Back

SQRT-Compo

The mission

The competitions aim was to optimize a routine calculating a 64k sqrt-tab by size. The routine with the least bytes used was supposed to win. The deadline was at **sunday, june 8, 1997**.

Rules:

- Only 68020++ integer asm-commands.
- 65536 one byte values have to be calculated (64kbyte).
- The given range is **0..65535**.
- The tables destination is a label called "sqrttab" in a bss-area.
- All registers are in an undefined state, when the routine is called.
- The routine has to be finished with "RTS", which is not accounted to the overall routine-length.
- All fraction parts have to be cut off (no rounding) (2.7 gets 2 then for example).
- No AmigaOS or whatever libs may be used.

The results

It seems, that the exercise was too easy :) Far over 50% of the 8 contributors reached a point, where no further optimizing seems possible. So we have 5 winners - but I have to point out that, except me (azure) who had to make to first contribution due to fairness reasons, **Dave** was the first one to contribute with a "best-case" routine.

Detailed results:

Place	Contributor	Length of the routine
1.	Dave	22 bytes
	Grey	22 bytes
	Morbid	22 bytes
	Skjeggspir	22 bytes
	Azure	22 bytes
2.	Kruztur	24 bytes
	Psalt and Accede	24 bytes
3.	Dig-It	26 bytes

How does it work ?

All contributed routines are using the same algorithm. Its based on the fact, that the intervalls betweem square-numbers increase by two, the higher numbers are getting. The 22 byte implementations were almost identical - they differed only a bit in register usage etc. I will use **Greys** routine as ar since it is the best documented one.

```
* ************************************************************** *
* **** Tiny square-root-table generator ****** (c) gREY in 1997 **** *
* ****** For the first official #amycoders coding competition ****** *
* *** Developed in 5 minutes while watching ST:TNG "Half a life" *** *
* ************************************************************** *

Sqrt:
        lea     sqrttab,a0      * load adress of table
        moveq   #0,d0           * clear d0 (start vlaue = 0)
.outer:
        move.w  d0,d1           * save d0
```

```
        add.w   d1,d1           * Multiply d1 with 2
                                * dbra below saves us from adding 1!
.inner:
        move.b  d0,(a0)+        * fill sqrttab
        dbra    d1,.inner       * as many times as needed

        addq.b  #1,d0           * increase d0 by 1
        bcc.s   .outer          * as long as d0 =<255! rts * ************************************************************
```

Psalt and **Accede** used a very nice trick to save some bytes while setting 3 registers to zero. They took advantage of the fact, that "fresh" BBS-area with zeros.

```
; Square root routine by Psalt & Accede
; Made for the #Amycoders sqrt competition.
; Somewhat inaccurate, like sqrt(15)=3
; But fast, and very small...
; bottom and the rts.
; Feel free to use it!
; contact us:
; Accede@hotmail.com
; Psalt@hotmail.com

initsqr lea     sqrttab,a0              ;
        movem.w (a0),d0/d2/d3           ;
.oloop  move.b  d2,(a0)+
        dbra    d0,.oloop
        addq.w  #2,d3
        move.w  d3,d0
        addq.b  #1,d2
.yo     bne.s   .oloop
        rts
```

Also **Dig-Its** routine is quite interesting: He is the only one, who made a routine writing the table backwards !

```
;;
;; dig-it @ TBL ...
;;


j:      lea     sqrttab+65536,a0
        moveq   #-1,d0
        move.w  #$1ff,d2

.loop1:
        move.w  d2,d1
.loop2:
        move.b  d0,-(a0)
        subq    #1,d1
        bne.s   .loop2
        subq    #1,d0
        subq    #2,d2
        bge.s   .loop1

end:    rts
```

Thats it - the other routines were more or less the same. Download a package with all contributions right here.

Last change: 16.01.2001