and NO repectively, we will have less "factor correcting" steps; agree?

Cheers.

Further specs

Message #3 Posted by ***Gene*** *on 10 Feb 2003, 3:18 p.m.,*
in response to message #2 by Vieira, Luiz C. (Brazil)

Assume it's a positive integer

Assume no scientific notation.

Just a number between 1 and 9,999,999,999

I'm just curious to see what someone else comes up with. Gene

An answer

Message #4 Posted by ***Andrés C. Rodríguez (Argentina)*** *on 10 Feb 2003, 5:04 p.m.,*
in response to message #1 by Gene

This solution takes 21 bytes (as far as I can check), uses only the stack registers, and works OK provided that:

a) The routine is called with the initial integer number in X and a value of 0 in Y.

b) If the number ends with a 0, the leading zero will be suppressed in the answer (i.e.: 120 will be converted into 21, which is the same than 021).

LBL 01

RCL X

10

STO * T

STO / Z

```
MOD

STO + Z

ROLL DOWN

INT

X<>0?

GTO 01

RDN

END
```

If the restriction about calling the routine with Y=0 is not allowed, then the routine will be a little longer, adding three steps as follows:

```
LBL 01

0

X<>Y

LBL 02

RCL X

10

STO * T

STO / Z

MOD

STO + Z

ROLL DOWN
```

```
INT

X<>0?

GTO 02

RDN

END
```

Good job! and my baseline routine

Message #5 Posted by ***Gene*** *on 10 Feb 2003, 5:46 p.m.,*
in response to message #4 by Andrés C. Rodríguez (Argentina)

Well, not MY routine. Had fun with this from the Key Notes Vol 5 No 2 P11 middle column modified to assume a Y register equal to zero. Bytes are 24 with the routine making Y equal zero and 21 without and the user manually making it zero. Can anyone go below 21 bytes?

The point about 510 being reversed as 15 is valid. Good point.

If you don't want to assume Y is zero, insert before the LBL 01 a

LBL 00 0 ENTER

then continue with this.

LBL 01 10 ST * Z / ENTER FRC ST + Z - X NE 0? GTO 01 x<>Y 10 * END

Oops.

Message #6 Posted by ***Gene*** *on 10 Feb 2003, 5:48 p.m.,*
in response to message #5 by Gene

Oops. That initial couple of extra steps should have been LBL 00 0 X<>Y

not

LBL 00 0 ENTER

Won't work the second way.

Oops! and clarifications

Message #7 Posted by ***Andrés C. Rodríguez (Argentina)*** *on 10 Feb 2003, 8:23 p.m.,*
in response to message #6 by Gene

The variation with LBL 01, 0, x<>y, LBL 02, ...: That's just what I posted (read the second part of my original post).

BTW 1: I have no access to the KeyNotes issue you mentioned, which is the routine suggested in KeyNotes?

BTW 2: My routine does not work with negative numbers, allowing for them may provide for a tricker challenge...

Byte count in HP 41 vs. HP 42

Message #8 Posted by ***Andrés C. Rodríguez (Argentina)*** *on 11 Feb 2003, 7:13 a.m.,*
in response to message #7 by Andrés C. Rodríguez (Argentina)

My two programs (requiring Y=0 or not) take 24 and 21 bytes on the HP41, but only 23 and 19 respectively on the HP42...

Re: Byte count in HP 41 vs. HP 42

Message #9 Posted by ***Werner Huysegoms*** *on 11 Feb 2003, 7:29 a.m.,*
in response to message #8 by Andrés C. Rodríguez (Argentina)

That's because the 42S does not count the END. That would make your 42 programs 26 and 23 bytes long, two bytes more than their 41 equivalents. Makes sense, as you have two number entry lines that are one byte longer on the 42 than on the 41 (except when you have two in a row)

Re: Essentially the same . . .

Message #10 Posted by ***Paul Brogger*** *on 10 Feb 2003, 8:44 p.m.,*
in response to message #5 by Gene

Here's basically the same approach, using only "stock" stack operations:

```
LBL 00
0
X<>Y
```

```
   LBL 01
   10
   /       (divide)
   IP
   X<>Y
   LAST X
   FP
   -       (subract)
   10
   *       (multiply)
   X<>Y
   X>0?
   GTO 01


   X<>Y
   STOP
```

This (I think) is 24 bytes also.

Another approach, which is of the style I call "delegation programming" can be done in 22 or fewer bytes (though it's less reliable):

```
   LBL 00
   "ENTER REVERSED#"
   PROMPT
   INPUT ST X
   STOP
```

{;^)

(Actually, that's the first "alpha" program I've written on the HP-42s -- an experienced *calculateer* could probably do better . . .)

Re: Essentially the same . . .

Message #11 Posted by ***Werner Huysegoms*** *on 11 Feb 2003, 3:45 a.m.,*
in response to message #10 by Paul Brogger

22 with LBL 00 0 X<>Y, 19 without:

```
LBL 00
0
X<>Y
LBL 01
FP
STO+ ST Y
X<> ST L
IP
10
STO* ST Z
/
X>0?
GTO 01
RDN
END
```

Remember that on a 42S, number entry lines always include the invisible NULL char - so the program here is two bytes longer on a 42 than on a 41

Re: HP-41 / 42 style RPN challenge

Message #12 Posted by ***Ex-PPC member*** *on 11 Feb 2003, 4:34 a.m.,*
in response to message #1 by Gene

This is a 16-step, 19-byte solution for the HP-15C:

```
01 LBL 0
02 STO 0
03 STO-0
04 LBL 1
05 FRAC
06 STO+0
07 LASTX
08 INT
09  1
10  0
11 STOx0
12  /
13 TEST 0
14 GTO 1
15 RCL 0
16 RTN
```

It will reverse 3141592654 in just 10 seconds. By the way, TEST 0 is the HP-15C's instruction for x<>0 ?, so using that instruction instead, this same routine will work on most any HP RPN machine, such as the HP-11C, HP-67, even on the venerable HP-25, provided you delete both labels and change the GTO 1 to GTO 03 and the RTN to GTO 00. That would be a 14-step, 14-byte routine !!

Re: HP-41 / 42 style RPN challenge

Message #13 Posted by ***Paul Brogger*** *on 11 Feb 2003, 10:47 a.m.,*
in response to message #12 by Ex-PPC member

My hat is off to Ex-PPC! I've wasted too much time on this, and haven't done nearly as well.

For what it's worth, his approach takes 21 bytes on an HP-42s. One might argue that the storage register takes several bytes as well. (On an HP-32S(II) this would be easily measured.) But then, ultimately, so do the stack registers, so that's a minor quibble.

Besides, here's his approach as a stack-only version, taking only 22 bytes:

```
LBL 00
0
X<>Y


LBL 01
FP
STO+ ST Y
ROLL DOWN
LAST x
IP
10
STO* ST Z
/
X>0?
GTO 01


ROLL DOWN
STOP
```

A very slick approach. The key, I think, is starting right out with FP, and performing the x>0? test against the just-shifted remnant. For some reason, I got hung up on testing the integer portion only, and that forces extra (obviously unnecessary) stack manipulation.

Also, with this algorithm, he gets away with entering "10" only once -- another benefit.

As Mr. Burns would say, "Excellent!"

Re: (Generalization of) HP-41 / 42 style RPN challenge

Message #14 Posted by ***Paul Brogger*** *on 13 Feb 2003, 10:12 a.m.,*
in response to message #1 by Gene

I couldn't resist adding to my "Useless generalizations of RPN challenges" series . . .

Given any 1-9 digit integer in *x*, return *any arbitrary selection and arrangement* of 1-9 of its digits, in a pattern specified by the user.

```
A01  LBL   A       "xeq A" to run
A02  STO   U       store User-supplied value
A03  0             initialize
A04  STO   R          Result
A05  INPUT P       obtain user's Pattern
A06  9             initialize loop counter


L01  LBL   L       processing Loop
L02  STO   C       save Counter value
L03  RCL   P       get Pattern (again)
L04  XEQ   D       execute "obtain Digit" subroutine
L05  RCL   U       recall User's original value
L06  XEQ   D       execute "obtain Digit" subroutine
L07  10            shift
L08  STO*  R          Result left
L09  ROLL DOWN           and
L10  STO+  R                "append" specified digit
L11  RCL   C       get Loop counter,
L12  1                decrement
L13  -                   and
L14  x>0?                   test for done
L15  GTO   L       not done


L16  RCL   R       bring Result into x
L17  STOP             and halt


D01  LBL   D       "return yth digit in x"
D02  x<>y          swap
```

```
D03  10**x         turn location into divisor
D04  /                shift y right x places and
D05  FP                  discard stuff on the left
D06  10            shift it left
D07  *                one place, and
D08  IP                  discard stuff on the right
D09  RTN           return specified digit
```

The pattern is encoded as follows:

```
"0" == insert a zero
"1" == one's digit
"2" == ten's digit
"3" == hundred's digit
    ...
"9" =  hundred million's digit
```

For the special case of reversing an *n*-digit value, simply XEQ A with some integer in *x*, and enter the first *n* digits of "123456789" when prompted for P.

I programmed this on an HP-32s, and it takes 48 bytes as written, and 32 more when run (for the four variables).